Panel de indicadores y aplicación web sistema avanzado de monitoreo y gestión de cultivos de caña de azúcar

José Mariano Reyes Hernández

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Panel de indicadores y aplicación web sistema avanzado de monitoreo y gestión de cultivos de caña de azúcar

Trabajo de graduación en modalidad de tesis presentado por
José Mariano Reyes Hernández
para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala, noviembre de 2024

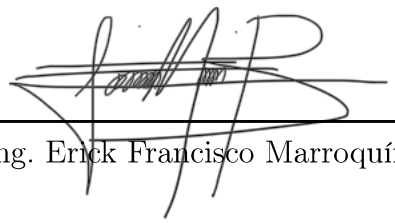
UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Panel de indicadores y aplicación web sistema avanzado de monitoreo y gestión de cultivos de caña de azúcar

Trabajo de graduación en modalidad de tesis presentado por
José Mariano Reyes Hernández
para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala, noviembre de 2024

Vo.Bo.:

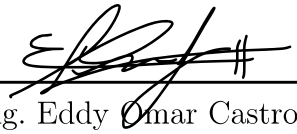
(f) 

Ing. Erick Francisco Marroquín

Tribunal Examinador:

(f) 

Ing. Erick Francisco Marroquín

(f) 

Ing. Eddy Omar Castro

(f) 

Msc. Moisés Alonso

Fecha de aprobación: Guatemala, 09 de diciembre de 2024.

Prefacio

En la actualidad, el sector agrícola enfrenta grandes desafíos debido a las exigencias de un mundo en constante cambio, en el que la productividad, la sostenibilidad y la eficiencia son pilares esenciales para el éxito. La caña de azúcar, siendo uno de los cultivos más importantes a nivel nacional, si no el más grande, no escapa a estos retos. Por lo mismo, resulta crucial innovar en las formas de monitoreo y gestión.

A lo largo de mis estudios en Ingeniería en Ciencias de la Computación y Tecnologías de la Información, he tenido la oportunidad de explorar y experimentar con diversas tecnologías que hoy en día están revolucionando la manera en que se abordan problemas complejos. Durante este tiempo de desarrolló del panel de control desde la fase de ideación, me quedó claro que la inteligencia artificial y los sistemas de información geográfica no solo pueden, sino que deben jugar un papel fundamental en la agricultura moderna. Estas tecnologías, correctamente aplicadas, tienen el poder de anticipar problemas, optimizar recursos y mejorar la toma de decisiones en tiempo real.

El presente proyecto nace de esa convicción y de la oportunidad de aplicar el conocimiento adquirido para generar un impacto tangible en un sector clave de la economía de nuestro país. A través de «AgroIntelligence», buscamos desarrollar una solución integral que combinara la flexibilidad y el dinamismo de React con las capacidades avanzadas de análisis espacial de ArcGIS, ofreciendo a los agricultores y gestores de cultivos de caña de azúcar una herramienta moderna, eficiente y poderosa para la gestión de sus operaciones.

Este proyecto no solo representa el cierre de una etapa académica, sino también el inicio de un camino hacia la búsqueda constante de soluciones tecnológicas innovadoras que contribuyan al bienestar social y al desarrollo sostenible.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas e instituciones que me han apoyado durante el desarrollo de este proyecto. En primer lugar, al Ingenio Pantaleón, por brindarme acceso a su plataforma web y por el valioso apoyo técnico y logístico a lo largo de este proceso. Su colaboración fue fundamental para la implementación y validación de las soluciones propuestas, permitiendo que este proyecto se ajustara a las necesidades reales del sector agrícola.

A mi asesor, el Ing. Erick Marroquín, le extiendo un especial reconocimiento por su guía, paciencia y dedicación. Sus observaciones y consejos fueron invaluables para el desarrollo y la mejora continua de este trabajo. Gracias por compartir su experiencia, conocimientos y por impulsar mi crecimiento profesional y académico.

Agradezco también a mis compañeros y amigos, quienes me brindaron su apoyo en el desarrollo de módulos integrales para el funcionamiento del proyecto. Su aliento y colaboración fueron esenciales para mantenerme enfocado y perseverante durante las etapas más desafiantes de este proyecto.

Finalmente, a mi familia, por su apoyo incondicional. Han sido mi mayor fuente de inspiración y su confianza en mí fue lo más importante para la culminación de este trabajo. A todos ustedes, mi más profundo agradecimiento.

Índice

Lista de figuras

Resumen

El proyecto «AgroIntelligence» tiene como objetivo transformar la gestión de cultivos de caña de azúcar mediante el desarrollo de un sistema integrado que combina dos tecnologías clave: un panel de control web desarrollado con React y una aplicación geoespacial basada en ArcGIS. Estas herramientas proporcionan a los agricultores y gestores de cultivos acceso a datos en tiempo real, predicciones basadas en inteligencia artificial (IA) y visualizaciones espaciales detalladas para optimizar la toma de decisiones y mejorar la productividad.

