INSTITUTO TECNOLÓGICO SUPERIOR DE CALKINÍ

“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN DE INFORMACIÓN GERIÁTRICA: DESARROLLO DE LA INFRAESTRUCTURA DE DATOS Y MÓDULOS ADMINISTRATIVOS”

TESIS PARA TITULACIÓN INTEGRAL

INGENIERIA EN SISTEMAS COMPUTACIONALES

EGRESADO:
RODOLFO RUBALCABA CHUC

DIRECTOR Y CO-DIRECTOR:
DR. GONZALO MIGUEL QUETZ AGUIRRE
DR. ANGEL FRANCISCO CAN CABRERA

Calkiní, Campeche, México 2025

Educación
Secretaría de Educación Pública

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico Superior de Calkiní
Subdirección de Planeación y Desarrollo

Instituto Tecnológico Superior de Calkiní
Departamento de Administración y
Apoyo al Estudiante

Calkiní, Campeche 45727
P/03/25/728
INGENIERÍA EN SISTEMAS COMPUTACIONALES

CONSTANCIA DE NO INCONVENIENCIA

MODALIDAD: **TESIS**

C. RODOLFO RUBALCABA CHUC 7001
PASANTE DE LA CARRERA DE
INGENIERÍA EN SISTEMAS COMPUTACIONALES.
PRESENTE

Con relación a lo establecido en el Reglamento de Estudios de Licenciatura, capítulo VII, fracción I, vigente en este Instituto, y una vez emitido el fallo favorable por el director DR. GONZALO MIGUEL QUETZ AGUIRRE, y el co-director DR. ANGEL FRANCISCO CAN CABRERA, respectivamente, este Instituto consta de no tener inconveniencia para la realización del acto protocolario de titulación integral de su trabajo profesional titulado: "**DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE GESTIÓN DE INFORMACIÓN GERIÁTRICA: DESARROLLO DE LA INFRAESTRUCTURA DE DATOS Y MÓDULOS ADMINISTRATIVOS.**".

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Sabiduría, Fortaleza de Nuestra Cultura®

ITESCAM
INSTITUTO TECNOLÓGICO SUPERIOR DE CALKINÍ
"DEPARTAMENTO DE ADMINISTRACIÓN ESCOLAR Y APOYO A ESTUDIANTES"
CLAVE: 04MSU0013P
CALKINÍ, CAMPECHE, MÉXICO.

Dr. Jorge Alfonso Hau Puc.
Departamento de Administración y Apoyo al Estudiante.

C.c.p. Archivo

2025 Año de La Mujer Indígena

DEDICATORIA

Agradezco principalmente a Dios, por darme la fuerza de terminar esta meta de mi vida, de siempre estar ahí cuando se necesita, y guiarme para lograr mi objetivo.

También a mis padres, quienes hicieron el esfuerzo de ayudarme y no rendirse para cumplir con mi meta, de siempre ayudarme a lograr terminar esta etapa de mi vida, en especial a mi madre, que no se rindió hasta verme graduar.

A los docentes, por su guía y enseñanzas, que fueron ayudaron en el desarrollo de cada etapa, sin ellos no pudiese hacer este proyecto, demostrando que ellos siempre estuvieron al pendiente de mi y por los avances que había logrado.

AGRADECIMIENTO

Agradezco a dios por la ayuda y perseverancia de no quedarme, de siempre luchar con las metas propuestas hasta lograrlo.

A mis padres, quienes estuvieron en esta etapa de mi vida, logrando salir con la frente el alto, sin ellos, no pude haberlo logrado, agradezco de que siempre estuvieron ahí cuando los necesitaba.

A los docentes y mentores, quienes con su sabiduría y dedicación me han inspirado a superar cada día y a seguir explorando el conocimiento con pasión y compromiso. De siempre guiarme con su conocimiento para terminar esta parte de mi vida y lograr mis metas.

A mis amigos y compañeros de estudio, con quienes compartimos momentos de aprendizaje, esfuerzo y crecimiento, convirtiendo este proceso en una experiencia enriquecedora y significativa.

Índice

Índice Ilustraciones.

Índice Tablas.

RESUMEN.

El presente proyecto pretende resolver la problemática de un centro médico, donde el manejo de los registros que se guardan, son en mayoría en papel, y que estos no tienen un orden para su almacenamiento. Esta aplicación ayuda a resolver el problema de organización dentro del centro médico, está diseñada para facilitar la captura de registro y diagnóstico del paciente, permitiendo al personal médico organizar y gestionar información de manera rápida y precisa.

Se hizo un análisis de los requisitos específicos de los especialistas en geriatría para comprender mejor sus necesidades en el manejo de información y diagnóstico de pacientes mayores, con ello la aplicación cuenta con una base de datos diseñada para almacenar información detallada de los pacientes, incluidos sus datos personales, historial médico, resultados de pruebas y observaciones clínicas. Esta aplicación (Pagina Web) integra algoritmos que analizan los resultados de evaluaciones y pruebas clínicas, detectando patrones y alertando sobre posibles problemas de salud. Estos algoritmos están diseñados para apoyar a los médicos en la identificación temprana de problemas y en el proceso de toma de decisiones, contribuyendo a un diagnóstico rápido y preciso.

El programa tiene dos roles de usuarios, primero es el personal médico que se encarga de hacer las evaluaciones de los pacientes, y el administrador, que se encarga de dar de alta a los doctores nuevos y eliminarlos.

Esta aplicación facilita la captura de datos, el registro de pacientes y el apoyo en el diagnóstico mediante un diseño intuitivo, facilitando al personal medico que no este familiarizado con la tecnología, pueda manejar esta aplicación sin ningún problema, cabe resaltar que solo hace captura de los registros de las encuestas que se le aplique al paciente.

Capítulo 1

1.1 INTRODUCCIÓN.

En los centros geriátricos, la administración de los datos de los pacientes y los recursos disponibles es un proceso manual y desorganizado. Esto conduce a demoras en la atención médica, duplicidad de tareas y falta de información precisa para la toma de decisiones clínicas. La digitalización de la información médica y administrativa en los centros geriátricos es esencial para garantizar una atención adecuada, eficiente y personalizada a una población en constante crecimiento y con necesidades complejas.

La atención a personas mayores requiere una gestión de información precisa, accesible y centralizada. La implementación de un sistema de gestión geriátrica es esencial para optimizar tanto los procesos administrativos como el seguimiento médico de cada paciente. La centralización de la información permite un acceso rápido y seguro a datos clave como historiales médicos, tratamientos, diagnósticos y evaluaciones de cada paciente, lo que facilita la toma de decisiones y minimiza los errores. Este tipo de sistema no solo beneficia al personal médico, al reducir el tiempo invertido en tareas administrativas, sino que también mejora la calidad de vida de los pacientes al garantizar una atención constante y personalizada.

El proyecto busca diseñar e implementar un sistema de gestión de información específicamente para centros geriátricos, donde la organización y el acceso rápido a datos médicos y administrativos son fundamentales para brindar atención de calidad a personas mayores. Actualmente, la gestión manual de datos y la falta de centralización generan retrasos y posibles errores en la atención. Este sistema permitirá centralizar historiales médicos, gestionar recursos, y optimizar el seguimiento de tratamientos, mejorando así la eficiencia del personal y la seguridad del paciente. Además, el proyecto cumple con normativas de seguridad de datos y se enfoca en facilitar la toma de decisiones clínicas informadas.

Además, los procesos manuales actuales, comunes en muchos centros, son propensos a errores y pueden llevar a la duplicación de información y pérdida de datos importantes, afectando la eficiencia y la seguridad del cuidado proporcionado. Un sistema de gestión geriátrica bien diseñado permite automatizar tareas administrativas, como el manejo de citas, el control de inventarios de medicamentos y la asignación de recursos, optimizando así el uso del tiempo y los recursos del centro.

El alcance del proyecto tiene como objetivo principal desarrollar una infraestructura de datos que centralice y organice toda la información geriátrica relevante, respetando las normativas de seguridad y privacidad de los datos médicos. Al mejorar el flujo de trabajo y asegurar la disponibilidad de datos en tiempo real, el sistema propuesto busca contribuir a una atención más eficiente, segura y personalizada para la población geriátrica.

1.1.1 PROPUESTA

Para este proyecto tuvo en cuenta que, la captura de los resultados de las encuestas es tardado, para múltiples personas, y para solucionarlo se implementaría una página donde haga la captura de los pacientes de tercera edad, evitando la demora de tiempo y facilitando el registro de cada paciente, ayudando así a automatizar la captura de los registros, haciéndolo fácil y rápido.

1.2 ANTECEDENTES.

La geriatría es la rama de la medicina que se enfoca en el cuidado de personas mayores, abordando condiciones específicas como enfermedades crónicas, deterioro cognitivo y problemas de movilidad (Cassel et al., 2012). En un contexto geriátrico, la gestión de la información de los pacientes requiere un sistema que no solo centralice datos clínicos, sino que también permita registrar detalles relevantes para un cuidado integral y personalizado. Esto incluye el historial de enfermedades, medicamentos, diagnósticos previos, y las interacciones con diversos especialistas.

La gestión de información geriátrica necesita ser precisa y completa debido a la vulnerabilidad de esta población, que presenta mayores riesgos de reacciones adversas a medicamentos, hospitalizaciones frecuentes y necesidades de atención personalizada. Un sistema de gestión geriátrica ayuda a reducir el riesgo de errores médicos y a coordinar mejor los cuidados, ya que todos los datos del paciente se encuentran centralizados y accesibles para el personal de salud.

Historia Clínica Electrónica (HCE)

La historia clínica electrónica es un componente fundamental de los sistemas de información en salud, que permite el registro detallado de los antecedentes médicos de un paciente (Shahmoradi et al., 2017). Una HCE digitalizada en un sistema geriátrico incluye información como el historial de diagnósticos, procedimientos, medicamentos, y notas de los profesionales médicos, lo cual es esencial para dar seguimiento a los pacientes geriátricos. La implementación de una HCE en centros geriátricos mejora la comunicación entre los profesionales de la salud y facilita la continuidad de los cuidados.

La HCE también permite analizar los datos para identificar patrones y tendencias en el estado de salud de los pacientes. Esto es particularmente útil en geriatría, donde el seguimiento longitudinal de los pacientes permite tomar decisiones más informadas y personalizadas.

Sistemas de Información en Salud (HIS) Los sistemas de información en salud (HIS) están diseñados para centralizar datos clínicos y administrativos, mejorando la eficiencia

en hospitales y centros médicos. Según Rindfleisch (1997), estos sistemas ayudan a reducir errores, mejorar el flujo de trabajo y optimizar el uso de recursos. En el contexto geriátrico, estos beneficios son especialmente relevantes debido a la naturaleza vulnerable de la población, que necesita atención especializada y personalizada. Los HIS permiten, además, la integración con otros sistemas de salud y aseguran que la información del paciente esté disponible para todos los profesionales involucrados en su cuidado.

Interoperabilidad y Seguridad en Sistemas de Salud La interoperabilidad es una característica fundamental de los sistemas de información en salud modernos, permitiendo que diferentes sistemas compartan datos de manera fluida y segura. Según Rowe, Fulmer, y Fried (2006), la interoperabilidad es crucial para asegurar una atención continua, especialmente en el caso de pacientes geriátricos que pueden requerir tratamiento en distintos centros. Además, la seguridad y privacidad de la información en sistemas de salud es un desafío debido a las estrictas regulaciones de protección de datos, como HIPAA en los Estados Unidos. Los sistemas de información geriátrica deben incluir medidas de encriptación y control de acceso para proteger los datos de los pacientes (Rindfleisch, 1997).

Los antecedentes de la geriatría en el campo médico reflejan la evolución de esta disciplina como una especialidad centrada en el cuidado integral de las personas mayores, un grupo de pacientes con necesidades específicas debido a las enfermedades crónicas, los cambios fisiológicos y la vulnerabilidad asociada con el envejecimiento. La geriatría comenzó a desarrollarse formalmente en el siglo XX, cuando se reconoció la necesidad de una atención especializada para la población mayor. A diferencia de otras especialidades, este enfoque reconocía la importancia de la valoración integral, dado que los ancianos presentan patrones de salud distintos, con múltiples condiciones crónicas que requieren un abordaje multidisciplinario (Cassel et al., 2012; Stuck et al., 1993).

Desde sus orígenes, la geriatría se ha caracterizado por un enfoque integral y multidisciplinario que va más allá del diagnóstico y tratamiento de enfermedades, considerando también el bienestar emocional y social de las personas mayores. Este abordaje busca reducir la fragilidad y fomentar la autonomía del adulto mayor mediante la colaboración de geriatras con otros profesionales, como enfermeros, terapeutas ocupacionales y trabajadores sociales (Katz & Karuza, 2005; Rubenstein & Josephson, 2002).

Con el tiempo, la geriatría ha evolucionado hacia un enfoque de manejo de enfermedades crónicas y atención centrada en el paciente, priorizando la calidad de vida sobre la cura absoluta de enfermedades. Esto incluye el uso de herramientas de valoración geriátrica integral para evaluar aspectos físicos, funcionales, mentales y sociales del paciente, lo que permite un cuidado más holístico y personalizado (Stuck et al., 1993). Además, el aumento de la esperanza de vida ha generado una mayor prevalencia de enfermedades crónicas y discapacidades asociadas con la edad, lo que convierte a la geriatría en una especialidad clave para enfrentar los desafíos derivados del envejecimiento poblacional (Rowe & Kahn, 1997).

La introducción de tecnologías como la historia clínica electrónica (HCE) y los sistemas de información en salud ha facilitado el seguimiento continuo de la salud de los adultos mayores y mejorado la coordinación entre los profesionales de salud. Los HCE permiten almacenar información clave sobre diagnósticos y tratamientos, asegurando una atención continua y segura, especialmente en casos de pacientes con múltiples condiciones (Cassel et al., 2012; Blumenthal & Tavenner, 2010). Estas innovaciones se consideran fundamentales en la geriatría, ya que ayudan a mejorar la precisión del diagnóstico y reducen errores, lo que resulta especialmente relevante en el cuidado de personas mayores que necesitan atención especializada y constante.

1.3 JUSTIFICACIÓN

Este sistema permitirá mejorar la calidad de la atención médica, optimizando el uso de los recursos y facilitando la toma de decisiones clínicas a través de un acceso rápido y

centralizado a la información de los pacientes. Además, contribuirá a la gestión eficiente al centro geriátrico, ayudando a mitigar el impacto de un aumento en la demanda de servicios geriátricos. Con esto se espera automatizar y mejorar el rendimiento de la captura de datos de cada paciente, guardando estos datos de una manera que no pueda ser tratados por terceros, con el fin de proteger la información de cada paciente, al igual del doctor al redactar información confidencial.

La realización de este proyecto es desarrollar una herramienta que pueda facilitar la recopilación de los datos realizados por los Tests que estos nos proporcionaron, automatizando la captura de los datos con solo seleccionar los síntomas que padece cada paciente y dar un informe de cuales son el posible riesgo que este puede padecer, así buscar una manera de prevenir dicho riesgo. La implementación de este módulo contribuirá a mejorar la calidad de vida de los pacientes geriátricos al proporcionar una atención médica más precisa y oportuna.

El sistema facilita la captura y gestión de información, lo que reduce significativamente el tiempo que el personal médico dedica a tareas administrativas, como el registro manual de datos. La interfaz intuitiva y el flujo de trabajo optimizado permiten a los profesionales de la salud acceder rápidamente a la información crítica del paciente, lo que mejora la eficiencia en la toma de decisiones y en la atención clínica. El modelado adecuado de la base de datos garantiza que toda la información relevante del paciente, como historial médico, resultados de pruebas y observaciones clínicas, esté centralizada y fácilmente accesible. Esto facilita el acceso rápido a la información actualizada, lo que es especialmente importante en centros geriátricos donde los pacientes suelen requerir atención continua y seguimiento a largo plazo.

La automatización de procesos, como la captura de datos e implementación de algoritmos de diagnóstico, reduce el riesgo de errores humanos en el registro y análisis de la información. Esto es especialmente importante en geriatría, donde la atención debe ser precisa y continua. Los errores en la gestión de datos pueden tener consecuencias graves, y el sistema contribuye a mitigar este riesgo. Al contar con un

sistema centralizado que permite a todos los profesionales de la salud involucrados en el cuidado del paciente acceder a la misma información en tiempo real, se mejora la coordinación y comunicación entre médicos, enfermeras, y otros especialistas. Esto facilita el trabajo en equipo y asegura que el tratamiento y seguimiento del paciente se realicen de manera coherente y fluida.

1.4 ALCANCES Y LIMITACIONES.

Alcance del Proyecto.

El proyecto de diseño e implementación de un sistema de gestión de información geriátrica abarca la creación de una plataforma digital que centralice y administre la información clínica y administrativa de pacientes geriátricos en centros de atención. Los principales componentes y funciones de este sistema incluirán:

1. Gestión de la Información del Paciente.
2. Módulos Administrativos.
3. Interfaz de Usuario Intuitiva.
4. Seguridad y Privacidad de Datos.
5. Soporte para la Toma de Decisiones Médicas.

Limitaciones.

1. *Alcance Limitado a Centros Geriátricos Específicos.*
2. *Limitación de Recursos Financieros y Técnicos.*
3. *Dependencia de Capacitación del Personal.*
4. *Adaptación y Personalización del Sistema.*
5. *Interoperabilidad con Otros Sistemas*

1.5 OBJETIVOS GENERALES Y ESPECIFICOS.

1) Objetivo general.

Diseñar e implementar un sistema de gestión de información geriátrica que centralice los datos de los pacientes y facilite la administración de los procesos en los centros geriátricos, mejorando la atención sanitaria y la gestión de los recursos.

2) Objetivos específicos

- *Desarrollar la infraestructura de datos para almacenar información de los pacientes geriátricos.*
- *Crear módulos administrativos que optimicen la gestión de recursos en el centro geriátrico.*
- *Asegurar la interoperabilidad del sistema con otros sistemas de información médica o gubernamentales.*
- *Incluir funcionalidades avanzadas para el análisis de datos que apoyen la toma de decisiones clínicas.*
- *Implementar medidas de seguridad para proteger la privacidad y confidencialidad de los datos de los pacientes.*
- *Implementar un módulo de historial de tratamientos y evolución del paciente*
- *Incorporar herramientas de análisis y visualización de datos para el personal administrativo y médico*
- *Diseñar reportes automatizados sobre el uso de recursos y la ocupación del centro geriátrico*

1.6 INFORMACIÓN DE LA EMPRESA

1.6.1 Lugar donde se realiza el Proyecto

Nombre del Departamento

División de Ingeniería en Sistemas Computacionales perteneciente a la Subdirección Académica de Investigación e Innovación.
Edificio A - Planta Alta
996-8134870 Ext. 1001

Estructura Departamental

Subdirección Académica de Investigación e Innovación

Croquis del Departamento

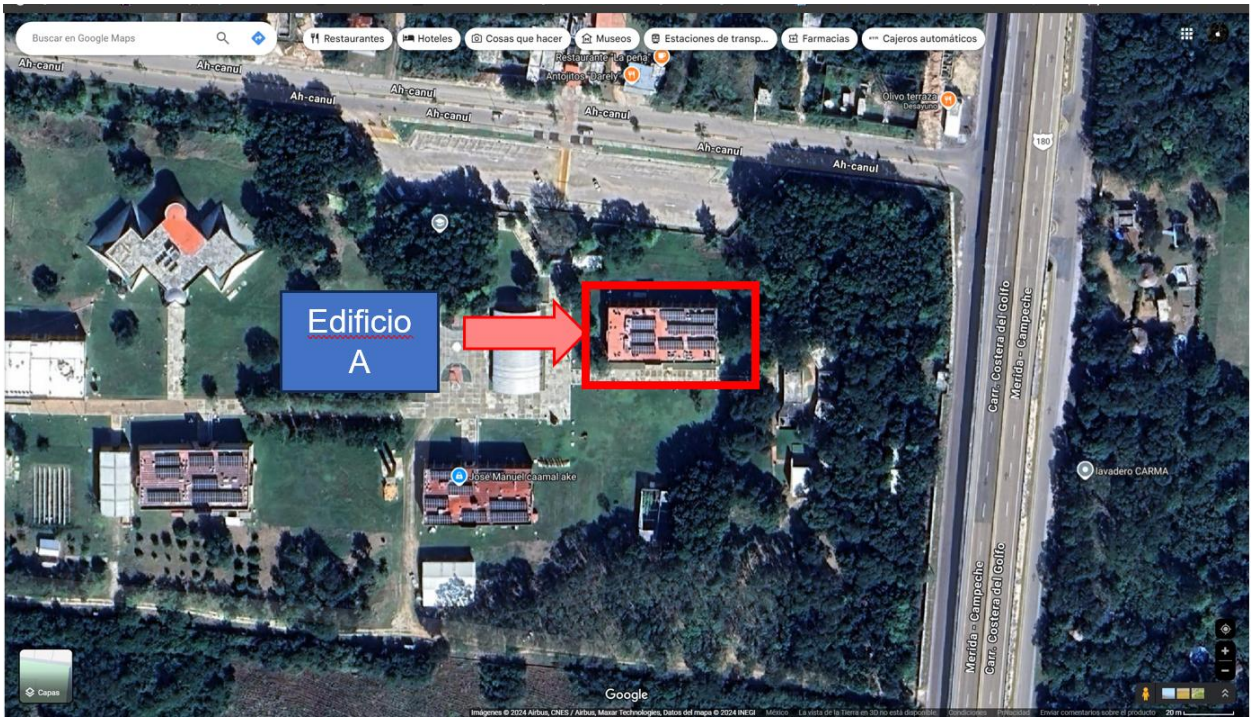


Ilustración 1 Mapa de Trabajo

INFORMACIÓN SOBRE LA EMPRESA, ORGANISMO O DEPENDENCIA PARA LA QUE SE DESARROLLARÁ EL PROYECTO.

LUGAR DE REALIZACIÓN:

NOMBRE DE LA EMPRESA Y DIRECCIÓN

Instituto Tecnológico Superior de Calkiní, en el Estado de Campeche
Av. Ah Canul S/N por Carretera Federal.
C.P. 24920 Calkiní, Campeche.
Tel. 996-813-48-70.

GIRO

Educación con un amplio sentido social y humano, al desarrollo sustentable de la región, del Estado y del país, atendiendo desde su ámbito de competencia, las necesidades de formación y actualización de profesionales competitivos, de investigación y desarrollo tecnológico y de conservación y extensión de la cultura.

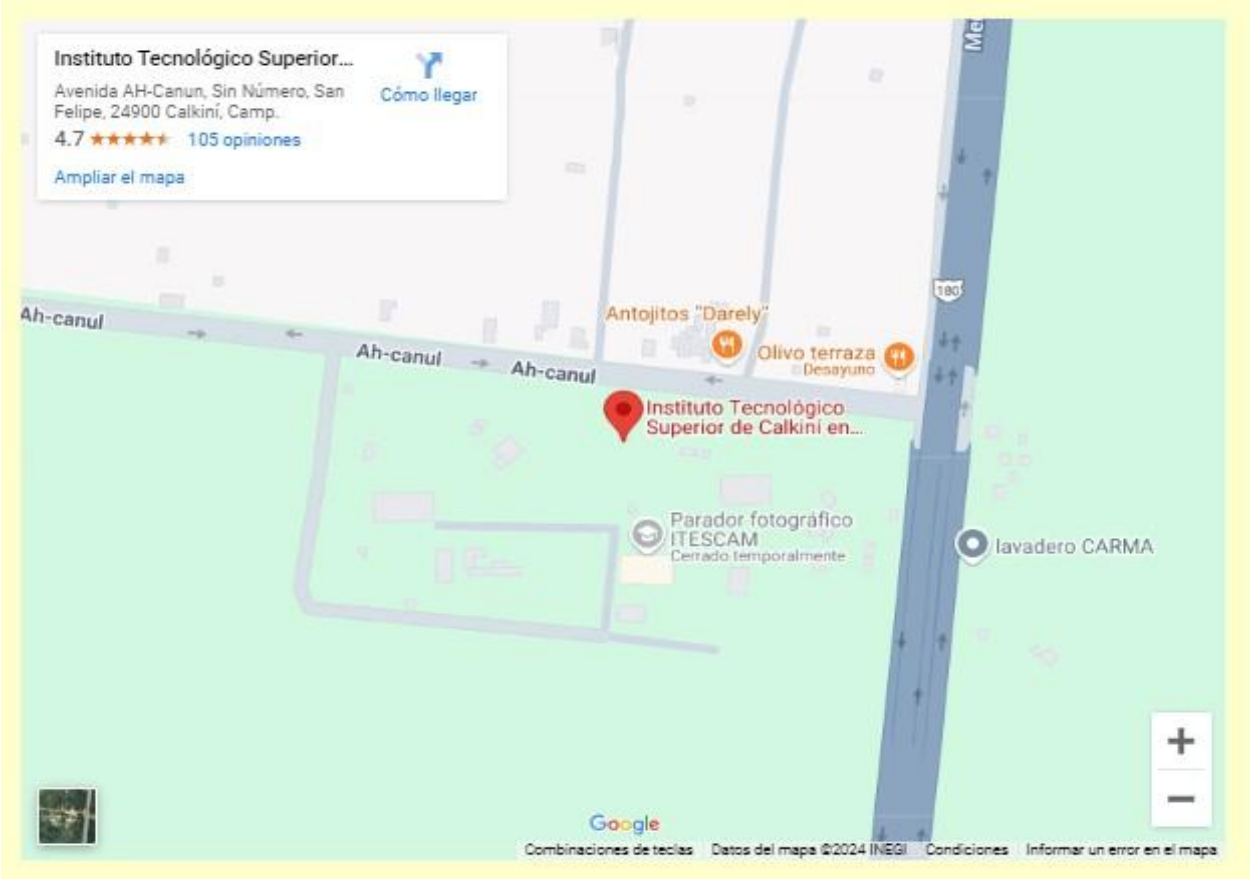


Ilustración 2 Mapa Escuela de Trabajo

RFC de la empresa

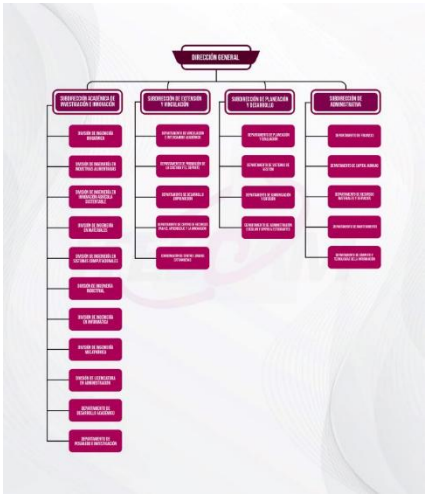


Ilustración 3 RFC Empresa

1.6.2 Misión

Brindar a los profesionales de la salud una herramienta tecnológica innovadora que facilite la evaluación integral de pacientes geriátricos mediante la implementación de Tests estandarizados y la visualización eficiente de resultados, promoviendo una atención médica de calidad, personalizada y accesible para mejorar la calidad de vida de las personas mayores.

1.6.3 Visión

Convertirse en un referente nacional en el desarrollo de soluciones tecnológicas para la gestión clínica geriátrica, consolidando un sistema integral que automatice y optimice los procesos de evaluación médica, contribuyendo al fortalecimiento de los servicios de salud y al bienestar de la población envejecida.

1.6.4 Organigrama

Ilustración 4 Organigrama de los representantes

Capítulo 2

2.1 Marco Teórico

La geriatría, como campo de estudio, se ha desarrollado en el contexto de la comprensión y aplicación de los principios geométricos en diversos entornos científicos y tecnológicos. Según estudios de Johnson y Peters (2018), se centra en la intersección entre la geometría aplicada y la adaptabilidad estructural en sistemas avanzados. Su evolución ha estado marcada por la integración de herramientas computacionales que permiten un análisis más preciso y detallado de estructuras y patrones geométricos.

Las aplicaciones tecnológicas de la geriatría han sido fundamentales en el desarrollo de herramientas de diseño digital. Según investigaciones de Martínez y Rodríguez (2021), los modelos geriátricos han sido clave en la creación de entornos virtuales, modelado paramétrico y la optimización de materiales en la industria aeroespacial. En la arquitectura y la ingeniería civil, el uso de software basado en principios geriátricos permite la creación de estructuras más eficientes y resistentes, reduciendo costos y tiempos de construcción.

Con la llegada de la inteligencia artificial (IA), ha cobrado un papel relevante en el aprendizaje automático aplicado a diseños complejos. Estudios recientes de Lee y Chen (2022) destacan que los algoritmos basados en principios geriátricos permiten mejorar la generación de estructuras adaptativas en tiempo real, lo que facilita la automatización en la manufactura avanzada. Además, la geriatría ha sido aplicada en la robótica, permitiendo el diseño de mecanismos con mayor precisión y eficiencia en la movilidad y manipulación de objetos.

El desarrollo del software ha encontrado recursos clave para la optimización de algoritmos de representación visual y modelado digital. Según estudios de Fernández y López (2024), los sistemas de software utilizados en el diseño asistido por computadora (CAD) se basan en principios geriátricos para generar modelos tridimensionales precisos. Además, la geriatría es aplicada en el desarrollo de motores gráficos y

simulaciones en tiempo real, permitiendo una representación fiel de objetos y entornos virtuales.

Otro ámbito donde este tiene gran influencia es en el desarrollo de software para análisis estructural y de materiales. De acuerdo con Ramírez y Torres (2023), los algoritmos permiten evaluar el comportamiento de materiales bajo distintas condiciones, optimizando la resistencia y eficiencia de las estructuras diseñadas en plataformas de software especializadas.

2.2 Metodología SCRUM

Scrum es una metodología ágil que permite gestionar proyectos complejos mediante la entrega iterativa e incremental de valor. Fue formalizada por Ken Schwaber y Jeff Sutherland en la década de 1990, quienes publicaron la "Guía Scrum", el documento que establece las bases para su implementación (Schwaber & Sutherland, 2020). Este marco se basa en principios de transparencia, inspección y adaptación, esenciales para asegurar el éxito en entornos de trabajo cambiantes.

Scrum es un enfoque estructurado que promueve la colaboración entre equipos multifuncionales. Su objetivo principal es maximizar el valor entregado al cliente mediante la creación de incrementos funcionales del producto al final de cada ciclo de trabajo, conocido como sprint. Según Schwaber y Sutherland (2020), este marco se utiliza en contextos donde los requisitos no están completamente definidos desde el inicio, permitiendo iteraciones rápidas y adaptaciones continuas.

Según Schwaber y Sutherland (2020), creadores de Scrum, esta metodología ágil es un marco de trabajo que permite a equipos resolver problemas complejos mientras entregan productos de valor máximo. Su enfoque se basa en la transparencia, inspección y adaptación, aplicando ciclos iterativos llamados "sprints". Este marco busca maximizar la productividad y fomentar el trabajo colaborativo en entornos dinámicos.

Highsmith (2009), Scrum es una de las metodologías ágiles más influyentes, diseñada para adaptarse a cambios rápidos en los requisitos del cliente y en los entornos de

desarrollo. Esto se logra mediante la implementación de pequeños equipos multifuncionales que trabajan en ciclos cortos para entregar incrementos del producto. Rubin (2012) describe Scrum como un marco iterativo e incremental que se centra en la entrega continua de valor al cliente. Los equipos Scrum son autogestionados y colaborativos, organizándose en torno a roles clave: el Product Owner, el Scrum Master y el Development Team.

Scrum es una de las variantes más populares de metodologías ágiles y se utiliza comúnmente en proyectos de desarrollo de software. Scrum se basa en un enfoque iterativo e incremental, donde el trabajo se divide en sprints cortos y cada sprint ofrece un producto o característica funcional. Según Molina et al. (2018) la metodología SCRUM es un marco de trabajo diseñado para lograr la colaboración eficaz del equipo de trabajo, emplea un conjunto de reglas y se definen roles para generar una estructura de correcto funcionamiento.

Scrum se basa en la teoría de control de procesos empírica o empirismo; existen tres pilares fundamentales que soportan el control del proceso empírico: transparencia, inspección y adaptación (López, 2015). Aunque SCRUM nació centrado en el desarrollo de software, su aplicación puede trasladarse casi a cualquier contexto (Jordán, 2020). de esta forma se convierte en una herramienta que debes dominar.

2.2.1 Principio De Metodología Scrum

La metodología Scrum se fundamenta en un conjunto de principios clave que estructuran el trabajo colaborativo y adaptativo en equipos. Según Schwaber y Sutherland (2020), Scrum se basa en tres pilares: transparencia, inspección y adaptación, que promueven la entrega incremental de valor en entornos complejos. Estos principios son esenciales para gestionar proyectos en condiciones de incertidumbre y cambio constante.

La metodología Scrum se ha establecido como un marco ágil de trabajo para gestionar proyectos, especialmente en entornos donde los requisitos son cambiantes y los proyectos son complejos. Según Pichler (2010), Scrum promueve un enfoque colaborativo entre los miembros del equipo y las partes interesadas, asegurando que

cada parte del proyecto se lleve a cabo de manera eficiente y con un fuerte enfoque en la entrega de valor.

2.2.2 Pilares De Scrum

Los tres pilares de Scrum dan forma a los principios ágiles subyacentes de la metodología Scrum, fomentando la eficiencia y la adaptabilidad en la gestión de proyectos. Scrum, conocido por su marco de proceso empírico, gira en torno a tres pilares fundamentales: transparencia, inspección y adaptación.

Transparencia. La transparencia es esencial para que todos los miembros del equipo y las partes interesadas tengan acceso a la misma información sobre el estado del proyecto. Según Ken Schwaber y Jeff Sutherland (2020) en su Scrum Guide, la transparencia asegura que las decisiones se basen en información objetiva y visible para todos, lo que facilita la colaboración.

La transparencia se manifiesta en varias facetas de Scrum, entre ellas:

1. Backlog del sprint : esta lista dinámica abarca las tareas comprometidas dentro de un sprint, lo que fomenta la claridad en el enfoque del equipo.
2. Product Backlog : un catálogo priorizado de características y requisitos, que alinea al equipo con los objetivos del proyecto.
3. Revisión de Sprint: una plataforma para mostrar el trabajo realizado, permitiendo a las partes interesadas evaluar y brindar comentarios valiosos.
4. Definición de Terminado (DoD): Un conjunto de criterios muy claros que definen la finalización de una tarea, eliminando la ambigüedad.

Inspección. La inspección implica la revisión continua del trabajo para identificar cualquier desviación o problema que pueda surgir. Según Mike Cohn (2010) en su obra Succeeding with Agile, la inspección frecuente es vital para detectar problemas a tiempo, evitando que afecten significativamente el progreso del proyecto.

Adaptación. La adaptación es la capacidad de realizar cambios en el proceso cuando se detectan problemas o cuando el proyecto no avanza como se esperaba. Ken Rubin

(2012), en su libro Essential Scrum, explica que la adaptación continua es crucial en Scrum para mantener el proyecto alineado con las expectativas y necesidades del cliente.

2.2.3 Metodología Scrum En Software

En Schwaber y Jeff Sutherland (2020) Ambos son los cocreadores de Scrum y han sido fundamentales en su evolución y popularización. En su obra The Scrum Guide, explican cómo Scrum se estructura en torno a la entrega incremental de productos, la colaboración entre equipos multifuncionales y la gestión flexible de los requisitos del software. Según Schwaber y Sutherland, Scrum permite una respuesta rápida a los cambios y fomenta un proceso iterativo que es clave en entornos de desarrollo de software complejos y cambiantes.

Mike Cohn (2010) Cohn es uno de los principales expertos en metodologías ágiles y Scrum. En su libro Succeeding with Agile, describe cómo Scrum se adapta particularmente bien a equipos de software, dado que su enfoque en la entrega temprana y continua permite a los desarrolladores y clientes ver el progreso del producto a medida que se construye. Cohn también destaca la importancia de las retrospectivas y la mejora continua en Scrum, algo que es crucial en el desarrollo de software, donde los requisitos y las tecnologías evolucionan rápidamente.

Ken Rubin (2012) Rubin, en su libro Essential Scrum, ofrece un enfoque práctico sobre cómo implementar Scrum específicamente en proyectos de software. Subraya la importancia de los roles dentro del equipo, como el Product Owner, Scrum Master y los desarrolladores, y cómo cada uno contribuye al éxito de un proyecto de software. Según Rubin, Scrum mejora la calidad del software a través de su enfoque en la comunicación continua y la priorización de las funcionalidades más valiosas.

2.3 SOFTWARE

2.3.1 Definición

El software es un conjunto de programas, procedimientos y rutinas asociadas que permiten realizar tareas específicas en una computadora. Este puede ser clasificado en

diferentes tipos, como software de sistema (que incluye sistemas operativos y utilidades) y software de aplicación (diseñado para ayudar en tareas específicas del usuario, como procesadores de texto y aplicaciones de gestión empresarial) (Pressman, 2014). El software es esencial para el funcionamiento de las tecnologías modernas, proporcionando la base para la interacción con el hardware y ejecutando tareas que facilitan el trabajo humano en diversas áreas, desde la administración hasta el entretenimiento.

Según Sommerville (2011), el software también puede ser considerado un "producto intangible" que no solo involucra la creación de código, sino también el proceso de diseño, prueba, mantenimiento y actualización para garantizar su funcionamiento eficiente y adecuado a las necesidades de los usuarios. Este concepto resalta la importancia de las metodologías ágiles y los procesos de desarrollo para asegurar que el software evolucione de acuerdo con los cambios tecnológicos y las expectativas del mercado.

2.3.2 Características

Intangibilidad: A diferencia del hardware, el software no tiene una forma física y no puede tocarse ni observarse directamente. Solo puede ser experimentado a través de su ejecución y funcionalidad (Sommerville, 2011).

Flexibilidad: El software se adapta a diversos entornos y necesidades. Su diseño permite modificaciones, actualizaciones y mejoras, lo que lo hace muy versátil y adecuado para proyectos que evolucionan rápidamente (Pressman, 2014).

Escalabilidad: Muchos sistemas de software pueden crecer con el tiempo, es decir, pueden ampliarse o modificarse para satisfacer las crecientes demandas de los usuarios o para adaptarse a nuevas tecnologías (Sommerville, 2011).

Interacción con el usuario: El software está diseñado para interactuar con los usuarios mediante interfaces gráficas o comandos. Esta interacción facilita la experiencia del usuario y permite realizar tareas específicas de forma eficiente (Sommerville, 2011).

Dependencia del hardware: Aunque el software es independiente, su funcionamiento está condicionado por las capacidades del hardware sobre el que se ejecuta (Pressman, 2014).

2.3.3 Tipos de Software

Software de Sistema: Este tipo de software es fundamental para el funcionamiento del hardware y la ejecución de otros programas. Se encarga de gestionar los recursos del sistema y proporciona servicios esenciales para que el software de aplicación funcione correctamente. Ejemplos de software de sistema incluyen los sistemas operativos (como Windows, macOS o Linux) y las utilidades (como antivirus o herramientas de compresión de archivos) (Sommerville, 2011).

Software de Aplicación: Este software está diseñado para realizar tareas específicas o resolver problemas en áreas concretas. Los programas de software de aplicación permiten a los usuarios interactuar con el sistema de manera productiva. Ejemplos incluyen procesadores de texto (Microsoft Word), hojas de cálculo (Excel), software de diseño gráfico (Adobe Photoshop) y programas de gestión empresarial (SAP) (Pressman, 2014).

Software de Desarrollo: Incluye herramientas como los lenguajes de programación, compiladores, editores de código y entornos de desarrollo integrados (IDE). Este tipo de software es utilizado para crear otros programas (Sommerville, 2011).

Software de Programación: Son herramientas que permiten a los desarrolladores escribir, editar y depurar otros programas. Estos incluyen lenguajes de programación como Python, Java, C++, entre otros (Pressman, 2014).

Tipo de Software	Descripción	Ejemplos
Software de Sistema	Controla y gestiona el hardware de la computadora, y proporciona servicios básicos.	Windows, macOS, Linux, BIOS, Drivers
Software de Aplicación	Permite al usuario realizar tareas específicas.	Microsoft Office, Adobe Photoshop, Google Chrome
Software de Desarrollo	Herramientas que ayudan a los programadores a crear, depurar y mantener otros software.	Visual Studio, Eclipse, PyCharm, Git
Software de Base de Datos	Gestiona y organiza datos, permitiendo su almacenamiento y recuperación.	MySQL, Oracle, Microsoft SQL Server, PostgreSQL
Software de Red	Permite la comunicación entre computadoras y gestiona el tráfico de datos.	Wireshark, Cisco IOS, NGINX, OpenVPN
Software de Seguridad	Protege las computadoras y redes de accesos no autorizados o daños.	Antivirus, Firewall, Cryptography tools
Software Multimedia	Permite la creación, edición y reproducción de contenido multimedia.	VLC Media Player, Adobe Premiere, Audacity, iTunes
Software Empresarial	Herramientas diseñadas para gestionar los procesos dentro de una organización.	SAP, Salesforce, Microsoft Dynamics, Oracle ERP
Software de Juegos	Diseñado para entretenimiento interactivo en plataformas como PC, consolas o móviles.	Fortnite, The Witcher 3, Minecraft, Angry Birds
Software Educativo	Utilizado para fines pedagógicos, ayudando en el proceso de enseñanza-aprendizaje.	Duolingo, Khan Academy, GeoGebra, Moodle
Software Científico	Usado para resolver problemas en campos como la ciencia, ingeniería y matemáticas.	MATLAB, Mathematica, AutoCAD, LabVIEW

Tabla 1 Tipos de Software que existen

2.4 Lenguaje de Programación para Nagevador Web

Un lenguaje de programación es un sistema formal que permite a los programadores escribir instrucciones que una computadora puede entender y ejecutar. A través de estos lenguajes, se facilita la creación de aplicaciones, software, sistemas operativos y otros

programas informáticos. Su propósito es ofrecer una interfaz comprensible entre los humanos y las máquinas, traduciendo tareas lógicas y algorítmicas en un formato que la computadora pueda procesar.

2.4.1 Tipos de Lenguajes de Programación

JavaScript: Es el lenguaje de programación más utilizado para el desarrollo en navegadores. Permite la creación de dinámicas interactivas, como animaciones, validación de formularios y manipulación de contenidos en la página sin necesidad de recargarla (client-side scripting). JavaScript también se puede usar en conjunto con tecnologías como HTML y CSS para crear páginas web interactivas (Flanagan, 2011).

HTML (HyperText Markup Language): Aunque no es un lenguaje de programación en el sentido estricto, HTML es el estándar utilizado para estructurar y presentar contenido en la web. Se utiliza para definir los elementos que aparecerán en el navegador, como títulos, párrafos, enlaces y formularios (Duckett, 2011).

CSS (Cascading Style Sheets): Similar a HTML, CSS no es un lenguaje de programación, pero es esencial para la presentación visual de una página web. CSS permite modificar los estilos de los elementos HTML, controlando su disposición, colores, fuentes, etc. (Meyer, 2007).

WebAssembly: Es un nuevo lenguaje binario diseñado para ejecutar código a una velocidad cercana al nativo en los navegadores. Permite ejecutar lenguajes como C, C++ y Rust en el navegador de manera eficiente, ampliando las posibilidades de desarrollo de aplicaciones web de alto rendimiento (Zeldman, 2018).

2.4.2 Características

Sintaxis: Cada lenguaje tiene una sintaxis específica que define cómo se deben escribir las instrucciones. Esto incluye palabras clave, operadores y estructuras de control como bucles y condicionales (Sommerville, 2011).

Semántica: La semántica se refiere al significado de las instrucciones en el lenguaje, es decir, lo que ocurre cuando se ejecuta el código.

Paradigmas de programación: Los lenguajes pueden seguir diferentes paradigmas de programación, como el imperativo, el orientado a objetos, el funcional y el lógico (Pressman, 2014). Cada uno de estos enfoques influye en cómo los programadores estructuran y resuelven problemas.

Paradigmas de Programación: Los lenguajes de programación pueden seguir diferentes paradigmas, como el imperativo, donde el programador define explícitamente el control del flujo del programa, el orientado a objetos, que organiza el código en objetos que representan entidades del mundo real, o el funcional, donde las funciones se utilizan como bloques de construcción principales del programa. Cada paradigma influye en la forma en que los desarrolladores abordan la solución de problemas (Sommerville, 2011; Flanagan, 2011).

Lenguajes de Bajo y Alto Nivel: Los lenguajes pueden clasificarse en dos grandes categorías: lenguajes de bajo nivel, como ensamblador o C, que están más cerca del hardware y ofrecen un control preciso de los recursos del sistema; y lenguajes de alto nivel, como Python o Java, que están diseñados para ser más fáciles de usar, con sintaxis más cercana al lenguaje humano (Pressman, 2014).

2.4.3 Aplicaciones

Aplicaciones web: Usando lenguajes como HTML, CSS, JavaScript, y PHP.

Desarrollo de software: Lenguajes como C++, Java, y C# se usan para crear aplicaciones de escritorio y sistemas operativos.

2.3.4.1 Tabla

Lenguaje de Programación	Propósito	Descripción
HTML (HyperText Markup Language)	Estructura de contenido	Define la estructura básica de la página web. Utiliza etiquetas para organizar texto, imágenes, enlaces y otros elementos multimedia.

CSS (Cascading Style Sheets)	Estilo y presentación	Se usa para definir el diseño visual de las páginas web, como colores, fuentes y disposición de los elementos.
JavaScript	Interactividad	Lenguaje de programación que permite agregar interactividad a las páginas web. Se usa para manejar eventos, validar formularios, y modificar dinámicamente el contenido HTML.
PHP (Hypertext Preprocessor)	Lado del servidor	Lenguaje de programación del lado del servidor para generar contenido dinámico y conectarse a bases de datos. Es comúnmente usado con HTML para crear páginas web interactivas.
SQL (Structured Query Language)	Gestión de bases de datos	Lenguaje utilizado para gestionar datos en bases de datos relacionales. Permite realizar consultas, insertar, actualizar y eliminar datos. Se usa junto con PHP para gestionar datos de usuarios y otros elementos de una página web.
Python	Lado del servidor	Aunque Python no se utiliza directamente en HTML, es popular para el desarrollo web del lado del servidor, especialmente con frameworks como Django y Flask. Se utiliza para lógica de negocio, interactuar con bases de datos y manejar aplicaciones web.
Ruby	Lado del servidor	Ruby, usado comúnmente con el framework Rails (Ruby on Rails), es un lenguaje dinámico orientado a objetos. Facilita el desarrollo de aplicaciones web completas y dinámicas.

Tabla 2 Partes que componen el programa

Capítulo 3.

3.1 METODOLOGÍA.

Utilice la metodología Scrum por la forma ágil de ser utilizado en el software, ya que proporciona un marco de trabajo flexible, iterativo y colaborativo. Su aplicación permite a los equipos adaptarse rápidamente a los cambios y entregar productos de alta calidad en menor tiempo. Podemos definirlo como un marco de gestión de proyectos. Ligero y fácil de aplicar, esta herramienta de gestión de procesos tiene como objetivo ayudar a los equipos a desarrollar soluciones flexibles y eficientes para procesos complejos, desde las etapas iniciales de organización desde los proyectos hasta su desarrollo ágil.

Mejora en la Adaptabilidad y Flexibilidad: Según Schwaber y Sutherland (los creadores de Scrum), esta metodología permite que el equipo sea más adaptable ante cambios y nuevos requisitos. En proyectos del campo geriátrico, donde las necesidades pueden evolucionar rápidamente, Scrum permite ajustes rápidos sin afectar el progreso del proyecto.

Mayor Transparencia y Colaboración: Rising y Janus destacan que Scrum fomenta la colaboración continua entre los equipos, los stakeholders y los usuarios. En un proyecto como Citango, donde se debe garantizar que las soluciones tecnológicas se ajusten a las necesidades de los pacientes y médicos, esta colaboración es clave. Las reuniones diarias de Scrum, como los Daily Standups, promueven la transparencia sobre el avance del proyecto y la resolución de bloqueos.

Reducción de Riesgos: Según Takeuchi y Nonaka, Scrum ayuda a reducir los riesgos en proyectos complejos al trabajar con ciclos cortos de retroalimentación. Al involucrar al cliente (o usuario final) en cada sprint, se identifican rápidamente los problemas y se pueden hacer ajustes antes de que se conviertan en grandes obstáculos. En proyectos geriátricos, donde las normativas y las necesidades de los pacientes pueden ser muy específicas, esta capacidad de adaptarse rápidamente es invaluable.

