



**OPTIMIZACIÓN DEL CONTROL DE PLAGAS CON UN SISTEMA DE GESTIÓN
ENTOMOLÓGICA**

**“OPTIMIZACION DEL CONTROL DE PLAGAS EN CULTIVOS MEDIANTE UN
SISTEMA DE GESTION DE DATOS ENTOMOLOGICOS.”**

TESIS PARA TITULACIÓN INTEGRAL

INGENIERÍA EN SISTEMAS COMPUTACIONALES

EGRESADO:

AXEL ARIEL BARAHONA MENA

DIRECTOR Y CO-DIRECTOR:

DRA. MARLENE MENDEZ MORENO

DR. FELIPE DE J. GONZALEZ RODRIGUEZ

Calkiní, Campeche, México 2025





**OPTIMIZACIÓN DEL CONTROL DE PLAGAS CON UN SISTEMA DE GESTIÓN
ENTOMOLÓGICA**

**“OPTIMIZACION DEL CONTROL DE PLAGAS EN CULTIVOS MEDIANTE UN
SISTEMA DE GESTION DE DATOS ENTOMOLOGICOS.”**

TESIS PARA TITULACIÓN INTEGRAL

INGENIERÍA EN SISTEMAS COMPUTACIONALES

EGRESADO:

AXEL ARIEL BARAHONA MENA

DIRECTOR Y CO-DIRECTOR:

DRA. MARLENE MENDEZ MORENO

DR. FELIPE DE J. GONZALEZ RODRIGUEZ

Calkiní, Campeche, México 2025





Educación
Secretaría de Educación Pública



INSTITUTO TECNOLÓGICO SUPERIOR DE CALKINÍ



Instituto Tecnológico Superior de Calkiní
Subdirección de Planeación y Desarrollo

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

Calkiní, Campeche 17 de 2025

P/03/25/661

INGENIERÍA EN SISTEMAS COMPUTACIONALES

CONSTANCIA DE NO INCONVENIENCIA

MODALIDAD: TESIS

C. AXEL ARIEL BARAHONA MENA 7266
PASANTE DE LA CARRERA DE
INGENIERÍA EN SISTEMAS COMPUTACIONALES.
PRESENTE

Con relación a lo establecido en el Reglamento de Estudios de Licenciatura, capítulo VII, fracción I, vigente en este Instituto, y una vez emitido el fallo favorable por el director DRA. MARLENE MENDEZ MORENO, y el co-director DR. FELIPE DE J. GONZALEZ RODRIGUEZ, respectivamente, este Instituto consta de no tener Inconveniencia para la realización del acto protocolario de titulación Integral de su trabajo profesional titulado: **""OPTIMIZACION DEL CONTROL DE PLAGAS EN CULTIVOS MEDIANTE UN SISTEMA DE GESTION DE DATOS ENTOMOLOGICOS""**.

ATENTAMENTE

Excelencia en Educación Tecnológica®
Sabiduría, Fortaleza de Nuestra Cultura®

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

C.c.p. Archivo



INSTITUTO TECNOLÓGICO SUPERIOR
DE CALKINÍ

"DEPARTAMENTO DE ADMINISTRACIÓN
ESCOLAR Y APOYO A
ESTUDIANTES"

CLAVE: 04MBU0013P
CALKINÍ, CAMPECHE, MEXICO.



2025
Año de
La Mujer
Indígena

Av. Ah Canul 6/N por Carretera Federal, Calkiní, Campeche,
C.P. 24000 Tel. 996-8134870,
e-mail: ee_calkin@teonm.mx | itesoam.edu.mx





Dedicatoria.

Quiero dedicarles este gran logro a aquellas personas;

A mi novia, por ser mi mayor apoyo, mi refugio y la fuerza que me impulsa a seguir adelante. Gracias por estar a mi lado en cada paso del camino, con tu amor incondicional, tus palabras de aliento y tu compañía en los momentos más desafiantes. Eres mi inspiración y mi mayor fortaleza...

A mi madre y a mi hermana, por su cariño inagotable y su fortaleza, que han sido mi inspiración constante. A mis tíos y tías, quienes con su afecto y confianza me han enseñado que nunca estoy solo en este camino. A mis abuelos y abuelas, guardianes de mi historia y raíces, cuya sabiduría y amor me guían en cada paso que doy.

A mis compañeros de cuatro patas, quienes con su compañía silenciosa y leal estuvieron conmigo en las noches más largas y en los días más difíciles, cuando sentía que no podía continuar.

A mi segunda familia, los amigos que encontré en la universidad, quienes con su apoyo, compañerismo y alegría hicieron de este viaje algo extraordinario. Gracias por compartir no solo risas, aprendizajes y sueños, sino también por ser un refugio en los momentos difíciles y una fuente constante de inspiración. Este camino fue más llevadero y enriquecedor gracias a ustedes, dejando recuerdos imborrables y la certeza de un vínculo eterno.

A mis profesores, como grupo en general, por sus enseñanzas, grandes consejos y por ser una fuente de inspiración que me impulsa a seguir adelante. Su paciencia y guía han sido faros en mi trayecto académico y personal. Y, finalmente, a mis asesores y a la junta de maestros, por su apoyo y dedicación incondicional, quienes con su experiencia y compromiso hicieron posible que este proyecto cobrara vida.

Con todo mi corazón, dedico este proyecto y los frutos de mi esfuerzo a ustedes.

Haz de tu vida un sueño, y de tu sueño una realidad

– Antoine de Saint-Exupéry



Agradecimientos.

Con el corazón lleno de gratitud, quiero dedicar estas palabras a todas las personas que hicieron posible la realización de este proyecto, cuya presencia y apoyo dejaron una huella imborrable en mi camino.

Primero, a mi novia, Marisa Rubí Ramírez Herrera, quien con su amor, paciencia y fortaleza fue mi refugio en los momentos más difíciles. Gracias por ser mi mayor apoyo y por estar siempre a mi lado, celebrando cada pequeño logro y ayudándome a superar los desafíos más grandes. Este logro también es tuyo.

También a mi madre, Antonia del Rosario Mena Che, y a mi hermana, Andrea Guadalupe Barahona Mena, por ser el pilar de mi vida. Su amor incondicional y su fe en mí han sido la luz que me guía en los días más oscuros. Ustedes son la razón por la que nunca me rendí.

Mi agradecimiento eterno a mi tía, Mirza Lucía Barahona Chávez, y a mi tío, Hugo Sánchez Espinoza, por sus sabios consejos que llegaron justo en el momento en que más los necesitaba. Gracias por ser un ejemplo de fortaleza y sabiduría en mi vida.

A mi abuelo, Juan Mena Euan, cuya vida ha sido una inspiración constante. Gracias por enseñarme, con tu ejemplo, la importancia del esfuerzo, la honestidad y la perseverancia.

No puedo olvidar a mis amigos Jordi, Fernando, Magdiel, Jesús, Martín, Jonathan, Juan y Raúl, quienes se convirtieron en mi segunda familia. Gracias por las risas, los aprendizajes y por estar ahí en cada paso del camino. Su apoyo incondicional hizo de este trayecto una experiencia inolvidable.

Un especial agradecimiento a mis maestros, José Luis Lira Turriza, Yaqueline Pech Huh y José Manuel Lira Turriza, quienes con sus enseñanzas, consejos y paciencia han dejado una huella profunda en mi formación. Gracias por compartir su conocimiento con dedicación y por ser una inspiración constante para enfrentar los desafíos académicos y personales con determinación.

Mi gratitud más profunda hacia mis asesores, Marlene Méndez Moreno y Gonzalo Miguel Quetz Aguirre, por su dedicación y guía inquebrantable. Gracias por su confianza en mi trabajo y por orientarme con sabiduría para hacer realidad este proyecto.

Igualmente, agradezco a los profesores de la junta académica, Mario Ben-Hur Chuc Armendariz y Ángel Francisco Can Cabrera, por sus valiosas aportaciones, su tiempo y su compromiso en mi formación. Su guía fue crucial para alcanzar este logro.

Finalmente, mi reconocimiento a la Fundación Ortiz Ávila, por el apoyo brindado para hacer posible este proyecto. Su contribución no solo facilitó mi camino, sino que también me motivó a dar lo mejor de mí.

A todos ustedes, mi más profundo y sincero agradecimiento. Este logro no habría sido posible sin cada uno de ustedes. Gracias por formar parte de mi historia.



Índice.

Índice Figuras.	10
Resumen.	12
Capítulo I.	14
Introducción.	14
1.2 PLANTAMIENTO	16
1.3 JUSTIFICACIÓN	17
1.4 OBJETIVOS GENERALES Y ESPECIFICOS.	18
1) Objetivo general.	18
2) Objetivos específicos	19
1.5 HIPÓTESIS.	19
1.6 ALCANCES Y LIMITACIONES.	20
Alcance del proyecto.	20
Limitaciones.	21
Capítulo II.	23
MARCO TEORICO.	23
2.1 Introduccion	23
2.2 Manejo de Plagas en la Agricultura	23
2.2.1 Impacto de las Plagas en la Producción Agrícola	24
2.2.2 Métodos de Control de Plagas	24
2.3 Trazabilidad de Datos Entomológicos	24
2.3.1 Importancia de la Trazabilidad en la Agricultura	24
2.3.2 Sistemas de Información para la Gestión de Plagas	24
2.4 Uso de Tecnologías en la Agricultura	25
2.4.1 Plataformas Web para la Gestión de Plagas	25
2.4.2 Aplicaciones de Inteligencia Artificial y Big Data en la Agricultura	25
2.5 Modelos de Toma de Decisiones en el Control de Plagas	25
2.5.1 Enfoques Basados en Datos	25
2.6 Estudios Previos sobre Sistemas Digitales para el Manejo de Plagas	26
Capítulo III.	27
Métodos y Materiales	27
3.1 METODOLOGIA DE INVESTIGACION	27



3.1.1 Enfoque metodológico	27
3.1.2 Metodología de Desarrollo	27
3.1.3 Aplicación de Scrum en el Proyecto	28
3.1.4 Recursos de Desarrollo	28
4.1 Inicio para usuarios.	30
4.1.1 Inicio	30
4.1.2 Índice de plagas para usuarios.	30
4.1.3 Plagas que existen por sección.	31
4.1.4 Información de las plagas para los usuarios.	32
4.1.5 Registro para nuevos usuarios.	34
4.1.6 Login para usuarios y administradores	35
4.1.7 Administración de la página login.	36
4.1.8 administración de plagas y aliados	37
4.2 Evaluación de la plataforma.	40
4.2.1 Prueba de Usuario	40
4.2.3 Primer acceso a la plataforma	40
4.2.4 Registro de nuevo usuario	41
4.2.5 Inicio de sesión tras el registro exitoso	42
4.2.6 Selección de cultivos por categoría	43
4.2.7 Visualización de plagas para el cultivo de maíz	44
4.2.8 Identificación de plagas en el cultivo de maíz	45
4.3 Pruebas de funcionalidades	46
4.3.1 Resultados de las Pruebas Funcionales:	46
4.3.2 Resultados de las Pruebas de Usabilidad:	46
4.3.3 Resultados de las Pruebas de Rendimiento:	47
4.3.4 Impacto en la Toma de Decisiones	47
4.4 Comparación con Métodos Tradicionales	47
4.5 Limitaciones y Desafíos	48
Capítulo IV.	49
Conclusiones y Recomendaciones.	49
4.2 Recomendaciones	50
Referencias Bibliográficas.	52



Anexos. _____	54
Anexo 1: Inicio _____	54
Funcionamiento Lógica: _____	54
Código Inicio: _____	54
Anexo 2: Navbar _____	55
Funcionamiento Lógico: _____	55
Código Navbar: _____	57
Anexo 3: Footer _____	58
Funcionamiento Lógico: _____	58
Código Footer: _____	59
Anexo 4: App.routes.ts _____	59
Funcionamiento Lógico: _____	59
Código app.routes.ts: _____	64
Anexo 5: pest-cards _____	72
Funcionamiento Lógico: _____	72
Código pest-card: _____	73
Anexo 6: Índice de plagas _____	73
Funcionamiento Lógico: _____	73
Código IndexPest: _____	74
Anexo 7: Plaga de Maiz _____	75
Funcionamiento Lógico: _____	75
Código Plaga maíz: _____	75
Anexo 8: Gusano cogollero – Ejemplo de vista _____	76
Funcionamiento Lógico: _____	76
Código PestGusanoCogollero: _____	78
Anexo 9: Índice aliados _____	79
Funcionamiento lógico: _____	79
Código IndexAllies: _____	80
Anexo 10: Chat de IA _____	81
Funcionamiento Lógico: _____	81
Código Chat IA: _____	82
Anexo 11: Login _____	84



Funcionamiento Lógico:	84
Código Login:	86
Anexo 12: Registro	88
Funcionamiento Lógico:	88
Código Registro:	90
Anexo 13: Dashboard-administrador	92
Funcionamiento Lógico:	92
Código Dashboard-Administrador:	100
Anexo 14: Chat.service.ts	111
Funcionamiento Lógico:	111
Código ChatService:	112
Anexo 15: Chat.service.spect.ts	114
Funcionamiento Lógico:	114
Código chatservicespect:	114
Código UserService:	117
Anexo 17: Plagas.service.ts	120
Funcionamiento Lógico:	120
Anexo 18:	124
Funcionamiento Lógico:	124
Código AlliesServices:	125

Índice Figuras.

Figura 1 Croquis de ubicación	<i>¡Error! Marcador no definido.</i>
Figura 2 Organigrama	<i>¡Error! Marcador no definido.</i>
Figura 3 Ilustración de metodología: inicio para usuarios	<i>¡Error! Marcador no definido.</i>
Figura 4 Ilustración de metodología: Índice de plagas	<i>¡Error! Marcador no definido.</i>
Figura 5 Ilustración de metodología: plagas por sección	<i>¡Error! Marcador no definido.</i>
Figura 6 Ilustración de metodología información de las plagas	<i>¡Error! Marcador no definido.</i>
Figura 7 Ilustración de metodología: seguimiento información de las plagas	<i>¡Error! Marcador no definido.</i>
Figura 8 Ilustración de metodología: Registro para nuevos usuarios	<i>¡Error! Marcador no definido.</i>
Figura 9 ilustración de metodología: Login usuarios y administradores	<i>¡Error! Marcador no definido.</i>
Figura 10 Ilustración de metodología: administración de Usuarios	<i>¡Error! Marcador no definido.</i>
Figura 11 Ilustración de metodología: agregar Usuarios	<i>¡Error! Marcador no definido.</i>
Figura 12 Ilustración de metodología: administración de plagas	<i>¡Error! Marcador no definido.</i>
Figura 13 Ilustración de metodología: editar plaga	<i>¡Error! Marcador no definido.</i>
Figura 14 ilustración de metodología: administración de aliados	<i>¡Error! Marcador no definido.</i>
Figura 15 ilustración diagrama de bloques de las etapas del proyecto	<i>¡Error! Marcador no definido.</i>
Figura 16 Resultado: pantalla de inicio	<i>¡Error! Marcador no definido.</i>
Figura 17 Resultado Índice de plagas	<i>¡Error! Marcador no definido.</i>
Figura 18 Resultado pantalla de plagas en maíz	<i>¡Error! Marcador no definido.</i>
Figura 19 Resultado Parte 1: Pantalla Detallada: Gusano Cogollero	<i>¡Error! Marcador no definido.</i>
Figura 21 Resultado Parte 2: Pantalla Detallada: Gusano Cogollero	<i>¡Error! Marcador no definido.</i>
Figura 20 Resultado Parte 3: Pantalla Detallada: Gusano Cogollero	<i>¡Error! Marcador no definido.</i>
Figura 22 Resultado Pantalla de Índice de aliados	<i>¡Error! Marcador no definido.</i>
Figura 23 Resultado Pantalla de Registro	<i>¡Error! Marcador no definido.</i>
Figura 24 Resultado Inicio de sesión de Usuarios	<i>¡Error! Marcador no definido.</i>
Figura 25 Resultado pantalla de inicio Usuario	<i>¡Error! Marcador no definido.</i>
Figura 26 Resultado Pantalla de chat interactivo	<i>¡Error! Marcador no definido.</i>
Figura 27 Resultado Pantalla de Tu amigo chat	<i>¡Error! Marcador no definido.</i>
Figura 28 Resultado Panel de Control (Dashboard) Administración de Usuarios	<i>¡Error! Marcador no definido.</i>
Figura 29 Resultado Agregar Usuario	<i>¡Error! Marcador no definido.</i>
Figura 30 Resultado Agregar Plaga	<i>¡Error! Marcador no definido.</i>
Figura 31 Resultado Administración de plagas	<i>¡Error! Marcador no definido.</i>
Figura 32 Resultado Agregar Aliado	<i>¡Error! Marcador no definido.</i>
Figura 33 Resultado Administracion de Aliados	<i>¡Error! Marcador no definido.</i>
Figura 34 Anexo ilustración Inicio	54
Figura 35 Anexo ilustracion Navbar	56
Figura 36 Anexo Ilustracion Navbar	57
Figura 37 Anexo ilustración footer	59
Figura 38 Anexo ilustracion parte 1 app.routes.ts	60
Figura 39 Anexo ilustracion parte 2 app.routes.ts	61
Figura 40 Anexo ilustracion parte 3 app.routes.ts	62
Figura 41 Anexo ilustracion parte 4 app.routes.ts	63



Figura 42 Anexo ilustración parte 5 app.routes.ts	64
Figura 43 Anexo ilustración pest-card.component.ts	72
Figura 44 Anexo ilustración IndexPestsGasteoropodosComponent.ts	74
Figura 45 Anexo ilustración IndexPestsGasteoropodosComponent.ts	75
Figura 46 Anexo ilustración parte1 PestsGusanoCogolleroComponent.ts	77
Figura 47 Anexo ilustración parte2 PestsGusanoCogolleroComponent.ts	77
Figura 48 Anexo ilustración IndexPestsAlliesComponent.ts	80
Figura 49 Anexo ilustración parte 1 ChatComponent.ts	81
Figura 50 Anexo ilustración parte 2 ChatComponent.ts	82
Figura 51 Anexo ilustración parte 1 LoginComponent.ts	85
Figura 52 Anexo ilustración parte 2 LoginComponent.ts	86
Figura 53 Anexo ilustración parte 1 RegisterComponent.ts	89
Figura 54 Anexo ilustración parte 2 RegisterComponent.ts	90
Figura 55 Anexo ilustración parte 1 DashboardComponent.ts	93
Figura 56 Anexo ilustración parte 2 DashboardComponent.ts	94
Figura 57 Anexo ilustración parte 3 DashboardComponent.ts	95
Figura 58 Anexo ilustración parte 4 DashboardComponent.ts	96
Figura 59 Anexo ilustración parte 5 DashboardComponent.ts	97
Figura 60 Anexo ilustración parte 6 DashboardComponent.ts	98
Figura 61 Anexo ilustración parte 7 DashboardComponent.ts	99
Figura 62 Anexo ilustración parte 8 DashboardComponent.ts	100
Figura 63 Anexo ilustración chat.service.ts	112
Figura 64 Anexo iilustre Chat.service.spect.ts	114
Figura 65 Anexo ilustracio parte 1 User.service.ts	116
Figura 66 Anexo ilustracio parte 2 User.service.ts	117
Figura 67 Anexo ilustración parte1 Plagas.service.ts	121
Figura 68 Anexo ilustración parte 2 Plagas.service.ts	121
Figura 69 Anexo ilustración parte 1 Aliados.service.ts	124
Figura 70 Anexo ilustración parte 2 Aliados.service.ts	125

Resumen.

El manejo de plagas en la agricultura es un desafío constante que afecta la productividad de los cultivos y la rentabilidad de los agricultores. En la región de Campeche, la falta de un sistema eficaz de identificación y control de plagas ha generado el uso excesivo de pesticidas químicos, aumentando los costos de producción y afectando el medio ambiente. Con el fin de optimizar la gestión de plagas, se desarrolló una plataforma de gestión de datos entomológicos, que permite a los agricultores identificar con precisión las plagas que afectan sus cultivos y recibir recomendaciones sobre los métodos de control más adecuados.

El objetivo principal de este estudio fue diseñar e implementar una herramienta tecnológica que centralizara y organizara información detallada sobre plagas agrícolas, brindando a los usuarios acceso a datos actualizados que faciliten la toma de decisiones fundamentadas. Para lograrlo, se utilizó la metodología Scrum, lo que permitió desarrollar la plataforma en sprints iterativos, asegurando mejoras continuas y una adaptación flexible a las necesidades de los usuarios.

La plataforma fue desarrollada utilizando Angular para el frontend, Node.js y Laravel para el backend, y Firebase para la base de datos en tiempo real. Entre sus principales funcionalidades destacan el registro y consulta de plagas, un mapeo interactivo de su distribución geográfica, un sistema de recomendaciones basado en datos entomológicos y un panel administrativo para la gestión de usuarios e información.

Los resultados obtenidos a través de pruebas funcionales y de usabilidad demostraron que la plataforma mejoró significativamente la identificación de plagas y la selección de estrategias de control, comparada con los métodos tradicionales. Asimismo, se observó una reducción en la dependencia de pesticidas químicos, promoviendo alternativas más sostenibles y eficaces para la protección de los cultivos. La hipótesis planteada en este estudio se



confirmó, evidenciando que los agricultores sin conocimientos especializados en entomología lograron mejorar sus decisiones gracias a la información proporcionada por el sistema.

Como conclusión, la plataforma representa una solución innovadora para la gestión de plagas en la región de Campeche, con un gran potencial de expansión a otras áreas y cultivos. Se recomienda su mejora continua mediante la integración de inteligencia artificial y análisis predictivo, así como la implementación de una versión offline para beneficiar a comunidades con acceso limitado a Internet.

Palabras clave: *Plagas agrícolas, gestión de datos entomológicos, plataforma web, sostenibilidad, tecnología aplicada a la agricultura.*

Capítulo I.

Introducción.

El control de plagas en la agricultura es un desafío constante que impacta directamente la productividad y rentabilidad de los cultivos. En la región de Campeche, esta problemática es especialmente relevante debido a la alta incidencia de plagas que afectan cultivos de importancia económica. La gestión ineficiente de estas plagas ha generado pérdidas significativas y ha incrementado la dependencia de pesticidas químicos, los cuales, además de ser costosos, pueden tener efectos negativos en la salud humana y el medio ambiente. El manejo de plagas tradicionalmente se ha basado en la experiencia empírica de los agricultores, lo que puede llevar a decisiones poco precisas y a un uso excesivo de productos químicos. En este contexto, surge la necesidad de una solución tecnológica que centralice la información sobre las plagas y brinde recomendaciones basadas en datos actualizados. La digitalización y el acceso a la información en tiempo real han demostrado ser herramientas clave en diversos sectores, incluyendo la agricultura, donde pueden contribuir a la toma de decisiones fundamentadas y mejorar la eficiencia de los procesos agrícolas.

Este proyecto tiene como objetivo principal el desarrollo de una plataforma de gestión de datos entomológicos, diseñada para optimizar el control de plagas en los cultivos de Campeche. Esta plataforma permite a los agricultores identificar plagas con mayor precisión, acceder a información detallada sobre sus características biológicas y recibir recomendaciones sobre los métodos de control más adecuados. Además, la implementación de una base de datos interactiva y un sistema de recomendaciones basado en el comportamiento de las plagas busca reducir la dependencia de pesticidas y fomentar prácticas agrícolas más sostenibles. El desarrollo de la plataforma se llevó a cabo utilizando la metodología Scrum, permitiendo un proceso iterativo y flexible para la creación del sistema. Se emplearon tecnologías como Angular, Node.js, Laravel y Firebase, garantizando una infraestructura robusta y escalable. Además, el diseño de la interfaz de usuario fue pensado para ser intuitivo y accesible,

facilitando su adopción por parte de los agricultores, incluso aquellos con poca experiencia en tecnología.

A lo largo de esta investigación, se evaluó la eficacia de la plataforma a través de pruebas funcionales, de usabilidad y de impacto en la toma de decisiones agrícolas. Los resultados obtenidos demostraron que la herramienta mejora significativamente la identificación de plagas y la selección de métodos de control, en comparación con las estrategias tradicionales utilizadas en la región.

1.2 ANTECEDENTES.

En los últimos años, diversos estudios han demostrado la importancia del uso de tecnologías digitales para optimizar el manejo de plagas agrícolas. Cañadas, del Águila y Palma (2011) desarrollaron SAVIA, un sistema web para la toma de decisiones en el control de plagas en cultivos de uva de mesa, destacando el impacto positivo de la centralización de datos y la generación de recomendaciones basadas en reglas predefinidas. De manera similar, López-Gorozabel et al. (2023) diseñaron un sistema de detección de plagas basado en Deep Learning, que emplea redes neuronales convolucionales para analizar imágenes y detectar la presencia de la mosca blanca en cultivos específicos, demostrando cómo la automatización puede mejorar la precisión en la identificación de plagas. Por su parte, Moreno Bejarano y Ponciano Camacho (2019) presentaron un sistema de información web aplicado al cultivo de palta, el cual permitió optimizar el registro y monitoreo de plagas, reduciendo los tiempos de respuesta ante infestaciones y facilitando el acceso a información actualizada. Siguiendo esta misma línea, Macas Bermeo (2022) implementó inteligencia artificial en la detección de plagas en cultivos de papa, logrando una precisión del 97% en la clasificación de infestaciones y reduciendo la dependencia de evaluaciones humanas. Finalmente, Molinares Becerra (2023) propuso una plataforma web con enfoque en trazabilidad, integrando registros de infestaciones y mapas interactivos para mejorar el monitoreo y control de plagas en la agricultura. Estos estudios evidencian que el uso de sistemas digitales en el manejo de plagas no solo optimiza la



toma de decisiones, sino que también reduce la dependencia de pesticidas y mejora la sostenibilidad agrícola. Sin embargo, la mayoría de estas soluciones están enfocadas en cultivos específicos o utilizan tecnologías avanzadas como redes neuronales, lo que resalta la necesidad de una plataforma más flexible y generalizada, capaz de centralizar datos entomológicos sobre diversas plagas agrícolas y facilitar su gestión a los productores. En este contexto, el desarrollo de un sistema de gestión de plagas basado en datos entomológicos representa una solución innovadora para la región de Campeche, permitiendo mejorar la identificación y el control de plagas mediante el acceso a información organizada y actualizada.