React fue elegido como la tecnología principal para desarrollar el panel de control web debido a su capacidad de crear interfaces de usuario dinámicas y altamente interactivas. Con React, se desarrollaron componentes reutilizables que permiten la actualización eficiente de datos, lo que facilita a los usuarios la visualización y gestión de indicadores clave de los cultivos de caña de azúcar. El panel de control proporciona una experiencia fluida y responsiva, permitiendo a los agricultores consultar modelos predictivos de rendimiento y enfermedades, así como gestionar datos provenientes de sensores y otros sistemas externos. La arquitectura de React, basada en el "Virtual DOM", asegura que solo las partes de la interfaz que cambian se actualicen, mejorando la eficiencia y la experiencia de usuario, incluso con grandes volúmenes de datos.

ArcGIS, por otro lado, fue implementado como una solución geoespacial complementaria que permite un análisis avanzado del terreno y las condiciones ambientales. Esta aplicación se diseñó para facilitar la visualización de datos espaciales, como mapas interactivos que muestran la distribución geográfica de los cultivos, patrones de enfermedades, y zonas de riesgo en tiempo real. ArcGIS permite a los usuarios realizar análisis espaciales avanzados, tales como la identificación de áreas con mayor probabilidad de infección por plagas o la optimización del uso de recursos basados en la ubicación de los campos. Al combinar estas funcionalidades con el análisis predictivo del panel de control, los usuarios pueden, no solo monitorear sus cultivos de forma precisa, sino también planificar y anticipar acciones que optimicen la producción.

En conjunto, React y ArcGIS permiten que «AgroIntelligence» ofrezca una solución integral que abarca tanto la gestión diaria de los cultivos como la exploración avanzada de datos geoespaciales, brindando a los agricultores las herramientas necesarias para enfrentar los retos actuales del sector de la caña de azúcar.

CAPÍTULO 1

Introducción

En la era de la transformación digital, donde la tecnología redefine los límites de lo posible, el sector agrícola enfrenta desafíos sin precedentes y oportunidades extraordinarias. La necesidad de optimizar los recursos, aumentar la productividad y garantizar la sostenibilidad ambiental ha impulsado la búsqueda de soluciones innovadoras. En este contexto, emerge «AgroIntelligence», un proyecto pionero destinado a revolucionar el monitoreo y gestión de cultivos de caña de azúcar mediante el uso de tecnologías avanzadas de inteligencia artificial y desarrollo web moderno.

La caña de azúcar es uno de los cultivos más importantes a nivel global, siendo Guatemala la productora número once de caña de azúcar a nivel global con casi 3 millones de toneladas [1]. Por su vasta aplicación en la industria alimentaria y energética, enfrenta numerosos retos, incluidas las enfermedades que pueden comprometer seriamente su rendimiento. Tradicionalmente, la detección y gestión de estas enfermedades han requerido métodos que consumen tiempo y recursos, a menudo con resultados subóptimos. «AgroIntelligence» se propone abordar estos desafíos a través del desarrollo de dos soluciones complementarias: un panel de control web avanzado y una aplicación ArcGIS especializada, el cual es un sistema de información geográfica que nos va a ayudar a visualizar mapas de una manera más avanzada.

El componente principal del proyecto es un panel de control web desarrollado con React, una biblioteca de JavaScript líder para la creación de interfaces de usuario interactivas y eficientes. Esta aplicación web moderna proporcionará una interfaz intuitiva y responsiva para la gestión de datos agrícolas, incluyendo la visualización de estadísticas clave, modelos predictivos y herramientas de análisis basadas en inteligencia artificial. Este tipo de tecnología ha demostrado ser crucial en el sector agrícola para transformar la manera en que se gestionan los cultivos y se abordan problemas complejos, como la detección temprana de enfermedades [2, 3].

Paralelamente, se desarrollará una aplicación independiente utilizando ArcGIS, una plataforma líder en sistemas de información geográfica. Esta aplicación se centrará en proporcionar capacidades avanzadas de análisis geoespacial y visualización de datos complejos específicos para el sector agrícola. Aunque separadas, ambas aplicaciones trabajarán en conjunto para ofrecer una solución completa y robusta.

El panel de control web incluirá funcionalidades clave como la integración de un punto de consumo para la recolección y centralización de datos en tiempo real, una sección para el análisis de imágenes de cultivos mediante inteligencia artificial [4], y un modelo predictivo de los niveles de TCH (toneladas de caña por hectárea), haciendo uso de analíticas predictivas [5]. Por su parte,

la aplicación ArcGIS se enfocará en proporcionar mapas interactivos detallados, análisis espaciales avanzados y visualizaciones geográficas específicas para el monitoreo de cultivos.