3.1.1 XAMPP. Para el uso del guardado de datos, seleccione este paquete de software libre que facilita la configuración de un entorno de desarrollo web en computadoras locales. Este sistema de gestión de bases de datos utiliza MySQL, en versiones más recientes, MariaDB. MySQL permite almacenar, gestionar y recuperar grandes cantidades de datos estructurados para aplicaciones web. Es indispensable su uso para almacenar los datos de manera local, dentro de la misma computadora o servidor, evitando que este sea compartido, manteniendo la información en un solo lugar.

XAMPP con MySQL ofrece varias funcionalidades para el desarrollo y la gestión de bases de datos, incluyendo:

- **Desarrollo y Pruebas Locales de Aplicaciones Web:** Permite a los desarrolladores simular un servidor web completo en su propia máquina, donde pueden crear y probar aplicaciones web sin necesidad de un servidor externo.
- **Manejo de Bases de Datos:** MySQL permite crear, modificar y eliminar bases de datos y sus tablas, gestionar relaciones entre datos, y realizar consultas para recuperar o manipular información.
- **Interacción con Lenguajes de Programación:** XAMPP incluye soporte para PHP y Perl, lo que facilita la creación de aplicaciones dinámicas que interactúan con MySQL para operaciones como autenticación de usuarios, gestión de inventarios, blogs, entre otros.
- **Interfaz de Gestión de Bases de Datos:** XAMPP incluye phpMyAdmin, una herramienta de administración de bases de datos basada en web que permite gestionar bases de datos MySQL/MariaDB visualmente, facilitando tareas como la creación de tablas, edición de datos, realización de consultas SQL, y copias de seguridad.

ESTRUCTURA DE LA BASE DE DATOS MySQL

4.1 Bases de Datos: Agrupación de tablas y otros objetos.
4.2 Tablas: Almacenan los datos en filas y columnas.
4.3 Columnas: Definidas con nombres y tipos de datos específicos.

4.4 Índices: Mejoran el rendimiento de las consultas.

4.5 Relaciones y Claves Foráneas: Establecen dependencias entre tablas.

4.6 Vistas: Consultas almacenadas que actúan como tablas virtuales.

4.7 Procedimientos y Funciones Almacenadas: Rutinan SQL almacenadas para operaciones complejas.

4.8 Triggers: Acciones automáticas basadas en eventos de la tabla.

4.9 Esquemas y Privilegios: Controlan el acceso y las modificaciones en los datos.

4.10 Claves Primarias y Foráneas: Las claves primarias identifican de forma única cada fila de una tabla, mientras que las claves foráneas crean relaciones entre tablas para representar la conexión entre distintos conjuntos de datos.

SEGURIDAD

Autenticación y Control de Acceso: MySQL gestiona el acceso a las bases de datos mediante un sistema de usuarios y privilegios. Los administradores pueden otorgar permisos específicos a usuarios, controlando qué operaciones pueden realizar.

Cifrado de Conexiones: MySQL soporta conexiones cifradas mediante SSL/TLS, lo que protege la comunicación entre el servidor de base de datos y los clientes contra interceptaciones y accesos no autorizados.

Logs y Auditoría: MySQL ofrece opciones de registro que permiten rastrear el acceso a la base de datos y las operaciones realizadas. Esta auditoría es útil para detectar comportamientos sospechosos y para cumplir con normas de seguridad de la información.

3.1.2 LARAVEL.

Es un framework de PHP de código abierto diseñado para simplificar el desarrollo de aplicaciones web complejas y escalables. Laravel sigue el patrón arquitectónico MVC (Modelo-Vista-Controlador) y cuenta con una estructura organizada que facilita la implementación de funcionalidades comunes en el desarrollo de aplicaciones, como autenticación, gestión de sesiones, enrutamiento y manejo de bases de datos.

SERVICIOS QUE OFRECE LARAVEL

- **Autenticación y Autorización:** Laravel cuenta con un sistema de autenticación fácil de configurar y una API de autorización flexible para gestionar el acceso a diferentes partes de la aplicación.
- **Eloquent ORM:** Es el sistema ORM (Object Relational Mapping) de Laravel, que facilita la interacción con bases de datos, permitiendo trabajar con modelos en lugar de sentencias SQL directas.
- **Migraciones y Seeders:** Laravel facilita la gestión y sincronización de bases de datos en diferentes entornos mediante migraciones. Los seeders permiten llenar la base de datos con datos de prueba.
- **API REST y soporte para GraphQL:** Laravel ofrece soporte para la creación de APIs RESTful y puede integrarse con GraphQL, permitiendo una comunicación fluida entre frontend y backend.
- **Manejo de colas y trabajos en segundo plano:** Laravel permite la ejecución de tareas en segundo plano para mejorar la eficiencia, como el envío de correos electrónicos o la generación de reportes.

IMPLEMENTACIÓN DE LARAVEL EN ANGULAR

Para el diseño del proyecto, mi selección fue Angular, este framework para el Frontend, se complementa con Laravel en Backend para construir aplicaciones web modernas. En este tipo de implementación, Angular se encarga de la interfaz de usuario y la experiencia del cliente, mientras que Laravel maneja las operaciones del lado del servidor, tales como la lógica de negocio, las operaciones con bases de datos y la gestión de sesiones.

3.1 **Crear la API en Laravel:** Laravel ofrece una manera simple de definir rutas y controladores para crear endpoints de API.

3.2 **Configurar Angular para consumir la API:** En el proyecto Angular, se configuran servicios para hacer solicitudes HTTP hacia los endpoints de Laravel.

3.3 **Manejo de autenticación:** Para la autenticación, Laravel proporciona el paquete Laravel Passport o Sanctum para gestionar tokens de acceso.

3.1.1 ANGULAR. Seleccione este software de desarrollo para construir sobre TypeScript. Es un framework basado en componentes para crear aplicaciones web escalables. Con esto comenzó a realizarse mi proyecto, con una amplia colección de bibliotecas bien integradas que cubren varias características, estos incluyen enrutamiento, administración de formularios, comunicación cliente-servidor y más. Basándome en la estructura del contenido que se operan, logre vincular las páginas una tras otra, para poder desplegarse de las ventanas.

Componente. Los componentes son la unidad básica en Angular. Representan una parte de la interfaz de usuario (UI) y están compuestos por tres elementos principales: una plantilla HTML, una clase de TypeScript y una hoja de estilos. Los componentes encapsulan la lógica y la visualización de una parte de la UI, permitiendo construir interfaces modulares y reutilizables

Módulos. Los módulos agrupan varios componentes y otros elementos de Angular en un solo lugar. Angular siempre tiene al menos un módulo raíz (AppModule), pero se pueden crear módulos adicionales para dividir la aplicación. Permiten la organización y encapsulación de diferentes partes de la aplicación, facilitando la gestión de dependencias y la carga de funcionalidades.

Servicios. Los servicios son clases que encapsulan lógica de negocio o lógica que se comparte entre varios componentes. Promueven la reutilización y la separación de responsabilidades al centralizar ciertas funcionalidades.

Angular Material. Biblioteca de componentes basada en Material Design, creada específicamente para aplicaciones Angular. Esta herramienta facilita el desarrollo de interfaces de usuario modernas, coherentes y responsivas, aprovechando los principios de diseño visual y usabilidad de Material Design. Con Angular Material, puedes incorporar rápidamente componentes preconstruidos, como botones, tablas, formularios y más, sin tener que diseñarlos desde cero.

3.1.4 BOOTSTRAP. Este framework de CSS, HTML y JavaScript lo utilice en desarrollo de aplicaciones web receptivas y móviles. Boostrap usa estilos que facilitan la interfaz del usuario sin necesidad de escribir tanto, utilizando la extensión en mi proyecto de angular, se puede utilizar sin la necesidad de conectarse a internet. El manejo de las vistas es parte de este framework que este hecho para dar color al diseño de una página web.

IMPLEMENTACIÓN DE BOOTSTRAP EN ANGULAR

Para utilizar Bootstrap en un proyecto Angular, es necesario incluir los archivos de estilos y scripts de este tiene al proyecto. La forma más común y recomendada de instalar Bootstrap en Angular es utilizando npm (Node Package Manager), que facilita la gestión de dependencias en el proyecto Angular. Para instalar Bootstrap, puedes ejecutar el siguiente comando en la terminal: *`npm install Bootstrap`*

En la implementación de Bootstrap se utiliza mas en diseños, dando varios estilos personalizados a un proyecto de Angular (Pagina Web), y mostrar una vista amigable, entre ellos se puede dar estilo a las tablas, botones, vistas, entre otros.

3.1.5 VISUAL CODE (VS CODE). Seleccione este editor de código fuente gratuito y de código abierto para hacer mi proyecto, ya que me fue fácil utilizarlo. Su gran ampliación de lenguajes que interpreta, ayudando a la programación del proyecto. Es compatible con múltiples lenguajes de programación, incluyendo JavaScript, Python, C++, PHP, entre otros.

3.1.6 GitHub. Es una plataforma basada en la web para la gestión de repositorios Git, que permite la colaboración y el control de versiones en proyectos de software. GitHub facilita a los desarrolladores alojar sus proyectos y colaborar en ellos de manera eficiente a través de la integración de Git.

3.1.7 POSTMAN. Esta herramienta de desarrollo, lo utilice para probar y depurar APIs (Interfaces de Programación de Aplicaciones) con mi proyecto entre angular y laravel.

Permite enviar solicitudes HTTP a un servidor y observar las respuestas de la API. Es ampliamente utilizada en el desarrollo de aplicaciones web y móviles para asegurarse de que las APIs funcionan correctamente. Con esto de hizo las pruebas de las conexiones que tiene con la base de datos, y así obtener los resultados esperados, con las conexiones que tiene el Frontend y Backend del proyecto.

Método HTTP	Descripción	Uso Común
GET	Recupera datos del servidor. No modifica ningún recurso.	Obtener información, leer datos (e.g., consultar usuarios).
POST	Envía datos al servidor para crear un nuevo recurso.	Crear nuevos registros, enviar formularios.
PUT	Actualiza un recurso existente en el servidor. Reemplaza todo el recurso con la nueva versión.	Actualizar un registro completo (e.g., editar un perfil).
PATCH	Realiza una actualización parcial de un recurso.	Actualizar parcialmente un recurso (e.g., cambiar un solo campo).
DELETE	Elimina un recurso del servidor.	Eliminar un registro o recurso (e.g., borrar un usuario).
HEAD	Similar a GET, pero solo recupera los encabezados de la respuesta (sin cuerpo).	Verificar metadatos de un recurso (e.g., comprobar si existe).
OPTIONS	Obtiene los métodos HTTP soportados por un servidor para un recurso específico.	Determinar qué métodos están disponibles para un recurso.
TRACE	Realiza una prueba de ruta, mostrando los pasos por los que ha pasado la solicitud en el servidor.	Depuración de redes, rastrear la ruta de una solicitud.
CONNECT	Establece una conexión de red para usar un protocolo como HTTP o HTTPS.	Usado principalmente para establecer túneles proxy.

Tabla 3 Método de Aplicación POSTMAN para las conexiones

3.2 DIAGRAMA DE BLOQUES DE LAS ETAPAS DEL PROYECTO.

3.2.1 DIAGRAMA DE ENTIDAD-RELACIÓN

Entidades Principales:

1. Paciente: Contiene los datos de los pacientes, como id, nombre, apellido, fecha_nacimiento, género, estado_civil, y contacto.
2. Doctor: Almacena información sobre los doctores, incluyendo id, nombre, especialidad, contacto, y email.
3. Encuesta: Representa los diferentes tipos de encuestas (por ejemplo, funcional, psicoafectiva) con atributos como id, nombre, y descripción.
4. Pregunta: Cada pregunta que forma parte de las encuestas, con atributos como id, contenido, y tipo_respuesta.
5. Respuesta: Almacena la respuesta seleccionada para una pregunta específica, con atributos como id, id_pregunta, opcion_seleccionada, y puntaje.
6. Resultado: Contiene los resultados de cada encuesta para cada paciente, como id, id_paciente, id_encuesta, puntaje_total, y diagnóstico.

Relaciones:

1. Paciente-Encuesta (1): Un paciente puede tener múltiples encuestas asociadas.
2. Encuesta-Pregunta (1): Cada encuesta contiene múltiples preguntas.
3. Pregunta-Respuesta (1): Una pregunta puede tener varias respuestas posibles (dependiendo del tipo de pregunta).
4. Paciente-Doctor(N): Relación de asignación de pacientes a doctores, si los pacientes pueden ser atendidos por varios doctores.

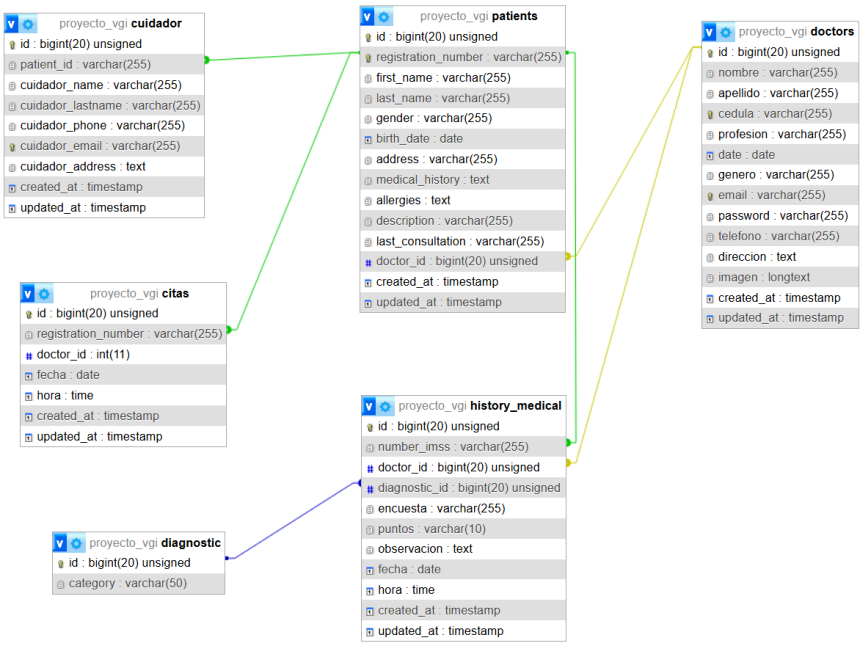


Ilustración 5 Diagrama Entidad Relación

3.2.2 DIAGRAMA DE CASOS DE USO.

Módulo de Gestión de Pacientes:

1. Registrar Paciente: Permite al personal médico o administrativo agregar nuevos pacientes al sistema.
2. Editar Información del Paciente: Permite actualizar los datos de un paciente existente.
3. Consultar Información del Paciente: Visualización detallada de la información de un paciente.

Módulo de Evaluación Geriátrica

1. Crear Encuesta: Permite crear diferentes tipos de encuestas.
2. Asignar Encuesta a Paciente: Permite asignar una encuesta a un paciente específico.
3. Responder Encuesta: Permite que el doctor o el mismo paciente responda las preguntas de la encuesta.

4. Calcular Resultado: Una vez completada la encuesta, se calcula el puntaje total y el diagnóstico.

Módulo de Resultados

1. Visualizar Resultados de Encuestas: Permite a los médicos visualizar el puntaje total y diagnóstico para cada encuesta realizada por un paciente.
2. Historial de Resultados: Permite ver el historial de encuestas y diagnósticos para cada paciente.

Módulo de Gestión de Doctores

1. Registrar Doctor: Permite agregar nuevos doctores al sistema.
2. Asignar Paciente a Doctor: Relaciona un paciente con uno o varios doctores.

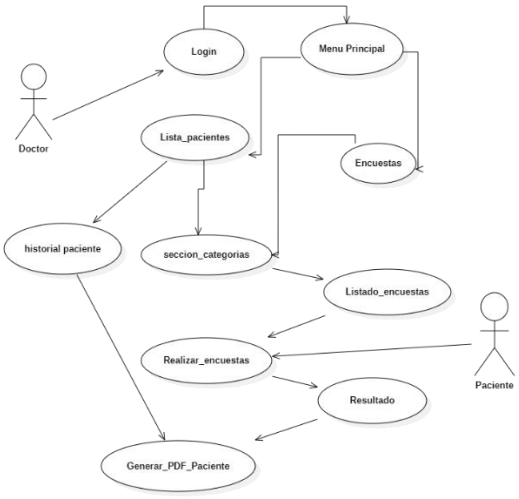


Ilustración 6 Diagrama de Caso de uso

Capítulo 4.

4.1 RESULTADOS.

En este apartado se hicieron las capturas del proyecto, dando como resultado las funciones que esta emplea, dando una interfaz sencilla que facilita el uso del programa sin importar el conocimiento que tenga, esta sección esta implementada su funcionalidad de cada apartado del programa.

4.1.1 Vista de Iniciar Sesión

Ventana de vista acerca de inicio de sesión del usuario, donde este ingresa sus datos para acceder a la plataforma, en tal caso deberá esta registrado por parte de la organización, en caso contrario muestra los mensajes de error respectivos de fallo que contenga, contraseña incorrecta o no está registrado.

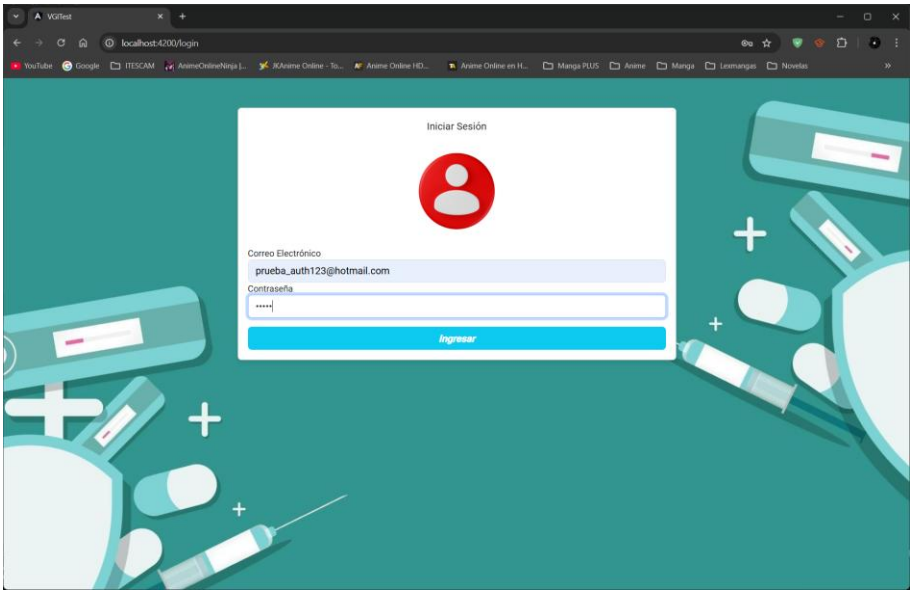


Ilustración 7 Sesión Administrador

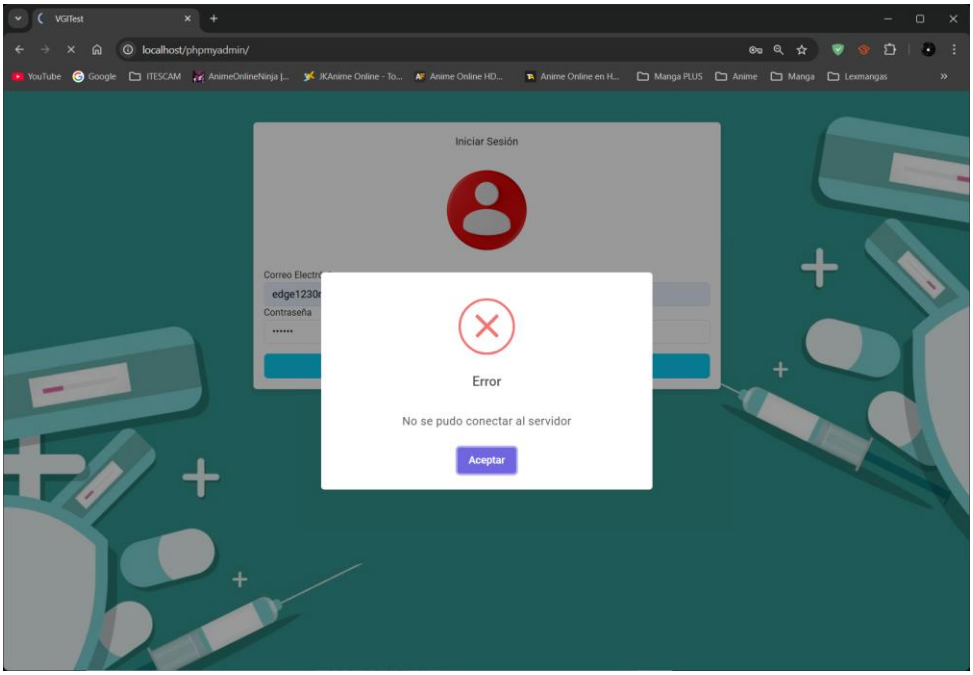


Ilustración 8 Vista del Error de Usuario/Contraseña

Funcionalidad:

- Validación de Usuario que este registrado en la plataforma.
- Vista de error en caso de error según sea su caso.

4.1.2 Dashboard Principal

Pantalla principal al entrar a la aplicación, en el lado izquierdo, se muestra las pestañas de acceso, donde se selecciona que se quiere realizar, hay dos vistas de principal, como parte del cliente, se autorizó que el propietario podía dar de alta a los médicos. Por lo que la vista del doctor no podrá ver este apartado.

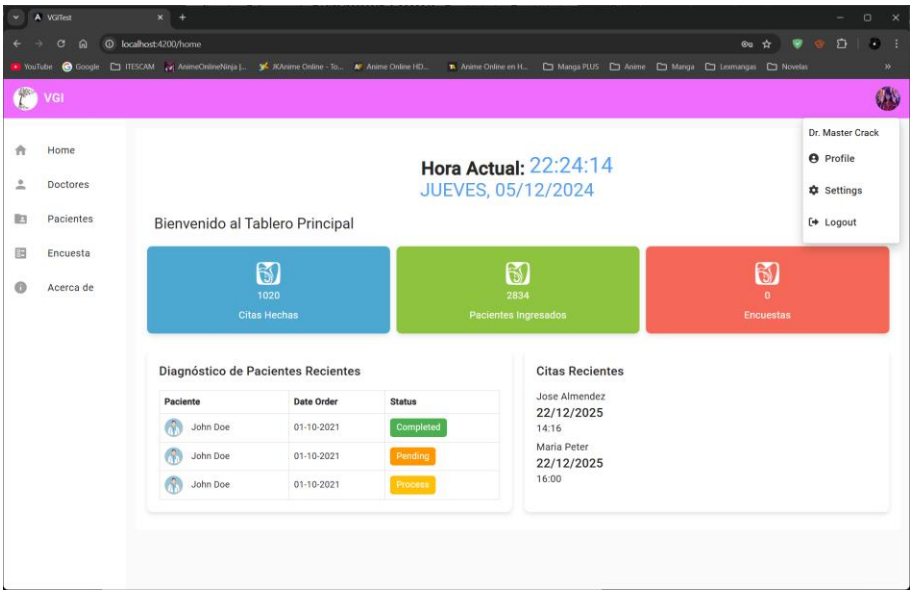


Ilustración 9 Vista Principal Administrador

Funcionalidad:

- Creación y vista de las citas realizadas por el doctor.
- Hora y Fecha en tiempo real.
- Vista general de Paciente ingresado, Diagnostico hechos y Citas Hechas el mismo día.
- Vista de pacientes diagnosticados por el doctor.

4.1.3 Vista de Tabla de Doctor (Administrador).

En esta vista, solo el administrador autorizado puede ver la lista de los doctores, en caso contrario no tendrá acceso a esta vista, excluyendo a si mismo de la lista, al mismo tiempo que puede cambiar la contraseña de los doctores en caso de pérdida o dar de baja alguno.

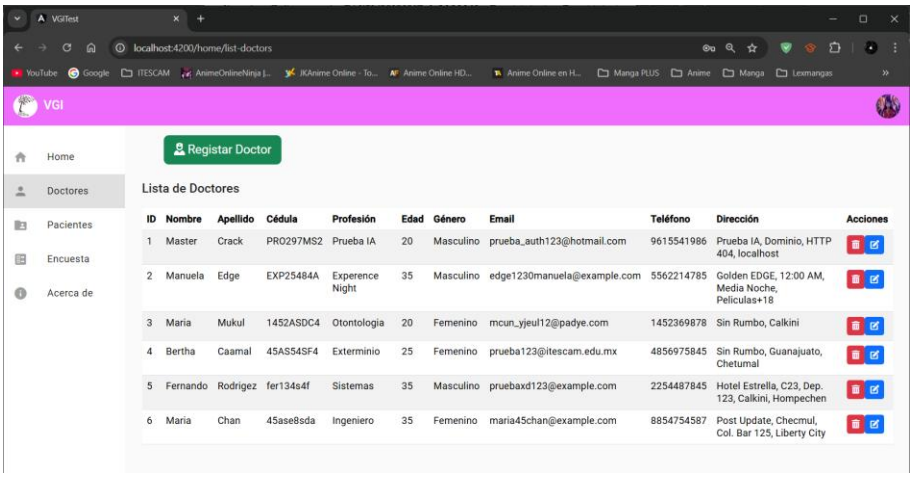


Ilustración 10 Tabla de doctores visible por administrador

Funcionalidad:

- Vista de algunos datos del Usuario (Doctor) excluyéndose.
- Editar contraseña de los usuarios (Perdida de contraseña) o dar de baja algún usuario.

4.1.4 Vista del Registro

La parte de registro está diseñado de tal manera que se facilite su uso, aquí se muestra el registro del paciente y doctor (Administrador), en caso del doctor solo el responsable puede hacerlo, y está diseñado para que únicamente puedan dar de alta en caso especial por parte del propietario.

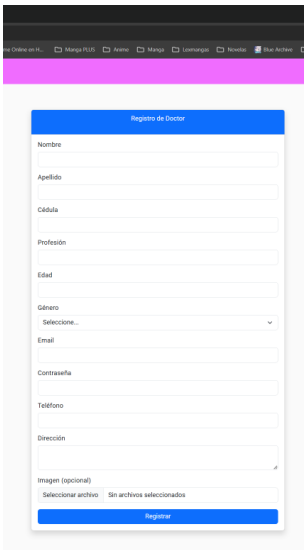


Ilustración 11 Registro del doctor por parte del administrador

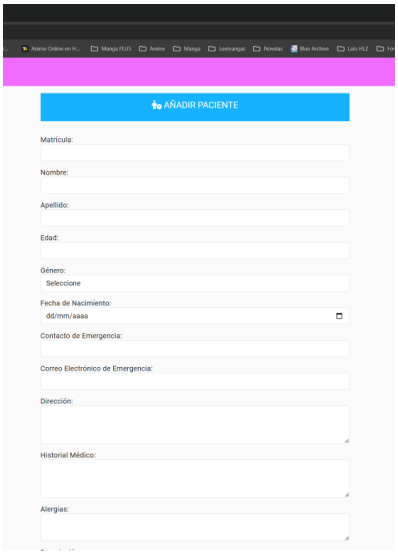


Ilustración 12 Vista Registro por doctor

Funcionalidad:

1. Registro del Paciente con interfaz sencilla de manejar.
2. Registro del Doctor (Administrador) con diseño sencillo para su uso.

4.1.5 Vista de la encuesta

Vista de la categoría para acceder a las encuestas a realizar por parte de los doctores, mostrando la lista de encuesta que se poder utilizar para su evaluación.

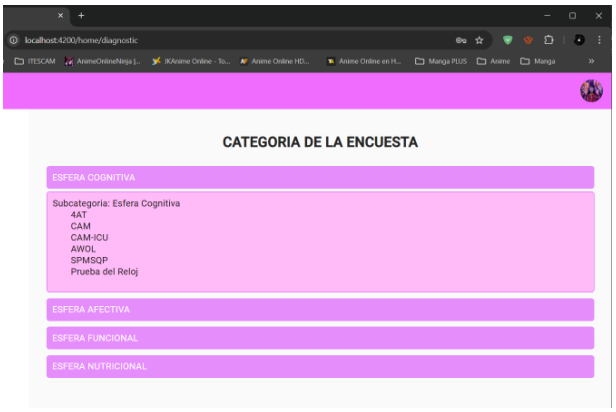


Ilustración 13 Vista de Encuesta

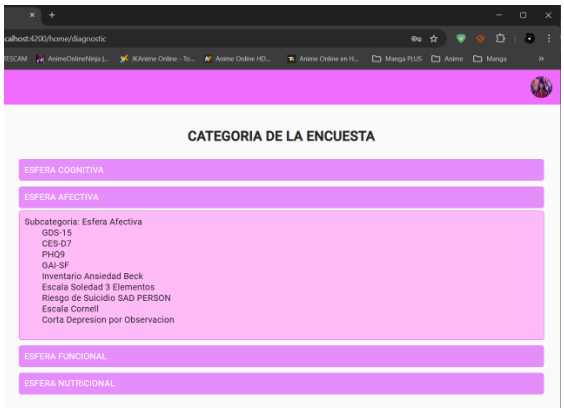


Ilustración 14 Vista de Encuesta 2

Funcionalidad:

- Amplia vista de encuesta a realizar por parte del doctor
- Esta parte se muestra una encuesta seleccionada, para ver su estructura y sencillez de usar la aplicación.

En esta parte se muestra la encuesta que se realizara el puntaje según marque el usuario (Doctor) para obtener el diagnostico.

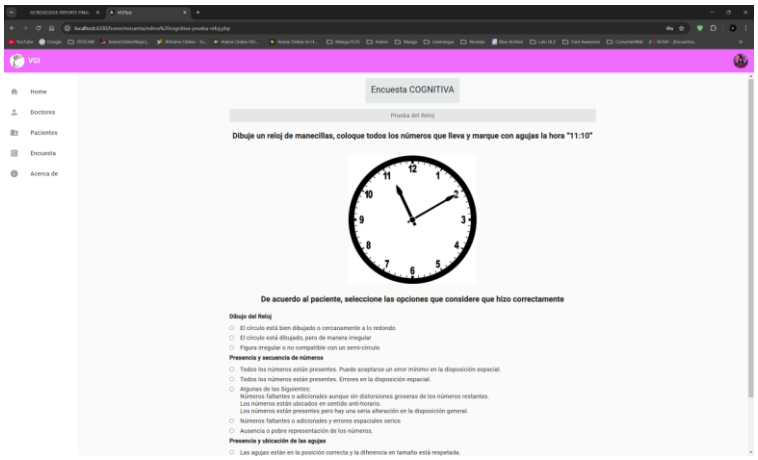


Ilustración 15 Vista de Encuesta

Funcionalidad:

- Muestra la encuesta de acuerdo con la ventana anterior.
- Calcula los puntajes antes de pasar a resultado.
- Múltiples encuestas en una sola ventana.

Aquí se muestra el resultado de acuerdo con la encuesta anterior, dando la información de acuerdo con el propietario, mostrando su diagnóstico que tiene el paciente.

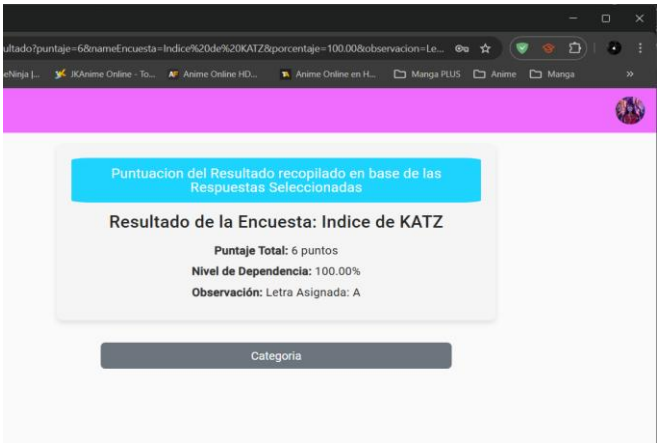


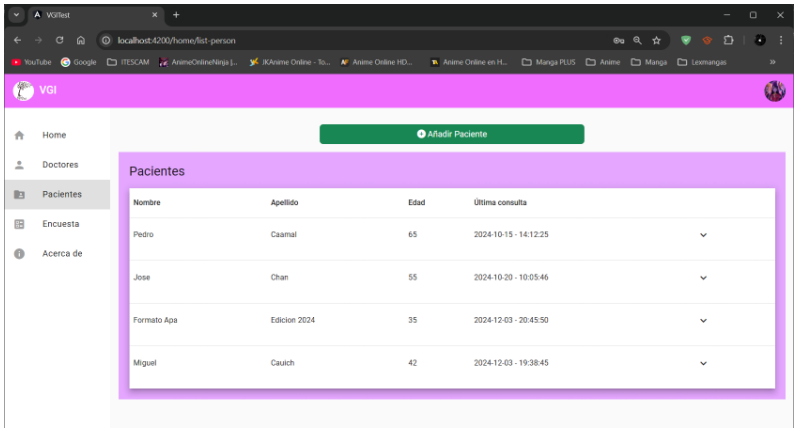
Ilustración 16 Vista de Resultado

Funcionalidad.

- Vista previa del diagnóstico del paciente
- En caso de guardar su id, se guarda automáticamente en la base de datos

4.1.6 Lista de Paciente y Perfil

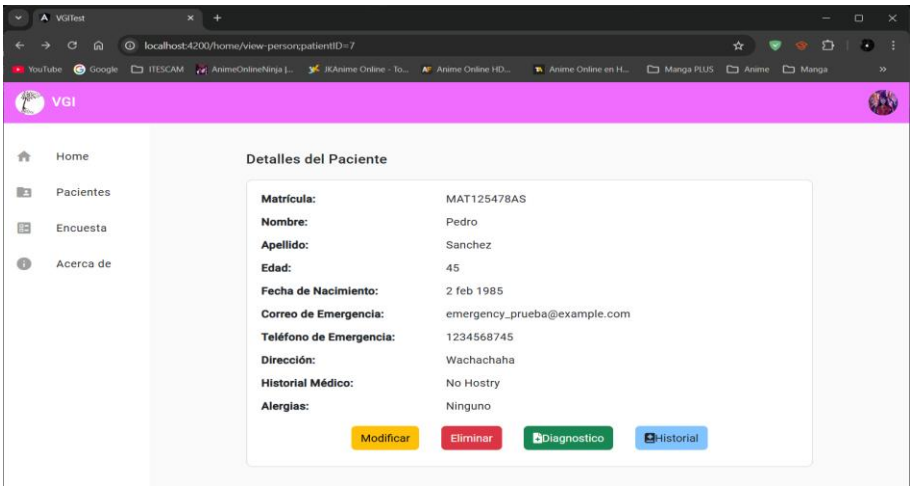
En esta ventana se muestra la lista paciente de cada doctor, únicamente del doctor que lo haya registrado puede ver su lista, aquí es donde se registra al paciente.



Funcionalidad.

- Generación de lista de paciente del doctor.
- Información básica del paciente
- Ultima consulta del paciente.

Por esta parte es la vista del paciente, su información básica necesario, como nombre, edad, genero, fecha etc. Igual aquí puede eliminar el paciente o modificar su ficha.



Funcionalidad

- Información básica del paciente
- Modificar, Eliminar e Historial del paciente

4.1.7 Vista del Perfil del Doctor.

En esta ventana muestra el perfil del doctor, su información básica, igualmente se puede cambiar sus datos en el icono del lápiz en caso de tener un error, posteriormente cambiar su contraseña o correo en caso de requerirlo.

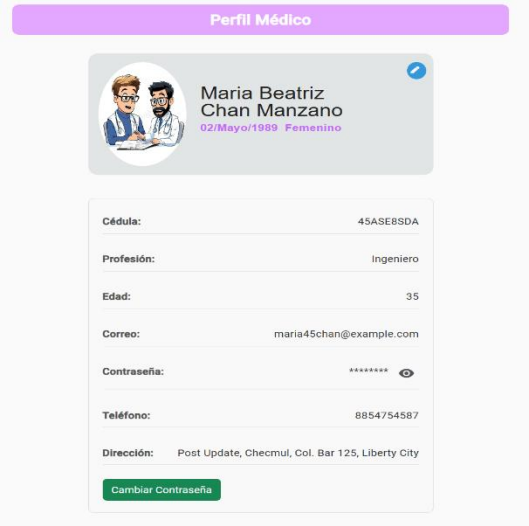


Ilustración 17 Vista del Perfil

Ilustración 18 Actualización de Contraseña y/o Correo

Ilustración 19 Cambio de Contraseña

Funcionalidad

- Vista de la información básica del doctor.
- Actualizar sus datos del doctor.
- Cambio de contraseña y/o Correo

4.2 Prueba de la Interfaz de Usuario

Criterio 1: Claridad de la Interfaz

Descripción: Los menús, botones y opciones están bien organizados y son comprensibles sin necesidad de instrucciones adicionales.

Resultado: ⚠ Parcialmente Correcto

Observaciones: El diseño de los botones, opciones y menús esta organizados para emplear su función de acuerdo con la sencilles de captar al usuario, se le dio al propietario y le fue amigable su estructura del programa.

Criterio 2: Simplicidad del diseño

Descripción: La aplicación evita sobrecarga de información y presenta un diseño limpio con elementos visuales fáciles de identificar.

Resultado: ✅ Correcto

Observaciones: El diseño no está saturado de información o colores con tonos fuertes, dándole al usuario una amigable interfaz de su agrado y fácil de manejar.

Criterio 3: Facilidad de navegación

Descripción: Los usuarios pueden moverse entre pantallas sin dificultad y sin perderse en la estructura del sistema.

Resultado: ✅ Correcto

Observaciones: Se han implementado rutas bien definidas y un flujo de navegación claro, lo que permite a los usuarios desplazarse por la aplicación sin confusión, sin la necesidad de perderse entre las páginas que en la aplicación, aumentando la sencilles de la aplicación.

Criterio 4: Uso de Lenguaje Sencillo

Descripción: Los textos y mensajes dentro de la aplicación son comprensibles para personas sin conocimientos técnicos. Se evitan términos ambiguos o jerga innecesaria.

Resultado: ✅ Correcto

Observaciones: La redacción de mensajes y etiquetas se ha trabajado para garantizar que cualquier usuario pueda comprender su significado sin dificultad, así mismo sea agradable el texto, sin poner mucho y que solo se use palabras cortas que lo describan sin excederlo de carga.

Criterio 5: Consistencia en la Interfaz

Descripción: Los elementos visuales y de navegación mantienen un diseño uniforme en toda la aplicación, asegurando coherencia en botones, fuentes y estilos. Sin embargo, algunas secciones pueden presentar variaciones en su estructura.

Resultado: ⚠ Parcialmente Correcto

Observaciones: Aunque la mayoría de los componentes siguen una misma línea de diseño, hay algunas diferencias en la disposición de elementos que podrían mejorar para dar una apariencia más homogénea.

Criterio 6: Accesibilidad Visual

Descripción: Se utilizan colores, contrastes y tamaños de letra adecuados para facilitar la lectura y la interacción, considerando aspectos como la visibilidad en diferentes dispositivos y condiciones de iluminación.

Resultado: ✅ Parcialmente Correcto

Observaciones: La combinación de colores y fuentes se ha implementado siguiendo buenas prácticas de accesibilidad, permitiendo una mejor experiencia de uso para personas con dificultades visuales.

Criterio 7: Errores y recuperación

Descripción: Si un usuario comete un error, la aplicación debería proporcionar mensajes claros y opciones para corregirlo fácilmente. Sin embargo, no se han implementado suficientes mecanismos para guiar al usuario en caso de equivocación.

Resultado: ❌ Parcialmente Correcto

Observaciones: No se presentan mensajes de error detallados ni opciones de recuperación rápida en ciertos procesos críticos, lo que puede generar confusión o frustración en los usuarios.

Criterio 8: Facilidad en Tareas Básicas

Descripción: Las acciones más comunes, como iniciar sesión, realizar una evaluación o enviar respuestas, pueden completarse sin dificultades. Los procesos son directos y no requieren pasos innecesarios.

Resultado: ✅ Parcialmente Correcto

Observaciones: Los flujos de interacción han sido optimizados para minimizar el tiempo y la complejidad de completar las tareas más frecuentes dentro de la aplicación.

Criterio 9: Feedback Inmediato

Descripción: La aplicación proporciona confirmaciones visuales o auditivas cuando un usuario realiza una acción, asegurando que este reciba una respuesta clara tras cada interacción.

Resultado: ✅ Parcialmente Correcto

Observaciones: Se han implementado notificaciones y cambios visuales (como resaltado de botones o mensajes emergentes) para informar al usuario sobre el estado de sus acciones.

Criterio 10: Compatibilidad con Dispositivos

Descripción: La interfaz debería adaptarse correctamente a diferentes dispositivos (tabletas, computadoras, celulares) mediante un diseño responsivo. Actualmente, presenta inconsistencias en su visualización en algunos tamaños de pantalla.

Resultado: ❌ Incorrecto

Observaciones: En ciertas resoluciones, los elementos se desordenan o no se ajustan correctamente, lo que afecta la experiencia del usuario en dispositivos móviles o pantallas más pequeñas.

4.3 Conclusiones y Recomendaciones.

En la realización de este proyecto, de tuvo en cuenta el punto de vista del cliente, de cómo quiere que funcione su aplicación, dando una idea de cómo va a ir tomando rumbo este proyecto, y es así como se concluye la realización del proyecto, asegurando que funcione como se había planteado anteriormente. El uso de los programas se llevó a cabo acerca del problema que surgió con los diagnósticos que tuvo el cliente, dándole una oportunidad sobre el uso de la tecnología en el campo geriátrico.

La compatibilidad de estos programas ayudo a resolver la problemática acerca de la obtención de resultados, facilitando el guardado de la información en la computadora y acceder en cualquier momento, la capacidad de la creación de las vistas del proyecto ayudo a entender que se requiere tiempo para poder hacer un diseño que sea adaptable para todos los usuarios, sin importar su conocimiento de tecnología, obtenido las pruebas de su funcionamiento.

Dado que la estructura de usuarios en este proyecto es sencilla, con roles básicos como administradores y usuarios, se puede gestionar eficientemente la autenticación y autorización sin necesidad de una configuración compleja. Además, se puede interactuar de manera sencilla con estas funcionalidades del Backend, mostrando solo la información relevante para los distintos roles de usuario y garantizando una experiencia de navegación rápida y segura.

Para resumir, el uso del programa de Angular Laravel junto a XAMPP, resulto ser uno de los proyectos que obtuvieron buenos resultados en cuanto a su funcionamiento requerido, finalizando así el proyecto con los roles y capturas de datos básico, donde se puede almacenar, y obtener los puntajes de los diagnósticos realizados por parte del programa.

Recomendación.

- El uso de la aplicación es propiedad del cliente, lo cual no se distribuirá a cualquiera.
- Revisión mensual o trimestral por mantenimiento o actualización del programa, y sus extensiones.

- Dar buen uso de esta aplicación por parte del administrador y/o empleados.
- Equipo necesario para su uso, encaso es una computadora o laptop.
- Instalar los componentes necesarios para su funcionamiento del programa.
- El propietario es responsable del uso del programa.

Referencias Bibliográficas.

Berg, M., & Goorman, E. (1999). The contextual nature of medical information. International Journal of Medical Informatics, 56(1-3), 51-60. https://doi.org/10.1016/S1386-5056(99)00039-3

Cassel, C. K., Leipzig, R. M., Cohen, H. J., Larson, E. B., & Meier, D. E. (2012). Geriatric medicine: An evidence-based approach (4th ed.). Springer.

Pardes, H., Levey, S., & Donohue, C. (2018). Challenges and opportunities for the use of health information technology in geriatric care. Journal of the American Geriatrics Society, 66(1), 21-28. https://doi.org/10.1111/jgs.15190

Rindfleisch, T. C. (1997). Privacy, information technology, and health care. Communications of the ACM, 40(8), 92-100. https://doi.org/10.1145/257874.257896

Rowe, J. W., Fulmer, T., & Fried, L. (2006). Building a better delivery system for older adults. Health Affairs, 25(2), 580-591. https://doi.org/10.1377/hlthaff.25.2.580

Shahmoradi, L., Safadari, R., & Jimma, W. (2017). Electronic health record implementation: A review of resources and recommendations. Journal of Medical Systems, 41(11), 182. https://doi.org/10.1007/s10916-017-0835-6

American Geriatrics Society. (2014). Improving health care for older adults: An integrated approach. Journal of the American Geriatrics Society, 62(8), 1481-1487. https://doi.org/10.1111/jgs.12991

Blumenthal, D., & Tavenner, M. (2010). The “meaningful use” regulation for electronic health records. New England Journal of Medicine, 363(6), 501-504.

Boehm, E., Eisenberg, M., & Freedman, V. A. (2014). The role of health information technology in improving care coordination for older adults. Healthcare, 2(3), 194-201.

Kohn, L. T., Corrigan, J. M., & Donaldson, M. S. (2000). To err is human: Building a safer health system. National Academy Press.

Menachemi, N., & Collum, T. H. (2011). Benefits and drawbacks of electronic health record systems. Risk Management and Healthcare Policy, 4, 47-55.

Katz, P. R., & Karuza, J. (2005). Specialized care for older adults: Focus on geriatrics. Journal of the American Medical Directors Association, 6(6), 399-403. https://doi.org/10.1016/j.jamda.2005.09.003

Rowe, J. W., & Kahn, R. L. (1997). Successful aging. The Gerontologist, 37(4), 433-440. https://doi.org/10.1093/geront/37.4.433

Rubenstein, L. Z., & Josephson, K. R. (2002). The epidemiology of falls and syncope. In B. J. Ham & L. L. Rubenstein (Eds.), Falls and syncope in elderly patients (pp. 1-12). Clinics in Geriatric Medicine.

Stuck, A. E., Siu, A. L., Wieland, G. D., Adams, J., & Rubenstein, L. Z. (1993). Comprehensive geriatric assessment: A meta-analysis of controlled trials. The Lancet, 342(8878), 1032-1036. https://doi.org/10.1016/0140-6736(93)92884-V

Angular. (s.f.). Components. Recuperado de https://angular.io/guide/component-overview

Bootstrap. (s.f.). Introduction. Recuperado de https://getbootstrap.com/docs/5.3/getting-started/introduction/

Bootstrap. (s.f.). JavaScript Bundle. Recuperado de https://getbootstrap.com/docs/5.3/getting-started/download/#javascript-bundle

Angular. (s.f.). CLI Workspace Configuration. Recuperado de https://angular.io/guide/workspace-config

Angular y Bootstrap. (2023). Add Bootstrap to Angular. Recuperado de https://angular.io/guide/using-libraries

Angular. (s.f.). Modules. Recuperado de https://angular.io/guide/ngmodules

Laravel. (s.f.). The PHP Framework for Web Artisans. Recuperado de https://laravel.com/docs

Oracle Corporation. (s.f.). MySQL. Recuperado de https://dev.mysql.com

Node.js Foundation. (s.f.). Node.js. Recuperado de https://nodejs.org

Microsoft. (s.f.). Visual Studio Code. Recuperado de https://code.visualstudio.com

Git SCM. (s.f.). Git. Recuperado de https://git-scm.com

GitHub, Inc. (s.f.). GitHub. Recuperado de https://github.com

Postman, Inc. (s.f.). Postman. Recuperado de https://www.postman.com

Composer. (s.f.). Composer. Recuperado de https://getcomposer.org

Martins, J. (2024, February 15). Scrum: conceptos clave y cómo se aplica en la gestión de proyectos [2024] • Asana. Asana. https://asana.com/es/resources/what-is-scrum

Qué es SCRUM. (2021, September 20). Proyectos Ágiles. https://proyectosagiles.org/que-es-scrum/

Schwaber, K., & Sutherland, J. (2020). The Scrum Guide. Scrum.org.

Cohn, M. (2010). Succeeding with Agile: Software Development Using Scrum. Addison-Wesley.

Rubin, K. S. (2012). Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley.

Highsmith, J. (2009). Agile Project Management: Creating Innovative Products. Addison-Wesley.

Schwaber, K., & Sutherland, J. (2020). The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game.

Monja, M. L. (2024, October 11). Metodologías ágiles: definición, manifiesto, principios, SCRUM, Kanban. Innovar o Morir. https://innovaromorir.com/metodologias-agiles-definicion-manifiesto-principios-scrum-kanban/

Atlasiano, E. (n.d.). Los tres pilares del scrum: conoce los principios fundamentales del scrum. Atlassian. https://www.atlassian.com/es/agile/project-management/3-pillars-scrum

Anexos.

