

TECNOLOGICO NACIONAL DE MEXICO

Instituto Tecnológico de Culiacán



Titulacion Integral

Opción:

Tesis

Desarrollo de Interfaces de Usuario para una Aplicación de
Predicción de Contaminantes del Agua mediante Aprendizaje
Profundo

Derian Molina Lopez

20170753

INGENIERIA EN SISTEMAS COMPUTACIONALE



Instituto Tecnológico de Culiacán

DIVISION DE ESTUDIOS PROFESIONALES
ASUNTO: **Autorización de Impresión**

Culiacán, Sinaloa, 23/09/2025 19:23

C. DERIAN MOLINA LÓPEZ
PASANTE(S) DE LA CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES
P R E S E N T E.

De acuerdo con fallo emitido por la Comisión Revisora Integrada por:

Cédula Profesional

Presidente:	Doctorado en Ciencias en Ingeniería Eléctrica Opción en Sistemas Computacionales	
	Héctor Rodríguez Rangel	9972835
Secretario:	Maestría en Ciencias en Ciencias de la Computación	
	Gloria Ekaterine Peralta Peñuñuri	6038275
Vocal:	Maestría en Ciencias de la Computación	
	José Luis Medina Jiménez	13778869

Y considerando que reúne los requisitos establecidos en la Opción: **TITULACIÓN INTEGRAL: TESIS PROFESIONAL "DESARROLLO DE INTERFACES DE USUARIO PARA UNA APLICACIÓN DE PREDICCIÓN DE CONTAMINANTES DEL AGUA MEDIANTE APRENDIZAJE PROFUNDO"**, damos a usted nuestra autorización para que proceda a imprimir su trabajo profesional.

Sin otro particular, quedo de usted.

ATENTAMENTE

Excelencia en Educación Tecnológica®


LIC. CONCEPCIÓN MENDOZA ROSALES
JEFA DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES

 **Educación** |  **TECNOLÓGICO NACIONAL DE MÉXICO**
Secretaría de Educación Pública
INSTITUTO TECNOLÓGICO DE CULIACÁN
DEPARTAMENTO DE
DIVISIÓN DE ESTUDIOS PROFESIONALES



2025
Año de
La Mujer
Indígena

Juan de Dios Bátiz 310 Pte. Col. Guadalupe C.P. 80220
Culiacán, Sinaloa. Tel. 667-454-0100
tecnm.mx www.culiacan.tecnm.mx



Índice

Capítulo I: introducción	6
1.Introducción	7
1.1. Descripción del problema	7
1.2. Hipótesis.....	8
1.3. Objetivos.....	9
1.3.1. Objetivo general	9
1.3.2. Objetivos específicos.....	9
1.4. Justificación	9
1.5. Estructura de la tesis.....	10
Capitulo II: Marco teórico	12
2.1 Design Thinking.	13
2.2 Fases metodológicas del Design Thinking.	14
2.3. Psicología del color.	15
2.4. Diseño moderno como un requerimiento para la identidad de la tu marca.	16
2.5. Modelos de aprendizaje profundo y el impacto en la toma de decisiones.	16
2.6. Aprendizaje profundo.	17
2.6.1. Aprendizaje supervisado.	18
2.6.2. Aprendizaje no supervisado.....	20
2.6.3. Redes Neuronales Artificiales.....	20
2.6.3.1. Redes Neuronales Convolucionales.	21
2.6.3.2 Redes neuronales recurrentes (RNN).....	21
2.6.3.3 Redes transformer.	21
2.7. La Evolución del Diseño y Desarrollo Web en la Era Digital	22
2.8. DOM Scripting y el virtual DOM	23
2.9. La creciente tendencia del desarrollo web con manejo de tipos de datos	25
2.10. Entornos de desarrollo y como afecta el paradigma de programación.	26
2.11. Conversión de modelos de aprendizaje profundo.	27

Capítulo III: Estado del arte.....	28
3.1 Introducción	29
3.2. Antecedentes.	29
3.3. Tendencias actuales	31
3.3.1 Aplicación móvil para gestión de incidencias de residuos.	31
3.3.2. Aplicación móvil para el reconocimiento facial con redes convolucionales.	32
3.3.3. Experiencias en el desarrollo de interfaces para la interacción con modelos de Inteligencia	34
Capítulo IV: Desarrollo del proyecto.	35
4.1. Recopilación y análisis de requisitos.	37
4.1.1 Requisitos funcionales.	38
4.1.2. Requisitos no funcionales	40
4.2. Planificación.	40
4.3. Diseño	41
4.4. Desarrollo	42
4.5. Pruebas.....	42
4.6. Implementación.	43
Capítulo V: Pruebas y Resultados	44
5.1 Propuestas iniciales y evolución creativa del diseño.	45
5.1.1 Vista de presentación.	45
5.1.2 Propuesta de formularios.	47
5.1.3. Interfaz de aplicación móvil.....	49
5.2. Despliegue.	51
5.3. Manual de usuario (aplicación web)	52
4. Manual de usuario en aplicación Android.....	58
Capítulo VI: Conclusiones y trabajo futuro	64
6.1. Hallazgos	65
6.2. Limitaciones y mejoras a futuro.	65

Índice de figuras.

Figura 1. Arquitectura del proyecto	17
Figura 2. Árbol de componentes.	24
Figura 3. Niveles de jerarquía de componentes.	24
Figura 5. Imagen de interfaz de Gradio	31
Figura 6. Arquitectura de software para aplicación móvil de gestión de residuos.	32
Figura 7. Interfaz de usuario planteada para el reconocimiento facial	33
Figura 8. Cronograma de actividades.	36
Figura 9. Interacción usuario a aplicación.	41
Figura 10. Etapas homologas al prototipado.	42
Figura 11. Primer borrador de presentación.	45
Figura 12. Segunda propuesta de presentación.	46
Figura 13. Propuesta final de vista de presentación.	47
Figura 14. Versión 1 de formulario.	47
Figura 15. Versión 2 de formulario.	48
Figura 16. Formulario final.	49
Figura 17. Interfaz gráfica de aplicativo móvil.	50
Figura 18. Formulario de Android.	50
Figura 19. Panel de pruebas de despliegue.	51
Figura 20. Panel de administración de un despliegue en específico.	52
Figura 21. Página principal del aplicativo.	53
Figura 22. Ventana modal con el formulario.	53
Figura 23. Ventana modal para la selección de nutrientes y entradas	54
Figura 24. estimación de nutriente.	55
Figura 25. Resultado de estimación masiva.	56
Figura 26. Validación de campos vacíos.	57
Figura 27. validación de rangos.	57
Figura 28. Página de presentación de aplicación Android.	58
Figura 29. Ventana modal para aplicación Android.	59

Figura 30. Selección estricta de los 3 nutrientes en aplicación Android.	60
Figura 31. Estimación masiva en aplicación Android.	61
Figura 32. Estimación simple en aplicación Android.	62
Figura 33. Validación de los campos vacíos.....	63
Figura 34. Validación numérica.	63

Capítulo I: introducción

1.Introducción

Abordar el desarrollo de una aplicación web basada en modelos de aprendizaje profundo, diseñada para que los empleados de la Comisión Nacional del Agua (CONAGUA) realicen predicciones de contaminantes clave para evaluar la calidad del agua, sin requerir conocimientos avanzados en tecnología. Se busca identificar la importancia de optimizar y automatizar los procesos de predicción, permitiendo que usuarios sin experiencia técnica obtengan resultados precisos a partir de datos simples, mejorando la accesibilidad y reduciendo los costos y tiempos asociados a los análisis tradicionales de laboratorio. El proyecto se estructurará en tres fases: en la primera, se implementarán modelos de aprendizaje profundo clasificados según el tipo, número de entradas y variable de estimación; en la segunda, se desarrollará la lógica de procesamiento utilizando Python para analizar y organizar los datos, seleccionando el modelo adecuado según el tipo de entrada y nutriente; y en la tercera, se diseñará una interfaz de usuario que integre y presente los resultados de manera clara y accesible. Se adoptará la metodología ágil Scrum para permitir flexibilidad en la gestión de tareas mediante sprints adaptados al cronograma preliminar. Asimismo, se realizarán reuniones regulares con asesores y miembros del equipo de trabajo para abordar problemas y proponer soluciones. Se deberán considerar limitaciones logísticas y geográficas, dado que la implementación se lleva a cabo en el Instituto Tecnológico de Culiacán, Sinaloa, contemplando la opción de trabajo remoto. La aplicación estará destinada exclusivamente a empleados de CONAGUA y se limitará a generar estimaciones basadas en los datos proporcionados, sin sustituir los análisis exhaustivos realizados en laboratorio.

1.1. Descripción del problema

La Comisión Nacional del Agua (CONAGUA) en México enfrenta desafíos críticos en la evaluación de la calidad del agua, particularmente en la medición de nutrientes como nitrógeno total, fósforo total, ortofosfato, nitrógeno amoniacal y nitrógeno total Kjeldahl. Actualmente, estos análisis dependen de métodos tradicionales de laboratorio, que implican costos elevados, tiempos prolongados (semanas) y logística compleja (transporte de muestras, personal especializado y equipos frágiles). Esta ineficiencia limita la capacidad de respuesta ante emergencias ambientales y obstaculiza la toma de decisiones informadas para la gestión sostenible de recursos hídricos.

En caso de haber una emergencia ambiental que se haya detectado semanas después de la toma de muestra y su respectivo análisis, eso causa que la toma de decisiones sea muy tardada o que haya alguna irregularidad que no debería haber, incluso cuando ya se tiene un resultado sobre la irregularidad detectada, la gravedad del asunto habrá cambiado de tal manera que, si se hace un nuevo estudio, se detectarían resultados diferentes para el cuerpo de agua donde se realizó el estudio. Este panorama refleja diversas problemáticas que se agrupan en tres dimensiones principales: la inaccesibilidad tecnológica, las limitaciones operativas y las barreras técnicas en la implementación de soluciones.

En cuanto a la inaccesibilidad tecnológica, los modelos de aprendizaje profundo existentes para la predicción de la calidad del agua no se encuentran integrados en herramientas fáciles de utilizar por personal sin experiencia técnica. Esto significa que, para los trabajadores de CONAGUA, resulta poco amigable el uso del equipamiento disponible, ya que este suele estar diseñado para especialistas haciendo que sea necesario la capacitación del personal, cuando se puede sustituir por un proceso automatizado. Además, la dependencia de equipos portátiles costosos y de laboratorios especializados complica aún más el panorama, particularmente en regiones rurales o de difícil acceso, donde este tipo de infraestructura no está disponible, lo que pone en riesgo la integridad del equipamiento por algún incidente en el campo de estudio y retrasando aun mas la entrega de los resultados en lo que se solicita un nuevo equipamiento.

Por otro lado, las limitaciones operativas muestran que los métodos tradicionales no permiten escalar el monitoreo de manera continua ni realizar análisis históricos de forma eficiente. Actualmente no existe una herramienta unificada que concentre la capacidad de realizar estimaciones en tiempo real y, al mismo tiempo, almacenar datos que puedan aprovecharse para elaborar reportes periódicos. Esta ausencia de integración reduce la capacidad de respuesta y de planeación estratégica.

Como último punto, en el ámbito técnico, el principal reto radica en cómo llevar a la práctica los modelos predictivos ya entrenados. Si bien el proyecto cuenta con modelos previamente optimizados para el análisis de nutrientes en el agua, se requiere implementar una estructura capaz de normalizar de manera dinámica los datos de entrada, asegurando que cada combinación de parámetros sea dirigida al modelo correspondiente. La ausencia de una aplicación accesible que integre estos modelos representa una limitación importante, ya que actualmente no existe una plataforma, ni en entorno web ni en dispositivos Android, que permita a los usuarios interactuar con los sistemas de predicción de manera sencilla y sin conocimientos técnicos especializados. Además, es indispensable que esta aplicación cuente con una interfaz intuitiva y responsiva, capaz de gestionar diferentes estados y opciones sin comprometer la usabilidad.

1.2. Hipótesis

El desarrollo de una aplicación web y una aplicación móvil que integran modelos de aprendizaje profundo facilitará la accesibilidad tecnológica para el personal de la Comisión Nacional del Agua (CONAGUA), permitiendo obtener resultados de estimación de nutrientes en el agua de manera rápida y confiable. De esta forma, en caso de detectarse una anomalía en un cuerpo de agua, será posible aplicar oportunamente los protocolos necesarios para su atención, optimizando la toma de decisiones y reduciendo la dependencia de métodos tradicionales de laboratorio. En los estudios de campo es donde se notará el cambio, una aplicación Android que tenga

integración de los modelos, les planteara el panorama a los investigadores sobre la gravedad del asunto antes de que las muestras sean procesadas por el laboratorio

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar dos aplicaciones, una web y otra para Android, que permitan a los empleados de la Comisión Nacional del Agua (CONAGUA) predecir futuros niveles de contaminantes mediante modelos de aprendizaje profundo, facilitando así una toma de decisiones informada y basada en datos. Ofrecer interfaces de usuario altamente intuitivas, visuales y accesibles, optimizadas para el análisis de datos y la estimación de niveles de químicos específicos. Diseñar un frontend adaptable y responsivo que proporcione una experiencia de usuario óptima en cualquier dispositivo, promoviendo su uso práctico y versátil en distintas situaciones y contextos laborales.

1.3.2. Objetivos específicos

1. Especificación de requisitos funcionales y no funcionales del proyecto.
2. Diseño de vistas y formularios adaptados a los requisitos del usuario.
3. Diseño consistente de las aplicaciones en diferentes navegadores o dispositivos.
4. Configurar y optimizar el flujo de datos de entrada con base a eventos de React para permitir cambios fáciles sin reestructurar el proyecto.
5. Implementación de los diseños en React.
6. Implementar un módulo que permita una serie de peticiones http para poder procesar los parámetros que vienen de entrada.

1.4. Justificación

El presente proyecto tiene una gran relevancia al abordar una de las problemáticas más críticas en el ámbito ambiental: la evaluación de la calidad del agua y la detección oportuna de anomalías en sus parámetros. Actualmente, los métodos tradicionales de análisis se basan en la recolección de muestras y su envío a laboratorios especializados, lo cual implica tiempos prolongados, costos elevados y una limitada capacidad de respuesta ante emergencias ambientales. Esta situación genera un retraso en la toma de decisiones y afecta la capacidad de instituciones como la Comisión Nacional del Agua (CONAGUA) para actuar de manera preventiva y eficiente en la gestión de los recursos hídricos.

Los principales desafíos identificados se relacionan con la inaccesibilidad tecnológica y la dependencia de infraestructura costosa, que dificultan la implementación de sistemas predictivos accesibles para usuarios no especializados, especialmente en regiones rurales o de difícil acceso. Asimismo, la falta de herramientas unificadas para realizar estimaciones en tiempo real y almacenar información para análisis históricos limita el aprovechamiento de los datos y reduce la eficiencia en los procesos de monitoreo y control.

Frente a este escenario, el desarrollo de una aplicación web y una aplicación móvil específica para Android, con modelos de aprendizaje profundo integrados, representa una solución innovadora que aporta beneficios significativos. Estas herramientas facilitan el acceso a predicciones confiables de parámetros clave como nitrógeno y fósforo, reducen los tiempos de respuesta y los costos asociados a los métodos tradicionales, y permiten un monitoreo remoto y continuo de la calidad del agua. Además, al contar con interfaces intuitivas y accesibles, favorecen la inclusión tecnológica y fortalecen la capacidad operativa de los empleados de CONAGUA, contribuyendo de manera directa a la sostenibilidad ambiental y a la mejora en la gestión de los recursos hídricos.

1.5. Estructura de la tesis

La tesis presentada consta de 6 capítulos de los cuales se segmenta el contexto del trabajo en cada uno de ellos, organizados de una forma secuencial como lo siguientes puntos:

Capítulo I introducción

En este capítulo recalca el contexto corporativo de la entidad de CONAGUA y el por qué es requerido este tipo de proyectos, además de mencionar que es lo que se espera del proyecto.

Capitulo II: Marco teórico.

En el apartado mencionado, se revisa el contenido contextual y filosófico detrás de la empatía de los usuarios llevando a cabo las fases metodológicas del Design thinking según diferentes autores quienes aplicaron la metodología en sus proyectos de diseño y otros de trabajo de desarrollo de software, además de un apartado sobre la psicología del color, haciendo énfasis sobre como el color mismo es una forma de entender la realidad de todo aquel objeto que nos rodea en el día a día.

Capitulo III: Estado del arte

Junto a una investigación sobre la literatura contemporánea respecto a las tendencias actuales que afectan al mundo de la tecnología, se ha dividido en 7 secciones, las cuales una contrasta con la otra, empezando desde modelos de aprendizaje profundo hasta incluir el manejo de

los entornos de desarrollo y la conversión de modelos usando herramientas de código abierto soportado por la comunidad tecnológica.

Capitulo IV: Desarrollo del proyecto

Al tratarse de un trabajo de ingeniería de software, hay que mostrar cómo es que el equipo trabajo de forma iterativa por medio de metodologías ágiles ante los cambios que llegan a presentar durante el desarrollo del proyecto, siendo así iniciando desde el cronograma de las actividades con sus respectivas tareas con x días que llevaría completar la tarea; Presentar los requisitos del cliente, tanto funcionales como no funcionales además de presentar una sección de especificación de requisitos.

Capitulo V: Resultados

Después del desarrollo de la aplicación, el equipo documento el proceso de las pruebas del programa y solución de problemas que suele haber en el despliegue de las aplicaciones, los resultados de ambas aplicaciones se mostrarían en forma de un manual, teniendo cada aplicación una sección respectiva para el manual de usuario y mostrar imágenes de cómo es que fluye las aplicaciones.