1.2 PLANTAMIENTO

El manejo de plagas es un desafío constante para los agricultores en la región de Campeche, especialmente en el cultivo de hortalizas. La presencia de plagas afecta la producción agrícola, disminuyendo el rendimiento y la calidad de los cultivos, lo que genera pérdidas económicas y compromete la seguridad alimentaria. Para controlar estos problemas, muchos productores siguen métodos tradicionales basados en su experiencia, lo que puede llevar a decisiones imprecisas y, en algunos casos, al uso excesivo de pesticidas.

El empleo continuo de productos químicos no solo incrementa los costos de producción, sino que también tiene consecuencias ambientales preocupantes, como la contaminación del suelo y el agua, además de contribuir a la resistencia de las plagas, lo que vuelve menos efectivos los tratamientos convencionales. Esta situación es aún más crítica en zonas rurales, donde los agricultores carecen de herramientas tecnológicas que les permitan llevar un registro detallado y confiable sobre la presencia y comportamiento de plagas en sus cultivos.

A pesar de la importancia del manejo de plagas en la agricultura, no existe en la región un sistema que facilite la centralización de datos entomológicos, lo que dificulta la consulta y el análisis de información clave sobre los ciclos de vida, patrones de comportamiento y distribución de plagas. Esta falta de un sistema adecuado limita la capacidad de los agricultores



para evaluar con precisión los riesgos en sus cultivos y tomar decisiones fundamentadas sobre los métodos de control más adecuados.

En respuesta a esta problemática, se propone el desarrollo de una plataforma web para la gestión de datos entomológicos. Esta plataforma permitirá a los agricultores registrar, consultar y analizar información sobre plagas agrícolas de manera estructurada, facilitando la toma de decisiones mediante el acceso a datos actualizados y específicos sobre las especies presentes en la región, su comportamiento y su impacto en los cultivos. A través de este sistema, los agricultores podrán realizar un seguimiento detallado de las infestaciones y los métodos de control utilizados, promoviendo el uso de estrategias más eficientes y sostenibles para la protección de los cultivos.

La implementación de esta plataforma busca mejorar el acceso a información confiable, reducir la dependencia de métodos de control ineficaces y generar prácticas agrícolas más responsables. Se espera que la plataforma no solo optimice el manejo de plagas, sino que también contribuya a una agricultura más sostenible y segura en la región de Campeche, al disminuir el uso excesivo de pesticidas y facilitar un enfoque basado en datos para el control de plagas.

1.3 JUSTIFICACIÓN

En la agricultura, el manejo eficiente de plagas es esencial para garantizar la producción y calidad de los cultivos, ya que su presencia puede causar daños severos y pérdidas económicas. En la región de Campeche, el control de plagas sigue dependiendo de métodos tradicionales y decisiones basadas en experiencia empírica, lo que dificulta la precisión en la identificación de plagas y la elección de estrategias adecuadas. Como consecuencia, muchos productores recurren al uso excesivo de pesticidas, lo que no solo incrementa los costos de producción, sino que también afecta el medio ambiente al contaminar suelos y fuentes de agua, además de favorecer la resistencia de las plagas.



A pesar de la importancia de esta problemática, no existe una herramienta tecnológica accesible que permita centralizar información sobre plagas agrícolas y proporcionar a los agricultores una referencia confiable para su manejo. En este contexto, se plantea la creación de un sistema digital que facilite la recopilación y consulta de información entomológica. A través de esta herramienta, los productores podrán acceder a datos detallados sobre las plagas que afectan sus cultivos, conocer sus características biológicas y los métodos de control más adecuados, asegurando así un manejo más informado y preciso.

El sistema también brindará a los agricultores y expertos una visión a largo plazo del comportamiento de las plagas, permitiendo identificar tendencias y aplicar estrategias de manejo más efectivas y sostenibles. La plataforma contribuirá a la trazabilidad de datos en la agricultura, lo que permitirá registrar el historial de infestaciones y los métodos de control utilizados. Esta función ayudará a reducir la dependencia de pesticidas y a promover prácticas agrícolas más sostenibles al ofrecer soluciones basadas en datos y hechos científicos.

El desarrollo de esta plataforma responde a la necesidad urgente de incorporar herramientas tecnológicas en el sector agrícola de Campeche. Facilitará el acceso a información confiable y ofrecerá un soporte técnico robusto para una toma de decisiones más efectiva y basada en evidencia. Además, al fomentar un uso más racional de los recursos y reducir el impacto ambiental causado por el uso indiscriminado de pesticidas, esta solución tiene el potencial de beneficiar tanto a los agricultores como al medio ambiente, contribuyendo a una agricultura más responsable y sostenible en la región.

1.4 OBJETIVOS GENERALES Y ESPECIFICOS.

1) Objetivo general.

Desarrollar una plataforma web de gestión de datos entomológicos que facilite la identificación y control de plagas agrícolas en la región de Campeche, con el fin de optimizar las decisiones en el manejo de cultivos y reducir la dependencia de pesticidas.

2) Objetivos específicos

- 1. diseñar e implementar una base de datos entomológica que permita almacenar información detallada sobre plagas y especies beneficiosas, facilitando la consulta rápida y la toma de decisiones efectivas en la agricultura de la región de Campeche.*
- 2. Desarrollar un módulo de análisis de datos que genere recomendaciones específicas para el control de plagas, utilizando como base los datos almacenados en la plataforma y las características particulares de cada cultivo, para mejorar la precisión en las estrategias de manejo.*
- 3. Diseñar una interfaz accesible y funcional que permita a los agricultores, sin importar su nivel de conocimiento tecnológico, acceder a la información de manera sencilla, promoviendo la adopción de la plataforma y optimizando la toma de decisiones en el manejo de plagas.*
- 4. Establecer un proceso de actualización continua de la base de datos, permitiendo la incorporación manual de nueva información sobre plagas y métodos de control, asegurando que las recomendaciones proporcionadas sean siempre relevantes y adaptadas a las condiciones agrícolas cambiantes.*

1.5 HIPÓTESIS.

La implementación de una plataforma de gestión de datos entomológicos proveerá de herramientas a los agricultores de la región de Campeche para que identifiquen con mayor precisión las plagas agrícolas que afectan sus cultivos, incluso sin poseer conocimientos especializados en entomología. Al centralizar información clave sobre las características biológicas, ciclos de vida y patrones de comportamiento de diversas especies, la plataforma facilitará la toma de decisiones sobre métodos de control adecuados. Esto contribuirá a reducir la dependencia de pesticidas químicos, promoviendo prácticas agrícolas más sostenibles en la

protección de los cultivos y reemplazando estrategias tradicionales que han demostrado ser menos efectivas o más costosas.

1.6 ALCANCES Y LIMITACIONES.

Alcance del proyecto.

Aplicación en el control de plagas agrícolas en la región de Campeche:

Este estudio evalúa el impacto y la viabilidad de una plataforma web para la gestión de datos entomológicos, centrada en el control de plagas agrícolas en la región de Campeche. La investigación se enfoca en analizar cómo esta herramienta permitirá a los agricultores identificar plagas y acceder a información detallada sobre sus características biológicas, ciclos de vida y patrones de comportamiento, mejorando la toma de decisiones sobre el manejo de plagas.

Base de datos sobre plagas y entomología beneficiosa:

El estudio aborda la creación de una base de datos entomológica que almacene información detallada sobre plagas y organismos beneficiosos. Esta base de datos permitirá a los agricultores consultar datos actualizados que faciliten la toma de decisiones informadas y fundamentadas en el manejo de plagas agrícolas.

Módulo de análisis para recomendaciones específicas:

El análisis de datos entomológicos dentro de la plataforma generará recomendaciones fundamentadas sobre métodos de control de plagas. Estas sugerencias se basarán en información sobre las especies presentes, sus ciclos de vida y patrones de comportamiento, lo que proporcionará un recurso valioso para el manejo de plagas de manera más eficiente y sostenible.

Accesibilidad y facilidad de consulta para los agricultores:

Uno de los enfoques del estudio es evaluar cómo una plataforma con una interfaz intuitiva y accesible puede facilitar la consulta de información clave sobre el control de plagas. Esto permitirá a los agricultores acceder a los datos de manera sencilla y rápida, mejorando la toma de decisiones relacionadas con el uso de pesticidas y la implementación de estrategias de control más efectivas.

 Actualización continua de la base de datos:

El estudio considera la importancia de mantener la base de datos con información constantemente actualizada sobre plagas y sus características. Esto garantizará que las recomendaciones proporcionadas a los agricultores sean precisas y estén alineadas con las condiciones agrícolas cambiantes de la región de Campeche.

Limitaciones.

 Dependencia de la conectividad a Internet:

La plataforma web depende de la disponibilidad de conexión a Internet. En zonas rurales de Campeche, donde el acceso a Internet puede ser limitado, esto podría restringir el uso de la plataforma, afectando la sincronización de datos y la actualización en tiempo real del mapeo interactivo de plagas.

 Capacitación del usuario:

Aunque la plataforma estará diseñada para ser intuitiva, se necesitará un proceso de capacitación adicional para que los agricultores puedan utilizar eficazmente las funcionalidades del sistema, como la entrada de datos y la consulta de información. Esto podría representar un desafío, especialmente en áreas rurales con limitados recursos para capacitación tecnológica.

 Costos de implementación y mantenimiento:

La implementación inicial y el mantenimiento continuo del sistema podrían representar una carga económica para los pequeños agricultores. Para



garantizar la adopción y el uso generalizado de la plataforma, será necesario desarrollar un modelo de precios accesible que facilite su implementación y mantenimiento sin poner en riesgo la viabilidad económica de los productores.

 Limitación en la automatización de actualizaciones:

Aunque la base de datos se actualizará manualmente con nueva información sobre plagas y métodos de control, esta dependencia de actualizaciones manuales podría generar demoras en la incorporación de nueva información relevante, afectando la capacidad de los agricultores para recibir datos completamente actualizados en tiempo real.

 Cobertura limitada al área de Campeche:

El sistema está diseñado específicamente para la región de Campeche, lo que limita su aplicación a otras áreas o contextos agrícolas distintos. Esto implica que, aunque el modelo puede ser replicable, no será aplicable en otras regiones sin modificaciones adicionales que consideren los contextos y especies particulares de cada zona.

Capítulo II.

MARCO TEORICO.

2.1 Introduccion

El desarrollo e implementación de una plataforma de gestión de datos entomológicos tiene el potencial de revolucionar el manejo de plagas en la agricultura, especialmente en áreas rurales donde el acceso a herramientas tecnológicas es limitado. Esta plataforma permitirá a personas sin conocimientos previos en agricultura identificar con mayor precisión las plagas agrícolas que afectan sus cultivos, al centralizar información clave sobre las características biológicas, ciclos de vida y patrones de comportamiento de diversas especies.

La plataforma no solo facilitará la identificación de plagas, sino que también proporcionará recomendaciones fundamentadas sobre métodos de control adecuados. Este enfoque permitirá a los agricultores tomar decisiones más informadas, reduciendo la dependencia de métodos tradicionales y del uso excesivo de pesticidas, lo que a su vez disminuirá los costos de producción y los impactos negativos sobre el medio ambiente.

Se espera que el uso de esta herramienta contribuya a una mejor protección de los cultivos, promoviendo prácticas agrícolas más sostenibles, eficaces y adaptadas a las condiciones específicas de la región de Campeche.

El desarrollo de esta plataforma web para la gestión de datos entomológicos requiere un marco teórico sólido que respalde los fundamentos científicos y tecnológicos involucrados en su implementación. Este capítulo aborda conceptos clave como el manejo de plagas en la agricultura, la trazabilidad de datos entomológicos, el uso de plataformas tecnológicas en la agricultura y los modelos de toma de decisiones en el control de plagas. Además, se revisan estudios previos sobre sistemas digitales aplicados a la identificación y control de plagas agrícolas, con el fin de establecer una base sólida para el desarrollo y la implementación exitosa de la plataforma.

2.2 Manejo de Plagas en la Agricultura

2.2.1 Impacto de las Plagas en la Producción Agrícola

Las plagas representan una de las principales amenazas para la producción agrícola, afectando el rendimiento de los cultivos y generando pérdidas económicas significativas. Según la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO), se estima que hasta un 40% de la producción agrícola mundial se pierde anualmente debido a plagas y enfermedades. En la región de Campeche, el manejo de plagas es un desafío constante para los agricultores, quienes en muchas ocasiones recurren a métodos tradicionales basados en su experiencia empírica, lo que puede derivar en una aplicación ineficiente de medidas de control.

2.2.2 Métodos de Control de Plagas

El control de plagas en la agricultura puede clasificarse en tres enfoques principales:

- *Control químico: Uso de pesticidas para eliminar plagas. Aunque es efectivo a corto plazo, su uso indiscriminado genera resistencia en las plagas y afecta el equilibrio ecológico.*
- *Control biológico: Introducción de depredadores naturales de las plagas para regular su población de manera sostenible.*
- *Control cultural y físico: Prácticas como la rotación de cultivos, el manejo del suelo y el uso de trampas para minimizar la incidencia de plagas.*

La integración de estos métodos en un sistema de gestión de datos permite una aplicación más precisa y eficiente, reduciendo los impactos negativos del uso excesivo de pesticidas y mejorando la sostenibilidad agrícola.

2.3 Trazabilidad de Datos Entomológicos

2.3.1 Importancia de la Trazabilidad en la Agricultura

La trazabilidad de datos entomológicos es fundamental para la gestión eficiente de plagas. Consiste en el registro y seguimiento de información sobre la presencia y comportamiento de plagas en los cultivos, permitiendo a los agricultores tomar decisiones basadas en evidencia.

2.3.2 Sistemas de Información para la Gestión de Plagas

El uso de bases de datos y plataformas digitales facilita el almacenamiento y consulta de información sobre plagas, permitiendo la identificación de patrones de infestación y la aplicación de estrategias de control más efectivas. Ejemplos de sistemas similares incluyen el desarrollo de plataformas como SAVIA y sistemas de detección basados en inteligencia artificial, los cuales han demostrado ser efectivos en la mejora de la identificación de plagas y la optimización de estrategias de manejo.

2.4 Uso de Tecnologías en la Agricultura

2.4.1 Plataformas Web para la Gestión de Plagas

Las plataformas digitales han revolucionado la manera en que los agricultores gestionan sus cultivos. Herramientas basadas en bases de datos y sistemas de monitoreo han permitido la recopilación y análisis de grandes volúmenes de información en tiempo real, optimizando la toma de decisiones en el campo. La plataforma desarrollada en este proyecto se basa en esta premisa, proporcionando a los agricultores acceso a información detallada sobre plagas de manera sencilla y accesible.

2.4.2 Aplicaciones de Inteligencia Artificial y Big Data en la Agricultura

El uso de inteligencia artificial y Big Data en la agricultura ha permitido el desarrollo de sistemas capaces de analizar patrones y predecir brotes de plagas con gran precisión. Aunque la plataforma propuesta emplea un chat bot para la ayuda de nuevos conocimientos acerca de los insectos para conocer más métodos de combate orgánicos, la integración de estas tecnologías en el futuro podría potenciar aún más su capacidad de predicción y gestión.

2.5 Modelos de Toma de Decisiones en el Control de Plagas

2.5.1 Enfoques Basados en Datos

Los modelos de toma de decisiones en la agricultura han evolucionado gracias a la incorporación de herramientas digitales que permiten la recopilación y análisis de datos en tiempo real. Entre los enfoques más utilizados se encuentran:

- ✚ *Modelos predictivos: Utilizan datos históricos y algoritmos de aprendizaje automático para anticipar la aparición de plagas.*
- ✚ *Sistemas de apoyo a la decisión: Proporcionan recomendaciones específicas basadas en información recopilada y procesada en tiempo real.*
- ✚ *Plataformas interactivas: Facilitan la consulta y visualización de información relevante para la toma de decisiones agrícolas.*

El desarrollo de una plataforma de gestión de datos entomológicos se fundamenta en estos enfoques, permitiendo a los agricultores acceder a información relevante para mejorar el manejo de plagas en sus cultivos.

2.6 Estudios Previos sobre Sistemas Digitales para el Manejo de Plagas

Diversas investigaciones han explorado el impacto del uso de plataformas digitales en la gestión de plagas. Entre los estudios más relevantes se encuentran:

- *SAVIA: Un sistema web desarrollado para la toma de decisiones en el control de plagas en cultivos de uva de mesa, destacando la importancia de la centralización de datos.*
- *Sistema de detección de plagas basado en Deep Learning: Utiliza redes neuronales convolucionales para identificar la presencia de plagas en cultivos mediante análisis de imágenes.*
- *Sistemas de información aplicados a la agricultura: Han demostrado mejorar la rapidez en la respuesta ante infestaciones y facilitar el acceso a información actualizada para los agricultores.*

Estos estudios refuerzan la importancia de desarrollar herramientas tecnológicas accesibles para mejorar la gestión de plagas, lo que respalda la relevancia del presente proyecto.

Capítulo III.

Métodos y Materiales

3.1 METODOLOGIA DE INVESTIGACION

3.1.1 Enfoque metodológico

Este proyecto sigue un enfoque de investigación aplicada, ya que busca desarrollar una solución tecnológica para resolver una problemática específica en la agricultura: la identificación y control de plagas en la región de Campeche. La investigación tiene un carácter exploratorio y descriptivo, ya que analiza el contexto actual del manejo de plagas y describe cómo la implementación de una plataforma web puede facilitar la toma de decisiones en este ámbito.

3.1.2 Metodología de Desarrollo

Para la implementación de este proyecto, se adoptó una metodología ágil basada en Scrum, la cual permitió organizar las actividades en ciclos iterativos y entregas incrementales denominadas sprints. Este enfoque favoreció la flexibilidad del desarrollo, permitiendo ajustes en función de la retroalimentación recibida durante el proceso.

Las fases del desarrollo fueron las siguientes:

- 1. Análisis de requisitos: Se recopilaron las necesidades de los agricultores y se definieron las funcionalidades clave del sistema, como el registro de plagas, generación de recomendaciones y mapeo interactivo.*
- 2. Diseño de la plataforma: Se realizó la maquetación del sistema utilizando Canvast, asegurando una interfaz accesible y fácil de usar para los agricultores.*
- 3. Desarrollo: Se implementó el frontend con Angular, el backend con Node.js y Laravel, y Firebase como base de datos en tiempo real.*
- 4. Pruebas: Se realizaron pruebas de funcionalidad y usabilidad con Postman y en un entorno local, verificando la correcta comunicación entre el frontend y el backend.*

5. *Despliegue: Se preparó la plataforma para su implementación, permitiendo su uso por parte de los agricultores en la región de Campeche.*

3.1.3 Aplicación de Scrum en el Proyecto

La metodología Scrum fue seleccionada debido a su capacidad para abordar proyectos tecnológicos de manera colaborativa y adaptable. Este enfoque permitió dividir el desarrollo en sprints estructurados, asegurando entregas incrementales y evaluación continua del progreso.

Principios de Scrum aplicados:

-  *Transparencia: Toda la información del proyecto se mantuvo accesible para los involucrados, garantizando una comunicación efectiva.*
-  *Inspección: Cada sprint culminó con una revisión para evaluar avances y detectar posibles mejoras.*
-  *Adaptación: Se realizaron ajustes en función de la retroalimentación obtenida en cada fase del desarrollo.*

Etapas en la implementación de Scrum:

-  **Product Backlog:** *Recopilación y priorización de funcionalidades clave del sistema.*
-  **Planificación de Sprint:** *Definición de objetivos y tareas específicas para cada iteración del desarrollo.*
-  **Desarrollo:** *Implementación de los módulos según las prioridades establecidas.*
-  **Revisión del Sprint:** *Presentación de avances y ajustes con base en la retroalimentación obtenida.*
-  **Retroalimentación y Adaptación:** *Incorporación de mejoras en los siguientes ciclos de desarrollo.*

3.1.4 Recursos de Desarrollo

Para la implementación del proyecto, se utilizaron diversas tecnologías que permitieron desarrollar una plataforma robusta y escalable.

Lenguajes y Frameworks:

-  *Angular (v13): Para el desarrollo del frontend y la visualización de datos entomológicos.*
-  *Node.js (v16.13.0): Para la gestión del backend y la conexión con la base de datos.*
-  *Laravel (v9): Para la creación de API y administración de operaciones en el servidor.*
-  *Firebase (SDK 9): Para la gestión de datos en tiempo real y la sincronización con usuarios.*
-  *Herramientas de Prueba y Desarrollo:*
-  *Postman: Para la validación de API y pruebas de comunicación entre el frontend y el backend.*
-  *Canvast: Para la maquetación de la interfaz de usuario.*
-  *Imágenes generadas por IA: Para asegurar originalidad y evitar problemas de copyright.*

Equipo de Desarrollo Utilizado:

-  *Computadora: Lenovo Ideapad Slim 3*
-  *Procesador: AMD Ryzen 7530*
-  *RAM: 8 GB*
-  *Almacenamiento: 512 GB SSD*
-  *Sistema Operativo: Windows 10 Pro (v21H1)*

Capítulo IV.

Resultados y Discusión.

4.1 Inicio para usuarios.

4.1.1 Inicio

En esta pantalla de bienvenida, los usuarios son introducidos a la enciclopedia de entomología agrícola. Aquí encuentran una breve explicación sobre el propósito de la plataforma, que incluye información detallada sobre plagas dañinas y entomofauna beneficiosa para el cultivo. Además, se invita a los visitantes a explorar los recursos disponibles, descubrir estrategias de control y mantenerse al día con las últimas novedades del sector agrícola.

La interfaz presenta un diseño amigable, con un menú de navegación en la parte superior para facilitar el acceso a distintas secciones como “Plagas”, “Aliados”, “Blog”, y opciones de inicio de sesión y registro para nuevos usuarios. La barra inferior contiene información de derechos reservados, añadiendo un toque profesional y organizado a la plataforma.

Figura 3

Ilustración de metodología: inicio para usuarios



4.1.2 Índice de plagas para usuarios.

Esta pantalla muestra una serie de tarjetas organizadas que agrupan las plagas agrícolas más comunes, clasificadas por tipo de daño o características. Cada tarjeta incluye una imagen representativa de la plaga y una breve etiqueta, permitiendo a los usuarios

identificar fácilmente categorías como "Maiz," "Tomate," "Calabaza y Papaya,". Esta disposición facilita una exploración visual y estructurada de las plagas, ayudando a los usuarios a acceder rápidamente a la información que necesitan.

Figura 4

Ilustración de metodología: Índice de plagas

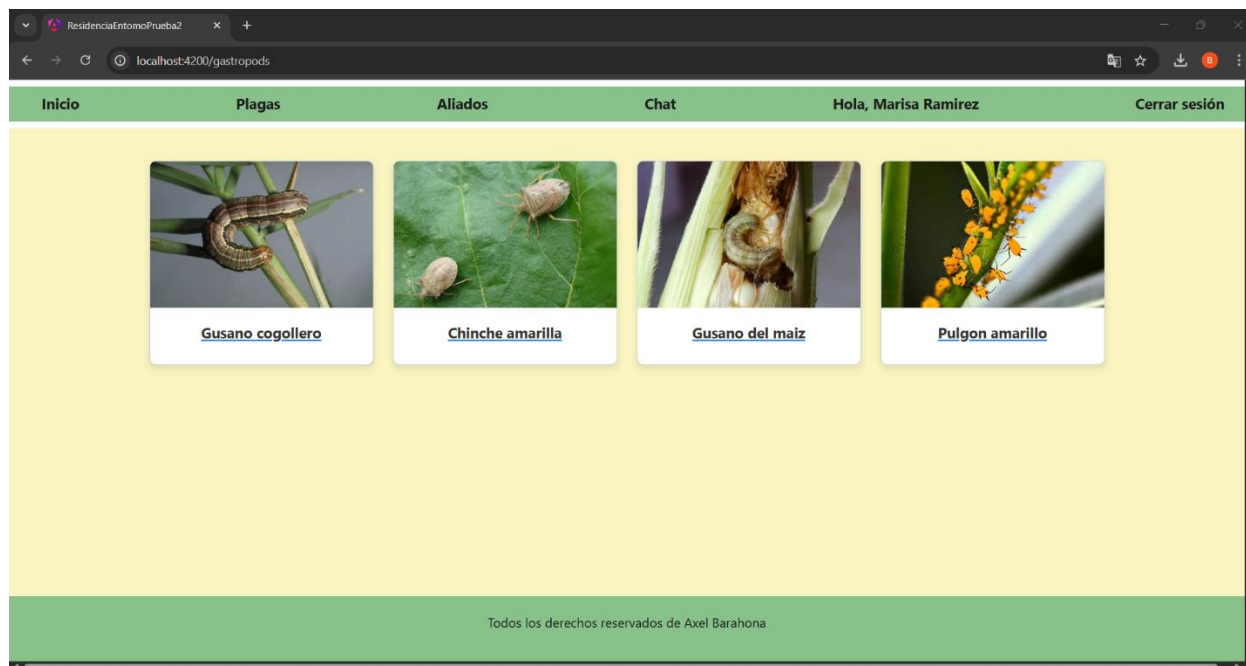


4.1.3 Plagas que existen por sección.

En esta sección, los usuarios pueden profundizar en las plagas específicas dentro de cada familia. Después de seleccionar una categoría en el índice general de plagas, esta pantalla despliega una serie de tarjetas dedicadas a plagas particulares, como el "Gusano cogollero," la

Figura 5

Ilustración de metodología: plagas por sección



4.1.4 Información de las plagas para los usuarios.

En esta sección, los usuarios pueden acceder a detalles exhaustivos sobre plagas específicas, como el "Gusano cogollero" (*Spodoptera frugiperda*). La pantalla se organiza en secciones informativas que incluyen una descripción general, características físicas, daños ocasionados y métodos de combate.