Capturas de Base de Datos de XAMPP

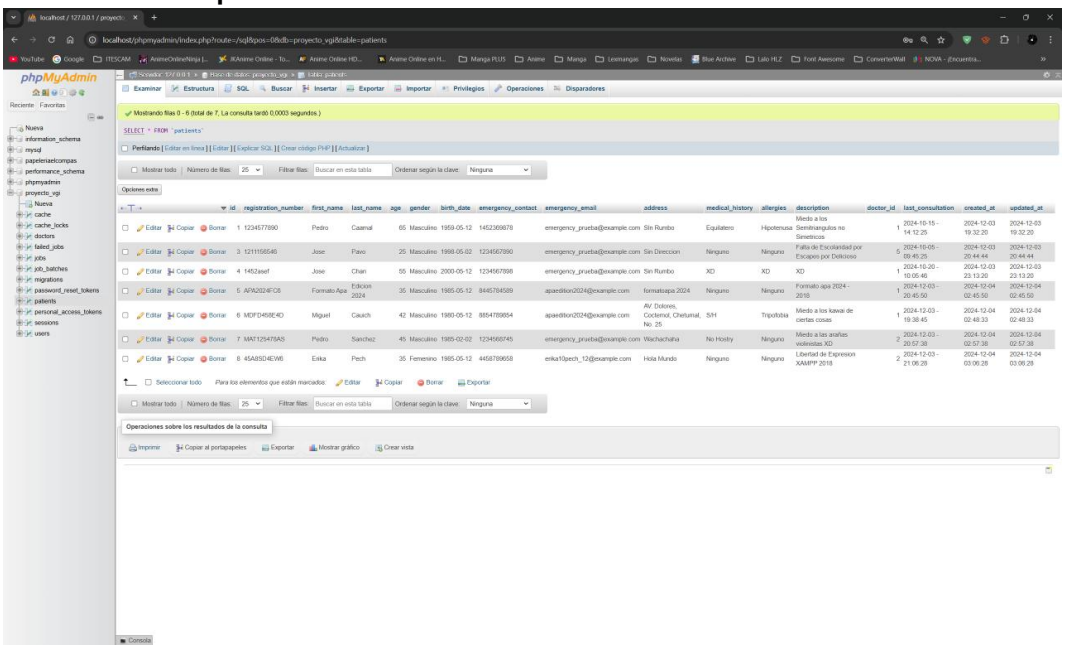


Ilustración 20 Tabla de Pacientes MySQL

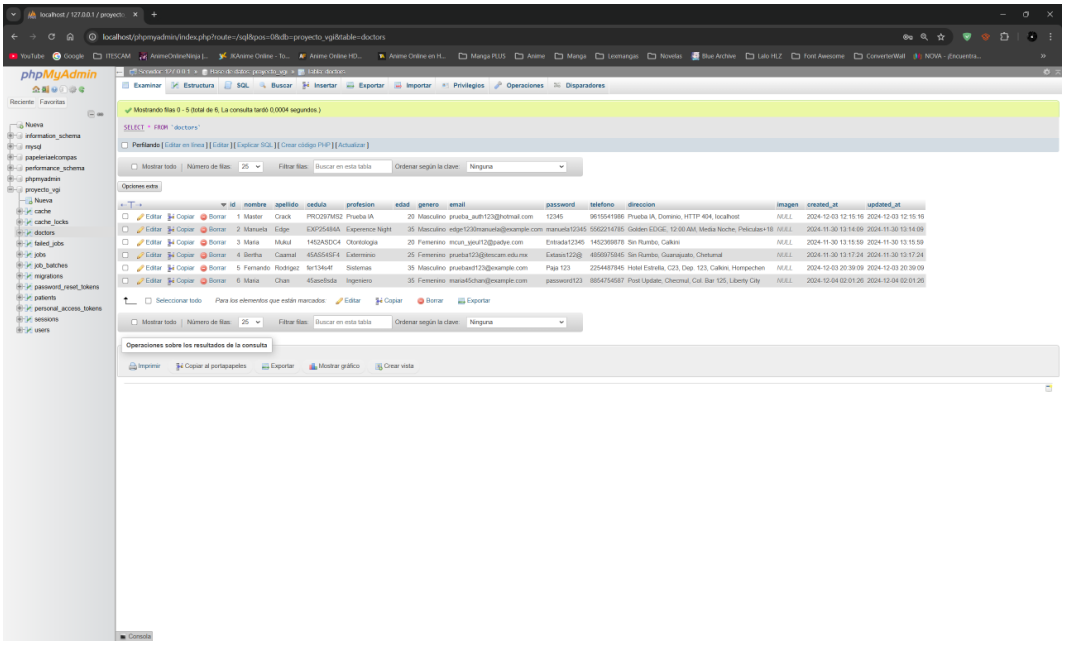


Ilustración 21 Tabla de Doctores MySQL

Captura de Código de Laravel

Código del Doctor

CREACIÓN DE LA TABLA

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('doctors', function (Blueprint $table) {
            $table->id(); // Esto define el campo ID como autoincremental
            $table->string('nombre');
            $table->string('apellido');
            $table->string('cedula')->unique();
            $table->string('profesion');
            $table->integer('edad');
            $table->string('genero');
            $table->string('email')->unique();
            $table->string('password');
            $table->string('telefono');
            $table->text('direccion');
            $table->string('imagen')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('doctors');
    }
};
```

OPERACIÓN DEL CONTROLADOR

```
<?php
```

```php
namespace App\Http\Controllers;

use App\Models\Doctors;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class DoctorsController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        return response()->json(Doctors::all(), 200);
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        // NO SE USA
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        // Validar los datos de entrada
        $validatedData = $request->validate([
            'nombre' => 'required|string|max:100',
            'apellido' => 'required|string|max:100',
            'cedula' => 'required|string|unique:doctors',
            'profesion' => 'required|string|max:35',
            'date' => 'required|date',
            'edad' => 'required|integer',
            'genero' => 'required|string|max:10',
            'email' => 'required|email|unique:doctors',
            'password' => 'required|string|min:6',
            'telefono' => 'nullable|string|max:15',
            'direccion' => 'required|string',
            'imagen' => 'nullable|string',
        ]);

        $data = $request->all();
```

```
    if ($request->hasFile('imagen')) {
        $file = $request->file('imagen');
        $filename = time() . '_' . $file->getClientOriginalName();
        $file->move(public_path('images'), $filename);
        $data['imagen'] = 'images/' . $filename;
    }

    $doctor = Doctors::create([
        'nombre' => $request->nombre,
        'apellido' => $request->apellido,
        'cedula' => $request->cedula,
        'profesion' => $request->profesion,
        'date' => $request->date,
        'edad' => $request->edad,
        'genero' => $request->genero,
        'email' => $request->email,
        'password' => $request->password,
        'telefono' => $request->telefono,
        'direccion' => $request->direccion,
        'imagen' => $request->imagen,
    ]);

    return response()->json($doctor, 201);
}

/**
 * Display the specified resource.
 */
public function show($id)
{
    try {
        // Buscar al doctor por ID
        $doctor = Doctors::find($id);

        // Retornar datos del doctor en formato JSON
        return response()->json([
            'success' => true,
            'data' => $doctor,
        ], 200);
    } catch (\Illuminate\Database\Eloquent\ModelNotFoundException $e) {
        // Manejar caso de no encontrar el doctor
        return response()->json([
            'success' => false,
            'message' => 'Doctor no encontrado',
        ], 404);
    } catch (\Exception $e) {
        // Manejar cualquier otro error
        return response()->json([
```

```
            'success' => false,
            'message' => 'Error al obtener los datos del doctor',
        ], 500);
    }
}

/**
 * Show the form for editing the specified resource.
 */
public function edit(Doctors $doctors)
{
    //
}

/**
 * Update the specified resource in storage.
 */
public function update(Request $request, Doctors $doctors)
{
    // Validar los datos de entrada
    $validatedData = $request->validate([
        'nombre' => 'nullable|string|max:255',
        'apellido' => 'nullable|string|max:255',
        'cedula' => 'nullable|string|unique:doctors,cedula,' . $doctors->id,
        'profesion' => 'nullable|string|max:255',
        'edad' => 'nullable|integer',
        'genero' => 'nullable|string|max:10',
        'email' => 'nullable|email|unique:doctors,email,' . $doctors->id,
        'password' => 'nullable|string|min:6',
        'telefono' => 'nullable|string|max:15',
        'direccion' => 'nullable|string',
        'imagen' => 'nullable|string',
    ]);

    // Actualizar el doctor
    if (isset($validatedData['password'])) {
        $validatedData['password'] = bcrypt($validatedData['password']);
    }

    $doctor->update($validatedData);

    return response()->json($doctors, 200);
}

/**
 * Remove the specified resource from storage.
 */
public function destroy($id)
```

```
    {
        $doctor = Doctors::find($id); // Busca el doctor por ID
        if (!$doctor) {
            return response()->json(['message' => 'Doctor no encontrado'], 404);
        }

        // Eliminar el doctor
        $doctor->delete();

        return response()->json(null, 204);
    }
}
```

CREACIÓN DEL MODELO

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

class Doctors extends Model
{
    use HasFactory;

    /**
     * Los campos que pueden asignarse masivamente.
     *
     * @var array<int, string>
     */
    // Nombre de la tabla (si no sigue la convención por defecto)
    protected $table = 'doctors';

    protected $fillable = [
        'nombre',
        'apellido',
        'cedula',
        'profesion',
        'date',
        'edad',
        'genero',
        'email',
        'password',
        'telefono',
        'direccion',
        'imagen',
    ];
```

```
}
```

Código del Paciente

CREACIÓN DE LA TABLA

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('patients', function (Blueprint $table) {
            $table->id(); // unsignedBigInteger
            $table->string('registration_number');
            $table->string('first_name');
            $table->string('last_name');
            $table->integer('age');
            $table->string('gender');
            $table->date('birth_date');
            $table->string('emergency_contact');
            $table->string('emergency_email');
            $table->string('address');
            $table->text('medical_history')->nullable();
            $table->text('allergies')->nullable();
            $table->string('description');
            $table->string('last_consultation');
            $table->unsignedBigInteger('doctor_id'); // Coincide con la clave
primaria en "doctors"
            $table->foreign('doctor_id')->references('id')->on('doctors')-
>onDelete('cascade');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('patients');
    }
```

```
};
```

OPERACIÓN DEL CONTROLADOR

```
<?php

namespace App\Http\Controllers;

use App\Models\Patient;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class PatientController extends Controller
{
    public function store(Request $request)
    {
        $validated = $request->validate([
            'registration_number' => 'required|string|unique:patients',
            'first_name' => 'required|string',
            'last_name' => 'required|string',
            'age' => 'required|integer',
            'gender' => 'required|string',
            'birth_date' => 'required|date',
            'emergency_contact' => 'required|string',
            'emergency_email' => 'required|email',
            'address' => 'required|string',
            'medical_history' => 'nullable|string',
            'allergies' => 'nullable|string',
            'description' => 'required|string',
            'last_consultation' => 'required|string',
            'doctor_id' => 'required|exists:doctors,id',
        ]);

        $patient = Patient::create($validated);

        return response()->json($patient, 201);
    }

    // Obtener un paciente por ID
    public function show($id)
    {
        // Buscar el paciente por el ID
        $patient = Patient::find($id);

        // Verificar si el paciente existe
        if (!$patient) {
            return response()->json([
                'success' => false,
```

```
                'message' => 'Paciente no encontrado'
            ], 404);  // Retorna un error 404 si no se encuentra
        }

        // Devolver los datos del paciente en formato JSON
        return response()->json([
            'success' => true,
            'patient' => $patient
        ], 200);
    }

    public function destroy($id)
    {
        $patient = Patient::find($id); // Busca el doctor por ID
        if (!$patient) {
            return response()->json(['message' => 'Paciente no encontrado'], 404);
        }

        // Eliminar el doctor
        $patient->delete();

        return response()->json(null, 204);
    }
}
```

CREACIÓN DEL MODELO

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Patient extends Model
{
    use HasFactory;

    /**
     * Campos que pueden ser asignados masivamente.
     */
    protected $fillable = [
        'registration_number', // Matricula
        'first_name',          // Nombre
        'last_name',           // Apellido
        'age',                 // Edad
        'gender',              // Género
        'birth_date',          // Fecha de Nacimiento
        'emergency_contact',   // Contacto de Emergencia
```

```
        'emergency_email',     // Email de Emergencia
        'address',             // Domicilio
        'medical_history',     // Antecedentes Patológicos
        'allergies',           // Alergias
        'description',         //Descripcion
        'last_consultation',   //Ultima Consulta
        'doctor_id',           // Relación con el doctor
    ];

    /**
     * Relación: un paciente pertenece a un doctor.
     */
    public function doctor()
    {
        return $this->belongsTo(Doctor::class);
    }
}
```

Control de acceso.

CÓDIGO AUTHCONTROLLER

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Doctors;
use App\Models\Patient;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;

class AuthController extends Controller
{
   // Método para autenticar al doctor
   public function loginDoctor(Request $request)
   {
        $request->validate([
            'email' => 'required|email',
            'password' => 'required|string',
        ]);
        // Buscar el usuario por correo electrónico
        $user = Doctors::where('email', $request->email)->first();

        // Verificar si el usuario existe y si la contraseña coincide con el hash
        if ($user && $request->password == $user->password) {
            return response()->json([
                'success' => true,
```

```
                'message' => 'Inicio de sesión exitoso',
                'user' => [
                    'id' => $user->id,
                    'nombre' => $user->nombre,  // Enviar el nombre del doctor
                    'apellido' => $user->apellido,
                    'email' => $user->email // Enviar el email si es necesario
                ]
            ], 200);
        }

        return response()->json(['message' => 'Correo o contraseña incorrectos'],
401);
    }

    // Método para obtener pacientes relacionados con un doctor
    public function getDoctorPatients($doctorId)
    {
        // Verificar si el doctor existe
        $doctor = Doctors::find($doctorId);

        if (!$doctor) {
            return response()->json(['message' => 'Doctor no encontrado'], 404);
        }

        // Obtener los pacientes relacionados
        $patients = Patient::where('doctor_id', $doctorId)->get();

        return response()->json([
            'success' => true,
            'doctor' => [
                'id' => $doctor->id,
                'nombre' => $doctor->nombre,
                'apellido' => $doctor->apellido,
                'email' => $doctor->email,
            ],
            'patients' => $patients
        ], 200);
    }

    // Método para cambiar la contraseña
    public function changePassword(Request $request)
    {
        // Validar los datos de entrada
        $validator = Validator::make($request->all(), [
            'correo' => 'required|email|exists:doctors,email',
            'password' => 'required|string|min:8',
        ]);
```

```
        if ($validator->fails()) {
            return response()->json([
                'success' => false,
                'errors' => $validator->errors()
            ], 422);
        }

        // Encontrar al doctor por correo
        $doctor = Doctors::where('email', $request->correo)->first();

        // Actualizar la contraseña encriptada
        $doctor->password = $request->password;
        $doctor->save();

        return response()->json([
            'success' => true,
            'message' => 'Contraseña actualizada con éxito.'
        ], 200);
    }
}
```

Ruta de conexión del Laravel con Angular

```
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\DoctorsController;
use App\Http\Controllers\PatientController;
use App\Http\Controllers\AuthController;

Route::get('/user', function (Request $request) {
    return $request->user();
})->middleware('auth:sanctum');

// DOCTORES
Route::get('/doctors', [DoctorsController::class, 'index']);
Route::post('/doctors', [DoctorsController::class, 'store']);
Route::get('/doctors{id}', [DoctorsController::class, 'show']);
Route::put('/doctors/{id}', [DoctorsController::class, 'update']);
Route::delete('/doctors/{id}', [DoctorsController::class, 'destroy']);
Route::resource('doctors', DoctorsController::class);
Route::post('/change-password', [AuthController::class, 'changePassword']); // Ruta
para cambiar la contraseña
Route::post('/validate-password', [AuthController::class, 'loginDoctor']);
// PACIENTES
Route::post('/add-patients', [PatientController::class, 'store']);
Route::get('/doctor{doctorId}/patients', [AuthController::class,
'getDoctorPatients']);
```

```
Route::get('patients/{id}', [PatientController::class, 'show']);
Route::delete('patients/{id}', [PatientController::class, 'destroy']);
Route::resource('patients', PatientController::class);
```

Captura del Código de Angular

Componentes creados para el proyecto

DIRECCIONAMIENTO PRINCIPAL

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from './Ventanas/login/login.component';
import { PrincipalComponent } from './Ventanas/principal/principal.component';
import { VistasComponent } from './Ventanas/vistas/vistas.component';
import { StructureComponent } from './Layout/structure/structure.component';
import { AuthGuard } from './guards/auth.guard';
import { LoginGuard } from './guards/login.guard';

const routes: Routes = [
  {
    path: 'home',
    component: StructureComponent,
    canActivate: [AuthGuard], // Protege la ruta con el guard
    children: [
      {
        path: '',
        loadChildren: () => import('./Ventanas/windows-routing.module').then(
          (m) => m.WindowsRoutingModule
        ),
      },
    ],
  },
  {
    path: 'login', // Ruta para el administrador
    component: LoginComponent,
    canActivate: [LoginGuard] // Evita el acceso a login si ya está autenticado
  },
  {
    path: '',
    redirectTo: '/login',
    pathMatch: 'full',
  },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

DIRECCIONAMIENTO SECUNDARIO

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterModule, Routes } from '@angular/router';

//Ad Guard para verificar Login
import { AuthGuard } from '../guards/auth.guard';

//Componentes del HTML
import { PrincipalComponent } from './principal/principal.component';
import { VistasComponent } from './vistas/vistas.component';
import { DiagnosticComponent } from './diagnostic/diagnostic.component';
import { AboutComponent } from './about/about.component';
import { Category1Component } from './Diagnosticos/Funcional/category1.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { ProfileMedicComponent } from './profile-medic/profile-medic.component';
import { EncuestaComponent } from './Encuesta/encuesta/encuesta.component';
import { ResultadosComponent } from './Encuesta/resultados/resultados.component';
import { LoginComponent } from './login/login.component';

import { EncuestaAfcComponent } from './Diagnosticos/Afectiva/encuesta-afc/encuesta-
afc.component';
import { EncuestaCESD7Component } from './Diagnosticos/Afectiva/encuesta-ces-
d7/encuesta-ces-d7.component';
import { EncuestaPhq9Component } from './Diagnosticos/Afectiva/encuesta-
phq9/encuesta-phq9.component';
import { EncuestaGaiSfComponent } from './Diagnosticos/Afectiva/encuesta-gai-
sf/encuesta-gai-sf.component';
import { EncuestaEscalaSoledadComponent } from './Diagnosticos/Afectiva/encuesta-
escala-soledad/encuesta-escala-soledad.component';
import { EncuestaSADPERSONSComponent } from './Diagnosticos/Afectiva/encuesta-sad-
persons/encuesta-sad-persons.component';
import { EncuestaCornellComponent } from './Diagnosticos/Afectiva/encuesta-
cornell/encuesta-cornell.component';
import { EncuestaOkeeffeComponent } from './Diagnosticos/Afectiva/encuesta-
okeeffe/encuesta-okeeffe.component';
import { EncuestaAssessmentSFComponent } from './Diagnosticos/Nutricional/encuesta-
assessment-sf/encuesta-assessment-sf.component';
import { EncuestaAssessmentComponent } from './Diagnosticos/Nutricional/encuesta-
assessment/encuesta-assessment.component';
import { EncuestaMustComponent } from './Diagnosticos/Nutricional/encuesta-
must/encuesta-must.component';
import { EncuestaSarcFComponent } from './Diagnosticos/Nutricional/encuesta-sarc-
f/encuesta-sarc-f.component';
//Componentes de la encuesta
import { EncuestaCogComponent } from './Diagnosticos/Cognitiva/encuesta-cog/encuesta-
cog.component';
```

```
import { EncuestaBeckAnxietyComponent } from './Diagnosticos/Afectiva/encuesta-beck-
anxiety/encuesta-beck-anxiety.component';
import { checkRedirectGuard } from '../guards/check-redirect.guard';
import { RegistroComponent } from './registro/registro.component';
import { ListDoctorsComponent } from './list-doctors/list-doctors.component';
import { PatientsComponent } from './patients/patients.component';
import { RegisterPatientComponent } from './register-patient/register-
patient.component';


const routes: Routes = [
  {
    path: '',
    redirectTo: 'home',
    pathMatch: 'full',
  },
  {
    path: '',
    component: PrincipalComponent,
  },
  {
    path: 'view-person',
    component: VistasComponent,
    canActivate: [AuthGuard]
  },
  {
    path: 'diagnostic',
    component: DiagnosticComponent,
    canActivate: [AuthGuard]
  },
  {
    path: 'list-person',
    component: PatientsComponent,
    canActivate: [AuthGuard]
  },
  {
    path: 'about',
    component: AboutComponent,
    canActivate: [AuthGuard]
  },
  {
    path:'dashboard',
    component: DashboardComponent,
    canActivate: [AuthGuard]
  },
  {
    path: 'profile',
    component: ProfileMedicComponent,
```

```
    canActivate: [AuthGuard]
  },
  {
    path: 'resultado',
    component: ResultadosComponent,
    canActivate: [AuthGuard]
  },
  {
    path: 'encuesta/:categoria',
    component: EncuestaComponent,
    canActivate: [checkRedirectGuard]
  },
  {
    path: 'doctors-register',
    component: RegistroComponent,
    canActivate: [AuthGuard]
  },
  {
    path: 'list-doctors',
    component: ListDoctorsComponent,
    canActivate: [AuthGuard]
  },
  {
    path: 'register-patient',
    component: RegisterPatientComponent,
    canActivate: [AuthGuard]
  },
  //Encuestas de los URL
  {
    path: 'encuesta-cog',
    component: EncuestaCogComponent,
    canActivate: [AuthGuard]
  },
  {
    path: 'encuesta-func',
    component: Category1Component,
    canActivate: [AuthGuard]
  },
  {
    path: 'encuesta-afc',
    component: EncuestaAfcComponent,
    canActivate: [AuthGuard]
  },
  {
    path: 'encuesta-ces-d7',
    component: EncuestaCESD7Component,
    canActivate: [AuthGuard]
  },
```

```
{
  path: 'encuesta-phq9',
  component: EncuestaPhq9Component,
  canActivate: [AuthGuard]
},
{
  path: 'encuesta-gai-sf',
  component: EncuestaGaiSfComponent,
  canActivate: [AuthGuard]
},
{
  path: 'encuesta-beck',
  component: EncuestaBeckAnxietyComponent,
  canActivate: [AuthGuard]
},
{
  path: 'encuesta-soledad',
  component: EncuestaEscalaSoledadComponent,
  canActivate: [AuthGuard]
},
{
  path: 'encuesta-Sad',
  component: EncuestaSADPERSONSComponent,
  canActivate: [AuthGuard]
},
{
  path: 'encuesta-cornell',
  component: EncuestaCornellComponent,
  canActivate: [AuthGuard]
},
{
  path: 'encuesta-okeeffe',
  component: EncuestaOkeeffeComponent,
  canActivate: [AuthGuard]
},
{
  path: 'encuesta-assessment-sf',
  component: EncuestaAssessmentSFComponent,
  canActivate: [AuthGuard]
},
{
  path: 'encuesta-asesment',
  component: EncuestaAssessmentComponent,
  canActivate: [AuthGuard]
},
{
  path: 'encuesta-must',
  component: EncuestaMustComponent,
```

```
    canActivate: [AuthGuard]
  },
  {
    path: 'encuesta-sarc-f',
    component: EncuestaSarcFComponent,
    canActivate: [AuthGuard]
  },
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class WindowsRoutingModule { }
```

Código del Angular Materia Utilizados

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatSidenavModule } from '@angular/material/sidenav';
import { MatButtonModule } from '@angular/material/button';
import { MatIconModule } from '@angular/material/icon';
import { MatDividerModule } from '@angular/material/divider';
import { MatListModule } from '@angular/material/list';
import {MatTableModule} from '@angular/material/table';
import {MatFormFieldModule} from '@angular/material/form-field';
import {MatPaginatorModule} from '@angular/material/paginator';

@NgModule({
  declarations: [],
  imports: [
    CommonModule
  ],
  //se exportan todos los componentes de Angular Material
  exports: [
    MatToolbarModule,
    MatSidenavModule,
    MatButtonModule,
    MatIconModule,
    MatDividerModule,
    MatListModule,
    MatTableModule,
    MatFormFieldModule,
    MatPaginatorModule
  ]
})
export class MaterialModuleModule { }
```

Código de todos los módulos del proyecto.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { ReactiveFormsModule } from '@angular/forms';
// import { MatListModule } from '@angular/material/list';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { LoginComponent } from './Ventanas/login/login.component';
import { PrincipalComponent } from './Ventanas/principal/principal.component';
import { FooterComponent } from './Layout/footer/footer.component';
import { VistasComponent } from './Ventanas/vistas/vistas.component';
import { StructureComponent } from './Layout/structure/structure.component';
import { WindowsRoutingModule } from './Ventanas/windows-routing.module';
import { DiagnosticComponent } from './Ventanas/diagnostic/diagnostic.component';
import { AboutComponent } from './Ventanas/about/about.component';
import { Category1Component } from
'./Ventanas/Diagnosticos/Funcional/category1.component';
import { EncuestaComponent } from './Ventanas/Encuesta/encuesta/encuesta.component';
import { EncuestaAssessmentSFComponent } from
'./Ventanas/Diagnosticos/Nutricional/encuesta-assessment-sf/encuesta-assessment-
sf.component';
import { EncuestaAssessmentComponent } from
'./Ventanas/Diagnosticos/Nutricional/encuesta-assessment/encuesta-
assessment.component';

//Incorporacion de Fecha
import { LOCALE_ID } from '@angular/core';
import { registerLocaleData } from '@angular/common';
import  localeEs  from '@angular/common/locales/es';
import { DashboardComponent } from './Ventanas/dashboard/dashboard.component';
import { ProfileMedicComponent } from './Ventanas/profile-medic/profile-
medic.component'

//Conexio  Http
import { HttpClient, HttpClientModule } from '@angular/common/http';
import { ResultadosComponent } from
'./Ventanas/Encuesta/resultados/resultados.component';
import { EncuestaCogComponent } from './Ventanas/Diagnosticos/Cognitiva/encuesta-
cog/encuesta-cog.component';
import { EncuestaAfcComponent } from './Ventanas/Diagnosticos/Afectiva/encuesta-
afc/encuesta-afc.component';
import { EncuestaCESD7Component } from './Ventanas/Diagnosticos/Afectiva/encuesta-
ces-d7/encuesta-ces-d7.component';
import { EncuestaPhq9Component } from './Ventanas/Diagnosticos/Afectiva/encuesta-
phq9/encuesta-phq9.component';
```

```
import { EncuestaGaiSfComponent } from './Ventanas/Diagnosticos/Afectiva/encuesta-
gai-sf/encuesta-gai-sf.component';
import { EncuestaBeckAnxietyComponent } from
'./Ventanas/Diagnosticos/Afectiva/encuesta-beck-anxiety/encuesta-beck-
anxiety.component';
import { EncuestaEscalaSoledadComponent } from
'./Ventanas/Diagnosticos/Afectiva/encuesta-escala-soledad/encuesta-escala-
soledad.component';
import { EncuestaSADPERSONSComponent } from
'./Ventanas/Diagnosticos/Afectiva/encuesta-sad-persons/encuesta-sad-
persons.component';
import { EncuestaCornellComponent } from './Ventanas/Diagnosticos/Afectiva/encuesta-
cornell/encuesta-cornell.component';
import { EncuestaOkeeffeComponent } from './Ventanas/Diagnosticos/Afectiva/encuesta-
okeeffe/encuesta-okeeffe.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { RegistroComponent } from './Ventanas/registro/registro.component';
import { MaterialModuleModule } from './material.module';
import { ListDoctorsComponent } from './Ventanas/list-doctors/list-
doctors.component';
import { PatientsComponent } from './Ventanas/patients/patients.component';
import { RegisterPatientComponent } from './Ventanas/register-patient/register-
patient.component';
import { EncuestaMustComponent } from './Ventanas/Diagnosticos/Nutricional/encuesta-
must/encuesta-must.component';
import { EncuestaSarcFComponent } from './Ventanas/Diagnosticos/Nutricional/encuesta-
sarc-f/encuesta-sarc-f.component';
//import { NotFoundComponent } from './Ventanas/not-found/not-found.component';

//Registro de Fecha
registerLocaleData(localeEs, 'es');

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    PrincipalComponent,
    FooterComponent,
    VistasComponent,
    StructureComponent,
    DiagnosticComponent,
    AboutComponent,
    Category1Component,
    DashboardComponent,
    ProfileMedicComponent,
    EncuestaComponent,
    ResultadosComponent,
    EncuestaCogComponent,
```

```
    EncuestaAfcComponent,
    EncuestaCESD7Component,
    EncuestaPhq9Component,
    EncuestaGaiSfComponent,
    EncuestaBeckAnxietyComponent,
    EncuestaEscalaSoledadComponent,
    EncuestaSADPERSONSComponent,
    EncuestaCornellComponent,
    EncuestaOkeeffeComponent,
    RegistroComponent,
    ListDoctorsComponent,
    PatientsComponent,
    RegisterPatientComponent,
    EncuestaAssessmentSFComponent,
    EncuestaAssessmentComponent,
    EncuestaMustComponent,
    EncuestaSarcFComponent,
    //NotFoundComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    WindowsRoutingModule,
    HttpClientModule,
    BrowserAnimationsModule,
    MaterialModuleModule
  ],
  providers: [ { provide: LOCALE_ID, useValue: 'es' } ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Código de control de acceso a rutas.

RUTA PROTEGIDA POR LOGIN

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { UserService } from '../Service/user.service';

@Injectable({
  providedIn: 'root'
})
export class LoginGuard implements CanActivate {

  constructor(private userService: UserService, private router: Router) {}
```

```
  canActivate(): boolean {
    if (this.userService.isAuthenticated()) {
      // Si el usuario ya está autenticado, redirige a home
      this.router.navigate(['home']);
      return false;
    }
    return true;
  }
}
```

RUTA PROTEGIDA POR ACCESO AL AUTENTICADO

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { UserService } from '../Service/user.service';

@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {

  constructor(private userService: UserService, private router: Router) {}

  canActivate(): boolean {

    if (!this.userService.isAuthenticated()) {
      this.router.navigate(['/login']);  //Redirige al login si no está autenticado
      //this.router.navigate(['/not-found']); //Denegar Acceso si no esta logueado
      return false;
    }
    return true;
  }
}
```

RUTA POR ACCESO DENEGADO.

```
import { Injectable } from '@angular/core';
import { CanActivate, CanActivateFn, Router } from '@angular/router';

@Injectable({
  providedIn: 'root'
})

export class checkRedirectGuard implements CanActivate{
  constructor(private router: Router) {}

  canActivate(): boolean {
    // Aquí decides si debe acceder o ser redirigido
```

```
    //const canAccessEncuesta = this.checkAccessToEncuesta(); // Implementa
tu lógica
    const canAccessEncuesta = localStorage.getItem('subCatSeleccionada');

    if (!canAccessEncuesta) {
      // Redirige al componente de categoría
      this.router.navigate(['/categoria']);
      return false; // Bloquea acceso al componente encuesta
    }
    return true; // Permite acceso
  }
}
```

Estructura que conforma la página

ESTRUCTURA HTML

```
<mat-toolbar>
    <!-- Botón de menú -->
    <button mat-icon-button *ngIf="sidenav.mode ==='over'"
(click)="sidenav.toggle()">
        <mat-icon *ngIf="!sidenav.opened">menu</mat-icon>
        <mat-icon *ngIf="sidenav.opened">close</mat-icon>
    </button>

    <!-- Logo -->
    <div class="logo" (click)="mainPage()" title="VGI Home">
        <img src="assets/Imagenes/icon-logo.png" title="VGI" alt="" class="icono">
        <h2>VGI</h2>
    </div>

    <!-- Espaciador -->
    <div class="spacer"></div>

    <!-- Perfil -->
    <div class="profile">
        <img
src="https://i.pinimg.com/1200x/88/f4/8f/88f48f94c2b60bb17a009a1513c53716.jpg"
alt="Profile Image" class="profile-image" onclick="toggleDropdown()">
        <div id="dropdownMenu" class="dropdown-menu">
            <span class="user-name">{{name}}</span> <!-- Nombre del usuario -->
            <a routerLink="profile">Profile<i class="fas fa-user-circle"></i></a>
            <a routerLink="#">Settings<i class="fas fa-cog"></i></a>
            <a routerLink="#" (click)="logout()">Logout<i class="fas fa-sign-out-
alt"></i></a>
        </div>
    </div>
</mat-toolbar>

<!--Sidenav-->
<mat-sidenav-container>
    <mat-sidenav>
        <mat-nav-list>
            <mat-list-item (click)="sidenav.mode === 'over' && sidenav.toggle()"
routerLink="/">
                <mat-icon matListItemIcon>home</mat-icon>
                Home
            </mat-list-item>
            <mat-list-item (click)="sidenav.mode === 'over' && sidenav.toggle()"
routerLink="list-doctors" *ngIf="doctorID == 1">
                <mat-icon matListItemIcon>person</mat-icon>
                Doctores
```

```
            </mat-list-item>
            <mat-list-item (click)="sidenav.mode === 'over' && sidenav.toggle()"
routerLink="list-person">
                <mat-icon matListItemIcon>folder_shared</mat-icon>
                Pacientes
            </mat-list-item>
            <mat-list-item (click)="sidenav.mode === 'over' && sidenav.toggle()"
routerLink="diagnostic">
                <mat-icon matListItemIcon>ballot</mat-icon>
                Encuesta
            </mat-list-item>
            <mat-list-item (click)="sidenav.mode === 'over' && sidenav.toggle()"
routerLink="about">
                <mat-icon matListItemIcon>info</mat-icon>
                Acerca de
            </mat-list-item>
        </mat-nav-list>
    </mat-sidenav>
    <mat-sidenav-content>
        <div class="content">
            <router-outlet></router-outlet>
        </div>
    </mat-sidenav-content>
</mat-sidenav-container>
```

ESTRUCTURA SCSS

```
mat-toolbar {
  background-color: #f06cff;
  color: #ffffff;
  display: flex;
  align-items: center;
}

mat-sidenav {
  width: 200px;
  border-right: none;
  padding: 16px 0;
}

.content {
  height: calc(100svh - 98px);
  border-radius: 10px;
  padding: 16px;
  max-width: 1520px;
  margin: 0 auto;
  font-size: 16px;
}
mat-sidenav-container {
```

```
  height: calc(100svh - 65px);
}

/* Espaciador para separar elementos */
.spacer {
    flex: 1 1 auto; /* Ocupa el espacio restante */
}

// LOGO DE LA APLICACION
.logo .icono {
    background: white;
    border-radius: 50%;
    width: 100%;
    max-width: 40px;
}
.logo:hover{
    cursor: pointer;
}
.logo .icono, .logo h2{
    display: inline;
}
.logo h2{
    padding-left: 10px;
}
// IMAGEN Y USUARIO [LOGIN]
.profile {
    position: relative;
    //width: 100px;
}
.user-name {
    color: #000;
    font-size: 15px;
    font-weight: 400;
    display: inline-flex;
    margin-left: 10px;
}
.profile-image {
    width: 40px;
    height: 40px;
    border-radius: 50%;
    cursor: pointer;
}

.dropdown-menu {
    position: absolute;
    top: 50px;
    right: 0;
    background-color: #fff;
```

```
    box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
    border-radius: 5px;
    overflow: hidden;
    display: none;
    padding-right: 10px;
}

.dropdown-menu a {
    display: block;
    padding: 10px;
    color: #333;
    text-decoration: none;
    text-align: right;
    i{
        margin-left: 10px;
    }
}
.profile a:hover{
    background: rgba(0,250,238,0.3);
}

.dropdown-menu.show {
    display: block;
}
```

ESTRUCTURA TYPESCRIPT

```
import { Component, OnInit, ViewChild } from '@angular/core';
import { MatSidenav } from '@angular/material/sidenav';
import { BreakpointObserver } from '@angular/cdk/layout';
import { Router } from '@angular/router';
import { NavbarService } from 'src/app/Service/navbar.service';
import { UserService } from 'src/app/Service/user.service';
import { SharedService } from 'src/app/Service/shared.service';
import { LoadJSService } from 'src/app/Service/load-js.service';

@Component({
  selector: 'app-structure',
  templateUrl: './structure.component.html',
  styleUrls: ['./structure.component.scss']
})
export class StructureComponent {
  @ViewChild(MatSidenav, { static: true })
  sidenav!: MatSidenav;

  dropdownOpen = false;
  showRegisterButton: boolean = false;
  name: string = '';
  doctorID = 0; //Determinar el ID del Doctor
```

```
  constructor(private observer: BreakpointObserver, private router: Router, private
authService: NavbarService, private userService: UserService, private sharedService:
SharedService, private LoadJS: LoadJSService){
    LoadJS.Carga(["Profile"]);
  }

  ngOnInit(): void{
    this.observer.observe(["(max-width: 720px)"])
      .subscribe((res) => {
        if (res.matches) {
          this.sidenav.mode = "over";
          this.sidenav.close();
        } else {
          this.sidenav.mode = "side";
          this.sidenav.open();
        }
      });
      this.name = 'Dr. ' + this.userService.getDoctorName();
      this.sharedService.showRegisterButton$.subscribe(show => {
        this.showRegisterButton = show;
      });
    this.doctorID = this.userService.getDoctorId(); //Solo el Doctor Master puede
registrar Doctores
  }

  mainPage(){
    this.router.navigate(['home']);
  }

  login() {
    this.authService.login();
    this.router.navigate(['/home/principal']); // Redirige a la página principal
  }

  logout() {
    this.userService.logout(); // Llama al método logout del servicio de usuario
    this.router.navigate(['/login']); // Redirige al login después de cerrar sesión
  }
}
```

Servicios que se implementaron

ALERTA PERSONALIZADA

```
import { Injectable } from '@angular/core';
import Swal from 'sweetalert2';

@Injectable({
```

```
  providedIn: 'root'
})
export class AlertService {

  constructor() { }

  // Método para mostrar un mensaje de éxito
  success(message: string, title: string) {
    Swal.fire({
        title: title,
        text: message,
        icon: 'success',
        confirmButtonText: 'Aceptar',
    });
  }

  // Método para mostrar un mensaje de error
  error(message: string, title: string) {
    Swal.fire({
        title: title,
        text: message,
        icon: 'error',
        confirmButtonText: 'Aceptar',
    });
  }

  // Método para mostrar una alerta de confirmación
  confirm(
    message: string,
    title: string = 'Confirmación',
    confirmButtonText: string = 'Sí',
    cancelButtonText: string = 'No'
): Promise<boolean> {
    return Swal.fire({
        title: title,
        text: message,
        icon: 'warning',
        showCancelButton: true,
        confirmButtonText: confirmButtonText,
        cancelButtonText: cancelButtonText,
    }).then((result) => result.isConfirmed);
  }

  // Método para mostrar una alerta de advertencia
  warning(message: string, title: string){
      Swal.fire({
        title: title,
        icon: 'warning',
```

```
        text: message,
        confirmButtonText: 'Aceptar'
      });
  }
}
```

CARGA JAVSCRIPT EN ALGUNOS COMPONENTES

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class LoadJSService {

  constructor() { }

  Carga(archivos:string[]){
    for(let archivo of archivos){
      let script = document.createElement("script");
      script.src = "./assets/JavaScript/" + archivo + ".js";
      let body = document.getElementsByTagName("body")[0];
      body.appendChild(script);
    }
  }
}
```

VERIFICACIÓN DEL LOGIN

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class NavbarService {
  public isAuthenticated : boolean = false;

  constructor() {}

  login() {
    this.isAuthenticated  = true;
  }

  logout() {
    this.isAuthenticated  = false;
  }

  isLoggedOut(){
    return this.isAuthenticated;
  }
}
```

CONEXIÓN CON EL COMPONENTE RESULTADO

```
import { Injectable } from '@angular/core';
import { Resultado } from '../Shared/Data';
import { BehaviorSubject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ResultadoService {
  private resultado = new BehaviorSubject<Resultado | null>(null);
  resultado$ = this.resultado.asObservable();

  constructor() {

   }

   setResultado(resultado: Resultado){
    this.resultado.next(resultado);
   }
}
```

CONEXIÓN CON LARAVEL Y ASIGNACIÓN URL CATEGORÍA

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http'
import { BehaviorSubject, Observable, Subject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ApiService {
  private urlApi = 'http://127.0.0.1:8000/api'; // Api Laravel (Backend)

  /* SECCION CATEGORIA Y SUBCATEGORIA */
  //Selecciona Subcategoria
  private selectSubCat = new BehaviorSubject<String>('Undefinid');
  selectEncuest$: Observable<String>;
  //Selecciona la Categoria
  private categoriaSeleccionadaSource: BehaviorSubject<String> = new
BehaviorSubject<String>('Undefinid');
  categoriaSeleccionada$: Observable<String>;

  constructor(private http: HttpClient) {
    // Recuperar la categoría/encuesta seleccionada del Local Storage o establecer un
valor predeterminado
    const categoriaGuardada = localStorage.getItem('categoriaSeleccionada') ||
'undefinid';
    const encuestaGuardada = localStorage.getItem('encuestaSeleccionada') ||
'undefined';
    // Inicializar BehaviorSubject con el valor recuperado
```

```
    this.categoriaSeleccionadaSource = new
BehaviorSubject<String>(categoriaGuardada);
    this.selectSubCat = new BehaviorSubject<String>(encuestaGuardada);
    // Asignar el observable a una propiedad pública
    this.categoriaSeleccionada$ = this.categoriaSeleccionadaSource.asObservable();
    this.selectEncuest$ = this.selectSubCat.asObservable();
  }
  //Seleccion de Categoria y Encuesta
  seleccionarEncuesta(encuesta: String){
    this.selectSubCat.next(encuesta);
    localStorage.setItem('encuestaSeleccionada', encuesta.toString());
  }
  seleccionarCategoria(categoria: string){
    // Actualizar la categoría y persistirla en Local Storage
    this.categoriaSeleccionadaSource.next(categoria);
    localStorage.setItem('categoriaSeleccionada', categoria.toString());
  }

  /* SECCION DE LARAVEL DE API */

  // Registrar un nuevo doctor
  registerDoctor(data: FormData): Observable<any> {
    return this.http.post<any>(`${this.urlApi}/doctors`, data);
    // return this.http.post<any>(this.urlApi,data);
  }
  // Login del doctor
  loginDoctor(credentials: any): Observable<any> {
    return this.http.post<any>(this.urlApi+'/validate-password', credentials);
  }
  //Obtener los registros Tabla Doctor
  getUsers(): Observable<any> {
    return this.http.get(this.urlApi+'/doctors');
  }
  //Eliminar Doctor
  deleteDoctor(id: number): Observable<any> {
    return this.http.delete(`${this.urlApi}/doctors/${id}`);
  }
  //Obtener Doctor por id
  getDoctorId(id: number): Observable<any> {
    return this.http.get(`${this.urlApi}/doctors${id}`);
  }
  // Método para cambiar la contraseña
  changePassword(payload: { correo: string, password: string }): Observable<any> {
    return this.http.post(`${this.urlApi}/change-password`, payload);
  }
  /* -------------------------- PACIENTE ----------------------------------- */
  //Crear Paciente
  crearPaciente(data: any): Observable<any> {
    return this.http.post<any>(`${this.urlApi}/add-patients`, data);
  }
  // En ApiService
  getPatientsByDoctor(doctorId: number): Observable<any> {
```

```
    return this.http.get<any>(`${this.urlApi}/doctor${doctorId}/patients`);
  }
  // Obtiene el paciente por id
  getPatientById(id: number): Observable<any> {
    return this.http.get(`${this.urlApi}/patients/${id}`);
  }
  //Eliminar Paciente
  deletePatient(patientId: number): Observable<any> {
    return this.http.delete<any>(`${this.urlApi}/patients/${patientId}`);
  }
}
```

VALIDACIÓN DEL INICIO DE SESIÓN

```
import { Injectable } from '@angular/core';
import Swal from 'sweetalert2';
import { ApiService } from './api.service';
import { SharedService } from './shared.service';

@Injectable({
  providedIn: 'root'
})
export class UserService {
  private apiService?: ApiService;
  showButton: boolean = false;
  usuario: string = '';

  isLoggedIn() {
    throw new Error('Method not implemented.');
  }
  private validar = {
    id: 1,
    email: 'prueba_auth123@hotmail.com',
    password: '12345'
  };

  constructor(private serviApi: ApiService, private sharedService: SharedService) { }

  logout(): void {
    localStorage.removeItem('isAuthenticated'); // Borra el token para cerrar sesión
    localStorage.removeItem('doctorName'); // Eliminar Nombre
    localStorage.removeItem('idDoctor'); //Emilinar el ID
  }

  isAuthenticated(): boolean {
    return !!localStorage.getItem('isAuthenticated'); // Verifica si el token está en
localStorage
  }
    // Verifica si el correo ya está registrado en la base de datos
    return !registeredUsers.some(user => user.email === email);
  }
```

```
  // LOGIN DE SUPER USER Y REVISION DE BASE DE DATOS

  async log(email: string, password: string): Promise<boolean> {
    try {

      // Busca un usuario con el correo y contraseña proporcionados
      const response: any = await this.serviApi.loginDoctor({ email, password
}).toPromise();

      if (response.success) {
        this.usuario = response.user.nombre + ' ' + response.user.apellido;
        localStorage.setItem('isAuthenticated', 'logged_in');
        localStorage.setItem('doctorName', this.usuario); // Guardar el nombre del
doctor
        localStorage.setItem('idDoctor', response.user.id.toString());
        this.sharedService.setShowRegisterButton(false);
        return true;
      }

      // Si no coincide con ninguno, muestra error
      Swal.fire({
        icon: 'error',
        title: 'Error',
        text: 'Correo electrónico y/o contraseña incorrectos',
        confirmButtonText: 'Aceptar',
      });

      return false;
    } catch (error) {
      console.error('Error al obtener usuarios:', error);
      Swal.fire({
        icon: 'error',
        title: 'Error',
        text: 'No se pudo conectar al servidor',
        confirmButtonText: 'Aceptar',
      });
      return false;
    }
  }

  getDoctorName(): string {
    return localStorage.getItem('doctorName') || '';
  }
  getDoctorId(): number {
    const id = localStorage.getItem('idDoctor');
    return id ? parseInt(id, 10) : 0; // Convierte a número o devuelve 0 si no está
definido
  }
}
```

Código de Interfaz utilizados

INTERFAZ DE LAS ENCUESTAS

```
export interface MedicalHistory {
    id: number;
    doctor: string;
    paciente: string;
    diagnostico: string;
    resultado: number;
    observaciones: string;
    fechaCaptura: string;
    hora: string;
    centroMedico: string;
  }

export interface Resultado {
    nombre: string;
    puntaje: number;
    observacion: string;
}

export interface categoria{
  name: String;
}

export interface PreguntaDosOpc{
  texto: String;
  respuesta: String;
}

// OPCIONES Y RESULTADO DE LA ENCUESTA [PRUEBA DE RELOJ] PUNTAJE Y PREGUNTAS
export interface Option{
  label: String;
  score: number;
}
export interface Question{
  text: String;
  options: Option[];
}
// OPCOINES BOOLEANAS
export interface Checkbox{
  text: String;
  seleccionada: boolean;
}
```