Capitulo VI: Conclusiones y trabajo futuro.

En el ultimo capitulo se explora de los puntos encontrados durante la elaboración del proyecto, identificando mejoras de interés para el mismo proyecto o un enfoque diferente incluso las limitaciones del proyecto por dependencias internas o externas.

Capítulo II: Marco teórico

El desarrollo de aplicaciones frontend en el contexto actual enfrenta un escenario técnico polarizado: mientras entornos basados en estados y variables (como React con su modelo de Hooks) redefinen la interacción con eventos de pantalla mediante flujos reactivos, otros sistemas tradicionales (como Angular o Svelte) mantienen un enfoque en la programación orientada a objetos (POO), priorizando jerarquías de clases y flujos predecibles. Esta dicotomía generó desafíos significativos en el cronograma del proyecto, no solo por las diferencias paradigmáticas, sino también por la necesidad de alinear ambos enfoques con un marco de trabajo de desarrollo de software estricto (ej: metodologías ágiles) y principios complementarios como el Design Thinking (empatía, iteración) y fundamentos de diseño (psicología del color, identidad de marca).

2.1 Design Thinking.

El marco de trabajo del Design Thinking se orienta específicamente a analizar la relación que establecen los clientes cuando interactúan con un producto o servicio, particularmente durante su primer contacto o experiencia de uso. Sin embargo, para lograr una comprensión genuina de esta dinámica, es fundamental abordar el problema desde múltiples ángulos que permitan captar la complejidad del comportamiento del público objetivo.

Como menciona Carlos Chimbo (2024) en su trabajo *"Aplicación web para la gestión de inventarios de la empresa Servicompu utilizando la metodología Design Thinking"*, este enfoque metodológico busca examinar diversas perspectivas para alcanzar una comprensión profunda no solo de las necesidades evidentes, sino también de los deseos implícitos y las respuestas emocionales de las personas afectadas por la problemática en estudio.

Es crucial entender que el desarrollo de experiencias satisfactorias trasciende ampliamente el ámbito de los programas informáticos y su funcionalidad técnica. La percepción holística del entorno juega un papel determinante. Para ilustrar este punto, consideremos el ejemplo de un cliente que visita un restaurante: desde el primer momento, su experiencia se configura a través de múltiples elementos como las características físicas de las instalaciones, la calidad y temperatura de la iluminación, los sonidos ambientales e incluso los olores característicos del lugar. Cuando una persona entra o sale de un establecimiento, recibe un verdadero bombardeo de información sensorial que puede resultar en una experiencia agradable y memorable, o, por el contrario, en una vivencia desagradable que disuade futuras visitas.

Este mismo principio se aplica al diseño de sistemas digitales. Cuando un usuario abre una ventana en su navegador y accede a una página web específica, la percepción inmediata que se forme - influenciada por factores como la disposición de los elementos, la paleta de colores, la tipografía o incluso los tiempos de carga - puede determinar decisivamente si continúa navegando en el sitio o lo abandona prematuramente.

2.2 Fases metodológicas del Design Thinking.

El Design Thinking propone un proceso estructurado en fases específicas para abordar sistemáticamente los problemas de diseño. El trabajo planteado por Bragean Luis Vargas Márquez y Luis Ángel Inga Hanampa (2021) describe etapas del marco de trabajo, de las cuales las ideas son las siguientes:

Empatizar: Es la etapa inicial que tiene dos enfoques clave, siendo la observación y la involucración, el objetivo es entender el día a día del público objetivo, la observación debe de ser de tal manera que no te interfieras en el orden “natural” de las cosas, de tal manera que puedes visualizar el cómo se dan los problemas y el sentimiento de los usuarios antes de que pase y después del suceso; En la involucración el analista, prepara una serie de preguntas clave que ayudara a profundizar en preguntas que puedan surgir en la observación, siendo el ¿Cómo?, el ¿Por qué?, pero siendo preguntas que no sean invasivas para evitar afectar al público objetivo, ya que una modificación en las variables del entorno, puede cambiar completamente el resultado.

Definir: Tras empatizar con el público objetivo es posible segmentar, las partes del proceso actual o las partes que conforman un todo de la situación de la que hay que resolver, así que, para hacerlo, se crean una serie de palabras clave, para organizar la información, tal cual la etapa se llama “definir” ya que trae claridad al proyecto estableciendo una comprensión compartida, de tal manera, permite tener una visualización clara para el equipo de trabajo o nuevos equipos

Idear: Esta etapa se centra en el proceso creativo de generación de conceptos y soluciones potenciales. El equipo de diseño realiza sesiones de lluvia de ideas, utilizando como insumos todos los datos y comprensiones obtenidas en las fases anteriores. Es fundamental destacar que, durante este proceso, ninguna idea debe descartarse prematuramente, ya que incluso las propuestas aparentemente menos convencionales pueden contener elementos valiosos para la solución final. De hecho, Manuel Serrano Ortega y Pilar Blázquez Ceballos en su libro "Design Thinking: Lidera el presente. Crea el futuro" (2016), recomiendan que, al finalizar dicha fase, deben de tener una referencia visual, por medio de fotos, diagramas o notas móviles para aclarar cada uno de los conceptos de la lluvia de ideas.,

Prototipar: Esta etapa práctica se enfoca en la materialización de las ideas seleccionadas, el equipo desarrolla elementos visuales o tangibles que representen las soluciones propuestas, con el objetivo principal de probar su viabilidad y efectividad. Estos prototipos pueden variar desde simples bosquejos en papel hasta maquetas interactivas más elaboradas, dependiendo de la naturaleza del proyecto y los recursos disponibles; Sin embargo el “Objeto” de prototipo puede variar un como el nombre lo dice “prototipar” es para probar una posibilidad de solución y que el usuario va a usar un prototipo puede ser cualquiera cosa sobre en la que el usuario pueda interactuar (Espinoza Vásquez & Espinoza Zapata, 2025), puede ser un

diagrama de proceso, una actividad, de igual manera que puede ser una interfaz gráfica, hay muchas posibilidades depende mucho del problema.

Testear: Una vez implementada la solución propuesta en forma de prototipo, es esencial obtener retroalimentación directa de los usuarios potenciales. Esta fase de prueba permite identificar fallas, puntos de confusión o riesgos potenciales en el diseño propuesto. La información recogida durante las pruebas sirve como base para realizar ajustes y refinamientos iterativos, con el objetivo final de mitigar o eliminar completamente los problemas detectados antes de llegar a la versión definitiva del producto o servicio.

2.3. Psicología del color.

El color, en su definición literal, es "la impresión producida al incidir en la retina rayos difundidos o reflejados por un cuerpo" (Mora, 2005). Sin embargo, tras esta explicación fisiológica yace una dimensión filosófica profunda: el color no es solo un fenómeno óptico, sino una construcción perceptual que media nuestra comprensión de la realidad.

Todo lo que vemos es, en esencia, luz interpretada; los objetos no poseen color intrínseco, sino que revelan aquella porción del espectro lumínico que rechazan. Así, el color se erige como un lenguaje sensorial que traduce la materia en experiencia, cuestionando incluso los límites entre lo objetivo (la longitud de onda reflejada) y lo subjetivo (la interpretación cerebral y cultural). Esta dualidad entre física y percepción convierte al color en un puente único entre el mundo externo y la conciencia humana, donde lo visible es siempre un diálogo entre la luz que nos llega y el significado que le otorgamos. El color en sí mismo no posee un significado inherente, son los seres humanos quienes lo dotan de sentido a partir de las reacciones sensoriales y emocionales que provoca. Esta influencia no se limita a respuestas visibles externamente, sino al impacto subjetivo que ejerce en la percepción de una experiencia diseñada.

En el marketing contemporáneo, un buen diseño visual ya no es suficiente las marcas más exitosas buscan activamente estimular los cinco sentidos para idealizar su producto, como señala Claudia Ruiz en su estudio *"Psicología del color; su importancia e influencia en la decisión de compra del consumidor: un enfoque desde el sector de alimentos y bebidas en Colombia"*. Ruiz destaca que el sentido de la vista es el más explorado por las empresas, representando un 58% de influencia en la modificación de la decisión de compra el dato mencionado refuerza la idea de que, en productos de software donde la experiencia del usuario es clave, el uso estratégico del color y otros estímulos visuales puede transformar la percepción funcional en una experiencia agradable y memorable.

2.4. Diseño moderno como un requerimiento para la identidad de la tu marca.

Anteriormente se habló sobre un marco de trabajo para entender a los usuarios, a tu público objetivo el cual ayudara tu aplicación o producto, pero al momento de aplicarlo, hay que atrapar a esos usuarios, al final del día hay que convencer a los usuarios de que tu producto es mejor, y un paso para hacerlo es por medio del diseño emocional, si tu usuario no puede completar la tarea para la que está hecho tu producto no pasara mucho tiempo en tu aplicación, para poder “atrapar” a tu usuario es necesario que tu aplicación sea placentera, usable, confiable y sobre todo funcional (Ignacio,2014), de lo contrario con la mínima experiencia negativa que tenga el usuario, bastara para que opte por usar otra herramienta de la que estas enfrentándote en tu competencia.

En la síntesis creativa de Jiménez, A. C. (2011) Steve y sus aportaciones al mundo del diseño habla sobre la vida de Steve Jobs, y sobre la importancia tipográfica que le da a los diseños de sus productos, creando una marca muy conocida ara hoy den días, de hecho hoy en día al ver la imagen de una manzana mordida, es muy fácil distinguir de que marca es la que se está hablando, eso se le llama posicionamiento de marca, un simple logo representa una increíble variedad de productos que conforman una compañía, por otro lado el consumidor quien es a quienes apunta la elaboración del diseño, desde pequeños cuando crecen, viendo diversos productos consumibles adoptan hábitos de consumo, bajo una “percepción de marca” la cual se dimensiona en gran medida por medio de investigación, marketing estratégico y economía del sector donde has definido tu público (Vinicio Mejia-Vayas & Cherres-Galarza,2020) la percepción de marca es la respuesta de un usuario ante una experiencia con un producto.

Debido a que la tesis presente se trata sobre dos aplicaciones, pero con un enfoque en el desarrollo frontend, se hará una revisión sobre la literatura de las cuales se hablara sobre cómo llevar a cabo un diseño de software agradable al usuario, tomando en cuenta el enfoque de trabajo del Design Thinking, la planificación de la “persona usuaria”, y el desarrollo frontend moderno como un requerimiento para la marca de la empresa, así que para ello fue necesario realizar una investigación sobre trabajos que sean de utilidad para respaldar los argumentos planteados.

2.5. Modelos de aprendizaje profundo y el impacto en la toma de decisiones.

Los modelos de estimación ha sido un gran auge en los últimos años con la salida de nuevas técnicas que permiten aumentar la precisión de los modelos, habiendo modelos de estimación para recomendaciones de películas, para intentar predecir el clima siendo el caso de este proyecto, modelos de dos tipos, escaladores y estimadores, uno que básicamente se encarga de transformar los datos de entrada para mejorar la precisión del modelo escalador, recibiendo x número de entrada de parámetros en orden específico, para luego ser procesado por el modelo de estimación y regresar una estimación del valor real.

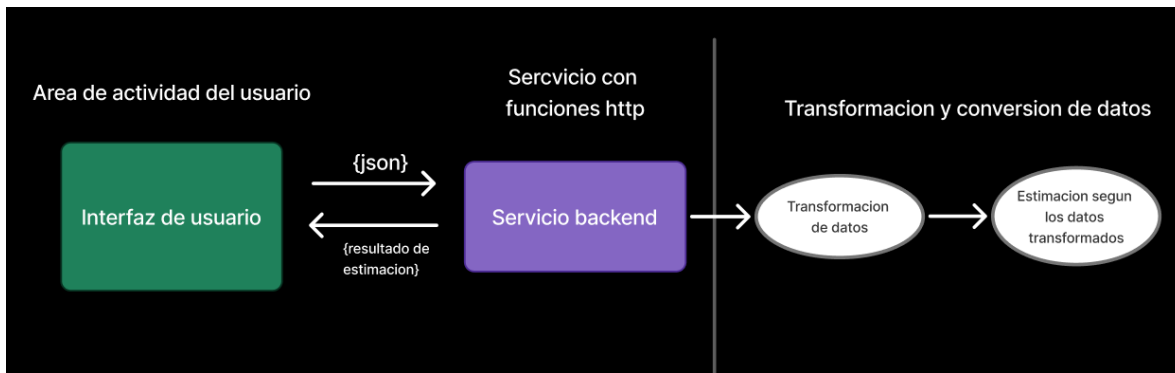


Figura 1. Arquitectura del proyecto

El aprendizaje automático en la época actual existe un mundo de plantillas para hacer tus modelos, incluso hay modelos preprogramados para cambiarle la naturaleza de los parámetros, sin embargo, la creación de modelos de aprendizaje profundo por si sola es compleja, ahí es donde entran los conceptos de CNN (Convolutional Neuronal Network), y las RNN (Recurrent Neuronal Network) siendo los principios para la creación y modelado de agentes inteligentes.

Los modelos de estimación tienen un amplio espectro de aplicaciones, cuyo impacto trasciende diversos sectores económicos. La investigación en inteligencia artificial incorpora criterios vinculados a las nociones estadísticas y económicas de racionalidad (Casadiego, 2020). Esto nos lleva a cuestionar: ¿qué relación existe entre la IA y los modelos? Fundamentalmente, la IA constituye una rama de la ciencia informática enfocada en desarrollar máquinas capaces de emular procesos cognitivos humanos (Bongianino, 2019). Por su parte, los modelos de aprendizaje profundo representan sistemas computacionales con capacidad de aprendizaje autónomo, sin requerir programación explícita para cada tarea específica (Díaz Ramírez, 2021). Esta sinergia entre IA y modelado permite crear sistemas adaptativos que optimizan continuamente su desempeño mediante el procesamiento de datos y la identificación de patrones complejos.

2.6. Aprendizaje profundo.

Los modelos de aprendizaje constituyeron la base del desarrollo de la inteligencia artificial moderna, al permitir que las máquinas pudieran adquirir conocimiento a partir de datos y mejorar su desempeño con la experiencia. Estos modelos, inspirados en procesos cognitivos humanos, fueron diseñados con el propósito de identificar patrones, extraer relaciones y generar predicciones que facilitaran la resolución de problemas en diversos contextos. Dentro de este campo, sobresalieron tres ejes temáticos fundamentales: el aprendizaje supervisado, el aprendizaje no supervisado y las redes neuronales artificiales, cada uno con características, técnicas y aplicaciones particulares.

La raíz de los modelos de aprendizaje profundo y, en general, de gran parte de la inteligencia artificial moderna, se encuentra en el perceptrón. Este modelo fue propuesto por Frank

Rosenblatt en 1958 como un algoritmo de aprendizaje supervisado capaz de realizar tareas de clasificación binaria a partir de datos etiquetados. Su importancia histórica radica en la introducción del concepto de neuronas artificiales, inspiradas en el funcionamiento de las neuronas biológicas, donde múltiples entradas se ponderan mediante pesos, se combinan y, finalmente, se procesan a través de una función de activación que determina la salida final.

En la obra *Introducción a los modelos de redes neuronales artificiales: el perceptrón simple y multicapa*, Tomás Alcalá describe cómo este modelo evolucionó en la década de 1960 gracias a los aportes del profesor Bernard Widrow, quien junto a Ted Hoff desarrolló el algoritmo Adaline (Adaptive Linear Element). A diferencia del perceptrón de Rosenblatt, Adaline permitía resolver problemas de clasificación utilizando funciones lógicas como AND y OR, aunque todavía mantenía la limitación de no poder resolver problemas no lineales como el XOR, lo que posteriormente impulsó el desarrollo de perceptrones multicapa.

El perceptrón simple constituye, por tanto, el punto de partida conceptual de las redes neuronales artificiales modernas, sentando las bases para arquitecturas más complejas con capas de entrada, ocultas y de salida. Actualmente, este modelo ha trascendido su forma original y se encuentra implementado en múltiples librerías y frameworks que facilitan su uso sin necesidad de profundizar en la complejidad matemática subyacente. Ejemplos destacados son TensorFlow y PyTorch, que permiten diseñar y entrenar redes neuronales de manera accesible. Incluso en entornos de despliegue modernos, como Docker Hub, es posible encontrar imágenes preconfiguradas con estos frameworks, lo que simplifica la instalación y acelera el desarrollo, haciendo que el campo del aprendizaje profundo sea más accesible y con una mejor experiencia de implementación.

El aprendizaje supervisado se centró en el uso de datos previamente etiquetados, permitiendo que los modelos aprendieran a clasificar o predecir resultados conocidos a partir de ejemplos. Por su parte, el aprendizaje no supervisado representó la contraparte metodológica, ya que operó con información sin etiquetas y buscó descubrir patrones ocultos, agrupaciones y representaciones latentes de los datos. Finalmente, las redes neuronales artificiales, núcleo del aprendizaje profundo, se consolidaron como arquitecturas computacionales capaces de simular, en cierta medida, la forma en que los humanos procesan la información, adaptándose mediante parámetros ajustables para enfrentar problemas complejos en áreas como la visión por computadora, el procesamiento de lenguaje natural o la predicción de fenómenos.