La Descripción proporciona un contexto sobre el impacto de la plaga en diversos cultivos, especialmente en el maíz, resaltando su capacidad de adaptación y resistencia a los pesticidas. En Características, se detallan aspectos físicos que facilitan su identificación, como la coloración y la estructura corporal. La sección de Daños explica los efectos nocivos de la plaga en los cultivos, incluyendo cómo afecta el crecimiento y la resistencia de las plantas.

Este diseño ofrece a los usuarios una guía integral, facilitando la comprensión y manejo de plagas agrícolas.

Figura 6
Ilustración de metodología información de las plagas

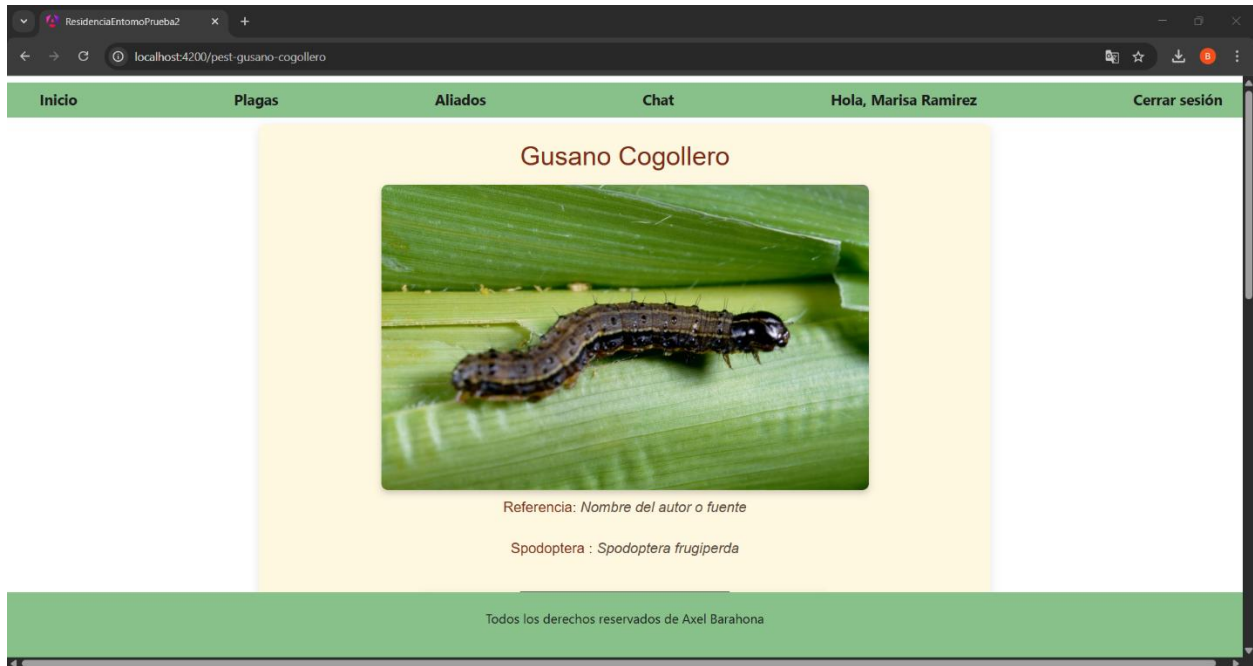
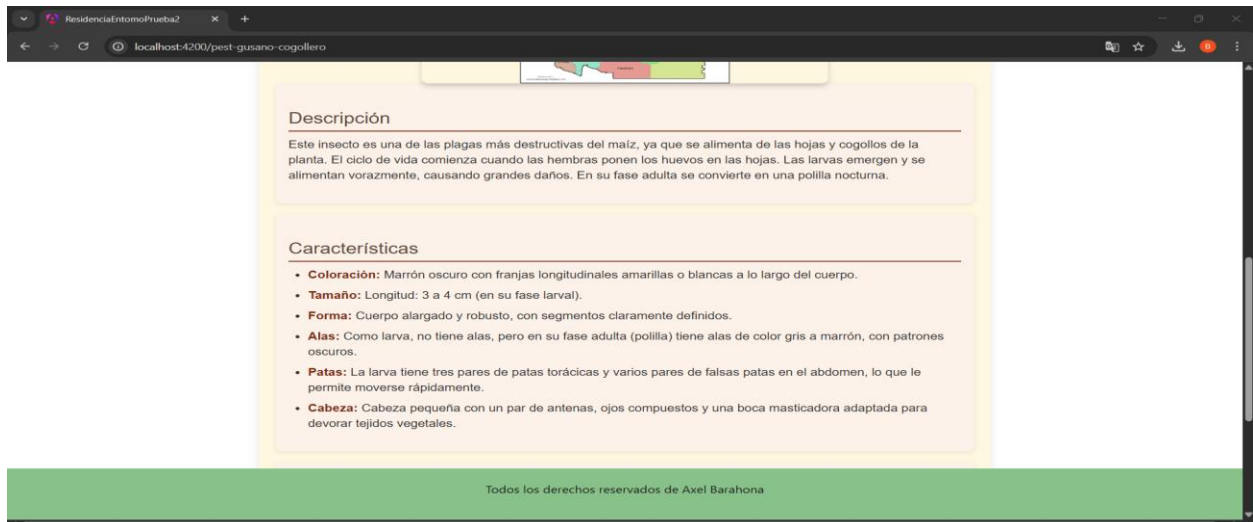
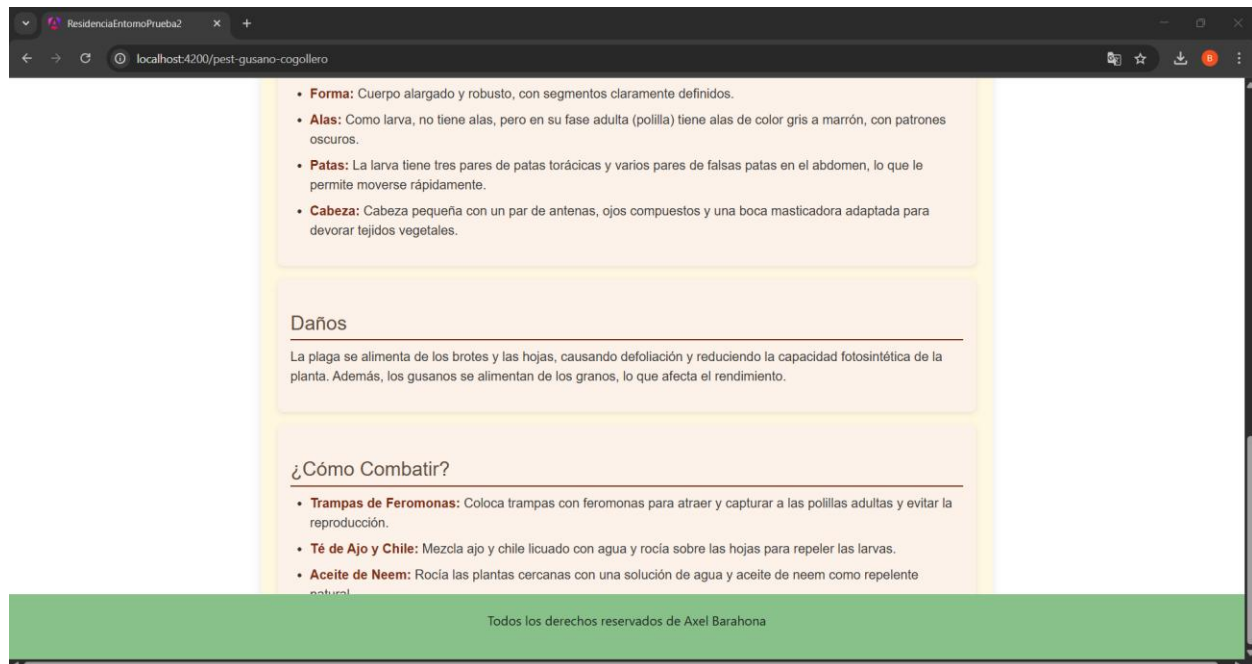


Ilustración de metodología: seguimiento información de las plagas





4.1.5 Registro para nuevos usuarios.

Esta sección permite a los nuevos usuarios registrarse en la plataforma de entomología.

El formulario de registro solicita datos básicos como el nombre completo, correo electrónico, y la creación de una contraseña, la cual debe ser confirmada para garantizar su correcta introducción. Esta estructura intuitiva y sencilla facilita el proceso de creación de cuentas, ofreciendo un acceso rápido y seguro a los recursos de la plataforma una vez completado el registro.

El diseño mantiene la uniformidad visual con el resto de la página, asegurando que el proceso de registro sea claro y accesible.

Ilustración de metodología: Registro para nuevos usuarios



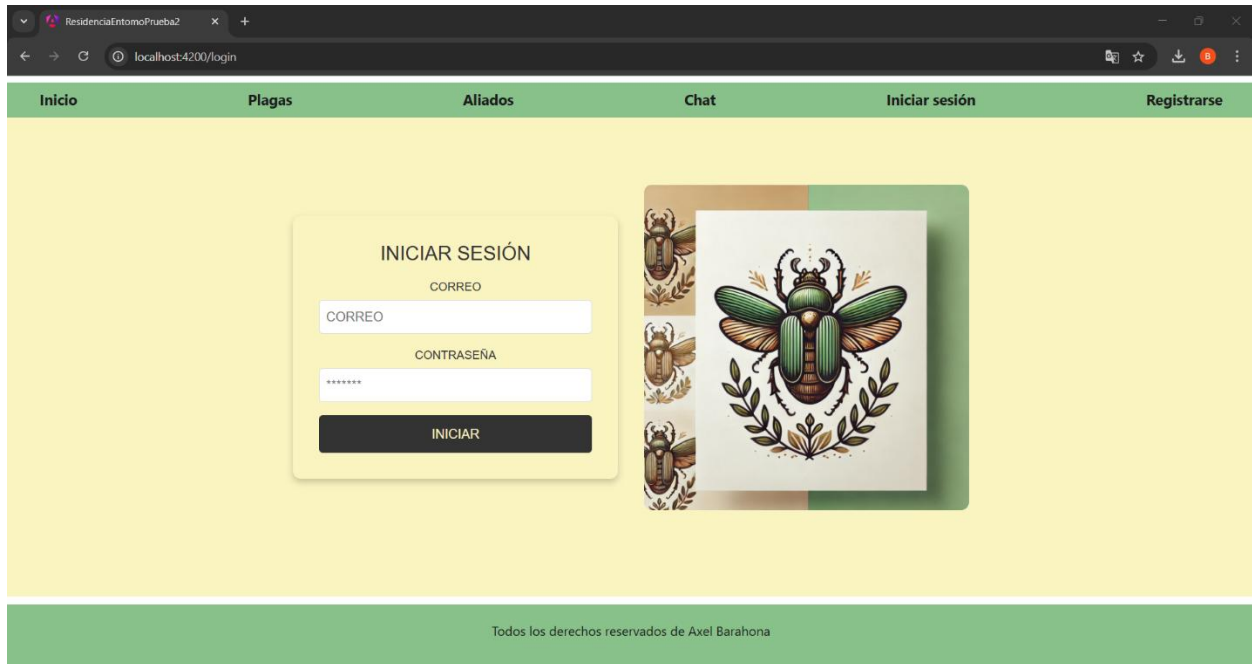
4.1.6 Login para usuarios y administradores

Esta pantalla permite el inicio de sesión tanto para usuarios regulares como para administradores de la plataforma. Los usuarios deben ingresar su correo electrónico y contraseña en los campos correspondientes y presionar el botón "Iniciar" para acceder a sus cuentas.

Dependiendo del tipo de cuenta, la plataforma redirige a los usuarios a diferentes secciones. Si la cuenta es de un usuario, se le enviará directamente a la pantalla de inicio de la plataforma (Home), donde podrá acceder a los recursos y la información general. Sin embargo, si la cuenta es de un administrador, el sistema lo redirige al panel de control (Dashboard), donde puede gestionar y supervisar el contenido y la administración de la página.

Este sistema de acceso diferenciado asegura que cada tipo de usuario tenga una experiencia personalizada acorde a sus permisos dentro de la plataforma.

Figura 9
ilustración de metodología: Login usuarios y administradores



4.1.7 Administración de la página login.

Esta pantalla en el panel de control permite a los administradores gestionar las cuentas de todos los usuarios registrados en la plataforma. La sección presenta una tabla con el nombre, correo electrónico, y rol de cada usuario, junto con opciones de acciones específicas. Los administradores pueden cambiar el rol de un usuario entre "Usuario" y "Administrador" utilizando un menú desplegable en la columna de roles. Además, existe una opción de "Eliminar" para cada usuario, facilitando la gestión y mantenimiento de la base de datos de usuarios.

asegurar que solo usuarios autorizados tengan acceso a los recursos y herramientas avanzadas, manteniendo la seguridad y organización de la plataforma.

Figura 10

Ilustración de metodología: administración de Usuarios

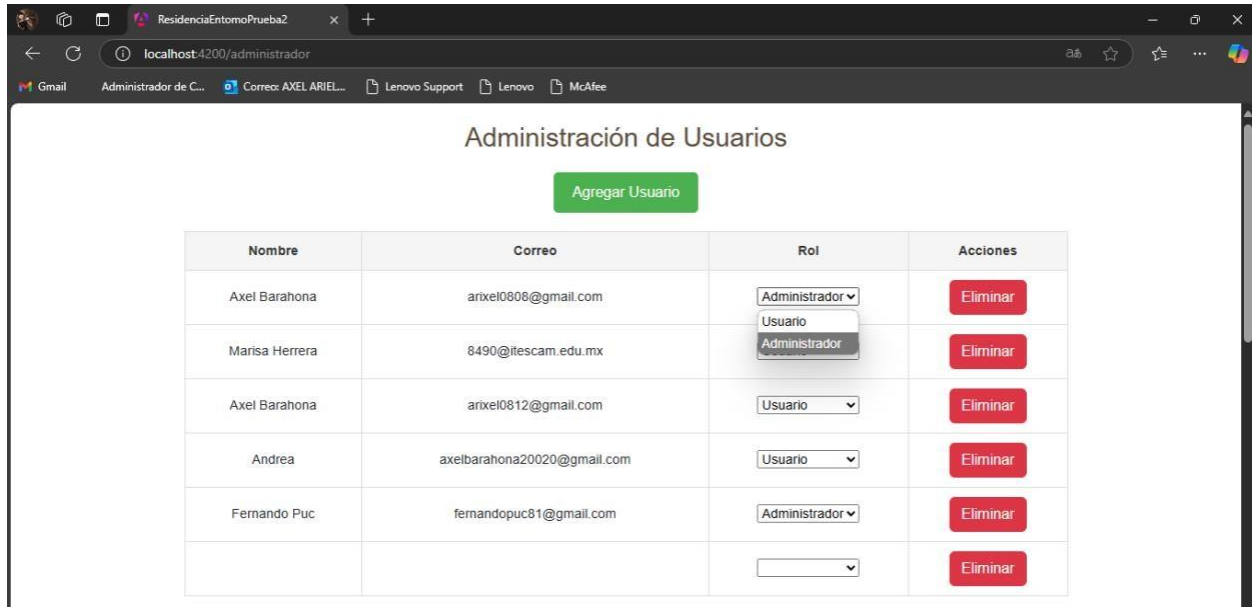
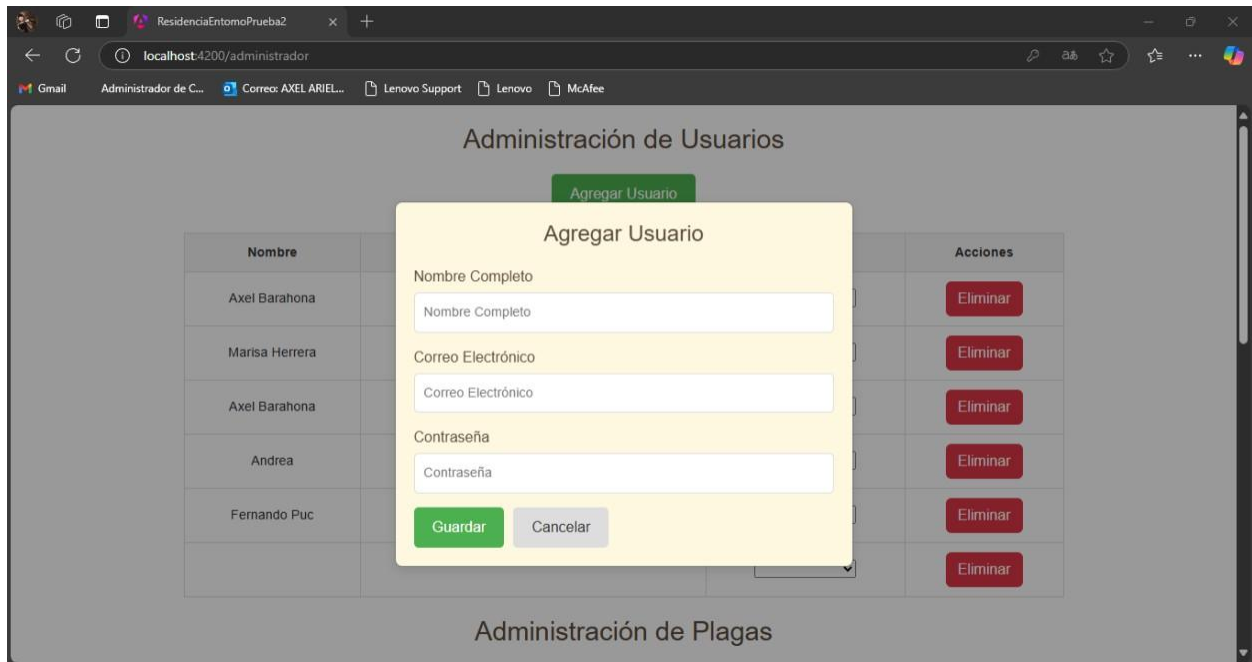


Figura 11
Ilustración de metodología: agregar Usuarios



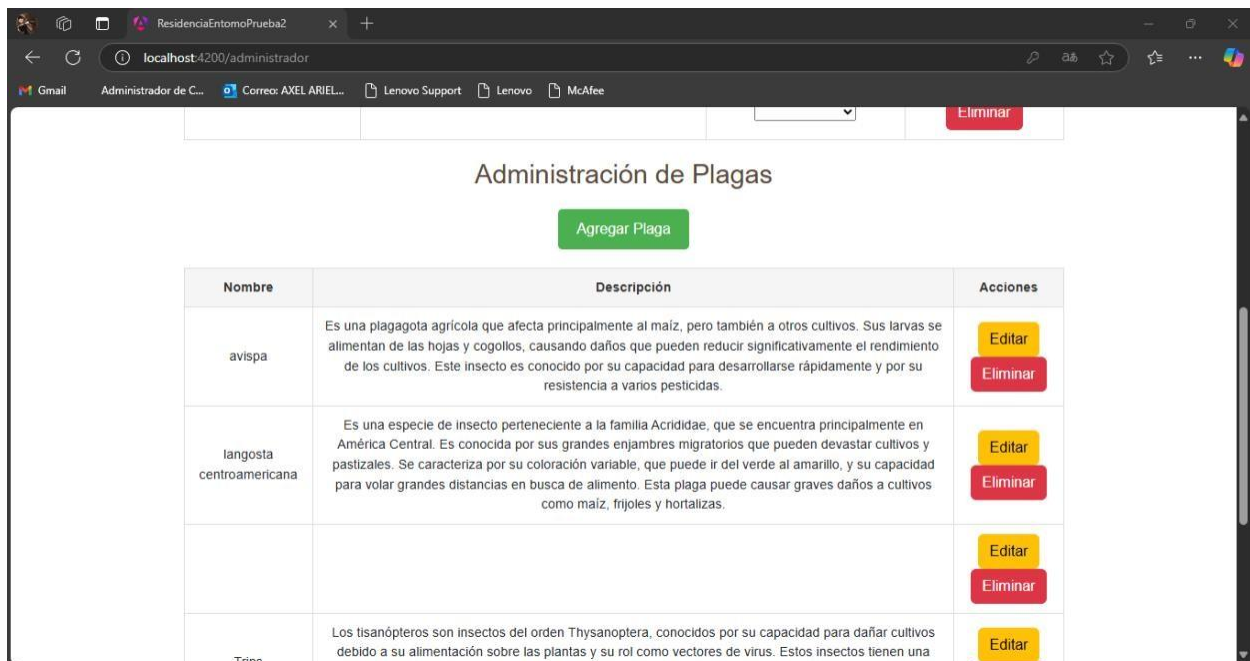
4.1.8 administración de plagas y aliados

En esta sección del panel de administración, los administradores tienen la capacidad de gestionar tanto las plagas como los aliados biológicos dentro de la plataforma. La pantalla muestra una tabla organizada con la lista de plagas, que incluye el nombre de cada plaga, una breve descripción y opciones de edición y eliminación. El botón "Editar" permite acceder a un formulario detallado para modificar la información de la plaga, incluyendo características, daños, y métodos de combate, asegurando que la información esté siempre actualizada y sea relevante para los usuarios.

Este sistema de administración integral permite mantener una base de datos precisa y organizada de las plagas y aliados, mejorando la funcionalidad de la plataforma y facilitando el acceso a información actualizada y confiable para los usuarios.