Esta combinación de tecnologías permitirá a los usuarios acceder a una gama completa de herramientas: desde una interfaz web ágil y fácil de usar para el manejo diario, hasta capacidades geoespaciales avanzadas para análisis en profundidad. Los agricultores y gestores podrán alternar entre ambas aplicaciones según sus necesidades específicas, beneficiándose de la flexibilidad y eficiencia de React para operaciones cotidianas y de la potencia analítica de ArcGIS para tareas más especializadas. Para asegurar la implementación continua y escalable de estas soluciones, la integración de metodologías CI/CD será clave [6].

Al ofrecer estas soluciones complementarias, «AgroIntelligence» se posiciona a la vanguardia de las soluciones agrícolas inteligentes, proporcionando herramientas poderosas y accesibles para la gestión eficiente de cultivos de caña de azúcar. Este enfoque dual permitirá a los usuarios aprovechar lo mejor de ambas tecnologías, facilitando una toma de decisiones más informada y una gestión más efectiva de sus cultivos.

CAPÍTULO 2

Objetivos

2.1. Objetivo general

Desarrollar una interfaz de usuario intuitiva para el panel de control interactivo y la aplicación web para facilitar a los agricultores y gestores de cultivos el acceso y la comprensión de los análisis de datos y recomendaciones ofrecidos por el sistema.

2.2. Objetivos específicos

- Validar la eficacia de la interfaz de usuario diseñada mediante pruebas con usuarios finales, para así asegurar que los agricultores y gestores de cultivos puedan utilizar efectivamente los análisis de datos y recomendaciones dadas.
- Implementar modelos de inteligencia artificial que procesan imágenes y datos sensoriales por medio del API con el Data Center para identificar y clasificar eficazmente enfermedades en cultivos de caña de azúcar.
- Desarrollar un panel de control interactivo para indicadores definidos para poder visualizar en tiempo real datos relevantes antes de fin de año.
- Implementar funcionalidades en el panel de control que presenten análisis de datos y recomendaciones de manera visualmente intuitiva y fácilmente comprensible, permitiendo a los agricultores y gestores realizar decisiones informadas para el manejo óptimo de sus cultivos.
- Modificar la interfaz y las funcionalidades del sistema basándose en la retroalimentación recibida para mejorar la usabilidad y eficacia del panel de control y la aplicación web.
- Implementar una aplicación especializada de ArcGIS como complemento del sistema principal, adaptada a las necesidades del Ingenio Pantaleón, con el fin de mejorar la precisión del mapeo de cultivos y optimizar la gestión de recursos agrícolas.

CAPÍTULO 3

Justificación

La eficiencia en el monitoreo y gestión de cultivos representa un desafío creciente en la agricultura moderna, especialmente en cultivos de alto valor como la caña de azúcar. Estos cultivos son fundamentales para diversas industrias, pero su producción está amenazada constantemente por enfermedades, condiciones ambientales adversas y la creciente demanda de recursos [1]. La necesidad de optimizar su manejo para garantizar la sostenibilidad y productividad agrícola nunca ha sido tan crítica. En este escenario, el proyecto «AgroIntelligence», que comprende un panel de control web desarrollado con React y una aplicación complementaria ArcGIS el cual es un sistema de información geográfico, emerge como una solución innovadora dirigida a transformar radicalmente la manera en que se monitorean y gestionan los cultivos de caña de azúcar.

Este proyecto se justifica, no solo por la urgencia de abordar las limitaciones de las prácticas agrícolas tradicionales, sino también por la oportunidad de liderar la incorporación de la inteligencia artificial y el análisis geoespacial avanzado en la agricultura. Al ofrecer un sistema integrado de monitoreo y gestión basado en IA y GIS, el proyecto facilita la detección temprana de enfermedades, la optimización de recursos y la toma de decisiones informadas, mejorando así la eficacia en la prevención de pérdidas de cosechas y maximizando el rendimiento [2, 3].

El desarrollo de un panel de control web y una aplicación ArcGIS específicos para este fin, permite una interacción directa y accesible para los agricultores y gestores de cultivos, democratizando el acceso a tecnologías de punta y promoviendo una agricultura más inteligente y sostenible [5, 6]. Este enfoque dual no solo aumenta la capacidad de respuesta ante los desafíos agrícolas, sino que también fomenta una mayor comprensión y adopción de prácticas agrícolas basadas en datos, lo cual es esencial para la transición hacia métodos de producción más eficientes y menos invasivos con el medio ambiente [4].

La integración de una aplicación ArcGIS especializada como complemento del sistema principal añade una dimensión crucial al proyecto. Esta herramienta permite un análisis geoespacial avanzado de los cultivos de caña de azúcar, facilitando la visualización de patrones espaciales, la identificación de zonas de riesgo y la optimización de recursos basada en la localización [7]. Para el Ingenio Pantaleón, esto significa una mejora significativa en la toma de decisiones estratégicas a nivel de campo y gerencial, permitiendo una gestión más precisa y eficiente de sus extensos cultivos [1].