2.6.1. Aprendizaje supervisado.

En el caso del aprendizaje supervisado, este se fundamenta en la clasificación de patrones de datos con el fin de segmentar y diferenciar unos de otros. Un ejemplo ilustrativo puede observarse en la manera en que los seres humanos distinguimos entre distintos animales. Desde edades tempranas, las personas reciben información sensorial del entorno y aprenden a diferenciar objetos mediante características como la forma, el color o el sonido. De forma análoga, los modelos de aprendizaje profundo requieren que los datos sean etiquetados para

que, al ser procesados, el sistema pueda identificar patrones y realizar inferencias sobre la información de entrada. Un caso aplicado es el trabajo de Fernández de Gorostiza Luengo, titulado “*Detección de ratones de laboratorio mediante técnicas de visión artificial clásica y aprendizaje profundo*”, en el que se calibró un sistema de percepción capaz de distinguir imágenes a través del uso de modelos entrenados con datos previamente clasificados.

Por otra parte, el aprendizaje profundo no se limita al campo de la visión artificial, sino que se aplica también en otros ámbitos de gran relevancia. Entre estos, destacan los modelos de estimación, cuyo propósito es predecir comportamientos futuros a partir de los patrones previamente aprendidos. Tal como afirman Carmona et al. (s.f.), este tipo de modelos predictivos se sustentan en el aprendizaje supervisado y abarcan tareas como la clasificación, la regresión o el análisis de series temporales, todas ellas orientadas a generar resultados a partir de datos previamente etiquetados.

El Procesamiento de Lenguaje Natural (PLN) constituyó un área clave dentro de la inteligencia artificial, orientada a dotar a las máquinas de la capacidad de comprender y generar lenguaje humano más allá del simple reconocimiento de caracteres o sonidos. Antes de la llegada de Internet, la lingüística computacional se fundamentaba en reglas y diccionarios elaborados manualmente por especialistas, lo que limitaba su alcance y hacía que los avances fueran lentos y fragmentados. Sin embargo, con la globalización y la masificación de Internet, la disponibilidad de grandes volúmenes de datos textuales se volvió más accesible, lo que permitió que el PLN transitara hacia un enfoque sustentado en el aprendizaje automático y en el análisis estadístico de la información. Este cambio metodológico facilitó el desarrollo de aplicaciones como la digitalización de documentos, los sistemas de búsqueda semántica, los resúmenes automáticos y los sistemas de respuesta a preguntas. Dicho progreso posibilitó que las máquinas procesaran información de manera más cercana a la comprensión humana, ampliando sus campos de aplicación y consolidándose como una herramienta esencial en la gestión del conocimiento (Gelbukh, A. 2010).

A la fecha, incluso existen modelos de inteligencia artificial capaces de emplear la técnica de *web scraping* para analizar no solamente el texto de la web, sino también la estructura semántica que lo constituye como sitio digital. Con esta técnica es posible llevar a cabo estudios avanzados, como el de Gómez Meza (2021), titulado *Análisis de sentimientos en noticias financieras y económicas hondureñas usando web scraping e inteligencia artificial*. En dicho trabajo, la técnica de *web scraping* permitió transformar los datos provenientes de portales de noticias en información estructurada y aprovechable, lo que facilitó el entrenamiento de un modelo de aprendizaje profundo capaz de generar estimaciones confiables.

2.6.2. Aprendizaje no supervisado.

El aprendizaje no supervisado representó una estrategia clave en el campo del aprendizaje automático, ya que permitió a los modelos identificar relaciones y estructuras ocultas en los datos sin necesidad de etiquetas previas. Este enfoque se caracterizó por su capacidad de descubrir agrupamientos naturales, asociaciones o representaciones simplificadas en grandes volúmenes de información, lo cual resultó de gran utilidad en escenarios donde la clasificación manual resultaba impráctica o costosa. A diferencia de los modelos supervisados, que dependían de ejemplos previamente categorizados, los algoritmos no supervisados trabajaron de manera autónoma para organizar los datos y revelar regularidades que permanecían ocultas a simple vista.

Entre las técnicas más empleadas del aprendizaje no supervisado destaco el clustering, orientado a la segmentación de datos en grupos con características comunes, y los métodos de reducción de dimensionalidad, como el Análisis de Componentes Principales (PCA) o el mapeo no lineal mediante ISOMAP y *Locally Linear Embedding* (LLE), que facilitaron la representación de información compleja en espacios más compactos y comprensibles. Estas aproximaciones no solo facilitaron la exploración de datos de alta dimensionalidad, sino que también demostraron ser útiles en aplicaciones prácticas. Un ejemplo concreto es observable en el estudio de Lucia de Lamadrid llamado: *Análisis computacional de la expresión genética mediante RNA-seq en la respuesta aviar a la infección por el virus IBDV: exploración de métodos de aprendizaje no supervisado* para el ámbito avícola, donde técnicas no supervisadas se aplicaron al estudio de enfermedades en pollos, como la enfermedad de Newcastle. En este caso, los algoritmos de agrupamiento permitieron detectar patrones epidemiológicos y segmentar regiones con mayor incidencia de la enfermedad sin contar con registros clínicos etiquetados, lo que facilitó el análisis de factores de riesgo y la predicción de posibles brotes. Dichos resultados evidenciaron que el aprendizaje no supervisado, además de ser una herramienta teórica, aportó un valor tangible en contextos de salud animal y seguridad alimentaria, reforzando su papel en la generación de conocimiento a partir de datos complejos y poco estructurados.

2.6.3. Redes Neuronales Artificiales.

Las redes neuronales artificiales están orientados a datos, es decir; son capaces de encontrar relaciones de forma inductiva basado en un algoritmo el cual fue entrenado previamente con datos similares (Salas, 2004), sin embargo este tipo de modelo es para la solución de problemas individuales, puede ser una ventaja y una desventaja a la vez, ya que por si solo no es un modelo muy poderoso, pero en combinación a otros modelos, ayuda a evitar la carga computacional en un solo procesamiento, si no en la división se pequeños módulos, son usadas para la clasificación de imágenes, caracteres, mercado de valores e incluso se ha usado para el hardware de computadoras (Ernesto Vela Arregoces,2011).

2.6.3.1. Redes Neuronales Convolucionales.

Es un modelo especial para la clasificación y reconocimiento de las imágenes, es basado en etiquetas es decir; para que pueda reconocer patrones, previamente debe de ser entrenado señalando que una imagen de cierto aspecto o una figura, tiene una etiqueta en especifica, lo que destaca de este tipo de modelos, es el proceso de convolución, además si se combina en un modelo multi capa, la convolución puede ser aplicada en cada una de las neuronas haciendo una “filtración” de información relevante (Beinat,2022). Las aplicaciones de la tecnología se emplean en el área medica para tratar de predecir enfermedades mortales antes de tiempo y estudios en radiología, así mismo lo menciona Federico Lubinus Badillo en su trabajo *“Redes neuronales convolucionales: un modelo de Deep Learning en imágenes diagnósticas. Revisión de tema”*.

2.6.3.2 Redes neuronales recurrentes (RNN)

Las Redes Neuronales Recurrentes (RNN) son un tipo de arquitectura diseñada para trabajar con datos en secuencia, a diferencia de las redes tradicionales que procesan únicamente entradas independientes. Su característica principal es que cuentan con un estado interno o memoria, lo que les permite aprovechar información de pasos anteriores y aplicarla en el presente. Gracias a esta capacidad, resultan especialmente útiles en tareas donde el orden y la dependencia temporal de los datos son determinantes.

Entre sus principales aplicaciones se encuentran el procesamiento del lenguaje natural, como en la traducción automática, el análisis de sentimientos y la generación de texto. Todas estas tareas comparten la particularidad de manejar secuencias dinámicas, donde cada elemento de la entrada depende del contexto previo. Por ejemplo, en una traducción, el significado de una palabra puede variar dependiendo de las palabras anteriores y posteriores en la oración.

No obstante, este tipo de redes presenta limitaciones importantes. Una de las más comunes es su fuerte dependencia de los datos pasados: cuando la secuencia es muy larga, la red puede llegar a “olvidar” información relevante o, por el contrario, dar demasiada importancia a ciertos valores, generando resultados alejados de la realidad. En la práctica, esto significa que, aunque una RNN puede reconocer patrones a corto plazo, le resulta más difícil mantener la coherencia en secuencias extensas o complejas.

2.6.3.3 Redes transformer.

Las redes transoformer es una red neuronal, para el procesamiento de texto basado en mecanismos de atención para procesar datos en paralelo captando las relaciones globales, es especial para el procesamiento de texto y para la clasificación de imágenes. Las redes de tipo transformer ayuda a realizar tareas múltiples como traducción de texto y también puede crear texto según el contexto, el concepto clave de este tipo de redes se llama “*decoder*” y

“*encoder*” el encoder genera una secuencia de vectores y cuando pasa al decoder realiza otra secuencia de vectores, pero cada *decoder*, depende de su antecesor, haciéndolo un modelo autorregresivo (Diego Ignacio,2023).

2.7. La Evolución del Diseño y Desarrollo Web en la Era Digital

La tecnología moderna, más que ser una herramienta, se ha convertido en una necesidad, desde el uso de agendas telefónicas hasta la implementación de computadoras para automatizar tareas en oficinas. Cada uno de estos programas, para ser funcional y accesible para los usuarios, parte de una estructura de trabajo particular. Como lo menciona Cardona en su obra *“Agencias de publicidad, e-branding y ecosistemas digitales: análisis articulado de los metamedios de las agencias de publicidad de Argentina, Chile, Colombia, España, Estados Unidos, México y Perú”*, de manera simple, un programa informático puede dividirse en dos capas: una "delantera" (conocida como frontend) y una "trasera" (correspondiente al backend y a la lógica de negocio). Asimismo, el autor señala que una interfaz estética no solo se enfoca en la transmisión de contenido, sino también en la experiencia de los usuarios, siendo la usabilidad el principal factor determinante para que un producto de software sea calificado como un producto de calidad.

Anteriormente se menciona sobre la experiencia de usuario, a la fecha año 2025, el posicionamiento de una empresa o de una marca no basta con tener un producto “decente” si no en el alcance que tiene tu producto, más allá de lo tangible llegando al nivel de la influencia, pero el origen de aquello es la “experiencia”, pero formalmente la metodología de experiencia de usuario es una filosofía de trabajo que busca la elaboración de productos empáticos para responder las necesidades de los usuarios (Carlos J. Sánchez Morales, 2021), las experiencias no se rigen solo en el contenido digital, sino también en cómo te sientes en un lugar en específico, como te hace sentir un color, o incluso en la misma música habrá hay sensaciones sonoras que algunas les parece agradable que va desde el ritmo hasta la letra de la canción, en secciones más adelante se profundizara más sobre el tema de experiencia de usuario.

La década de 2010 a 2020 marcó un punto de inflexión en la evolución de la web, transformándola de un lujo accesible para unos pocos en una necesidad global. Durante los años 2000, el acceso a internet era limitado y se consideraba un privilegio; sin embargo, en la década siguiente, se convirtió en una herramienta esencial para el trabajo, la educación, la investigación y el entretenimiento. Este cambio no solo se reflejó en la accesibilidad, sino también en la forma en que se diseñan y desarrollan las aplicaciones web, dando lugar a disciplinas especializadas como el diseño de experiencia de usuario (UX) y diseño de interfaces de usuario (UI), enfocadas en mejorar la interacción entre los usuarios y las plataformas digitales.

En este contexto, el diseño web adquirió un papel crucial, especialmente en sectores como el comercio electrónico, donde la psicología del color, la disposición de elementos y la

usabilidad se convirtieron en factores determinantes para captar y retener a los usuarios. Plataformas como Amazon, eBay y Shopify demostraron cómo un diseño atractivo y funcional puede influir directamente en la decisión de compra y la satisfacción del cliente.

Paralelamente, la tecnología detrás del desarrollo web también experimentó una evolución significativa. JavaScript, un lenguaje inicialmente diseñado para agregar interactividad a las páginas web, se consolidó como el pilar del desarrollo frontend. Su versatilidad y capacidad para ejecutarse en el navegador lo convirtieron en la opción predilecta para crear experiencias dinámicas y responsivas. Sin embargo, a medida que las aplicaciones web se volvieron más complejas, surgió la necesidad de estructurar mejor el código y facilitar su mantenimiento. Esto llevó al nacimiento de frameworks y librerías basadas en componentes, como Angular (lanzado en 2010), React (creado por Facebook en 2013) y Vue.js (desarrollado en 2014). Estas herramientas revolucionaron el desarrollo frontend al permitir la creación de interfaces modulares, reutilizables y altamente eficientes, ya lo siguiente de ello serían cómo evoluciona los ecosistemas de cada uno, siendo para bien o para mal, pero agregando características bastante buenas en cada framework.

En particular, React se ha destacado por su flexibilidad y su curva de aprendizaje relativamente corta, lo que lo convirtió en una opción popular para desarrolladores y empresas. Su enfoque basado en componentes y el uso del Virtual DOM permitieron construir aplicaciones web rápidas y escalables, mejorando tanto la experiencia del usuario como la productividad del desarrollador. Además, React facilitó la integración con otras tecnologías y servicios, consolidándose como una herramienta clave para el desarrollo moderno de frontend.

El frontend, en este sentido, dejó de ser simplemente una capa visual para convertirse en un servicio esencial que facilita el acceso de los usuarios a funcionalidades complejas de manera intuitiva. Gracias a frameworks como React, Angular y Vue.js, los desarrolladores pueden crear interfaces atractivas y funcionales que no solo cumplen con las expectativas de los usuarios, sino que también optimizan procesos y mejoran la eficiencia en la interacción con sistemas backend.

2.8. DOM Scripting y el virtual DOM

El DOM (Document Object Model) es un modelo de objetos utilizado por JavaScript para interactuar con documentos HTML. Este modelo representa la estructura del documento como un árbol de nodos, donde cada etiqueta HTML se mapea como un objeto. En esta jerarquía, una etiqueta "padre" puede contener múltiples etiquetas "hijas", mientras que cada etiqueta hija solo tiene un único padre. Para acceder a un elemento específico, es necesario navegar a través de su ancestro, lo que requiere un manejo preciso de la estructura en forma de árbol. Gracias al DOM, JavaScript puede manipular HTML de manera dinámica, permitiendo acciones como la creación y eliminación de nodos, así como la asignación de eventos a elementos específicos (Jaimez González, 2014).

Una de las principales ventajas de React es su capacidad para modularizar el código HTML mediante el uso de componentes, los cuales pueden ser entendidos de forma análoga a las clases en la programación orientada a objetos. En lugar de clases, React emplea componentes que conforman un árbol jerárquico, para utilizar React es necesario contar con un elemento principal donde se renderiza todo el contenido. Este elemento se conoce como "root", y se declara con ese mismo nombre en la configuración del entorno de desarrollo (Meta Platforms, Inc.,2024).

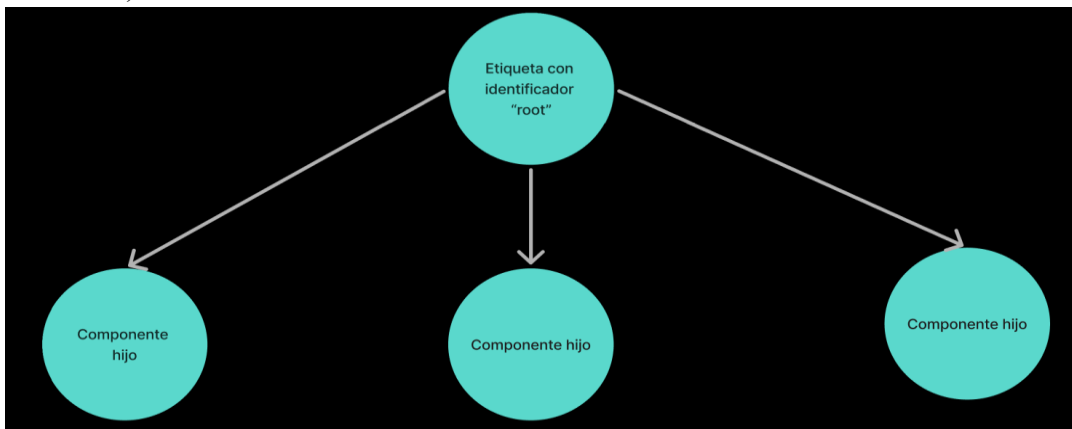


Figura 2. Árbol de componentes.

Para poder renderizar los componentes se hace por medio de una referencia llamada "hydrateRoot" sirve para adjuntar el HTML ya existente (las cuales son aquellas que no tienen jerarquía con root) y el HTML del virtual dom, para poder renderizarlo en un entorno de servidor, ocupando un puerto en específico del equipo.

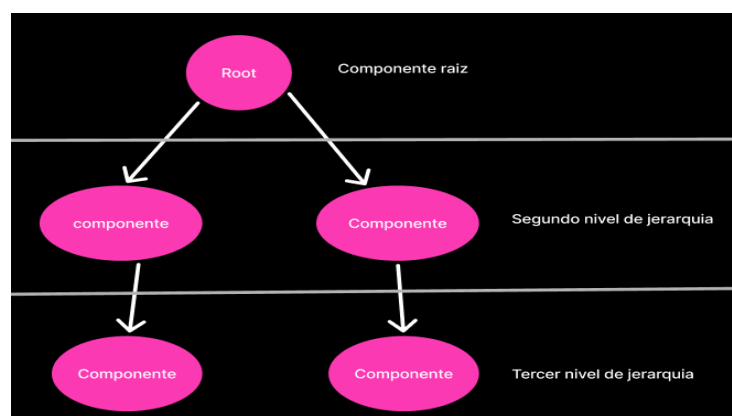


Figura 3. Niveles de jerarquía de componentes.