Figura 12

Ilustración de metodología: administración de plagas



ResidenciaEntomoPrueba2

localhost:4200/administrador

Gmail Administrador de C... Correos AXEL ARIEL... Lenovo Support Lenovo McAfee

Eliminar

Administración de Plagas

Agregar Plaga

Nombre	Descripción	Acciones
avispa	Es una plagagota agrícola que afecta principalmente al maíz, pero también a otros cultivos. Sus larvas se alimentan de las hojas y cogollos, causando daños que pueden reducir significativamente el rendimiento de los cultivos. Este insecto es conocido por su capacidad para desarrollarse rápidamente y por su resistencia a varios pesticidas.	<button>Editar</button> <button>Eliminar</button>
langosta centroamericana	Es una especie de insecto perteneciente a la familia Acrididae, que se encuentra principalmente en América Central. Es conocida por sus grandes enjambres migratorios que pueden devastar cultivos y pastizales. Se caracteriza por su coloración variable, que puede ir del verde al amarillo, y su capacidad para volar grandes distancias en busca de alimento. Esta plaga puede causar graves daños a cultivos como maíz, frijoles y hortalizas.	<button>Editar</button> <button>Eliminar</button>
		<button>Editar</button> <button>Eliminar</button>
Trips	Los tisanópteros son insectos del orden Thysanoptera, conocidos por su capacidad para dañar cultivos debido a su alimentación sobre las plantas y su rol como vectores de virus. Estos insectos tienen una	<button>Editar</button>

Figura 13

Ilustración de metodología: editar plaga

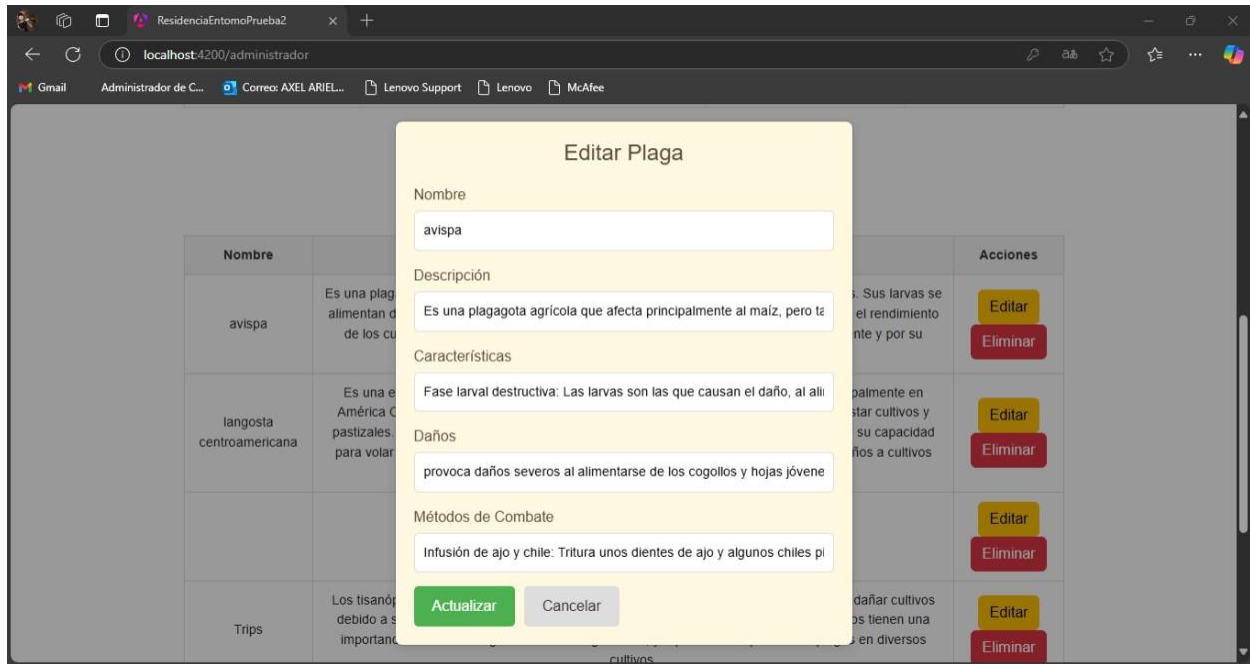
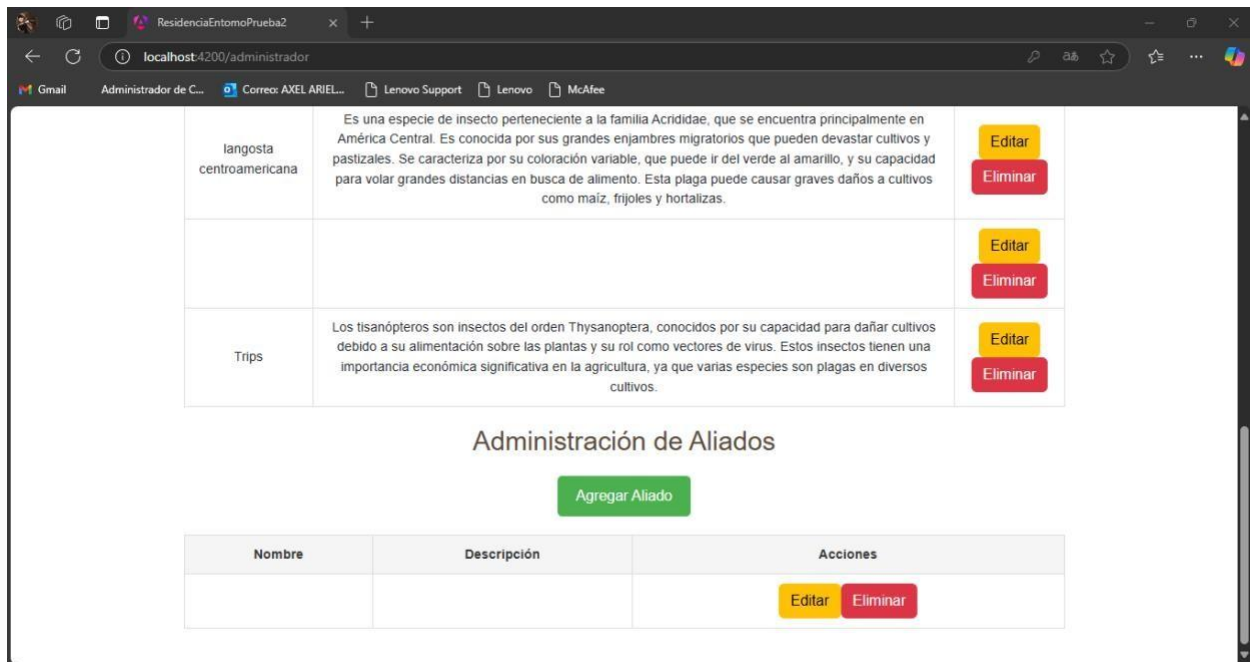


Figura 14
ilustración de metodología: administración de aliados

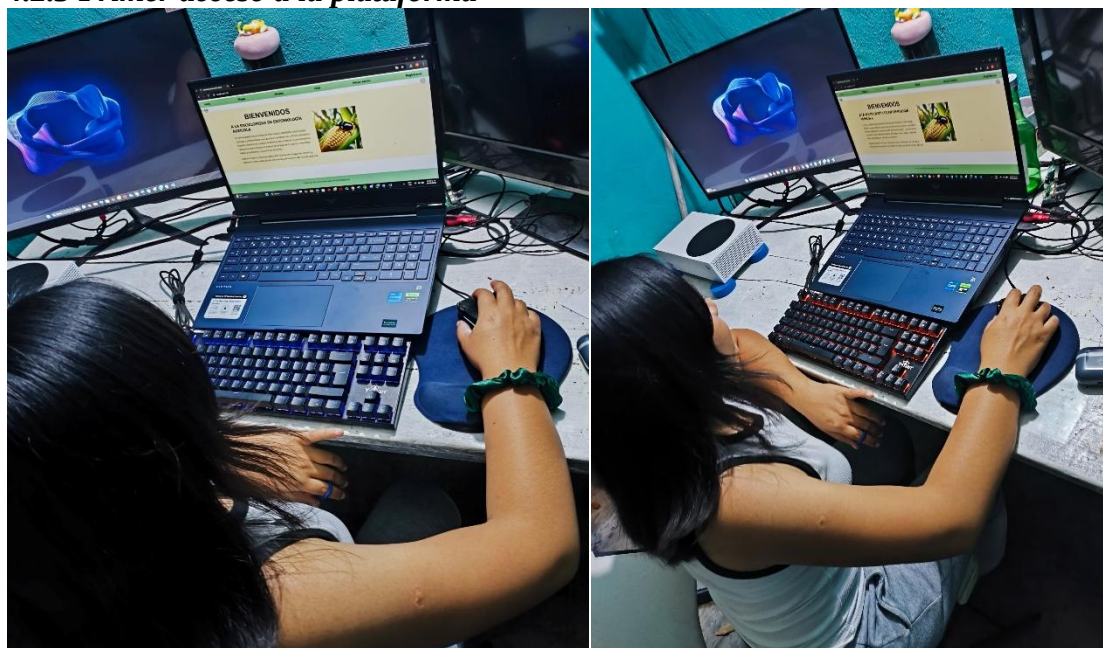


4.2 Evaluación de la plataforma.

El sistema de gestión de datos entomológicos fue evaluado a través de una serie de pruebas que permitieron determinar su efectividad y viabilidad en el manejo de plagas agrícolas en la región de Campeche. En esta sección, se presenta el análisis de los resultados obtenidos a partir de las pruebas funcionales, de usabilidad y de rendimiento

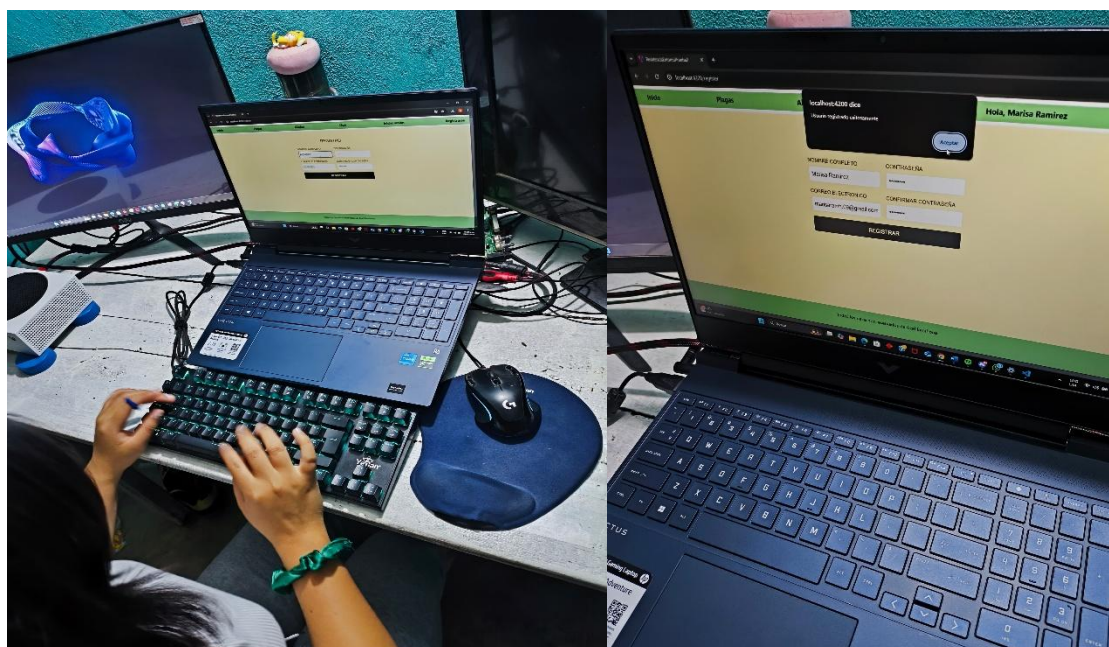
4.2.1 Prueba de Usuario

4.2.3 Primer acceso a la plataforma



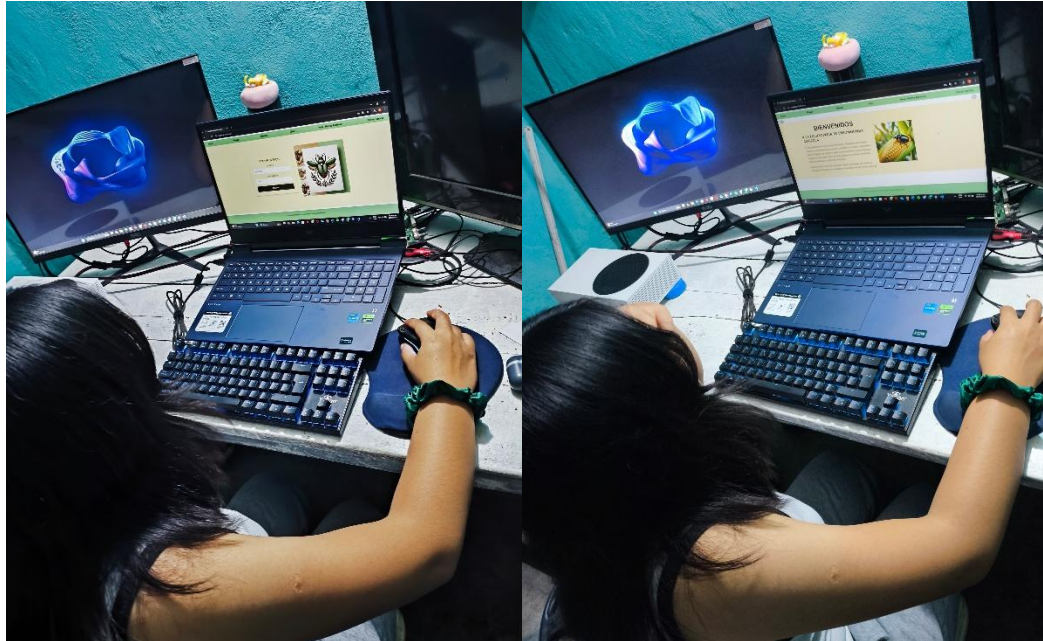
En estas imágenes, se muestra a un usuario abriendo la plataforma por primera vez. Al ingresar, la página le da la bienvenida al usuario, presentando una interfaz amigable y accesible. La plataforma cuenta con varias secciones principales: el Home, Plagas, Chat, Iniciar sesión y Registrarse, que facilitan la navegación y el acceso a las distintas funcionalidades del sistema. Esta estructura fue diseñada para ofrecer una experiencia de usuario intuitiva y eficiente desde el primer contacto.

4.2.4 Registro de nuevo usuario



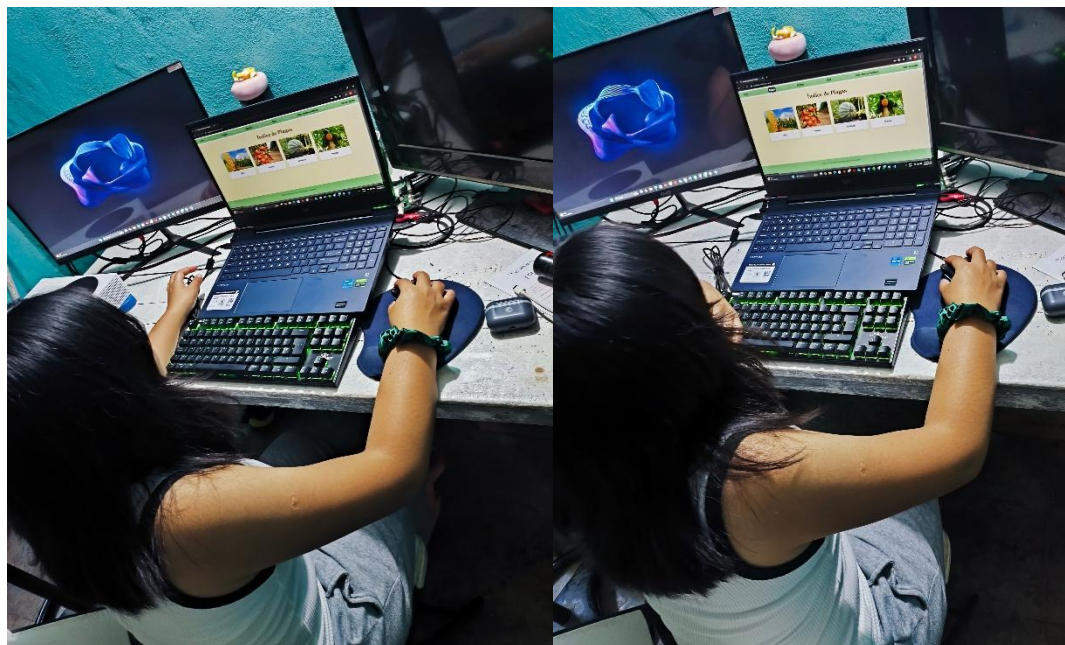
el sujeto está en el proceso de registrarse como nuevo usuario en la plataforma. El formulario de registro requiere que el usuario ingrese un correo electrónico válido, lo cual es fundamental para que la base de datos de Firebase pueda verificar la existencia del correo y aceptarlo en el sistema. Esta funcionalidad garantiza que solo los usuarios con información válida puedan acceder a las características de la plataforma, asegurando la integridad y la seguridad de los datos registrados. Además, si el usuario se crea con éxito, aparecerá un mensaje de confirmación, indicando que el nuevo usuario ha sido registrado correctamente en el sistema.

4.2.5 Inicio de sesión tras el registro exitoso



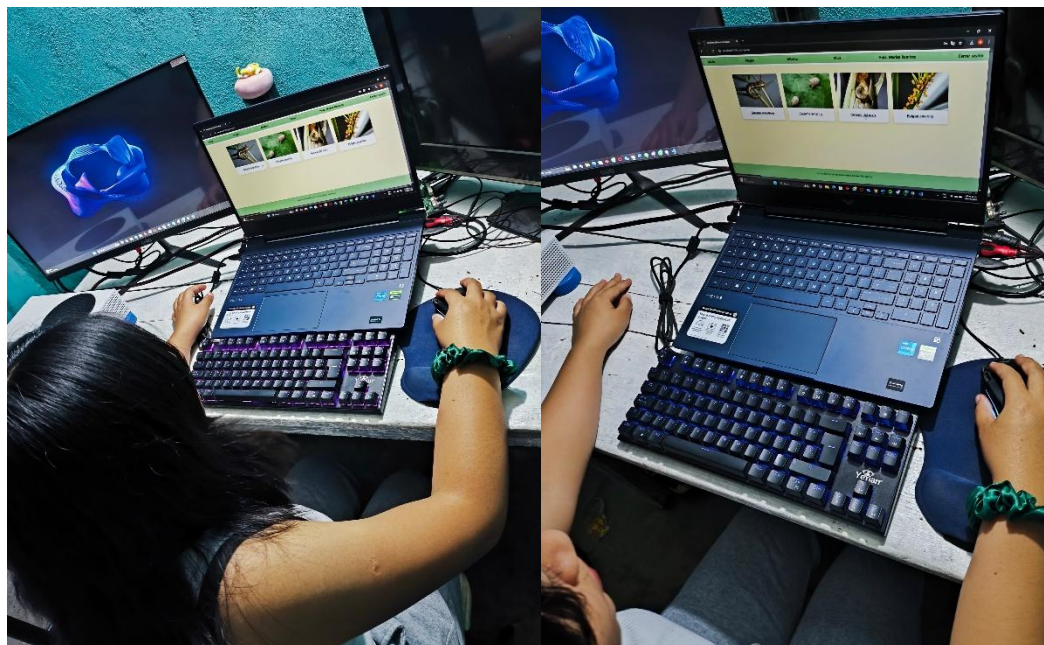
Después de crear un nuevo usuario con éxito, la página se refresca automáticamente, permitiendo al usuario iniciar sesión con su correo electrónico y la contraseña seleccionada durante el registro. En esta imagen, el sujeto está en el proceso de iniciar sesión en la plataforma utilizando sus credenciales recién creadas. Esta funcionalidad asegura que el usuario pueda acceder a las secciones del sistema de manera segura y eficiente, comenzando con su experiencia en la plataforma desde el momento en que se registra.

4.2.6 Selección de cultivos por categoría



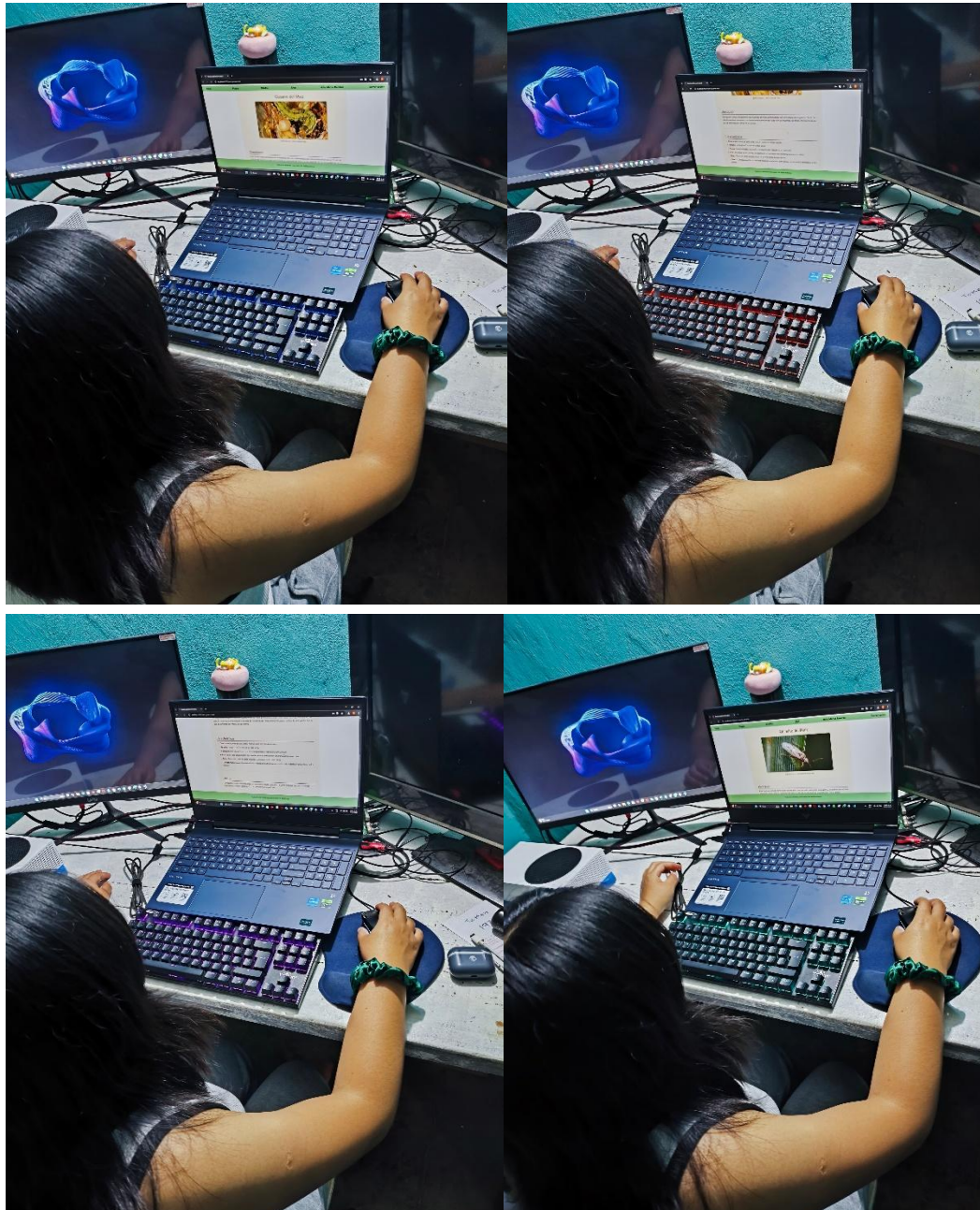
Al ingresar a la categoría de Plagas, el usuario se encuentra con un índice dividido en 4 cultivos principales, que son Maíz, Tomate, Calabaza y Papaya. Estos cultivos fueron seleccionados inicialmente para organizar la información de plagas de manera eficiente. Al hacer clic en cada uno de estos cultivos, el usuario podrá acceder a información detallada sobre las plagas que afectan a cada especie, permitiendo un manejo más específico y adecuado según las necesidades de cada tipo de cultivo.

4.2.7 Visualización de plagas para el cultivo de maíz



Después de seleccionar el cultivo de Maíz, el sistema presenta al usuario 4 especies de plagas que son los principales atacantes de este cultivo. En la imagen se muestra la interfaz con las imágenes de las plagas, lo que facilita su identificación visual. El usuario puede acceder a información detallada sobre cada una de estas plagas, incluyendo sus características biológicas, ciclo de vida y métodos de control recomendados. Esta funcionalidad ayuda a los agricultores a gestionar mejor las infestaciones y aplicar medidas de control de manera más específica y eficaz.

4.2.8 Identificación de plagas en el cultivo de maíz



Está realizando la identificación de las plagas que están atacando su cultivo de maíz.

Durante el proceso, logró localizar dos de las cuatro plagas disponibles en el sistema. La plataforma proporciona información detallada sobre cada plaga, incluyendo su descripción, características y el daño que puede causar en el cultivo. Siguiendo las indicaciones del sistema, el usuario también consultó los métodos de combate caseros recomendados para

estas plagas. Esta funcionalidad no solo permite al usuario identificar rápidamente las plagas presentes, sino también tomar decisiones informadas sobre cómo manejar la infestación de manera eficaz y utilizando métodos sostenibles y accesibles, como los remedios caseros.

4.3 Pruebas de funcionalidades

4.3.1 Resultados de las Pruebas Funcionales:

- **Consulta de Plagas:** El sistema cumplió con éxito la función de consulta de plagas, permitiendo a los usuarios buscar información detallada sobre las especies registradas. La búsqueda y visualización de información fueron rápidas y precisas, lo que demostró la eficiencia del motor de búsqueda implementado.
- **Recomendaciones de Control:** Las recomendaciones generadas por el sistema fueron validadas por los agricultores, y en su mayoría, resultaron ser pertinentes y ajustadas a las condiciones locales de los cultivos. Las sugerencias sobre métodos de control fueron claras y adecuadas, mejorando la toma de decisiones de los usuarios.

4.3.2 Resultados de las Pruebas de Usabilidad:

- **Interfaz de Usuario:** Las pruebas de usabilidad indicaron que la plataforma fue bien recibida por los agricultores, especialmente por aquellos con poca experiencia tecnológica. La navegación por el sistema fue sencilla, gracias a la interfaz amigable y el diseño intuitivo. Los usuarios pudieron acceder rápidamente a la información sobre plagas y utilizar el mapa interactivo sin dificultades.
- **Evaluación de la Experiencia del Usuario:** Los agricultores destacaron la facilidad para acceder a los datos y la utilidad de la plataforma para identificar plagas y tomar decisiones informadas sobre el manejo de sus cultivos. Sin

embargo, se sugirieron algunas mejoras en la visualización de ciertos datos, como las recomendaciones más específicas para el manejo de plagas.

4.3.3 Resultados de las Pruebas de Rendimiento:

- **Tiempo de Respuesta:** *Las pruebas de rendimiento indicaron que la plataforma responde de manera eficiente, incluso en condiciones de conectividad limitada, lo que es crucial en áreas rurales de Campeche donde el acceso a Internet puede ser intermitente.*
- **Escalabilidad:** *El sistema demostró ser escalable y capaz de manejar grandes volúmenes de datos sin afectar su rendimiento, lo que es una característica importante para futuros desarrollos*

4.3.4 Impacto en la Toma de Decisiones

Una de las principales ventajas del sistema es su capacidad para mejorar la toma de decisiones en el manejo de plagas. Gracias a la centralización de datos sobre las plagas y sus características, los agricultores pudieron acceder a información precisa que les permitió tomar decisiones fundamentadas sobre el control de plagas. Este enfoque basado en datos se ha demostrado más eficiente que los métodos tradicionales, que dependían de la experiencia empírica de los agricultores.

El sistema también redujo la dependencia de pesticidas químicos, ya que las recomendaciones generadas por la plataforma incluían métodos de control más sostenibles y adaptados a las condiciones locales. Esto, a su vez, contribuyó a mejorar la sostenibilidad de las prácticas agrícolas y a reducir los impactos negativos sobre el medio ambiente.

4.4 Comparación con Métodos Tradicionales

En comparación con los métodos tradicionales de manejo de plagas, que se basan en la experiencia empírica de los agricultores y el uso generalizado de pesticidas, el sistema ofreció una solución más precisa y eficiente. Los agricultores que utilizaron la plataforma reportaron

una mejora en la precisión de sus decisiones, lo que resultó en una mayor protección de sus cultivos y una reducción en los costos asociados con el uso de productos químicos.