EJEMPLO DE PACIENTES

```
export interface Patient {
  id: number; // Identificador único del paciente.
```

```
  first_name: string; // Nombre del paciente.
  last_first_name: string; // Apellido del paciente.
  age: number; // Edad del paciente en años.
  gender: string; // Género del paciente. Ejemplo: "Femenino" o "Masculino".
  pulse: number; // Pulso cardíaco del paciente (latidos por minuto).
  blood_pressure: string; // Presión arterial del paciente. Ejemplo: "120/80".
  weight: number; // Peso del paciente en kilogramos (kg).
  height: number; // Altura del paciente en metros (m). Ejemplo: 1.75.
  bmi: string; // Índice de Masa Corporal (IMC), calculado con el peso y la altura.
Ejemplo: "22.3 (normal)".
  allergies: string; // Alergias conocidas del paciente. Ejemplo: "Alergia a la
penicilina".
  diagnosis: string; // Diagnóstico principal del paciente. Ejemplo: "Hipertensión
arterial".
  treatment: string; // Tratamiento prescrito al paciente. Ejemplo: "Enalapril 10 mg
diarios".
  last_consultation: string; // Fecha de la última consulta médica. Ejemplo: "2024-
11-10".
  emergency_contact: string; // Teléfono del contacto de emergencia. Ejemplo: "555-
123-4567".
  doctor_notes: string; // Notas del médico sobre el paciente. Ejemplo: "Controlar
dieta para mejorar el manejo de la diabetes".
  description: string; // Breve descripción del estado o historial del paciente.
Ejemplo: "Paciente con historial de hipertensión controlada".

}

export const PATIENTS_DATA: Patient[] = [
  {
    id: 1,
    first_name: 'María',
    last_first_name: 'González',
    age: 78,
    gender: 'Femenino',
    pulse: 72,
    blood_pressure: '130/85',
    weight: 65,
    height: 1.55,
    bmi: '27.1 (sobrepeso)',
    allergies: 'Ninguna conocida',
    diagnosis: 'Hipertensión arterial',
    treatment: 'Enalapril 10 mg diarios',
    last_consultation: '2024-10-15',
    emergency_contact: '555-123-4567',
    doctor_notes: 'Controlar dieta y reducir consumo de sal.',
    description:
      'Paciente con historial de hipertensión controlada, sin complicaciones
recientes.',
  },
  {
    id: 2,
    first_name: 'José',
```

```
    last_first_name: 'López',
    age: 82,
    gender: 'Masculino',
    pulse: 68,
    blood_pressure: '120/80',
    weight: 70,
    height: 1.68,
    bmi: '24.8 (normal)',
    allergies: 'Ibuprofeno',
    diagnosis: 'Osteoartritis',
    treatment: 'Paracetamol 500 mg c/8 hrs',
    last_consultation: '2024-09-30',
    emergency_contact: '555-987-6543',
    doctor_notes: 'Recomendar ejercicios para fortalecer articulaciones.',
    description:
      'Presenta dolor en las articulaciones, principalmente en las rodillas.
Movilidad moderada.',
  },
  {
    id: 3,
    first_name: 'Carmen',
    last_first_name: 'Ramírez',
    age: 85,
    gender: 'Femenino',
    pulse: 75,
    blood_pressure: '125/78',
    weight: 60,
    height: 1.50,
    bmi: '26.7 (sobrepeso)',
    allergies: 'Penicilina',
    diagnosis: 'Demencia senil',
    treatment: 'Rivastigmina 4.5 mg/día',
    last_consultation: '2024-10-05',
    emergency_contact: '555-456-7890',
    doctor_notes: 'Supervisar actividades diarias y medicamentos.',
    description:
      'Declive cognitivo progresivo. Requiere supervisión constante para actividades
diarias.',
  },
  {
    id: 4,
    first_name: 'Juan',
    last_first_name: 'Martínez',
    age: 76,
    gender: 'Masculino',
    pulse: 80,
    blood_pressure: '135/90',
    weight: 85,
    height: 1.75,
    bmi: '27.8 (sobrepeso)',
    allergies: 'Ninguna conocida',
    diagnosis: 'Diabetes tipo 2',
```

```
    treatment: 'Metformina 850 mg c/12 hrs',
    last_consultation: '2024-11-01',
    emergency_contact: '555-333-2222',
    doctor_notes: 'Control metabólico debe mejorar. Revisar dieta.',
    description:
      'Control metabólico moderado. Necesita monitoreo constante de glucosa.',
  },
  {
    id: 5,
    first_name: 'Rosalía',
    last_first_name: 'Pérez',
    age: 79,
    gender: 'Femenino',
    pulse: 70,
    blood_pressure: '118/76',
    weight: 62,
    height: 1.60,
    bmi: '24.2 (normal)',
    allergies: 'Mariscos',
    diagnosis: 'Insuficiencia cardíaca',
    treatment: 'Bisoprolol 5 mg/día',
    last_consultation: '2024-10-20',
    emergency_contact: '555-111-2233',
    doctor_notes: 'Evitar esfuerzos físicos prolongados.',
    description:
      'Historial de hospitalización reciente. Requiere control de líquidos y
reposo.',
  },
  {
    id: 6,
    first_name: 'Miguel',
    last_first_name: 'Hernández',
    age: 81,
    gender: 'Masculino',
    pulse: 65,
    blood_pressure: '110/70',
    weight: 68,
    height: 1.70,
    bmi: '23.5 (normal)',
    allergies: 'Aspirina',
    diagnosis: 'Parkinson',
    treatment: 'Levodopa 200 mg c/8 hrs',
    last_consultation: '2024-11-10',
    emergency_contact: '555-444-5555',
    doctor_notes: 'Recomendar terapia física para mejorar movilidad.',
    description:
      'Temblor en reposo y rigidez muscular. Necesita apoyo para caminar largas
distancias.',
  },
  {
    id: 7,
    first_name: 'Antonia',
```

```
    last_first_name: 'García',
    age: 84,
    gender: 'Femenino',
    pulse: 77,
    blood_pressure: '140/88',
    weight: 58,
    height: 1.55,
    bmi: '24.1 (normal)',
    allergies: 'Ninguna conocida',
    diagnosis: 'Insuficiencia renal crónica',
    treatment: 'Hemodiálisis semanal',
    last_consultation: '2024-10-25',
    emergency_contact: '555-666-7777',
    doctor_notes: 'Monitorear niveles de potasio en sangre.',
    description:
      'Bajo tratamiento de hemodiálisis con restricciones alimenticias estrictas.',
  },
  {
    id: 8,
    first_name: 'Raúl',
    last_first_name: 'Sánchez',
    age: 80,
    gender: 'Masculino',
    pulse: 73,
    blood_pressure: '125/80',
    weight: 75,
    height: 1.72,
    bmi: '25.4 (sobrepeso leve)',
    allergies: 'Polen',
    diagnosis: 'Cataratas',
    treatment: 'Pendiente de cirugía oftalmológica',
    last_consultation: '2024-11-12',
    emergency_contact: '555-888-9999',
    doctor_notes: 'Evaluar aptitud para cirugía.',
    description:
      'Visión borrosa y dificultad para realizar actividades que requieran enfoque
visual.',
  },
  {
    id: 9,
    first_name: 'Laura',
    last_first_name: 'Ortiz',
    age: 77,
    gender: 'Femenino',
    pulse: 74,
    blood_pressure: '130/82',
    weight: 65,
    height: 1.58,
    bmi: '26.0 (sobrepeso)',
    allergies: 'Gluten',
    diagnosis: 'Fractura de cadera',
    treatment: 'Rehabilitación física diaria',
```

```
    last_consultation: '2024-11-09',
    emergency_contact: '555-999-8888',
    doctor_notes: 'Evaluar progreso de movilidad semanalmente.',
    description:
      'En recuperación tras cirugía de reemplazo de cadera. Movilidad reducida.',
  },
  {
    id: 10,
    first_name: 'Francisco',
    last_first_name: 'Domínguez',
    age: 83,
    gender: 'Masculino',
    pulse: 69,
    blood_pressure: '115/75',
    weight: 72,
    height: 1.65,
    bmi: '26.4 (sobrepeso)',
    allergies: 'Sulfas',
    diagnosis: 'Alzheimer',
    treatment: 'Memantina 10 mg/día',
    last_consultation: '2024-10-28',
    emergency_contact: '555-222-3333',
    doctor_notes: 'Reforzar actividades cognitivas diarias.',
    description:
      'Avance gradual de pérdida de memoria y confusión. Requiere ayuda constante.',
  },
];
```

Código del Inicio de Sesión

ESTRUCTURA HMTL

```
<div class="container-fluid">
  <div class="row justify-content-center">
    <div class="col-md-6">
      <div class="card mt-5">
        <div class="card-body">
          <h3 class="card-title text-center">Iniciar Sesión</h3>
          <div class="text-center">
            <img
src="https://static.vecteezy.com/system/resources/thumbnails/020/788/814/small_2x/ico
n-symbol-or-website-admin-social-login-element-concept-3d-rendering-png.png"
alt="Imagen de Inicio de Sesión" class="img-fluid mb-3">
          </div>
          <form (ngSubmit)="onSubmit()" #loginForm="ngForm">
            <div class="form-group">
              <label for="email">Correo Electrónico</label>
              <input
                type="email"
                class="form-control"
                id="email"
                name="email"
```

```
                required
                [(ngModel)]="email"
              />
            </div>
            <div class="form-group">
              <label for="password">Contraseña</label>
              <input
                type="password"
                class="form-control"
                id="password"
                name="password"
                required
                [(ngModel)]="password"
              />
            </div>
            <button type="submit" class="btn btn-info btn-block">Ingresar</button>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>
```

ESTRUCTURA SCSS

```
.container {
    margin-top: 100px;
  }
  .card {
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  }
  .btn-block {
    width: 100%;
  }
  .btn{
    margin-top: 15px;
    color: #FFFFFF;
    font-style: italic;
    font-weight: bold;
  }
  .text-center img {
    max-width: 150px;
    width: 50%;
  }

  .container-fluid {
    height: 100vh;
    background-image:
url('https://static.vecteezy.com/system/resources/previews/002/104/158/non_2x/healthc
are-medical-technology-and-science-wallpaper-template-illustration-vector.jpg');
    background-size: cover;
    background-position: center;
  }
```

ESTRUCTURA TYPESCRIPT

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { ApiService } from 'src/app/Service/api.service';
import { SharedService } from 'src/app/Service/shared.service';
import { UserService } from 'src/app/Service/user.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})
export class LoginComponent implements OnInit{
  email: string = '';
  password: string = '';
  errorMessage: string = '';

  constructor(private userService: UserService, private router: Router, private
apiService: ApiService, private sharedService: SharedService) {}

  ngOnInit(): void {

  }
    async onSubmit() {
      const isLoggedIn = await this.userService.log(this.email, this.password);

      if (isLoggedIn) {
        //this.sharedService.setShowRegisterButton(false);
        this.updateRegisterButtonVisibility();
        this.router.navigate(['home'], { replaceUrl: true });
      } else {
        this.errorMessage = 'Correo electrónico o contraseña incorrectos';
      }
    }

    updateRegisterButtonVisibility(): void {
      this.apiService.getUsers().subscribe((users: any[]) => {
        // Determina si se muestra el botón usando UserService
        const showButton = this.userService.canShowRegisterButton(this.email, users);
        // Actualiza el estado en el servicio compartido
        this.sharedService.setShowRegisterButton(showButton);
      });
    }
}
```

Código de la ventana Principal

ESTRUCTURA HTML

```
<div class="container">
    <div class="fecha-hora">
        <div class="reloj-container">
            <p class="hours">Hora Actual:</p>
            <div class="reloj">
              {{ horaActual | date:'HH:mm:ss' }}
            </div>
        </div>
        <div class="container-fecha">
            <p class="date">{{ fecha | date: 'EEEE, dd/MM/yyyy' }}</p>
        </div>
    </div>

    <div class="dashboard">
        <h1>Bienvenido al Tablero Principal</h1>
        <div class="statistics row">
            <div class="stat-box1 col">
                <img src="assets/icons/Logo_IMSS.png" alt="Home Icon" width="50"
height="50">
                <p>1020</p>
                <span>Citas Hechas</span>
            </div>
            <div class="stat-box2 col">
                <img src="assets/icons/Logo_IMSS.png" alt="Home Icon" width="50"
height="50">
                <p>2834</p>
                <span>Pacientes Ingresados</span>
            </div>
            <div class="stat-box3 col">
                <img src="assets/icons/Logo_IMSS.png" alt="Home Icon" width="50"
height="50">
                <p>0</p>
                <span>Encuestas</span>
            </div>
        </div>

        <div class="content row">
            <div class="patients col">
                <h2>Diagnóstico de Pacientes Recientes</h2>
                <table>
                    <tr>
                        <th>Paciente</th>
                        <th>Date Order</th>
                        <th>Status</th>
                    </tr>
                    <tr>
                        <td><img src="https://img.freepik.com/vector-premium/perfil-
medico-icono-servicio-medico_617655-48.jpg" class="patient-img" alt="John Doe"> John
Doe</td>
```

```
                        <td>01-10-2021</td>
                        <td><span class="status completed">Completed</span></td>
                    </tr>
                    <tr>
                        <td><img src="https://img.freepik.com/vector-premium/perfil-
medico-icono-servicio-medico_617655-48.jpg" class="patient-img" alt="John Doe"> John
Doe</td>
                        <td>01-10-2021</td>
                        <td><span class="status pending">Pending</span></td>
                    </tr>
                    <tr>
                        <td><img src="https://img.freepik.com/vector-premium/perfil-
medico-icono-servicio-medico_617655-48.jpg" class="patient-img" alt="John Doe"> John
Doe</td>
                        <td>01-10-2021</td>
                        <td><span class="status process">Process</span></td>
                    </tr>
                </table>
            </div>

            <div class="todos col">
                <h2>Citas Recientes</h2>
                <div class="citaReciente">
                    <p class="nombre" style="margin: 0;">Jose Almendez</p>
                    <h2 class="fecha" style="margin: 0;">22/12/2025</h2>
                    <p class="hora" style="margin: 0;">14:16</p>
                </div>
                <div class="citaReciente">
                    <p class="nombre" style="margin: 0;">Maria Peter</p>
                    <h2 class="fecha" style="margin: 0;">22/12/2025</h2>
                    <p class="hora" style="margin: 0;">16:00</p>
                </div>
            </div>
        </div>
    </div>
</div>
```

ESTRUCTURA SCSS

```
.container {
  background-color: #ffffff;
  padding: 20px;
  max-width: 100%; /* Asegúrate de que nunca se salga de los bordes */
  overflow: hidden; /* Evita el desbordamiento visual */
}
* {
  box-sizing: border-box;
}

.reloj-container {
  display: inline-flex;
  width: max-content;
  margin: 10px;
```

```
}
.reloj {
  font-size: 35px;
  //font-weight: bold;
  color: #519ff4;
}
.hours {
  margin: 5px 7px;
  font-size: 30px;
  font-weight: bold;
}

.date {
  color: #519ff4;
  font-size: 30px;
  text-transform: uppercase;
  //font-weight: bold;
}
.container-fecha {
  margin: 0px 15px;
  width: 100%;
}
.fecha-hora {
  //background: #ACD;
  width: 100%;
  max-width: 350px;
  margin: 20px auto;
}

.dashboard {
  padding: 10px;
}

.statistics {
  display: flex;
  justify-content: space-between;
  gap: 10px;
  margin-bottom: 20px;
}
.statistics .col {
  margin: 5px 0;
}

.stat-box1, .stat-box2, .stat-box3 {
  flex: 1;
  text-align: center;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  color: #ffffff;
}
```

```
.stat-box1 {
  background-color: #4ca7d1;
}

.stat-box2 {
  background-color: #8ec33f;
}

.stat-box3 {
  background-color: #f56858;
}

.content {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
}

.patients, .todos {
  flex: 1;
  min-width: 280px;
  max-width: 100%;
  padding: 20px;
  border-radius: 10px;
  background-color: #ffffff;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.patients table {
  width: 100%;
  border-collapse: collapse;
  font-size: 0.9rem;
}

.patients th, .patients td {
  border: 1px solid #ddd;
  padding: 8px;
}

.patient-img {
  width: 30px;
  height: 30px;
  border-radius: 50%;
  margin-right: 10px;
}

.status {
  display: inline-block;
  padding: 5px 10px;
  border-radius: 5px;
  color: #fff;
}
```

```
.completed {
  background-color: #4caf50;
}

.pending {
  background-color: #ff9800;
}

.process {
  background-color: #ffc107;
}

.citaReciente {
  margin-bottom: 10px;
}

@media (max-width: 720px) {
  .statistics {
    flex-direction: column;
    align-items: center; /* Asegura que los hijos estén centrados */
    gap: 15px;
  }

  .stat-box1, .stat-box2, .stat-box3 {
    width: 100%; /* Ocupan todo el ancho del contenedor */
  padding: 15px;
  }

  .content {
    flex-direction: column;
  }

  .fecha-hora {
    margin-bottom: 20px;
  }
}
```

ESTRUCTURA TYPESCRIPT

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { MedicalHistory } from 'src/app/Shared/Data';

@Component({
  selector: 'app-principal',
  templateUrl: './principal.component.html',
  styleUrls: ['./principal.component.scss']
})
export class PrincipalComponent {

  fecha = new Date();
  constructor(private router: Router){
```

```
  }

  pacientes(){
    this.router.navigate(["home/list-person"]);
  }

  horaActual: Date = new Date();

  ngOnInit(): void {
    setInterval(() => {
      this.horaActual = new Date();
    }, 1000);
  }
}
```

Código de lista de doctores

ESTRUCTURA HTML

```
<div class="boton">
  <button type="button" class="btn btn-success btn-lg" (click)="register()"><i
class="fa-solid fa-user-nurse"></i> Registar Doctor</button>
</div>
<div class="tablaDoctores">
    <div class="container">
      <h2 class="mt-4">Lista de Doctores</h2>
      <table class="table table-striped">
        <thead>
          <tr>
            <th>ID</th>
            <th>Nombre</th>
            <th>Apellido</th>
            <th>Cédula</th>
            <th>Profesión</th>
            <th>Edad</th>
            <th>Género</th>
            <th>Email</th>
            <th>Teléfono</th>
            <th>Dirección</th>
            <th>Acciones</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let user of users">
            <td>{{ user.id }}</td>
            <td>{{ user.nombre }}</td>
            <td>{{ user.apellido }}</td>
            <td>{{ user.cedula }}</td>
            <td>{{ user.profesion }}</td>
            <td>{{ user.edad }}</td>
            <td>{{ user.genero }}</td>
            <td>{{ user.email }}</td>
```

```
            <td>{{ user.telefono }}</td>
            <td>{{ user.direccion }}</td>
            <td>
              <div class="botones">
                <button class="btn btn-sm btn-danger"
(click)="eliminarUsuario(user.id, user.nombre)" title="Elminar {{user.nombre}}">
                  <i class="fas fa-trash-alt"></i> <!-- Icono de eliminar -->
                </button>
                <button class="btn btn-sm btn-primary"
(click)="actualizarUsuario(user.id)" title="Cambiar Contraseña">
                  <i class="fas fa-edit"></i> <!-- Icono de actualizar -->
                </button>
              </div>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
```

ESTRUCTURA TYPESCRIPT

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AlertService } from 'src/app/Service/alert.service';
import { ApiService } from 'src/app/Service/api.service';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-list-doctors',
  templateUrl: './list-doctors.component.html',
  styleUrls: ['./list-doctors.component.scss']
})
export class ListDoctorsComponent implements OnInit{
  showTableMaster: boolean = false; //vista del boton SuperMaster
  // Doctores
  users: any[] = [];
  user: any = {};

  constructor(private apiService: ApiService, private alertService: AlertService,
private sharedService: SharedService, private router: Router){}

  ngOnInit(): void {
    this.fetchUsers();
      this.sharedService.showRegisterButton$.subscribe(show => {
        this.showTableMaster = show;
      });
  }

  eliminarUsuario(id: number, name: string): void {
    console.log('Eliminar usuario con ID:', id ,name);
    // Utilizar el servicio AlertService para mostrar la confirmación
    this.alertService
```

```
    .confirm(`¿Estás seguro de que deseas eliminar al doctor "${name}"?`,
'Confirmación')
    .then((isConfirmed) => {
      if (isConfirmed) {
        // Llamar al API para eliminar al doctor si se confirma
        this.apiService.deleteDoctor(id).subscribe({
          next: (response) => {
            // Mostrar mensaje de éxito
            this.alertService.success(
              `El doctor "${name}" ha sido eliminado exitosamente.`,
              'Eliminado'
            );
            this.cargarDoctores(); // Refrescar la lista de doctores
          },
          error: (error) => {
            console.error('Error al eliminar el doctor:', error);
            // Mostrar mensaje de error
            this.alertService.error(
              `No se pudo eliminar al doctor "${name}". Inténtalo nuevamente.`,
              'Error'
            );
          },
        });
      }
    });
    // Agrega aquí la lógica para eliminar el usuario
  }

  actualizarUsuario(id: number): void {
    console.log('Actualizar usuario con ID:', id);
    // Agrega aquí la lógica para actualizar el usuario
  }

  cargarDoctores(): void {
    this.apiService.getUsers().subscribe({
      next: (users) => this.users = users, // Asigna los usuarios recibidos al array
"users"
      error: (error) => console.error('Error al cargar la lista de doctores:', error)
    });
  }

  fetchUsers() {
    this.apiService.getUsers().subscribe((data) => {
      this.users = data;
    });
  }

  register(){
    this.router.navigate(['home/doctors-register']);
  }
}
```

Registro del Doctor

ESTRUCTURA HTML

```
<div class="container mt-5">
    <div class="row justify-content-center">
      <div class="col-md-6">
        <div class="card shadow">
          <div class="card-header  text-center">
            <h2>Registro de Doctor <i class="fa-solid fa-user-doctor"></i></h2>
          </div>
          <div class="card-body">
            <form [formGroup]="registerForm" (ngSubmit)="registro()">
              <!-- Nombre -->
              <div class="mb-3">
                <label for="nombre" class="form-label">Nombre</label>
                <input id="nombre" type="text" class="form-control"
formControlName="nombre" />
                <div *ngIf="registerForm.get('nombre')?.invalid &&
registerForm.get('nombre')?.touched" class="text-danger">
                  Este campo es obligatorio.
                </div>
              </div>

              <!-- Apellido -->
              <div class="mb-3">
                <label for="apellido" class="form-label">Apellido</label>
                <input id="apellido" type="text" class="form-control"
formControlName="apellido" />
                <div *ngIf="registerForm.get('apellido')?.invalid &&
registerForm.get('apellido')?.touched" class="text-danger">
                  Este campo es obligatorio.
                </div>
              </div>

              <!-- Cédula -->
              <div class="mb-3 cedula">
                <label for="cedula" class="form-label">Cédula</label>
                <input id="cedula" type="text" class="form-control"
formControlName="cedula" />
                <div *ngIf="registerForm.get('cedula')?.invalid &&
registerForm.get('cedula')?.touched" class="text-danger">
                  Este campo es obligatorio.
                </div>
              </div>

              <!-- Profesión -->
              <div class="mb-3">
                <label for="profesion" class="form-label">Profesión</label>
                <input id="profesion" type="text" class="form-control"
formControlName="profesion" />
                <div *ngIf="registerForm.get('profesion')?.invalid &&
registerForm.get('profesion')?.touched" class="text-danger">
```

```
                Este campo es obligatorio.
              </div>
            </div>

            <!-- Edad -->
            <div class="mb-3">
              <label for="date" class="form-label">Fecha de Nacimiento</label>
              <input id="date" type="date" class="form-control"
formControlName="date" />
              <div *ngIf="registerForm.get('date')?.invalid &&
registerForm.get('date')?.touched" class="text-danger">
                Ingrese su Fecha de Nacimiento
              </div>
            </div>


            <!-- Edad -->
            <div class="mb-3">
              <label for="edad" class="form-label">Edad</label>
              <input id="edad" type="number" class="form-control"
formControlName="edad" />
              <div *ngIf="registerForm.get('edad')?.invalid &&
registerForm.get('edad')?.touched" class="text-danger">
                La edad debe ser entre 18 y 99 años.
              </div>
            </div>

            <!-- Género -->
            <div class="mb-3">
              <label for="genero" class="form-label">Género</label>
              <select id="genero" class="form-select" formControlName="genero">
                <option value="">Seleccione...</option>
                <option value="Masculino">Masculino</option>
                <option value="Femenino">Femenino</option>
              </select>
              <div *ngIf="registerForm.get('genero')?.invalid &&
registerForm.get('genero')?.touched" class="text-danger">
                Este campo es obligatorio.
              </div>
            </div>

            <!-- Email -->
            <div class="mb-3">
              <label for="email" class="form-label">Email</label>
              <input id="email" type="email" class="form-control"
formControlName="email" />
              <div *ngIf="registerForm.get('email')?.invalid &&
registerForm.get('email')?.touched" class="text-danger">
                Ingrese un email válido.
              </div>
            </div>
```

```
              <!-- Contraseña -->
              <div class="mb-3">
                <label for="password" class="form-label">Contraseña</label>
                <input id="password" type="password" class="form-control"
formControlName="password" />
                <div *ngIf="registerForm.get('password')?.invalid &&
registerForm.get('password')?.touched" class="text-danger">
                  La contraseña debe tener al menos 6 caracteres.
                </div>
              </div>

              <!-- Teléfono -->
              <div class="mb-3">
                <label for="telefono" class="form-label">Teléfono</label>
                <input id="telefono" type="text" class="form-control"
formControlName="telefono" maxlength="10"/>
                <div *ngIf="registerForm.get('telefono')?.invalid &&
registerForm.get('telefono')?.touched" class="text-danger">
                  Ingrese solo números.
                </div>
              </div>

              <!-- Dirección -->
              <div class="mb-3">
                <label for="direccion" class="form-label">Dirección</label>
                <textarea id="direccion" class="form-control"
formControlName="direccion"></textarea>
                <div *ngIf="registerForm.get('direccion')?.invalid &&
registerForm.get('direccion')?.touched" class="text-danger">
                  Este campo es obligatorio.
                </div>
              </div>

              <!-- Imagen -->
              <div class="mb-3">
                <label for="imagen" class="form-label"><i class="fa-solid fa-
image"></i> Imagen (opcional)</label>
                <input id="imagen" type="file" class="form-control"
(change)="onImageChange($event)" accept="image/*" />
              </div>

              <!-- Botón de Envío -->
              <div class="d-grid">
                <button type="submit" class="btn btn-primary">Registrar</button>
              </div>
            </form>
          </div>
        </div>
      </div>
    </div>
  </div>
```

ESTRUCTURA SCSS

```
.cedula input{
    text-transform: uppercase;
}
.card-header h2{
    font-size: 1.5rem;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    color: white;
}
.card-header{
    background: #2986ff;
    border: none;
}
```

ESTRUCTURA TYPESCRIPT

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { AlertService } from 'src/app/Service/alert.service';
import { ApiService } from 'src/app/Service/api.service';
import Swal from 'sweetalert2';

@Component({
  selector: 'app-registro',
  templateUrl: './registro.component.html',
  styleUrls: ['./registro.component.scss']
})
export class RegistroComponent implements OnInit{
  registerForm!: FormGroup;
  submitted = false;
  errors: any = {}; // Almacena mensajes de error personalizados
  selectedImage!: File | null; // Almacena la imagen seleccionada


   constructor(private fb: FormBuilder,private  doctorsApi: ApiService, private
router: Router, private alert: AlertService){}

   ngOnInit(): void {
    this.registerForm = this.fb.group({
      nombre: ['', Validators.required],
      apellido: ['', Validators.required],
      cedula: ['', Validators.required],
      profesion: ['', Validators.required],
      date: ['', Validators.required],
      edad: ['', [Validators.required, Validators.min(18), Validators.max(99)]],
      genero: ['', Validators.required],
      email: ['', [Validators.required, Validators.email]],
      password: ['', [Validators.required, Validators.minLength(6)]],
      telefono: ['', [Validators.required, Validators.pattern(/^\d+$/)]],
      direccion: ['', Validators.required],
      imagen: [''], // Imagen es opcional
    });
   }
```

```
   registro(){
    if (this.registerForm.invalid) {
      console.log('Formulario no válido');
      this.alert.warning('No deje campos vacios, por favor de llenarlo', 'Campos
Vacios');
      return;
    }

    const formData = new FormData();
    Object.keys(this.registerForm.value).forEach((key) => {
      if (key !== 'imagen') {
        formData.append(key, this.registerForm.get(key)?.value || '');
      }
    });

    // Si se seleccionó una imagen, añadirla al FormData
    if (this.selectedImage) {
      formData.append('imagen', this.selectedImage);
    }

    if (this.registerForm.valid) {
      this.doctorsApi.registerDoctor(formData).subscribe(
        (response) => {
          console.log('Doctor registrado:', response);
          this.alert.success('Registro de usuario Exitoso', 'Succesfuly Register');
          this.router.navigate(['home/list-doctors']);
        },
        (error) => {
          console.error('Error al registrar doctor:', error);
          this.alert.error('Error al registrar al Usuario', 'Failed Register');
        }
      );
    } else {
      console.log('Formulario no válido');
      this.alert.warning('Error de Formulatio', 'Register Failed');
    }
   }

   onImageChange(event: any): void {
    const file = event.target.files[0];
    if (file) {
      this.selectedImage = file;
    }
  }
}
```

Código de la lista de los pacientes

ESTRUCTURA HTML

```
<!-- Botón para Abrir el Modal -->
```

```
<div class="d-grid gap-2 col-6 btn_click">
  <button class="btn btn-success" type="button" (click)="patient()"><i class="fa-
solid fa-circle-plus"></i> Añadir Paciente</button>
</div>

<div class="container">
  <h1>Pacientes</h1>
  <table mat-table [dataSource]="dataSource" multiTemplateDataRows class="mat-
elevation-z8">
    <!-- Column definitions -->
    <ng-container matColumnDef="{{column}}" *ngFor="let column of columnsToDisplay">
      <th mat-header-cell *matHeaderCellDef> {{columnHeaders[column]}} </th>
      <td mat-cell *matCellDef="let element"> {{element[column]}} </td>
    </ng-container>
    <!-- Expand button column -->
    <ng-container matColumnDef="expand">
      <th mat-header-cell *matHeaderCellDef aria-label="row actions"> </th>
      <td mat-cell *matCellDef="let element">
        <button mat-icon-button aria-label="expand row"
          (click)="(expandedElement = expandedElement === element ? null : element);
$event.stopPropagation()">
          <mat-icon *ngIf="expandedElement !== element">keyboard_arrow_down</mat-
icon>
          <mat-icon *ngIf="expandedElement === element">keyboard_arrow_up</mat-icon>
        </button>
      </td>
    </ng-container>

    <!-- Expanded Content Column - The detail row is made up of this one column that
spans across all columns -->
    <ng-container matColumnDef="expandedDetail">
      <td mat-cell *matCellDef="let element"
[attr.colspan]="columnsToDisplayWithExpand.length">
        <div class="example-element-detail" [@detailExpand]="element ==
expandedElement ? 'expanded' : 'collapsed'">
          <div class="example-element-diagram">
            <!--Foto del paciente-->
            <img src="assets/icons/Logo_IMSS.png" alt="Home Icon" width="50"
height="50" class="perfil">
          </div>
          <div class="botons">
            <button type="button" class="btn btn-info"
(click)="perfilUser(element.id)">Perfil Paciente</button>
          </div>
          <div class="example-element-description">
            {{element.description}}
          </div>
        </div>
      </td>
    </ng-container>
     <!-- Table rows -->
    <tr mat-header-row *matHeaderRowDef="columnsToDisplayWithExpand"></tr>
```

```
    <tr mat-row *matRowDef="let element; columns: columnsToDisplayWithExpand;"
class="example-element-row"
      [class.example-expanded-row]="expandedElement === element"
      (click)="expandedElement = expandedElement === element ? null : element">
    </tr>
    <tr mat-row *matRowDef="let row; columns: ['expandedDetail']" class="example-
detail-row"></tr>
  </table>
</div>
```

ESTRUCTURA SCSS

```
.container {
  background-color: #e4a6ff;
  padding: 20px;
  max-width: 100%;
  /* Asegúrate de que nunca se salga de los bordes */
  overflow-x: auto;
  /* Permite desplazamiento horizontal en caso necesario */
}

* {
  box-sizing: border-box;
}

table {
  width: 100%;
  border-collapse: collapse;
}

th,
td {
  text-align: left;
  padding: 8px;
  font-size: 14px;
  /* Ajusta el tamaño de fuente para pantallas pequeñas */
}

tr.example-detail-row {
  height: 0;
}

tr.example-element-row:not(.example-expanded-row):hover {
  background: whitesmoke;
}

tr.example-element-row:not(.example-expanded-row):active {
  background: #efefef;
}

.example-element-row td {
  border-bottom-width: 0;
}
```

```
.example-element-detail {
  overflow: hidden;
  display: flex;
  flex-wrap: wrap; /* Asegúrate de que los elementos internos se acomoden */
}

.example-element-diagram {
  min-width: 80px;
  border: 2px solid black;
  padding: 8px;
  font-weight: lighter;
  margin: 8px 0;
  height: 104px;
}

.example-element-symbol {
  font-weight: bold;
  font-size: 40px;
  line-height: normal;
}

.example-element-description {
  padding: 16px;
}

.example-element-description-attribution {
  opacity: 0.5;
}

.perfil {
  background-color: aqua;
}

/* BOTONES */
.btn_click {
  max-width: 500px;
  margin: 15px auto;
}

@media screen and (max-width: 768px) {
  th,
  td {
    font-size: 12px; /* Reduce el tamaño de fuente en pantallas pequeñas */
    padding: 6px;
  }

  .example-element-detail {
    flex-direction: column; /* Ajusta el diseño del detalle en columnas */
  }

  table {
```

```
    font-size: 12px;
  }
}

@media screen and (max-width: 480px) {
  th,
  td {
    font-size: 10px; /* Más pequeño en pantallas extra pequeñas */
    padding: 4px;
  }

  .container {
    padding: 10px; /* Ajusta el margen en pantallas pequeñas */
  }
}
```

ESTRUCTURA TYPESCRIPT

```
import { Component, ViewChild } from '@angular/core';
import { MatPaginator } from '@angular/material/paginator';
import { MatTableDataSource } from '@angular/material/table';
import { animate, state, style, transition, trigger,} from '@angular/animations';
import { PATIENTS_DATA } from '../data/patients-data';
import { Router } from '@angular/router';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { ApiService } from 'src/app/Service/api.service';
import { UserService } from 'src/app/Service/user.service';

@Component({
  selector: 'app-patients',
  templateUrl: './patients.component.html',
  styleUrls: ['./patients.component.scss'],
  animations: [
    trigger('detailExpand', [
      state('collapsed', style({ height: '0px', minHeight: '0' })),
      state('expanded', style({ height: '*' })),
      transition(
        'expanded <=> collapsed',
        animate('225ms cubic-bezier(0.4, 0.0, 0.2, 1)')
      ),
    ]),
  ],
})
export class PatientsComponent {
  title = 'Pacientes';
  columnsToDisplay: string[] = [
    'first_name',
    'last_name',
    'age',
    'last_consultation',
  ];
  columnsToDisplayWithExpand = [...this.columnsToDisplay, 'expand'];
  dataSource = new MatTableDataSource(PATIENTS_DATA); // Usa los datos importados
```

```
  expandedElement: any | null = null;

  @ViewChild(MatPaginator, { static: true }) paginator:
    | MatPaginator
    | undefined;

    constructor(private router: Router, private apiService: ApiService, private
userService: UserService){

    }

  ngOnInit() {
    // Configura el paginador
    if (this.paginator) {
      this.dataSource.paginator = this.paginator;
    }

    // Obtener pacientes del doctor
    this.getPatients();
  }

  columnHeaders: { [key: string]: string } = {
    first_name: 'Nombre',
    last_name: 'Apellido',
    age: 'Edad',
    last_consultation: 'Última consulta',
  };

  // Método para redirigir al perfil del paciente
  perfilUser(id: number){
    // Redirige al componente del perfil de paciente pasando el id en la URL
    this.router.navigate(['home/view-person',{patientID: id}]);
    console.log(id)
  }

  patient(){
    this.router.navigate(['/home/register-patient'])
  }

  getPatients(){
    this.apiService.getPatientsByDoctor(this.userService.getDoctorId()).subscribe(
      (response) =>{
        this.dataSource.data = response.patients;
      }, (error) =>{
        alert('Error al obtener los pacientes');
      }
    );
  }
}
```

Registro del paciente

ESTRUCTURA HTML

```
<!-- Modal para Registro de Paciente -->
<div class="container">
    <div class="title">
        <h1 class="modal-title fs-5" id="staticBackdropLabel">
            <i class="fa-solid fa-person-circle-plus"></i> Añadir Paciente
        </h1>
    </div>

    <div class="modal-body">
        <!-- Formulario de Registro -->
        <form [formGroup]="pacienteForm" (ngSubmit)="onSubmit()">
            <!-- Matrícula -->
            <div class="form-group">
                <label for="matricula">Matrícula:</label>
                <input type="text" class="form-control" id="matricula"
formControlName="matricula">
                <div *ngIf="submitted && pacienteForm.controls['matricula'].errors">
                    <div
*ngIf="pacienteForm.controls['matricula'].errors['required']" class="alert alert-
danger m-2">
                        Matrícula es requerida
                    </div>
                </div>
            </div>

            <!-- Nombre -->
            <div class="form-group">
                <label for="nombre">Nombre:</label>
                <input type="text" class="form-control" id="nombre"
formControlName="nombre" [(ngModel)]="pacient.nombre">
                <div *ngIf="submitted && pacienteForm.controls['nombre'].errors">
                    <div *ngIf="pacienteForm.controls['nombre'].errors['required']"
class="alert alert-danger m-2">
                        Nombre es requerido
                    </div>
                </div>
            </div>

            <!-- Apellido -->
            <div class="form-group">
                <label for="apellido">Apellido:</label>
                <input type="text" class="form-control" id="apellido"
formControlName="apellido">
                <div *ngIf="submitted && pacienteForm.controls['apellido'].errors">
                    <div *ngIf="pacienteForm.controls['apellido'].errors['required']"
class="alert alert-danger m-2">
                        Apellido es requerido
                    </div>
                </div>
```

```
            </div>

            <!-- Edad -->
            <div class="form-group">
                <label for="edad">Edad:</label>
                <input type="number" class="form-control" id="edad"
formControlName="edad" maxlength="3">
                <div *ngIf="submitted && pacienteForm.controls['edad'].errors"
class="alert alert-danger m-2">
                    <div
*ngIf="pacienteForm.controls['edad'].errors['required']">Edad es requerida</div>
                    <div *ngIf="pacienteForm.controls['edad'].errors['min']">Edad
mínima es 20</div>
                </div>
            </div>

            <!-- Género -->
            <div class="form-group">
                <label for="genero">Género:</label>
                <select class="form-control" id="genero" formControlName="genero"
[(ngModel)]="pacient.genero">
                    <option value="">Seleccione</option>
                    <option value="Masculino">Masculino</option>
                    <option value="Femenino">Femenino</option>
                </select>
                <div *ngIf="submitted && pacienteForm.controls['genero'].errors">
                    <div *ngIf="pacienteForm.controls['genero'].errors['required']"
class="alert alert-danger m-2">Género es requerido</div>
                </div>
            </div>

            <!-- Fecha de Nacimiento -->
            <div class="form-group">
                <label for="fechaNacimiento">Fecha de Nacimiento: </label>
                <input type="date" id="fechaNacimiento" name="bday" class="form-
control" [(ngModel)]="pacient.fecha_nacimiento" formControlName="fecha_nacimiento">
                <div *ngIf="submitted &&
pacienteForm.controls['fecha_nacimiento'].errors" class="alert alert-danger m-2">
                    <div
*ngIf="pacienteForm.controls['fecha_nacimiento'].errors['required']" >Introduzca la
fecha</div>
                </div>
            </div>

            <!-- Contacto de Emergencia -->
            <div class="form-group">
                <label for="telefono">Contacto de Emergencia:</label>
                <input type="text" class="form-control" id="telefono"
formControlName="telefono" maxlength="15" [(ngModel)]="pacient.telefono">
                <div *ngIf="submitted && pacienteForm.controls['telefono'].errors"
class="alert alert-danger m-2">
```

```
                    <div
*ngIf="pacienteForm.controls['telefono'].errors['required']">El contacto de
emergencia es obligatorio.</div>
                    <div
*ngIf="pacienteForm.controls['telefono'].errors['pattern']">Solo se permiten
números.</div>
                    <div
*ngIf="pacienteForm.controls['telefono'].errors['exactLength']">
                        El número debe ser exactamente 10 dígitos. (Actual: {{
pacienteForm.controls['telefono'].errors['exactLength'].actualLength }})
                    </div>
                </div>
            </div>


            <!-- Correo Electrónico de Emergencia -->
            <div class="form-group">
                <label for="email">Correo Electrónico de Emergencia:</label>
                <input type="email" class="form-control" id="email"
formControlName="email" [(ngModel)]="pacient.email">
                <div *ngIf="submitted && pacienteForm.controls['email'].errors"
class="alert alert-danger m-2">
                    <div
*ngIf="pacienteForm.controls['email'].errors['required']">Correo de emergencia
requerido</div>
                    <div *ngIf="pacienteForm.controls['email'].errors['email']">Solo
acepta correo valido</div>
                </div>
            </div>

            <!-- Dirección -->
            <div class="form-group">
                <label for="direccion">Dirección:</label>
                <textarea class="form-control" id="direccion"
formControlName="direccion" rows="3" [(ngModel)]="pacient.direccion"></textarea>
                <div *ngIf="submitted && pacienteForm.controls['direccion'].errors"
class="alert alert-danger m-2">
                    <div
*ngIf="pacienteForm.controls['direccion'].errors['required']">Dirección es
requerida</div>
                </div>
            </div>

            <!-- Historial Médico -->
            <div class="form-group">
                <label for="historialMedico">Historial Médico:</label>
                <textarea class="form-control" id="historialMedico"
formControlName="historialMedico" rows="3"></textarea>
                <div *ngIf="submitted &&
pacienteForm.controls['historialMedico'].errors" class="alert alert-danger m-2">
```

```
                    <div
*ngIf="pacienteForm.controls['historialMedico'].errors['required']">Historial Médico
es requerido</div>
                </div>
            </div>

            <!-- Alergias -->
            <div class="form-group">
                <label for="alergias">Alergias:</label>
                <textarea class="form-control" id="alergias"
formControlName="alergias" rows="2"></textarea>
                <div *ngIf="submitted && pacienteForm.controls['alergias'].errors"
class="alert alert-danger m-2">
                    <div
*ngIf="pacienteForm.controls['alergias'].errors['required']">Alergias son
requeridas</div>
                </div>
            </div>

            <!-- Descripción -->
            <div class="form-group">
                <label for="caracteristicas">Descripción:</label>
                <textarea class="form-control" id="caracteristicas"
formControlName="caracteristicas" rows="3"></textarea>
                <div *ngIf="submitted &&
pacienteForm.controls['caracteristicas'].errors" class="alert alert-danger m-2">
                    <div
*ngIf="pacienteForm.controls['caracteristicas'].errors['required']">Agrege una breve
descriocion para mostrar</div>
                </div>
            </div>

            <!-- Confirmación -->
            <div class="form-group form-check">
                <input type="checkbox" class="form-check-input" id="confirmacion"
formControlName="confirmacion">
                <label class="form-check-label" for="confirmacion">Confirmo que los
datos Ingresados son correctos</label>
                <div *ngIf="submitted &&
pacienteForm.controls['confirmacion'].errors" class="alert alert-danger m-2">
                    <div
*ngIf="pacienteForm.controls['confirmacion'].errors['required']">La confirmación es
requerida</div>
                </div>
            </div>

            <!-- Botones de acción -->
            <div class="text-center">
                <button type="submit" class="btn btn-success"><i class="fa-solid fa-
check"></i> Guardar</button>
                <button type="button" class="btn btn-danger" data-bs-
dismiss="modal"><i class="fa-solid fa-circle-xmark"></i> Cancelar</button>
```

```
            </div>
        </form>
    </div>
</div>
```

ESTRUCTURA SCSS

```
.text-center .btn {
  margin: 5px 15px;
}
.boxBtn .btn {
  margin: 0px 15px;
}
.container {
  width: 100%;
  max-width: 720px;
}

.container .title {
  width: 100%;
  background: #17b2ff;
  padding: 5px;
  text-align: center;
  margin-bottom: 2rem;

  h1 {
    margin-top: 5px;
    color: #ffffff;
    text-transform: uppercase;
  }
}

.form-group{
  margin-bottom: 1rem;
}
```

ESTRUCTURA TYPESCRIPT

```
import { Component, OnDestroy, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators, AbstractControl, ValidationErrors,
ValidatorFn } from '@angular/forms';
import { Router } from '@angular/router';
import { AlertService } from 'src/app/Service/alert.service';
import { ApiService } from 'src/app/Service/api.service';
import { UserService } from 'src/app/Service/user.service';

@Component({
  selector: 'app-register-patient',
  templateUrl: './register-patient.component.html',
  styleUrls: ['./register-patient.component.scss']
})
export class RegisterPatientComponent implements OnInit, OnDestroy{
  /* REGISTRO DE PACIENTE */
  pacienteForm: FormGroup;
```

```
  pacientes: any[] = []; // Array para almacenar los pacientes registrados
  submitted = false;

  doctor = {
    id: 0, // Este sería el ID del doctor autenticado
    name: 'Undefinied'
  };


  // Objeto para almacenar los datos del formulario
  pacient = {
    nombre: '',
    apellido: '',
    genero: '',
    fecha_nacimiento: '',
    telefono: '',
    direccion: '',
    email: '',
    edad: '',
    enfermedad: '',
    caracteristicas: ''
  };

  constructor(private formBuilder: FormBuilder, private apiService: ApiService,
private alertService: AlertService, private userService: UserService, private router:
Router) {
    // Definición del formulario reactivo
    this.pacienteForm = this.formBuilder.group({
      matricula: ['', Validators.required], //Matricula
      nombre: ['', Validators.required], //Nombre
      apellido: ['', Validators.required], //Apellido
      genero: ['', Validators.required], //Genero
      fecha_nacimiento: ['', [Validators.required]], //Fecha Nacimiento
      telefono: ['', [ //Telefono de Emergencia
          Validators.required,
          Validators.pattern(/^[0-9]*$/), // Solo números
          exactLength(10)      // Máximo de 10 dígitos
        ]
      ],
      direccion: ['', Validators.required], // Direccion
      email: ['', [Validators.required, Validators.email]], //Correo de Emergencia
      edad: ['', [Validators.required, Validators.min(20)]], //Edad
      historialMedico: ['', Validators.required], //Historial Medico
      alergias: ['', Validators.required], // Alergias que tiene
      caracteristicas: ['', Validators.required], // Descripcion
      confirmacion: [false, Validators.requiredTrue]
    });
  }

  ngOnInit(): void {
      this.doctor.id = this.userService.getDoctorId();
      this.doctor.name = this.userService.getDoctorName();
```

```
      console.log('Nombre: '+this.doctor.name +' ID: '+this.doctor.id)

  }

  ngOnDestroy(): void {
      console.log('Componente destruido manualmente');
  }

  // Método para manejar el envío del formulario
  onSubmit() {
    this.submitted = true;

    // Generar la fecha y hora actual en el formato [YYYY-MM-DD - HH:mm:ss]
    const now = new Date();
    const formattedDate = `${now.getFullYear()}-${(now.getMonth() +
1).toString().padStart(2, '0')}-${now.getDate().toString().padStart(2, '0')}`;
    const formattedTime = `${now.getHours().toString().padStart(2,
'0')}:${now.getMinutes().toString().padStart(2,
'0')}:${now.getSeconds().toString().padStart(2, '0')}`;
    const lastConsultation = `${formattedDate} - ${formattedTime}`;

    // Validación del formulario
    if (this.pacienteForm.invalid) {
      this.alertService.warning('Por favor, completa todos los campos requeridos.',
'Formulario Incompleto');
      console.log(this.pacienteForm.valid);
      return;
    }

    // Obtener datos del formulario y añadir el ID del doctor
    const pacienteData = {
      registration_number: this.pacienteForm.value.matricula,
      first_name: this.pacienteForm.value.nombre,
      last_name: this.pacienteForm.value.apellido,
      age: this.pacienteForm.value.edad,
      gender: this.pacienteForm.value.genero,
      birth_date: this.pacienteForm.value.fecha_nacimiento,
      emergency_contact: this.pacienteForm.value.telefono,
      emergency_email: this.pacienteForm.value.email,
      address: this.pacienteForm.value.direccion,
      medical_history: this.pacienteForm.value.historialMedico,
      allergies: this.pacienteForm.value.alergias,
      description: this.pacienteForm.value.caracteristicas,
      last_consultation: lastConsultation, // Fecha y hora de la consulta
      doctor_id: this.doctor.id // Suponiendo que el objeto `doctor` contiene los
datos del doctor actual
    };

    this.apiService.crearPaciente(pacienteData).subscribe({
      next: (response) => {
        this.alertService.success('Paciente registrado correctamente.', 'Éxito');
        console.log('Respuesta del servidor:', response);
```

```
        // Reiniciar el formulario
        this.pacienteForm.reset();
        this.submitted = false;
        this.router.navigate(['home/list-person']);
      },
      error: (error) => {
        this.alertService.error('Ocurrió un error al registrar el paciente. Por
favor, inténtelo de nuevo.', 'Error');
        console.error('Error al registrar paciente:', error);
      }
    });

    console.log(lastConsultation);
  }
}

// Validador personalizado TELEFONO
export function exactLength(length: number): ValidatorFn {
  return (control: AbstractControl): ValidationErrors | null => {
    const value = control.value;
    if (value && value.length !== length) {
      return { exactLength: { requiredLength: length, actualLength: value.length } };
    }
    return null;
  };
}
```