En cuanto a etiquetas, no importa qué tipo de etiqueta selecciones tu como root, normalmente cuando se crea el entorno de desarrollo, el empaquetador crea una etiqueta div, con el identificador de raíz, esto puede cambiarse de ser necesario o dependiendo la situación en la que se encuentre el desarrollador, pero no es algo que pase comúnmente, esta

forma de árbol de componentes es bastante útil a comparación de otros frameworks web, React trabaja con un solo archivo y dependiendo de la versión con la que se esté trabajando o con el empaquetador que uses; por ejemplo antes se usaba create-react-app el cual por defecto usaba archivos .js, pero después en otros empaquetadores de aplicaciones como vite, se usa la extensión .jsx o .tsx, los 3 tipos de archivos trabajan con un solo archivo para trabajar en un componente, la cual se divide en 3 partes siendo una la declaración del componente siendo el cuerpo del componente y el retorno de componente el cual regresa el código HTML, otros frameworks web basados en js, trabajan con 2 o más archivos, uno para el componente y uno para estilos, pero hay otros que combinan la lógica del componente(código de js para la lógica), el retorno del HTML y el código CSS, algo que puede llegar a ser incómodo para algunas personas, estar desarrollando y cambiando de archivo en cada momento.

Dentro de la lógica de react, la forma en la que se pueden crear eventos, que sería lo más resaltante del entorno, es por medio de la herencia de propiedades, y cuando se habla de herencia, no se refiere a la definición clásica de la POO (Programación Orientada a Objetos) Si no a la transferencia de propiedades denominadas “props”, cada componente en React tiene atributos que son propiedades del componente. Estas propiedades permiten la comunicación entre componentes en diferentes niveles de jerarquía. Por ejemplo, para compartir una variable desde el componente raíz hacia un componente en un tercer nivel de jerarquía, es necesario enviar esa información a través de las props. De esta manera, el componente receptor puede interpretar la información y renderizar los elementos visuales correspondientes. Además, en React existen tres tipos de props: "propTypes", "defaultProps" y “children” (Jiménez, 2016, p. 18-21).

2.9. La creciente tendencia del desarrollo web con manejo de tipos de datos

La comunidad del desarrollo web ha mejorado con el paso del tiempo, y no es poca cosa la comunidad que sostiene la paquetería npm y derivados como lo sería Bun, Yarn, pnpm, la comunidad ha conseguido realizar mejoras que por sí solo usando js es difícil para el usuario programador realizarlo, desde el manejo de las fechas, hasta incluso librerías para poder competir contra Python en entrenamiento de modelos.

Js es potente para lo que es el desarrollo de interfaces de usuario, incluso al grado de llegar a ser usado en microservicios, pero en sistemas más grandes y robustos el asunto se complica ya que js al ser tipado dinámico, afecta directamente a como es que los programadores le dan mantenimiento a una aplicación legacy, pero cuando se desarrolla con Typescript ayuda a la detección temprana de errores (Zheng Gao,2017), las migraciones pueden ser parciales o totales dependiendo como lo solicite la organización, ya que Typescript su tipado es opcional permitiendo al programador usar tipado dinámico como tipado estático.

La creciente necesidad de mejorar el entorno de desarrollo en JavaScript ha impulsado la adopción de TypeScript (TS) como una capa opcional para proyectos web, tanto en frontend como en backend, mejorando herramientas como ESLint. Es importante destacar que, hasta

la fecha, TS funciona principalmente como una herramienta de desarrollo diseñada para optimizar la experiencia del programador. Aunque algunos entornos están comenzando a ofrecer soporte nativo para TS, esta capacidad aún no está ampliamente implementada en la mayoría de los servidores. Esto plantea una pregunta clave: si TS no se ejecuta directamente en la mayoría de los entornos de producción y su propósito principal es asistir en el desarrollo, ¿por qué adoptarlo como un "lenguaje" en sí mismo? Como explica Boris Cherny en su libro "Programming TypeScript", a diferencia de lenguajes como Java, que compilan a bytecode para ejecutarse en una máquina virtual, TS opera de manera distinta: al ejecutarse, todo el código se “traduce” a JavaScript puro, lo que significa que, en esencia, TS actúa como una herramienta de traducción avanzada que mejora el código antes de su ejecución real en JS. Esta característica permite a los desarrolladores beneficiarse de un sistema de tipos robusto y detección temprana de errores, sin romper la compatibilidad con el ecosistema existente de JavaScript.

2.10. Entornos de desarrollo y como afecta el paradigma de programación.

Esta tesis no solo explora cómo el desarrollo web destaca en la creación de aplicaciones, sino también cómo el desarrollo en entornos Android ofrece una variante que traslada la experiencia de una página web a la palma de la mano, sin la necesidad de acceder a un navegador. Este enfoque representa un contraste significativo entre los entornos de desarrollo, ya que no es lo mismo desarrollar para un entorno web, que se ejecuta en un navegador, que crear una aplicación que corre directamente en un sistema operativo móvil. Estas diferencias introducen variantes clave que deben considerarse, pues, aunque el lenguaje de programación pueda ser el mismo, las reglas y herramientas de trabajo cambian considerablemente. Este cambio es aún más notable en entornos Android, donde se tiene la opción de trabajar directamente con un dispositivo físico o mediante un emulador.

Previamente, se mencionó cómo React gestiona sus componentes a través de un concepto conocido como **Virtual DOM** (Document Object Model Virtual). En el caso de Android Studio, la complejidad aumenta de manera significativa. Para contextualizar, las aplicaciones desarrolladas en Java tradicionalmente se compilan y ejecutan mediante la **JVM** (Java Virtual Machine). Sin embargo, al cambiar el paradigma y dirigirse a un sistema operativo móvil como Android, se utiliza una máquina virtual diferente denominada **ART** (Android Runtime). Esta transición ocurrió a partir de la versión Android 5.0 Lollipop, donde ART reemplazó a Dalvik como el entorno de ejecución predeterminado. La principal diferencia entre JVM y ART radica en su enfoque mientras que JVM está diseñada para aplicaciones de escritorio y servidor, ART está optimizada para dispositivos móviles, ofreciendo un mejor rendimiento y eficiencia en el manejo de recursos, la gran línea que hay entre la potencia de un servidor y un dispositivo móvil son muy grandes necesitando un entorno confiable y óptimo para la plataforma. La forma como trabaja java con Android es muy similar a su librería swing, cada vista es una capa o como se le conoce dentro de su entorno, un “panel” dichas capas sirve para la separación tanto lógica y técnica de los componentes, ya sea

imágenes, botones, cuadros de texto, botones de tipo radio etc. Claro todo podría ir en una misma capa, pero el acoplamiento de una misma clase hace que sea difícil de manejar, siendo muy similares swing y Android, pero en el caso de en Adroid cada una de esas capas, se crea por medio de archivos xml, los cuales contienen etiquetas para el esquema, para contextos o redimensionado de imágenes siendo, bastante más avanzado que a como se programa en java swing.

2.11. Conversión de modelos de aprendizaje profundo.

La aplicación web no representa un riesgo potencial para el cronograma de las actividades, es posible realizar un servicio api con los modelos de estimación ya realizados en el lenguaje Python, sin embargo en la aplicación Android primeramente se planteó seguir usando react, en este caso siendo React Native, el detalle es que la librería necesaria para cargar los modelos sigue usando js y la sintaxis de commonjs para la importación de módulos, problema que causa conflictos ya que el entorno de trabajo actual es TS con problemas potenciales al usar versiones anteriores que sigan usando js.

Al plantear un riesgo de compatibilidad el autor junto al equipo de trabajo cambia el entorno de desarrollo, a uno que trabaje directamente con java, sin embargo, el riesgo de compatibilidad de modelos sigue estando ahí, para neutralizar ese posible riesgo, fue necesario la conversión de los modelos de aprendizaje profundo a los de tipo ONNX, debido a la interoperabilidad que ofrece dicho formato.

ONNX es un formato abierto diseñado para representar modelos de aprendizaje profundo, basado en operadores, lo que permite ser usado en diferentes frameworks, con una gran comunidad y soportado por medio de aportaciones en sus repositorios (onnx, 2025).

Capitulo III: Estado del arte.

3.1 Introducción

El desarrollo de interfaces que permiten a los usuarios interactuar con modelos de aprendizaje profundo ha experimentado un crecimiento acelerado en los últimos años, impulsado por la necesidad de acercar la inteligencia artificial a ámbitos prácticos y accesibles para la automatización de tareas. En el capítulo anterior se abordaron los fundamentos técnicos de la IA y, en particular, del aprendizaje profundo, analizando arquitecturas como redes convolucionales, recurrentes y transformadores. Sin embargo, en el presente capítulo se profundiza en los avances aplicados: plataformas en la nube, aplicaciones móviles, APIs con integración de la IA. Un aspecto clave de estas soluciones es que, aunque la complejidad técnica se ejecuta “tras bambalinas” mediante algoritmos avanzados, el usuario interactúa con ellas a través de interfaces gráficas que facilitan la experiencia y orientan la herramienta hacia tareas específicas. Esta revisión resulta esencial no solo para identificar tendencias y limitaciones en la integración de IA en productos actuales, sino también para resaltar la relevancia del diseño como puente entre la capacidad técnica de los modelos y la usabilidad que hace posible su adopción por parte de usuarios no expertos.

3.2. Antecedentes.

Históricamente, la interacción con modelos de inteligencia artificial ha estado condicionada por la complejidad técnica de los entornos de ejecución. La introducción de datos debía realizarse en formatos específicos, generalmente mediante lenguajes de programación o plataformas orientadas a especialistas, lo que limitaba su uso por parte de un público general. Aunque surgieron herramientas con interfaces gráficas como SPSS, WEKA o incluso plataformas de AutoML, estas resultaban en muchos casos genéricas y poco adaptadas a contextos particulares, lo que mantenía la dependencia de expertos para el manejo adecuado de los modelos.

Uno de los principales retos identificados en trabajos previos ha sido precisamente la automatización de la entrada de datos y la simplificación del flujo de interacción, de manera que el usuario pueda concentrarse en la interpretación de los resultados sin necesidad de conocimientos técnicos avanzados. Esta problemática constituye el punto de partida de la presente investigación, que busca diseñar una interfaz de usuario como puente entre los modelos de aprendizaje profundo y las personas, garantizando accesibilidad, eficiencia y confiabilidad en la estimación de valores.

En el marco del proyecto se contó con una serie de modelos de aprendizaje profundo previamente entrenados, los cuales corresponden a combinaciones específicas de parámetros: algunos consideran al menos tres, otros incorporan cuatro e incluso cinco variables en secuencia. Cada modelo responde a una configuración particular, pero lo que hacía falta para llevar a cabo el sistema completo era una estructura de backend que gestionara dichos modelos y, sobre todo, un medio de interacción que acercara esas capacidades al usuario final. Es precisamente en este punto donde se enmarca el presente trabajo, al enfocarse en el desarrollo de la interfaz frontal que permite al usuario interactuar de forma intuitiva con los modelos y obtener resultados de manera práctica.

De forma general, existen librerías en distintos lenguajes de programación que permiten crear interfaces gráficas (Java, C#, entre otros). En el caso de Python, destaca **Gradio**, una herramienta diseñada para crear interfaces automatizadas que facilitan la interacción con modelos de inteligencia artificial. Esta librería elimina la necesidad de construir interfaces tradicionales en HTML y CSS, permitiendo conectar directamente entradas como texto, números o imágenes con funciones que procesan los modelos de IA, y devolviendo resultados en componentes visuales comprensibles para el usuario (Abid et al., 2019).

No obstante, su uso aún requiere experiencia en programación, ya que es necesario preparar los componentes de entrada y salida para que el modelo procese los datos correctamente. Como se muestra en la documentación oficial de Gradio (Figuras 4 y 5), el flujo consiste en vincular los parámetros de entrada con la función del modelo, procesar la información y retornar resultados visibles para el usuario. Este tipo de herramientas refleja cómo se ha intentado resolver la problemática de accesibilidad en IA mediante interfaces automatizadas, lo que a su vez justifica la necesidad de continuar investigando y desarrollando soluciones que reduzcan aún más la barrera técnica para usuarios no especializados, como lo plantea el presente proyecto.

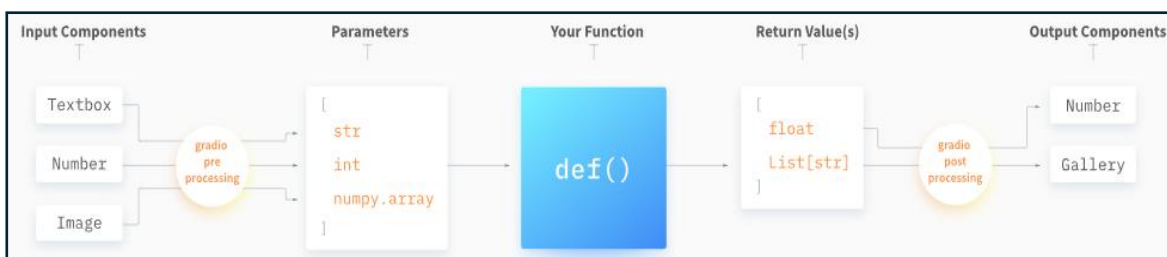


Figura 4. Preprocesamiento de información a través de interfaz de Gradio. (Nota: La imagen fue tomada de la documentación oficial de Gradio en la sección de componentes)

```
gradio.AnnotatedImage(...)
```

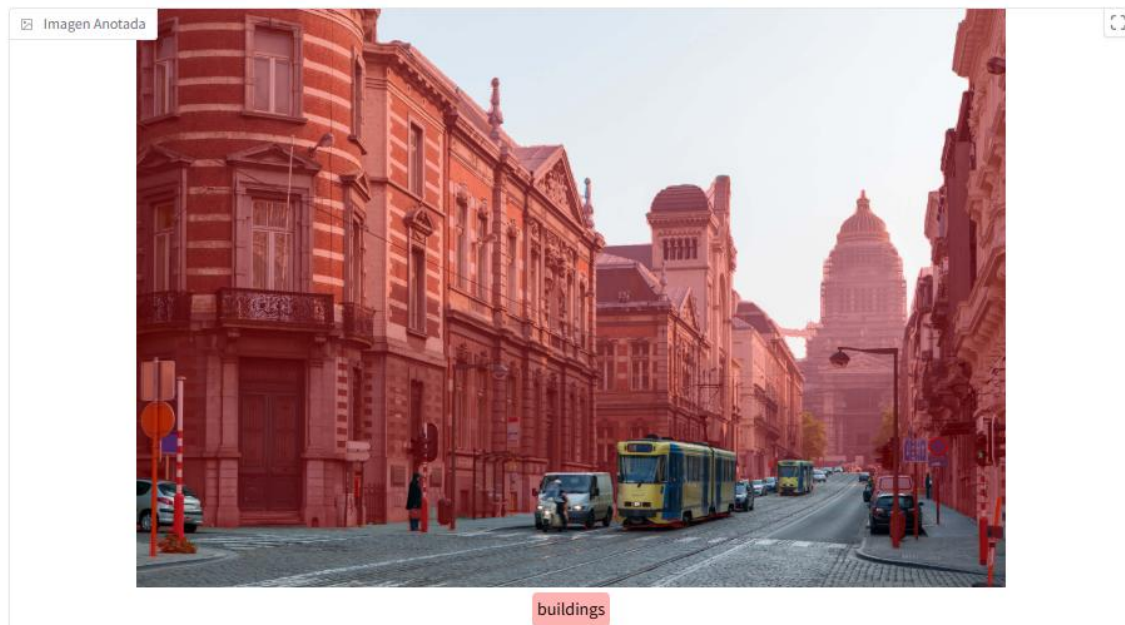


Figura 5. Imagen de interfaz de Gradio (Nota: la imagen fue tomada de la documentación oficial de Gradio en la sección de playground)

3.3. Tendencias actuales

3.3.1 Aplicación móvil para gestión de incidencias de residuos.

Un caso representativo de la integración de modelos de aprendizaje automático en aplicaciones móviles es el trabajo titulado “Aplicativo móvil de procesamiento de imágenes usando Machine Learning para el reconocimiento de incidencias de residuos sólidos en la Municipalidad de Punta Hermosa” (Jonathan Ortiz, 2022). El propósito de este proyecto fue desarrollar una herramienta que, a partir de imágenes capturadas con dispositivos móviles, pudiera identificar y clasificar incidencias vinculadas con la gestión de residuos sólidos.

Desde el punto de vista técnico, la propuesta incorporó un modelo de *machine learning* optimizado para ejecutarse en entornos móviles, empleando *frameworks* especializados que permitieron su despliegue en dispositivos con recursos limitados. La aplicación ofrecía a los usuarios la posibilidad de registrar y enviar imágenes, al mismo tiempo que realizaba inferencias locales para proporcionar resultados inmediatos sobre la presencia de residuos. Este planteamiento demostró la viabilidad de trasladar modelos entrenados en infraestructuras de cómputo hacia plataformas móviles de uso cotidiano.

Este estudio resulta valioso para la presente tesis porque evidencia cómo un modelo de inteligencia artificial puede integrarse en una interfaz sencilla y funcional, lo que facilita su adopción por usuarios no especializados; en este caso, por la ciudadanía en general. De esta

manera, se refuerza la idea de que el éxito de una solución con IA no depende únicamente de la precisión del modelo, sino también del modo en que se incorpora dentro de una aplicación accesible y de alcance masivo.