Por tanto, el proyecto tiene un potencial para servir como catalizador en la adopción de innovaciones tecnológicas en el sector agrícola. Al integrar análisis avanzados, aprendizaje automático y capacidades geoespaciales en un sistema accesible y fácil de usar, «AgroIntelligence» no solo aborda

necesidades inmediatas en la gestión de cultivos de caña de azúcar, sino que también establece un modelo para futuras aplicaciones de IA y GIS en la agricultura [8].

Así, este proyecto contribuye significativamente a los esfuerzos globales para asegurar disminuir el impacto ambiental, la eficiencia en el uso de recursos y la adaptación al cambio climático [9]. Al proporcionar herramientas avanzadas de monitoreo y análisis, «AgroIntelligence» no solo mejora la productividad agrícola, sino que también promueve prácticas más sostenibles, reduciendo el impacto ambiental y mejorando la resiliencia de los sistemas agrícolas frente a los desafíos futuros [10].

CAPÍTULO 4

Marco teórico

4.1. Revisión de literatura

4.1.1. Tecnología y agricultura

Uso de inteligencia artificial en agricultura:

Ponce Pivaral menciona el desarrollo de una aplicación para el conteo y clasificación de huevos de mosquito utilizando inteligencia artificial, lo que demuestra la aplicabilidad de la IA en escenarios agrícolas y similares para monitoreo y análisis detallado [11].

Con *Smart Farming Robot* [12] se explora la utilización de un robot autónomo en invernaderos para detectar condiciones ambientales, utilizando algoritmos de aprendizaje no supervisado, lo que refleja la tendencia hacia la automatización en la agricultura utilizando tecnologías avanzadas.

Se están desarrollando sistemas de IA para identificar enfermedades en los cultivos con alta precisión mediante sistemas de detección de enfermedades basados en visión artificial. Por ejemplo, se han diseñado sistemas para distinguir entre la enfermedad de Pierce y el amarillamiento de la vid con una precisión superior al 92 %.

La IA se utiliza para mejorar el monitoreo de la salud del suelo, permitiendo un análisis más completo que los métodos tradicionales. Esto incluye la identificación de microbios y deficiencias de nutrientes en muestras de suelo utilizando algoritmos de aprendizaje automático [3].

Los agricultores están utilizando la IA para optimizar el uso de agua y fertilizantes en los cultivos, reduciendo significativamente los desechos y previniendo la contaminación del suelo y el agua [13].

Las tecnologías como drones y *software* impulsado por IA se utilizan para tareas como el monitoreo de cultivos, análisis de sistemas de riego y aplicación precisa de pesticidas y herbicidas. La IA permite un uso más eficiente de los recursos y ayuda a identificar áreas específicas de un campo que requieren atención, optimizando así los procesos [3].

Las aplicaciones de IA en la agricultura también incluyen la predicción de rendimientos, que ayuda a los agricultores a planificar mejor y maximizar la producción utilizando información basada en datos. Por ejemplo, los modelos de IA pueden simular los impactos del clima y optimizar los

tiempos de siembra y cosecha [2].

4.1.2. Sistemas de monitoreo y gestión de cultivos

Plataformas y *dashboards*:

Aerobotics ofrece análisis detallados de árboles y frutas mediante el uso de drones e inteligencia artificial, destacando la importancia de la recopilación de datos en tiempo real para optimizar el rendimiento de los cultivos.

HEMAV utiliza técnicas de IA y aprendizaje automático para proporcionar a los agricultores datos específicos y accionables, demostrando cómo la tecnología puede mejorar significativamente la productividad y rentabilidad.

4.1.3. Detectores y aplicaciones móviles

Aplicaciones en la detección de enfermedades y diagnósticos:

PlantVillage es una plataforma que permite a los agricultores obtener diagnósticos de enfermedades en sus cultivos mediante fotografías, lo cual es un ejemplo directo de cómo la IA puede ser utilizada para la identificación y clasificación de problemas en plantas.

Tumaini, una aplicación para *smartphones*, utiliza aprendizaje automático para ayudar a los agricultores de banano a detectar enfermedades y plagas importantes con un 90 % de éxito, demostrando la aplicación práctica de la IA en la gestión de la salud de los cultivos [2].

4.1.4. Desarrollo de tecnologías para la agricultura sostenible

Impacto en la sostenibilidad y eficiencia:

La necesidad de integrar soluciones tecnológicas para mejorar la sostenibilidad en la agricultura es crucial. «AgronIntelligence» tiene como objetivo centralizar y optimizar la gestión agrícola mediante un panel avanzado que integra modelos de soluciones capaces de analizar datos complejos y proporcionar conocimientos valiosos para decisiones agrícolas eficaces.