Además, la plataforma permitió un enfoque más dinámico y ajustado a las necesidades de cada cultivo, proporcionando recomendaciones basadas en datos actualizados y específicos sobre las plagas presentes. Esta capacidad de personalización representó una mejora significativa en comparación con los métodos de control estándar.

4.5 Limitaciones y Desafíos

- **Conectividad a Internet:** Aunque el sistema funcionó bien en áreas con buena conectividad, algunos usuarios en zonas rurales con acceso limitado a Internet experimentaron dificultades en la actualización en tiempo real de los datos. Esto se debe tener en cuenta para futuras mejoras del sistema.
- **Capacitación de Usuarios:** Aunque la interfaz fue fácil de usar, algunos agricultores necesitaron capacitación adicional para aprovechar al máximo todas las funcionalidades de la plataforma, especialmente en lo que respecta a la entrada de datos y el uso de la funcionalidad del mapa interactivo.
- **Escalabilidad en el Futuro:** Si bien el sistema ha demostrado ser escalable en términos de rendimiento, es importante continuar mejorando la infraestructura para poder manejar aún más datos y usuarios, especialmente si se expande a otras regiones.

Capítulo IV.

Conclusiones y Recomendaciones.

La presente investigación tuvo como objetivo el desarrollo e implementación de una plataforma de gestión de datos entomológicos para optimizar el control de plagas en los cultivos de la región de Campeche. A partir de los resultados obtenidos en las pruebas de funcionalidad, usabilidad y rendimiento, se concluye que la plataforma cumple con los propósitos establecidos y representa una herramienta efectiva para la identificación y gestión de plagas en los cultivos.

En relación con la hipótesis planteada, los hallazgos de la investigación confirman su validez. Se demostró que la plataforma permite a los agricultores identificar con mayor precisión las plagas que afectan sus cultivos, incluso sin poseer conocimientos especializados en entomología. La centralización de información sobre características biológicas, ciclos de vida y patrones de comportamiento de diversas especies ha facilitado la toma de decisiones fundamentadas, mejorando la estrategia de control de plagas en la región.

Además, se verificó que la plataforma proporciona recomendaciones adecuadas sobre métodos de control, lo que ha permitido reducir la dependencia de pesticidas químicos. En comparación con las estrategias tradicionales utilizadas por los agricultores, que suelen basarse en prueba y error, el sistema ha demostrado ser más eficaz y menos costoso a largo plazo.

Las pruebas realizadas evidenciaron una alta aceptación por parte de los usuarios finales, especialmente en términos de accesibilidad y facilidad de uso. La plataforma ha permitido mejorar la eficiencia en el control de plagas, optimizando la toma de decisiones agrícolas y promoviendo una mayor sostenibilidad en el manejo de los cultivos.

No obstante, se identificaron algunas limitaciones que deben ser abordadas en futuras mejoras. Entre ellas, se encuentra la dependencia de conexión a Internet en zonas rurales, lo que podría limitar su implementación en comunidades con acceso restringido a tecnología. Asimismo,

algunos usuarios necesitaron capacitación para utilizar la plataforma de manera eficiente, lo que resalta la importancia de diseñar estrategias de formación y soporte técnico.

En conclusión, la plataforma desarrollada representa una solución viable y efectiva para la gestión de plagas en la región de Campeche, con el potencial de ser replicada y mejorada en el futuro. Su impacto positivo en la reducción del uso de pesticidas y la optimización de los métodos de control refuerza su relevancia como herramienta clave para una agricultura más sostenible y eficiente.

4.2 Recomendaciones

Con base en las conclusiones obtenidas, se presentan las siguientes recomendaciones para mejorar la plataforma y su aplicación en el sector agrícola:

1. Expansión de la base de datos entomológica

Se recomienda ampliar la base de datos incorporando nuevas especies de plagas y organismos beneficiosos, así como actualizaciones constantes sobre los métodos de control y su eficacia. Esto permitirá que la plataforma continúe siendo una herramienta relevante y confiable para los agricultores.

2. Integración de tecnologías avanzadas

Incorporar inteligencia artificial y análisis predictivo para detectar patrones en la aparición de plagas y proporcionar alertas tempranas a los agricultores. Esto fortalecería la capacidad de prevención y permitiría una respuesta más rápida ante posibles infestaciones.

3. Optimización de la accesibilidad en zonas rurales

Implementar una versión offline de la plataforma que permita a los agricultores registrar datos y consultar información sin necesidad de conexión a Internet, sincronizando los datos cuando la conexión esté disponible.

4. Capacitación y soporte a los usuarios

Desarrollar programas de capacitación y guías interactivas para que los agricultores y técnicos agrícolas puedan aprovechar al máximo las funcionalidades de la plataforma. Esto contribuirá a mejorar la adopción y el uso eficaz del sistema.

5. Expansión a otras regiones y cultivos

Dado el éxito del sistema en la región de Campeche, se recomienda su expansión a otras áreas con características agrícolas similares. Esto podría lograrse mediante colaboraciones con instituciones gubernamentales, universidades y asociaciones agrícolas.

6. Evaluaciones continuas del impacto del sistema

Realizar estudios periódicos sobre el impacto de la plataforma en la reducción del uso de pesticidas y la mejora en la producción agrícola. Esto permitirá identificar oportunidades de mejora y justificar futuras inversiones en su desarrollo.

7. Mapeo de identificación:

Integrar un sistema de mapeo interactivo que permita visualizar de forma clara la presencia y distribución de plagas en la región de Campeche, basándose en la información proporcionada por los usuarios y otros registros disponibles, para facilitar la identificación temprana de riesgos y la aplicación de medidas preventivas.

En resumen, la implementación de estas recomendaciones permitirá fortalecer la plataforma y garantizar su eficacia a largo plazo, asegurando que siga siendo una herramienta clave en la optimización del control de plagas en la región de Campeche y en otras comunidades agrícolas.

Referencias Bibliográficas.

Referencia bibliográfica

- "Aguinaga Restrepo, S. (2017). Propuesta de expansión de mercado en municipios aledaños al Valle de Aburrá para una empresa de control de plagas. Universidad EAFIT."
- "Atehortúa Millán, C. H. (2019). Diseño de una herramienta para la gestión de procesos agrícolas con integración de metodologías ágiles [Tesis de maestría, Universidad Nacional de Colombia]. Repositorio Institucional UNAL. Recuperado de https://repositorio.unal.edu.co/bitstream/handle/unal/76710/2019-Carlos_Humberto_Atehortua_Millan.pdf"
- "Cañadas, J., del Águila, I. M., & Palma, J. T. (2011). SAVIA. Un sistema web de ayuda a la decisión para control de plagas en el cultivo de uva de mesa. II Simposio Nacional de Ingeniería Hortícola: Automatización y TICs en Agricultura, 229-232."
- "Fouquet Calderón, F. (2021). Inteligencia Artificial aplicada a la agricultura de precisión: Control de plagas agrícolas [Tesis de maestría, Universidad de Málaga]. Repositorio Institucional UMA. Recuperado de <https://riuma.uma.es/xmlui/bitstream/handle/10630/23510/Fouquet%20Calder%C3%B3n%20Fabrce%20Memoria.pdf?sequence=1>"
- "García, L. M., & Rodríguez, P. A. (2020). Implementación de la metodología Scrum en el desarrollo de sistemas de información agrícola. Revista de Tecnología y Computación, 12(3), 45-58."
- "Gámez de la Hoz, J., & Padilla Fortes, A. (2015). Análisis de contenidos de páginas web sobre servicios de control de plagas en salud pública. Revista General de Información y Documentación, 25(2), 227-244: http://dx.doi.org/10.5209/rev_RGID.2015.v25.n2.51229"
- "ITC Web Solutions. (2023, enero 6). Cómo implementar la metodología Scrum en el desarrollo de sitios web. Recuperado de <https://itcwebsolutions.com/desarrollo-y-metodologias/agile/como-implementar-la-metodologia-scrum-en-el-desarrollo-de-sitios-web/>"
- "López, J. P., & Martínez, S. R. (2021). Aplicación de Scrum en el desarrollo de una plataforma web para el monitoreo de plagas en cultivos de maíz. Journal of Agricultural Informatics, 10(2), 112-125."
- "López-Gorozabel, O. A., Malla-Valdiviezo, R. O., Morejón-López, G. E., & Intriago-Cedeño, M. L. (2023). Sistema de detección de plagas en los cultivos. Revista Digital Publisher CEIT, 9(1), 128-137: <https://doi.org/10.33386/593dp.2024.1.1898>"
- "Macas Bermeo, J. R. (2022). Desarrollo de una aplicación para la detección de plagas en el cultivo de papas aplicando redes neuronales convolucionales. Universidad Técnica de Machala."
- "Molinarés Becerra, M. (2023). Modelo de página web para el control de plagas y enfermedades en agricultura (Trabajo Fin de Máster). Universidad de Cádiz: <https://hdl.handle.net/2133/18616>"
- "Moreno Bejarano, L. G., & Ponciano Camacho, D. J. (2019). Sistema de información vía web para mejorar el control de plagas en el cultivo de palta de la empresa los Viñedos S.A. Universidad César Vallejo."
- Overstand. (2023). Inteligencia Artificial para la prevención y manejo de plagas agrícolas. Recuperado de <https://overstand.es/blog/post/prevencion-plagas-agricolas-con-ia>
- "Pérez, A. L., & Gómez, M. T. (2019). Desarrollo de un sistema de gestión de datos entomológicos utilizando metodologías ágiles. Revista Colombiana de Computación, 20(1), 33-47."



- "Rodríguez, F. J., & Hernández, C. A. (2022). Plataforma web para el control de plagas en cultivos de café: Un enfoque ágil. Revista de Innovación y Desarrollo Tecnológico, 5(4), 78-90."
- "SignificadosWeb. (2023), 50 Ejemplos de Proyectos con Metodología Scrum: Características, ¿qué es, cómo usar?. Recuperado de <https://significadosweb.com/50-ejemplos-de-proyectos-con-metodologia-scrum-%E2%96%B6%E2%8F%B8-caracteristicas-que-es-como-usar/>"
- "Singh, G. (2022). Plataforma web multidispositivo para la participación ciudadana y de las administraciones en un Sistema de Alerta Temprana de Plagas (Trabajo de Fin de Grado). Universidad de Cádiz."
- "Sánchez, M. E., & Torres, L. F. (2020). Uso de Scrum en el desarrollo de aplicaciones web para la gestión agrícola. Revista Latinoamericana de Tecnología Agrícola, 8(3), 99-110."
- "Valles Valles, D. (2023). Sistema de información web y su influencia en el monitoreo de plagas y enfermedades de palma en las Palmas de Shanusi. Universidad Nacional de San Martín."
- "Vargas Saravia, I. R., & Machicao Terrazas, L. F. (2023). Aplicación web para la identificación y monitoreo de plagas (complejo gorgojo y complejo polillas) en el cultivo de la papa. Revista de Investigación e Innovación Agropecuaria y de Recursos Naturales, 10(1), 46-55.: <https://doi.org/10.53287/pqyj8962ae54v>"
- "Vargas, D. M., & Ramírez, J. L. (2021). Implementación de una plataforma web para la gestión de plagas agrícolas mediante Scrum. Revista de Ingeniería de Software, 14(2), 65-77."

Anexos.

Anexo 1: Inicio

Funcionamiento Logica:

El archivo `home.component.ts` implementa el componente principal de la pantalla de inicio de la plataforma. Este archivo actúa como un contenedor que organiza y presenta otros componentes clave, garantizando que los usuarios tengan una experiencia clara y estructurada desde el principio. El componente importa módulos y elementos esenciales, como `CarouselComponent`, que posiblemente muestra un carrusel de imágenes o información relevante; `ContentHome1Component`, que puede contener texto introductorio o secciones destacadas; y `PestCardsComponent`, encargado de mostrar tarjetas informativas relacionadas con las plagas. Además, se utiliza `Angular Router` para manejar la navegación entre diferentes secciones de la aplicación.

Figura 1

Anexo ilustración Inicio

```
1 import { Component } from '@angular/core';
2 import { CarouselComponent } from '../carousel/carousel.component';
3 import { ContentHome1Component } from '../content-home1/content-home1.component';
4 import { RouterModule, RouterOutlet } from '@angular/router';
5 import { PestCardsComponent } from '../pest-cards/pest-cards.component';
6
7 @Component({
8   selector: 'app-home',
9   standalone: true,
10  imports: [RouterModule, RouterOutlet, CarouselComponent, ContentHome1Component, PestCardsComponent ],
11  templateUrl: './home.component.html',
12  styleUrls: ['./home.component.scss']
13 })
14 export class HomeComponent {
15   title = 'angularproject';
16 }
17
```

Código Inicio:

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



```
import { CarouselComponent } from "../carousel/carousel.component";

import { ContentHome1Component } from "../content-home1/content-
home1.component";

import { RouterModule, RouterOutlet } from '@angular/router';

import { PestCardsComponent } from '../pest-cards/pest-cards.component';

@Component({
  selector: 'app-home',
  standalone: true,
  imports: [RouterModule, RouterOutlet, CarouselComponent,
ContentHome1Component, PestCardsComponent ],
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.scss']
})
export class HomeComponent {
  title = 'angularproject';
}
```

Anexo 2: Navbar

Funcionamiento Lógico:

Este código implementa el componente de la barra de navegación ("navbar") de la plataforma.

La barra de navegación es esencial para que los usuarios puedan moverse entre las diferentes secciones del sitio, como inicio, plagas y aliados. Además, este componente muestra si un usuario ha iniciado sesión y le permite cerrar sesión cuando lo desee. En resumen, este código controla la visualización y funcionalidad de la barra de navegación, facilitando la interacción del usuario con la plataforma.

Figura 2

Anexo ilustracion Navbar

```
1  import { CommonModule } from '@angular/common';
2  import { Component } from '@angular/core';
3  import { RouterLink } from '@angular/router';
4  import { UserService } from '../../administrador/services/user.service'; // Ru
5
6  @Component({
7    selector: 'app-navbar',
8    standalone: true,
9    imports: [CommonModule, RouterLink],
10   templateUrl: './navbar.component.html',
11   styleUrls: ['./navbar.component.scss']
12 })
13 export class NavbarComponent {
14   user: any | null = null;
15
16   constructor(private userService: UserService) {
17     // Suscribirse al estado del usuario
18     this.userService.currentUser$.subscribe((currentUser: any) => {
19       this.user = currentUser;
20     });
21   }
22
23   logout() {
24     this.userService.logout(); // Cerrar sesión
25   }
26 }
27
```

Figura 3

Anexo Ilustracion Navbar

```
1 import { Component } from '@angular/core';
2 import { CarouselComponent } from '../carousel/carousel.component';
3 import { ContentHome1Component } from '../content-home1/content-home1.component';
4 import { RouterModule, RouterOutlet } from '@angular/router';
5 import { PestCardsComponent } from '../pest-cards/pest-cards.component';
6
7 @Component({
8   selector: 'app-home',
9   standalone: true,
10  imports: [RouterModule, RouterOutlet, CarouselComponent, ContentHome1Component, PestCardsComponent ],
11  templateUrl: './home.component.html',
12  styleUrls: ['./home.component.scss']
13 })
14 export class HomeComponent {
15   title = 'angularproject';
16 }
17
```

Código Navbar:

```
import { CommonModule } from '@angular/common';

import { Component } from '@angular/core';

import { RouterLink } from '@angular/router';

import { UserService } from '../administrador/services/user.service'; // Ruta relativa
```

```
@Component({
  selector: 'app-navbar',
  standalone: true,
  imports: [CommonModule, RouterLink],
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.scss']
})

export class NavbarComponent {
  user: any | null = null;
```

```
constructor(private userService: UserService) {  
  
  // Suscribirse al estado del usuario  
  
  this.userService.currentUser$.subscribe((currentUser: any) => {  
  
    this.user = currentUser;  
  
  });  
}  
  
logout() {  
  
  this.userService.logout(); // Cerrar sesión  
  
}  
}
```

Anexo 3: Footer

Funcionamiento Lógico:

Este código implementa el componente de pie de página ("footer") de la plataforma. Su propósito principal es mostrar información estática como derechos reservados, enlaces adicionales o cualquier contenido relevante en la parte inferior de cada página. Es un componente independiente (standalone), lo que permite su reutilización en diferentes partes de

la aplicación sin depender de un módulo específico.

Figura 4

Anexo ilustración footer

```
1 // footer.component.ts
2 import { Component } from '@angular/core';
3
4 @Component({
5   selector: 'app-footer',
6   standalone: true, // Asegúrate de que sea st
7   templateUrl: './footer.component.html',
8   styleUrls: ['./footer.component.scss']
9 })
10 export class FooterComponent { }
11
```

Código Footer:

// footer.component.ts

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

```
@Component({
  selector: 'app-footer',
  standalone: true, // Asegúrate de que sea standalone
  templateUrl: './footer.component.html',
  styleUrls: ['./footer.component.scss']
})
export class FooterComponent { }
```

Anexo 4: App.routes.ts

Funcionamiento Lógico:

Este archivo define las rutas de la aplicación, estableciendo cómo se cargan los diferentes componentes según la URL ingresada por el usuario. Utiliza carga diferida para

optimizar el rendimiento, asegurándose de que cada componente se cargue solo cuando sea necesario.

Cada ruta tiene un *path* (parte de la URL) asociado a un componente que se carga dinámicamente. Por ejemplo, el *path*: 'home' carga el componente *HomeComponent*, y rutas

Figura 5

Anexo ilustración parte 1 *app.routes.ts*

```
1 import { Routes } from '@angular/router';
2
3 export const routes: Routes = [
4   {
5     path: '',
6     loadComponent: () =>
7       import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga diferida del HomeComponent
8     children: [
9       {
10        path: '',
11        loadComponent: () =>
12          import('./home/home1/home.component').then((c) => c.HomeComponent),
13      },
14      {
15        path: 'home',
16        loadComponent: () =>
17          import('./home/home1/home.component').then((c) => c.HomeComponent),
18      },
19    ],
20    path: 'carousel',
21    loadComponent: () =>
22      import('./home/carousel/carousel.component').then((c) => c.CarouselComponent), // Ruta para cargar el Carousel
23  },
24  {
25    path: 'content-home1',
26    loadComponent: () =>
27      import('./home/content-home1/content-home1.component').then((c) => c.ContentHome1Component), // Ruta para cargar el ContentHome1Component
28  },
29  {
30    path: 'chat',
31    loadComponent: () =>
32      import('./chat/chat.component').then((c) => c.ChatComponent), // Ruta para cargar el ContentHome1Component
33  },
34  {
35    path: 'login',
36    loadComponent: () =>
37      import('./login/login-content1/login-content1.component').then((c) => c.LoginContent1Component),
38  },
39 ]
```

específicas como *path*: 'chat' cargan el componente *ChatComponent*. También se organizan rutas relacionadas en estructuras anidadas, como las de administrador, que agrupan funcionalidades administrativas. Esto permite una navegación eficiente y modular dentro de la aplicación.

Figura 6

Anexo ilustración parte 2 app.routes.ts

```
3 export const routes: Routes = [
8   children: [
37     import('./login/login-content1/login-content1.component').then((c) => c.LoginContent1Component),
38   ],
39   {
40     path: 'register',
41     loadComponent: ()=>
42       import('./register/content-register1/content-register1.component').then((c) => c.ContentRegister1Component),
43   },
44   {
45     path: 'pest-cards',
46     loadComponent: ()=>
47       import('./home/pest-cards/pest-cards.component').then((c) => c.PestCardsComponent),
48   },
49   {
50     path: 'gastropods',
51     loadComponent: ()=>
52       import('./index-pests/index-pests-gasteoropodos/index-pests-gasteoropodos.component').then((c) => c.IndexPest),
53   },
54   {
55     path: 'pest-gusano-cogollero',
56     loadComponent: ()=>
57       import('./administrador/pest-gusano-cogollero/pest-gusano-cogollero.component').then((c) => c.PestGusanoCogol),
58   },
59   {
60     path: 'pest-chinche-amarilla',
61     loadComponent: ()=>
62       import('./administrador/pest-chinche-maiz/pest-chinche-maiz.component').then((c) => c.PestChincheMaizComponen),
63   },
64   {
65     path: 'pest-gusano-maiz',
66     loadComponent: ()=>
67       import('./administrador/pest-gusano-maiz/pest-gusano-maiz.component').then((c) => c.PestGusanoMaizComponent),
68   },
69   {
70     path: 'pest-pulgon-amarillo',
71     loadComponent: ()=>
```

Lín. 2, col. 1 — Espacios: 2 — UTF-8 — CRLF — TypeScript 88

Figura 7

Anexo ilustracion parte 3 app.routes.ts

```

3 export const routes: Routes = [
8   children: [
71     loadComponent: ()=>
72       import('./administrador/pest-pulgón-amarillo/pest-pulgón-amarillo.component').then((c) => c.PestPulgónAmarillo),
73     },
74     {
75       path: 'acaros',
76       loadComponent: ()=>
77         import('./index-pests/index-pests-acaros/index-pests-acaros.component').then((c) => c.IndexPestsAcarosComponent),
78     },
79     {
80       path: 'pest-gusano-tomate',
81       loadComponent: ()=>
82         import('./administrador/pest-gusano-tomate/pest-gusano-tomate.component').then((c) => c.PestGusanoTomateComponent),
83     },
84     {
85       path: 'pest-mosca-blanca',
86       loadComponent: ()=>
87         import('./administrador/pest-mosca-blanca/pest-mosca-blanca.component').then((c) => c.PestMoscaBlancaComponent),
88     },
89     {
90       path: 'pest-pulgón-tomate',
91       loadComponent: ()=>
92         import('./administrador/pest-pulgón-tomate/pest-pulgón-tomate.component').then((c) => c.PestPulgónTomateComponent),
93     },
94     {
95       path: 'pest-cochinilla-algodon',
96       loadComponent: ()=>
97         import('./administrador/pest-cochinilla-algodon/pest-cochinilla-algodon.component').then((c) => c.PestCochinillaAlgodonComponent),
98     },
99     {
100      path: 'pest-pulgon',
101      loadComponent: ()=>
102        import('./administrador/pest-pulgon/pest-pulgon.component').then((c) => c.PestPulgonComponent),
103    },
104    {
105      path: 'nematodos',

```

Figura 8

Anexo ilustracion parte 4 app.routes.ts

```
3  export const routes: Routes = [  
8  children: [  
106    loadComponent: ()=>  
107      import('./index-pests/index-pests-nematodos/index-pests-nematodos.component').then((c) => c.IndexPestsNematod  
108    },  
109    {  
110      path: 'pest-mosca-calabaza',  
111      loadComponent: ()=>  
112        import('./administrador/pest-mosca-calabaza/pest-mosca-calabaza.component').then((c) => c.PestMoscaCalabazaCo  
113    },  
114    {  
115      path: 'pest-gusano-cortador',  
116      loadComponent: ()=>  
117        import('./administrador/pest-gusano-cortador/pest-gusano-cortador.component').then((c) => c.PestGusanoCortado  
118    },  
119    {  
120      path: 'pest-pulgon-calabaza',  
121      loadComponent: ()=>  
122        import('./administrador/pest-pulgon-calabaza/pest-pulgon-calabaza.component').then((c) => c.PestPulgonCalabaz  
123    },  
124    {  
125      path: 'pest-cochinilla-calabaza',  
126      loadComponent: ()=>  
127        import('./administrador/pest-cochinilla-calabaza/pest-cochinilla-calabaza.component').then((c) => c.PestCochi  
128    },  
129    {  
130      path: 'insectos',  
131      loadComponent: ()=>  
132        import('./index-pests/index-pests-insectos/index-pests-insectos.component').then((c) => c.IndexPestsInsectosC  
133    },  
134    {  
135      path: 'allies-pest-cards',  
136      loadComponent: ()=>  
137        import('./Allies-pest/allies-pest-cards/allies-pest-cards.component').then((c) => c.AlliesPestCardsComponent)  
138    },  
139    {  
140      path: 'gastronods-allies'
```

Figura 9

Anexo ilustración parte 5 app.routes.ts

```
3 export const routes: Routes = [  
8   children: [  
143     import('./allies-pest/allies-pest-gasteropodos/allies-pest-gasteropodos.component').then((c) => c.AlliesPestGasteropodosComponent),  
144     {  
145       path: 'acaros-allies',  
146       loadComponent: ()=>  
147         import('./Allies-pest/allies-pest-acaros/allies-pest-acaros.component').then((c) => c.AlliesPestAcarosComponent),  
148     },  
149     {  
150       path: 'nematodos-allies',  
151       loadComponent: ()=>  
152         import('./Allies-pest/allies-pest-nematodos/allies-pest-nematodos.component').then((c) => c.AlliesPestNematodosComponent),  
153     },  
154     {  
155       path: 'insectos-allies',  
156       loadComponent: ()=>  
157         import('./Allies-pest/allies-pest-insectos/allies-pest-insectos.component').then((c) => c.AlliesPestInsectosComponent),  
158     },  
159   ],  
160 },  
161 ],  
162 {  
163   path: 'administrador',  
164   loadComponent: ()=>  
165     import('./administrador/administrador-dashboard/administrador-dashboard.component').then((c) => c.AdminDashboardComponent),  
166   children: [  
167     ] // Redirige a la raíz si no se encuentra la ruta  
168   },  
169 ];  
170
```