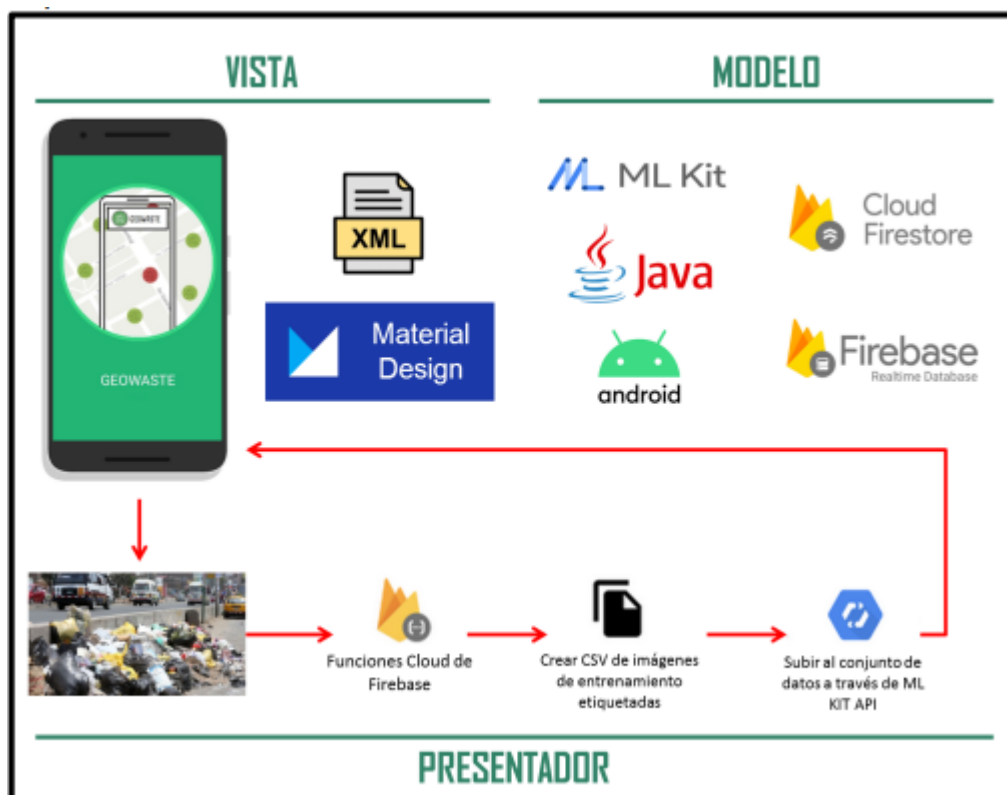


Figura 6. Arquitectura de software para aplicación móvil de gestión de residuos.

(Nota: tomado del trabajo de Jonathan Ortiz, 2022 pag. 93)

Asimismo, conviene señalar la arquitectura planteada: el sistema fue diseñado bajo un esquema desacoplado compuesto por dos partes principales. Por un lado, el *frontend*, correspondiente a la interfaz del aplicativo móvil; y por otro, el *backend*, encargado no solo de procesar los parámetros de entrada y ejecutar el modelo, sino también de coordinar la interacción con servicios externos. Entre estos se incluyeron la API de Google Maps, utilizada para la localización, y Firebase, que permitió gestionar notificaciones y servicios en la nube, garantizando así que la información llegara de manera eficiente a los usuarios de la aplicación.

3.3.2. Aplicación móvil para el reconocimiento facial con redes convolucionales.

Un referente en la integración de modelos de aprendizaje profundo en aplicaciones móviles es el trabajo desarrollado en la Universidad San Francisco de Quito por Samy Amaru Conejo Gualacata (2022) llamado *Prototipo de sistema móvil para reconocimiento de personas*

requeridas por organismos de seguridad y control a través de reconocimiento facial aplicando Redes Neuronales Convolucionales y Transferencia de Aprendizaje. Orientado al reconocimiento facial en dispositivos iOS mediante el uso de CoreML. En este proyecto se entrenaron y adaptaron arquitecturas de *deep learning* como VGG-19 e Inception V3, las cuales fueron convertidas para ejecutarse de manera eficiente en dispositivos móviles. El objetivo central consistió en comprobar la viabilidad de trasladar modelos de visión por computadora desde entornos de alto rendimiento hacia plataformas móviles sin comprometer de forma considerable la precisión del reconocimiento.

Uno de los aspectos destacados de esta tesis es que plantea un flujo de trabajo completo: desde el entrenamiento de los modelos en un entorno de cómputo, pasando por la conversión a un formato compatible con iOS, hasta su despliegue en una aplicación funcional. Los resultados alcanzados mostraron un nivel de precisión cercano al 78 %, lo que permitió validar el potencial de CoreML como puente entre modelos de *deep learning* y aplicaciones de uso cotidiano. Además, este caso pone en evidencia la importancia de la interacción entre la interfaz móvil y el modelo de IA, demostrando que la adopción de estas tecnologías en productos finales depende tanto de la calidad del algoritmo como de la forma en que se logra integrarlo en una plataforma accesible y funcional para el usuario. La demo, denominada “Face AI”, fue concebida como base para una aplicación de seguridad enfocada en reconocimiento facial. En ella se mostraron resultados a partir de una fotografía, vinculados con la información previamente almacenada durante la fase de entrenamiento del modelo, lo que ejemplifica un ciclo completo de desarrollo e integración.

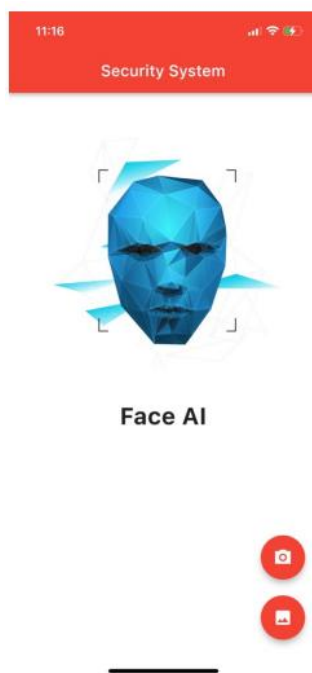


Figura 7. Interfaz de usuario planteada para el reconocimiento facial

(Nota: tomado del trabajo de Samy Amaru Conejo, 2022 pag. 43)

3.3.3. Experiencias en el desarrollo de interfaces para la interacción con modelos de Inteligencia Artificial

Un ejemplo interesante en la integración de modelos de inteligencia artificial con interfaces web es el trabajo desarrollado en la Universidad Estatal Península de Santa Elena por Tomala Tigreros (2025), titulado “Desarrollo de una interfaz basada en algoritmos de procesamiento de lenguaje natural (NLP) para la gestión de conocimiento”. El proyecto consistió en la implementación de un sistema cliente-servidor utilizando Django respaldado con Python para cargar los modelos directamente en el lenguaje y PostgreSQL como base de datos general, siendo las tecnologías principales, con el objetivo de facilitar la búsqueda y recuperación de información mediante consultas en lenguaje natural.

La propuesta tomó como base el modelo de arquitectura de Kerschberg, y para la clasificación de textos se empleó un algoritmo SVM, que alcanzó una precisión del 96 %. Un aspecto innovador fue el uso de datos sintéticos generados con ChatGPT para complementar el entrenamiento, lo que permitió robustecer el modelo sin necesidad de contar con grandes volúmenes de datos reales.

Este caso es especialmente útil como referencia porque demuestra cómo el procesamiento de lenguaje natural puede combinarse con un diseño de interfaz orientado a la simplicidad de uso. En consecuencia, se observa que el valor de una solución basada en IA no radica únicamente en la potencia de los algoritmos, sino también en la capacidad de integrarlos dentro de una interfaz clara y accesible que permita a los usuarios interactuar con el sistema de forma eficiente.

Capítulo IV: Desarrollo del proyecto.

4.1 Enfoque metodológico

La metodología empleada en el presente proyecto se basó en un enfoque de desarrollo de software estructurado por etapas, lo que permitió avanzar de manera ordenada y asegurar que cada fase produjera entregables útiles para la siguiente. Este enfoque resultó adecuado al tratarse de la construcción de un producto de software con requerimientos definidos pero sujetos a ajustes durante el proceso. El trabajo se organizó en sprints con una duración variable de una a cinco semanas, en función de la complejidad de cada tarea y de posibles contingencias técnicas. Esta dinámica permitió mantener flexibilidad en el avance y, al mismo tiempo, garantizar que se cumpliera con los tiempos establecidos.

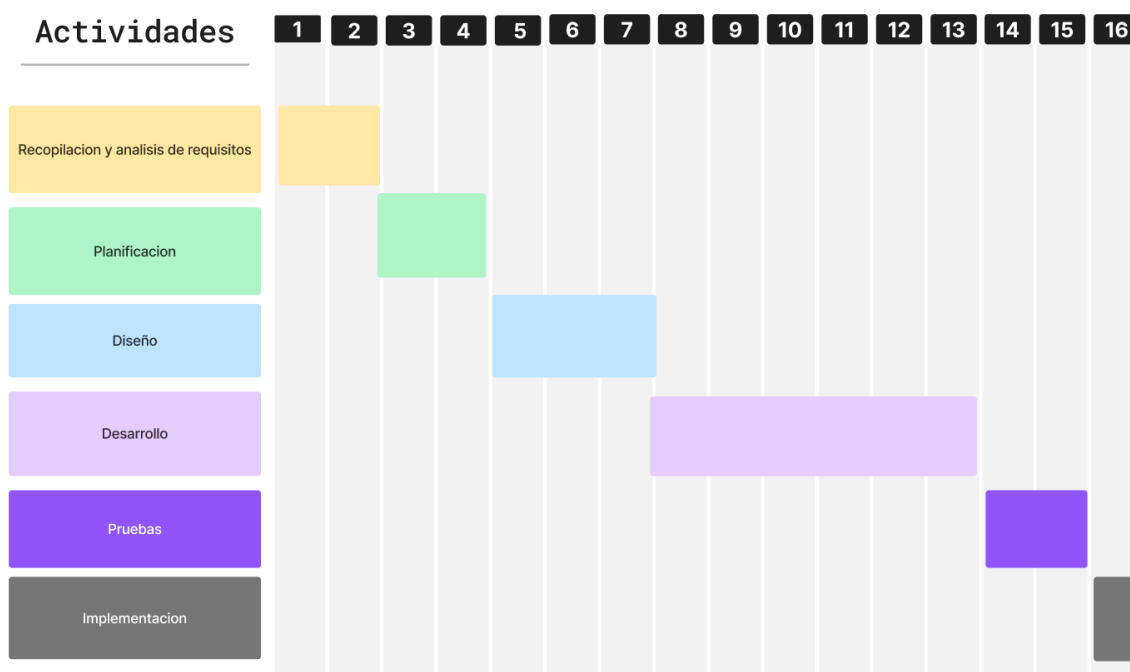


Figura 8. Cronograma de actividades.

4.2. planificación del proyecto

Para organizar las actividades se elaboró un cronograma de trabajo que definió la duración y secuencia de cada fase del proyecto. Este cronograma, representado en la Figura X, contempló un total de 16 semanas de ejecución, de las cuales había 4 evaluaciones para los avances del proyecto y la retroalimentación del avance del proyecto, las semanas serían divididas de la siguiente manera:

- Semanas 1–2: recopilación y análisis de requisitos.
- Semana 3-4: planificación del proyecto.
- Semanas 5–7: diseño de prototipos y validación.
- Semanas 8–13: desarrollo de la aplicación (frontend y backend).

- Semanas 14–15: pruebas funcionales y de usabilidad.
- Semana 16: implementación y despliegue final.

4.1. Recopilación y análisis de requisitos.

En la primera etapa del desarrollo se llevaron a cabo reuniones con el cliente con el propósito de contextualizar a los involucrados respecto al proyecto y clarificar su objetivo. Durante estas sesiones, el equipo documentó los aspectos clave que debían considerarse para la construcción del sistema, entre los cuales destacaron:

1. La necesidad de una estructura capaz de gestionar los parámetros de entrada de los modelos de estimación.
2. La comunicación de los modelos con una API para el envío de resultados vía HTTP.
3. La inclusión de una página de presentación que introduzca la finalidad de la herramienta.
4. El enfoque del sistema en la estimación de nutrientes en el agua, diferenciándose de la noción de contaminantes.
5. La importancia de generar múltiples propuestas de interfaz para iterar y refinar el diseño, evitando depender de un único borrador.
6. El requisito de que la interfaz principal fuera responsiva y compatible con las medidas estándar en entornos web.
7. La adaptabilidad de la aplicación móvil a dispositivos con Android.
8. La integración de los modelos de IA directamente en la aplicación móvil, con el fin de garantizar su funcionamiento sin necesidad de conexión a internet.
9. La identificación de los usuarios finales, que incluyen investigadores de campo, personal administrativo y de laboratorio.
10. La definición preliminar del esquema de los formularios de entrada, otorgando a los desarrolladores una base inicial sobre las demandas de la aplicación.

Las reuniones iniciales permitieron aclarar dudas técnicas y delimitar el alcance del proyecto en dos aplicaciones: una web y otra móvil para Android. Asimismo, se definió que los primeros avances se centrarían en los modelos proporcionados al equipo de desarrollo. En cuanto al espacio de trabajo, se adoptó una modalidad principalmente virtual desde la ciudad de Culiacán, Sinaloa, y, en caso de ser necesaria la presencialidad, se dispuso del uso de los laboratorios de Inteligencia Artificial del Instituto Tecnológico de Culiacán.

Una parte esencial de un proceso de desarrollo de software es la ingeniería de requisitos, es una documentación compleja y requiere de una integración del cliente como si fuera un

miembro del equipo, ahí entran en juego metodologías de trabajo que incluyen a un representante de la empresa para poder comunicar y dudas del equipo de trabajo, en este proceso hay que identificar los requisitos del sistema que deben cumplir para satisfacer las necesidades de los usuarios finales (Escalona, M. J., & Koch, N. 2002).

4.1.1 Requisitos funcionales.

RECF01

Presentación y apertura:

La aplicación al entrar debe de contar con una presentación que hable sobre de lo que se trata la aplicación, y debe de contar con un botón para poder abrir el formulario.

RECF02

Ventana modal

El formulario debe ser una ventana modal, de tal manera que cuando se seleccione un botón de inicio de la página de presentación, que se oscurezca el fondo y muestre una ventana modal que contiene el formulario. La ventana modal debe de contener un botón que, de cierre, que al presionarlo la ventana modal se cierre.

REF03

Flujo y validación.

El formulario al entrar solicita al usuario que escoja uno de los nutrientes que quiere estimar, de los cuáles serán:

Nitrógeno total.

Fosforo total.

Ortofosfato.

Nitrógeno amoniacal.

Nitrógeno total Kjeldahl.

Además de un apartado que se llame “estimar todos”, cada una de las opciones debe de ser en forma de “radiobutton”, si se selecciona uno de ellos y luego seleccionas otro, se debe de borrar la selección anterior.

REF04.

Campos de entrada.

EL formulario debe de contener 5 espacios de texto acompañado de “checkboxes” para la selección de estos, cuando se selecciona uno de estos parámetros de entrada, se debe habilitar

el campo de texto que acompaña a este “checkbox”, además de mostrar los rangos permitidos del parámetro de entrada.

REF05.

Validación de los campos de entrada.

Los campos de entrada del formulario deben respetar las siguientes condiciones:

El numero está dentro de los valores permitidos.

No se aceptan valores en blanco.

Solo se aceptan valores numéricos.

REF06.

Detección de combinación.

Cuando el usuario escoja, un nutriente y al menos 3 parámetros, el frontend debe de mandar una petición hacia el servidor, con esa misma información y lo que muestra en la interfaz es el nivel de

asertividad del modelo, haciendo que sea consciente el usuario de que es lo que estima y la precisión con la que lo hará.

Esta misma dinámica debe de funcionar, aunque selecciones otro parámetro u otro nutriente, ese nivel de confianza debe de cambiar según la combinación, y es necesario validar si la información de la combinación es la correcta y mostrar ese nivel de confianza

REF07.

Formato de la respuesta.

Una vez el usuario completo la selección del nutriente, los valores están dentro de los parámetros normales, mandar una petición con esa información, la aplicación tiene que mostrar el nivel de ese nutriente en medidas de “mg/L”, además de hacer énfasis en el nombre del nutriente del cual se realizó la medición.

REF08.

Todas las combinaciones.

La aplicación debe de contener un campo junto a los nutrientes, que se llame “estimar todos” esta opción al hacer el envío de la información, va a hacer una estimación por cada uno de los nutrientes, los parámetros de entrada no cambian en nada, solo será una comparación por cada uno, el formato debe de ser igual que en el mencionado en “REF07” solo que, en vez de ser una sola impresión, será la impresión de todos los resultados.

REF08.

4.1.2. Requisitos no funcionales

RENF01

Colores.

El formulario debe de presentar uno o dos colores que vienen en el logo de Conagua, además de crear una presentación con una imagen alusiva de fondo de dicha institución.

RENF02

Spinner.

Cuando haya un momento de carga de datos, al hacer una petición al servidor, la vista para no mostrarse estática muestra la animación de un spinner esperando a que los datos estén listos.

4.2. Planificación.

Una vez comprendido el contexto del problema se procedió a elaborar una planificación detallada y definición sobre el alcance del proyecto, teniendo como principios clave como a quienes o a que entidad beneficia el proyecto, además de delimitar el desarrollo hasta el punto de que en caso de requerir un producto adicional, que no tenga que ver con los entregables planteados, sea para otro proyecto de desarrollo, en cuanto a su homologo que sería de definición al combinarlo con la fase de planificación, se clasificaron algunas palabras clave para reconocer estructuras o puntos referenciales, de los cuales serían:

- Diccionarios: Estructura que consiste en llave y valor que representa información representativa de un modelo específico.
- Colección de datos: conjunto de parámetros clasificados por llave y valor, siendo un nombre de un nutriente y el valor que será enviado para representar dicha cantidad (dentro del programa la cantidad de parámetros sería algo a tomar en cuenta).
- Modelo escalar: Sistema computacional para reconocer patrones numéricos y transformar valores simples y unidimensionales.
- Modelo de estimación: Sistema computacional para reconocer patrones numéricos transformados e inferir valores desconocidos a partir de datos observados.
- Entorno virtual: Entorno de ejecución de código Python para simplificar el manejo de paquetes evitando problemas de variables de entorno siendo un “universo” aislado del sistema base.