Vista del Paciente

ESTRUCTURA HTML

```
<!-- Visualizacion de Datos del Paciente -->
<div class="container">
  <div class="row justify-content-center">
    <div class="container my-4">
      <h2>Detalles del Paciente</h2>
      <div class="card">
        <div class="card-body">
          <!-- Información del paciente -->
          <div class="row mb-3">
            <div class="col-md-4"><strong>Matrícula:</strong></div>
            <div class="col-md-8">{{ patientData.registration_number }}</div>
          </div>
          <div class="row mb-3">
            <div class="col-md-4"><strong>Nombre:</strong></div>
            <div class="col-md-8">{{ patientData.first_name }}</div>
          </div>
          <div class="row mb-3">
            <div class="col-md-4"><strong>Apellido:</strong></div>
            <div class="col-md-8">{{ patientData.last_name }}</div>
          </div>
          <div class="row mb-3">
```

```
            <div class="col-md-4"><strong>Edad:</strong></div>
            <div class="col-md-8">{{ patientData.age }}</div>
          </div>
          <div class="row mb-3">
            <div class="col-md-4"><strong>Fecha de Nacimiento:</strong></div>
            <div class="col-md-8">{{ patientData.birth_date | date }}</div>
          </div>
          <div class="row mb-3">
            <div class="col-md-4"><strong>Correo de Emergencia:</strong></div>
            <div class="col-md-8">{{ patientData.emergency_email }}</div>
          </div>
          <div class="row mb-3">
            <div class="col-md-4"><strong>Teléfono de Emergencia:</strong></div>
            <div class="col-md-8">{{ patientData.emergency_contact }}</div>
          </div>
          <div class="row mb-3">
            <div class="col-md-4"><strong>Dirección:</strong></div>
            <div class="col-md-8">{{ patientData.address }}</div>
          </div>
          <div class="row mb-3">
            <div class="col-md-4"><strong>Historial Médico:</strong></div>
            <div class="col-md-8">{{ patientData.medical_history }}</div>
          </div>
          <div class="row mb-3">
            <div class="col-md-4"><strong>Alergias:</strong></div>
            <div class="col-md-8">{{ patientData.allergies }}</div>
          </div>
          <!-- Botones Modificar y Eliminar dentro del cuadro -->
          <div class="text-center mt-3 boxBtn">
            <button type="button" class="btn btn-warning mr-2"><i class="fa-regular
fa-pen-to-square"></i> Modificar</button>
            <button type="button" class="btn btn-danger" (click)="deletePatient()"><i
class="fa-solid fa-trash"></i> Eliminar</button>
            <button type="button" class="btn btn-success" (click)="categoria()"><i
class="fa-solid fa-file-medical"></i> Diagnostico</button>
            <button type="button" class="btn btn-history" (click)="history()"><i
class="fa-solid fa-book-medical"></i> Historial</button>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

ESTRUCTURA SCSS

```
.boxBtn .btn {
  margin: 0px 15px;
}
.text-center .btn {
  margin: 5px 15px;
}
```

```
.btn-history {
  background: #81c4ff;
}
.btn-history:hover {
  background: #5db3ff;
}

.container {
  max-width: 800px;
}

.card {
  border-radius: 10px;
}

.card-body {
  padding: 20px;
}
```

ESTRUCTURA TYPESCRIPT

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Route, Router } from '@angular/router';
import { AlertService } from 'src/app/Service/alert.service';
import { ApiService } from 'src/app/Service/api.service';

@Component({
  selector: 'app-vistas',
  templateUrl: './vistas.component.html',
  styleUrls: ['./vistas.component.scss']
})
export class VistasComponent implements OnInit {
  patientData: any;
  patientID?: number;

  constructor(private router: Router, private route: ActivatedRoute, private
patientService: ApiService, private alert: AlertService) {

  }

  ngOnInit(): void {
    // Obtener el patientId desde los parámetros de la ruta
    this.route.paramMap.subscribe(params => {
      this.patientID = +params.get('patientID')!;

      // Llamar a la API para obtener los detalles del paciente
      this.patientService.getPatientById(this.patientID).subscribe(response => {
        this.patientData = response.patient;
      });
    });
  }
  categoria() {
    this.router.navigate(['home/diagnostic']);
```

```
  }
  history(){
    this.router.navigate(['home/dashboard'])
  }
  deletePatient(){
    this.route.paramMap.subscribe(params => {
      this.patientID = +params.get('patientID')!;

      this.alert
      .confirm('¿Estás seguro de que deseas eliminar este paciente?', 'Confirmación
de eliminación')
      .then((isConfirmed) => {
        if (isConfirmed) {
          // Si el usuario confirma, se realiza la eliminación
          this.patientService.deletePatient(this.patientID!).subscribe(
            (response) => {
              this.router.navigate(['home/list-person']);
              this.alert.success('Paciente eliminado correctamente', 'Éxito');
            },
            (error) => {
              this.alert.error('Ocurrió un error al eliminar el paciente', 'Error');
              console.error('Error al eliminar paciente:', error);
            }
          );
        }
      });
    });
  }
}
```

Código de la categoría

ESTRUCTURA HTML

```
<div class="categoryContent">
  <div class="title">
      <p>Categoria de la Encuesta</p>
  </div>
  <div *ngFor="let category of categoria; let i = index" class="category">
      <div class="category-title" (click)="toggleCategoryIndex(i)">
        {{ category.title }}
      </div>
      <div class="category-content" [ngStyle]="{ display: selectedCategoryIndex === i
? 'block' : 'none' }">
        {{ category.content }}
        <ul class="lista">
          <li (click)="vent(category.cat, 1)">{{category.cont1}}</li>
          <li (click)="vent(category.cat, 2)">{{category.cont2}}</li>
          <li (click)="vent(category.cat, 3)">{{category.cont3}}</li>
          <li (click)="vent(category.cat, 4)">{{category.cont4}}</li>
          <li (click)="vent(category.cat, 5)">{{category.cont5}}</li>
          <li (click)="vent(category.cat, 6)">{{category.cont6}}</li>
          <li (click)="vent(category.cat, 7)">{{category.cont7}}</li>
```

```
          <li (click)="vent(category.cat, 8)">{{category.cont8}}</li>
          <li (click)="vent(category.cat, 9)">{{category.cont9}}</li>
        </ul>
      </div>
  </div>
</div>
```

ESTRUCTURA SCSS

```
.category {
    max-width: 100%;
    margin: 10px auto;
}

.category-title {
    background-color: #e68dfc;
    color: white;
    padding: 10px;
    cursor: pointer;
    border-radius: 5px;
}
.category-title:hover{
    background-color: #e063ff;
}

.category-content {
    display: none;
    padding: 10px;
    background-color: #ffbbf7;
    border: 1px solid #d64eff;
    border-radius: 5px;
    margin-top: 5px;
}

.title p{
    text-align: center;
    font-size: 35px;
    text-transform: uppercase;
    font-weight: bold;
    margin: 2rem 0;
}
.lista li{
    cursor: pointer;
    max-width: 250px;
    list-style: none;
    display: block;
}

.lista li:hover{
    opacity: 0.6;
}

.categoryContent{
```

```
    width: 100%;
    max-width: 1366px;
    margin: auto;
    height: 70vh;
}

@media (max-width: 1360px) {
    .categoryContent{
       padding: 0px 15px;
    }

    .title p{
        font-size: 25px;
    }
}
```

ESTRUCTURA TYPESCRIPT

```
import { Component, EventEmitter, Input, OnDestroy, OnInit, Output } from
'@angular/core';
import { Router } from '@angular/router';
import { ApiService } from 'src/app/Service/api.service';

@Component({
  selector: 'app-diagnostic',
  templateUrl: './diagnostic.component.html',
  styleUrls: ['./diagnostic.component.scss']
})
export class DiagnosticComponent implements OnInit, OnDestroy{
  //Salida de Datos del Componente
  @Output() categoriaSeleccionada = new EventEmitter<String>();

  //Nombre de las Categorias
  ec: String = 'Esfera Cognitiva'; sub: String = "Subcategoria:"; ea: String =
"Esfera Afectiva"; ef: String = "Esfera Funcional"; en: String = "Esfera
Nutricional";
  selectedCategoryIndex: number | null = null;

  //Lista de la Categoria a Mostrar
  categoria = [
    { title: this.ec.toUpperCase(), cat: 1, content: this.sub + ' ' + this.ec, cont1:
'4AT', cont2: 'CAM', cont3: 'CAM-ICU', cont4: 'AWOL', cont5: 'SPMSQP', cont6: 'Prueba
del Reloj' },
    { title: this.ea.toUpperCase(), cat: 2, content: this.sub + ' ' + this.ea,
      cont1: 'GDS-15', cont2: 'CES-D7', cont3: 'PHQ9', cont4: 'GAI-SF', cont5:
'Inventario Ansiedad Beck', cont6: 'Escala Soledad 3 Elementos', cont7: 'Riesgo de
Suicidio SAD PERSON', cont8: 'Escala Cornell', cont9:'Corta Depresion por
Observacion' },
    { title: this.ef.toUpperCase(), cat: 3, content: this.sub + ' ' + this.ef, cont1:
'KATZ', cont2: 'Indice Barthel', cont3: 'Lawton y Brody', cont4: 'FRAIL', cont5:
'Criterios Ensrud', cont6: 'Time UP and Go' },
```

```
    { title: this.en.toUpperCase(), cat: 4, content: this.sub + ' ' + this.en,
cont1:'Mini Nutritional Assessment SF', cont2:'Mini Nutritional Assessment',
cont3:'MUST', cont4:'Criterios Glim', cont5:'Sarc-F',cont6:'EAT-10'},
  ];
  constructor(private router: Router, private serviceApi: ApiService){}

  ngOnInit(): void {
      // Bloquea el clic derecho en toda la página
    document.addEventListener('contextmenu', this.disableRightClick);
  }

  // Función que evita el clic derecho
  disableRightClick(event: MouseEvent) {
    event.preventDefault();
  }
  ngOnDestroy(): void {
    // Eliminar el listener al destruir el componente
    document.removeEventListener('contextmenu', this.disableRightClick);
  }

  toggleCategoryIndex(index: number){
    if (this.selectedCategoryIndex === index) {
      this.selectedCategoryIndex = null; // Cerrar si se vuelve a hacer clic en la
misma categoría
    } else {
      this.selectedCategoryIndex = index;
    }
  }

  vent(c: number, s: number) {
    console.log(c,s);
    switch(c){
      case 1:
        this.seleccionarCategoria('cognitiva');
        switch(s){
          case 1:
            //this.router.navigate(['home/encuesta/{categoria}']);
            //this.cambiarCategoria(this.ec.toLowerCase()+'-4at.php');
            this.cambiarCategoria(this.url(this.ec,'4at')); //Enviar tipo URL
            this.seleccionarSubCategoria('4at'); //Enciat nombre (Clave) para cada
encuesta
            break;
          case 2:
            this.cambiarCategoria(this.url(this.ec,'cam'))
            this.seleccionarSubCategoria('cam');
            break;
          case 3:
            this.cambiarCategoria(this.url(this.ec,'cam-icu'));
            this.seleccionarSubCategoria('camicu');
            break;
          case 4:
            this.cambiarCategoria(this.url(this.ec,'awol'));
```

```
            this.seleccionarSubCategoria('awol');
            break;
          case 5:
            this.cambiarCategoria(this.url(this.ec,'spmsqp'));
            this.seleccionarSubCategoria('spmsqp');
            break;
          case 6:
            this.cambiarCategoria(this.url(this.ec,'prueba-reloj'));
            this.seleccionarSubCategoria('reloj');
            break;
        }
        break;
      case 2:
        this.seleccionarCategoria('afectiva');
        // this.seleccionarSubCategoria('Undefinied');
        switch(s){
          case 1:
            this.cambiarCategoria(this.url(this.ea,'gds-15'));
            this.seleccionarSubCategoria('gds15');
            break;
          case 2:
            this.cambiarCategoria(this.url(this.ea,'ces-d7-depresion-
epidemioloicos'));
            this.seleccionarSubCategoria('cesd7');
            break;
          case 3:
            this.cambiarCategoria(this.url(this.ea,'phq9'));
            this.seleccionarSubCategoria('phq9');
            break;
          case 4:
            this.cambiarCategoria(this.url(this.ea, 'gai-sf'));
            this.seleccionarSubCategoria('gaisf');
            break;
          case 5:
            this.cambiarCategoria(this.url(this.ea, 'ansiedad-becky'));
            this.seleccionarSubCategoria('ansbeck');
            break;
          case 6:
            this.cambiarCategoria(this.url(this.ea, 'soledad-3-elementos'));
            this.seleccionarSubCategoria('soledad3');
            break;
          case 7:
            this.cambiarCategoria(this.url(this.ea, 'riesgo-suicido-sad-person'));
            this.seleccionarSubCategoria('sadperson');
            break;
          case 8:
            this.cambiarCategoria(this.url(this.ea,'escala-cornell'));
            this.seleccionarSubCategoria('cornell');
            break;
          case 9:
            this.cambiarCategoria(this.url(this.ea,'depresion-corta-okeeffe'));
            this.seleccionarSubCategoria('cortdepresion');
```

```
            break;
        }
        break;
      case 3:
        this.seleccionarCategoria('funcional');
        switch(s){
          case 1:
            this.cambiarCategoria(this.url(this.ef,'indice-katz'));
            this.seleccionarSubCategoria('katz');
            break;
          case 2:
            this.cambiarCategoria(this.url(this.ef,'indice-barthel'));
            this.seleccionarSubCategoria('barthel');
            break;
          case 3:
            this.cambiarCategoria(this.url(this.ef,'indice-lawton-y-brody'));
            this.seleccionarSubCategoria('lawton');
            break
          case 4:
            this.cambiarCategoria(this.url(this.ef, 'test-frail'));
            this.seleccionarSubCategoria('frail');
            break;
          case 5:
            this.cambiarCategoria(this.url(this.ef, 'criterios-ensrud'));
            this.seleccionarSubCategoria('ensrud');
            break;
          case 6:
            this.cambiarCategoria(this.url(this.ef, 'time-up-and-go'));
            this.seleccionarSubCategoria('timeup');
            break;
        }
        break;
    }
  }

  //URL Dinamica de acuerdo a la encuesta seleccionada
  cambiarCategoria(categoria: string): void {
    this.router.navigate([`home/encuesta/${categoria}`]);
    console.log('URL: '+categoria)
  }
  //Selecciona el Nombre de la Encuesta y Categoria
  seleccionarSubCategoria(encuesta: String){
    this.serviceApi.seleccionarEncuesta(encuesta);
    this.categoriaSeleccionada.emit(encuesta);
    localStorage.setItem('subCatSeleccionada', encuesta.toString());
    console.log(`Subcategoria seleccionada en DiagnosticComponent: ${encuesta}`); //
Debug
  }
  seleccionarCategoria(categoria: string){
    this.serviceApi.seleccionarCategoria(categoria);
    console.log('Categoria recibida en EncuestaComponent: ',categoria);
  }
```

```
  //Asginacion URL
  url(esfera: String, url: string): string{
    return esfera.toLowerCase() + '-' + url + '.php';
  }
}
```

Código de la ventana encuesta

ESTRUCTURA HTML

```
<div class="tituloEncuesta">
  <h1>Encuesta {{name}}</h1>
</div>
<!--
<div class="iframe">
  <iframe [src]="safeUrl" frameborder="0" class="ventana"></iframe>
</div>-->
<!-- <app-diagnostic
(categoriaSeleccionada)="cambiarCategoria($event.toString())"></app-diagnostic> -->
 <!-- ENCUESTA COGNITIVA -->
<app-encuesta-cog #encuestaComponent *ngIf="categoria == 'cognitiva'"></app-encuesta-
cog>
<!-- ENCUESTA FUNCIONAL -->
<app-category1 *ngIf="categoria == 'funcional'"></app-category1>
<!-- ENCUESTAS DE LA CATEGORIA AFECTIVA -->
<app-encuesta-afc *ngIf="subCat == 'gds15'"></app-encuesta-afc>
<app-encuesta-beck-anxiety *ngIf="subCat == 'ansbeck'"></app-encuesta-beck-anxiety>
<app-encuesta-sad-persons *ngIf="subCat == 'sadperson'"></app-encuesta-sad-persons>
<app-encuesta-ces-d7 *ngIf="subCat == 'cesd7'"></app-encuesta-ces-d7>
<app-encuesta-phq9 *ngIf="subCat == 'phq9'"></app-encuesta-phq9>
<app-encuesta-escala-soledad *ngIf="subCat == 'soledad3'"></app-encuesta-escala-
soledad>
<app-encuesta-gai-sf *ngIf="subCat == 'gaisf'"></app-encuesta-gai-sf>
<app-encuesta-cornell *ngIf="subCat == 'cornell'"></app-encuesta-cornell>
<app-encuesta-okeeffe *ngIf="subCat == 'cortdepresion'"></app-encuesta-okeeffe>
```

ESTRUCTURA SCSS

```
.ventana{
  width: 100%;
  margin: auto;
}
.iframe{
  width: 100%;
  max-width: 720px;
  margin: auto;
  height: 100vh;
}
.tituloEncuesta{
  width: max-content;
  background: #e5e8e8;
  margin: auto;
  padding: 15px 15px 5px 15px;
}
```

ESTRUCTURA TYPESCRIPT

```
import { Component, OnDestroy, OnInit, ViewChild } from '@angular/core';
import { DomSanitizer, SafeResourceUrl, SafeUrl } from '@angular/platform-browser';
import { ActivatedRoute, Route, Router } from '@angular/router';
import { ApiService } from 'src/app/Service/api.service';
import { categoria } from 'src/app/Shared/Data';
import { EncuestaCogComponent } from '../../Diagnosticos/Cognitiva/encuesta-
cog/encuesta-cog.component';

@Component({
  selector: 'app-encuesta',
  templateUrl: './encuesta.component.html',
  styleUrls: ['./encuesta.component.scss']
})

export class EncuestaComponent implements OnInit, OnDestroy{
  //Recibe Parametro ENtrante del Componente
  @ViewChild('encuestaComponent') encuestaComponent!: EncuestaCogComponent;

  name?: String; //Nombre Categoria
  categoria?: String; // Categoria
  subCat?: String; // Subcategoria
  public safeUrl!: SafeResourceUrl;
  private  BaseURL?: String = 'http://localhost:4200/'; //URL Fijo

  constructor(private router: Router, private serviceApi: ApiService, private route:
ActivatedRoute, private sanitazer: DomSanitizer) {

  }

  ngOnInit(): void {
    // Obtener el parámetro 'categoria' desde la URL (URL Dinamica)
      this.categoria = this.route.snapshot.paramMap.get('categoria')!;

      // Recibir en nombre Clave del componente Diagnostico
      this.serviceApi.categoriaSeleccionada$.subscribe(cat => {
        this.categoria = cat; //Recibe clave de Categoria
        this.updateTitle(cat);
        console.log(`Categoria recibida en EncuestaComponent: ${this.categoria,' | ',
cat}`); // Debug
      });

      this.serviceApi.selectEncuest$.subscribe(subC => {
        this.subCat = subC;
        console.log('Subcategoria recibida en EncuestaComponent: ',subC);
      });

      document.addEventListener('contextmenu',this.disableRightClick);

  }
```

```
  disableRightClick(event: MouseEvent){
    event.preventDefault();
  }

  ngOnDestroy(): void {
    localStorage.removeItem('subCatSeleccionada'); // Limpia selección SubCategoria
    localStorage.removeItem('encuestaSeleccionada'); // Limpia selección SubCategoria
    localStorage.removeItem('categoriaSeleccionada')// Limpiar seleccion Categoria
    document.removeEventListener('contextmenu', this.disableRightClick);
  }

  cambiarCategoria(categoria: string) {
    this.encuestaComponent.actualizarEncuesta(categoria);
    this.name = categoria;
  }

  updateTitle(categoria: String): void{
    switch (categoria) {
      case 'cognitiva':
        this.name = 'COGNITIVA';
        break;
      case 'afectiva':
        this.name = 'AFECTIVA';
        break;
      case 'funcional':
        this.name = 'FUNCIONAL';
        break;
      default:
        this.name = 'None'; // Valor predeterminado
        break;
    }
  }
}
```

Código del apartado Resultado

ESTRUCTURA HTML

```
<div class="resultado-container">
  <p class="texto">Puntuacion del Resultado recopilado en base de las Respuestas
Seleccionadas</p>
  <h2>Resultado de la Encuesta: {{ nombreEncuesta }}</h2>
  <p class="resultadoPuntaje"><strong>Puntaje Total:</strong> {{ puntaje }}
puntos</p>
  <p class="resultadoPuntaje"><strong>Nivel de Dependencia:</strong> {{ porcentaje
}}%</p>
  <!-- <p><strong>Nivel de Dependencia:</strong> {{ obtenerNivel() }}</p> -->
  <p class="resultadoPuntaje"><strong>Observación:</strong> {{ observacion }}</p>
</div>

<div class="d-grid gap-2 col-6 botones">
  <!-- <button class="btn btn-primary" type="button"
(click)="funcionBoton(1)">Encuesta</button> -->
```

```
  <button class="btn btn-secondary" type="button"
(click)="funcionBoton(2)">Categoria</button>
</div>
```

ESTRUCTURA SCSS

```
.resultado-container {
    max-width: 600px;
    margin: 0 auto;
    padding: 20px;
    border-radius: 8px;
    background-color: #f5f5f5;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    text-align: center;

    h2 {
      font-size: 1.5rem;
      margin-bottom: 15px;
    }

    .resultadoPuntaje {
      font-size: 1rem;
      margin: 10px 0;

      strong {
        color: #333;
      }
    }

    /* Estilos responsivos */
    @media (max-width: 720px) {
      h2 {
        font-size: 1.3rem;
      }
      p {
        font-size: 0.9rem;
      }
    }

    @media (max-width: 480px) {
      h2 {
        font-size: 1.2rem;
      }
      p {
        font-size: 0.85rem;
      }
    }
  }
.botones{
  margin: 2rem auto;
  width: 100%;
  max-width: 480px;
}
```

```
.texto{
  background: #1DD4FF;
  width: 100%;
  padding: 12px;
  color: #FFFFFF;
  border-radius: 5%;
  font-size: 1.2rem;
}
```

ESTRUCTURA TYPESCRIPT

```
import { Component, Input, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';

@Component({
  selector: 'app-resultados',
  templateUrl: './resultados.component.html',
  styleUrls: ['./resultados.component.scss']
})
export class ResultadosComponent implements OnInit{
  nombreEncuesta: string = ''; //Encuesta
  puntaje: number = 0; //Control del Puntaje
  observacion: string = ''; //Observacion
  porcentaje: number = 0;

  constructor(private router: Router, private route: ActivatedRoute) {
    //const navigation = this.router.getCurrentNavigation();
    //this.encuestas = navigation?.extras.state?.['encuestas'] || [];
  }

  ngOnInit(): void {
    // Obtener los puntos desde los queryParams
    this.route.queryParams.subscribe(params => {
      this.puntaje = params['puntaje'] || 0;  // Si no hay puntos, se asigna 0
      this.nombreEncuesta = params['nameEncuesta'] || '';
      this.porcentaje = params['porcentaje'] || '';
      this.observacion = params['observacion'] || '';
    });
  }

  funcionBoton(n: number): void{
    switch(n){
      case 1:
        this.router.navigate(['home/encuesta-cog']);
        break;
      case 2:
        this.router.navigate(['home/diagnostic']);
        break;
      case 3:
        this.router.navigate(['home/encuesta-afc']);
        break;
    }
```

```
  }
}
```

Código de las encuestas

Encuesta de Categoría Cognitiva

ESTRUCTURA HTML

```
<!-- Encuesta 4AT -->
<div class="4at containerEncuesta" *ngIf="categoria == '4at'">
  <div class="title">
    4AT
  </div>
  <div class="p">Esto incluye que puedan presentarse somnolencia o
ahitado/hiperactivo.</div>
  <div class="opciones">
    <label for="">
      <input type="checkbox"> Normal
    </label>
    <label for="">
      <input type="checkbox" name="" id="">Somnolencia Breve &#60;10 Segundos
    </label>
    <label for="">
      <input type="checkbox">Claramente Normal
    </label>
  </div>
</div>

<!-- Encuesta CAM -->
<div class="cam containerEncuesta" *ngIf="categoria == 'cam'">
  <div class="title">
    CAM
  </div>
  <div class="opciones">
    <p class="texto">
      El Diagnostico de Delirium por CAM requeire la presencia de AMBAS
caracteristicas A + B
    </p>
    <table class="tabla-delirium">
      <thead>
        <tr>
          <th colspan="3">
            El diagnóstico de Delirium por CAM requiere la presencia de AMBAS
características A + B
          </th>
        </tr>
      </thead>
      <tbody>
        <!-- Característica A -->
        <tr>
          <td><strong>A. Comienzo agudo y curso fluctuante</strong></td>
```

```
          <td colspan="2">¿Hay evidencia de un cambio agudo en el estado mental del
paciente distinto a su basal?</td>
        </tr>
        <tr>
          <td></td>
          <td colspan="2">El comportamiento anormal...
            <div>
              <label><input type="radio" name="comportamiento" /> a. Va y
viene?</label>
            </div>
            <div>
              <label><input type="radio" name="comportamiento" /> b. Fluctúa durante
el día?</label>
            </div>
            <div>
              <label><input type="radio" name="comportamiento" /> c. Aumenta o
disminuye en severidad?</label>
            </div>
          </td>
        </tr>
        <!-- Característica B -->
        <tr>
          <td><strong>B. Inatención</strong></td>
          <td colspan="2">El paciente...
            <div>
              <label><input type="radio" name="inatencion" /> a. Tiene dificultad
para enfocar su atención?</label>
            </div>
            <div>
              <label><input type="radio" name="inatencion" /> b. Se distrae
fácilmente?</label>
            </div>
            <div>
              <label><input type="radio" name="inatencion" /> c. Tiene dificultad
para seguir una conversación?</label>
            </div>
          </td>
        </tr>
        <!-- Separador -->
        <tr>
          <th colspan="3">Y la presencia de al menos una de las características C o
D</th>
        </tr>
        <!-- Característica C -->
        <tr>
          <td><strong>C. Pensamiento desorganizado</strong></td>
          <td colspan="2">¿El pensamiento del paciente es...
            <div>
              <label><input type="radio" name="pensamiento" /> a.
Desorganizado?</label>
            </div>
            <div>
```

```
              <label><input type="radio" name="pensamiento" /> b.
Incoherente?</label>
            </div>
            <p>Por ejemplo, ¿ha visto usted que el paciente tenga...</p>
            <div>
              <label><input type="radio" name="pensamientoEjemplo" /> a. Discurso
divagante / conversaciones
                 irrelevantes?</label>
            </div>
            <div>
              <label><input type="radio" name="pensamientoEjemplo" /> b. Cambios de
tema impredecibles durante una
                 conversación?</label>
            </div>
            <div>
              <label><input type="radio" name="pensamientoEjemplo" /> c. Flujo de
ideas ilógico o poco claro?</label>
            </div>
          </td>
        </tr>
        <!-- Característica D -->
        <tr>
          <td><strong>D. Alteración en el nivel de conciencia</strong></td>
          <td colspan="2">¿Generalmente cómo consideraría el nivel de conciencia del
paciente?
            <div>
              <label><input type="radio" name="conciencia" /> 1. Alerta
(normal)</label>
            </div>
            <div>
              <label><input type="radio" name="conciencia" /> 2. Vigilante
(hiperalerta)</label>
            </div>
            <div>
              <label><input type="radio" name="conciencia" /> 3. Letárgico
(somnoliento pero fácil de despertar)</label>
            </div>
            <div>
              <label><input type="radio" name="conciencia" /> 4. Estuporoso
(difícilmente despertable)</label>
            </div>
            <div>
              <label><input type="radio" name="conciencia" /> 5. Comatoso (no
despertable)</label>
            </div>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</div>
```

```
<!-- Encuesta CAM-ICU -->
<div class="camicu containerEncuesta" *ngIf="categoria == 'camicu'">
  <div class="title">
    CAM-ICU
  </div>
  <div class="encuesta">
    <div class="instruccion">Confusion Assessments Method for the ICU &#40;CAM-
ICU&#41;</div>
    <form> <!--(ngSubmit)="enviarEncuesta()">-->
      <div *ngFor="let pregunta of preguntasCamIcu; let i = index" class="pregunta">
        <label [for]="'pregunta-' + i" class="preguntas">{{ pregunta.texto }}</label>
        <div class="opciones">
          <div class="opcSi" (click)="pregunta.respuesta = 'si'"
[class.selected]="pregunta.respuesta == 'si'">
            <input type="radio" [name]="'pregunta-' + i" [id]="'si-' + i" value="si"
[(ngModel)]="pregunta.respuesta" required />
            <label [for]="'si-' + i">Si</label>
          </div>
          <div class="opcNo" (click)="pregunta.respuesta = 'no'"
[class.selected]="pregunta.respuesta == 'no'">
            <input type="radio" [name]="'pregunta-' + i" [id]="'no-' + i" value="no"
[(ngModel)]="pregunta.respuesta" required />
            <label [for]="'no-' + i">No</label>
          </div>
        </div>
        <!-- Mensaje de error para preguntas no respondidas -->
        <div *ngIf="!pregunta.respuesta && showErrors" class="text-danger small">
          Por favor selecciona una respuesta.
        </div>
      </div>
    </form>
  </div>
</div>

<!-- Encuesta AWOL -->
<div class="awol containerEncuesta" *ngIf="categoria == 'awol'">
  <div class="title">Awol</div>
  <div class="intruccion">
    <p>Seleccione las letras para cambiar de <b>COLOR</b>, al estar en gris, se
considera 0 puntos, cada color es 1
      punto</p>
  </div>
  <div class="opciones">
    <p [ngClass]="{ 'text-green': isSelected['A'] }"
(click)="toggleTextColor('A')">A</p>
    <p [ngClass]="{ 'text-red': isSelected['W'] }"
(click)="toggleTextColor('W')">W</p>
    <p [ngClass]="{ 'text-purple': isSelected['O'] }"
(click)="toggleTextColor('O')">O</p>
    <p [ngClass]="{ 'text-orange': isSelected['L'] }"
(click)="toggleTextColor('L')">L</p>
  </div>
```

```
  <div class="respuesta">
    <p class="green">A: Edad &#62; 80 A&#241;os</p>
    <p class="red">W: &#191;Incapaz de deletrear la Palabre: &#34;MUNDO&#34; al
reves&#63;</p>
    <p class="purple">O: &#191;Desorientado en Ciudad, Estado, Pa&#237;s, Hospital y
Piso&#63;</p>
    <p class="orange">L: Evaluacion Enfermedad &#191;Moderado, Grave o
Moribundo&#63;</p>
  </div>
</div>

<!-- Encuesta Question Pfeiffer -->
<div class="spmsqp containerEncuesta" *ngIf="categoria == 'spmsqp'">
  <div class="title">Short Portable Mental State Questionnaire Pfeiffer</div>
  <div class="encuesta">
    <div class="instruccion">Responde la pregintas de acuerdo a la respuesta del
Paciente</div>
    <form>
      <div class="pregunta" *ngFor="let pregunta of preguntasMMSE; let i = index">
        <label [for]="'pregunta-' + i" class="preguntas">{{pregunta.texto}}</label>
        <div class="opciones">
          <div class="opcSi" (click)="pregunta.respuesta = 'si'"
[class.selected]="pregunta.respuesta == 'si'">
            <input type="radio" [name]="'pregunta-'+ i" [id]="'si-'+i" value="si"
[(ngModel)]="pregunta.respuesta" required />
            <label [for]="'si-'+ i">Si</label>
          </div>
          <div class="opcNo" (click)="pregunta.respuesta = 'no'"
[class.selected]="pregunta.respuesta == 'no'">
            <input type="radio" [name]="'pregunta-'+ i" [id]="'si-'+ i" value="no"
[(ngModel)]="pregunta.respuesta" required />
            <label [for]="'no-'+ i">No</label>
          </div>
        </div>
        <div class="text-danger samll" *ngIf="!pregunta.respuesta && showErrors">
          Por favor responde todas las preguntas
        </div>
      </div>
    </form>
  </div>
</div>

<!-- Prueba del Reloj -->
 <div class="reloj containerEncuesta" *ngIf="categoria == 'reloj'">
  <div class="title">Prueba del Reloj</div>
  <div class="instruccion">
    <p>Dibuje un reloj de manecillas, coloque todos los n&#250;meros que lleva y
marque con agujas la hora &#8220;11&#58;10&#8221;</p>
    <img src="assets/Imagenes/reloj.jpg" draggable="false" alt="" class="image">
    <p>De acuerdo al paciente, seleccione las opciones que considere que hizo
correctamente</p>
  </div>
```

```
  <div class="opciones">
    <form [formGroup]="questionnaireForm" (ngSubmit)="encuestas('reloj')">
      <!-- Pregunta 1 -->
    <div class="question">
      <p><strong>{{questions[0].text}}</strong></p>
      <div *ngFor="let option of questions[0].options" class="question1">
        <label>
          <input type="radio" [formControlName]="'question1'" [value]="option.score"
/>
          {{ option.label }}
        </label>
      </div>
    </div>
    <div *ngIf="hasError('question1')" class="text-danger small">
      Por favor selecciona una respuesta.
    </div>


    <!-- Pregunta 2 -->
    <div class="question">
      <p><strong>{{questions[1].text}}</strong></p>
      <div *ngFor="let option of questions[1].options">
        <label [ngStyle]="{ 'white-space': 'pre-line' }">
          <input type="radio" [formControlName]="'question2'"
[value]="option.score"/>
          {{ option.label }}
        </label>
      </div>
    </div>
    <div *ngIf="hasError('question1')" class="text-danger small">
      Por favor selecciona una respuesta.
    </div>

    <!-- Pregunta 3 -->
    <div class="question">
      <p><strong>{{questions[2].text}}</strong></p>
      <div *ngFor="let option of questions[2].options">
        <label>
          <input type="radio" [formControlName]="'question3'"
[value]="option.score"/>
          {{ option.label }}
        </label>
      </div>
    </div>
    <div *ngIf="hasError('question1') && showErrors" class="text-danger small">
      Por favor selecciona una respuesta.
    </div>

    </form>
  </div>
 </div>
```

```
<!-- Boton de Envio -->
<div class="boton">
  <button type="submit" class="btn btn-success"
(click)="finalizarEncuesta()">Finalizar</button>
</div>
```

ESTRUCTURA SCSS

```
//Titulo General
.title {
  background: #dedede;
  opacity: 0.7;
  padding: 0.5rem;
  text-align: center;
  margin: 1rem auto;
}
/* Estilos del Contenedor de Encuesta */
.containerEncuesta{
  width: 100%;
  max-width: 1000px;
  margin:  1rem auto;
}

// Boton Finalizar
.boton{
  width: 100%;
  max-width: max-content;
  margin: 1rem auto;

  button{
    padding: 10px 30px;
  }
}

.opciones label,
.opciones input {
  display: block;
  cursor: pointer;
  width: max-content;
}

/* Estilos de la Encuesta CAM */
.cam .tabla-delirium {
  width: 100%;
  border-collapse: collapse;
}

.cam .tabla-delirium th,
.cam .tabla-delirium td {
  border: 1px solid #ccc;
  padding: 10px;
  text-align: left;
  vertical-align: top;
```

```
}
.cam .tabla-delirium th {
  background-color: #ddd;
  font-weight: bold;
  text-align: center;
}
.cam .tabla-delirium td div {
  margin-bottom: 5px;
}

.cam .tabla-delirium label {
  font-weight: normal;
}

/* Estilos de Encuesta AWOL */
.awol .text-green {
  color: green;
  font-weight: bold;
  border: green solid 5px;
}

.awol .text-orange {
  color: orange;
  font-weight: bold;
  border: orange solid 5px;
}

.awol .text-purple {
  color: purple;
  font-weight: bold;
  border: purple solid 5px;
}
.awol .text-red {
  color: red;
  font-weight: bold;
  border: red solid 5px;
}
.awol .opciones p{
    font-size: 3rem;
    margin: 10px 2rem;
    padding: 1rem;
    cursor: pointer;
    border-radius: 50%;
    opacity: 0.5;
}
.awol .opciones p:hover{
  opacity: 1;
}
.awol .opciones{
    display: flex;
    width: 100%;
    max-width: 500px;
```

```
    margin: 25px auto;
}
.awol .respuesta p{
    font-size: 1.1rem;
    margin: 1rem auto;
    max-width: 50%;
    padding: 10px 5px;
    font-weight: bold;
}
.awol .green{
    background: #82e0aa;
}
.awol .red{
    background: #FF6565;
}
.awol .orange{
    background: orange;
}
.awol .purple{
    background: rgba(128, 0, 128, 0.5);
}

/* Opciones a Si No a lado */
.containerEncuesta .opciones .opcSi,
.containerEncuesta .opciones .opcNo{
  display: inline-flex;
  margin: 0px 10px 0px 15px;
  cursor: pointer;
  font-size: 1.1rem;
}
.spmsqp .opciones input,
.camicu .opciones input{
  margin-right: 10px;
}

/* Estilo de Prueba reloj */
.reloj .opciones .question label{
  display: flex;
  margin: 7px 0px;
}
.reloj .opciones .question input{
  margin-right: 1rem;
}
.reloj .opciones .question label input[type="radio"]{
  margin-top: 0.2rem;
  align-self: start;
}
.reloj {
  .instruccion p{
    font-weight: bold;
    font-size: 1.3rem;
  }
```

```
  .image{
    width: 35%;
    margin: 2rem auto;
  }
  .instruccion{
    text-align: center;
    margin: 1.7rem 0px;
  }
  .opciones{
    margin-top: 1rem;
  }
}
```

ESTRUCTURA TYPESCRIPT

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, Validator, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { left } from '@popperjs/core';
import { float, FLOAT } from 'html2canvas/dist/types/css/property-descriptors/float';
import { ApiService } from 'src/app/Service/api.service';
import { PreguntaDosOpc, Question } from 'src/app/Shared/Data';

@Component({
  selector: 'app-encuesta-cog',
  templateUrl: './encuesta-cog.component.html',
  styleUrls: ['./encuesta-cog.component.scss']
})
export class EncuestaCogComponent implements OnInit{
  isSelected: { [key: string]: boolean } = {};// Validacion de color de AWOL
false/true
  puntos: number = 0; //Puntaje
  categoria: String = 'Undefinid';//Recibir nombre clave
  cons: String[] = ['Entrastes en: ', 'Salida de: '];
  nameEncuesta?: String;
  observacion: String ='';

  //Preguntas Si No
  preguntasCamIcu: PreguntaDosOpc[] = [
    {texto: 'Existencia de Cambio agudo del Estado Mental', respuesta:''},
    {texto: 'Estado Mental del Paciente ha fluctuado durante las últimas 24hrs',
respuesta:''},
    {texto: 'Apretar Mano (Médico) cuando el paciente diga A', respuesta:''},
    {texto: 'Diga al Paciente que Deletree C-A-S-A-B-L-A-N-C-A', respuesta:''},
    {texto: 'Vigile al pacinete y verifique: Alerta de RASS0', respuesta:''},
    {texto: 'Las Piedras flotan en el Agua', respuesta:''},
    {texto: 'Hay Peces en el mar', respuesta: '' },
    {texto: '1kg pesa más que 2kg', respuesta: '' },
    {texto: 'Los martillos sirven para poner clavos', respuesta: '' }
  ]
  preguntasMMSE: PreguntaDosOpc[] = [
    {texto: 'Que fecha es hoy (dia, mes, año)', respuesta:''},
    {texto: 'Que dia de la semana es hoy', respuesta:''},
```

```
    {texto: 'Donde estamos ahora (Lugar o Edificio)', respuesta:''},
    {texto: 'Cual es su numero de telefono (O idreccion si no tiene uno)',
respuesta:''},
    {texto: 'Que edad tiene', respuesta:''},
    {texto: 'La fecha cuando nacio', respuesta:''},
    {texto: 'Como se llama el presidente del Gobierno', respuesta: '' },
    {texto: 'Como se llama el anterior presidente', respuesta: '' },
    {texto: 'Primer apellido de su Madre', respuesta: '' },
    {texto: 'Pudo restar de tres en tres desde veite', respuesta: ''}
  ]
  //Preguntas Reloj
  questionnaireForm: FormGroup;
  questions: Question[] = [
    { text: 'Dibujo del Reloj',
      options: [
        { label: 'El círculo está bien dibujado o cercanamente a lo redondo', score:
2 },
        { label: 'El círculo está dibujado, pero de manera irregular', score: 1 },
        { label: 'Figura irregular o no compatible con un semi-círculo', score: 0 }
      ]
    },
    { text: 'Presencia y secuencia de números',
      options: [
        { label: 'Todos los números están presentes. Puede aceptarse un error mínimo
en la disposición espacial.', score: 4 },
        { label: 'Todos los números están presentes. Errores en la disposición
espacial.', score: 3 },
        { label: 'Algunas de las Siguientes:\nNúmeros faltantes o adicionales aunque
sin distorsiones groseras de los números restantes.\nLos números están ubicados en
sentido anti-horario.\nLos números están presentes pero hay una seria alteración en
la disposición general.', score: 2 },
        { label: 'Números faltantes o adicionales y errores espaciales serios',
score: 1 },
        { label: 'Ausencia o pobre representación de los números.', score: 0 }
      ]
    },
    { text: 'Presencia y ubicación de las agujas',
      options: [
        { label: 'Las agujas están en la posición correcta y la diferencia en tamaño
está respetada.', score: 4 },
        { label: 'Errores discretos en la representación de las agujas o la
diferencia de tamaño entre las agujas.', score: 3 },
        { label: 'Errores mayores en la ubicación de las agujas (de manera
significativa, incluyendo 10 a 11).', score: 2 },
        { label: 'Se dibuja solamente UNA aguja o el dibujo de ambas agujas es
notoriamente pobre.', score: 1 },
        { label: 'No se dibujan las agujas o se dibujan varias de manera
perseverativa.', score: 0 }
      ]
    }
  ];
  showErrors: boolean = false;
```

```
  constructor(private router: Router, private categoriaEncuestaComponenet:
ApiService, private fb: FormBuilder){
    this.questionnaireForm = this.fb.group({
      question1: [null, Validators.required],
      question2: [null, Validators.required],
      question3: [null, Validators.required],
    });

  }

  ngOnInit(): void {
      this.categoriaEncuestaComponenet.selectEncuest$.subscribe(subCategoria => {
        this.categoria = subCategoria;
      });
      console.log(this.categoria);
  }

  //Metodo del boton Finalizar
  finalizarEncuesta(){
    //this.puntos = 0;
    this.encuestas(this.categoria);// Entrar a la encuesta segun sea seleccionada

  }

  //Receptaculo Categoria
  actualizarEncuesta(encuestaS: String){
    this.categoria = encuestaS;
  }

  /* METODOS DE LAS ENCUESTAS PARA CALCULAR LOS PUNTOS OBTENIDO */
  encuestas(enc: String){
    switch(enc){
      case 'camicu':
        console.log(this.cons[0], this.categoria);
        const todasRespondidas = this.preguntasCamIcu.every(p => p.respuesta != null
&& p.respuesta != undefined && p.respuesta != '');
        console.log('Preguntas CAM-ICU: '+ todasRespondidas);

        if (!todasRespondidas) {
          // Mostrar errores si hay preguntas sin responder
          this.showErrors = true;
          return;
        } else{ this.showErrors = true}

        let puntajeSi = 0;
        for (const pregunta of this.preguntasCamIcu) {
          if (pregunta.respuesta == 'si') {
            puntajeSi++;
          }
        }
```

```
        this.encuestaResulto(puntajeSi, 'CAM-ICU','Sin
observaciones',(puntajeSi/9)*100);
        console.log(this.cons[1], 'CAM-ICU');
        break;
      case 'awol':
        var del: String = 'Riesgo de Delirium: '; //Validacion riesgo
        console.log(this.cons[0], this.categoria); //Verificar si funciona en consola
        //Calcular la observacion segun el puntaje
        if (this.puntos >= 4) {
          this.observacion = del + '64%';
        } else if (this.puntos == 3) {
          this.observacion = del + '20%';
        } else if (this.puntos == 2) {
          this.observacion = del + '14%';
        } else if(this.puntos == 1){
          this.observacion = del + '4%';
        } else {
          this.observacion = del + '2%';
        }

        this.showErrors = true;// An finalizar marca verdadero si, dependiendo si
selecciona la letra
        console.log(this.cons[1], 'AWOL');//Verificar que si hay salida
        this.encuestaResulto(this.puntos, 'AWOL', this.observacion,
(this.puntos/4)/100); //Envia los parametros al metodo
        break;
      case 'spmsqp':
        var err: String = 'Errores Obtenido: ';//Acortar observacion
        console.log(this.cons[0], this.categoria);
        const questionAll = this.preguntasMMSE.every(p => p.respuesta != null &&
p.respuesta != undefined && p.respuesta != '');// No se acepta respuestas sin
responder
        console.log('Pregunats MMSE: '+ questionAll);
        //Validar que todas las opciones sean seleccionadas
        if(!questionAll){
          this.showErrors = true;
          return;
        } else { this.showErrors = true}
        // Sumar si el puntaje es Si
        let respuestaNo = 0;
        for(const pregunta of this.preguntasMMSE){
          if(pregunta.respuesta == 'no'){
            respuestaNo++;
          }
        }
        //Calculo de observacion segun el puntaje
        if(respuestaNo >= 8){
          this.observacion = err + 'Importante, deterioro Congnitivo';
        } else if(respuestaNo >= 5 && respuestaNo <= 7){
          this.observacion = err + 'Moderado, Deteriodocognitivo, patologico';
        } else if(respuestaNo <= 2){
          this.observacion = err + 'Normal, Sin deteriodo'
```

```
        } else {
          this.observacion = err + 'Leve, Deteriodo cognitivo'
        }

        this.encuestaResulto(respuestaNo, 'Question Pfeiffer', this.observacion,
(respuestaNo/10)*100); //Envio de los parametros
        console.log(this.cons[1], 'spmsqp');
        break;
      case 'reloj':
        this.showErrors = true;
        console.log(this.cons[0], this.categoria);
        if(this.questionnaireForm.valid){
          this.puntos = 0; //Reiniciar los puntos

          // Iterar sobre las claves del formulario
          for(const key in this.questionnaireForm.controls){
            if(this.questionnaireForm.controls[key].value != null &&
this.questionnaireForm.controls[key].value != ''){
              this.puntos += this.questionnaireForm.controls[key].value;
            }
          }
          this.showErrors = false;
        }

        if(this.puntos >= 9){
          this.observacion = 'NORMAL';
        } else if(this.puntos == 8){
          this.observacion = 'DEFICIT LIMITE';
        } else if(this.puntos == 6 || this.puntos == 7){
          this.observacion = 'DEFICIT LEVE';
        } else if(this.puntos == 4 || this.puntos == 5){
          this.observacion = 'DEFICIT MODERADO';
        } else {
          this.observacion = 'DEFICIT SEVERO';
        }

        //Control de Envio si esta respondido la encuesta
        if(!this.showErrors){
          console.log('Entrada If del Reloj');
          this.showErrors = true; //Convertirse en verdadero antes de llamar al
metod, para enviar al siguiente componente
          this.encuestaResulto(this.puntos, 'Prueba de Reloj',this.observacion,
(this.puntos/10)*100);
        }
        console.log(this.cons[1], 'Prueba Reloj');
        break;
    }

  }

  /* ALGUNOS METODOS POR CADA ENCUESTA A REALIZAR */
```

```
   //Envia los parametros al Componente Resultado segun reciba
   encuestaResulto(puntos: number, nameEncuesta: String, observacion: String,
porcentaje: FLOAT){
    if(this.showErrors){
     // Al navegar, enviamos los puntos al componente de resultado
     this.router.navigate(['home/resultado'], {queryParams: {
       puntaje: puntos, //Puntaje Obtenido
       nameEncuesta: nameEncuesta, //Nombre de la Encuesta
       porcentaje: porcentaje.toFixed(2),
       observacion: observacion //Observaciones
     }});
   }
 }