Código app.routes.ts:

```
import { Routes } from '@angular/router';
```

```
export const routes: Routes = [  
  {  
    path: "",  
    loadComponent: () =>  
      import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
    // diferida del HomeComponent  
    children: [  
      {  
        path: "",  
        loadComponent: () =>  
          import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
          // diferida del HomeComponent  
        children: [  
          {  
            path: "",  
            loadComponent: () =>  
              import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
              // diferida del HomeComponent  
            children: [  
              {  
                path: "",  
                loadComponent: () =>  
                  import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                  // diferida del HomeComponent  
                children: [  
                  {  
                    path: "",  
                    loadComponent: () =>  
                      import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                      // diferida del HomeComponent  
                    children: [  
                      {  
                        path: "",  
                        loadComponent: () =>  
                          import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                          // diferida del HomeComponent  
                        children: [  
                          {  
                            path: "",  
                            loadComponent: () =>  
                              import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                              // diferida del HomeComponent  
                            children: [  
                              {  
                                path: "",  
                                loadComponent: () =>  
                                  import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                                  // diferida del HomeComponent  
                                children: [  
                                  {  
                                    path: "",  
                                    loadComponent: () =>  
                                      import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                                      // diferida del HomeComponent  
                                ]  
                              }  
                            ]  
                          }  
                        ]  
                      }  
                    ]  
                  }  
                ]  
              }  
            ]  
          }  
        ]  
      }  
    ]  
  }  
];
```

diferida del HomeComponent

```
children: [  
  {  
    path: "",  
    loadComponent: () =>  
      import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
      // diferida del HomeComponent  
    children: [  
      {  
        path: "",  
        loadComponent: () =>  
          import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
          // diferida del HomeComponent  
        children: [  
          {  
            path: "",  
            loadComponent: () =>  
              import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
              // diferida del HomeComponent  
            children: [  
              {  
                path: "",  
                loadComponent: () =>  
                  import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                  // diferida del HomeComponent  
                children: [  
                  {  
                    path: "",  
                    loadComponent: () =>  
                      import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                      // diferida del HomeComponent  
                    children: [  
                      {  
                        path: "",  
                        loadComponent: () =>  
                          import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                          // diferida del HomeComponent  
                        children: [  
                          {  
                            path: "",  
                            loadComponent: () =>  
                              import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                              // diferida del HomeComponent  
                            children: [  
                              {  
                                path: "",  
                                loadComponent: () =>  
                                  import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                                  // diferida del HomeComponent  
                                children: [  
                                  {  
                                    path: "",  
                                    loadComponent: () =>  
                                      import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                                      // diferida del HomeComponent  
                                ]  
                              }  
                            ]  
                          }  
                        ]  
                      }  
                    ]  
                  }  
                ]  
              }  
            ]  
          }  
        ]  
      }  
    ]  
  }  
];
```

```
path: "",  
loadComponent: () =>  
  import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
  // diferida del HomeComponent  
children: [  
  {  
    path: "",  
    loadComponent: () =>  
      import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
      // diferida del HomeComponent  
    children: [  
      {  
        path: "",  
        loadComponent: () =>  
          import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
          // diferida del HomeComponent  
        children: [  
          {  
            path: "",  
            loadComponent: () =>  
              import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
              // diferida del HomeComponent  
            children: [  
              {  
                path: "",  
                loadComponent: () =>  
                  import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                  // diferida del HomeComponent  
                children: [  
                  {  
                    path: "",  
                    loadComponent: () =>  
                      import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                      // diferida del HomeComponent  
                    children: [  
                      {  
                        path: "",  
                        loadComponent: () =>  
                          import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                          // diferida del HomeComponent  
                        children: [  
                          {  
                            path: "",  
                            loadComponent: () =>  
                              import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                              // diferida del HomeComponent  
                            children: [  
                              {  
                                path: "",  
                                loadComponent: () =>  
                                  import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                                  // diferida del HomeComponent  
                                children: [  
                                  {  
                                    path: "",  
                                    loadComponent: () =>  
                                      import('./margen/inicio/inicio.component').then((c) => c.InicioComponent), // Carga  
                                      // diferida del HomeComponent  
                                ]  
                              }  
                            ]  
                          }  
                        ]  
                      }  
                    ]  
                  }  
                ]  
              }  
            ]  
          }  
        ]  
      }  
    ]  
  }  
];
```



```
loadComponent: () =>
    import('./home/home1/home.component').then((c) => c.HomeComponent),
},
{
    path: 'home',
    loadComponent: () =>
        import('./home/home1/home.component').then((c) => c.HomeComponent),
},
{
    path: 'carousel',
    loadComponent: () =>
        import('./home/carousel/carousel.component').then((c) => c.CarouselComponent),
// Ruta para cargar el CarouselComponent
},
{
    path: 'content-home1',
    loadComponent: () =>
        import('./home/content-home1/content-home1.component').then((c) =>
c.ContentHome1Component), // Ruta para cargar el ContentHome1Component
},
{
    path: 'chat',
    loadComponent: () =>
        import('./chat/chat.component').then((c) => c.ChatComponent), // Ruta para cargar
el ContentHome1Component
},
```



```
{  
  path: 'login',  
  loadComponent: ()=>  
    import('./login/login-content1/login-content1.component').then((c) =>  
c.LoginContent1Component),  
},  
{  
  path: 'register',  
  loadComponent: ()=>  
    import('./register/content-register1/content-register1.component').then((c) =>  
c.ContentRegister1Component),  
},  
{  
  path: 'pest-cards',  
  loadComponent: ()=>  
    import('./home/pest-cards/pest-cards.component').then((c) =>  
c.PestCardsComponent),  
},  
{  
  path: 'gastropods',  
  loadComponent: ()=>  
    import('./index-pests/index-pests-gasteoropodos/index-pests-  
gasteoropodos.component').then((c) => c.IndexPestsGasteoropodosComponent),  
},  
{  
  path: 'pest-gusano-cogollero',
```



```
loadComponent: ()=>

import('./administrador/pest-gusano-cogollero/pest-gusano-
cogollero.component').then((c) => c.PestGusanoCogolleroComponent),

},

{

  path: 'pest-chinche-amarilla',

  loadComponent: ()=>

    import('./administrador/pest-chinche-maiz/pest-chinche-maiz.component').then((c)
=> c.PestChincheMaizComponent),

  },

  {

    path: 'pest-gusano-maiz',

    loadComponent: ()=>

      import('./administrador/pest-gusano-maiz/pest-gusano-maiz.component').then((c)
=> c.PestGusanoMaizComponent),

    },

    {

      path: 'pest-pulgon-amarillo',

      loadComponent: ()=>

        import('./administrador/pest-pulgon-amarillo/pest-pulgon-
amarillo.component').then((c) => c.PestPulgonAmarilloComponent),

      },

      {

        path: 'acaros',

        loadComponent: ()=>
```

```
import('./index-pests/index-pests-acaros/index-pests-acaros.component').then((c)
=> c.IndexPestsAcarosComponent),

},

{

  path: 'pest-gusano-tomate',

  loadComponent: ()=>

    import('./administrador/pest-gusano-tomate/pest-gusano-
tomate.component').then((c) => c.PestGusanoTomateComponent),

},

{

  path: 'pest-mosca-blanca',

  loadComponent: ()=>

    import('./administrador/pest-mosca-blanca/pest-mosca-blanca.component').then((c)
=> c.PestMoscaBlancaComponent),

},

{

  path: 'pest-pulgon-tomate',

  loadComponent: ()=>

    import('./administrador/pest-pulgon-tomate/pest-pulgon-
tomate.component').then((c) => c.PestPulgonTomateComponent),

},

{

  path: 'pest-cochinilla-algodon',

  loadComponent: ()=>

    import('./administrador/pest-cochinilla-algodon/pest-cochinilla-
algodon.component').then((c) => c.PestCochinillaAlgodonComponent),
```



```
},  
  
{  
  path: 'pest-pulgonas',  
  loadComponent: ()=>  
    import('./administrador/pest-pulgonas/pest-pulgonas.component').then((c) =>  
c.PestPulgonasComponent),  
  },  
  
{  
  path: 'nematodos',  
  loadComponent: ()=>  
    import('./index-pests/index-pests-nematodos/index-pests-  
nematodos.component').then((c) => c.IndexPestsNematodosComponent),  
  },  
  
{  
  path: 'pest-mosca-calabaza',  
  loadComponent: ()=>  
    import('./administrador/pest-mosca-calabaza/pest-mosca-  
calabaza.component').then((c) => c.PestMoscaCalabazaComponent),  
  },  
  
{  
  path: 'pest-gusano-cortador',  
  loadComponent: ()=>  
    import('./administrador/pest-gusano-cortador/pest-gusano-  
cortador.component').then((c) => c.PestGusanoCortadorComponent),  
  },  
  
{
```



```
path: 'pest-pulgon-calabaza',  
loadComponent: ()=>  
  import('./administrador/pest-pulgon-calabaza/pest-pulgon-  
calabaza.component').then((c) => c.PestPulgonCalabazaComponent),  
},  
{  
  path: 'pest-cochinilla-calabaza',  
  loadComponent: ()=>  
    import('./administrador/pest-cochinilla-calabaza/pest-cochinilla-  
calabaza.component').then((c) => c.PestCochinillaCalabazaComponent),  
},  
{  
  path: 'insectos',  
  loadComponent: ()=>  
    import('./index-pests/index-pests-insectos/index-pests-  
insectos.component').then((c) => c.IndexPestsInsectosComponent),  
},  
{  
  path: 'allies-pest-cards',  
  loadComponent: ()=>  
    import('./Allies-pest/allies-pest-cards/allies-pest-cards.component').then((c) =>  
c.AlliesPestCardsComponent),  
},  
{  
  path: 'gastropods-allies',  
  loadComponent: ()=>
```



```
import('./Allies-pest/allies-pest-gasteropodos/allies-pest-  
gasteropodos.component').then((c) => c.AlliesPestGasteropodosComponent),  
  
},  
  
{  
  
  path: 'acaros-allies',  
  
  loadComponent: ()=>  
  
    import('./Allies-pest/allies-pest-acaros/allies-pest-acaros.component').then((c) =>  
c.AlliesPestAcarosComponent),  
  
},  
  
{  
  
  path: 'nematodos-allies',  
  
  loadComponent: ()=>  
  
    import('./Allies-pest/allies-pest-nematodos/allies-pest-  
nematodos.component').then((c) => c.AlliesPestNematodosComponent),  
  
},  
  
{  
  
  path: 'insectos-allies',  
  
  loadComponent: ()=>  
  
    import('./Allies-pest/allies-pest-insectos/allies-pest-insectos.component').then((c)  
=> c.AlliesPestInsectosComponent),  
  
},  
  
]  
  
},  
  
{  
  
  path: 'administrador',
```

```
loadComponent: ()=>

import('./administrador/administrador-dashboard/administrador-
dashboard.component').then((c) => c.AdminDashboardComponent),

children: [

] // Redirige a la raíz si no se encuentra la ruta

}

];
```

Anexo 5: pest-cards

Funcionamiento Lógico:

Este archivo implementa el componente *PestCardsComponent*, que probablemente muestra tarjetas informativas relacionadas con diferentes plagas. Estas tarjetas pueden contener detalles como el nombre de la plaga, su descripción, y posibles métodos de control. El componente utiliza *RouterLink* para permitir la navegación a páginas específicas relacionadas con cada plaga. Es un componente independiente (*standalone*), lo que facilita su reutilización en otras partes de la aplicación.

Figura 10

Anexo ilustración *pest-card.component.ts*

```
1  import { Component } from '@angular/core';
2  import { RouterLink } from '@angular/router';
3
4  @Component({
5    selector: 'app-pest-cards',
6    standalone: true,
7    imports: [RouterLink],
8    templateUrl: './pest-cards.component.html',
9    styleUrls: ['./pest-cards.component.scss']
10 })
11 export class PestCardsComponent {
12
13 }
14
```

Código pest-card:

```
import { Component } from '@angular/core';  
  
import { RouterLink } from '@angular/router';
```

```
@Component({  
  
  selector: 'app-pest-cards',  
  
  standalone: true,  
  
  imports: [RouterLink],  
  
  templateUrl: './pest-cards.component.html',  
  
  styleUrls: ['./pest-cards.component.scss']  
})  
  
export class PestCardsComponent {  
  
}
```

Anexo 6: Índice de plagas

Funcionamiento Lógico:

El componente IndexPestsGasteoropodosComponent organiza y presenta información específica sobre las plagas gasterópodos. Integra capacidades de navegación mediante RouterLink y RouterModule, permitiendo a los usuarios acceder fácilmente a páginas relacionadas. Su diseño se centra en ofrecer un índice claro y funcional que facilite la consulta y exploración de datos sobre este tipo de plagas.

Figura 11

Anexo ilustración *IndexPestsGasteoropodosComponent.ts*

```
1  ∨ import { Component } from '@angular/core';  
2  import { RouterLink, RouterModule } from '@angular/router';  
3  
4  ∨ @Component({  
5      selector: 'app-index-pests-gasteoropodos',  
6      standalone: true,  
7      imports: [RouterLink,RouterModule],  
8      templateUrl: './index-pests-gasteoropodos.component.html',  
9      styleUrls: ['./index-pests-gasteoropodos.component.scss']  
10 })  
11 ∨ export class IndexPestsGasteoropodosComponent {  
12  
13 }  
14
```

Código IndexPest:

```
import { Component } from '@angular/core';  
  
import { RouterLink, RouterModule } from '@angular/router';  
  
@Component({  
  
    selector: 'app-index-pests-gasteoropodos',  
  
    standalone: true,  
  
    imports: [RouterLink,RouterModule],  
  
    templateUrl: './index-pests-gasteoropodos.component.html',  
  
    styleUrls: ['./index-pests-gasteoropodos.component.scss']  
  
})  
  
export class IndexPestsGasteoropodosComponent {  
  
  
  
}
```

Anexo 7: Plaga de Maíz

Funcionamiento Lógico:

El componente `IndexPestsGasteoropodosComponent` se utiliza para mostrar un índice específico relacionado con las plagas gasterópodos. Incluye herramientas de navegación, como `RouterLink` y `RouterModule`, que permiten redirigir a otras secciones de la plataforma para obtener más información o detalles relacionados. Este componente sirve como punto de referencia dentro de la categoría de plagas gasterópodos, proporcionando un acceso organizado y funcional.

Figura 12

Anexo ilustración `IndexPestsGasteoropodosComponent.ts`

```
1  import { Component } from '@angular/core';
2  import { RouterLink, RouterModule } from '@angular/router';
3
4  @Component({
5    selector: 'app-index-pests-gasteoropodos',
6    standalone: true,
7    imports: [RouterLink, RouterModule],
8    templateUrl: './index-pests-gasteoropodos.component.html',
9    styleUrls: ['./index-pests-gasteoropodos.component.scss']
10 })
11 export class IndexPestsGasteoropodosComponent {
12
13 }
14
```

Código Plaga maíz:

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

```
import { RouterLink, RouterModule } from '@angular/router';
```

```
@Component({
```

```
  selector: 'app-index-pests-gasteoropodos',
```

```
  standalone: true,
```

```
imports: [RouterLink,RouterModule],  
  
templateUrl: './index-pests-gasteoropodos.component.html',  
  
styleUrl: './index-pests-gasteoropodos.component.scss'  
  
})  
  
export class IndexPestsGasteoropodosComponent {  
  
  
  
}
```

Anexo 8: Gusano cogollero – Ejemplo de vista

Funcionamiento Lógico:

El componente PestGusanoCogolleroComponent se encarga de mostrar información específica sobre la plaga conocida como "Gusano Cogollero". Este componente utiliza un servicio (PlagaService) para obtener datos de la plaga desde la base de datos, identificándola mediante un ID específico. Además, incluye un marcador simulado para representar la ubicación geográfica de la plaga en un mapa, lo que facilita la visualización de su distribución.

El método principal, getPlagaData, es llamado al inicializar el componente (dentro de ngOnInit) y utiliza una suscripción para manejar los datos obtenidos o registrar errores si la carga falla. Este componente combina datos dinámicos con herramientas de navegación y lógica de presentación para enriquecer la experiencia del usuario.

Figura 13

Anexo ilustración parte1 PestsGusanoCogolleroComponent.ts

```
1 import { Component, OnInit } from '@angular/core';
2 import { RouterLink, RouterModule } from '@angular/router';
3 import { CommonModule } from '@angular/common'; // Importa CommonModule para utilizar *ngIf
4 import { PlagaService } from '../services/plaga.service';
5 import { Plaga } from '../services/plaga.model';
6
7 @Component({
8   selector: 'app-pest-gusano-cogollero',
9   standalone: true,
10  imports: [RouterModule, RouterLink, CommonModule], // Incluye CommonModule aquí
11  templateUrl: './pest-gusano-cogollero.component.html',
12  styleUrls: ['./pest-gusano-cogollero.component.scss']
13 })
14 export class PestGusanoCogolleroComponent implements OnInit {
15   plaga: Plaga | undefined;
16
17   // Coordenadas del marcador para simular la ubicación en el mapa
18   markerPosition = {
19     top: '60%', // Porcentaje de la altura de la imagen
20     left: '70%' // Porcentaje de la anchura de la imagen
21   };
22
23   constructor(private plagaService: PlagaService) {}
24
25   ngOnInit(): void {
26     this.getPlagaData();
27   }
28
29   getPlagaData(): void {
30     // Ajusta este ID o la lógica para obtener la plaga deseada (en este caso, Gusano Cogollero)
31     const plagaId = 'gusano-cogollero-id'; // Reemplaza con el ID real de la plaga en Firebase
32     this.plagaService.getPlagaById(plagaId).subscribe(
33       (plaga) => {
34         this.plaga = plaga;
35       },
36       (error) => {
37         console.error('Error al cargar los datos de la plaga:', error);
38       }
39     );
40   }
41 }
42
```

Figura 14

Anexo ilustración parte2 PestsGusanoCogolleroComponent.ts

```
14 export class PestGusanoCogolleroComponent implements OnInit {
29   getPlagaData(): void {
35     },
36     (error) => {
37       console.error('Error al cargar los datos de la plaga:', error);
38     }
39   );
40 }
41 }
42
```



Código PestGusanoCogollero:

```
import { Component, OnInit } from '@angular/core';

import { RouterLink, RouterModule } from '@angular/router';

import { CommonModule } from '@angular/common'; // Importa CommonModule para
utilizar *ngIf

import { PlagaService } from '../services/plaga.service';
import { Plaga } from '../services/plaga.model';

@Component({
  selector: 'app-pest-gusano-cogollero',
  standalone: true,
  imports: [RouterModule, RouterLink, CommonModule], // Incluye CommonModule aquí
  templateUrl: './pest-gusano-cogollero.component.html',
  styleUrls: ['./pest-gusano-cogollero.component.scss']
})
export class PestGusanoCogolleroComponent implements OnInit {
  plaga: Plaga | undefined;

  // Coordenadas del marcador para simular la ubicación en el mapa
  markerPosition = {
    top: '60%', // Porcentaje de la altura de la imagen
    left: '70%' // Porcentaje de la anchura de la imagen
  };

  constructor(private plagaService: PlagaService) {}
```

```
ngOnInit(): void {
```

```
  this.getPlagaData();
```

```
}
```

```
getPlagaData(): void {
```

```
  // Ajusta este ID o la lógica para obtener la plaga deseada (en este caso, Gusano
```

Cogollero)

```
  const plagald = 'gusano-cogollero-id'; // Reemplaza con el ID real de la plaga en
```

Firebase

```
  this.plagaService.getPlagaById(plagald).subscribe(
```

```
    (plaga) => {
```

```
      this.plaga = plaga;
```

```
    },
```

```
    (error) => {
```

```
      console.error('Error al cargar los datos de la plaga:', error);
```

```
    }
```

```
  );
```

```
}
```

```
}
```

Anexo 9: Índice aliados

Funcionamiento lógico:

El componente *AlliesPestCardsComponent* presenta tarjetas informativas sobre los aliados naturales en el control de plagas. Estas tarjetas probablemente contienen detalles como el nombre del aliado, su función en el ecosistema, y su impacto en el manejo de plagas.

Además, utiliza *RouterLink* para permitir la navegación hacia páginas con información más

detallada o específica sobre cada aliado. Este componente organiza la información de manera clara y visualmente atractiva para los usuarios.

Figura 15

Anexo ilustración *IndexPestsAlliesComponent.ts*

```
1  import { Component } from '@angular/core';
2  import { RouterLink } from '@angular/router';
3
4  @Component({
5    selector: 'app-allies-pest-cards',
6    standalone: true,
7    imports: [RouterLink],
8    templateUrl: './allies-pest-cards.component.html',
9    styleUrls: ['./allies-pest-cards.component.scss']
10  })
11  export class AlliesPestCardsComponent {
12
13  }
14
```

Código *IndexAllies*:

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

```
import { RouterLink } from '@angular/router';
```

```
@Component({
```

```
  selector: 'app-allies-pest-cards',
```

```
  standalone: true,
```

```
  imports: [RouterLink],
```

```
  templateUrl: './allies-pest-cards.component.html',
```

```
  styleUrls: ['./allies-pest-cards.component.scss']
```

```
})
```

```
export class AlliesPestCardsComponent {
```

}

Anexo 10: Chat de IA

Funcionamiento Lógico:

El componente ChatComponent implementa un sistema de chat interactivo donde los usuarios pueden enviar mensajes y recibir respuestas generadas automáticamente mediante la API de OpenAI. Los mensajes ingresados por el usuario se muestran en la interfaz junto con las respuestas generadas por el sistema.

Cuando el usuario envía un mensaje, este se agrega a una lista local de mensajes y se realiza una solicitud a la API de OpenAI para obtener la respuesta. La respuesta del bot se procesa y se muestra en la interfaz. Además, el campo de entrada se limpia automáticamente después de enviar el mensaje. Este componente utiliza FormsModule para el manejo del formulario de entrada y CommonModule para directivas como *ngFor y *ngIf, que controlan la visualización dinámica de los mensajes en la interfaz.

Figura 16

Anexo ilustración parte 1 ChatComponent.ts

```
1 import { Component } from '@angular/core';
2 import { FormsModule } from '@angular/forms'; // Para usar [(ngModel)]
3 import { CommonModule } from '@angular/common'; // Para directivas como *ngFor
4
5 @Component({
6   selector: 'app-chat',
7   templateUrl: './chat.component.html',
8   styleUrls: ['./chat.component.scss'],
9   standalone: true,
10  imports: [
11    FormsModule, // Para [(ngModel)]
12    CommonModule, // Para directivas estructurales (*ngFor, *ngIf)
13  ],
14 })
15 export class ChatComponent {
16   public messages: any[] = []; // Mensajes mostrados en la interfaz
17   public userInput: string = ''; // Entrada del usuario
18
19   async sendMessage(): Promise<void> {
20     // Agrega el mensaje del usuario a la lista
21     const userMessage = { text: this.userInput, author: 'user' };
22     this.messages.push(userMessage);
23
24     try {
25       // Realiza la solicitud a OpenAI usando fetch
26       const response = await fetch('https://api.openai.com/v1/chat/completions', {
27         method: 'POST',
28         headers: {
29           'Content-Type': 'application/json',
30           Authorization: `Bearer sk-proj-pM2lNVYqgP918RHm4BJHcguApqmCBO5bXLGfLIjV9vRR6LXCrkKiZXHX4-o6q8ESEMy8ZPQFMdT3B1`
31         },
32         body: JSON.stringify({
33           model: 'gpt-3.5-turbo', // Modelo de OpenAI
34           messages: [{ role: 'user', content: this.userInput }], // Mensaje del usuario
35           max_tokens: 150, // Máximo de tokens en la respuesta
36         }),
37       });
38     } catch {
39       // Manejo de errores
40     }
41   }
42 }
```

Figura 17

Anexo ilustración parte 2 ChatComponent.ts

```
38
39  ✓    if (!response.ok) {
40        throw new Error(`Error en la respuesta: ${response.status}`);
41    }
42
43    // Procesa la respuesta del bot
44    const data = await response.json();
45    const botMessage = { text: data.choices[0].message.content, author: 'bot' };
46    this.messages.push(botMessage);
47  ✓    } catch (error) {
48        console.error('Error al obtener respuesta del bot:', error);
49        this.messages.push({ text: 'Hubo un problema al obtener la respuesta.', author: 'bot' });
50    }
51
52    // Limpia el campo de entrada después de enviar el mensaje
53    this.userInput = '';
54  }
55  }
56
```

Código Chat IA:

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

```
import { FormsModule } from '@angular/forms'; // Para usar [(ngModel)]
```

```
import { CommonModule } from '@angular/common'; // Para directivas como *ngFor
```

```
@Component({
```

```
  selector: 'app-chat',
```

```
  templateUrl: './chat.component.html',
```

```
  styleUrls: ['./chat.component.scss'],
```

```
  standalone: true,
```

```
  imports: [
```

```
    FormsModule, // Para [(ngModel)]
```

```
    CommonModule, // Para directivas estructurales (*ngFor, *ngIf)
```

```
],
```



```
})

export class ChatComponent {

  public messages: any[] = []; // Mensajes mostrados en la interfaz

  public userInput: string = ""; // Entrada del usuario


  async sendMessage(): Promise<void> {

    // Agrega el mensaje del usuario a la lista

    const userMessage = { text: this.userInput, author: 'user' };

    this.messages.push(userMessage);


    try {

      // Realiza la solicitud a OpenAI usando fetch

      const response = await fetch('https://api.openai.com/v1/chat/completions', {

        method: 'POST',

        headers: {

          'Content-Type': 'application/json',

          Authorization: `Bearer sk-proj-

pM2INVYqgP918RHm4BJHcguApqmCBO5bXLGfLIjV9vRR6LXCrkKiZXHX4-

o6q8ESEMy8ZPQFMdT3BlbkFJYm5yw9_3wbLbIUc3Q0YeK2wZnORj43caYfb0oqwpUoGmDF

qJKo15pr5StKupPssJWSusIRqIkA`, // Clave API de OpenAI

        },

        body: JSON.stringify({

          model: 'gpt-3.5-turbo', // Modelo de OpenAI

          messages: [{ role: 'user', content: this.userInput }], // Mensaje del usuario

          max_tokens: 150, // Máximo de tokens en la respuesta

        })

      });

    } catch (error) {

      console.log(error);

    }

  }

}
```

```
});

if (!response.ok) {
  throw new Error(`Error en la respuesta: ${response.status}`);
}

// Procesa la respuesta del bot
const data = await response.json();
const botMessage = { text: data.choices[0].message.content, author: 'bot' };
this.messages.push(botMessage);
} catch (error) {
  console.error('Error al obtener respuesta del bot:', error);
  this.messages.push({ text: 'Hubo un problema al obtener la respuesta.', author: 'bot'
});
}

// Limpia el campo de entrada después de enviar el mensaje
this.userInput = "";
}
}
```

Anexo 11: Login

Funcionamiento Lógico:

El componente `LoginContent1Component` permite a los usuarios iniciar sesión en la plataforma utilizando su correo electrónico y contraseña. Utiliza `Firebase` para autenticar a los usuarios y consultar sus datos almacenados en `Firestore`.