- Gestión de paquetes: manipulación de librerías por medio del gestor pip, pero para el cambio de versiones y verificar vulnerabilidades que representen dichas librerías.
- Pares: En el contexto del proyecto desarrollado, los "pares" son una estructura de datos que relaciona un modelo estimador con su correspondiente modelo escalador, garantizando que los datos de entrada cumplan con las especificaciones requeridas antes de ser procesados.
- combinación: Serie de parámetros de entrada para ser procesados por modelos.

4.3. Diseño

Aunque no hay un estándar de desarrollo, de modo que tenemos demasiadas elecciones a tomar Anchundia Medrano, A. (2022) al ser una aplicación que no requiere manejo de usuario ni de sesiones, es posible simplificarlo con una arquitectura de cliente a servidor, en el caso de la información a transmitir por medio de los componentes, será obligatorio usar el protocolo http, estándar en el desarrollo web y más aún que el backend está hecho con Flask, el cual soporta http.

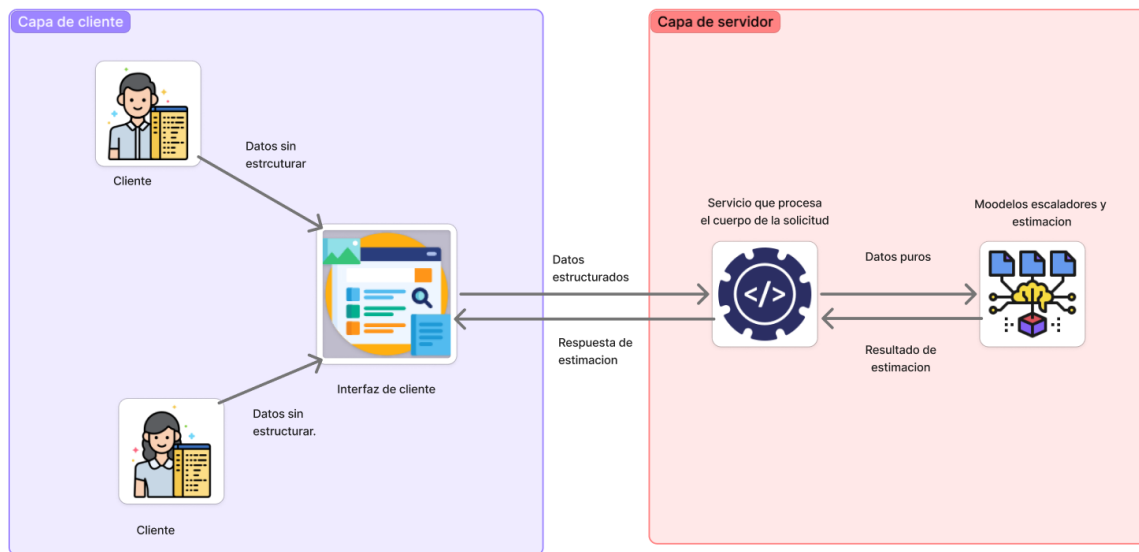


Figura 9. Interacción usuario a aplicación.

En la fase de diseño y pruebas se categoriza dentro de la fase del prototipado en el marco de trabajo del Design Thinking ya que no hay una solución definitiva, abordando problemas que se presenten ya sea a nivel arquitectura o a nivel de codificación y por la interdependencia entre fases, en el diseño se crea una arquitectura para comunicar ambos proyectos, además de que, en el desarrollo, se corrobora la funcionalidad por medio de pruebas adelantándose a la fase de prueba.

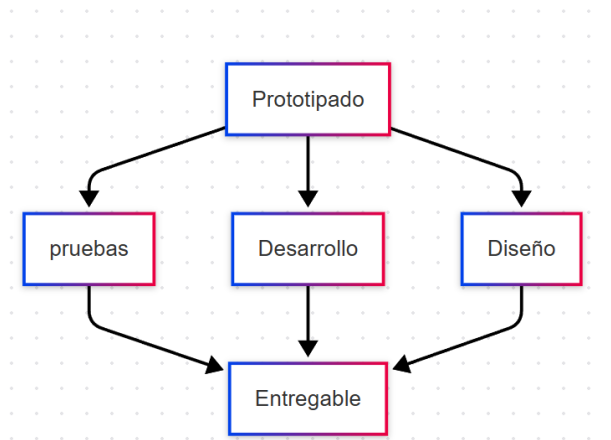


Figura 10. Etapas homologas al prototipado.

4.4. Desarrollo

La fase de desarrollo y más larga en el cronograma de actividades, dicha fase es larga debido a que fue necesario escoger las tecnologías necesarias dentro de un gran repertorio de opciones que hoy en día existen, el autor junto a su colaborador escogió un entorno de desarrollo basado en javascript, que a su vez escogió ReateJs para el desarrollo de interfaces junto a tailwindcss para estilos y css para estilos personalizados, siendo de esa manera un entorno pequeño las demás tecnologías un poco más pequeñas pero que giran en torno de react y javascript serian ReactToastify, Axios, React Router Dom y React Loader Spinner.

Para poder asegurar la integridad del proyecto fue necesario incluir git para la gestión de versiones siendo un “seguro” en caso de que exista un error que parezca irreversible, pero para aumentar aún más la seguridad y evitar perder el proyecto se incluyó Github, un servicio en la nube para almacenar proyectos, que permite a los colaboradores acceder al código, realizar copias y propuestas de solución (Pull request), además servirá más adelante, porque las herramientas de despliegue se apoyan de repositorios de GitHub para desplegar las páginas, lo único necesario a agregar a mano son las variables de entorno, las cuales serían variables que permanecen ocultas para prevenir filtración de información y ocultar credenciales al servidor.

4.5. Pruebas.

En la etapa de pruebas se garantizó la integridad del programa y el correcto manejo de las peticiones HTTP. Inicialmente, se llevaron a cabo pruebas de comportamiento para validar que el programa respondiera adecuadamente a diferentes combinaciones y restricciones. Se estableció que no se permitiría el envío de peticiones si el número de combinaciones seleccionadas era menor a 3 o mayor a 5, siempre y cuando se hubiera seleccionado al menos un nutriente. En caso de que las combinaciones no cumplieran con las restricciones, se programaron mensajes de error específicos: si faltaban parámetros, se mostraba un mensaje

indicando su ausencia, mientras que, si los parámetros estaban fuera del rango permitido, se detallaba el error al usuario. Estas pruebas iniciales aseguraron que el sistema respondiera de manera controlada y mostrara mensajes claros según cada caso.

Posteriormente, se procedió con las pruebas de peticiones HTTP para verificar la comunicación efectiva entre el cliente y el servidor. Se validó que las peticiones enviaran un objeto JSON con el formato requerido, el cual debía incluir un atributo que especificara el nutriente a evaluar y un objeto anidado con los parámetros de entrada y sus respectivos valores. Durante este proceso, se identificaron y corrigieron problemas como nombres de variables incorrectos y formatos de JSON que no cumplían con los requisitos establecidos. Hay varios protocolos para la transferencia de los datos, como SLL o HTTPS (Elias Niño, 2012). Pero el usado en el proyecto es el HTTP, el cual tiene un conjunto de reglas para el envío, como texto, imágenes y sonido, en este caso se enviarían JSON, los cuales se leen como si fuera texto plano en el protocolo http.

Finalmente, se realizaron pruebas exhaustivas de diferentes casos de uso para evaluar combinaciones y escenarios potenciales, con el objetivo de identificar posibles errores tanto en el frontend como en la API. Este proceso permitió depurar y corregir problemas de manera efectiva, asegurando que el programa funcionara de forma consistente bajo diversas condiciones. La etapa de pruebas no solo validó el correcto funcionamiento del sistema, sino que también contribuyó a mejorar su calidad antes de su implementación final.

4.6. Implementación.

Antes de implementar el proyecto en un servidor, se realizó una evaluación de en donde crear el despliegue se evaluaron plataformas de despliegue, entre esas Vercel, Heroku, Railway el que se escogió al final fue Vercel para el frontend y Heroku por no tener tantos problemas con una cuenta de prueba que cubre lo necesario para que la aplicación funcione además de que tiene fuerte compatibilidad con empaquetadores de aplicaciones, siendo el de este proyecto, el de vite, y no requerir ninguna configuración más que las de las variables de entorno, además es un proyecto relativamente pequeño.

Capitulo V: Pruebas y Resultados

5.1 Propuestas iniciales y evolución creativa del diseño.

En el primer capítulo del proyecto, se detallaron las herramientas seleccionadas para el desarrollo. Entre ellas, Figma emergió como la herramienta clave debido a su capacidad para el diseño colaborativo de interfaces de usuario, facilitando la edición simultánea mediante grupos de trabajo.

Las interfaces que se presentan a continuación corresponden a la fase de prototipado, alineada con la metodología Design Thinking. Estas fueron desarrolladas en Figma para optimizar tiempos y cumplir con el cronograma establecido, manteniendo flexibilidad ante posibles ajustes.

Dado el volumen de material generado, solo se incluyen aquellas interfaces que sufrieron modificaciones relevantes, omitiendo versiones menores o iteraciones similares. Estas representan la evolución creativa del diseño, basada en las necesidades identificadas y los feedbacks del asesor. Para un análisis detallado de las interfaces funcionales finales, se remite al lector al manual de usuario adjunto, donde se explican sus interacciones y flujos.

5.1.1 Vista de presentación.

Durante el desarrollo del proyecto, se plantearon una serie de propuestas de diseño con el objetivo de satisfacer las necesidades del proyecto, cabe a destacar este apartado la evolución de las vistas según las observaciones realizadas con los asesores a cargo del proyecto, para empezar como primer requerimiento es necesario una página de presentación, el corazón de la aplicación es el formulario de entrada de los datos, pero para mostrar una vista amigable hay que mostrar los objetivos de dicha aplicación sin que la presentación se vea como una presentación formal, así que la siguiente interfaz sería el primer borrador.



Figura 11. Primer borrador de presentación.

Observaciones:

La interfaz presentada en la figura 9 como base representa problemas con los fondos, demasiado color blanco hace que la interfaz se vea sin contenido alguno, algunos aciertos son el manejo de la paleta de los colores según el logo de Conagua, además de mostrar una imagen que hace referencia a dicha institución, sin embargo no deja de ser un borrador inicial así que las propuestas de diseño deben solucionar 2 problemas, la ausencia de contenido y el posible manejo de más texto en la presentación de la aplicación, problema del que surgió la figura 10.



Figura 12. Segunda propuesta de presentación.

Observaciones: La segunda propuesta de presentación soluciona el problema de la ausencia de contenido, sin embargo, ahora hay 2 problemas nuevos, ausencia de dialogo y problemas en la calidad de la imagen, poco a poco se fue acercando a lo que es al producto final, las siguientes propuestas de diseño juega con la calidad de la imagen, y agregando una descripción más precisa de lo que trata la aplicación, como se muestra en la figura 13 siendo el producto final debido a que las siguientes propuestas son iguales solo con variaciones en la descripción, la opacidad de la zona sombreada y la alineación textual.



Figura 13. Propuesta final de vista de presentación.

Observaciones:

La propuesta de diseño final soluciona los problemas clave que había en comparación con la primera propuesta, ausencia de contenido, usando una imagen en alta calidad de la institución, y la ausencia de descripción y precisión, habiendo suficiente espacio, y el tamaño de la letra necesaria para agregar suficiente contenido, una vez definido que componentes están en la interfaz hace más fácil la tarea para hacer la aplicación responsiva para el formulario.

5.1.2 Propuesta de formularios.

En la etapa de requisitos funcionales, se habló sobre la necesidad de un formulario dinámico, el cual reacciona según los nutrientes que sea escogido, los campos o incluso si se quiere seleccionar, así como primer borrador se diseñó lo que se observa en la imagen 14 la cual tiene las siguientes observaciones.

Figura 14. Versión 1 de formulario

El formulario representa las necesidades principales para la lógica del programa pero cabe a destacar que es necesario simplificarlo aún mas como para que un oficinista pueda hacer las estimaciones rápidamente o incluso una persona que haga trabajo de campo, la interfaz que sería la versión 1 representa una complejidad visual a la hora de mostrar las opciones, para ello se agregaron algunos requisitos más como el despliegue de las opciones de los nutrientes, los campos de entrada que representen una caja de texto por cada uno y que sea seleccionable.

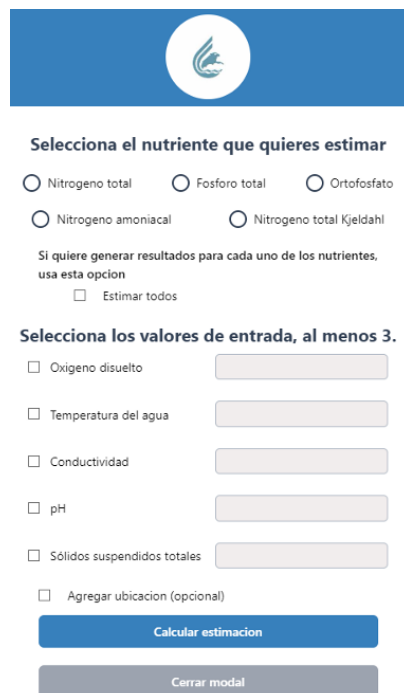
La segunda versión incluye la mejora visual que es necesaria para mostrar las opciones directamente, la figura 13 muestra el contenido total de lo que obtendría según los flujos, en por ejemplo el área en roja mostrar a los campos faltantes que se requiere datos de entrada de forma obligatoria, ahora muestra donde va el nivel de confianza para que aquel que haga la estimación pueda saber que tan acertado puede ser el modelo que se ha seleccionado para esa estimación.

La versión 2 muestra las mejoras visuales, sin embargo, es solamente una demo porque hay mejoras visuales que se pueden hacer como lo sería los campos de entrada, ya que, de forma intuitiva, no se sabe en qué orden se está agregando las áreas de texto de los valores a seleccionar, así que hay un nuevo requisito visual, y es solamente mover de lugar la selección de los campos de entrada con la posibilidad de habilitar y deshabilitar a como se ha seleccionado.

Figura 15. Versión 2 de formulario

La última versión tuvo muy pocas variaciones con su versión preliminar, debido a que se agregó algunos campos como la estimación múltiple de valores según un mismo nutriente pero haciendo cada estimación posible, desde los 3 hasta los 5 valores, además a como se fue

agregando contenido a la interfaz fue necesario hacer más pequeño algunos componentes, pero no al grado de hacerlo imposible de interactuar, además la impresión de los resultado hace necesario que en la misma ventana modal, sea necesario usar una barra para poder navegar de forma vertical, como se observa en la figura 16 siendo la interfaz definitiva.



Selecciona el nutriente que quieres estimar

☐ Nitrogeno total ☐ Fosforo total ☐ Ortofosfato

☐ Nitrogeno amoniacal ☐ Nitrogeno total Kjeldahl

Si quiere generar resultados para cada uno de los nutrientes, usa esta opcion

☐ Estimar todos

Selecciona los valores de entrada, al menos 3.

☐ Oxigeno disuelto

☐ Temperatura del agua

☐ Conductividad

☐ pH

☐ Sólidos suspendidos totales

☐ Agregar ubicacion (opcional)

Calcular estimacion

Cerrar modal

Figura 16. Formulario final.

5.1.3. Interfaz de aplicación móvil.

El diseño en una aplicación web puede presentar una tarea relativamente sencilla dependiendo de la complejidad de los requerimientos visuales como los tamaños, el estándar para dispositivos aún más pequeños y demás cosas, sin embargo traspasar una interfaz que fue hecha para la web a un aplicativo móvil representa el reto de 2 puntos, hacer una interfaz desde cero priorizando los tamaños o tratar de redimensionar los tamaños de la web, siendo el primer caso el adoptado por el autor y el equipo de trabajo.

Como ya hay un esquema de como son las interfaces según la web, en la página de presentación se ha traspasado el texto, dejando como diferencia el formato como se renderizan los componentes de Android, solo que hay diferencias como se muestra en la figura 17, se agregó un botón para ver los resultados de estimaciones hechas desde ese dispositivo antes debido a la necesidad de que el aplicativo móvil, sea para trabajar sin la necesidad de señal, en el caso de que la persona que lo use, se encuentre en un lugar remoto o con problemas climáticos que interfieran en la señal.

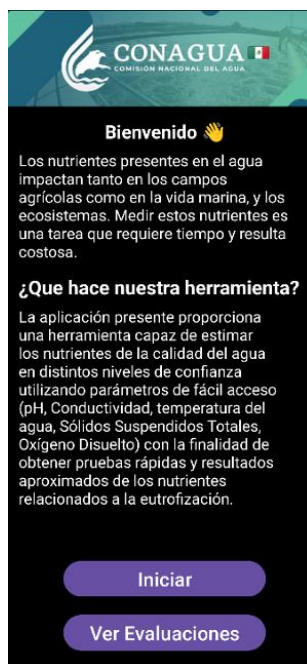


Figura 17. Interfaz gráfica de aplicativo móvil.