  // Función para alternar el color
  toggleTextColor(textKey: string): void {
    this.isSelected[textKey] = !this.isSelected[textKey];

    //Al estar en color se suma 1 punto, en caso contrario 0
    if (this.isSelected[textKey]) {
      this.puntos += 1;  // Si se selecciona, suma un punto
    } else {
      this.puntos -= 1;  // Si se deselecciona, resta un punto
    }
  }

  // Método para verificar que la respuesta de la pregunta este seleccionado
  hasError(fieldName: string): boolean {
    const control = this.questionnaireForm.get(fieldName);
    return !!control?.invalid && (control?.touched || this.showErrors);
  }

}
```

Encuesta de Categoría Funcional

ESTRUCTURA HTML

```
<!-- Encuesta de KATZ -->
<div class="katz containerEncuesta opcmult" *ngIf="encuestaSelect == 'katz'">
  <div class="titulo">
    <h2>Indice de KATZ</h2>
  </div>
  <div class="opciones">
    <div *ngFor="let question of preguntasKATZ; let i = index" class="question">
      <p class="preguntas">{{ question.text }}</p>
      <div *ngFor="let option of question.options">
        <label>
          <input type="radio" name="question{{i}}" [value]="option.score"
(change)="selectedOptions[i] = option.score; errors[i] = false"/>
          {{ option.label }}
        </label>
```

```
      </div>

      <!-- Mensaje de error específico para la pregunta -->
    <div *ngIf="errors[i]" class="alert alert-danger mt-2">
      Por favor, selecciona una opción para esta pregunta.
    </div>
    </div>

  </div>
</div>

<!-- Encuesta de Barthel -->
<div class="barthel containerEncuesta opcmult" *ngIf="encuestaSelect == 'barthel'">
  <div class="titulo">
    <h2>Indice de Barthel</h2>
  </div>
  <div class="opciones">
    <p class="instrucciones">Seleccione</p>
    <div *ngFor="let pregunta of preguntasBarthel; let i = index">
      <p class="preguntas">{{ pregunta.text }}</p>
      <div *ngFor="let opcion of pregunta.options">
        <label>
          <input type="radio" name="pregunta{{i}}" [value]="opcion.score"
(change)="seleccionarRespuesta(i, opcion.score)"/> <!--
[(ngModel)]="pregunta.respuestaSeleccionada"/>-->
          {{ opcion.label }}
        </label>
      </div>
    </div>
    <div *ngIf="mensajeError != ''" class="alert alert-danger">
      {{ mensajeError }}
    </div>
  </div>
</div>

<!-- Encuesta de Lawton y Brody -->
<div class="lawton containerEncuesta opcmult" *ngIf="encuestaSelect == 'lawton'">
  <div class="titulo">
    <h2>Lawton y brody</h2>
  </div>
  <div class="opciones">
      <div *ngFor="let question of preguntasLowton; let i = index" class="question
mb-3">
        <p class="preguntas">{{ question.text }}</p>
        <div *ngFor="let option of question.options" class="form-check">
          <label>
            <input type="radio" name="question{{i}}" [value]="option.score"
(change)="optionL[i] = option.score; errorsLowton[i] = false"/>
            {{ option.label }}
          </label>
        </div>
```

```
        <div *ngIf="errorsLowton[i]" class="alert alert-danger mt-2">
          Por favor, selecciona una opción para esta pregunta.
        </div>
      </div>
  </div>
</div>

<!-- Encuesta de Ensrud -->
<div class="ensrud containerEncuesta" *ngIf="encuestaSelect == 'ensrud'">
  <div class="titulo">
    <h2>Criterios de Ensrud</h2>
  </div>
  <div class="opciones">
    <p class="instrucciones">Seleccione la respuesta segun considere que sea posible
o que aplique segun su capacidad</p>
    <div *ngFor="let pregunta of preguntasEnsrud; let i = index">
      <label>
        <input type="checkbox" class="form-check-input"
[checked]="pregunta.seleccionada" (change)="toggleCheckbox(i, $event)"/>
        {{ pregunta.text }}
      </label>
    </div>
  </div>
</div>

<!-- Encuesta FRAIL -->
 <div class="frail containerEncuesta" *ngIf="encuestaSelect == 'frail'">
  <div class="titulo">
    <h2>FRAIL</h2>
  </div>
  <div class="opciones">
    <div class="container mt-4">
      <!-- Tabla con colores personalizados -->
      <form>
        <table class="table table-bordered">
          <thead class="table-light">
            <tr class="table-header">
              <th class="text-center">Observación</th>
              <th class="text-center">Puntuación</th>
            </tr>
          </thead>
          <tbody>
            <!-- FATIGA -->
            <tr class="row-blue">
              <td>
                <strong>[Fatigue (fatiga)]</strong><br />
                En las últimas 4 semanas, ¿Qué tanto tiempo se sintió?
              </td>
              <td>
                <div class="form-check">
                  <input type="radio" id="todoTiempo" name="fatigue" value="Todo el
tiempo" class="form-check-input" (change)="onAnswerChange('fatigue', 1)"/>
```

```
                  <label for="todoTiempo" class="form-check-label">Todo el
tiempo</label>
                </div>
                <div class="form-check">
                  <input type="radio" id="mayorParte" name="fatigue" value="La mayor
parte del tiempo" class="form-check-input" (change)="onAnswerChange('fatigue', 1)"/>
                  <label for="mayorParte" class="form-check-label">La mayor parte del
tiempo</label>
                </div>
                <div class="form-check">
                  <input type="radio" id="algoTiempo" name="fatigue" value="Algo de
tiempo" class="form-check-input" (change)="onAnswerChange('fatigue', 0)" />
                  <label for="algoTiempo" class="form-check-label">Algo de
tiempo</label>
                </div>
                <div class="form-check">
                  <input type="radio" id="muyPoco" name="fatigue" value="Muy poco
tiempo" class="form-check-input" (change)="onAnswerChange('fatigue', 0)" />
                  <label for="muyPoco" class="form-check-label">Muy poco
tiempo</label>
                </div>
                <div class="form-check">
                  <input type="radio" id="nadaTiempo" name="fatigue" value="Nada de
tiempo" class="form-check-input" (change)="onAnswerChange('fatigue', 0)" />
                  <label for="nadaTiempo" class="form-check-label">Nada de
tiempo</label>
                </div>
              </td>
            </tr>

            <!-- RESISTENCIA -->
            <tr class="row-gray">
              <td>
                <strong>[Resistance (resistencia)]</strong><br />
                ¿Tiene dificultad para subir 10 escalones (una escalera)?
              </td>
              <td>
                <div class="form-check">
                  <input type="radio" id="resistanceSi" name="resistance" value="1"
class="form-check-input" (change)="onAnswerChange('resistance', 1)"/>
                  <label for="resistanceSi" class="form-check-label">Sí</label>
                </div>
                <div class="form-check">
                  <input type="radio" id="resistanceNo" name="resistance" value="0"
class="form-check-input" (change)="onAnswerChange('resistance', 0)"/>
                  <label for="resistanceNo" class="form-check-label">No</label>
                </div>
              </td>
            </tr>

            <!-- ACTIVIDAD -->
            <tr class="row-blue">
```

```
              <td>
                <strong>[Aerobic (actividad aeróbica)]</strong><br />
                ¿Tiene dificultad para caminar 100 metros (dos cuadras) sin
descansar?
              </td>
              <td>
                <div class="form-check">
                  <input type="radio" id="aerobicSi" name="aerobic" value="1"
class="form-check-input" (change)="onAnswerChange('aerobic', 1)"/>
                  <label for="aerobicSi" class="form-check-label">Sí</label>
                </div>
                <div class="form-check">
                  <input type="radio" id="aerobicNo" name="aerobic" value="0"
class="form-check-input" (change)="onAnswerChange('aerobic', 0)"/>
                  <label for="aerobicNo" class="form-check-label">No</label>
                </div>
              </td>
            </tr>

            <!-- ENFERMEDADES -->
            <tr class="row-gray">
              <td>
                <strong>[Illnesses (enfermedades)]</strong><br />
                ¿Algún doctor o médico le ha comentado que tiene [mencionar la
enfermedad]?
              </td>
              <td>
                <div class="form-check">
                  <input type="radio" id="illnessesSi" name="illnesses" value="1"
class="form-check-input" (change)="onAnswerChange('illnesses', 1)"/>
                  <label for="illnessesSi" class="form-check-label">Sí</label>
                </div>
                <div class="form-check">
                  <input type="radio" id="illnessesNo" name="illnesses" value="0"
class="form-check-input" (change)="onAnswerChange('illnesses', 0)"/>
                  <label for="illnessesNo" class="form-check-label">No</label>
                </div>
              </td>
            </tr>

            <!-- PESO -->
            <tr class="row-blue">
              <td>
                <strong>[Lost of weight (pérdida de peso)]</strong><br/>
                El paciente no debe portar los zapatos al momento de ser pesado
              </td>
              <td>
                <label for="pesoHaceUnAnio" class="form-label">&#191;Cuánto pesaba
hace un a&#241;o&#63;</label>
                <input type="text" id="pesoHaceUnAnio" class="form-control mb-2"
placeholder="Peso hace un año" maxlength="4" (input)="validateInput($event,
'pesoYear')"/>
```

```
                <!-- Mensaje de error -->
                <div *ngIf="pesoHaceUnAnioError" class="text-danger mt-1">
                  {{ pesoHaceUnAnioError }}
                </div>
                <label for="pesoActual" class="form-label">&#191;Cuánto pesa
ahora&#63;</label>
                <input type="text" id="pesoActual" class="form-control"
placeholder="Peso actual" maxlength="4" (input)="validateInput($event,
'pesoActual')"/>
                <!-- Mensaje de error -->
                <div *ngIf="pesoActualError" class="text-danger mt-1">
                  {{ pesoActualError }}
                </div>
              </td>
            </tr>
          </tbody>
        </table>
        <div *ngIf="mensajeError" class="alert alert-danger mt-3">
          {{ mensajeError }}
        </div>
      </form>
    </div>


  </div>
 </div>

 <!-- Time Up and Go -->
 <div class="cronometro containerEncuesta" *ngIf="encuestaSelect == 'timeup'">
  <div class="titulo">
    <h2>Time UP and DOWN</h2>
  </div>
  <div class="instrucciones">
    <p>La persona puede usar su calzado habitual y puede utilizar cualquier
dispositivo de ayuda que normalmente usa.</p>
    <ul>
      <li>Indicarle a la persona mayor, sentarse en la silla con la espalda apoyada
en el respaldo.</li>
      <li>Pídale a la persona que se levante de la silla, camine a paso normal una
distancia de 3 metros, haga que la persona de la vuelta camine nuevamente hacia la
silla y se vuelva a sentar.</li>
      <li>Mida el tiempo en que la persona mayor realiza la prueba. El cronometraje
comienza cuando la persona comienza a levantarse de la silla y termina cuando regresa
a la silla y se sienta.</li>
      <li>Dar un intento de prueba.</li>
    </ul>
    <p>El cronometraje comienza cuando la persona comienza a levantarse de la silla y
termina cuando regresa a la silla y se sienta.</p>
    <p>La persona debe de dar un intento de práctica y luego repite 3 intentos más.
Se promedian los tres ensayos reales y se promedian.</p>
  </div>
  <div class="opciones">
```

```
    <h1>{{ tiempo | date:'mm:ss.SSS':'UTC' }}</h1>
    <button (click)="iniciarPausar()" class="btn btn-primary">
      {{ corriendo ? 'Pause' : 'Start' }}
    </button>
    <button (click)="reiniciar()" class="btn btn-primary">Reset</button>
    <button (click)="capturarTiempo()" [disabled]="!corriendo" class="btn btn-
primary">Time Capture</button>
    <button (click)="deleteScore()" class="btn btn-primary">Default</button>

    <div *ngIf="capturas.length">
      <h3>Tiempos Capturados:</h3>
      <ul>
        <li *ngFor="let captura of capturas; let i = index">
          Captura {{ i + 1 }}: {{ captura }}
        </li>
      </ul>
    </div>
  </div>
  <img src="assets/Imagenes/timeupdown.png" alt="" class="time" draggable="false">
</div>
<div class="boton">
  <button type="submit" class="btn btn-success"
(click)="finalizarEncuesta()">Finalizar</button>
</div>
```

ESTRUCTURA SCSS

```
//Estilos de los titulos y contenedor
.containerEncuesta .titulo {
  width: 100%;
  max-width: 900px;
  margin: 1.5rem auto;
  background: #19c3ff;
  padding: 10px 5px;
  border-radius: 10px;
}
.containerEncuesta h2 {
  margin-left: 3rem;
  font-size: 1.7rem;
  margin-top: 5px;
  font-weight: bold;
  color: #fbfcfc;
}
.containerEncuesta{
  width: 100%;
  max-width: 1000px;
  margin: auto;
}
label{
    cursor: pointer;
    margin: 5px 0px;
}
.boton{
```

```
  width: 100%;
  text-align: center;
  margin: 1.2rem 0;
}
.boton button{
  padding: 10px 30px;
}
/* ESTILOS DE BARTHEL, LAWTON Y KATZ */
.opcmult .opciones {
  label input[type="radio"]{
    margin-top: 0.1rem;
    align-self: start;
  }
  label{
    margin-left: 1rem;
  }
  .preguntas{
    padding: 10px 15px;
    background:   #d7dbdd;
    color: rgba(0, 0, 0, 0.7);
    font-family: Arial, Helvetica, sans-serif;
    font-size: 1.3rem;
  }
}
/* ESTILOS DE FRAIL */
.frail .opciones {
  .table-header th {
    background: rgb(0, 176, 255);
    color: white;
    font-weight: bold;
    font-size: 1.3rem;
    text-align: center;
    text-transform: uppercase;
    padding: 13px 0;
  }

  /* Colores de las filas */
  .row-blue td{
    background: rgb(202, 237, 251); /* Azul claro */
  }

  .row-gray td{
    background: rgb(242, 243, 244); /* Gris claro */
  }

  /* Ajustes adicionales */
  .table td {
    vertical-align: middle; /* Centrar verticalmente las celdas */
  }
}

/*  ESTILOS TIME UP */
```

```
.cronometro ul{
  list-style-type: cjk-earthly-stem;
}
.cronometro .time{
  width: 400px;
}
.cronometro .opciones button{
  margin: 0 10px;
}
```

ESTRUCTURA TYPESCRIPT

```
import { Component, ElementRef, Input, OnInit, Renderer2, ViewChild } from
'@angular/core';
import { FormGroup } from '@angular/forms';
import { ResultadoService } from 'src/app/Service/resultado.service';
import { FormBuilder } from '@angular/forms';
import { FormControl } from '@angular/forms';
import { ActivatedRoute, Router } from '@angular/router';
import { DiagnosticComponent } from '../../diagnostic/diagnostic.component';
import { ApiService } from 'src/app/Service/api.service';
import { Checkbox, Question } from 'src/app/Shared/Data';
import { float, FLOAT } from 'html2canvas/dist/types/css/property-descriptors/float';

@Component({
  selector: 'app-category1',
  templateUrl: './category1.component.html',
  styleUrls: ['./category1.component.scss'],
})
export class Category1Component implements OnInit {
  encuestaSelect: String = '';
  puntos: number = 0;
  observacion: string = '';
  showErrors: boolean = false;
  categoria?: String;
  mensajeError: string = '';
  ent: String [] = ['Entrada de ', 'Salida de ']

  // Propiedades para los valores de peso
  peso1: FLOAT = 0.0; // Year
  peso2: FLOAT = 0.0; // Actual
  pesoYear: string = '';
  pesoActual: string = '';
  // Mensajes de error
  pesoHaceUnAnioError: string = '';
  pesoActualError: string = '';
  //Validacion de Numero
  pesoIn: boolean = false; pesoIn2: boolean = false;

  /* Preguntas y Opciones de BARTHEL */
  preguntasBarthel: Question[] = [
    { text: 'Alimentación',
      options: [
```

```
        { label: 'Independiente: Capaz de utilizar cualquier instrumento, come en
tiempo, puede ser servida o cocinada por otra persona', score: 10 },
        { label: 'Ayuda: Necesita ayuda pero capaz de comer solo', score: 5 },
        { label: 'Dependiente: Depende de otra persona para comer', score: 0 }
      ]
    },
    { text: 'Bañarse/Ducharse',
      options: [
        { label: 'Independiente: Entra y sale solo del baño, sin supervisión', score:
5 },
        { label: 'Dependiente: Necesita ayudao o supervisión', score: 0 }
      ]
    },
    { text: 'Vestirse',
      options: [
        { label: 'Independiente: Capaz de ponerse y de quitarse la ropa, abotonarse,
atarse los zapatos' , score: 10 },
        { label: 'Ayuda: Necesita de personal, pero al menos puede realiza las tareas
con tiempo razonable' , score: 5},
        { label: 'Dependiente: Necesita ayuda para algunas actividades', score: 0 }
      ]
    },
    { text: 'Aseo Personal',
      options: [
        { label: 'Independiente: Realiza todas las actividades sin personal' , score:
5 },
        { label: 'Dependiente: Requiere de personal para asearse en algunas
actividades', score: 0 }
      ]
    },
    { text: 'Control de Orina',
      options: [
        { label: 'Continente: No presenta inconsistencia, capaz de atender solo su
cuidado', score: 10 },
        { label: 'Incontinencia Ocasional: Como máximo incontigencia en 24hrs,
requiere ayuda para el cuidado', score: 5 },
        { label: 'Incontinencia: Episodios de incontigencia con frecuencia una vez en
24hrs, incapaz de manejar su cuidado', score: 0 }
      ]
    },
    { text: 'Control de Heces',
      options: [
        { label: 'Continencia Normal: No presenta episodios, capaz de administrase
solo', score: 10 },
        { label: 'Ocasional: Ocasionalmente una vez por semana, necesita ayuda por
supositorios', score: 5 },
        { label: 'Incontinencia: Más de un episodio por semana', score: 0 }
      ]
    },
    { text: 'Uso del Retrete (TAZA)',
      options: [
        { label: 'Independiente: Usa el retrete por si mismo sin ayuda', score: 10 },
```

```
        { label: 'Ayuda: Requiere de personal oara mantener el equilibrio sentado,
limpiarse o ponerse de pie', score: 5 },
        { label: 'Dependiente: Requiere totalmente de ayuda personal para ir la
baño', score: 0 }
      ]
    },
    { text: 'Traslado Sillón-Cama',
      options: [
        { label: 'Independiente: Puede ir del sillón a la cama sin personal
supervisor', score: 15 },
        { label: 'Mínima ayuda: Requiere supervisión o pequeña ayuda para traslado',
score: 10 },
        { label: 'Gran Ayuda: Requiere de ayuda para traslado, capaz de permanecer
sentado sin ayuda', score: 5 },
        { label: 'Dependiente: Requiere de 2 personas o gura para traslado, incapaz
de sentrase solo', score: 0 }
      ]
    }, { text: 'Dezplazamiento',
      options: [
        { label: 'Independiente: Puede caminar 50 metros o equivalente sin
supervisor' , score: 15 },
        { label: 'Ayuda: Camina 50m, pero necesita personal (física o verbal) o
suvervisor', score: 10 },
        { label: 'Independiente en silla de ruedas: Propulsa su silla 50m sin ayuda o
supervisión', score: 5 },
        { label: 'Dependiente: No puede caminar solo o propulsar su silla', score: 0
}
      ]
    },
    { text: 'Escalones',
      options: [
        { label: 'Independiente: Puede bajar y subir escaleras sin supervición' ,
score: 10 },
        { label: 'Ayuda: Requiere ayuda física o supervisión para subir o bajar',
score: 5 },
        { label: 'Dependiente: Incapaz de subir y bajar escaleras, requiere asesor',
score: 0 }
      ],
    }
  ]
  respuestas: number[] = new Array(this.preguntasBarthel.length).fill(-1); // Array
para almacenar respuestas (-1 indica no respondida)

  /* Preguntas ENSRUD */
  preguntasEnsrud: Checkbox[] = [
    { text: '1.- Pérdida de peso de 5% o mayor en los últimos 30 días', seleccionada:
false },
    { text: '2.- Inhabilitado para levantarse de una silla 5 veces sin emplear los
brazos', seleccionada: false },
    { text: '3.- Pobre energía identificado con una respuesta negativa: "¿Siente
usted con energía?"', seleccionada: false },
  ];
```

```
  // Respuestas de las preguntas Ensrud
  answers: {[key: string]:number | null} = {
      resistance: null,
      aerobic: null,
      illnesses: null,
      fatigue: null
  };
  // Mensaje de error por campo
  fieldErrors: { [key: string]: string } = {};

  /* CRONOMETRO */
  tiempo: number = 0; // Tiempo en milisegundos
  intervalId: any;
  capturas: string[] = []; // Lista para guardar tiempos capturados
  corriendo: boolean = false;

  //Encuesta KATZ
  preguntasKATZ: Question[] = [
      { text: 'BAÑO (Esponja, regadera o tina)',
        options: [
          { label: 'No recibe asistencia (puede entrar y salir de la tina u otra
forma de baño).', score: 1 },
          { label: 'Que reciba asistencia durante el baño en una sola parte del
cuerpo', score: 1 },
          { label: 'Dependiente: Que reciba asistencia durante el baño en más de una
parte.', score: 0 }
        ]
      },
      { text: 'VESTIDO',
        options: [
          { label: 'Que pueda tomar las prendas y vestirse completamente, sin
asistencia.', score: 1 },
          { label: 'Que pueda tomar las prendas y vestirse sin asistencia excepto en
abrocharse los zapatos.', score: 1 },
          { label: 'Que reciba asistencia para tomar las prendas y vestirse.', score:
0 }
        ]
      },
      { text: 'USO DEL SANITARIO',
        options: [
          { label: 'Sin ninguna asistencia (puede utilizar objeto de soporte: bastón
o silla de ruedas y/o puede arreglar su ropa o el uso de pañal o cómodo).', score: 1
},
          { label: 'Que reciba asistencia al ir al baño, en limpiarse y que pueda
manejar por si mismo/a el pañal o cómodo vaciándolo.', score: 1 },
          { label: 'Que no vaya al baño por si mismo/a.', score: 0 }
        ]
      },
      { text: 'TRANSFERENCIAS',
        options: [
          { label: 'Que se mueva dentro y fuera de la cama y silla sin ninguna
asistencia (utiliza un auxiliar de la marcha u objeto de soporte).', score: 1 },
```

```
          { label: 'Que pueda moverse dentro y fuera de la cama y silla con
asistencia.', score: 1 },
          { label: 'Que no pueda salir de la cama.', score: 0 }
        ]
      },
      { text: 'CONTINENCIA',
        options: [
          { label: 'Control total de esfínteres.', score: 1 },
          { label: 'Que tenga accidentes ocasionales que no afectan su vida social.',
score: 1 },
          { label: 'Necesita ayuda para supervición del control de enfínteres,
utiliza sonda o es incontinente.', score: 0 }
        ]
      },
      { text: 'ALIMENTACIÓN',
        options: [
          { label: 'Que se alimente por si solo sin asistencia alguna.', score: 1 },
          { label: 'Que se alimente solo y que tenga asistencia sólo para cortar la
carne o untar mantequilla. ', score: 1 },
          { label: 'Que reciba asistencia en la alimentación o que se alimente
parcial o totalmente por vía enteral o parenteral.', score: 0 }
        ]
      },
  ];
  selectedOptions: number[] = Array(this.preguntasKATZ.length).fill(-1);
  errors: boolean[] = Array(this.preguntasKATZ.length).fill(false); // Array para
rastrear errores por pregunta

  //Encuesta Brody
  preguntasLowton: Question[] = [
    {
      text: 'CAPACIDAD PARA USAR TELÉFONO',
      options: [
        { label: 'Lo opera por iniciativa propia, lo marca sin problemas.', score: 1
},
        { label: 'Marca sólo unos cuantos números bien conocidos.', score: 1 },
        { label: 'Contesta el teléfono pero no llama.', score: 1 },
        { label: 'No usa el teléfono.', score: 0 }
      ]
    },
    {
      text: 'COMPRAS',
      options: [
        { label: 'Vigila sus necesidades independientemente.', score: 1 },
        { label: 'Hace independientemente sólo pequeñas compras.', score: 0 },
        { label: 'Necesita compañía para cualquier compra.', score: 0 },
        { label: 'Incapaz de cualquier compra.', score: 0 }
      ]
    },
    {
      text: 'COCINA',
      options: [
```

```
        { label: 'Planea, prepara y sirve los alimentos correctamente.', score: 1 },
        { label: 'Prepara los alimentos sólo si se le provee lo necesario.', score: 0
},
        { label: 'Calienta, sirve y prepara pero no lleva una dieta adecuada.',
score: 0 },
        { label: 'Necesita que le preparen los alimentos.', score: 0 }
      ]
    },
    {
      text: 'CUIDADO DEL HOGAR',
      options: [
        { label: 'Mantiene la casa solo o con ayuda mínima.', score: 1 },
        { label: 'Efectúa diariamente trabajo ligero eficientemente.', score: 1 },
        { label: 'Efectúa diariamente trabajo ligero sin eficiencia.', score: 1 },
        { label: 'Necesita ayuda en todas las actividades.', score: 0 },
        { label: 'No participa.', score: 0 }
      ]
    },
    {
      text: 'LAVANDERÍA',
      options: [
        { label: 'Se ocupa de su ropa independientemente.', score: 1 },
        { label: 'Lava sólo pequeñas cosas.', score: 1 },
        { label: 'Todos se lo tienen que lavar.', score: 0 }
      ]
    },
    {
      text: 'TRANSPORTE',
      options: [
        { label: 'Se transporta solo/a.', score: 1 },
        { label: 'Se transporta solo/a, únicamente en taxi pero no puede usar otros
recursos.', score: 1 },
        { label: 'Viaja en transporte colectivo acompañado.', score: 1 },
        { label: 'Viaja en taxi o auto acompañado.', score: 0 },
        { label: 'No sale.', score: 0 }
      ]
    },
    {
      text: 'MEDICACIÓN',
      options: [
        { label: 'Es capaz de tomarla a su hora y dosis correctas.', score: 1 },
        { label: 'Se hace responsable sólo si le preparan por adelantado.', score: 0
},
        { label: 'Es incapaz de hacerse cargo.', score: 0 }
      ]
    },
    {
      text: 'FINANZAS',
      options: [
        { label: 'Maneja sus asuntos independientemente.', score: 1 },
        { label: 'Sólo puede manejar lo necesario para pequeñas compras.', score: 1
},
```

```
        { label: 'Es incapaz de manejar dinero.', score: 0 }
      ]
    }
  ];
  optionL: number[] = Array(this.preguntasLowton.length).fill(-1);
  errorsLowton: boolean[] = Array(this.preguntasLowton.length).fill(false);

  /* ------------------------------ CONSTRUC DE LA CLASE --------------------------
*/
  constructor(private fb: FormBuilder, private encuestaService: ApiService, private
router: Router, private route: ActivatedRoute, private render: Renderer2) {

  }
   /* ------------------------------ ONINIT = INICIALIZADOR ----------------------
--- */
  ngOnInit(): void {
    this.categoria = this.route.snapshot.paramMap.get('categoria')!;
    this.encuestaService.selectEncuest$.subscribe(cat => {
      this.encuestaSelect = cat;
    });
  }

  // En el componente 'Category1Component'
  finalizarEncuesta() {
    this.encuesta(this.encuestaSelect);
  }

  cambiarCategoria(categoria: String){
    this.encuestaSelect = categoria.toString();
  }

  encuesta(enc: String){
    switch(enc){
      case 'katz':
        console.log(this.ent[0], this.encuestaSelect);
        //VERIFICAR SI LAS OPCIONES ESTEN SELECCIONADAS
        this.errors = this.selectedOptions.map(score => score == -1);

        let katzT = 0;
        for (let i = 0; i < this.selectedOptions.length; i++) {
          katzT += this.selectedOptions[i];
        }
        if(!this.errors.includes(true)){
          const letter = this.getIndependenceCategory();
          this.puntos = katzT;//Pasa los puntos obtenido al puntos
          this.showErrors = true;
          this.encuestaResultado(this.puntos, 'Indice de KATZ', 'Letra Asignada: ' +
letter,(this.puntos/6)*100)
          console.log(this.ent[1] + 'KATZ');
        }
        break;
      case 'barthel':
```

```
        console.log(this.ent[0], this.encuestaSelect);
        // Verifica si todas las preguntas tienen respuesta seleccionada
        for (let i = 0; i < this.respuestas.length; i++) {
          if (this.respuestas[i] == -1) {
            this.mensajeError = 'Debe responder todas las preguntas antes de calcular
el puntaje.';
            this.puntos = 0;
            return;
          } else {
            this.showErrors = true
            this.mensajeError = '';} //En caso de que se respondas todos, desbloquea
el metodo encuestaResultado
        }
        // Calcula el puntaje total sumando los valores seleccionados
        let total = 0;
        for (let i = 0; i < this.respuestas.length; i++) {
          total += this.respuestas[i];
        }
        this.puntos = total;//Pasa los puntos obtenido al puntos

        if(this.puntos <= 20){
          this.observacion = 'Dependencia Total';
        } else if(this.puntos >= 21 && this.puntos <= 60){
          this.observacion = 'Dependencia grave';
        } else if(this.puntos >= 61 && this.puntos <= 90){
          this.observacion = 'Dependencia Moderada';
        } else if( this.puntos >= 91 && this.puntos <= 99){
          this.observacion = 'Dependencia Escasa';
        } else {
          this.observacion = 'Independiente';
        }

        this.encuestaResultado(this.puntos, 'Indice de Barthel', this.observacion,
(this.puntos/100)*100);
        console.log(this.ent[1] + 'Barthel');
        break;
      case 'lawton':
        console.log(this.ent[0], this.encuestaSelect);
        this.errorsLowton = this.optionL.map(score => score == -1);

        let lowtonT = 0;
        for (let i = 0; i < this.optionL.length; i++) {
          lowtonT += this.optionL[i];
        }
        this.puntos = lowtonT;

        if(this.puntos == 8){
          this.observacion = 'Muy activos: actividades instrumentales completas';
        } else if(this.puntos >= 5 && this.puntos <= 7){
          this.observacion = 'Activos: actividades limitadas';
        } else if(this.puntos <= 0){
          this.observacion = 'Inactivos: no realizan actividades instrumentales'
```

```
        } else {
          this.observacion = 'Poco activos: limitación del 50 % o más de esas
actividades';
        }

        if(!this.errorsLowton.includes(true)){
          this.showErrors = true;
          this.encuestaResultado(this.puntos,'Escala Lowton y Brody',
this.observacion, (this.puntos/8)*100);
        }

        console.log(this.ent[1] + 'Brody y Lowton');
        break;
      case 'frail':
          console.log('Entrada de: FRAIL');

          let allAnswered = true;
          // Validar cada pregunta
          for (const question in this.answers) {
            if (this.answers[question] === null) {
              this.fieldErrors[question] = 'Por favor, seleccione una opción.';
              allAnswered = false;
            } else {
              this.fieldErrors[question] = ''; // Limpia errores si hay respuesta
            }
          }

          if(allAnswered){
            this.mensajeError = '';
            //Calculo del Peso
            if(this.pesoIn && this.pesoIn2){
              this.peso1 = parseFloat(this.pesoYear); //Year Anterior
              this.peso2 = parseFloat(this.pesoActual);//Actual
            } else { this.peso1 = 0.0; this.peso2 = 0.0}
            const value = (((this.peso1 - this.peso2)/this.peso1)*100).toFixed(2);
//Convertir para validar en if
            if(parseFloat(value) >= 5){
              // Cuando el peso cumple la condicion pasaa ser 1
              this.puntos =+ 1;
            }
            //Calcular Puntaje
            for (const question in this.answers) {
              if (this.answers[question] == 1) {
                this.puntos += 1; // Suma 1 por cada respuesta positiva
              }
            }
            this.showErrors = true;


          } else {
            this.mensajeError = 'Por favor, responda todas las preguntas.';
          }
```

```
          if(this.puntos >= 5){
            this.observacion = 'Probable Fragilidad';
          } else if(this.puntos <= 0){
            this.observacion = 'Sin fragilidad o robuztez';
          } else {
            this.observacion = 'Probable pre-fatiga'
          }

          this.encuestaResultado(this.puntos, 'FRAIL', this.observacion,
(this.puntos/5)*100);

          console.log("Salida de Frail | ", this.puntos);
          break;
      case 'ensrud':
        console.log('Entrada de: Ensrud');
        this.puntos = this.preguntasEnsrud.filter(p => p.seleccionada).length;
        this.showErrors = true;
        if(this.puntos >= 2){
          this.observacion = 'Estado Fragil';
        } else if(this.puntos == 1){
          this.observacion = 'Estado Prefragil'
        } else {
          this.observacion = 'Estado Ningun criterio (Robusto)';
        }
        console.log('Salida de: Ensrud');
        this.encuestaResultado(this.puntos, 'Criterio Ensrud', this.observacion,
(this.puntos*3)/100);
        break;
    }
  }

  //Metodo para pasar al siguiente componente, en case de que este respondidos las
respuestas
  encuestaResultado(puntos: number, nameEncuesta: String, observacion: String,
porcentaje: FLOAT){
    if(this.showErrors){
      // Al navegar, enviamos los puntos al componente de resultado
      this.router.navigate(['home/resultado'], {queryParams: {
        puntaje: puntos, //Puntaje Obtenido
        nameEncuesta: nameEncuesta, //Nombre de la Encuesta
        porcentaje: porcentaje.toFixed(2),
        observacion: observacion //Observaciones
      }});
      localStorage.removeItem('subCatSeleccionada');
   }
  }
  //Seleccionar respuesta de Opcion Multiple (BARTHEL)
  seleccionarRespuesta(index: number, score: number): void {
    this.respuestas[index] = score;
  }
  //Si esta seleccionado el checkbox se suma 1 punto
```

```
toggleCheckbox(index: number, event: any): void {
  this.preguntasEnsrud[index].seleccionada = event.target.checked;
}
//Valida la entrada del usuario para que solo acepte números y un punto decimal.
validateInput(event: Event, field: string): void {
  const input = event.target as HTMLInputElement;
  let value = input.value;
  // Mensaje de error por defecto
  let errorMessage = '';

  // Permitir solo números y un punto decimal
  if (!/^\d*\.?\d*$/.test(value)) {
    errorMessage = 'Solo se permiten números y un punto decimal.';
    value = value.replace(/[^0-9.]/g, ''); // Eliminar caracteres no válidos
  }

  // Permitir solo un punto decimal
  const parts = value.split('.');
  if (parts.length > 2) {
    value = parts[0] + '.' + parts[1]; // Mantener solo el primer punto decimal
  }

  // Limitar a 4 caracteres como máximo
  if (value.length > 4) {
    value = value.slice(0, 4);
  }

  // Actualizar el valor en el modelo y muestra mensage de error
  if (field == 'pesoYear') {
    this.pesoYear = value;
    this.pesoHaceUnAnioError = errorMessage;
    this.pesoIn = true;
  } else if (field == 'pesoActual') {
    this.pesoActual = value;
    this.pesoActualError = errorMessage;
    this.pesoIn2 = true;
  }
}
onAnswerChange(question: string, value: number): void {
  this.answers[question] = value;
  this.fieldErrors[question] = ''; // Limpia errores si selecciona respuesta
  this.mensajeError ='';
}
//FUNCION CRONOMETRO
iniciarPausar(): void {
  if (this.corriendo) {
    clearInterval(this.intervalId);
  } else {
    this.intervalId = setInterval(() => {
      this.tiempo += 10; // Incrementa cada 10ms
    }, 10);
  }
```

```
    this.corriendo = !this.corriendo;
  }

  reiniciar(){
    clearInterval(this.intervalId);
    this.tiempo = 0;
    //this.capturas = [];
    this.corriendo = false;
  }
  capturarTiempo(){
    const minutos = Math.floor(this.tiempo / 60000);
    const segundos = Math.floor((this.tiempo % 60000) / 1000);
    const milisegundos = Math.floor((this.tiempo % 1000) / 10);

    this.capturas.push(`${this.pad(minutos)}:${this.pad(segundos)}.${this.pad(miliseg
undos)}`);
  }
  private pad(num: number): string {
    return num < 10 ? `0${num}` : `${num}`;
  }
  deleteScore(){
    this.capturas = [];
    this.reiniciar();
  }

  //Funcion KATZ
  getIndependenceCategory(): string {
    // Define los niveles de independencia/dependencia para cada pregunta
    const independence = this.selectedOptions.map(score => score == 1);
    const dependenceCount = independence.filter(dep => !dep).length;

    if (dependenceCount == 0) {
      return 'A'; // Independencia total
    }

    if (dependenceCount == 1) {
      return 'B'; // Independencia en todas menos una
    }

    const dependentActivities = {
      baño: !independence[0],
      vestido: !independence[1],
      sanitario: !independence[2],
      transferencias: !independence[3],
      continencia: !independence[4],
      alimentación: !independence[5]
    };

    if (dependentActivities.baño && dependenceCount == 2) {
      return 'C';
    }
```

```
    if (dependentActivities.baño && dependentActivities.vestido && dependenceCount ==
3) {
      return 'D';
    }

    if (dependentActivities.baño && dependentActivities.vestido &&
dependentActivities.sanitario && dependenceCount == 4) {
      return 'E';
    }

    if (dependentActivities.baño && dependentActivities.vestido &&
dependentActivities.sanitario && dependentActivities.transferencias &&
dependenceCount == 5) {
      return 'F';
    }

    if (dependenceCount == 6) {
      return 'G'; // Dependencia total
    }

    if (dependenceCount == 2) {
      return 'H'; // Dependencia en dos actividades no clasificadas en C, D, E, F
    }

    return 'Error'; // Caso no contemplado
  }

}
```

Encuesta de Categoría Afectiva

```
<!-- GDS - 15 -->
<div class="container mt-4">
  <h1 class="text-center mb-4">GDS - 15</h1>
  <form [formGroup]="gdsForm">
    <div class="table-responsive">
      <table class="table table-bordered table-striped">
        <thead class="table-primary">
          <tr>
            <th>#</th>
            <th>Pregunta</th>
            <th>Sí (1)</th>
            <th>No (0)</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let pregunta of preguntas; let i = index">
            <td>{{ i + 1 }}</td>
            <td>{{ pregunta }}</td>
            <td>
              <input
                type="radio"
                [value]="1"
```

```
                [formControlName]="'pregunta' + i"
              />
            </td>
            <td>
              <input
                type="radio"
                [value]="0"
                [formControlName]="'pregunta' + i"
              />
            </td>
          </tr>
        </tbody>
      </table>
    </div>
    <div class="text-center">
      <button type="button" class="btn btn-primary"
(click)="calcularPuntuacion()">Calcular y Ver Resultados</button>
    </div>
  </form>
</div>

/* Estilos generales para la tabla */
.table {
    font-size: 0.9rem; /* Ajusta el tamaño de fuente para mejor legibilidad */
  }

  .table th,
  .table td {
    text-align: center;
    vertical-align: middle;
    word-wrap: break-word;
  }

  .table th {
    background-color: #007bff;
    color: white;
  }

  .table td {
    padding: 10px;
  }

  /* Responsividad para dispositivos pequeños */
  @media (max-width: 768px) {
    .table-responsive {
      overflow-x: auto;
    }

    .table {
      font-size: 0.8rem;
    }
```

```
    h1 {
      font-size: 1.5rem;
    }
  }

  /* Estilos para botones */
  .btn-primary {
    background-color: #007bff;
    border-color: #0056b3;
  }