La literatura revisada sugiere una tendencia creciente hacia la integración de tecnologías avanzadas en la agricultura. Los estudios y aplicaciones discutidos resaltan el potencial de la inteligencia artificial para transformar el sector agrícola, haciendo especial énfasis en la mejora de la eficiencia, la productividad y la sostenibilidad. Este proyecto se alinea con estos desarrollos, proponiendo un enfoque innovador que podría servir como referencia para futuras investigaciones y aplicaciones en el campo.

4.2. Fundamentos teóricos de los modelos de análisis y predicción

4.2.1. Panel de control

En este proyecto, el uso de tecnología de *dashboard* en ArcGIS es crucial para la visualización efectiva y la gestión de datos geoespaciales. Los fundamentos teóricos detrás de esta tecnología implican la integración y síntesis de grandes conjuntos de datos provenientes de diversas fuentes, lo que facilita la toma de decisiones informadas. Los *dashboards* actúan como plataformas interactivas que no solo presentan datos en tiempo real sino que también permiten manipular y analizar estos datos para prever tendencias y resultados. Al ser este el objeto de estudio de mi módulo del proyecto se detalla en la siguiente sección más a detalle las tecnologías y herramientas que se usarán.

4.2.2. Modelo para clasificar plagas en campo

Este segmento del proyecto utiliza modelos de aprendizaje automático para clasificar y predecir la presencia de plagas en cultivos agrícolas basados en imágenes satelitales. Los fundamentos teóricos incluyen técnicas de procesamiento de imágenes y análisis predictivo que ayudan a determinar los niveles de infestación de plagas. Este modelo es esencial para implementar estrategias de manejo integrado de plagas, optimizando el uso de recursos y minimizando el impacto ambiental.

El procesamiento de imágenes es un componente esencial en aplicaciones de inteligencia artificial, particularmente en la agricultura, para optimizar la gestión y monitoreo de cultivos. [4] Este proceso implica varias etapas clave:

- Captura de imágenes: recolectar imágenes mediante dispositivos como drones o satélites, que pueden captar datos en diversos espectros.
- Preprocesamiento: ajustar las imágenes para mejorar su calidad y uniformidad, lo cual es crucial para análisis precisos.
- Segmentación: dividir la imagen en segmentos basados en características similares, útil para identificar zonas de interés como áreas afectadas por enfermedades.
- Extracción de características: identificar y extraer características relevantes de las imágenes, como texturas o colores asociados a condiciones específicas del cultivo.
- Clasificación y análisis: utilizar algoritmos de aprendizaje automático para clasificar las imágenes y hacer inferencias, como determinar la presencia de enfermedades o plagas.
- Visualización e interpretación: presentar los resultados de manera accesible para facilitar decisiones informadas, como mediante mapas de calor o reportes.

El análisis predictivo en inteligencia artificial (IA) se refiere al uso de técnicas de modelado estadístico, aprendizaje automático y minería de datos para analizar datos actuales e históricos con el objetivo de hacer predicciones sobre eventos futuros o desconocidos. En esencia, el análisis predictivo permite a las organizaciones y a los investigadores tomar decisiones más informadas y proactivas [5].

- Modelado de datos: utiliza diversos algoritmos y modelos estadísticos para estimar o predecir resultados futuros basados en datos históricos. Por ejemplo, en el comercio minorista, podría predecir las tendencias de compra de los consumidores y ayudar a optimizar el inventario.

- Aprendizaje automático: incorpora algoritmos de aprendizaje automático que se entrenan con conjuntos de datos para identificar patrones y relaciones. Una vez entrenado, el modelo puede hacer predicciones sobre datos nuevos y no vistos.
- Aplicaciones: el análisis predictivo se aplica en una amplia gama de campos como finanzas, salud, marketing y más recientemente, en agricultura y gestión de recursos naturales. Por ejemplo, en el sector financiero, se utiliza para evaluar el riesgo crediticio de los clientes.
- Herramientas y tecnologías: se utilizan diversas herramientas y plataformas de *software* que facilitan el análisis predictivo, incluyendo Python, R, SAS y sistemas especializados como SPSS y Tableau.
- Beneficios: la capacidad de prever eventos futuros permite a las empresas y organizaciones reducir riesgos, optimizar operaciones y maximizar los resultados. En el contexto de la IA, potencia estas capacidades al automatizar y escalar los análisis en grandes volúmenes de datos.

4.2.3. Modelo para detección de madurez de cultivos mediante satélite

Este proyecto aplica modelos predictivos para evaluar riesgos de inundación y el grado de madurez de los cultivos utilizando datos satelitales. Los fundamentos incluyen la interpretación de índices de vegetación y análisis de superficie terrestre para prever eventos que puedan impactar negativamente en la agricultura. Estos modelos son cruciales para planificar de manera proactiva las actividades agrícolas, optimizando la cosecha y reduciendo las pérdidas debido a condiciones climáticas adversas.

Los índices de vegetación (VI) son herramientas esenciales en la agricultura de precisión para evaluar la salud y el crecimiento de las plantas mediante el uso de datos obtenidos por sensores remotos. Estos índices aprovechan las diferencias en las propiedades espectrales de las plantas para proporcionar información valiosa sobre su estado fisiológico y su respuesta a diferentes condiciones ambientales.