El esquema de la ventana modal cambia debido al espacio de la aplicación, sin embargo sigue representando cada uno de los valores planteados en la ingeniería de requisitos, solo que la dinámica en cómo funciona cambia, cuando seleccionas uno de los valores, se despliegan cada uno de los campos, al seleccionar mínimo 3 se muestra áreas de texto para que ingreses los valores de entrada, a cómo vas usando la aplicación, la navegación vertical va siendo necesaria para seguir observando los valores, datos que se observaran en la siguiente fase que es de un manual de usuario.

Figura 18. Formulario de Android.

5.2. Despliegue.

El producto web en total fue necesario realizar 2 despliegues para el funcionamiento correcto de la aplicación, siendo el de este proyecto un despliegue en la plataforma de vercel, para hacer el despliegue fue necesario apoyarse de la plataforma de GitHub, el cual permite subir código de tu maquina local, a la nube para que otros equipos puedan descargar ese mismo código, vercel tiene una implementación en la que cuando creas la cuenta, te permite llamar a esos repositorios y hacer un despliegue rápidamente, sin requerir instalación o configuración adicional, la única configuración que necesita es el archivo de variables de entorno, donde se guarda la dirección del backend, configuración de conexión con el servidor, credenciales y demás cosas, como es posible apreciar en la imagen 19, la serie de pruebas que el equipo realizo, al final el repositorio que predomino como la versión final para el lanzamiento seria el de “Conagua-definitivo-[tag]”.

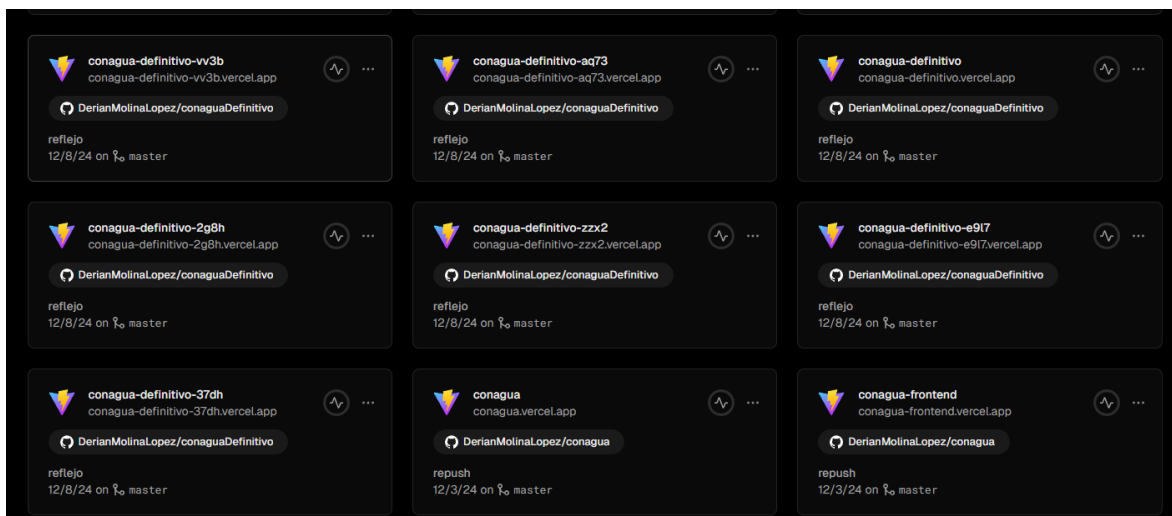


Figura 19. Panel de pruebas de despliegue.

Al acceder al panel de despliegue correspondiente al proyecto “Conagua-definitivo-vv3b”, se visualiza el estado actual del despliegue, junto con metadatos críticos para su auditoría. Entre estos destacan:

Fecha de creación: Indicador temporal que suele coincidir con la entrega del producto, lo que permite verificar la actualidad del despliegue.

Source (origen): Referencia a la rama principal del repositorio de GitHub utilizada para la construcción del aplicativo, asegurando la trazabilidad del código implementado.

Adicionalmente, el panel ofrece funcionalidades clave para la gestión del proyecto:

Logs: Registros detallados que permiten analizar el comportamiento del sistema durante y después del despliegue.

Rollback: Mecanismo para revertir a versiones anteriores en caso de errores, garantizando la estabilidad operativa. En la sección Dominios, se identifican tres URLs asociadas al despliegue:

Enlace público: Dirección accesible para los usuarios finales, diseñada para interactuar con la interfaz del aplicativo y consumir los servicios del backend. Este es el recurso entregado formalmente al cliente.

Enlace de GitHub: Vinculado automáticamente a la rama del repositorio, actualizándose con cada push para reflejar cambios en tiempo real (útil para entornos de desarrollo continuo).

Enlace privado: URL restringida al administrador del proyecto, gestionada por Vercel con fines de monitorización interna.

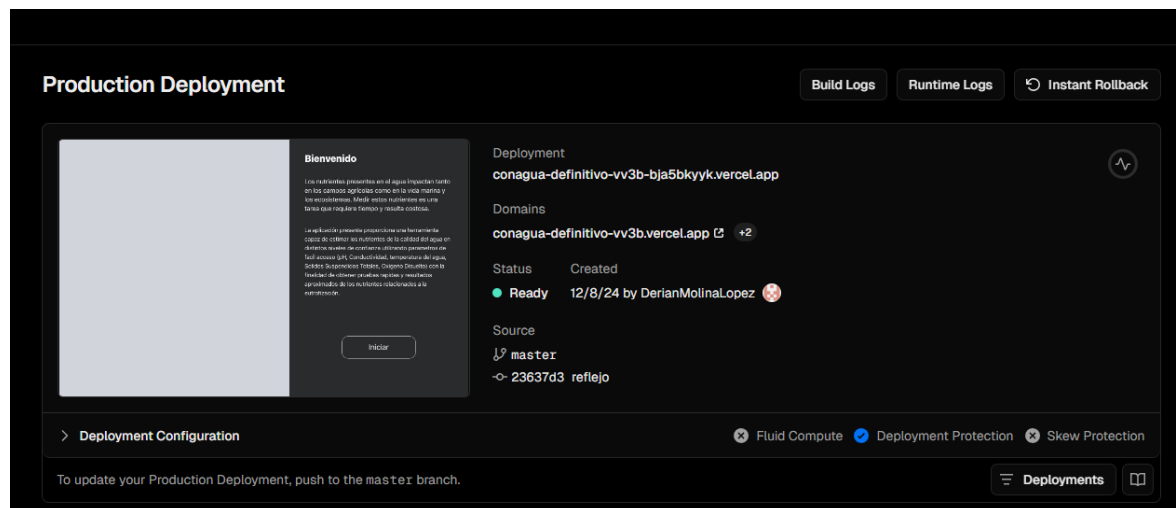


Figura 20. Panel de administración de un despliegue en específico.

5.3.Manual de usuario (aplicación web)

Para cumplir con los requerimientos de la aplicación, el residente realizó una serie de maquetaciones iniciales, las cuales fueron complementadas con la retroalimentación proporcionada por los asesores. A lo largo del proceso, las interfaces fueron diseñadas y modificadas en su debido momento, siguiendo las indicaciones y sugerencias recibidas. Este enfoque iterativo permitió ajustar las interfaces de acuerdo con las necesidades del proyecto, asegurando que cumplieran con los objetivos de funcionalidad y usabilidad establecidos.



Figura 21. Página principal del aplicativo.

La página principal de la aplicación está diseñada para proporcionar una descripción clara sobre qué es la aplicación, qué funciones realiza y cuál es su objetivo principal. Esta información introductoria se presenta de manera concisa para que el usuario comprenda rápidamente el propósito de la aplicación. Además, se incluye un botón de "Inicio" que, al ser presionado, abre una ventana modal. Dentro de esta ventana modal, el usuario encuentra un formulario de estimación, donde puede seleccionar los parámetros necesarios para realizar las estimaciones relacionadas con los nutrientes y la calidad del agua.

Figura 22. Ventana modal con el formulario.

La ventana modal está diseñada para guiar al usuario a través de una secuencia clara de pasos. En primer lugar, el usuario debe seleccionar el nutriente que desea estimar. Este paso es esencial, ya que el objetivo principal de la aplicación es estimar el porcentaje de ese nutriente. Además, el usuario tiene la opción de estimar todos los valores, lo que le permite calcular los porcentajes de varios nutrientes de forma simultánea.

Una vez que se ha seleccionado el nutriente, el siguiente paso es elegir los valores de entrada. La aplicación solicita estrictamente tres valores, y cada campo de entrada incluye una caja de selección que permite al usuario habilitar o deshabilitar el campo según sea necesario. Esto permite una interacción flexible y controlada, asegurando que el usuario seleccione solo los datos relevantes para la estimación.

The screenshot shows a web interface for selecting nutrients and input values. At the top, there is a blue header with a circular logo containing a stylized leaf. Below the header, the text "Selecciona el nutriente que quieres estimar" is displayed. Underneath, there are five radio button options: "Nitrogeno total", "Fosforo total", "Ortofosfato", "Nitrogeno amoniacal", and "Nitrogeno total Kjeldahl". The "Nitrogeno total Kjeldahl" option is selected. Below these options, there is a text prompt: "Si quiere generar resultados para cada uno de los nutrientes, usa esta opcion", followed by a checkbox labeled "Estimar todos". Below this, the text "Nivel de confianza: 86.0 %" is displayed in red. The next section is titled "Selecciona los valores de entrada, al menos 3.". It contains five checkboxes: "Oxigeno disuelto", "Temperatura del agua", "Conductividad", "pH", and "Sólidos suspendidos totales". The first three are checked. To the right of each checked checkbox is a text box showing a range of values: "Rango de valores 1 - 21.66" for "Oxigeno disuelto", "Rango de valores 16.3 - 39.6" for "Temperatura del agua", and "Rango de valores 63 - 148300" for "Conductividad". Below these, there are two more checkboxes: "Agregar ubicacion (opcional)" and "Calcular estimacion". The "Calcular estimacion" checkbox is checked, and there is a blue button labeled "Calcular estimacion" at the bottom.

Figura 23. Ventana modal para la selección de nutrientes y entradas

Una vez seleccionado los 3 valores de entrada cuando hay un nutriente también seleccionado, la aplicación manda una petición del api para verificar cual es el nivel de confianza de esa combinación, pero si quieres ingresar valores, tienes que hacerlo dentro de los limites aceptados, el oxígeno disuelto acepta valores entre 1 – 21.66 unidades de esa entrada.

Selecciona el nutriente que quieres estimar

☐ Nitrogeno total ☐ Fosforo total ☐ Ortofosfato

☐ Nitrogeno amoniacal ☒ Nitrogeno total Kjeldahl

Si quiere generar resultados para cada uno de los nutrientes, usa esta opción

☐ Estimar todos

Nivel de confianza: 86.0 %

Selecciona los valores de entrada, al menos 3.

☒ Oxigeno disuelto 1

☒ Temperatura del agua 16.34

☒ Conductividad 64

☐ pH

☐ Sólidos suspendidos totales

Valor estimado 14.079 mg/L del Nitrogeno total Kjeldahl

Figura 24. estimación de nutriente.

Una vez que la petición con la información recolectada es enviada al servidor, este procesa los datos y devuelve el valor estimado para el nutriente seleccionado. La unidad estándar que se maneja en este proyecto para los valores estimados es mg/L (miligramos por litro), que es la medida comúnmente utilizada para la concentración de nutrientes en el agua.

En el caso de la estimación de todos los nutrientes, la aplicación proporciona una estimación individual para cada nutriente seleccionado, y cada uno de estos resultados se muestra en el mismo formato de mg/L. De tal manera que, aunque el usuario haya seleccionado la opción de estimar todos los nutrientes, cada nutriente será estimado por separado y se presentará su resultado correspondiente en la unidad estándar de mg/L, asegurando que la información se mantenga consistente y clara como se observa en la figura 25.

Valores estimados para cada nutriente

Nutriente: Nitrógeno total

Valor estimado: **7.016 mg/L**

Nivel de confianza: **78.0%**

Nutriente: Fósforo total

Valor estimado: **1.302 mg/L**

Nivel de confianza: **71.5%**

Nutriente: Ortofosfato

Valor estimado: **0.687 mg/L**

Nivel de confianza: **72.0%**

Nutriente: Nitrógeno amoniacal

Valor estimado: **6.308 mg/L**

Nivel de confianza: **86.0%**

Nutriente: Nitrógeno total Kjeldahl

Valor estimado: **7.514 mg/L**

Nivel de confianza: **86.0%**

Figura 25. Resultado de estimación masiva

El flujo mostrado con las figuras anteriores es en el caso del éxito, ahora hay que mostrar el caso de fracaso de la petición, o más bien las limitaciones programadas para asegurar, que el hilo de ejecución principal se cumpla.

Selección del nutriente que quieres estimar

☐ Nitrogeno total
 ☐ Fosforo total
 ☐ Ortofosfato
☒ Nitrogeno amoniacal
 ☐ Nitrogeno total Kjeldahl

Si quiere generar resultados para cada uno de los nutrientes, usa esta opción

☐ Estimar todos

Nivel de confianza: 74.74 %

Selección los valores de entrada, al menos 3.

☒ Oxigeno disuelto Rango de valores 1 - 21.66
☐ Temperatura del agua
☐ Conductividad
☒ pH Rango de valores 0.2 - 10
☒ Sólidos suspendidos totales Rango de valores 5 - 12440
☐ Agregar ubicacion (opcional)

Calcular estimación

Bienvenido

Debes seleccionar al menos 3 valores y llenar sus campos

Los nutrientes presentes en el agua impactan tanto en los campos agrícolas como en la vida marina y los ecosistemas. Medir estos nutrientes es una tarea que requiere tiempo y resulta costosa.

La aplicación presente proporciona una herramienta capaz de estimar los nutrientes de la calidad del agua en distintos niveles de confianza utilizando parametros de facil acceso (pH, Conductividad, temperatura del agua, Solidos Suspendidos Totales, Oxigeno Disuelto) con la finalidad de obtener pruebas rapidas y resultados aproximados de los nutrientes relacionados a la eutrofización.

Iniciar

Figura 26. Validación de campos vacíos.

En términos simples, como los modelos que están en el servidor solo acepta valores numéricos, así que se lanza una advertencia para que el usuario haga la corrección correspondiente como se aprecia en la figura 27.

Selección del nutriente que quieres estimar

☐ Nitrogeno total
 ☐ Fosforo total
 ☐ Ortofosfato
☐ Nitrogeno amoniacal
☒ Nitrogeno total Kjeldahl

Si quiere generar resultados para cada uno de los nutrientes, usa esta opción

☐ Estimar todos

Nivel de confianza: 86.0 %

Selección los valores de entrada, al menos 3.

☒ Oxigeno disuelto 50
☒ Temperatura del agua 17
☒ Conductividad 64
☐ pH
☐ Sólidos suspendidos totales
☐ Agregar ubicacion (opcional)

Calcular estimación

Bienvenido

El valor de Oxigeno disuelto está fuera de rango (1 - 21.66)

Los nutrientes presentes en el agua impactan tanto en los campos agrícolas como en la vida marina y los ecosistemas. Medir estos nutrientes es una tarea que requiere tiempo y resulta costosa.

La aplicación presente proporciona una herramienta capaz de estimar los nutrientes de la calidad del agua en distintos niveles de confianza utilizando parametros de facil acceso (pH, Conductividad, temperatura del agua, Solidos Suspendidos Totales, Oxigeno Disuelto) con la finalidad de obtener pruebas rapidas y resultados aproximados de los nutrientes relacionados a la eutrofización.

Iniciar

Figura 27. validación de rangos.

La validación de los rangos numéricos es posible visualizarse gráficamente, dando una guía a los usuarios que valores tiene que permitir y cuales son posibles, si la aplicación tiene varios campos que están fuera de rango, el manejo de los errores ira por orden desde el primer parámetro hasta el último hasta descartar cada error que haya.

4.Manual de usuario en aplicación Android.

En el caso de la aplicación Android, mucha de la interacción se simplifica aún más, sigue teniendo su presentación que muestra que es lo que hace la aplicación, además de contar un formulario adaptado para los dispositivos, el desarrollo de esta aplicación se realzo con Android Studio trabajando con java.

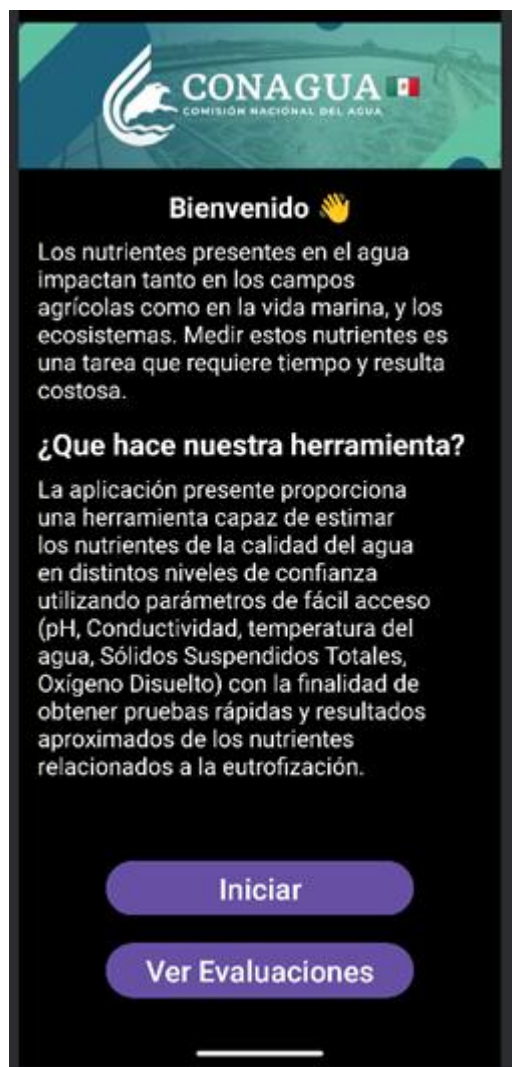


Figura 28. Página de presentación de aplicación Android.