  .btn-primary:hover {
    background-color: #0056b3;
    border-color: #003d80;
  }
```

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-encuesta-afc',
  templateUrl: './encuesta-afc.component.html',
  styleUrls: ['./encuesta-afc.component.scss']
})
export class EncuestaAfcComponent implements OnInit {
  gdsForm!: FormGroup;
  preguntas: string[] = [
    '¿En general, está satisfecho(a) con su vida?',
    '¿Ha abandonado muchas de sus tareas habituales y aficiones?',
    '¿Siente que su vida está vacía?',
    '¿Se siente con frecuencia aburrido(a)?',
    '¿Se encuentra de buen humor la mayor parte del tiempo?',
    '¿Teme que algo malo pueda ocurrirle?',
    '¿Se siente feliz la mayor parte del tiempo?',
    '¿Con frecuencia se siente desamparado(a), desprotegido(a)?',
    '¿Prefiere usted quedarse en casa, más que salir y hacer cosas nuevas?',
    '¿Cree que tiene más problemas de memoria que la mayoría de la gente?',
    '¿En estos momentos, piensa que es estupendo estar vivo(a)?',
    '¿Actualmente se siente un(a) inútil?',
    '¿Se siente lleno(a) de energía?',
    '¿Se siente sin esperanza en este momento?',
    '¿Piensa que la mayoría de la gente está en mejor situación que usted?'
  ];
  puntaje: number | null = null;
  observacion: string = '';
  showErrors = false;

  constructor(private fb: FormBuilder, private router: Router, private shared:
SharedService) {}
```

```
ngOnInit(): void {
  // Inicializa el formulario
  this.gdsForm = this.fb.group(
    this.preguntas.reduce((controles, _, index) => {
      controles['pregunta' + index] = [null, Validators.required];
      return controles;
    }, {} as { [key: string]: any })
  );
}

calcularPuntuacion(): void {
  if (this.gdsForm.invalid) {
    alert('Por favor, responde todas las preguntas.');
    return;
  }

  // Obtiene las respuestas con el tipo explícito
  const respuestas: { [key: string]: number } = this.gdsForm.value;

  // Suma las puntuaciones
  this.puntaje = Object.values(respuestas).reduce(
    (acc: number, val: unknown) => acc + (typeof val === 'number' ? val : 0),
    0
  );

  // Interpreta el resultado
  this.observacion =
    this.puntaje <= 4
      ? 'Normal'
      : 'Presencia de síntomas depresivos';

  this.enviarResultados(this.puntaje, this.observacion);
}

enviarResultados(puntaje: number, observacion: string): void {
  // Captura los datos en tipo any
  const resultado = {
    point: puntaje,
    encuesta: 'Escala AFC',
    observable: observacion,
    porcent: ((puntaje / 100) * 15).toFixed(2)
  }
  if (puntaje === null) {
    alert('Primero calcula el puntaje antes de enviar los resultados.');
    return;
  } else {
    this.showErrors = true;
  }

  console.log('Puntaje: ', puntaje, '\nObservación: ', observacion);

  // Redirige al componente Resultados con los datos mediante el estado
```

```
    if (this.showErrors) {
      this.shared.saveResults(resultado); //Envia los datos capturados al servicio
      this.router.navigate(['home/resultado']);
    }
  }
}


<!-- Encuesta Becky -->
<div class="container">
  <h2>Inventario de Ansiedad de Beck</h2>
  <p>
    Lea cada enunciado e indique cuál ha sido el nivel de afectación en la última
semana, incluyendo el día de hoy.
  </p>
  <form #beckForm="ngForm" (ngSubmit)="finalizarEncuesta()">
    <table class="table table-bordered">
      <thead>
        <tr>
          <th>#</th>
          <th>Enunciado</th>
          <th>En absoluto (0)</th>
          <th>Levemente (1)</th>
          <th>Moderadamente (2)</th>
          <th>Severamente (3)</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let pregunta of preguntas; let i = index">
          <td>{{ i + 1 }}</td>
          <td>{{ pregunta.texto }}</td>
          <td>
            <input
              type="radio"
              [id]="'pregunta-' + i + '-0'"
              name="pregunta-{{ i }}"
              value="0"
              [(ngModel)]="pregunta.respuesta"
              required
            />
          </td>
          <td>
            <input
              type="radio"
              [id]="'pregunta-' + i + '-1'"
              name="pregunta-{{ i }}"
              value="1"
              [(ngModel)]="pregunta.respuesta"
            />
          </td>
          <td>
            <input
```

```
              type="radio"
              [id]="'pregunta-' + i + '-2'"
              name="pregunta-{{ i }}"
              value="2"
              [(ngModel)]="pregunta.respuesta"
            />
          </td>
          <td>
            <input
              type="radio"
              [id]="'pregunta-' + i + '-3'"
              name="pregunta-{{ i }}"
              value="3"
              [(ngModel)]="pregunta.respuesta"
            />
          </td>
        </tr>
      </tbody>
    </table>
    <!-- Botón para finalizar encuesta -->
    <button
      type="button"
      class="btn btn-success"
      (click)="finalizarEncuesta()"
      [disabled]="!beckForm.valid"
    >
      Finalizar Encuesta
    </button>
  </form>
</div>

/* Estilos */
.container {
    max-width: 900px;
    margin: 0 auto;
    padding: 20px;
    background-color: #ffffff;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  }

  h2 {
    text-align: center;
    color: #333333;
    margin-bottom: 20px;
    font-size: 1.8rem;
  }

  p {
    font-size: 1rem;
    margin-bottom: 20px;
  }
```

```
table {
  width: 100%;
  border-collapse: collapse;
}

th, td {
  text-align: center;
  padding: 10px;
  font-size: 0.95rem;
}

thead th {
  background-color: #007bff;
  color: #ffffff;
}

tr:nth-child(even) {
  background-color: #f9f9f9;
}

tr:hover {
  background-color: #f1f1f1;
}

button[type="submit"] {
  width: 100%;
  padding: 10px;
  font-size: 1rem;
  font-weight: bold;
  color: #ffffff;
  background-color: #007bff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease-in-out;
}

button[type="submit"]:hover {
  background-color: #0056b3;
}

button[type="submit"]:disabled {
  background-color: #cccccc;
  cursor: not-allowed;
}

@media (max-width: 768px) {
  h2 {
    font-size: 1.5rem;
  }
```

```
    th, td {
      font-size: 0.9rem;
    }

    button[type="submit"] {
      font-size: 0.9rem;
    }
  }

  @media (max-width: 576px) {
    h2 {
      font-size: 1.3rem;
    }

    th, td {
      font-size: 0.85rem;
    }

    button[type="submit"] {
      font-size: 0.85rem;
      padding: 8px;
    }
  }
```

```
/* Codigo */
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-encuesta-beck-anxiety',
  templateUrl: './encuesta-beck-anxiety.component.html',
  styleUrls: ['./encuesta-beck-anxiety.component.scss']
})
export class EncuestaBeckAnxietyComponent {
  // Preguntas del inventario
  preguntas = [
    { texto: 'Torpe o entumecido.', respuesta: null },
    { texto: 'Acalorado.', respuesta: null },
    { texto: 'Con temblor en las piernas.', respuesta: null },
    { texto: 'Incapaz de relajarse.', respuesta: null },
    { texto: 'Con temor a que ocurra lo peor.', respuesta: null },
    { texto: 'Mareado, o que se le va la cabeza.', respuesta: null },
    { texto: 'Con latidos fuertes del corazón y acelerados.', respuesta: null },
    { texto: 'Inestable.', respuesta: null },
    { texto: 'Atemorizado o asustado.', respuesta: null },
    { texto: 'Nervioso.', respuesta: null },
    { texto: 'Con sensación de bloqueo.', respuesta: null },
    { texto: 'Con temblores en las manos.', respuesta: null },
    { texto: 'Inquieto, inseguro.', respuesta: null },
    { texto: 'Con miedo a perder el control.', respuesta: null },
    { texto: 'Con sensación de ahogo.', respuesta: null },
```

```
    { texto: 'Con temor a morir.', respuesta: null },
    { texto: 'Con miedo.', respuesta: null },
    { texto: 'Con problemas digestivos.', respuesta: null },
    { texto: 'Con desvanecimientos.', respuesta: null },
    { texto: 'Con rubor facial.', respuesta: null },
    { texto: 'Con sudores, fríos o calientes.', respuesta: null }
  ];

  puntaje: number = 0;
  observacion: string = '';
  showErrors: boolean = false;

  constructor(private router: Router, private share: SharedService) {}

  // Método para calcular el puntaje
  calcularPuntaje(): void {
    this.puntaje = this.preguntas.reduce((total, pregunta) => total +
(Number(pregunta.respuesta) || 0), 0);

    if (this.puntaje <= 5) {
      this.observacion = 'Ausente o mínima ansiedad.';
    } else if (this.puntaje <= 15) {
      this.observacion = 'Ansiedad leve.';
    } else if (this.puntaje <= 30) {
      this.observacion = 'Ansiedad moderada.';
    } else {
      this.observacion = 'Ansiedad grave.';
    }

    console.log('Puntaje calculado: ', this.puntaje, 'Observación: ',
this.observacion);
  }

  // Método para enviar resultados al componente de resultados
  enviarResultados(puntaje: number, observacion: string): void {
   if (puntaje === null) {
      alert('Primero calcula el puntaje antes de enviar los resultados.');
      return;
    } else {
      this.showErrors = true;
    }

    const resultados = {
      point: this.puntaje,
      encuesta: 'Inventario de Ansiedad de Beck',
      observable: this.observacion,
      porcent: ((this.puntaje / 21) * 100).toFixed(2)
    }

    console.log('Puntaje: ', puntaje, '\nObservación: ', observacion);

    // Redirige al componente Resultados con los datos mediante el estado
```

```
    if (this.showErrors) {
      this.share.saveResults(resultados);
      this.router.navigate(['home/resultado']);
    }
  }

  // Método que combina el cálculo y el envío de los resultados
  finalizarEncuesta(): void {
    // Calcular el puntaje
    this.calcularPuntaje();

    // Enviar los resultados al componente de resultados
    this.enviarResultados(this.puntaje, this.observacion);
  }
}


<!-- Encuesta D7 -->
<div class="container mt-4">
  <h2 class="text-center">CES – D 7 Escala de Depresión</h2>
  <p class="text-center">Durante la última semana, ¿usted...?</p>

  <form [formGroup]="cesForm">
    <div class="table-responsive">
      <table class="table table-bordered text-center">
        <thead class="table-light">
          <tr>
            <th>Preguntas</th>
            <th>Rara vez o nunca<br>(menos de 1 día)<br>0 puntos</th>
            <th>Pocas veces<br>(1-2 días)<br>1 punto</th>
            <th>Un número considerable<br>(3-4 días)<br>2 puntos</th>
            <th>Todo el tiempo<br>(5-7 días)<br>3 puntos</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let pregunta of preguntas; let i = index">
            <td>{{ pregunta.texto }}</td>
            <td>
              <input
                type="radio"
                [value]="0"
                [formControlName]="'pregunta' + i"
              />
            </td>
            <td>
              <input
                type="radio"
                [value]="1"
                [formControlName]="'pregunta' + i"
              />
            </td>
            <td>
```

```
              <input
                type="radio"
                [value]="2"
                [formControlName]="'pregunta' + i"
              />
            </td>
            <td>
              <input
                type="radio"
                [value]="3"
                [formControlName]="'pregunta' + i"
              />
            </td>
          </tr>
        </tbody>
      </table>
    </div>

    <div class="text-center mt-3">
      <button type="button" class="btn btn-success" (click)="finalizarEncuesta()">
        Finalizar Encuesta
      </button>
    </div>
  </form>

/* Estilos */
.table-responsive {
    overflow-x: auto;
  }

  .table {
    font-size: 0.9rem;

    @media (max-width: 768px) {
      font-size: 0.8rem;
    }
  }

  .btn {
    font-size: 1rem;

    @media (max-width: 768px) {
      font-size: 0.9rem;
    }
  }

  h2 {
    font-size: 1.5rem;

    @media (max-width: 768px) {
      font-size: 1.2rem;
    }
```

```
  }

// Código
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup } from '@angular/forms';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-encuesta-ces-d7',
  templateUrl: './encuesta-ces-d7.component.html',
  styleUrls: ['./encuesta-ces-d7.component.scss']
})
export class EncuestaCESD7Component implements OnInit {
  cesForm!: FormGroup;
  preguntas = [
    { texto: '¿Sentía como si no pudiera quitarse la tristeza?' },
    { texto: '¿Le costaba concentrarse en lo que estaba haciendo?' },
    { texto: '¿Se sintió deprimido/a?' },
    { texto: '¿Le parecía que todo lo que hacía era un esfuerzo?' },
    { texto: '¿No durmió bien?' },
    { texto: '¿Disfrutó de la vida?' },
    { texto: '¿Se sintió triste?' }
  ];
  resultado: number = 0;
  interpretacion: string = '';
  showErrors: boolean = false;

  constructor(private fb: FormBuilder, private router: Router, private share:
SharedService) {}

  ngOnInit(): void {
    // Crea un grupo de controles para cada pregunta
    this.cesForm = this.fb.group(
      this.preguntas.reduce((controls, _, index) => {
        controls['pregunta' + index] = [null];
        return controls;
      }, {} as { [key: string]: any })
    );
  }

  // Método para calcular la puntuación
  calcularPuntuacion(): void {
    const respuestas = this.cesForm.value;

    // Calcula la puntuación total
    this.resultado = (Object.values(respuestas) as (number | null)[]).reduce(
      (acc: number, val: number | null) => acc + (val || 0),
      0 // Inicializa el acumulador con 0
    );

    // Determina la interpretación
```

```
    this.interpretacion =
      this.resultado >= 5
        ? 'Síntomas depresivos significativos'
        : 'Normal';
  }

  // Método para enviar los resultados
  enviarResultados(): void {
    const resultados = {
      point: this.resultado,
      encuesta: 'CES-D7',
      observable: this.interpretacion,
      porcent: ((this.resultado / 21) * 100).toFixed(2)
    }
    if (this.resultado === 0 && !this.showErrors) {
      alert('Primero calcula el puntaje antes de enviar los resultados.');
      this.showErrors = true;
      return;
    }

    console.log('Puntaje: ', this.resultado, '\nObservación: ', this.interpretacion);

    // Redirige al componente Resultados con los datos mediante el estado
    this.share.saveResults(resultados);
    this.router.navigate(['home/resultado']);
  }

  // Método para finalizar la encuesta: calcula el puntaje y luego envía los
resultados
  finalizarEncuesta(): void {
    if (this.cesForm.valid) {
      this.calcularPuntuacion();  // Calcula la puntuación
      this.enviarResultados();    // Envía los resultados
    } else {
      alert('Por favor, complete todas las preguntas antes de finalizar.');
    }
  }
}

<!-- Encuesta Cornell -->
<div class="cornell-scale">
  <h2>Escala de Cornell</h2>
  <p>Manifestaciones.</p>

  <div *ngFor="let question of questions; let i = index" class="question">
    <label>{{ i + 1 }}. {{ question.text }}</label>
    <div class="options">
      <label>
        <input type="radio" name="q{{ i }}" /> No Valorable
      </label>
      <label>
        <input type="radio" name="q{{ i }}" (change)="setScore(i, 0)" /> Ausente
```

```
      </label>
      <label>
        <input type="radio" name="q{{ i }}" (change)="setScore(i, 1)" /> Leve
      </label>
      <label>
        <input type="radio" name="q{{ i }}" (change)="setScore(i, 2)" /> Grave
      </label>
    </div>
  </div>

  <div class="buttons">
    <button class="btn btn-primary" (click)="calculateScore()">Calcular
Puntaje</button>
    <button
      class="btn btn-success"
      [disabled]="resultado === 0"
      (click)="enviarResultados()">
      Enviar Resultados
    </button>
  </div>

  .cornell-scale {
  max-width: 800px;
  margin: 0 auto;
  font-family: Arial, sans-serif;
  padding: 20px;
  background-color: #f9f9f9;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.cornell-scale h2 {
  text-align: center;
  color: #333;
  margin-bottom: 20px;
}

.cornell-scale p {
  text-align: center;
  font-size: 1.1em;
  color: #555;
}

.question {
  padding: 15px 10px;
  margin-bottom: 15px;
  border-bottom: 1px solid #ddd;
}

.question:last-child {
  border-bottom: none;
}
```

```
.options {
  display: flex;
  justify-content: space-between;
  margin-top: 10px;
}

.options label {
  font-size: 1em;
  color: #555;
  cursor: pointer;
}

.options input {
  margin-right: 8px;
}

.buttons {
  display: flex;
  justify-content: center;
  gap: 15px;
  margin-top: 20px;
}

.buttons .btn {
  padding: 10px 20px;
  font-size: 1em;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

.buttons .btn-primary {
  background-color: #007bff;
  color: #fff;
}

.buttons .btn-primary:hover {
  background-color: #0056b3;
}

.buttons .btn-success {
  background-color: #28a745;
  color: #fff;
}

.buttons .btn-success:hover {
  background-color: #218838;
}

.result {
  text-align: center;
```

```
  margin-top: 20px;
  background-color: #e9ecef;
  padding: 15px;
  border-radius: 8px;
}

.result h4 {
  color: #333;
}

.result p {
  font-size: 1.1em;
  color: #555;
}

import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-encuesta-cornell',
  templateUrl: './encuesta-cornell.component.html',
  styleUrls: ['./encuesta-cornell.component.scss']
})
export class EncuestaCornellComponent {
  questions = [
    { text: 'Ansiedad (Expresion ansiosa, rumiacion de ideas, preocupacion)', score:
0 },
    { text: 'Tristesa (Expresion triste, voz apagada, tendencia al llanto)', score: 0
},
    { text: 'Falta de reactividad a los acontesimientos placenteros', score: 0 },
    { text: 'irritabilidad (facilmente enojable, poco temperamento)', score: 0 },
    { text: 'Agitacion (incapacidad de permanecer quieto, jugar con las manos,
tirarse el pelo)', score: 0 },
    { text: 'Enlentecimiento (Movimientos, habla y reacciones enlentecidos)', score:
0 },
    { text: 'Quejas Fisicas multiples (puntue 0 si son unicamente
gastrointestinales)', score: 0 },
    { text: 'perdida del interes (menos envuelto en las actividades habituales;
puntue solamente si el cambio ha sido agudo, p. ej., en menos de 1 mes)', score: 0 },
    { text: 'Perdida de apetito (come menos de lo habitual)', score: 0 },
    { text: 'Perdida de peso (puntue 2 si 2.5kg en 1 mes )', score: 0 },
    { text: 'Perdida de energia (Se fatiga facilmente, incapaz de mantener
actividades; puntue solamente si el cambio ha sido agudo, p. ej., en menos de 1 mes
)', score: 0 },
    { text: 'variaciones diurnas del estado de animo (los sintomas empeoran por la
mañana)', score: 0 },
    { text: 'Dificultad para conciliar el sueño (mas de lo habitual para el paciente
) ', score: 0 },
    { text: 'Despertares multiples durante el sueño', score: 0 },
    { text: 'Despertar precoz (antes de lo habitual para el paciente)', score: 0 },
```

```
    { text: 'Suicidio (siente que la vida no merece ser vivida, tiene deseos suicidas
o realiza intentos de suicidio) ', score: 0 },
    { text: 'Baja autoestima (autoculpa, autodepreciacion, sentimiento de fracaso)',
score: 0 },
    { text: 'Pesimismo (anticipacion a lo peor)', score: 0 },
    { text: 'Delirios congruentes con el estado de ánimo (delirios de pobresa,
enfermedades o pérdidas)', score: 0 },
  ];

  resultado: number = 0;
  observacion: string = '';
  showErrors: boolean = false;

  constructor(private router: Router, private share: SharedService) {}

  setScore(index: number, score: number): void {
    this.questions[index].score = score;
  }

  calculateScore(): void {
    this.resultado = this.questions.reduce((sum, question) => sum + question.score,
0);

    if (this.resultado <= 12) {
      this.observacion = 'depresion (Menor o mayor) ';
    } else if (this.resultado <= 38 ) {
      this.observacion = 'Depresión mayor';
    }

    this.showErrors = true; // Activa la validación para enviar resultados.
  }

  enviarResultados(): void {
    const resultados = {
      point: this.resultado,
      encuesta: 'Escala de Cornell',
      observable: this.observacion,
      porcent: ((this.resultado / 21) * 100).toFixed(2)
    }

    if (!this.showErrors || this.resultado === null) {
      alert('Primero calcula el puntaje antes de enviar los resultados.');
      return;
    }

    console.log('Puntaje:', this.resultado, '\nObservación:', this.observacion);

    this.share.saveResults(resultados);
    this.router.navigate(['home/resultado']);
  }
}
```

```
<!-- Encuesta Escala de Soledad -->
<div class="container">
    <h2>Escala de Soledad de 3 Elementos</h2>
    <p>
      indicaciones: se le dira a la persona lo siguiente "estas preguntas son sobre
como se siente sobre los diferentes aspectos de su vida. para cada pregunta responda
con que frecuencia se siente asi".
    </p>
    <form #soledadForm="ngForm" (ngSubmit)="calcularPuntaje()">
      <table class="table table-bordered">
        <thead>
          <tr>
            <th>#</th>
            <th>Pregunta</th>
            <th>Casi nunca (1)</th>
            <th>A veces (2)</th>
            <th>A Menudo (3)</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let pregunta of preguntas; let i = index">
            <td>{{ i + 1 }}</td>
            <td>{{ pregunta.texto }}</td>
            <td>
              <input
                type="radio"
                [id]="'pregunta-' + i + '-1'"
                name="pregunta-{{ i }}"
                value="1"
                [(ngModel)]="pregunta.respuesta"
                required
              />
            </td>
            <td>
              <input
                type="radio"
                [id]="'pregunta-' + i + '-2'"
                name="pregunta-{{ i }}"
                value="2"
                [(ngModel)]="pregunta.respuesta"
              />
            </td>
            <td>
              <input
                type="radio"
                [id]="'pregunta-' + i + '-3'"
                name="pregunta-{{ i }}"
                value="3"
                [(ngModel)]="pregunta.respuesta"
              />
            </td>
          </tr>
```

```
        </tbody>
      </table>
      <button type="submit" class="btn btn-primary" [disabled]="!soledadForm.valid">
        Finalizar Encuesta
      </button>
    </form>
  </div>

.container {
    max-width: 700px;
    margin: 0 auto;
    padding: 20px;
    background-color: #ffffff;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  }

  h2 {
    text-align: center;
    color: #333333;
    margin-bottom: 20px;
    font-size: 1.8rem;
  }

  p {
    font-size: 1rem;
    margin-bottom: 20px;
  }

  table {
    width: 100%;
    border-collapse: collapse;
  }

  th, td {
    text-align: center;
    padding: 10px;
    font-size: 0.95rem;
  }

  thead th {
    background-color: #007bff;
    color: #ffffff;
  }

  tr:nth-child(even) {
    background-color: #f9f9f9;
  }

  tr:hover {
    background-color: #f1f1f1;
  }
```

```
button[type="submit"] {
  width: 100%;
  padding: 10px;
  font-size: 1rem;
  font-weight: bold;
  color: #ffffff;
  background-color: #007bff;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease-in-out;
}

button[type="submit"]:hover {
  background-color: #0056b3;
}

button[type="submit"]:disabled {
  background-color: #cccccc;
  cursor: not-allowed;
}

@media (max-width: 768px) {
  h2 {
    font-size: 1.5rem;
  }

  th, td {
    font-size: 0.9rem;
  }

  button[type="submit"] {
    font-size: 0.9rem;
  }
}

@media (max-width: 576px) {
  h2 {
    font-size: 1.3rem;
  }

  th, td {
    font-size: 0.85rem;
  }

  button[type="submit"] {
    font-size: 0.85rem;
    padding: 8px;
  }
}
```

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-encuesta-escala-soledad',
  templateUrl: './encuesta-escala-soledad.component.html',
  styleUrls: ['./encuesta-escala-soledad.component.scss']
})
export class EncuestaEscalaSoledadComponent {
  // Preguntas de la escala
  preguntas = [
    { texto: 'Con qué frecuencia siente que le falta compañía.', respuesta: null },
    { texto: 'Con qué frecuencia se siente excluido/a.', respuesta: null },
    { texto: 'Con qué frecuencia se siente aislado/a de los demás.', respuesta: null
}
  ];

  constructor(private router: Router, private share: SharedService) {}

  // Calcular puntaje total y redirigir al componente de resultados
  calcularPuntaje() {
    const puntaje = this.preguntas.reduce((total, pregunta) => total +
(Number(pregunta.respuesta) || 0), 0);

    let observacion: string;
    if (puntaje <= 3) {
      observacion = 'Bajo nivel de soledad.';
    } else if (puntaje <= 6) {
      observacion = 'Nivel moderado de soledad.';
    } else {
      observacion = 'Alto nivel de soledad.';
    }

    // Llamar al método para enviar los resultados
    this.enviarResultados(puntaje, observacion);
  }

  // Método para enviar los resultados al componente de resultados
  enviarResultados(puntaje: number, observacion: string): void {
    if (puntaje === null) {
      alert('Primero calcula el puntaje antes de enviar los resultados.');
      return;
    }

    const resultado = {
      point: puntaje,
      encuesta: 'Escala de Soledad de 3 Elementos',
      observable: observacion,
      porcent: ((puntaje / 21) * 100).toFixed(2)
    }
```

```
    console.log('Puntaje: ', puntaje, '\nObservación: ', observacion);
    this.share.saveResults(resultado);
    // Redirige al componente de resultados con los datos mediante queryParams
    this.router.navigate(['home/resultado']);
  }
}

<!-- Encuesta GAI-SF -->
<div class="container">
    <h2>Inventario de Ansiedad Geriátrica Versión Corta (GAI-SF)</h2>
    <p>Conteste los enunciados de acuerdo con cómo se sintió la semana pasada.</p>
    <form #gaiForm="ngForm" (ngSubmit)="calcularPuntaje()">
      <table class="table table-bordered">
        <thead>
          <tr>
            <th>#</th>
            <th>Enunciados</th>
            <th>Coincide</th>
            <th>No coincide</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let pregunta of preguntas; let i = index">
            <td>{{ i + 1 }}</td>
            <td>{{ pregunta.texto }}</td>
            <td>
              <input
                type="radio"
                [id]="'pregunta-' + i + '-coincide'"
                name="pregunta-{{ i }}"
                value="1"
                [(ngModel)]="pregunta.respuesta"
                required
              />
            </td>
            <td>
              <input
                type="radio"
                [id]="'pregunta-' + i + '-no-coincide'"
                name="pregunta-{{ i }}"
                value="0"
                [(ngModel)]="pregunta.respuesta"
              />
            </td>
          </tr>
        </tbody>
      </table>
      <button type="submit" class="btn btn-primary" [disabled]="!gaiForm.valid">
        Finalizar Encuesta
      </button>
    </form>
  </div>
```

```
/* Contenedor principal */
.container {
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
    background-color: #ffffff;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  }

  /* Encabezado */
  h2 {
    text-align: center;
    color: #333333;
    margin-bottom: 20px;
    font-size: 1.8rem;
  }

  /* Tabla */
  table {
    width: 100%;
    border-collapse: collapse;
  }

  th, td {
    text-align: center;
    padding: 10px;
    font-size: 0.95rem;
  }

  thead th {
    background-color: #007bff;
    color: #ffffff;
  }

  tr:nth-child(even) {
    background-color: #f9f9f9;
  }

  tr:hover {
    background-color: #f1f1f1;
  }

  /* Botón */
  button[type="submit"] {
    width: 100%;
    padding: 10px;
    font-size: 1rem;
    font-weight: bold;
    color: #ffffff;
```

```
    background-color: #007bff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease-in-out;
  }

  button[type="submit"]:hover {
    background-color: #0056b3;
  }

  button[type="submit"]:disabled {
    background-color: #cccccc;
    cursor: not-allowed;
  }

  /* Estilos Responsivos */
  @media (max-width: 768px) {
    h2 {
      font-size: 1.5rem;
    }

    th, td {
      font-size: 0.9rem;
    }

    button[type="submit"] {
      font-size: 0.9rem;
    }
  }

  @media (max-width: 576px) {
    h2 {
      font-size: 1.3rem;
    }

    th, td {
      font-size: 0.85rem;
    }

    button[type="submit"] {
      font-size: 0.85rem;
      padding: 8px;
    }
  }
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-encuesta-gai-sf',
  templateUrl: './encuesta-gai-sf.component.html',
```

```
  styleUrls: ['./encuesta-gai-sf.component.scss']
})
export class EncuestaGaiSfComponent {
  // Preguntas de la encuesta
  preguntas = [
    { texto: 'Me paso mucho tiempo preocupado.', respuesta: null },
    { texto: 'Las pequeñas cosas me molestan mucho.', respuesta: null },
    { texto: 'Me considero una persona preocupada.', respuesta: null },
    { texto: 'A menudo me siento nervioso.', respuesta: null },
    { texto: 'Mis propios pensamientos me hacen sentir ansioso.', respuesta: null }
  ];

  constructor(private router: Router, private share: SharedService) {}

  // Calcular el puntaje total y redirigir al componente de resultados
  calcularPuntaje() {
    const puntaje = this.preguntas.reduce((total, pregunta) => total +
(Number(pregunta.respuesta) || 0), 0);

    let observacion: string;
    if (puntaje >= 3) {
      observacion = 'Posible presencia de ansiedad geriátrica.';
    } else {
      observacion = 'No se detecta ansiedad significativa.';
    }

    // Llamar al método para enviar los resultados
    this.enviarResultados(puntaje, observacion);
  }

  // Método para enviar los resultados al componente de resultados
  enviarResultados(puntaje: number, observacion: string): void {
    if (puntaje === null) {
      alert('Primero calcula el puntaje antes de enviar los resultados.');
      return;
    }

    const resultado = {
      point: puntaje,
      encuesta: 'GAI-SF',
      observable: observacion,
      porcent: ((puntaje / 21) * 100).toFixed(2)
    }

    console.log('Puntaje: ', puntaje, '\nObservación: ', observacion);
    this.share.saveResults(resultado);
    // Redirige al componente de resultados con los datos mediante queryParams
    this.router.navigate(['home/resultado']);
  }
}

<!-- Encuesta Okeeffe -->
```

```
<div class="container mt-4">
    <h2 class="text-center">Escala Corta de Depresión por Observación Hammond-
O'Keeffe</h2>

    <form [formGroup]="encuestaForm">
      <div class="table-responsive">
        <table class="table table-bordered text-center">
          <thead class="table-light">
            <tr>
              <th>Pregunta</th>
              <th>NO (0 puntos)</th>
              <th>SI (1 punto)</th>
            </tr>
          </thead>
          <tbody>
            <tr *ngFor="let pregunta of preguntas; let i = index">
              <td>{{ pregunta.texto }}</td>
              <td>
                <input
                  type="radio"
                  [value]="0"
                  [formControlName]="'pregunta' + i"
                />
              </td>
              <td>
                <input
                  type="radio"
                  [value]="1"
                  [formControlName]="'pregunta' + i"
                />
              </td>
            </tr>
          </tbody>
        </table>
      </div>

      <div class="text-center mt-3">
        <button type="button" class="btn btn-success" (click)="finalizarEncuesta()">
          Finalizar Encuesta
        </button>
      </div>
    </form>
  </div>


// Variables para la paleta de colores
$primary-color: #007bff;
$success-color: #28a745;
$danger-color: #dc3545;
$neutral-color: #6c757d;
$background-color: #f8f9fa;
$border-color: #dee2e6;
```

```
// Estilo general para el contenedor
.container {
  margin-top: 40px;
  background-color: $background-color;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  padding: 20px;
  max-width: 800px;
  margin-left: auto;
  margin-right: auto;
}

// Título de la encuesta
h2 {
  font-size: 1.75rem;
  color: $primary-color;
  font-weight: 600;
  text-align: center;
  margin-bottom: 15px;
}

// Instrucciones debajo del título
p {
  font-size: 1rem;
  color: $neutral-color;
  text-align: center;
  margin-bottom: 20px;
}

// Tabla de preguntas
.table {
  border: 1px solid $border-color;
  border-radius: 8px;
  width: 100%;
  margin-top: 20px;

  th {
    background-color: #f1f3f5;
    font-weight: bold;
    color: #343a40;
  }

  td {
    vertical-align: middle;
  }

  td input[type="radio"] {
    margin-right: 10px;
  }

  // Estilo para la tabla cuando es de formulario
```

```
  tbody tr {
    background-color: white;
    border-bottom: 1px solid $border-color;
    transition: background-color 0.3s ease;

    &:hover {
      background-color: #f1f3f5;
    }
  }
}

// Botones
button {
  padding: 10px 20px;
  font-size: 1rem;
  font-weight: 600;
  border-radius: 5px;
  border: none;
  transition: background-color 0.3s ease;

  &.btn-success {
    background-color: $success-color;
    color: white;

    &:hover {
      background-color: darken($success-color, 10%);
    }
  }

  &.btn-danger {
    background-color: $danger-color;
    color: white;

    &:hover {
      background-color: darken($danger-color, 10%);
    }
  }

  &.btn-primary {
    background-color: $primary-color;
    color: white;

    &:hover {
      background-color: darken($primary-color, 10%);
    }
  }
}

// Resultados
.text-center {
  margin-top: 30px;
}
```

```
h3 {
  font-size: 1.25rem;
  color: $primary-color;
  font-weight: 500;
}

.fw-bold {
  font-weight: bold;
}

p {
  font-size: 1rem;
  color: $neutral-color;
  margin-bottom: 20px;
}

// Mensajes de error o advertencia
.alert {
  margin-top: 20px;
  padding: 10px;
  border-radius: 5px;
  font-size: 1rem;
  color: white;
  background-color: $danger-color;
}

// Estilo de los radio buttons cuando son seleccionados
input[type="radio"]:checked {
  background-color: #28a745;
}

// Estilos responsivos para pantallas más pequeñas
@media (max-width: 767px) {
  .container {
    padding: 15px;
  }

  h2 {
    font-size: 1.5rem;
  }

  .table th, .table td {
    font-size: 0.875rem;
  }

  button {
    font-size: 0.875rem;
    padding: 8px 15px;
  }
}
```

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup } from '@angular/forms';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-encuesta-okeeffe',
  templateUrl: './encuesta-okeeffe.component.html',
  styleUrls: ['./encuesta-okeeffe.component.scss']
})
export class EncuestaOkeeffeComponent implements OnInit {
  encuestaForm!: FormGroup;
  preguntas = [
    { texto: 'Se observa al paciente la mayor parte del tiempo triste, infeliz o
deprimido ' },
    { texto: 'Alguna ves el paciente presenta llanto o parece lloroso' },
    { texto: 'El paciente parece estar agitado, inquieto o ancioso' },
    { texto: 'El paciente esta letárgico o refractario a movilizarse' },
    { texto: 'El paciente necesita mucho animo para hacer  las cosas por él (ella)'
},
    { texto: 'El paciente parece reservado, muestra minimo interes en el entorno
social' }
  ];
  resultado: number = 0;
  interpretacion: string = '';
  showErrors: boolean = false;

  constructor(private fb: FormBuilder, private router: Router, private share:
SharedService) {}

  ngOnInit(): void {
    // Crea un grupo de controles para cada pregunta
    this.encuestaForm = this.fb.group(
      this.preguntas.reduce((controls, _, index) => {
        controls['pregunta' + index] = [null];
        return controls;
      }, {} as { [key: string]: any })
    );
  }

  // Método para calcular la puntuación
  calcularPuntuacion(): void {
    const respuestas = this.encuestaForm.value;

    // Calcula la puntuación total
    this.resultado = (Object.values(respuestas) as (number | null)[]).reduce(
      (acc: number, val: number | null) => acc + (val || 0),
      0 // Inicializa el acumulador con 0
    );

    // Determina la interpretación
    this.interpretacion =
```

```
      this.resultado >= 3
        ? 'Depresión significativa'
        : 'Depresión leve o ausencia de depresión';
  }

  // Método para enviar los resultados
  enviarResultados(): void {
    if (this.resultado === 0 && !this.showErrors) {
      alert('Primero calcula el puntaje antes de enviar los resultados.');
      this.showErrors = true;
      return;
    }

    const resultado = {
      point: this.resultado,
      encuesta: 'Escala Corta de Depresión por Observación Hammond-O\'Keeffe',
      observable: this.interpretacion,
      porcent: ((this.resultado / 21) * 100).toFixed(2)
    }
    console.log('Puntaje: ', this.resultado, '\nObservación: ', this.interpretacion);
    this.share.saveResults(resultado);
    // Redirige al componente Resultados con los datos mediante el estado
    this.router.navigate(['home/resultado']);
  }

  // Método para finalizar la encuesta: calcula el puntaje y luego envía los
resultados
  finalizarEncuesta(): void {
    if (this.encuestaForm.valid) {
      this.calcularPuntuacion();  // Calcula la puntuación
      this.enviarResultados();    // Envía los resultados
    } else {
      alert('Por favor, complete todas las preguntas antes de finalizar.');
    }
  }
}

<!-- Encuesta PHQ9 -->
<div class="container">
    <h2>PHQ-9</h2>
    <form #phqForm="ngForm" (ngSubmit)="calcularPuntaje()">
      <div *ngFor="let pregunta of preguntas; let i = index" class="mb-3">
        <label>{{ i + 1 }}. {{ pregunta.texto }}</label>
        <div class="form-check" *ngFor="let opcion of opciones">
          <input
            type="radio"
            [id]="'pregunta-' + i + '-opcion-' + opcion.valor"
            class="form-check-input"
            name="pregunta-{{ i }}"
            [value]="opcion.valor"
            [(ngModel)]="pregunta.respuesta"
            required
```

```
            />
            <label
              class="form-check-label"
              [for]="'pregunta-' + i + '-opcion-' + opcion.valor"
            >
              {{ opcion.texto }}
            </label>
          </div>
        </div>

        <button type="submit" class="btn btn-primary" [disabled]="!phqForm.valid">
          Finalizar Encuesta
        </button>
      </form>
    </div>

/* Contenedor principal */
.container {
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    background-color: #ffffff;
    border-radius: 10px;
  }

  /* Encabezado */
  h2 {
    text-align: center;
    color: #0a38ce;
    font-size: 1.8rem;
    margin-bottom: 20px;
  }

  /* Preguntas */
  label {
    font-size: 1rem;
    color: #555555;
    display: block;
    margin-bottom: 8px;
  }

  /* Opciones de respuesta */
  .form-check {
    margin-bottom: 10px;
  }

  .form-check-input {
    margin-right: 8px;
  }

  .form-check-label {
```

```
    font-size: 0.95rem;
    color: #333333;
  }

  /* Botón de finalizar */
  button[type="submit"] {
    width: 100%;
    padding: 10px 15px;
    font-size: 1rem;
    font-weight: bold;
    color: #ffffff;
    background-color: #007bff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease-in-out;
  }

  button[type="submit"]:hover {
    background-color: #0056b3;
  }

  button[type="submit"]:disabled {
    background-color: #cccccc;
    cursor: not-allowed;
  }
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-encuesta-phq9',
  templateUrl: './encuesta-phq9.component.html',
  styleUrls: ['./encuesta-phq9.component.scss']
})
export class EncuestaPhq9Component {
  // Preguntas y opciones de la encuesta
  preguntas = [
    { texto: 'Poco interés o placer en hacer cosas', respuesta: null },
    { texto: 'Se ha sentido decaído(a), deprimido(a) o sin esperanzas', respuesta:
null },
    { texto: 'Ha tenido dificultad para quedarse o permanecer dormido(a), o ha
dormido demasiado', respuesta: null },
    { texto: 'Se ha sentido cansado(a) o con poca energía', respuesta: null },
    { texto: 'Sin apetito o ha comido en exceso', respuesta: null },
    { texto: 'Se ha sentido mal con usted mismo(a) – o que es un fracaso o que ha
quedado mal con usted mismo(a) o con su familia', respuesta: null },
    { texto: 'Ha tenido dificultad para concentrarse en ciertas actividades, tales
como leer el periódico o ver la televisión', respuesta: null },
    { texto: '¿Se ha movido o hablado tan lento que otras personas podrían haberlo
notado? O lo contrario – muy inquieto(a) o agitado(a)', respuesta: null },
```

```
    { texto: 'Pensamientos de que estaría mejor muerto(a) o de lastimarse de alguna
manera', respuesta: null }
  ];

  opciones = [
    { texto: 'Ningún día', valor: 0 },
    { texto: 'Varios días', valor: 1 },
    { texto: 'Más de la mitad de los días', valor: 2 },
    { texto: 'Casi todos los días', valor: 3 }
  ];

  constructor(private router: Router, private share: SharedService) {}

  // Calcular el puntaje total y redirigir al componente de resultados
  calcularPuntaje() {
    const puntaje = this.preguntas.reduce((total, pregunta) => total +
(pregunta.respuesta || 0), 0);
    let observacion: string;

    if (puntaje <= 4) {
      observacion = 'Mínima existencia o ausencia de síntomas depresivos.';
    } else if (puntaje <= 9) {
      observacion = 'Síntomas depresivos leves.';
    } else if (puntaje <= 14) {
      observacion = 'Síntomas depresivos moderados.';
    } else if (puntaje <= 19) {
      observacion = 'Síntomas depresivos moderados a graves.';
    } else {
      observacion = 'Síntomas depresivos graves.';
    }

    // Llamar al método para enviar los resultados
    this.enviarResultados(puntaje, observacion);
  }

  // Método para enviar los resultados al componente de resultados
  enviarResultados(puntaje: number, observacion: string): void {
    if (puntaje === null) {
      alert('Primero calcula el puntaje antes de enviar los resultados.');
      return;
    }

    const res = {
      point: puntaje,
      encuesta: 'PHQ-9',
      observable: observacion,
      porcent: ((puntaje / 21) * 100).toFixed(2)
    }

    console.log('Puntaje: ', puntaje, '\nObservación: ', observacion);
    this.share.saveResults(res);
    // Redirige al componente de resultados con los datos mediante queryParams
```

```
    this.router.navigate(['home/resultado']);
  }
}

<!-- Encuesta Sad Person -->
<div class="container">
    <h2>Escala SAD PERSONS</h2>
    <p>
      Por favor, responda las siguientes preguntas indicando "Sí" o "No" según
corresponda.
      Cada respuesta afirmativa suma 1 punto.
    </p>
    <form #sadForm="ngForm" (ngSubmit)="calcularPuntaje()">
      <table class="table table-bordered">
        <thead>
          <tr>
            <th>#</th>
            <th>Factor de riesgo</th>
            <th>Sí (1)</th>
            <th>No (0)</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let pregunta of preguntas; let i = index">
            <td>{{ i + 1 }}</td>
            <td>{{ pregunta.texto }}</td>
            <td>
              <input
                type="radio"
                [id]="'pregunta-' + i + '-1'"
                name="pregunta-{{ i }}"
                value="1"
                [(ngModel)]="pregunta.respuesta"
                required
              />
            </td>
            <td>
              <input
                type="radio"
                [id]="'pregunta-' + i + '-0'"
                name="pregunta-{{ i }}"
                value="0"
                [(ngModel)]="pregunta.respuesta"
              />
            </td>
          </tr>
        </tbody>
      </table>
      <button type="submit" class="btn btn-primary" [disabled]="!sadForm.valid">
        Finalizar Encuesta
      </button>
    </form>
```

```
  </div>


.container {
    max-width: 700px;
    margin: 0 auto;
    padding: 20px;
    background-color: #ffffff;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  }

  h2 {
    text-align: center;
    color: #333333;
    margin-bottom: 20px;
    font-size: 1.8rem;
  }

  p {
    font-size: 1rem;
    margin-bottom: 20px;
  }

  table {
    width: 100%;
    border-collapse: collapse;
  }

  th, td {
    text-align: center;
    padding: 10px;
    font-size: 0.95rem;
  }

  thead th {
    background-color: #007bff;
    color: #ffffff;
  }

  tr:nth-child(even) {
    background-color: #f9f9f9;
  }

  tr:hover {
    background-color: #f1f1f1;
  }

  button[type="submit"] {
    width: 100%;
    padding: 10px;
    font-size: 1rem;
```

```
    font-weight: bold;
    color: #ffffff;
    background-color: #007bff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease-in-out;
  }

  button[type="submit"]:hover {
    background-color: #0056b3;
  }

  button[type="submit"]:disabled {
    background-color: #cccccc;
    cursor: not-allowed;
  }

  @media (max-width: 768px) {
    h2 {
      font-size: 1.5rem;
    }

    th, td {
      font-size: 0.9rem;
    }

    button[type="submit"] {
      font-size: 0.9rem;
    }
  }

  @media (max-width: 576px) {
    h2 {
      font-size: 1.3rem;
    }

    th, td {
      font-size: 0.85rem;
    }

    button[type="submit"] {
      font-size: 0.85rem;
      padding: 8px;
    }
  }

import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
```

```
  selector: 'app-encuesta-sad-persons',
  templateUrl: './encuesta-sad-persons.component.html',
  styleUrls: ['./encuesta-sad-persons.component.scss']
})
export class EncuestaSADPERSONSComponent {
  // Factores de riesgo en la escala SAD PERSONS
  preguntas = [
    { texto: 'S: Sexo masculino', respuesta: null },
    { texto: 'A: Edad (Age) <20 o >45 años', respuesta: null },
    { texto: 'D: Depresión', respuesta: null },
    { texto: 'P: Tentativa suicida previa', respuesta: null },
    { texto: 'E: Abuso de alcohol o drogas', respuesta: null },
    { texto: 'R: Falta de pensamiento racional (psicosis o trastornos cognitivos)',
respuesta: null },
    { texto: 'S: Carencia de apoyo social', respuesta: null },
    { texto: 'O: Plan de suicidio organizado', respuesta: null },
    { texto: 'N: No pareja o cónyuge', respuesta: null },
    { texto: 'S: Enfermedad somática', respuesta: null }
  ];

  constructor(private router: Router, private share: SharedService) {}

  // Calcular puntaje y redirigir al componente de resultados
  calcularPuntaje() {
    const puntaje = this.preguntas.reduce((total, pregunta) => total +
(Number(pregunta.respuesta) || 0), 0);

    let observacion: string;
    if (puntaje <= 2) {
      observacion = 'Alta médica al domicilio con seguimiento ambulatorio';
    } else if (puntaje <= 4) {
      observacion = 'Seguimiento ambulatorio intensivo, considerando ingreso';
    } else if (puntaje <= 6) {
      observacion = 'Recomendado ingreso, sobre todo si hay ausencia de apoyo
social';
    } else {
      observacion = 'Ingreso obligatorio incluso en contra de su voluntad';
    }

    // Llamar al método para enviar los resultados
    this.enviarResultados(puntaje, observacion);
  }

  // Método para enviar los resultados al componente de resultados
  enviarResultados(puntaje: number, observacion: string): void {
    if (puntaje === null) {
      alert('Primero calcula el puntaje antes de enviar los resultados.');
      return;
    }
    const resultado = {
      point: puntaje,
      encuesta: 'Escala SAD PERSONS',
```

```
      observable: observacion,
      porcent: ((puntaje / 21) * 100).toFixed(2)
    }
    console.log('Puntaje: ', puntaje, '\nObservación: ', observacion);
    this.share.saveResults(resultado);
    // Redirige al componente de resultados con los datos mediante queryParams
    this.router.navigate(['home/resultado']);
  }
}
```