Uno de los índices más utilizados es el índice de vegetación de diferencia normalizada (NDVI), que es efectivo para evaluar la biomasa verde y la salud de la vegetación. El NDVI utiliza las bandas espectrales del rojo y el infrarrojo cercano (NIR) para maximizar la sensibilidad a la cantidad y vitalidad de la vegetación [8].

Además del NDVI, otros índices como el índice de vegetación ,ejorado (EVI) y el índice de vegetación ajustado al suelo (SAVI) se utilizan para superar algunas limitaciones del NDVI, como los efectos de saturación en áreas con vegetación densa. El EVI, por ejemplo, es más sensible a las diferencias en las áreas densas de la canopia y proporciona una señal más clara de la variabilidad de la vegetación [8].

El uso de estos índices no se limita solo a la evaluación de la salud de la vegetación, sino que también apoya la gestión de recursos, la optimización de las tasas de aplicación de insumos como fertilizantes y pesticidas y puede ayudar en la gestión de riesgos y desastres relacionados con el clima. Al integrar datos de diferentes partes del espectro electromagnético, los VI permiten acentuar ciertas características de la vegetación que serían difíciles o imposibles de discernir a simple vista [8].

4.2.4. Centro de datos para inteligencia empresarial agrícola

Este componente del proyecto se centra en la creación y mantenimiento de un centro de datos que procesa y almacena imágenes satelitales para análisis de inteligencia empresarial. Los fundamentos teóricos se basan en la gestión de grandes bases de datos y la integración de sistemas de información geográfica (SIG) con plataformas de análisis avanzado. Este enfoque permite una mejor comprensión

y respuesta a las dinámicas del mercado agrícola, mejorando la precisión en la toma de decisiones estratégicas.

La gestión de grandes bases de datos y SIG con plataformas de análisis avanzado son aspectos cruciales en numerosos campos, incluyendo la agricultura, la gestión ambiental y la planificación urbana. Estos procesos implican no solo almacenar y procesar grandes volúmenes de datos geoespaciales y temporales, sino también utilizar estos datos para análisis complejos y toma de decisiones basada en evidencia.

4.2.5. Gestión de grandes bases de datos

Almacenamiento y escalabilidad: las bases de datos NoSQL como MongoDB y Cassandra son ideales para gestionar grandes volúmenes de datos geoespaciales debido a su capacidad de escalar horizontalmente. MongoDB, en particular, es conocido por su modelo de datos basado en documentos, que ofrece flexibilidad en el diseño del esquema y facilita la escalabilidad. Por otro lado, Cassandra destaca por su alta disponibilidad y rendimiento en operaciones de escritura, siendo adecuada para aplicaciones que requieren un gran volumen de escrituras y una distribución eficiente de los datos a través de múltiples nodos [14].

- Indexación y consulta eficiente: tanto MongoDB como Cassandra ofrecen capacidades robustas de indexación para mejorar el rendimiento de las consultas. MongoDB soporta índices secundarios, lo que permite una mejora significativa en el rendimiento de las consultas. Cassandra, aunque tiene capacidades limitadas para índices secundarios, también permite ciertas optimizaciones a través de su lenguaje de consulta CQL [14].
- Integridad y seguridad de los datos: ambas bases de datos ofrecen características para asegurar la integridad y seguridad de los datos. MongoDB proporciona funciones de replicación incorporadas para la tolerancia a fallos y soporta encriptación de extremo a extremo. Cassandra, conocida por su consistencia eventual, también ofrece configuraciones de consistencia ajustables para balancear la disponibilidad y la precisión de los datos [14].

4.3. Tecnologías y herramientas del panel de control

El desarrollo del panel de control en el proyecto «AgronIntelligence» se apoya en un conjunto robusto de tecnologías web modernas que permiten una experiencia de usuario optimizada, un rendimiento eficiente y una integración adecuada de datos. A continuación, se describen en detalle las herramientas principales seleccionadas para este propósito.

4.3.1. Github

GitHub es una plataforma de desarrollo colaborativo basada en la nube que permite a los desarrolladores gestionar proyectos de *software* utilizando Git, un sistema de control de versiones distribuido. Fundada en 2008, GitHub se ha convertido en una herramienta esencial para el desarrollo de *software* moderno, facilitando la colaboración entre equipos, el control de versiones y la integración continua en el ciclo de vida del desarrollo de *software*.

Git y GitHub: conceptos fundamentales

Git es un sistema de control de versiones creado por Linus Torvalds en 2005. Es utilizado para rastrear los cambios en los archivos y coordinar el trabajo entre varios desarrolladores en un proyecto. Git permite a los desarrolladores gestionar el historial de versiones de su código, revertir cambios, crear ramas para trabajar en nuevas características de manera aislada y fusionar esos cambios en una rama principal.