La interfaz de Android cambia para poder tener una mejor adaptabilidad a los dispositivos, cuenta con la misma opción de inicio donde se encuentra el formulario de estimación pero cuenta con una opción extra la cual es de ver las evaluaciones, en la aplicación web, las estimaciones se guardaban en una base de datos, en este caso también puede guardar en base

de datos pero al tener en cuenta un caso de uso que no se había previsto en las entrevistas, es que los dispositivos se pueden quedar sin señal ya que hay lugares de muestreo donde la señal no llega y puede generar complicaciones, así que, si un usuario usa la aplicación sin señal, no hay problema con los registros ya que puede tener guardados en registros locales, lo que hay que seguir para cumplir con el flujo de ejecución es presionar el botón de iniciar.

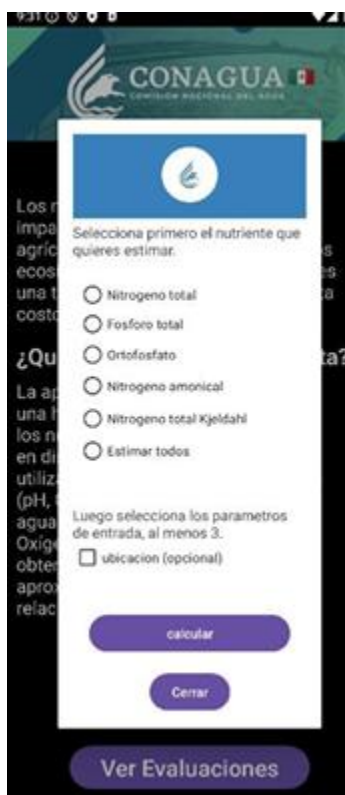


Figura 29. Ventana modal para aplicación Android

Después de presionar el botón aparecerá la ventana modal, la cual si es posible ver, no tiene las cajas de texto, y eso es debido que, para mejorar la experiencia del usuario, las cajas de texto van apareciendo de forma dinámica a como continúe con los requerimientos que necesita, siendo el primero, seleccionar una de las opciones para luego escoger los parámetros de entrada.

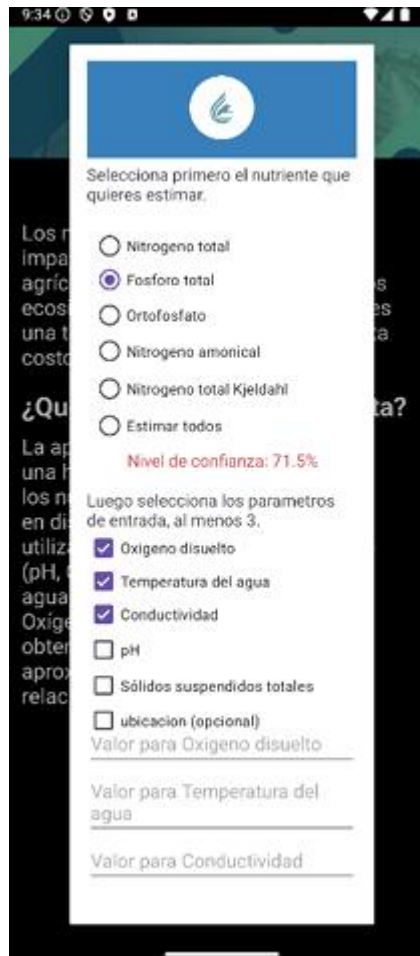


Figura 30. Selección estricta de los 3 nutrientes en aplicación Android.

Una vez mostrado los parámetros de entrada, se mostraran de forma dinámica los área de texto para la entrada, suponiendo que si tienes 3 valores de entrada seleccionados y eliminas la selección de uno, todas las cajas de texto van a desaparecer, haciendo al usuario que para que siga el flujo de éxito, tenga que escoger mínimo los 3, pero en caso de que hayan 4 y quitas la selección de uno, no hay problema ya que el área de texto de el parámetro de entrada que escogiste, se eliminara del formulario y tendrías exactamente los mismos que tienes seleccionadas.

☐ ubicacion (opcional)

17

64

5

Valor estimado: 1.4043084
para el nutriente: nitrogeno
con un grado de confianza de: 78.0

Valor estimado: 0.41924086
para el nutriente: fosforo
con un grado de confianza de: 71.5

Valor estimado: 0.042340443
para el nutriente: ortofosfato
con un grado de confianza de: 72.0

Valor estimado: 0.21671227
para el nutriente: amoniaco
con un grado de confianza de: 86.0

Valor estimado: 1.5628687
para el nutriente: TKN
con un grado de confianza de: 86.0

Figura 31. Estimación masiva en aplicación Android.

Si el usuario selecciona la opción de "Estimar todos", deben abrirse los campos de texto correspondientes. Dependiendo de los campos que seleccione, se enviará la petición con los mismos datos, pero para diferentes modelos. Lo que diferencia cada una de las peticiones es el nutriente que se va a evaluar, lo que marcará la diferencia en los valores numéricos que se devuelvan, además de indicar qué nutriente fue estimado. Hasta este punto, se ha mostrado el flujo principal y de éxito del proyecto; sin embargo, es necesario abordar los casos de fracaso. En estos casos, el programa debe limitar las acciones del usuario mediante validaciones, como la verificación de campos vacíos o la validación de caracteres numéricos en los campos de entrada.

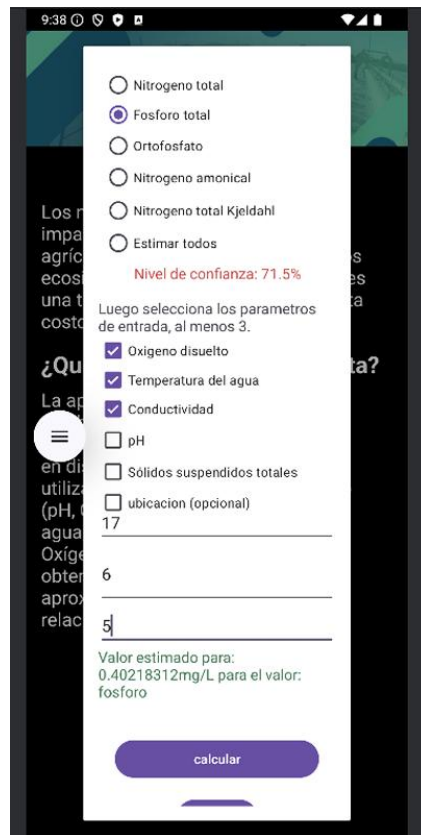


Figura 32. Estimación simple en aplicación Android.

La aplicación muestra el valor de la estimación y para que nutriente fue estimado, teniendo en cuenta el que se seleccionó en la parte superior de la interfaz, igual que como ocurre en la aplicación web, que muestra el resultado estimado junto a la medida necesaria.

Luego selecciona los parametros de entrada, al menos 3.

- ☒ Oxigeno disuelto
- ☒ Temperatura del agua
- ☒ Conductividad
- ☐ pH
- ☐ Sólidos suspendidos totales
- ☐ ubicacion (opcional)

Valor para Oxigeno disuelto

Valor para Temperatura del agua

Valor para Conductividad

calcular

 Hay campos vacíos.

Figura 33. Validación de los campos vacíos.

El usuario debe de llenar los 3 campos seleccionados para estimar la temperatura, en caso de que deje los campos vacíos, a este le saltara una pequeña leyenda que diga que hay campos vacíos, con un campo vacío que haya sirve para que no te acepte la petición.

Estimar todos

Nivel de confianza: 71.5%

Luego selecciona los parametros de entrada, al menos 3.

- ☐ Oxigeno disuelto
- ☒ Temperatura del agua
- ☒ Conductividad
- ☐ pH
- ☒ Sólidos suspendidos totales
- ☐ ubicacion (opcional)

dqdqa

3

4

calcular

 Todos los campos deben ser números válidos.

Figura 34. Validación numérica.

Si el usuario llena uno de los campos con algún carácter que no sea numérico, tendrá la leyenda mostrada en la imagen 34, la cual lo obligara a hacer la corrección, teniendo en cuenta eso, el usuario, aunque agregue números con letras por alguna razón, el modelo no tendrá problemas porque no se ejecutara, al contrario, va a pedir la corrección.

Capitulo VI: Conclusiones y trabajo futuro

6.1. Hallazgos

Entre los hallazgos más relevantes se encuentra la exploración y uso de plataformas en la nube como Vercel para el despliegue de aplicaciones. Tradicionalmente, los sistemas se alojaban en servidores físicos que requerían de una IP pública para permitir el acceso externo; sin embargo, con el uso de estas plataformas se simplificó el proceso de despliegue, limitándose únicamente a la configuración de variables de entorno para la correcta comunicación con la API. Esta investigación abrió un catálogo de opciones que incluye proveedores especializados en hosting de aplicaciones backend, frontend, servicios disparadores de eventos y tableros de gestión. En el caso de la aplicación Android, surgió la necesidad de convertir los modelos al formato ONNX, lo que permitió aprovechar la librería disponible en Android Studio y, de esta manera, implementar un backend integrado directamente en la aplicación.

6.2. Limitaciones y mejoras a futuro.

Dentro de los alcances del proyecto se cumplieron los objetivos planteados tanto en la parte web como en la móvil. Para la aplicación Android se logró generar el archivo APK, mientras que para la aplicación web se realizaron los despliegues correspondientes. No obstante, para una futura modificación de la API ya sea la incorporación de más parámetros de entrada, nuevos modelos o versiones optimizadas de los existentes será necesario respetar la estructura de archivos definida en los diccionarios de Python, pues cualquier alteración incorrecta podría ocasionar errores en la recepción de parámetros de entrada.

En cuanto a las mejoras futuras, se considera una buena práctica reemplazar el alojamiento en una máquina virtual por el uso de contenedores Docker orquestados con Kubernetes. Esto permitiría que, en caso de una liberación a nivel nacional, la aplicación pueda escalar de forma horizontal y responder adecuadamente al tráfico generado por los usuarios. En el caso del frontend, se plantea una estrategia similar para evitar la saturación de un único dominio y asegurar un rendimiento estable.

VII. Bibliografía

- Amaro Calderón, S. D. (2007). *Metodologías Ágiles*. Trujillo : Universidad Nacional de Trujillo.
- Anchundia Medrano, L. A. (2022). *Análisis comparativo de tecnologías Front End Angular Js Vs React Js, en el modelo de procesos para el desarrollo de aplicaciones web*. UNIVERSIDAD TÉCNICA DE BABAHOYO.
- BEINAT, N. C. (2022). *REDES NEURONALES CONVOLUCIONALES Y APLICACIONES*. Madrid: Universidad Complutense de Madrid.
- Blázquez, M. S. (s.f.). *Design Thinking*. ESIC.
- Bongianino, C. S. (2019). *LA APLICACIÓN DE LA INTELIGENCIA ARTIFICIAL EN LA CONTABILIDAD PRIVADA Y EN EL SECTOR GUBERNAMENTAL*. UNIVERSIDAD DE LA PLATA FACULTADES DE CIENCIAS ECONOMICAS.
- Bragean Luis Vargas Márquez, L. A. (2021). *Design Thinking aplicado al Diseño de Experiencia de Usuario*. Arequipa, Perú: Universidad Nacional de San Agustín de Arequipa, Arequipa, Perú.
- Cardona, S. (2020). *Agencias de publicidad, e-branding y ecosistemas digitales: análisis articulado de los metamedios de las agencias de publicidad de Argentina, Chile, Colombia, España, Estados Unidos, México y Perú*. Universidad Autonoma de Barcelona.
- Carlos J. Sánchez Morales, E. A. (2021). *Metodología UX: Innovación pedagógica en tiempos de formación*. Universidad de Las Palmas de Gran Canaria.
- Carlos, C. C. (2024). *Aplicación web para la gestión de inventarios de la empresa*. Riobamba, Ecuador: UNIVERSIDAD NACIONAL DE CHIMBORAZO.
- Casadiego, Y. A. (2020). *Inteligencia artificial (IA) y sus aplicaciones en la agricultura moderna*. Universidad Nacional Abierta y a Distancia (UNAD).
- Cherny, B. (2019). *Programing Typescript*. O'Reilly Media, Inc.
- Díaz Ramírez, J. (2021). *Aprendizaje automático y aprendizaje profundo*. Universidad de Tarapacá Departamento de Ingeniería y Tecnologías Iquique, Chile.
- Ernesto Varela Arregocés, E. C. (2011). *Redes Neuronales Artificiales: Una Revision del Estado del Arte, Aplicaciones Futuras*. Barranquilla: Universidad Simon Bolivar.
- Espinoza Vásquez, J. C., & Espinoza Zapata, E. E. (2025). *Marco de trabajo en base a Design Thinking y*. Universidad Peruana de Ciencias Aplicadas (UPC).
- Ignacio, D. N. (2014). *Diseño emocional y experiencia de usuario*.

- Jaimez González, C. R. (2014). *Programación de Web Dinámico*. Ciudad de México (: UNIVERSIDAD AUTÓNOMA METROPOLITANA.
- Jiménez, A. (2011). Steve y sus aportaciones al mundo del Diseño. *Espacio Diseño*, 12-16.
- Jimenez, J. (2016). *Descubre React 2da edicion*.
- María José Escalona, N. K. (2002). *Ingeniería de Requisitos en aplicaciones web - un estudio comparativo*. Sevilla: Escuela Técnica Superior de Ingeniería Informática Universidad de Sevilla.
- Mora, V. M. (2005). *Psicología del color y forma*. Universidad de Londres.
- onnx. (5 de 05 de 2025). *ONNX*. Obtenido de onnx-ai: <https://onnx.ai>
- Ortega, M. S. (s.f.). *Design Thinking*. ESIC.
- Ruiz, C. P. (2016). *PSICOLOGIA DEL COLOR, SU IMPORTANCIA E INFLUENCIA EN LA DESICION DE COMPRA DEL CONSUMIDOR: UN ENFOQUE DESDE EL SECTOR DE ALIMENTOS Y BEBIDAS EN COLOMBIA*. BOGOTÁ: UNIVERSIDAD MILITAR NUEVA GRANADA.
- Salas, R. (2004). *Redes Neuronales Artificiales*. Valparaiso: Universidad de Valparaiso.
- Vinicio Mejía-Vayas, C., & Cherres-Galarza, D. A. (2020). *Percepción de marca y comportamiento de compra del consumidor en empresas del sector comercial Tungurahua*. Digital Publisher CEIT.
- Lubinus Badillo, F. ., Rueda Hernández, C. A. ., Marconi Narváez, B. ., & Arias Trillos, Y. E. . (2021). Redes neuronales convolucionales: un modelo de Deep Learning en imágenes diagnósticas. Revisión de tema. *Revista Colombiana De radiología*, 32(3), 5591–5599.
- Muñoz Rojas, D. I. (2023). Redes neuronales del tipo transformer como herramienta de clasificación de sobreexpresión de proteína HER2 en imágenes de biopsias de cáncer gástrico.
- Gao, Z., Bird, C., & Barr, E. T. (2017). To type or not to type: Quantifying detectable bugs in JavaScript. Microsoft Research. <https://www.microsoft.com/en-us/research/publication/to-type-or-not-to-type/>
- Carmona, C. J., Pulgar-Rubio, F. J., García-Vico, A. M., González, P., & Del Jesús, M. J. (2015). Análisis descriptivo mediante aprendizaje supervisado basado en patrones emergentes. In *Proceedings of the VII Simposio Teoria y Aplicaciones de Minería de Datos* (pp. 685-694).

De Gorostiza Luengo, J. F. (2024). Detección de ratones de laboratorio mediante técnicas de visión artificial clásica y aprendizaje profundo.

Gelbukh, A. (2010). Procesamiento de lenguaje natural y sus aplicaciones. *Komputer Sapiens*, 1, 6-11.

Carrillo, J. R. V., & Báez, C. G. G. (2024). Análisis Computacional de la Expresión Génica mediante RNA-Seq en la Respuesta Aviar a la Infección por el Virus IBDV: Exploración de Métodos de Aprendizaje No Supervisados.

Ortiz Leon, D. J., & Alcalde Castañeda, M. A. (2022). Aplicativo móvil de procesamiento de imágenes usando Machine Learning para el reconocimiento de incidencias de residuos sólidos en la municipalidad de Punta Hermosa.

Conejo Gualacata, S. A. (2022). Prototipo de sistema móvil para reconocimiento de personas requeridas por organismos de seguridad y control a través de reconocimiento facial aplicando Redes Neuronales Convolucionales y Transferencia de Aprendizaje [Tesis de grado, Universidad San Francisco de Quito]. Repositorio Digital USFQ.
<http://repositorio.usfq.edu.ec/handle/23000/12192>

Tomala Tigreros, J. A. (2025). Desarrollo de una interfaz basada en algoritmos de procesamiento de lenguaje natural (NLP) para la gestión de conocimiento [Tesis de grado, Universidad Estatal Península de Santa Elena]. Repositorio Digital UPSE.
<https://repositorio.upse.edu.ec/handle/46000/12861>

Abid, A., Abdalla, A., Abid, A., Khan, D., Alfozan, A., & Zou, J. (2019).

Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild. *arXiv preprint arXiv:1906.02569*. <https://arxiv.org/abs/1906.02569>