Encuesta de Categoría Nutricional

```
<!-- Encuesta Mini Assessment -->
<h1 class="titulo-encuesta">Mini Nutritional Assessment</h1>

<form [formGroup]="mnaForm" (ngSubmit)="enviarResultados()" class="container
encuesta-container">
  <!-- Sección de Cribaje -->
  <h2 class="subtitulo">Cribaje</h2>
  <table class="table table-striped">
    <thead>
      <tr>
        <th>Pregunta</th>
        <th>Opciones</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let pregunta of cribajePreguntas; let i = index">
        <td>
          <label>{{ pregunta.texto }}</label>
        </td>
        <td>
          <div *ngFor="let opcion of pregunta.opciones" class="form-check">
            <input
              type="radio"
              class="form-check-input"
              [formControlName]="'pregunta' + i"
              [value]="opcion.valor"
              id="cribajePregunta{{ i }}-opcion{{ opcion.valor }}"
            />
            <label
              class="form-check-label"
              for="cribajePregunta{{ i }}-opcion{{ opcion.valor }}"
            >
              {{ opcion.texto }}
            </label>
          </div>
        </td>
      </tr>
    </tbody>
  </table>
```

```
  <!-- Sección de Evaluación -->
  <h2 class="subtitulo">Evaluación</h2>
  <table class="table table-striped">
    <thead>
      <tr>
        <th>Pregunta</th>
        <th>Opciones</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let pregunta of evaluacionPreguntas; let i = index">
        <td>
          <label>{{ pregunta.texto }}</label>
        </td>
        <td>
          <!-- Preguntas normales -->
          <ng-container *ngIf="!pregunta.subPreguntas">
            <div *ngFor="let opcion of pregunta.opciones" class="form-check">
              <input
                type="radio"
                class="form-check-input"
                [formControlName]="'evaluacionPregunta' + i"
                [value]="opcion.valor"
                id="evaluacionPregunta{{ i }}-opcion{{ opcion.valor }}"
              />
              <label
                class="form-check-label"
                for="evaluacionPregunta{{ i }}-opcion{{ opcion.valor }}"
              >
                {{ opcion.texto }}
              </label>
            </div>
          </ng-container>

          <!-- Pregunta "K" con subpreguntas -->
          <ng-container *ngIf="pregunta.subPreguntas">
            <div formGroupName="consumePaciente">
              <div *ngFor="let subPregunta of pregunta.subPreguntas" class="form-
check">
                <input
                  type="checkbox"
                  class="form-check-input"
                  [formControlName]="subPregunta.texto"
                  id="subpregunta-{{ subPregunta.texto }}"
                />
                <label
                  class="form-check-label"
                  for="subpregunta-{{ subPregunta.texto }}"
                >
                  {{ subPregunta.texto }}
                </label>
              </div>
```

```
            </div>
          </ng-container>
        </td>
      </tr>
    </tbody>
  </table>

  <!-- Botón de enviar -->
  <button
    type="submit"
    [disabled]="mnaForm.invalid"
    class="btn btn-primary mt-3"
  >
    Enviar Resultados
  </button>
</form>

/* Contenedor principal */
.container.encuesta-container {
  max-width: 800px;
  margin: 20px auto;
  padding: 20px;
  background-color: #ffffff;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

/* Título de la encuesta */
.titulo-encuesta {
  text-align: center;
  font-size: 2rem;
  font-weight: bold;
  color: #4a4a4a;
  margin-bottom: 20px;
}

/* Subtítulos */
.subtitulo {
  font-size: 1.5rem;
  font-weight: bold;
  color: #007bff;
  margin-top: 20px;
  margin-bottom: 10px;
  text-align: left;
  border-bottom: 2px solid #007bff;
}

/* Estilo para la tabla */
table.encuesta-table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 20px;
```

```
}

table.encuesta-table th,
table.encuesta-table td {
  padding: 10px;
  text-align: left;
  border: 1px solid #ddd;
}

table.encuesta-table th {
  background-color: #007bff;
  color: #fff;
  font-weight: bold;
  font-size: 1rem;
}

table.encuesta-table td {
  font-size: 1rem;
  color: #555;
}

/* Estilo para las opciones */
table.encuesta-table label {
  font-size: 1rem;
  color: #555;
}

input[type="radio"] {
  margin-right: 5px;
}

/* Botón de envío */
button.btn {
  display: block;
  width: 100%;
  padding: 10px;
  font-size: 1.2rem;
  font-weight: bold;
  color: #fff;
  background-color: #007bff;
  border: none;
  border-radius: 5px;
  transition: background-color 0.3s ease;
  margin-top: 20px;
}

button.btn:hover {
  background-color: #0056b3;
}

/* Estilo para texto */
p {
```

```
  margin-top: 10px;
  font-size: 1rem;
  color: #666;
}

/* Responsividad */
@media (max-width: 600px) {
  .titulo-encuesta {
    font-size: 1.5rem;
  }

  .subtitulo {
    font-size: 1.2rem;
  }

  table.encuesta-table th,
  table.encuesta-table td {
    font-size: 0.8rem;
    padding: 8px;
  }

  button.btn {
    font-size: 1rem;
  }
}

.table {
  width: 100%;
  margin-bottom: 20px;
  border-collapse: collapse;
}

.table th, .table td {
  padding: 10px;
  vertical-align: middle;
}

.table th {
  background-color: #f2f2f2;
  color: #333;
}

.table-striped tbody tr:nth-child(odd) {
  background-color: #f9f9f9;
}

@media (max-width: 768px) {
  .table {
    font-size: 0.9rem;
  }
}
import { Component } from '@angular/core';
```

```
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

interface MNAForm {
  [key: string]: number | null;
}

@Component({
  selector: 'app-encuesta-assessment',
  templateUrl: './encuesta-assessment.component.html',
  styleUrls: ['./encuesta-assessment.component.scss']
})

export class EncuestaAssessmentComponent {
  mnaForm: FormGroup;
  cribajePreguntas = [
    {
      texto: 'A. ¿Ha perdido el apetito?',
      opciones: [
        { texto: 'Ha comido mucho menos', valor: 0 },
        { texto: 'Ha comido menos', valor: 1 },
        { texto: 'Ha comido igual', valor: 2 },
      ],
    },
    {
      texto: 'B. Pérdida reciente de peso (< 3 meses)',
      opciones: [
        { texto: 'Pérdida de peso > 3 kg', valor: 0 },
        { texto: 'No lo sabe', valor: 1 },
        { texto: 'Pérdida de peso entre 1 y 3 kg', valor: 2 },
        { texto: 'No ha habido pérdida de peso', valor: 3 },
      ],
    },
    {
      texto: 'C. Movilidad ',
      opciones: [
        { texto: 'de la cama al sillon', valor: 0 },
        { texto: 'autonomia en el interior', valor: 1 },
        { texto: 'sale del domicilio', valor: 2},
      ],
    },
    {
      texto: 'D. Ha tenido una enfermedad aguda o situacion de estres psicologico en
los ultimos 3 meses',
      opciones: [
        { texto: 'Si', valor: 0 },
        { texto: 'No', valor: 2 },
      ],
    },
    {
      texto: 'E. Problemas neuropsicologicos',
```

```
      opciones: [
        { texto: 'Demensia o depresion grave', valor: 0 },
        { texto: 'Demensia moderada', valor: 1 },
        { texto: 'Sin problemas psicologicos', valor: 2 },
      ],
    },
    {
      texto: 'F. Indice de masa corporal (IMC) = Peso en Kg/(talla en m2)',
      opciones: [
        { texto: 'IMC < 19', valor: 0 },
        { texto: '19 < IMC < 21', valor: 1 },
        { texto: '21 < IMC < 23', valor: 2 },
        { texto: 'IMC > 23', valor: 3 },
      ],
    },
  ];

  evaluacionPreguntas = [
    {
      texto: 'G. ¿El paciente vive independientemente?',
      opciones: [
        { texto: 'Sí', valor: 1 },
        { texto: 'No', valor: 0 },
      ],
    },
    {
      texto: 'H. ¿Toma más de 3 medicamentos al día?',
      opciones: [
        { texto: 'Sí', valor: 0 },
        { texto: 'No', valor: 1 },
      ],
    },
    {
      texto: 'I. Ulceras o lesiones cutaneas?',
      opciones: [
        { texto: 'Sí', valor: 0 },
        { texto: 'No', valor: 1 },
      ],
    },
    {
      texto: 'J. Cuantas Comidas comidas toma al dia?',
      opciones: [
        { texto: '1 comida', valor: 0 },
        { texto: '2 comidas', valor: 1 },
        { texto: '3 comidas', valor: 2 },
      ],
    },
    {
      texto: 'K. Consume el paciente alguno de los siguientes grupos alimenticios
regularmente:',
      subPreguntas: [
        { texto: 'Lácteos', valor: 0 },
```

```
        { texto: 'Huevos y legumbres', valor: 0 },
        { texto: 'Carne, pescado o aves', valor: 0 },
      ],
    },
    {
      texto: 'L. consume Frutas o verduras al menos 2 veces al dia ?',
      opciones: [
        { texto: 'No', valor: 0 },
        { texto: 'Si', valor: 1 },
      ],
    },
    {
      texto: 'M. cuantos vasos de agua u otros liquidos toma al dia? (agua, zumo,
cafe, té, leche, vino, cerveza...)',
      opciones: [
        { texto: 'Menos de 3 vasos', valor: 0.0 },
        { texto: 'de 3 a 5 vasos', valor: 0.5 },
        { texto: 'mas de 5 vasos', valor: 1.0 },
      ],
    },
    {
      texto: 'N. Forma de alimentarse',
      opciones: [
        { texto: 'Necesita ayuda', valor: 0 },
        { texto: 'se alimenta solo con dificultad', valor: 1 },
        { texto: 'se alimenta solo sin dificultad', valor: 2 },
      ],
    },
    {
      texto: 'O. Se considera el pacente que esta bien nutrido?',
      opciones: [
        { texto: 'Mal nutricion grave', valor: 0 },
        { texto: 'No lo sabe o malnutricion moderada', valor: 1 },
        { texto: 'sin problemas de nutricion', valor: 2 },
      ],
    },
    {
      texto: 'P. En comparacion con las personas en su edad, como encuestra el
paciente su salud?',
      opciones: [
        { texto: 'peor', valor: 0.0 },
        { texto: 'No lo sabe', valor: 0.5 },
        { texto: 'igual', valor: 1.0 },
        { texto: 'Mejor', valor: 2.0 },
      ],
    },
    {
      texto: 'Q. circunferencia branquial (CD en cm)',
      opciones: [
        { texto: 'CB < 21', valor: 0.0 },
        { texto: '21 < CB < 22', valor: 0.5 },
        { texto: 'CB > 22', valor: 1.0 },
```

```
      ],
    },
    {
      texto: 'R. Circunferencia de la pantorrilla (CP en cm)',
      opciones: [
        { texto: 'CP < 31', valor: 0 },
        { texto: 'CP > 31', valor: 1 },
      ],
    },
  ];


  constructor(private fb: FormBuilder, private router: Router, private share:
SharedService) {
    const formControls: any = {};

    // Configurar controles para las preguntas de cribaje
    this.cribajePreguntas.forEach((_, index) => {
      formControls[`pregunta${index}`] = [null, Validators.required];
    });

    // Configurar controles para las preguntas de evaluación
    this.evaluacionPreguntas.forEach((pregunta, index) => {
      if (!pregunta.subPreguntas) {
        formControls[`evaluacionPregunta${index}`] = [null, Validators.required];
      }
    });

    // Configurar controles para las subpreguntas de "K"
    const consumePacienteControls: any = {};
    this.evaluacionPreguntas
      .find(p => p.texto.startsWith('K.'))?.subPreguntas
      ?.forEach(subPregunta => {
        consumePacienteControls[subPregunta.texto] = [false]; // Checkbox
inicializado en `false`
      });

      formControls['consumePaciente'] = this.fb.group({
        'Lácteos': [null,Validators.required],
        'Huevos y legumbres': [null,Validators.required],
        'Carne, pescado o aves': [null,Validators.required],
      });

    this.mnaForm = this.fb.group(formControls);
  }

  enviarResultados() {
    if (this.mnaForm.invalid) {
      alert('Completa todas las preguntas antes de enviar.');
      return;
    }
```

```
    const puntajeCribaje = this.calculateScore(this.cribajePreguntas, 'pregunta');
    const puntajeEvaluacion = this.calculateScore(this.evaluacionPreguntas,
'evaluacionPregunta');
    const puntajeK = this.calculateConsumePacienteScore();
    const puntajeTotal = puntajeCribaje + puntajeEvaluacion + puntajeK;

    let observacion;
    if (puntajeTotal >= 24) {
      observacion = 'Estado nutricional normal';
    } else if (puntajeTotal >= 17) {
      observacion = 'Riesgo de desnutrición';
    } else {
      observacion = 'Desnutrición';
    }

    console.log('Puntaje total:', puntajeTotal, 'Resultado:', observacion);

    const resultado = {
      point: puntajeTotal,
      encuesta: 'Mini Nutritional Assessment',
      observable: observacion,
      porcent: ((puntajeTotal/30) * 100).toFixed(2)
    }
    this.share.saveResults(resultado);
    this.router.navigate(['home/resultado']);
  }

  // Calcular el puntaje de las preguntas normales
  calculateScore(preguntas: any[], prefix: string): number {
    return preguntas.reduce((total, pregunta, index) => {
      if (pregunta.subPreguntas) return total; // Excluir preguntas con subpreguntas
      const respuesta = this.mnaForm.get(`${prefix}${index}`)?.value;
      return total + (respuesta !== null ? Number(respuesta) : 0);
    }, 0);
  }

  // Calcular el puntaje para la pregunta "K" (subpreguntas)
  calculateConsumePacienteScore(): number {
    const consumePaciente = this.mnaForm.get('consumePaciente')?.value;

    if (!consumePaciente) return 0;

    // Contar subpreguntas seleccionadas (checkboxes marcados como `true`)
    const seleccionados = Object.values(consumePaciente).filter(value =>
value).length;

    if (seleccionados === 1) return 0;
    if (seleccionados === 2) return 0.5;
    if (seleccionados >= 3) return 1.0;

    return 0;
  }
```

```
}

<!-- Encuesta Assessment -->
<div class="encuesta-container">
  <h1 class="encuesta-title">Mini Nutritional Assessment SF</h1>
  <form [formGroup]="mnaForm" (ngSubmit)="enviarResultados()">
    <!-- Pregunta 1 -->
    <div class="form-group">
      <label>A. ¿Ha perdido el apetito? Ha comido menos por falta de apetito,
problemas digestivos, dificultades de masticación o deglución en los últimos 3
meses?</label>
      <div>
        <label><input type="radio" formControlName="question1" value="0" /> Ha comido
mucho menos</label>
        <label><input type="radio" formControlName="question1" value="1" /> Ha comido
menos</label>
        <label><input type="radio" formControlName="question1" value="2" /> Ha comido
igual</label>
      </div>
    </div>

    <!-- Pregunta 2 -->
    <div class="form-group">
      <label>B. ¿Ha perdido peso recientemente (menor a 3 meses)?</label>
      <div>
        <label><input type="radio" formControlName="question2" value="0" /> Pérdida
de peso mayor a 3 kilos</label>
        <label><input type="radio" formControlName="question2" value="1" /> No lo
sabe</label>
        <label><input type="radio" formControlName="question2" value="2" /> Pérdida
de peso entre 1 y 3 kilos</label>
        <label><input type="radio" formControlName="question2" value="3" /> No ha
habido pérdida de peso</label>
      </div>
    </div>

    <!-- Pregunta 3 -->
    <div class="form-group">
      <label>C. Movilidad</label>
      <div>
        <label><input type="radio" formControlName="question3" value="0" /> De la
cama al sillón</label>
        <label><input type="radio" formControlName="question3" value="1" /> Autonomía
en el interior</label>
        <label><input type="radio" formControlName="question3" value="2" /> Sale del
domicilio</label>
      </div>
    </div>

    <!-- Pregunta 4 -->
    <div class="form-group">
```

```
      <label>D. ¿Ha tenido una enfermedad aguda o situación de estrés psicológico en
los últimos 3 meses?</label>
      <div>
        <label><input type="radio" formControlName="question4" value="0" />
Sí</label>
        <label><input type="radio" formControlName="question4" value="2" />
No</label>
      </div>
    </div>

    <!-- Pregunta 5 -->
    <div class="form-group">
      <label>E. Problemas neuropsicológicos</label>
      <div>
        <label><input type="radio" formControlName="question5" value="0" /> Demencia
o depresión grave</label>
        <label><input type="radio" formControlName="question5" value="1" /> Demencia
moderada</label>
        <label><input type="radio" formControlName="question5" value="2" /> Sin
problemas psicológicos</label>
      </div>
    </div>

    <!-- Pregunta 6 -->
    <div class="form-group">
      <label>F1. Índice de masa corporal (IMC) = Peso en KG / (Talla en m²)</label>
      <div>
        <label><input type="radio" formControlName="question6" value="0" /> IMC &#60;
19 kg/m²</label>
        <label><input type="radio" formControlName="question6" value="1" /> IMC entre
19 y 21 kg/m²</label>
        <label><input type="radio" formControlName="question6" value="2" /> IMC entre
21 y 23 kg/m²</label>
        <label><input type="radio" formControlName="question6" value="3" /> IMC &#62;
23 kg/m²</label>
      </div>
    </div>

    <!-- Pregunta 7 -->
    <div class="form-group">
      <label>F2. Perímetro de pantorrilla (cm)</label>
      <div>
        <label><input type="radio" formControlName="question7" value="0" /> &#60; 31
cm</label>
        <label><input type="radio" formControlName="question7" value="1" /> &#62; 31
cm</label>
      </div>
    </div>

    <!-- Botón de envío -->
    <button type="button" class="btn btn-primary mt-3"
(click)="enviarResultados()">Finalizar y Enviar</button>
```

```
  </form>
</div>
```

```
.encuesta-container {
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
    background-color: #f9f9f9;
    border-radius: 8px;
    box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
  }

  .encuesta-title {
    font-size: 1.8rem;
    font-weight: bold;
    text-align: center;
    color: #333;
    margin-bottom: 20px;
    border-bottom: 2px solid #007bff;
    padding-bottom: 10px;
  }

  .form-group {
    margin-bottom: 20px;

    label {
      font-size: 1rem;
      font-weight: 500;
      color: #555;
    }

    input[type="radio"] {
      margin-right: 8px;
    }
  }

  button {
    display: block;
    width: 100%;
    font-size: 1rem;
    padding: 10px;
    background-color: #007bff;
    border: none;
    color: #fff;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s ease;

    &:hover {
      background-color: #0056b3;
```

```
    }
  }

  /* Estilos responsivos */
  @media (max-width: 600px) {
    .encuesta-title {
      font-size: 1.5rem;
    }

    button {
      font-size: 0.9rem;
      padding: 8px;
    }
  }
```

```
 import { Component } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-encuesta-assessment-sf',
  templateUrl: './encuesta-assessment-sf.component.html',
  styleUrls: ['./encuesta-assessment-sf.component.scss']
})
export class EncuestaAssessmentSFComponent {
  mnaForm: FormGroup;
  showErrors: boolean = false;

  constructor(private fb: FormBuilder, private router: Router, private share:
SharedService) {
    this.mnaForm = this.fb.group({
      question1: [null, Validators.required],
      question2: [null, Validators.required],
      question3: [null, Validators.required],
      question4: [null, Validators.required],
      question5: [null, Validators.required],
      question6: [null, Validators.required],
      question7: [null, Validators.required]
    });
  }

  /**
   * Calcula el puntaje total basado en las respuestas.
   */
  calculateScore(): number {
    const values = this.mnaForm.value;
    return Object.keys(values)
      .map(key => Number(values[key]))
      .reduce((total, value) => total + value, 0);
  }
```

```
  /**
   * Genera la observación basada en el puntaje obtenido.
   */
  getObservation(puntaje: number): string {
    if (puntaje >= 12 && puntaje <= 14) {
      return 'Sin desnutrición';
    } else if (puntaje >= 8 && puntaje <= 11) {
      return 'Riesgo de desnutrición';
    } else if (puntaje >= 0 && puntaje <= 7) {
      return 'Desnutrición';
    } else {
      return 'Puntaje inválido';
    }
  }

  /**
   * Envía los resultados de la encuesta al componente de resultados.
   */
  enviarResultados(): void {
    if (!this.mnaForm.valid) {
      alert('Completa todas las preguntas antes de enviar los resultados.');
      this.showErrors = true;
      return;
    }

    const puntaje = this.calculateScore();
    const observacion = this.getObservation(puntaje);

    console.log('Puntaje: ', puntaje, '\nObservación: ', observacion);

    const resultado = {
      point: puntaje,
      encuesta: 'Mini Nutritional Assessment SF',
      observable: observacion,
      porcent: ((puntaje/14) * 100).toFixed(2)
    }
    this.share.saveResults(resultado);
    this.router.navigate(['home/resultado']);
  }
}

<!-- Encuesta Chumlea -->
<div class="encuesta-chumlea">
    <h1>Fórmulas Chumlea</h1>
    <form [formGroup]="chumleaForm" (ngSubmit)="calcularResultados()">
      <h2>Ingresar Medidas</h2>
      <div class="form-group">
        <!-- Entrada de datos -->
        <label for="sexo">Sexo</label>
        <select id="sexo" formControlName="sexo">
          <option value="mujer">Mujer</option>
          <option value="hombre">Hombre</option>
```

```
        </select>

        <label for="circBrazo">Circunferencia del brazo (cm)</label>
        <input type="number" id="circBrazo" formControlName="circBrazo" />

        <label for="circPantorrilla">Circunferencia de la pantorrilla (cm)</label>
        <input type="number" id="circPantorrilla" formControlName="circPantorrilla"
/>

        <label for="pliegueSubescapular">Pliegue subescapular (mm)</label>
        <input type="number" id="pliegueSubescapular"
formControlName="pliegueSubescapular" />

        <label for="alturaRodilla">Altura de la rodilla (cm)</label>
        <input type="number" id="alturaRodilla" formControlName="alturaRodilla" />

        <label for="alturaTalonRodilla">Altura talón-rodilla (cm)</label>
        <input type="number" id="alturaTalonRodilla"
formControlName="alturaTalonRodilla" />

        <label for="mediaEnvergadura">Media envergadura del brazo (cm)</label>
        <input type="number" id="mediaEnvergadura" formControlName="mediaEnvergadura"
/>

        <label for="edad">Edad (años)</label>
        <input type="number" id="edad" formControlName="edad" />
      </div>

      <!-- Botón para calcular -->
      <button type="submit" [disabled]="!chumleaForm.valid">Calcular</button>
    </form>

    <div *ngIf="resultado" class="resultados">
      <h2>Resultados</h2>
      <p>Peso estimado: {{ resultado.peso }} kg</p>
      <p>Talla estimada: {{ resultado.talla }} cm</p>
      <p>Estatura por media envergadura: {{ resultado.estatura }} cm</p>
    </div>
  </div>


.encuesta-chumlea {
    text-align: center;
    padding: 20px;

    h1, h2 {
      color: #2a4d8c;
    }

    .form-group {
      display: flex;
      flex-direction: column;
```

```
      gap: 15px;
      align-items: center;

      label {
        font-weight: bold;
      }

      input, select {
        width: 50%;
        padding: 10px;
        font-size: 16px;
        border: 1px solid #ccc;
        border-radius: 5px;
      }
    }

    button {
      margin-top: 20px;
      padding: 10px 20px;
      background-color: #2a4d8c;
      color: #fff;
      border: none;
      border-radius: 5px;
      cursor: pointer;

      &:hover {
        background-color: #1f3a6f;
      }

      &:disabled {
        background-color: #ccc;
        cursor: not-allowed;
      }
    }

    .resultados {
      margin-top: 20px;
      font-size: 18px;

      p {
        margin: 10px 0;
      }
    }
  }
```

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
@Component({
  selector: 'app-encuesta-chumlea',
  templateUrl: './encuesta-chumlea.component.html',
  styleUrls: ['./encuesta-chumlea.component.scss']
})
```

```
export class EncuestaChumleaComponent implements OnInit {
  chumleaForm: FormGroup;
  resultado: any;

  constructor(private fb: FormBuilder) {
    this.chumleaForm = this.fb.group({
      sexo: ['', Validators.required],
      circBrazo: [null, Validators.required],
      circPantorrilla: [null, Validators.required],
      pliegueSubescapular: [null, Validators.required],
      alturaRodilla: [null, Validators.required],
      alturaTalonRodilla: [null, Validators.required],
      mediaEnvergadura: [null, Validators.required],
      edad: [null, Validators.required],
    });
  }

  ngOnInit(): void {}

  calcularResultados(): void {
    const formValues = this.chumleaForm.value;
    const sexo = formValues.sexo;

    // Cálculo de peso
    const peso =
      sexo === 'mujer'
        ? this.calcularPesoMujer(formValues)
        : this.calcularPesoHombre(formValues);

    // Cálculo de talla
    const talla =
      sexo === 'mujer'
        ? this.calcularTallaMujer(formValues)
        : this.calcularTallaHombre(formValues);

    // Cálculo de estatura por media envergadura
    const estatura =
      sexo === 'mujer'
        ? 1.35 * formValues.mediaEnvergadura + 60.1
        : 1.4 * formValues.mediaEnvergadura + 57.8;

    // Guardar resultados
    this.resultado = {
      peso: peso.toFixed(2),
      talla: talla.toFixed(2),
      estatura: estatura.toFixed(2),
    };
  }

  calcularPesoMujer(values: any): number {
    return (
      values.circBrazo * 0.98 +
```

```
      values.circPantorrilla * 1.27 +
      values.pliegueSubescapular * 0.4 +
      values.alturaRodilla * 0.87 -
      62.35
    );
  }

  calcularPesoHombre(values: any): number {
    return (
      values.circBrazo * 1.73 +
      values.circPantorrilla * 0.98 +
      values.pliegueSubescapular * 0.37 +
      values.alturaRodilla * 1.16 -
      81.69
    );
  }

  calcularTallaMujer(values: any): number {
    return 1.83 * values.alturaTalonRodilla - 0.24 * values.edad + 84.88;
  }

  calcularTallaHombre(values: any): number {
    return 2.02 * values.alturaTalonRodilla - 0.04 * values.edad + 64.19;
  }
}

<!-- Encuesta Disfagia -->
<div class="image-container">
    <h1>Exploracion de Disfagia</h1>
    <img src="assets/Imagenes/disfagia.png" alt="Exploracion de Disfagia"
class="responsive-image" />
  </div>
.image-container {
    text-align: center; // Centra la imagen y título
    margin: 0 auto;
    padding: 10px;

    h1 {
      font-size: 24px;
      margin-bottom: 15px;
      color: #333;
    }

    .responsive-image {
      max-width: 100%; // La imagen no excederá el ancho de su contenedor
      height: auto;    // Mantiene la proporción de la imagen
      border: 1px solid #ddd; // Opcional: bordes sutiles
      border-radius: 8px;     // Bordes redondeados
      box-shadow: 0px 2px 5px rgba(0, 0, 0, 0.2); // Sombra ligera
    }
  }
  @media (max-width: 768px) {
```

```
    .image-container {
      h1 {
        font-size: 20px; // Reduce el tamaño del título en pantallas pequeñas
      }
    }
  }

<!-- Encuesta EAT-10 -->
<div class="d-flex justify-content-center align-items-center mt-4">
    <div class="survey-container p-4 shadow rounded">
      <h2 class="title">Encuesta EAT-10</h2>
      <div class="instrucciones">
        Responda cada pregunta escribiendo en el recuadro el n&#250;mero de puntos.
<br>
        &#191;Hasta que punto usted percibe los siguientes problemas&#63;
      </div>
      <!-- Formulario de la encuesta -->
      <form (ngSubmit)="calculateScore()">
        <div *ngFor="let question of questions; let i = index" class="mb-4">
          <p class="fw-bold">{{ question.text }}</p>

          <!-- Opciones de respuestas -->
          <div *ngFor="let option of question.option">
            <div class="form-check opciones">
              <input
                type="radio"
                class="form-check-input"
                [id]="'q' + i + option.score"
                [name]="'question' + i"
                [value]="option.score"
                [(ngModel)]="question.selectedScore"
                (change)="onOptionSelected()"
                [ngModelOptions]="{standalone: true}"
              />
              <label class="form-check-label" [for]="'q' + i + option.score">
                {{ option.label }}
              </label>
            </div>
          </div>
        </div>
        <!-- Error si faltan preguntas -->
        <div *ngIf="showError" class="alert alert-danger mt-3 text-center">
          ¡Por favor, responda todas las preguntas antes de enviar la encuesta!
        </div>

        <!-- Botón de enviar -->
        <div class="text-center">
          <button type="submit" class="btn btn-success px-4">Finaliza
Encuesta</button>
        </div>
      </form>
    </div>
```

```
  </div>

/* Contenedor principal de la encuesta */
.survey-container {
    max-width: 900px; /* Limita el ancho máximo */
    width: 100%;
    background-color: #ffffff; /* Fondo blanco */
  }

  /* Texto principal */
  h2 {
    font-size: 1.8rem;
  }

  /* Pregunta */
  p {
    font-size: 1.1rem;
  }

  /* Input de radio */
  .form-check-input {
    margin-right: 10px;
  }

  .opciones{
    cursor: pointer;
    max-width: max-content;
    label, input{
        cursor: pointer;
        margin-bottom: 13px;
    }
  }

  /* Botón de enviar */
  .btn-primary {
    background-color: #007bff;
    border: none;
    font-size: 1rem;
  }

  /* Contenedor centrado */
  .d-flex {
    min-height: 100vh;
  }
  .title {
    width: 100%;
    background: #9B59B6;
    padding: 10px;
    text-align: center;
    color: white;
  }
  .instrucciones {
```

```
    font-size: 1.3rem;
    padding: 5px;
    width: 100%;
    margin: 15px 0;
    color: rgba(255, 0, 0, 0.6);
    font-weight: bold;
  }

import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';
import { Option, QuestionSelec } from 'src/app/Shared/Data';

@Component({
  selector: 'app-encuesta-eat10',
  templateUrl: './encuesta-eat10.component.html',
  styleUrls: ['./encuesta-eat10.component.scss']
})
export class EncuestaEat10Component {
  totalScore: number = 0;
  showError: boolean = false;
  message: string = '';
  // Opciones de respuestas con puntajes
  options: Option[] = [
    { label: 'Ningún problema', score: 0 },
    { label: 'Poco problema', score: 1 },
    { label: 'Problemático', score: 2 },
    { label: 'Muy problemático', score: 3 },
    { label: 'Es un problema serio', score: 4 },
  ];

  questions: QuestionSelec[] = [
    { text: '1. Mi problema para tragar me ha llevado a perder peso', option:
this.options },
    { text: '2. Mi problema para tragar interfiere con mi capacidad para comer fuera
de casa', option: this.options },
    { text: '3. Tragar líquidos me supone un esfuerzo extra', option: this.options },
    { text: '4. Tragar sólidos me supone un esfuerzo extra', option: this.options },
    { text: '5. Tragar pastillas me supone un esfuerzo extra', option: this.options
},
    { text: '6. Tragar es doloroso', option: this.options },
    { text: '7. El placer de comer se ve afectado por mi problema para tragar',
option: this.options },
    { text: '8. Cuando trago, la comida se pega en mi garganta', option: this.options
},
    { text: '9. Toso cuando como', option: this.options },
    { text: '10. Tragar es estresante', option: this.options },
  ];

  constructor(private router: Router, private share: SharedService){}

  // Método para verificar si todas las preguntas tienen respuestas
```

```
  allQuestionsAnswered(): boolean {
    return this.questions.every(question => question.selectedScore != undefined);
  }

  // Calcular el puntaje total
  calculateScore() {
    // Si no todas las preguntas tienen respuestas, muestra error
    if (!this.allQuestionsAnswered()) {
      this.showError = true;
      return;
    }

    this.showError = false;
    this.totalScore = 0;

    for (let question of this.questions) {
      if (question.selectedScore == undefined) {
        this.showError = true;
        return;
      }
      this.totalScore += question.selectedScore;
    }
    this.nextComponent(this.totalScore);
  }

   // Se ejecuta cuando el usuario selecciona una opción
   onOptionSelected(): void {
    if (this.allQuestionsAnswered()) {
      this.showError = false; // Oculta el error automáticamente
    }
  }

  nextComponent(points: number){
    if(points >= 0 && points <= 10){
      this.message = 'No hay un problema significativo con la deglución.';
    } else if(points >= 11 && points <= 20){
      this.message = 'Hay una dificultad leve para tragar.';
    } else if(points >= 21 && points <= 30){
      this.message = 'Se observa una dificultad moderada para tragar.';
    } else {
      this.message = 'Existe una dificultad severa para tragar. Consulte a un
especialista.';
    }

    const resultado = {
      point: points,
      encuesta: 'EAT-10',
      observable: this.message,
      porcent: ((points/40) * 100).toFixed(2)
    }
    this.share.saveResults(resultado);
    this.router.navigate(['home/resultado']);
```

```
  }
}

<!-- Encuesta GLIM -->
<div class="encuesta-glim">
    <h2>Criterios GLIM</h2>
    <p>La desnutrición requiere al menos 1 criterio fenotípico y 1 etiológico:</p>

    <form [formGroup]="glimForm" (ngSubmit)="enviarResultados()">
      <h3>Criterios Fenotípicos</h3>
      <div class="criterio">
        <label for="perdidaPeso">
          <strong>Pérdida de peso involuntaria:</strong> ¿Ha perdido peso en los
últimos meses?
        </label>
        <select id="perdidaPeso" formControlName="perdidaPeso" class="form-control">
          <option value="0">No</option>
          <option value="1">Sí, más del 5% en los últimos 6 meses</option>
          <option value="2">Sí, más del 10% en más de 6 meses</option>
        </select>
      </div>

      <div class="criterio">
        <label for="imc">
          <strong>Índice de masa corporal bajo:</strong> ¿Cuál es su IMC?
        </label>
        <select id="imc" formControlName="imc" class="form-control">
          <option value="0">≥20 (menor de 70 años)</option>
          <option value="1">≥22 (70 años o más)</option>
          <option value="2">&lt;20 (menor de 70 años)</option>
          <option value="3">&lt;22 (70 años o más)</option>
        </select>
      </div>

      <div class="criterio">
        <label for="masaMuscular">
          <strong>Reducción de la masa muscular:</strong> ¿Ha tenido reducción de
masa muscular según técnicas validadas?
        </label>
        <select id="masaMuscular" formControlName="masaMuscular" class="form-
control">
          <option value="0">No</option>
          <option value="1">Sí</option>
        </select>
      </div>

      <h3>Criterios Etiológicos</h3>
      <div class="criterio">
        <label for="ingestaAlimentos">
          <strong>Disminución de la ingesta o asimilación de alimentos:</strong>
        </label>
```

```
        <select id="ingestaAlimentos" formControlName="ingestaAlimentos" class="form-
control">
          <option value="0">No</option>
          <option value="1">Sí, ≤50% durante 1 semana</option>
          <option value="2">Sí, ≤100% durante 2 semanas</option>
        </select>
      </div>

      <div class="criterio">
        <label for="cargaInflamatoria">
          <strong>Carga inflamatoria:</strong> ¿Tiene alguna condición inflamatoria
activa?
        </label>
        <select id="cargaInflamatoria" formControlName="cargaInflamatoria"
class="form-control">
          <option value="0">No</option>
          <option value="1">Sí, lesión/inflamación aguda</option>
          <option value="2">Sí, patología inflamatoria crónica</option>
        </select>
      </div>

      <button type="submit" [disabled]="glimForm.invalid" class="btn btn-
primary">Enviar Resultados</button>
    </form>
  </div>


.encuesta-glim {
    max-width: 600px;
    margin: 20px auto;
    padding: 20px;
    background-color: #f9f9f9;
    border: 1px solid #ddd;
    border-radius: 8px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

    h2 {
      text-align: center;
      color: #007bff;
      margin-bottom: 10px;
      text-transform: uppercase;
      border-bottom: 2px solid #007bff;
      padding-bottom: 10px;
    }

    h3 {
      color: #333;
      margin-top: 20px;
      font-size: 18px;
      border-bottom: 1px solid #ccc;
      padding-bottom: 5px;
    }
```

```
    .criterio {
      margin-bottom: 15px;

      label {
        font-size: 14px;
        font-weight: bold;
        color: #555;
        display: block;
        margin-bottom: 5px;
      }

      .form-control {
        width: 100%;
        padding: 8px;
        font-size: 14px;
        border: 1px solid #ccc;
        border-radius: 4px;
        transition: border-color 0.3s;

        &:focus {
          border-color: #007bff;
          outline: none;
          box-shadow: 0 0 4px rgba(0, 123, 255, 0.4);
        }
      }
    }

    .btn {
      width: 100%;
      padding: 10px;
      font-size: 16px;
      font-weight: bold;
      background-color: #007bff;
      color: #fff;
      border: none;
      border-radius: 4px;
      cursor: pointer;
      transition: background-color 0.3s;

      &:hover {
        background-color: #0056b3;
      }

      &:disabled {
        background-color: #ccc;
        cursor: not-allowed;
      }
    }
  }
```

```
import { Component } from '@angular/core';
```

```
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';

@Component({
  selector: 'app-encuesta-glim',
  templateUrl: './encuesta-glim.component.html',
  styleUrls: ['./encuesta-glim.component.scss']
})
export class EncuestaGlimComponent {
  glimForm: FormGroup;
  showErrors: boolean = false;

  constructor(private fb: FormBuilder, private router: Router) {
    this.glimForm = this.fb.group({
      perdidaPeso: [null, Validators.required],
      imc: [null, Validators.required],
      masaMuscular: [null, Validators.required],
      ingestaAlimentos: [null, Validators.required],
      cargaInflamatoria: [null, Validators.required],
    });
  }

  calculateScore(): number {
    let resul = 0;
    // Obtén el puntaje sumando los valores seleccionados de los controles
    const perdidaPeso = this.glimForm.value.perdidaPeso;
    const imc = this.glimForm.value.imc;
    const masaMuscular = this.glimForm.value.masaMuscular;
    const ingestaAlimentos = this.glimForm.value.ingestaAlimentos;
    const cargaInflamatoria = this.glimForm.value.cargaInflamatoria;

    return perdidaPeso + imc + masaMuscular + ingestaAlimentos + cargaInflamatoria;
  }

  getObservation(score: number): string {
    // Devuelve una observación según el puntaje
    if (score >= 3) {
      return 'Desnutrición diagnosticada';
    } else {
      return 'No cumple con criterios de desnutrición';
    }
  }

  enviarResultados(): void {
    if (!this.glimForm.valid) {
      alert('Completa todas las preguntas antes de enviar los resultados.');
      this.showErrors = true;
      return;
    }

    const puntaje = this.calculateScore();
    const observacion = this.getObservation(puntaje);
```

```
    console.log('Puntaje: ', puntaje, '\nObservación: ', observacion);

    this.router.navigate(['home/resultado'], {
      queryParams: {
        nombreEncuesta: 'Criterios GLIM',
        puntaje: puntaje,
        observacion: observacion,
        porcentaje: ((puntaje / 2) * 100).toFixed(2) // Calcula el porcentaje basado
en un puntaje máximo (puedes ajustar el máximo si es diferente)
      }
    });
  }
}

<!-- Encuesta MUST -->
<form [formGroup]="MustForm" (ngSubmit)="enviarFormulario()">
  <h2 class="form-title">Encuesta MUST</h2>
  <div>
    <label for="imc">IMC (Kg/m²):</label>
    <input id="imc" formControlName="imc" type="number" />
  </div>

  <div>
    <label for="perdidaPeso">Pérdida de peso (%):</label>
    <input id="perdidaPeso" formControlName="perdidaPeso" type="number" />
  </div>

  <div>
    <label for="enfermedadAguda">¿Enfermedad aguda?</label>
    <input id="enfermedadAguda" formControlName="enfermedadAguda" type="checkbox" />
  </div>

  <button type="submit" [disabled]="MustForm.invalid">Enviar</button>
</form>

 form {
  max-width: 400px;
  margin: 20px auto;
  padding: 20px;
  background-color: #f9f9f9;
  border: 1px solid #ddd;
  border-radius: 8px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

  .form-title {
    text-align: center;
    font-size: 20px;
    font-weight: 700;
    margin-bottom: 20px;
    color: #007bff;
    text-transform: uppercase;
```

```
  border-bottom: 2px solid #007bff;
  padding-bottom: 10px;
}

div {
  margin-bottom: 15px;

  label {
    display: block;
    font-size: 14px;
    font-weight: 600;
    color: #333;
    margin-bottom: 5px;
  }

  input {
    width: 100%;
    padding: 10px;
    font-size: 14px;
    border: 1px solid #ccc;
    border-radius: 4px;
    transition: border-color 0.3s;

    &:focus {
      border-color: #007bff;
      outline: none;
      box-shadow: 0 0 4px rgba(0, 123, 255, 0.4);
    }

    &[type='checkbox'] {
      width: auto;
      margin-right: 10px;
    }
  }
}

button {
  width: 100%;
  padding: 10px;
  font-size: 16px;
  font-weight: 600;
  color: #fff;
  background-color: #007bff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  transition: background-color 0.3s;

  &:hover {
    background-color: #0056b3;
  }
```

```
    &:disabled {
      background-color: #ccc;
      cursor: not-allowed;
    }
  }
}


import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-encuesta-must',
  templateUrl: './encuesta-must.component.html',
  styleUrls: ['./encuesta-must.component.scss']
})
export class EncuestaMustComponent implements OnInit {
  MustForm: FormGroup = this.fb.group({ // Inicialización directa aquí
    imc: [null, Validators.required],
    perdidaPeso: [null, Validators.required],
    enfermedadAguda: [null, Validators.required],
    riesgoGlobal: [{ value: '', disabled: true }],
  });

  imcPuntuacion: number = 0;
  perdidaPesoPuntuacion: number = 0;
  enfermedadAgudaPuntuacion: number = 0;
  puntaje: number = 0;
  observacion: string = '';

  constructor(private fb: FormBuilder, private router: Router, private share:
SharedService) {}

  ngOnInit(): void {
    // Ya no es necesario inicializar mustForm aquí
    this.MustForm.get('imc')?.valueChanges.subscribe(() =>
this.calcularPuntuaciones());
    this.MustForm.get('perdidaPeso')?.valueChanges.subscribe(() =>
this.calcularPuntuaciones());
    this.MustForm.get('enfermedadAguda')?.valueChanges.subscribe(() =>
this.calcularPuntuaciones());
  }

  calcularPuntuaciones(): void {
    // Asignar puntuación para el IMC
    const imc = this.MustForm.get('imc')?.value;
    if (imc >= 20) {
      this.imcPuntuacion = 0;
    } else if (imc >= 18.5) {
      this.imcPuntuacion = 1;
```

```
    } else {
      this.imcPuntuacion = 2;
    }

    // Asignar puntuación para la pérdida de peso
    const perdidaPeso = this.MustForm.get('perdidaPeso')?.value;
    if (perdidaPeso <= 5) {
      this.perdidaPesoPuntuacion = 0;
    } else if (perdidaPeso <= 10) {
      this.perdidaPesoPuntuacion = 1;
    } else {
      this.perdidaPesoPuntuacion = 2;
    }

    // Asignar puntuación para la enfermedad aguda
    const enfermedadAguda = this.MustForm.get('enfermedadAguda')?.value;
    if (enfermedadAguda) {
      this.enfermedadAgudaPuntuacion = 2;
    } else {
      this.enfermedadAgudaPuntuacion = 0;
    }

    // Calcular la puntuación total
    this.puntaje = this.imcPuntuacion + this.perdidaPesoPuntuacion +
this.enfermedadAgudaPuntuacion;

    // Determinar el riesgo global
    if (this.puntaje <= 2) {
      this.observacion = 'Riesgo Bajo: Cuidados clínicos rutinarios';
    } else if (this.puntaje === 3) {
      this.observacion = 'Riesgo Medio: Observar se recomienda hospital y cuidados
domiciliarios';
    } else {
      this.observacion = 'Riesgo Alto: Tratar en domicilio, hospital o comunidad';
    }

    // Actualizar el valor de "riesgoGlobal" en el formulario
    this.MustForm.get('riesgoGlobal')?.setValue(this.observacion);
  }

  enviarFormulario(): void {
    if (this.MustForm.invalid) {
      alert('Por favor complete todos los campos.');
      return;
    }

    console.log('Formulario enviado:', this.MustForm.value);
    // Aquí podrías navegar a otra página o procesar el formulario
    const resultado = {
      point: this.puntaje,
      encuesta: 'MUST',
      observable: this.observacion,
```

```
      porcent: ((this.puntaje/6) * 100).toFixed(2)
    }
    this.share.saveResults(resultado);
    this.router.navigate(['home/resultado']);
  }
}

<!-- Encuesta Rabito -->
<div class="encuesta-rabito">
    <h1>Fórmulas de Rabito</h1>
    <form [formGroup]="rabitoForm" (ngSubmit)="calcularResultadosRabito()">
      <h2>Ingresar Medidas</h2>
      <div class="form-group">
        <!-- Entrada de datos -->
        <label for="sexoRabito">Sexo</label>
        <select id="sexoRabito" formControlName="sexo">
          <option value="1">Hombre</option>
          <option value="2">Mujer</option>
        </select>

        <label for="circBrazoRabito">Circunferencia del brazo (cm)</label>
        <input
          type="number"
          id="circBrazoRabito"
          formControlName="circBrazo"
          placeholder="Ej. 30"
        />

        <label for="circAbdominal">Circunferencia abdominal (cm)</label>
        <input
          type="number"
          id="circAbdominal"
          formControlName="circAbdominal"
          placeholder="Ej. 85"
        />

        <label for="circPantorrillaRabito">Circunferencia de pantorrilla (cm)</label>
        <input
          type="number"
          id="circPantorrillaRabito"
          formControlName="circPantorrilla"
          placeholder="Ej. 35"
        />

        <label for="mediaEnvergaduraRabito">Envergadura media del brazo (cm)</label>
        <input
          type="number"
          id="mediaEnvergaduraRabito"
          formControlName="mediaEnvergadura"
          placeholder="Ej. 50"
        />
```

```
        <label for="edadRabito">Edad (años)</label>
        <input
          type="number"
          id="edadRabito"
          formControlName="edad"
          placeholder="Ej. 45"
        />
      </div>

      <!-- Botón para calcular -->
      <button type="submit" [disabled]="!rabitoForm.valid">Calcular</button>
    </form>

    <div *ngIf="resultadoRabito" class="resultados">
      <h2>Resultados</h2>
      <p><strong>Peso estimado:</strong> {{ resultadoRabito.peso }} kg</p>
      <p><strong>Talla estimada:</strong> {{ resultadoRabito.talla }} cm</p>
    </div>
  </div>


.encuesta-rabito {
    max-width: 600px;
    margin: 0 auto;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 8px;
    background-color: #f9f9f9;

    h1 {
      text-align: center;
      color: #007bff;
    }

    .form-group {
      display: flex;
      flex-direction: column;
      gap: 15px;

      label {
        font-weight: bold;
      }

      input,
      select {
        padding: 8px;
        border: 1px solid #ccc;
        border-radius: 4px;
      }
    }

    button {
```

```
      margin-top: 20px;
      padding: 10px 15px;
      background-color: #007bff;
      color: #fff;
      border: none;
      border-radius: 4px;
      cursor: pointer;

      &:disabled {
        background-color: #ccc;
        cursor: not-allowed;
      }
    }

    .resultados {
      margin-top: 20px;
      padding: 15px;
      border: 1px solid #ccc;
      border-radius: 8px;
      background-color: #e9f7ef;

      p {
        font-size: 16px;
        font-weight: bold;
      }
    }
  }

import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
@Component({
  selector: 'app-encuesta-rabito',
  templateUrl: './encuesta-rabito.component.html',
  styleUrls: ['./encuesta-rabito.component.scss']
})
export class EncuestaRabitoComponent implements OnInit {
  rabitoForm: FormGroup; // Formulario reactivo
  resultadoRabito: any; // Resultados del cálculo

  constructor(private fb: FormBuilder) {
    // Configuración del formulario
    this.rabitoForm = this.fb.group({
      sexo: ['', Validators.required], // Sexo: Hombre (1) o Mujer (2)
      circBrazo: [null, Validators.required],
      circAbdominal: [null, Validators.required],
      circPantorrilla: [null, Validators.required],
      mediaEnvergadura: [null, Validators.required],
      edad: [null, Validators.required],
    });
  }

  ngOnInit(): void {}
```

```
  calcularResultadosRabito(): void {
    const formValues = this.rabitoForm.value;

    // Fórmula para el peso según Rabito
    const peso =
      0.5759 * formValues.circBrazo +
      0.5263 * formValues.circAbdominal +
      1.2452 * formValues.circPantorrilla -
      4.8689 * formValues.sexo -
      32.9241;

    // Fórmula para la talla según Rabito
    const talla =
      63.525 -
      3.237 * formValues.sexo -
      0.06904 * formValues.edad +
      1.293 * formValues.mediaEnvergadura;

    // Guardar los resultados
    this.resultadoRabito = {
      peso: peso.toFixed(2),
      talla: talla.toFixed(2),
    };
  }
}

<!-- Encuesta sarc-f -->
<div class="encuesta-sarcf">
    <h2>Encuesta SARC-F</h2>
    <form [formGroup]="sarcFForm">
      <table class="table table-bordered table-responsive">
        <thead>
          <tr>
            <th>Preguntas</th>
            <th>Puntaje</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td><strong>Strength (Fuerza):</strong> ¿Qué tanta dificultad tiene para
llevar o cargar 4.5 kg?</td>
            <td>
              <select formControlName="strength" class="form-control">
                <option value="0">Ninguna</option>
                <option value="1">Alguna</option>
                <option value="2">Mucha o incapaz</option>
              </select>
            </td>
          </tr>
          <tr>
```

```
            <td><strong>Assistance in walking (Asistencia para caminar):</strong>
¿Qué tanta dificultad tiene para cruzar caminando por un cuarto?</td>
            <td>
              <select formControlName="walking" class="form-control">
                <option value="0">Ninguna</option>
                <option value="1">Alguna</option>
                <option value="2">Mucha, usando auxiliares o incapaz</option>
              </select>
            </td>
          </tr>
          <tr>
            <td><strong>Rise from chair (Levantarse de una silla):</strong> ¿Qué
tanta dificultad tiene para levantarse de una silla o cama?</td>
            <td>
              <select formControlName="chair" class="form-control">
                <option value="0">Ninguna</option>
                <option value="1">Alguna</option>
                <option value="2">Mucha o incapaz sin ayuda</option>
              </select>
            </td>
          </tr>
          <tr>
            <td><strong>Climb stairs (Subir escaleras):</strong> ¿Qué tanta
dificultad tiene para subir 10 escalones?</td>
            <td>
              <select formControlName="stairs" class="form-control">
                <option value="0">Ninguna</option>
                <option value="1">Alguna</option>
                <option value="2">Mucha o incapaz</option>
              </select>
            </td>
          </tr>
          <tr>
            <td><strong>Falls (Caídas):</strong> ¿Cuántas veces se ha caído en el
último año?</td>
            <td>
              <select formControlName="falls" class="form-control">
                <option value="0">Ninguna</option>
                <option value="1">1-3 caídas</option>
                <option value="2">4 o más caídas</option>
              </select>
            </td>
          </tr>
        </tbody>
      </table>

      <button type="button" class="btn btn-success"
(click)="enviarResultados()">Enviar Resultados</button>
    </form>
  </div>

.encuesta-sarcf {
```

```
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;

    h2 {
      text-align: center;
      margin-bottom: 20px;
    }

    .table {
      width: 100%;
      border-collapse: collapse;

      th, td {
        padding: 10px;
        text-align: left;
        border: 1px solid #ddd;
      }

      th {
        background-color: #f4f4f4;
      }
    }

    .form-control {
      width: 100%;
      padding: 8px;
    }

    .resultado, .interpretacion {
      margin-top: 20px;
      font-size: 16px;
      text-align: center;
    }

    button {
      display: block;
      margin: 20px auto;
      padding: 10px 20px;
      font-size: 16px;
    }
  }
```

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { SharedService } from 'src/app/Service/shared.service';

@Component({
  selector: 'app-encuesta-sarc-f',
  templateUrl: './encuesta-sarc-f.component.html',
  styleUrls: ['./encuesta-sarc-f.component.scss']
```

```
})
export class EncuestaSarcFComponent implements OnInit{
  sarcFForm: FormGroup;
  puntuacionTotal: number = 0;

  constructor(private fb: FormBuilder, private router: Router, private shared:
SharedService) {
    this.sarcFForm = this.fb.group({
      strength: [0, Validators.required],
      walking: [0, Validators.required],
      chair: [0, Validators.required],
      stairs: [0, Validators.required],
      falls: [0, Validators.required],
    });
  }

  ngOnInit(): void {}

  calcularPuntuacion() {
    const values = this.sarcFForm.value;
    this.puntuacionTotal =
      +values.strength +
      +values.walking +
      +values.chair +
      +values.stairs +
      +values.falls;
  }

  enviarResultados() {
    if (this.sarcFForm.invalid) {
      alert('Completa todas las preguntas antes de enviar.');
      return;
    }

    const puntajeTotal = this.calcularTotal();
    let observacion;

    if (puntajeTotal >= 4) {
      observacion = 'Alta probabilidad de sarcopenia';
    } else {
      observacion = 'Baja probabilidad de sarcopenia';
    }

    console.log('Puntaje total:', puntajeTotal, 'Resultado:', observacion);

    const resultado = {
      point: puntajeTotal,
      encuesta: 'SARC-F',
      observable: observacion,
      porcent: ((puntajeTotal/10)*100).toFixed(2)
    }
```

```
    this.shared.saveResults(resultado);
    this.router.navigate(['home/resultado']);
  }

  private calcularTotal(): number {
    const values = this.sarcFForm.value;
    return (
      +values.strength +
      +values.walking +
      +values.chair +
      +values.stairs +
      +values.falls
    );
  }
}

<!-- Encuesta Volumen -->
<div class="encuesta-volumen-viscosidad">
    <h1>Prueba Volumen-Viscosidad</h1>
    <form [formGroup]="pruebaForm" (ngSubmit)="enviarResultados()">
      <!-- Tabla de opciones -->
      <table>
        <thead>
          <tr>
            <th>Viscosidad</th>
            <th>Bolo Volumen</th>
            <th>Deglución Segura</th>
            <th>Alteración Seguridad</th>
          </tr>
        </thead>
        <tbody>
          <!-- Viscosidad néctar -->
          <tr *ngFor="let volumen of volumenes">
            <td rowspan="3">Néctar</td>
            <td>{{ volumen }} mL</td>
            <td>
              <input type="radio" [formControlName]="'nectar_' + volumen"
value="segura" />
            </td>
            <td>
              <input type="radio" [formControlName]="'nectar_' + volumen"
value="alterada" />
            </td>
          </tr>
          <!-- Viscosidad líquida -->
          <tr *ngFor="let volumen of volumenes">
            <td rowspan="3">Líquida</td>
            <td>{{ volumen }} mL</td>
            <td>
              <input type="radio" [formControlName]="'liquida_' + volumen"
value="segura" />
            </td>
```

```
            <td>
              <input type="radio" [formControlName]="'liquida_' + volumen"
value="alterada" />
            </td>
          </tr>
          <!-- Viscosidad pudding -->
          <tr *ngFor="let volumen of volumenes">
            <td rowspan="3">Pudding</td>
            <td>{{ volumen }} mL</td>
            <td>
              <input type="radio" [formControlName]="'pudding_' + volumen"
value="segura" />
            </td>
            <td>
              <input type="radio" [formControlName]="'pudding_' + volumen"
value="alterada" />
            </td>
          </tr>
        </tbody>
      </table>

      <!-- Botón de envío -->
      <button type="submit" [disabled]="!pruebaForm.valid">Enviar Resultados</button>
    </form>
  </div>


.encuesta-volumen-viscosidad {
    text-align: center;
    margin: 0 auto;
    padding: 20px;

    h1 {
      color: #2a4d8c;
      font-size: 26px;
      margin-bottom: 20px;
    }

    table {
      width: 100%;
      border-collapse: collapse;
      margin: 0 auto;

      th, td {
        border: 1px solid #ddd;
        padding: 10px;
        text-align: center;
      }

      th {
        background-color: #f2f2f2;
        color: #333;
```

```
      }

      input[type='radio'] {
        transform: scale(1.2);
      }
    }

    button {
      margin-top: 20px;
      padding: 10px 20px;
      background-color: #2a4d8c;
      color: #fff;
      border: none;
      border-radius: 5px;
      cursor: pointer;
      transition: background 0.3s ease;

      &:hover {
        background-color: #1f3a6f;
      }
    }

    @media (max-width: 768px) {
      table, th, td {
        font-size: 12px;
        padding: 5px;
      }

      h1 {
        font-size: 20px;
      }
    }
  }
```

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup } from '@angular/forms';
import { Router } from '@angular/router';
@Component({
  selector: 'app-encuesta-volumen',
  templateUrl: './encuesta-volumen.component.html',
  styleUrls: ['./encuesta-volumen.component.scss']
})
export class EncuestaVolumenComponent implements OnInit {
  pruebaForm: FormGroup;
  volumenes: number[] = [5, 10, 20]; // Volúmenes definidos en el test

  constructor(private fb: FormBuilder, private router: Router) {
    this.pruebaForm = this.fb.group({});
  }

  ngOnInit(): void {
    this.crearControles();
```

```
  }

  // Genera dinámicamente los controles del formulario
  crearControles(): void {
    for (let viscosidad of ['nectar', 'liquida', 'pudding']) {
      for (let volumen of this.volumenes) {
        this.pruebaForm.addControl(`${viscosidad}_${volumen}`,
this.fb.control(null));
      }
    }
  }

  // Método de envío
  enviarResultados(): void {
    const resultados: any = {};

    for (let viscosidad of ['nectar', 'liquida', 'pudding']) {
      for (let volumen of this.volumenes) {
        resultados[`${viscosidad}_${volumen}`] = this.pruebaForm.get(
          `${viscosidad}_${volumen}`
        )?.value;
      }
    }

    console.log('Resultados:', resultados);

    // Envía resultados a otro componente usando router y queryParams
    this.router.navigate(['home/resultado'], {
      queryParams: {
        nombreEncuesta: 'Prueba Volumen-Viscosidad',
        resultados: JSON.stringify(resultados),
      },
    });
  }
}
```