GitHub amplía las capacidades de Git al proporcionar una interfaz web y servicios adicionales que facilitan la colaboración y gestión de proyectos. GitHub no solo almacena repositorios Git, sino que también ofrece herramientas para la gestión de proyectos, revisión de código, documentación y despliegue continuo.

Tiene algunos componentes clave:

- Repositorios: un repositorio en GitHub es un lugar donde se almacenan los archivos del proyecto, junto con su historial de versiones. Los repositorios pueden ser públicos o privados, lo que permite a los desarrolladores elegir si desean compartir su código con la comunidad o mantenerlo confidencial.
- Control de versiones: GitHub permite gestionar el control de versiones de un proyecto utilizando Git. Los desarrolladores pueden clonar repositorios a sus máquinas locales, hacer cambios en el código y luego subir esos cambios al repositorio central en GitHub mediante commits y pushes. La creación de branches (ramas) permite trabajar en diferentes versiones de un proyecto simultáneamente. Las pull requests facilitan la revisión y discusión de cambios antes de que se fusionen en la rama principal, asegurando que el código sea revisado y probado antes de ser integrado.
- Colaboración y revisión de código: GitHub fomenta la colaboración entre desarrolladores a través de pull requests y revisiones de código. Los colaboradores pueden comentar líneas específicas de código, sugerir cambios y aprobar o solicitar modificaciones antes de que los cambios se integren en el código base.
- Seguridad y gestión de proyectos: GitHub proporciona características de seguridad como el escaneo de vulnerabilidades en dependencias y la implementación de flujos de trabajo de revisión de código para mantener la calidad y seguridad del *software*.

GitHub es una plataforma fundamental en el ecosistema del desarrollo de *software*. Su combinación de control de versiones, gestión de proyectos y herramientas de colaboración lo hace imprescindible para cualquier equipo que desee desarrollar *software* de alta calidad de manera eficiente y colaborativa. La capacidad de GitHub para integrarse con flujos de trabajo de integración y despliegue continuo, así como su enfoque en la seguridad y la gestión de la documentación, refuerza su papel como una herramienta clave en el desarrollo de *software* contemporáneo [15].

4.3.2. React

React es una biblioteca de JavaScript que fue desarrollada por Facebook. Fue diseñada para construir interfaces de usuario, en especial para aplicaciones web de una sola página. React se basa en la creación de componentes reutilizables que representan partes de la interfaz de usuario. Estos componentes pueden gestionar su propio estado y se combinan para formar aplicaciones complejas de manera declarativa.

Como principal característica está "Virtual DOM", que permite a la biblioteca actualizar de manera eficiente solo aquellas partes de la interfaz que han cambiado, mejorando así el rendimiento.

React es conocido por su simplicidad y la modularidad, permiten a los desarrolladores construir aplicaciones complejas con una estructura mantenible. La comunidad de React es amplia y cuenta con un vasto ecosistema de herramientas y bibliotecas que complementan su funcionalidad básica, como React Router para la gestión de rutas y Redux para la gestión de estado global.

Importancia de los *hooks* en React

Los *hooks* son una característica fundamental en React, introducidos en la versión 16.8, que permiten a los desarrolladores utilizar el estado y otros ciclos de vida en componentes funcionales, los cuales antes solo estaban disponibles en componentes de clase. Los *hooks* más utilizados son useState y useEffect.

- useState: permite a los componentes funcionales tener su propio estado interno. Esto es crucial para gestionar datos dinámicos dentro del componente, como formularios, contadores y más.
- useEffect: facilita la ejecución de efectos secundarios en componentes funcionales, como la obtención de datos, la suscripción a eventos y la manipulación directa del DOM. useEffect reemplaza la necesidad de métodos de ciclo de vida como componentDidMount, componentDidUpdate y componentWillUnmount en componentes de clase.

Los *hooks* han transformado la forma en que se desarrollan las aplicaciones en React, estos permiten un código más limpio, fácil de entender y de probar y reduciendo la necesidad de escribir componentes de clase, lo que simplifica la estructura y la lógica del código [16].

4.3.3. Vite

Vite es una herramienta moderna de construcción y desarrollo para aplicaciones web que se destaca por su rapidez y simplicidad. Desarrollado por Evan You, el creador de Vue.js, Vite se ha convertido en una opción popular entre los desarrolladores web debido a su capacidad para mejorar significativamente la experiencia de desarrollo en comparación con herramientas más tradicionales como Webpack o Parcel.

Arquitectura y funcionamiento de Vite

Vite se basa en una arquitectura innovadora que aprovecha las características de los módulos ES (ECMAScript Modules) nativos del navegador, lo que permite un entorno de desarrollo mucho más rápido y eficiente. La herramienta está diseñada para abordar dos problemas principales en el desarrollo web moderno: los tiempos de inicio lentos y las reconstrucciones ineficientes.