Cuando un usuario intenta iniciar sesión, se valida su información mediante la función `signInWithEmailAndPassword`. Si la autenticación es exitosa, se recuperan los datos del usuario desde `Firestore`. Dependiendo del rol del usuario (administrador o usuario regular), se redirige a la página correspondiente utilizando el enrutador (`Router`). Además, el componente interactúa con un servicio de usuario (`UserService`) para establecer el usuario actual en la sesión. En caso de error, se muestra un mensaje indicando que las credenciales son incorrectas o que hubo un problema al procesar la solicitud.

Figura 18

Anexo ilustración parte 1 `LoginComponent.ts`

```
1 import { Component } from '@angular/core';
2 import { Auth, signInWithEmailAndPassword } from '@angular/fire/auth';
3 import { Firestore, doc, getDoc } from '@angular/fire/firestore';
4 import { Router } from '@angular/router';
5 import { FormsModule } from '@angular/forms';
6 import { UserService } from '../../administrador/services/user.service'; // Ruta relativa
7
8 @Component({
9   selector: 'app-login-content1',
10   standalone: true,
11   imports: [FormsModule],
12   templateUrl: './login-content1.component.html',
13   styleUrls: ['./login-content1.component.scss'],
14 })
15 export class LoginContent1Component {
16   email: string = '';
17   password: string = '';
18
19   constructor(
20     private auth: Auth,
21     private firestore: Firestore,
22     private router: Router,
23     private userService: UserService
24   ) {}
25
26   async login() {
27     try {
28       const userCredential = await signInWithEmailAndPassword(this.auth, this.email, this.password);
29       const user = userCredential.user;
30
31       if (user) {
32         const userDocRef = doc(this.firestore, `users/${user.uid}`);
33         const userDoc = await getDoc(userDocRef);
34
35         if (userDoc.exists()) {
36           const userData = userDoc.data();
37           this.userService.setCurrentUser(userData); // Establece el usuario actual
38         }
39       }
40     } catch (error) {
41       // Manejo de errores
42     }
43   }
44 }
```

Figura 19

Anexo ilustración parte 2 LoginComponent.ts

```
38
39      // Redirige según el rol del usuario
40      if (userData['role'] === 'admin') {
41        this.router.navigate(['/administrador']); // Redirige al dashboard si es admin
42      } else {
43        this.router.navigate(['/home']); // Redirige a la página principal si es usuario regular
44      }
45    } else {
46      alert('No se encontró información del usuario.');
```

Código Login:

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

import { Auth, signInWithEmailAndPassword } from '@angular/fire/auth';

import { Firestore, doc, getDoc } from '@angular/fire/firestore';

import { Router } from '@angular/router';

import { FormsModule } from '@angular/forms';

import { UserService } from '../administrador/services/user.service'; // Ruta relativa
```

```
@Component({
  selector: 'app-login-content1',
  standalone: true,
  imports: [FormsModule],
  templateUrl: './login-content1.component.html',
  styleUrls: ['./login-content1.component.scss'],
})
```

```
export class LoginContent1Component {
```



email: string = "";

password: string = "";

constructor(

private auth: Auth,

private firestore: Firestore,

private router: Router,

private userService: UserService

) {}

async login() {

try {

*const userCredential = await signInWithEmailAndPassword(this.auth, this.email,
this.password);*

const user = userCredential.user;

if (user) {

const userDocRef = doc(this.firestore, `users/\${user.uid}`);

const userDoc = await getDoc(userDocRef);

if (userDoc.exists()) {

const userData = userDoc.data();

this.userService.setCurrentUser(userData); // Establece el usuario actual

// Redirige según el rol del usuario

if (userData['role'] === 'admin') {

```
        this.router.navigate(['/administrador']); // Redirige al dashboard si es admin
    } else {
        this.router.navigate(['/home']); // Redirige a la página principal si es usuario
        regular
    }
} else {
    alert('No se encontró información del usuario.');
```

```
    }
}
} catch (error) {
    alert('Error al iniciar sesión. Verifica tus credenciales.');
```

```
    }
}
}
```

Anexo 12: Registro

Funcionamiento Lógico:

El componente ContentRegister1Component permite a los usuarios registrarse en la plataforma mediante un formulario que solicita su nombre completo, correo electrónico, y contraseña. Antes de proceder con el registro, valida que la contraseña cumpla con los requisitos mínimos y que coincida con su confirmación.

Al enviar el formulario, se utiliza Firebase para crear una cuenta de usuario mediante `createUserWithEmailAndPassword`. Una vez creada la cuenta, los datos adicionales del usuario (como su nombre completo y rol predeterminado) se almacenan en Firestore. Tras un registro exitoso, el usuario es notificado y redirigido a la página de inicio de sesión. En caso de error, se

muestra un mensaje explicativo. Este componente asegura una experiencia intuitiva y segura para el registro de nuevos usuarios.

Figura 20

Anexo ilustración parte 1 RegisterComponent.ts

```
1 import { Component } from '@angular/core';
2 import { Auth, createUserWithEmailAndPassword } from '@angular/fire/auth';
3 import { Firestore, doc, setDoc } from '@angular/fire/firestore';
4 import { Router } from '@angular/router';
5 import { FormsModule } from '@angular/forms'; // Importar FormsModule aquí
6
7 @Component({
8   selector: 'app-content-register1',
9   standalone: true,
10  imports: [FormsModule], // Asegúrate de que FormsModule está aquí
11  templateUrl: './content-register1.component.html',
12  styleUrls: ['./content-register1.component.scss'],
13 })
14 export class ContentRegister1Component {
15   fullName: string = '';
16   email: string = '';
17   password: string = '';
18   confirmPassword: string = '';
19
20   constructor(private auth: Auth, private firestore: Firestore, private router: Router) {}
21
22   async register() {
23     // Validación de contraseñas
24     if (this.password.length < 6) {
25       alert('La contraseña debe tener al menos 6 caracteres');
26       return;
27     }
28     if (this.password !== this.confirmPassword) {
29       alert('Las contraseñas no coinciden');
30       return;
31     }
32
33     try {
34       const userCredential = await createUserWithEmailAndPassword(this.auth, this.email, this.password);
35       const user = userCredential.user;
36
37       // Guardar datos adicionales en Firestore
```

Figura 21

Anexo ilustración parte 2 RegisterComponent.ts

```
38  await setDoc(doc(this.firestore, 'users', user.uid), {
39      fullName: this.fullName,
40      email: this.email,
41      uid: user.uid,
42      role: 'user' // Aquí defines el rol predeterminado
43  });
44
45  alert('Usuario registrado exitosamente');
46  this.router.navigate(['/login']);
47  } catch (error) {
48      console.error('Error al registrar el usuario:', error);
49      alert('Error al registrar el usuario');
50  }
51  }
52  }
53
```

Código Registro:

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

import { Auth, createUserWithEmailAndPassword } from '@angular/fire/auth';

import { Firestore, doc, setDoc } from '@angular/fire/firestore';

import { Router } from '@angular/router';

import { FormsModule } from '@angular/forms'; // Importar FormsModule aquí
```

```
@Component({

  selector: 'app-content-register1',

  standalone: true,

  imports: [FormsModule], // Asegúrate de que FormsModule está aquí

  templateUrl: './content-register1.component.html',

  styleUrls: ['./content-register1.component.scss'],

})
```



```
export class ContentRegister1Component {  
  
  fullName: string = "";  
  
  email: string = "";  
  
  password: string = "";  
  
  confirmPassword: string = "";  
  
  constructor(private auth: Auth, private firestore: Firestore, private router: Router) {}  
  
  async register() {  
  
    // Validación de contraseñas  
  
    if (this.password.length < 6) {  
  
      alert('La contraseña debe tener al menos 6 caracteres');  
  
      return;  
  
    }  
  
    if (this.password !== this.confirmPassword) {  
  
      alert('Las contraseñas no coinciden');  
  
      return;  
  
    }  
  
    try {  
  
      const userCredential = await createUserWithEmailAndPassword(this.auth,  
this.email, this.password);  
  
      const user = userCredential.user;  
  
  
      // Guardar datos adicionales en Firestore  
  
      await setDoc(doc(this.firestore, 'users', user.uid), {
```

```
fullName: this.fullName,  
email: this.email,  
uid: user.uid,  
role: 'user' // Aquí defines el rol predeterminado  
});  
  
alert('Usuario registrado exitosamente');  
this.router.navigate(['/login']);  
} catch (error) {  
  console.error('Error al registrar el usuario:', error);  
  alert('Error al registrar el usuario');  
}  
}  
}
```

Anexo 13: Dashboard-administrador

Funcionamiento Lógico:

El componente AdminDashboardComponent gestiona las operaciones administrativas de la plataforma, como la administración de usuarios, plagas y aliados biológicos. Proporciona una interfaz para agregar, editar y eliminar elementos en cada categoría, utilizando formularios y modales para facilitar estas acciones. También permite cambiar entre pestañas para navegar entre las diferentes funcionalidades.

El componente carga los datos de usuarios, plagas y aliados al inicializarse (ngOnInit) y utiliza servicios específicos para interactuar con la base de datos. Cada operación (como agregar un usuario o actualizar una plaga) valida los datos antes de enviarlos, asegurando integridad y consistencia en la información almacenada. Los modales se utilizan para mantener

la experiencia de usuario limpia y organizada, mientras que las listas dinámicas se actualizan automáticamente después de cada acción.

Figura 22

Anexo ilustración parte 1 DashboardComponent.ts

```
1 import { Component, OnInit } from '@angular/core';
2 import { FormsModule, ReactiveFormsModule } from '@angular/forms';
3 import { UserService } from '../services/user.service';
4 import { User } from '../services/user.model';
5 import { PlagaService } from '../services/plaga.service';
6 import { Plaga } from '../services/plaga.model';
7 import { AliadoService } from '../services/aliado.service';
8 import { Aliado } from '../services/aliado.model';
9 import { CommonModule } from '@angular/common';
10
11 @Component({
12   selector: 'app-administrador-dashboard',
13   templateUrl: './administrador-dashboard.component.html',
14   styleUrls: ['./administrador-dashboard.component.scss'],
15   standalone: true,
16   imports: [FormsModule, ReactiveFormsModule, CommonModule],
17 })
18 export class AdminDashboardComponent implements OnInit {
19   activeTab: string = 'usuarios';
20
21   // Datos
22   users: User[] = [];
23   plagas: Plaga[] = [];
24   aliados: Aliado[] = [];
25
26   // Modales
27   isAddUserModalOpen: boolean = false;
28   isAddPlagaModalOpen: boolean = false;
29   isEditPlagaModalOpen: boolean = false;
30   isAddAliadoModalOpen: boolean = false;
31   isEditAliadoModalOpen: boolean = false;
32
33   // Formularios
34   newUser: Partial<User & { password?: string }> = { fullName: '', email: '', role: 'user', password: '' };
35   newPlaga: Partial<Plaga> = { nombre: '', descripcion: '', características: '', danos: '', combate: '', cultivo: '' };
36   selectedPlaga: Partial<Plaga> = {};
37   newAliado: Partial<Aliado> = { nombre: '', descripcion: '', características: '', apoyar: '', beneficios: '', cultivo: '' };
```

Figura 23

Anexo ilustración parte 2 DashboardComponent.ts

```
38 selectedAliado: Partial<Aliado> = {};  
39  
40 // Lista de cultivos disponibles  
41 cultivos: string[] = ['Maíz', 'Tomate', 'Calabaza', 'Papaya'];  
42  
43 constructor(  
44   private userService: UserService,  
45   private plagaService: PlagaService,  
46   private aliadoService: AliadoService  
47 ) {}  
48  
49 ngOnInit(): void {  
50   this.loadUsers();  
51   this.loadPlagas();  
52   this.loadAliados();  
53 }  
54  
55 // Cambiar Tabs  
56 setActiveTab(tab: string): void {  
57   this.activeTab = tab;  
58 }  
59  
60 // Cargar Datos  
61 private loadUsers(): void {  
62   this.userService.getUsers().subscribe(  
63     (users) => (this.users = users),  
64     (error) => console.error('Error al obtener usuarios:', error)  
65   );  
66 }  
67  
68 private loadPlagas(): void {  
69   this.plagaService.getPlagas().subscribe(  
70     (plagas) => (this.plagas = plagas),  
71     (error) => console.error('Error al obtener plagas:', error)
```

Figura 24

Anexo ilustración parte 3 DashboardComponent.ts

```
72 | );  
73 | }  
74 |  
75 | private loadAliados(): void {  
76 |   this.aliadoService.getAliados().subscribe(  
77 |     (aliados) => (this.aliados = aliados),  
78 |     (error) => console.error('Error al obtener aliados:', error)  
79 |   );  
80 | }  
81 |  
82 | // Métodos para Usuarios  
83 | openAddUserModal(): void {  
84 |   this.isAddUserModalOpen = true;  
85 | }  
86 |  
87 | closeAddUserModal(): void {  
88 |   this.isAddUserModalOpen = false;  
89 |   this.resetForms();  
90 | }  
91 |  
92 | onRoleChange(userId: string | undefined, newRole: string): void {  
93 |   if (userId) {  
94 |     this.userService.updateUserRole(userId, newRole).then(  
95 |       () => console.log('Rol actualizado exitosamente'),  
96 |       (error) => console.error('Error al actualizar el rol:', error)  
97 |     );  
98 |   } else {  
99 |     console.error('Error: userId no definido');  
100 |   }  
101 | }  
102 |  
103 | addUser(): void {  
104 |   if (this.newUser.fullName && this.newUser.email && this.newUser.password) {  
105 |     this.userService.addUser(this.newUser as User).then(  

```

Figura 25

Anexo ilustración parte 4 DashboardComponent.ts

```
106  >      () => {
107      >          console.log('Usuario agregado exitosamente');
108      >          this.loadUsers();
109      >          this.closeAddUserModal();
110      >      },
111      >      (error) => console.error('Error al agregar usuario:', error)
112      >  );
113  >  } else {
114  >      alert('Todos los campos son requeridos');
115  >  }
116  >  }
117  >
118  >  deleteUser(userId: string | undefined): void {
119  >      if (userId) {
120  >          if (confirm('¿Estás seguro de que deseas eliminar este usuario?')) {
121  >              this.userService.deleteUser(userId).then(
122  >                  () => {
123  >                      console.log('Usuario eliminado exitosamente');
124  >                      this.loadUsers();
125  >                  },
126  >                  (error) => console.error('Error al eliminar usuario:', error)
127  >              );
128  >          }
129  >      } else {
130  >          console.error('Error: userId no definido');
131  >      }
132  >  }
133  >
134  >  // Métodos para Plagas
135  >  openAddPlagaModal(): void {
136  >      this.selectedPlaga = {}; // Resetea el formulario
137  >      this.isAddPlagaModalOpen = true;
138  >  }
139  >
140  >  closeAddPlagaModal(): void {
```

Figura 26

Anexo ilustración parte 5 DashboardComponent.ts

```
143 }
144
145 addPlaga(): void {
146   if (this.newPlaga.nombre && this.newPlaga.descripcion && this.newPlaga.cultivo) {
147     this.plagaService.addPlaga(this.newPlaga as Plaga).then(
148       () => {
149         console.log('Plaga agregada exitosamente');
150         this.loadPlagas(); // Actualizar la lista de plagas
151         this.closeAddPlagaModal(); // Cerrar el modal
152       },
153       (error) => console.error('Error al agregar plaga:', error)
154     );
155   } else {
156     alert('Todos los campos son requeridos');
157   }
158 }
159
160 openEditPlagaModal(plaga: Plaga): void {
161   this.isEditPlagaModalOpen = true;
162   this.selectedPlaga = { ...plaga }; // Asegúrate de que selectedPlaga tenga los valores correctos
163 }
164
165 closeEditPlagaModal(): void {
166   this.isEditPlagaModalOpen = false;
167   this.resetForms();
168 }
169
170 updatePlaga(): void {
171   if (this.selectedPlaga.id) {
172     this.plagaService.updatePlaga(this.selectedPlaga.id, this.selectedPlaga as Plaga).then(
173       () => {
174         console.log('Plaga actualizada exitosamente');
175         this.loadPlagas();
176         this.closeEditPlagaModal();
177       }
178     );
179   }
180 }
```

Figura 27

Anexo ilustración parte 6 DashboardComponent.ts

```
177     },
178     (error) => console.error('Error al actualizar plaga:', error)
179   );
180   }
181 }
182
183 deletePlaga(plagaId: string | undefined): void {
184   if (plagaId) {
185     if (confirm('¿Estás seguro de que deseas eliminar esta plaga?')) {
186       this.plagaService.deletePlaga(plagaId).then(
187         () => {
188           console.log('Plaga eliminada exitosamente');
189           this.loadPlagas();
190         },
191         (error) => console.error('Error al eliminar plaga:', error)
192       );
193     }
194   } else {
195     console.error('Error: plagaId no definido');
196   }
197 }
198
199 // Métodos para Aliados
200 openAddAliadoModal(): void {
201   this.selectedAliado = {}; // Resetea el formulario
202   this.isAddAliadoModalOpen = true;
203 }
204
205 closeAddAliadoModal(): void {
206   this.isAddAliadoModalOpen = false;
207   this.resetForms();
208 }
209
210 addAliado(): void {
211   if (this.newAliado.nombre && this.newAliado.descripcion && this.newAliado.cultivo) {
```

Figura 28

Anexo ilustración parte 7 DashboardComponent.ts

```
18  export class AdminDashboardComponent implements OnInit {
210  addAliado(): void {
213    () => {
217      },
218      (error) => console.error('Error al agregar aliado:', error)
219    );
220  } else {
221    alert('Todos los campos son requeridos');
222  }
223  }
224
225  openEditAliadoModal(aliado: Aliado): void {
226    this.isEditAliadoModalOpen = true;
227    this.selectedAliado = { ...aliado }; // Asegúrate de que selectedAliado tenga los valores correctos
228  }
229
230  closeEditAliadoModal(): void {
231    this.isEditAliadoModalOpen = false;
232    this.resetForms();
233  }
234
235  updateAliado(): void {
236    if (this.selectedAliado.id) {
237      this.aliadoService.updateAliado(this.selectedAliado.id, this.selectedAliado as Aliado).then(
238        () => {
239          console.log('Aliado actualizado exitosamente');
240          this.loadAliados();
241          this.closeEditAliadoModal();
242        },
243        (error) => console.error('Error al actualizar aliado:', error)
244      );
245    }
246  }
247
248  deleteAliado(aliadoId: string | undefined): void {
249    if (aliadoId) {
250      if (confirm('¿Estás seguro de que deseas eliminar este aliado?')) {
```

Figura 29

Anexo ilustración parte 8 DashboardComponent.ts

```
251     this.aliadoService.deleteAliado(aliadoId).then(  
252         () => {  
253             console.log('Aliado eliminado exitosamente');  
254             this.loadAliados();  
255         },  
256         (error) => console.error('Error al eliminar aliado:', error)  
257     );  
258     }  
259     } else {  
260         console.error('Error: aliadoId no definido');  
261     }  
262 }  
263  
264 private resetForms(): void {  
265     this.newUser = { fullName: '', email: '', role: 'user', password: '' };  
266     this.newPlaga = { nombre: '', descripcion: '', características: '', danos: '', combate: '', cultivo: '' };  
267     this.selectedPlaga = {};  
268     this.newAliado = { nombre: '', descripcion: '', características: '', apoyar: '', beneficios: '', cultivo: '' };  
269     this.selectedAliado = {};  
270 }  
271 }  
272
```

Código Dashboard-Administrador:

```
import { Component, OnInit } from '@angular/core';  
  
import { FormsModule, ReactiveFormsModule } from '@angular/forms';  
  
import { UserService } from '../services/user.service';  
  
import { User } from '../services/user.model';  
  
import { PlagaService } from '../services/plaga.service';  
  
import { Plaga } from '../services/plaga.model';  
  
import { AliadoService } from '../services/aliado.service';  
  
import { Aliado } from '../services/aliado.model';  
  
import { CommonModule } from '@angular/common';  
  
@Component({  
    selector: 'app-administrador-dashboard',  
    templateUrl: './administrador-dashboard.component.html',  
    styleUrls: ['./administrador-dashboard.component.scss'],  
})
```



```
standalone: true,  
  
imports: [FormsModule, ReactiveFormsModule, CommonModule],  
  
})  
  
export class AdminDashboardComponent implements OnInit {  
  
  activeTab: string = 'usuarios';  
  
  
  
  // Datos  
  
  users: User[] = [];  
  
  plagas: Plaga[] = [];  
  
  aliados: Aliado[] = [];  
  
  
  
  // Modales  
  
  isAddUserModalOpen: boolean = false;  
  
  isAddPlagaModalOpen: boolean = false;  
  
  isEditPlagaModalOpen: boolean = false;  
  
  isAddAliadoModalOpen: boolean = false;  
  
  isEditAliadoModalOpen: boolean = false;  
  
  
  
  // Formularios  
  
  newUser: Partial<User & { password?: string }> = { fullName: "", email: "", role: 'user',  
password: "" };  
  
  newPlaga: Partial<Plaga> = { nombre: "", descripcion: "", características: "", danos: "",  
combate: "", cultivo: "" };  
  
  selectedPlaga: Partial<Plaga> = {};  
  
  newAliado: Partial<Aliado> = { nombre: "", descripcion: "", características: "", apoyar: "",  
beneficios: "", cultivo: "" };
```



```
selectedAliado: Partial<Aliado> = {};
```

```
// Lista de cultivos disponibles
```

```
cultivos: string[] = ['Maíz', 'Tomate', 'Calabaza', 'Papaya'];
```

```
constructor(
```

```
    private userService: UserService,
```

```
    private plagaService: PlagaService,
```

```
    private aliadoService: AliadoService
```

```
) {}
```

```
ngOnInit(): void {
```

```
    this.loadUsers();
```

```
    this.loadPlagas();
```

```
    this.loadAliados();
```

```
}
```

```
// Cambiar Tabs
```

```
setActiveTab(tab: string): void {
```

```
    this.activeTab = tab;
```

```
}
```

```
// Cargar Datos
```

```
private loadUsers(): void {
```

```
    this.userService.getUsers().subscribe(
```

```
        (users) => (this.users = users),
```





```
(error) => console.error('Error al obtener usuarios:', error)

);

}
```

```
private loadPlagas(): void {

  this.plagaService.getPlagas().subscribe(

    (plagas) => (this.plagas = plagas),

    (error) => console.error('Error al obtener plagas:', error)

  );

}
```

```
private loadAliados(): void {

  this.aliadoService.getAliados().subscribe(

    (aliados) => (this.aliados = aliados),

    (error) => console.error('Error al obtener aliados:', error)

  );

}
```

```
// Métodos para Usuarios

openAddUserModal(): void {

  this.isAddUserModalOpen = true;

}
```

```
closeAddUserModal(): void {

  this.isAddUserModalOpen = false;

  this.resetForms();

}
```

}

```
onRoleChange(userId: string | undefined, newRole: string): void {
```

```
  if (userId) {
```

```
    this.userService.updateUserRole(userId, newRole).then(
```

```
      () => console.log('Rol actualizado exitosamente'),
```

```
      (error) => console.error('Error al actualizar el rol:', error)
```

```
    );
```

```
  } else {
```

```
    console.error('Error: userId no definido');
```

```
  }
```

```
}
```

```
addUser(): void {
```

```
  if (this.newUser.fullName && this.newUser.email && this.newUser.password) {
```

```
    this.userService.addUser(this.newUser as User).then(
```

```
      () => {
```

```
        console.log('Usuario agregado exitosamente');
```

```
        this.loadUsers();
```

```
        this.closeAddUserModal();
```

```
      },
```

```
      (error) => console.error('Error al agregar usuario:', error)
```

```
    );
```

```
  } else {
```

```
    alert('Todos los campos son requeridos');
```

```
}
```



}

```
deleteUser(userId: string | undefined): void {  
  if (userId) {  
    if (confirm('¿Estás seguro de que deseas eliminar este usuario?')) {  
      this.userService.deleteUser(userId).then(  
        () => {  
          console.log('Usuario eliminado exitosamente');  
          this.loadUsers();  
        },  
        (error) => console.error('Error al eliminar usuario:', error)  
      );  
    }  
  } else {  
    console.error('Error: userId no definido');  
  }  
}
```

// Métodos para Plagas

```
openAddPlagaModal(): void {  
  this.selectedPlaga = {}; // Resetea el formulario  
  this.isAddPlagaModalOpen = true;  
}
```

```
closeAddPlagaModal(): void {  
  this.isAddPlagaModalOpen = false;
```

```
this.resetForms();  
  
}  
  
addPlaga(): void {  
  
  if (this.newPlaga.nombre && this.newPlaga.descripcion && this.newPlaga.cultivo) {  
  
    this.plagaService.addPlaga(this.newPlaga as Plaga).then(  
  
      () => {  
  
        console.log('Plaga agregada exitosamente');  
  
        this.loadPlagas(); // Actualizar la lista de plagas  
  
        this.closeAddPlagaModal(); // Cerrar el modal  
  
      },  
  
      (error) => console.error('Error al agregar plaga:', error)  
  
    );  
  
  } else {  
  
    alert('Todos los campos son requeridos');  
  
  }  
  
}  
  
  
openEditPlagaModal(plaga: Plaga): void {  
  
  this.isEditPlagaModalOpen = true;  
  
  this.selectedPlaga = { ...plaga }; // Asegúrate de que selectedPlaga tenga los valores  
correctos  
  
}  
  
  
closeEditPlagaModal(): void {  
  
  this.isEditPlagaModalOpen = false;
```



```
this.resetForms();
```

```
}
```

```
updatePlaga(): void {
```

```
  if (this.selectedPlaga.id) {
```

```
    this.plagaService.updatePlaga(this.selectedPlaga.id, this.selectedPlaga as  
Plaga).then(
```

```
      () => {
```

```
        console.log('Plaga actualizada exitosamente');
```

```
        this.loadPlagas();
```

```
        this.closeEditPlagaModal();
```

```
      },
```

```
      (error) => console.error('Error al actualizar plaga:', error)
```

```
    );
```

```
  }
```

```
}
```

```
deletePlaga(plagaId: string | undefined): void {
```

```
  if (plagaId) {
```

```
    if (confirm('¿Estás seguro de que deseas eliminar esta plaga?')) {
```

```
      this.plagaService.deletePlaga(plagaId).then(
```

```
        () => {
```

```
          console.log('Plaga eliminada exitosamente');
```

```
          this.loadPlagas();
```

```
        },
```

```
        (error) => console.error('Error al eliminar plaga:', error)
```



```
);  
}  
} else {  
    console.error('Error: plagald no definido');  
}  
}
```

// Métodos para Aliados

```
openAddAliadoModal(): void {  
    this.selectedAliado = {}; // Resetea el formulario  
    this.isAddAliadoModalOpen = true;  
}
```

```
closeAddAliadoModal(): void {  
    this.isAddAliadoModalOpen = false;  
    this.resetForms();  
}
```

```
addAliado(): void {  
    if (this.newAliado.nombre && this.newAliado.descripcion && this.newAliado.cultivo) {  
        this.aliadoService.addAliado(this.newAliado as Aliado).then(  
            () => {  
                console.log('Aliado agregado exitosamente');  
                this.loadAliados(); // Actualizar la lista de aliados  
                this.closeAddAliadoModal(); // Cerrar el modal  
            },
```



```
(error) => console.error('Error al agregar aliado:', error)

);
} else {
    alert('Todos los campos son requeridos');
}
}

openEditAliadoModal(aliado: Aliado): void {
    this.isEditAliadoModalOpen = true;
    this.selectedAliado = { ...aliado }; // Asegúrate de que selectedAliado tenga los valores
correctos
}

closeEditAliadoModal(): void {
    this.isEditAliadoModalOpen = false;
    this.resetForms();
}

updateAliado(): void {
    if (this.selectedAliado.id) {
        this.aliadoService.updateAliado(this.selectedAliado.id, this.selectedAliado as
Aliado).then(
            () => {
                console.log('Aliado actualizado exitosamente');
                this.loadAliados();
                this.closeEditAliadoModal();
            }
        );
    }
}
```



```
    },  
    (error) => console.error('Error al actualizar aliado:', error)  
  );  
}  
}  
  
deleteAliado(aliadold: string | undefined): void {  
  if (aliadold) {  
    if (confirm('¿Estás seguro de que deseas eliminar este aliado?')) {  
      this.aliadoService.deleteAliado(aliadold).then(  
        () => {  
          console.log('Aliado eliminado exitosamente');  
          this.loadAliados();  
        },  
        (error) => console.error('Error al eliminar aliado:', error)  
      );  
    }  
  } else {  
    console.error('Error: aliadold no definido');  
  }  
}
```

```
private resetForms(): void {  
  this.newUser = { fullName: "", email: "", role: 'user', password: "" };  
  this.newPlaga = { nombre: "", descripcion: "", características: "", danos: "", combate: "",  
cultivo: "" };  
}
```

```
this.selectedPlaga = {};  
  
this.newAliado = { nombre: "", descripcion: "", características: "", apoyar: "", beneficios: "",  
cultivo: "" };  
  
this.selectedAliado = {};  
  
}  
  
}
```

Anexo 14: Chat.service.ts

Funcionamiento Lógico:

El servicio ChatService permite la comunicación con la API de OpenAI para generar respuestas en un sistema de chat. Este servicio encapsula la lógica para realizar solicitudes HTTP, simplificando la interacción desde los componentes que lo utilizan.

El método getResponse envía un mensaje del usuario (definido por el parámetro prompt) a la API, utilizando un encabezado de autorización que incluye la clave de la API. También especifica el modelo a utilizar (gpt-3.5-turbo) y el límite de tokens en las respuestas. Este diseño facilita la reutilización del servicio en diferentes partes de la aplicación para generar respuestas dinámicas de manera eficiente.

Figura 30

Anexo ilustración chat.service.ts

```
1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders } from '@angular/common/http';
3 import { Observable } from 'rxjs';
4
5 @Injectable({
6   providedIn: 'root',
7 })
8 export class ChatService {
9   private apiUrl = '/api/v1/chat/completions'; // Apunta al proxy
10
11   private apiKey = 'sk-sk-proj-sk-proj-qTuyKuXnbXk_S4hpgiqdMKdvREq7Q9T4AoG1FH1Sw5pHp8TV3Q4bnndJh449eEl1c8VOBaYgpdT3B1b';
12
13   constructor(private http: HttpClient) {}
14
15   getResponse(prompt: string): Observable<any> {
16     const headers = new HttpHeaders({
17       'Content-Type': 'application/json',
18       Authorization: `Bearer ${this.apiKey}`,
19     });
20
21     const body = {
22       model: 'gpt-3.5-turbo',
23       messages: [{ role: 'user', content: prompt }],
24       max_tokens: 150,
25     };
26
27     return this.http.post(this.apiUrl, body, { headers });
28   }
29 }
30
```

Código ChatService:

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

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root',
})

export class ChatService {

  private apiUrl = '/api/v1/chat/completions'; // Apunta al proxy
```



```
private apiKey = 'sk-sk-proj-sk-proj-  
qTuyKuXnbXk_S4hpgiqdMKdvREq7Q9T4AoG1FH1Sw5pHp8TV3Q4bnndJh449eEl1c8VOBaY  
gpdT3BlbkFJTWoyQC8--  
qGTvyMJjUyQ4DHXYmUU91FITtTcRp6ayr0baRm3Vef2A8tVWzm2yWx118T2289Y0A-  
oj83n2vtT3BlbkFJcqOPlw8tq4uB6A9EQ1DOeue6cy_6aqGVB4AuriESSKsK2J9uXILDVUmhhHt  
dDhuZWKGW7GD-IA'; // Clave API válida
```

```
constructor(private http: HttpClient) {}
```

```
getResponse(prompt: string): Observable<any> {
```

```
    const headers = new HttpHeaders({  
        'Content-Type': 'application/json',  
        Authorization: `Bearer ${this.apiKey}`,  
    });
```

```
    const body = {  
        model: 'gpt-3.5-turbo',  
        messages: [{ role: 'user', content: prompt }],  
        max_tokens: 150,  
    };
```

```
    return this.http.post(this.apiUrl, body, { headers });
```

```
    }  
}
```

Anexo 15: Chat.service.spect.ts

Funcionamiento Lógico:

Este archivo implementa una prueba básica para el servicio ChatService, asegurando que se crea correctamente dentro del entorno de pruebas. Utiliza el framework de pruebas de Angular para configurar un módulo de prueba (TestBed) y verificar que el servicio se inicializa sin problemas.

La prueba principal (it('should be created')) utiliza una aserción para confirmar que la instancia del servicio existe (Truthy). Este tipo de prueba es útil para garantizar que la configuración y la inyección de dependencias del servicio funcionan como se espera. Es una base inicial para añadir pruebas más detalladas en el futuro.

Figura 31

Anexo ilustre Chat.service.spect.ts

```
1  import { TestBed } from '@angular/core/testing';
2
3  import { ChatService } from './chat.service';
4
5  describe('ChatService', () => {
6    let service: ChatService;
7
8    beforeEach(() => {
9      TestBed.configureTestingModule({});
10     service = TestBed.inject(ChatService);
11   });
12
13   it('should be created', () => {
14     expect(service).toBeTruthy();
15   });
16 });
17
```

Código chatservicespect:

```
import { TestBed } from '@angular/core/testing';
```

```
import { ChatService } from './chat.service';
```

```
describe('ChatService', () => {
```

```
  let service: ChatService;
```

```
  beforeEach(() => {
```

```
    TestBed.configureTestingModule({
```

```
      service = TestBed.inject(ChatService);
```

```
    });
```

```
    it('should be created', () => {
```

```
      expect(service).toBeTruthy();
```

```
    });
```

```
  });
```

Anexo 16: User.service.ts

Funcionamiento Lógico:

El servicio UserService gestiona la interacción con la base de datos de usuarios en Firebase y el estado de autenticación. Proporciona métodos para manejar el estado del usuario actual, así como operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en la colección de usuarios.

El estado del usuario actual se maneja mediante un BehaviorSubject, que se actualiza automáticamente cuando cambian las credenciales en Firebase (onAuthStateChanged). Este servicio permite cerrar sesión, establecer manualmente el usuario actual, obtener la lista de usuarios, actualizar roles, agregar nuevos usuarios y eliminar usuarios existentes. Es esencial

para centralizar la gestión de usuarios y facilitar la comunicación entre diferentes partes de la aplicación.

Figura 32

Anexo ilustración parte 1 User.service.ts

```
1 import { Injectable } from '@angular/core';
2 import { Firestore, doc, updateDoc, collection, collectionData, deleteDoc, addDoc, getDoc } from '@angular/fire/firestore';
3 import { Observable, BehaviorSubject } from 'rxjs';
4 import { Auth, onAuthStateChanged, signInOut } from '@angular/fire/auth';
5
6 @Injectable({
7   providedIn: 'root',
8 })
9 export class UserService {
10   private currentUserSubject = new BehaviorSubject<any | null>(null); // Estado del usuario actual
11   public currentUser$ = this.currentUserSubject.asObservable(); // Observable para suscribirse al estado del usuario
12
13   constructor(private firestore: Firestore, private auth: Auth) {
14     // Escuchar cambios en el estado de autenticación de Firebase
15     onAuthStateChanged(this.auth, async (user) => {
16       if (user) {
17         const userDocRef = doc(this.firestore, `users/${user.uid}`);
18         const userDoc = await getDoc(userDocRef);
19
20         if (userDoc.exists()) {
21           this.currentUserSubject.next(userDoc.data()); // Actualizar el estado del usuario actual
22         }
23       } else {
24         this.currentUserSubject.next(null); // Si no hay usuario, reseteamos el estado
25       }
26     });
27   }
28
29   // Método para cerrar sesión
30   async logout(): Promise<void> {
31     await signInOut(this.auth);
32     this.currentUserSubject.next(null); // Resetear el estado del usuario
33   }
34
35   // Método para establecer manualmente el usuario actual
36   setCurrentUser(userData: any): void {
37     this.currentUserSubject.next(userData); // Actualiza el BehaviorSubject con los datos del usuario
38   }
39 }
```

Figura 33

Anexo ilustracio parte 2 User.service.ts

```
38     }
39
40     // Método para obtener todos los usuarios
41     getUsers(): Observable<any[]> {
42         const usersRef = collection(this.firestore, 'users');
43         return collectionData(usersRef, { idField: 'id' }) as Observable<any[]>;
44     }
45
46     // Método para actualizar el rol del usuario
47     updateUserRole(userId: string, newRole: string): Promise<void> {
48         const userDocRef = doc(this.firestore, `users/${userId}`);
49         return updateDoc(userDocRef, { role: newRole });
50     }
51
52     // Método para eliminar un usuario
53     deleteUser(userId: string): Promise<void> {
54         const userDocRef = doc(this.firestore, `users/${userId}`);
55         return deleteDoc(userDocRef);
56     }
57
58     // Método para añadir un nuevo usuario
59     addUser(newUser: any): Promise<void> {
60         const usersRef = collection(this.firestore, 'users');
61         return addDoc(usersRef, newUser).then(() => {});
62     }
63 }
64
```

Código UserService:

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

import { Firestore, doc, updateDoc, collection, collectionData, deleteDoc, addDoc,
getDoc } from '@angular/fire/firestore';

import { Observable, BehaviorSubject } from 'rxjs';

import { Auth, onAuthStateChanged, signOut } from '@angular/fire/auth';

@Injectable({
```



```
providedIn: 'root',  
  
})  
  
export class UserService {  
  
  private currentUserSubject = new BehaviorSubject<any | null>(null); // Estado del  
usuario actual  
  
  public currentUser$ = this.currentUserSubject.asObservable(); // Observable para  
suscribirse al estado del usuario  
  
  constructor(private firestore: Firestore, private auth: Auth) {  
  
    // Escuchar cambios en el estado de autenticación de Firebase  
    onAuthStateChanged(this.auth, async (user) => {  
      if (user) {  
        const userDocRef = doc(this.firestore, `users/${user.uid}`);  
        const userDoc = await getDoc(userDocRef);  
  
        if (userDoc.exists()) {  
          this.currentUserSubject.next(userDoc.data()); // Actualizar el estado del usuario  
actual  
        }  
      } else {  
        this.currentUserSubject.next(null); // Si no hay usuario, reseteamos el estado  
      }  
    });  
  }  
  
  // Método para cerrar sesión
```



```
async logout(): Promise<void> {
```

```
  await signOut(this.auth);
```

```
  this.currentUserSubject.next(null); // Resetear el estado del usuario
```

```
}
```

```
// Método para establecer manualmente el usuario actual
```

```
setCurrentUser(userData: any): void {
```

```
  this.currentUserSubject.next(userData); // Actualiza el BehaviorSubject con los datos  
del usuario
```

```
}
```

```
// Método para obtener todos los usuarios
```

```
getUsers(): Observable<any[]> {
```

```
  const usersRef = collection(this.firestore, 'users');
```

```
  return collectionData(usersRef, { idField: 'id' }) as Observable<any[]>;
```

```
}
```

```
// Método para actualizar el rol del usuario
```

```
updateUserRole(userId: string, newRole: string): Promise<void> {
```

```
  const userDocRef = doc(this.firestore, `users/${userId}`);
```

```
  return updateDoc(userDocRef, { role: newRole });
```

```
}
```

```
// Método para eliminar un usuario
```

```
deleteUser(userId: string): Promise<void> {
```

```
  const userDocRef = doc(this.firestore, `users/${userId}`);
```

```
return deleteDoc(userDocRef);  
  
}  
  
// Método para añadir un nuevo usuario  
addUser(newUser: any): Promise<void> {  
  const usersRef = collection(this.firestore, 'users');  
  return addDoc(usersRef, newUser).then(() => {});  
}  
}
```

Anexo 17: Plagas.service.ts

Funcionamiento Lógico:

El servicio *PlagaService* permite gestionar las plagas en la base de datos. Proporciona métodos para obtener la lista completa de plagas (*getPlagas*) o una plaga específica por su ID (*getPlagaById*). También permite agregar nuevas plagas (*addPlaga*), actualizar datos existentes (*updatePlaga*), y eliminar registros (*deletePlaga*). Estos métodos interactúan directamente con *Firestore*, manejando las operaciones de forma eficiente y asegurando que los datos estén siempre actualizados.

Figura 34

Anexo ilustración parte1 Plagas.service.ts

```
1 import { Injectable } from '@angular/core';
2 import { Firestore, doc, updateDoc, collection, collectionData, deleteDoc, addDoc, DocumentReference, docData } from '@angular/fire';
3 import { Observable } from 'rxjs';
4 import { Plaga } from './plaga.model';
5
6 @Injectable({
7   providedIn: 'root',
8 })
9 export class PlagaService {
10   constructor(private firestore: Firestore) {}
11
12   // Obtener todas las plagas
13   getPlagas(): Observable<Plaga[]> {
14     const plagasRef = collection(this.firestore, 'plagas');
15     return collectionData(plagasRef, { idField: 'id' }) as Observable<Plaga[]>;
16   }
17
18   // Obtener una plaga por su ID
19   getPlagaById(plagaId: string): Observable<Plaga> {
20     const plagaDocRef = doc(this.firestore, `plagas/${plagaId}`);
21     return docData(plagaDocRef, { idField: 'id' }) as Observable<Plaga>;
22   }
23
24   // Actualizar una plaga
25   updatePlaga(plagaId: string, plagaData: Partial<Plaga>): Promise<void> {
26     const plagaDocRef = doc(this.firestore, `plagas/${plagaId}`);
27     return updateDoc(plagaDocRef, plagaData).then(() => {
28       console.log('Plaga actualizada exitosamente');
29     });
30   }
31
32   // Eliminar una plaga
33   deletePlaga(plagaId: string): Promise<void> {
34     const plagaDocRef = doc(this.firestore, `plagas/${plagaId}`);
35     return deleteDoc(plagaDocRef).then(() => {
36       console.log('Plaga eliminada exitosamente');
37     });
38   }
39
40   // Agregar una nueva plaga
41   addPlaga(newPlaga: Plaga): Promise<void> {
42     const plagasRef = collection(this.firestore, 'plagas');
43     return addDoc(plagasRef, newPlaga).then(() => {
44       console.log('Plaga agregada exitosamente');
45     });
46   }
47 }
48
```

Figura 35

Anexo ilustración parte 2 Plagas.service.ts

```
38 }
39
40 // Agregar una nueva plaga
41 addPlaga(newPlaga: Plaga): Promise<void> {
42   const plagasRef = collection(this.firestore, 'plagas');
43   return addDoc(plagasRef, newPlaga).then(() => {
44     console.log('Plaga agregada exitosamente');
45   });
46 }
47 }
48
```



Código PlagaService:

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

import { Firestore, doc, updateDoc, collection, collectionData, deleteDoc, addDoc,
DocumentReference, docData } from '@angular/fire/firestore';

import { Observable } from 'rxjs';

import { Plaga } from './plaga.model';

@Injectable({
  providedIn: 'root',
})
export class PlagaService {
  constructor(private firestore: Firestore) {}

  // Obtener todas las plagas
  getPlagas(): Observable<Plaga[]> {
    const plagasRef = collection(this.firestore, 'plagas');
    return collectionData(plagasRef, { idField: 'id' }) as Observable<Plaga[]>;
  }

  // Obtener una plaga por su ID
  getPlagaById(plagaId: string): Observable<Plaga> {
    const plagaDocRef = doc(this.firestore, `plagas/${plagaId}`);
    return docData(plagaDocRef, { idField: 'id' }) as Observable<Plaga>;
  }

  // Actualizar una plaga
```



```
updatePlaga(plagaId: string, plagaData: Partial<Plaga>): Promise<void> {  
  const plagaDocRef = doc(this.firestore, `plagas/${plagaId}`);  
  return updateDoc(plagaDocRef, plagaData).then(() => {  
    console.log('Plaga actualizada exitosamente');  
  });  
}
```

// Eliminar una plaga

```
deletePlaga(plagaId: string): Promise<void> {  
  const plagaDocRef = doc(this.firestore, `plagas/${plagaId}`);  
  return deleteDoc(plagaDocRef).then(() => {  
    console.log('Plaga eliminada exitosamente');  
  });  
}
```

// Agregar una nueva plaga

```
addPlaga(newPlaga: Plaga): Promise<void> {  
  const plagasRef = collection(this.firestore, 'plagas');  
  return addDoc(plagasRef, newPlaga).then(() => {  
    console.log('Plaga agregada exitosamente');  
  });  
}
```

Anexo 18:

Funcionamiento Lógico:

El servicio *AliadoService* maneja las operaciones relacionadas con los aliados biológicos en la base de datos. Permite obtener todos los aliados (*getAliados*) o uno específico por su ID (*getAliadoById*). También incluye métodos para agregar nuevos aliados (*addAliado*), actualizar los existentes (*updateAliado*), y eliminar registros (*deleteAliado*). Estas operaciones interactúan directamente con *Firestore*, garantizando la gestión eficiente y actualizada de los aliados en la plataforma.

Figura 36

Anexo ilustración parte 1 *Aliados.service.ts*

```
1 import { Injectable } from '@angular/core';
2 import { Firestore, doc, updateDoc, collection, collectionData, deleteDoc, addDoc, DocumentReference, docData } from '@angular/fire';
3 import { Observable } from 'rxjs';
4 import { Aliado } from '../aliado.model'; // Modelo de Aliado
5
6 @Injectable({
7   providedIn: 'root',
8 })
9 export class AliadoService {
10   constructor(private firestore: Firestore) {}
11
12   // Obtener todos los aliados
13   getAliados(): Observable<Aliado[]> {
14     const aliadosRef = collection(this.firestore, 'aliados'); // Colección 'aliados' en Firebase
15     return collectionData(aliadosRef, { idField: 'id' }) as Observable<Aliado[]>;
16   }
17
18   // Obtener un aliado por su ID
19   getAliadoById(aliadoId: string): Observable<Aliado> {
20     const aliadoDocRef = doc(this.firestore, `aliados/${aliadoId}`);
21     return docData(aliadoDocRef, { idField: 'id' }) as Observable<Aliado>;
22   }
23
24   // Actualizar un aliado
25   updateAliado(aliadoId: string, aliadoData: Partial<Aliado>): Promise<void> {
26     const aliadoDocRef = doc(this.firestore, `aliados/${aliadoId}`);
27     return updateDoc(aliadoDocRef, aliadoData).then(() => {
28       console.log('Aliado actualizado exitosamente');
29     }).catch((error) => {
30       console.error('Error al actualizar el aliado:', error);
31       throw error;
32     });
33   }
34
35   // Eliminar un aliado
36   deleteAliado(aliadoId: string): Promise<void> {
37     const aliadoDocRef = doc(this.firestore, `aliados/${aliadoId}`);
```

Figura 37

Anexo ilustración parte 2 Aliados.service.ts

```
38  return deleteDoc(aliadoDocRef).then(() => {
39      console.log('Aliado eliminado exitosamente');
40  }).catch((error) => {
41      console.error('Error al eliminar el aliado:', error);
42      throw error;
43  });
44  }
45
46  // Agregar un nuevo aliado
47  addAliado(newAliado: Aliado): Promise<void> {
48      const aliadosRef = collection(this.firestore, 'aliados');
49      return addDoc(aliadosRef, newAliado).then(() => {
50          console.log('Aliado agregado exitosamente');
51      }).catch((error) => {
52          console.error('Error al agregar el aliado:', error);
53          throw error;
54      });
55  }
56  }
57
```

Código AlliesServices:

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

import { Firestore, doc, updateDoc, collection, collectionData, deleteDoc, addDoc,
DocumentReference, docData } from '@angular/fire/firestore';

import { Observable } from 'rxjs';

import { Aliado } from './aliado.model'; // Modelo de Aliado

@Injectable({
  providedIn: 'root',
})
export class AliadoService {
  constructor(private firestore: Firestore) {}

  // Obtener todos los aliados
  getAliados(): Observable<Aliado[]> {
```

```
const aliadosRef = collection(this.firestore, 'aliados'); // Colección 'aliados' en Firebase  
return collectionData(aliadosRef, { idField: 'id' }) as Observable<Aliado[]>;  
}
```

// Obtener un aliado por su ID

```
getAliadoById(aliadoId: string): Observable<Aliado> {  
    const aliadoDocRef = doc(this.firestore, `aliados/${aliadoId}`);  
    return docData(aliadoDocRef, { idField: 'id' }) as Observable<Aliado>;  
}
```

// Actualizar un aliado

```
updateAliado(aliadoId: string, aliadoData: Partial<Aliado>): Promise<void> {  
    const aliadoDocRef = doc(this.firestore, `aliados/${aliadoId}`);  
    return updateDoc(aliadoDocRef, aliadoData).then(() => {  
        console.log('Aliado actualizado exitosamente');  
    }).catch((error) => {  
        console.error('Error al actualizar el aliado:', error);  
        throw error;  
    });  
}
```

// Eliminar un aliado

```
deleteAliado(aliadoId: string): Promise<void> {  
    const aliadoDocRef = doc(this.firestore, `aliados/${aliadoId}`);  
    return deleteDoc(aliadoDocRef).then(() => {  
        console.log('Aliado eliminado exitosamente');  
    });  
}
```



```
}).catch((error) => {  
  console.error('Error al eliminar el aliado:', error);  
  throw error;  
});  
}  
  
// Agregar un nuevo aliado  
addAliado(newAliado: Aliado): Promise<void> {  
  const aliadosRef = collection(this.firestore, 'aliados');  
  return addDoc(aliadosRef, newAliado).then(() => {  
    console.log('Aliado agregado exitosamente');  
  }).catch((error) => {  
    console.error('Error al agregar el aliado:', error);  
    throw error;  
  });  
}  
}
```