- Desarrollo basado en módulos: a diferencia de los paquetes de herramientas tradicionales que empaquetan y transpilan todo el código antes de que el navegador lo cargue, Vite sirve directamente los módulos ES. Esto significa que solo los módulos que realmente se utilizan en una página se cargan y procesan, lo que reduce significativamente los tiempos de inicio en el entorno de desarrollo. Vite aprovecha las capacidades modernas de los navegadores para importar módulos JavaScript directamente, eliminando la necesidad de un empaquetado inicial pesado y permitiendo que el servidor de desarrollo arranque en milisegundos, incluso en proyectos grandes.
- Hot Module Replacement (HMR) instantáneo: Vite implementa un sistema de reemplazo de módulos en caliente (HMR) ultra rápido. Cuando un archivo se actualiza, Vite solo recompila

ese módulo específico, en lugar de reconstruir toda la aplicación. Esto permite que los cambios se reflejen instantáneamente en el navegador sin la necesidad de una recarga completa de la página, mejorando la productividad del desarrollador.

- Soporte para *frameworks* y lenguajes modernos: Vite ofrece soporte inmediato para *frameworks* populares como React, Vue.js, Svelte y más, permitiendo a los desarrolladores utilizar las herramientas y tecnologías más recientes con mínima configuración. Además, Vite soporta el uso de TypeScript, JSX y otros preprocesadores de forma nativa. Gracias a su diseño modular, Vite puede extenderse fácilmente mediante plugins para manejar una amplia variedad de lenguajes y herramientas.
- Optimización de producción: aunque Vite es altamente eficiente en el entorno de desarrollo, también está diseñado para generar builds de producción optimizadas. Utiliza Rollup, una herramienta de empaquetado de módulos avanzada, para crear bundles optimizados que son adecuados para la producción, con características como el tree-shaking (eliminación de código muerto) y la minimización de código.
- Configuración simplificada: Vite viene con una configuración mínima que funciona perfectamente fuera de la caja. Sin embargo, para aplicaciones más complejas, Vite ofrece un archivo de configuración (vite.config.js) donde los desarrolladores pueden personalizar aspectos como la resolución de módulos, los alias de rutas, la configuración del servidor y más.
- Ecosistema de *plugins*: Vite cuenta con un ecosistema robusto de plugins que extienden sus capacidades. Estos plugins permiten a los desarrolladores agregar soporte para características adicionales, como el manejo de imágenes, transformaciones específicas de archivos, o la integración con otras herramientas y plataformas.

Concluyendo, es una herramienta innovadora que ha redefinido la experiencia de desarrollo web mediante su enfoque en la velocidad, la simplicidad y la eficiencia. Con su arquitectura centrada en los módulos ES, su rápido sistema de HMR y su capacidad para generar builds optimizados para producción, se ha convertido en una solución poderosa para el desarrollo de aplicaciones web modernas. Al mejorar la productividad y facilitar la integración con tecnologías actuales, está marcando un nuevo estándar en el ecosistema de desarrollo *front-end*[17].

4.3.4. Node.js

Node.js es un entorno de ejecución de JavaScript construido sobre el motor V8 de Google Chrome. Permite a los desarrolladores utilizar JavaScript para la programación del lado del servidor, lo que unifica el lenguaje de programación tanto en el *front-end* como en el back-end. Node.js es conocido por su arquitectura orientada a eventos y su capacidad de manejar operaciones de entrada/salida de manera no bloqueante, lo que lo hace ideal para aplicaciones que requieren un alto rendimiento y escalabilidad. En el proyecto «AgronIntelligence», Node.js se utiliza para manejar la lógica del servidor, la gestión de API y la integración con bases de datos, proporcionando una infraestructura backend robusta y eficiente [18].

4.3.5. Bootstrap

Bootstrap es un *framework* de *front-end* que facilita el diseño y desarrollo de sitios web responsivos y móviles. Incluye una amplia colección de componentes preconstruidos, como botones, formularios, menús de navegación y tablas, que se pueden personalizar para ajustarse a las necesidades del proyecto. Asegura que el panel de control no solo sea funcional, sino también estéticamente agradable y accesible en una variedad de dispositivos y tamaños de pantalla. En «AgronIntelligence»,

Bootstrap ayuda a estandarizar el diseño visual del panel, garantizando una experiencia de usuario coherente y profesional [19].

4.3.6. ESLint

ESLint es una herramienta de análisis estático de código, desarrollada inicialmente por Nicholas C. Zakas en 2013, que se ha convertido en un estándar de facto en el ecosistema de JavaScript. Esta herramienta examina sistemáticamente el código fuente sin ejecutarlo, identificando patrones problemáticos basándose en un conjunto de reglas configurables que abarcan tanto la calidad del código como aspectos estilísticos. Su marco teórico se fundamenta en los principios de la ingeniería de *software* relacionados con la mantenibilidad, legibilidad y consistencia del código, implementando un sistema de *plugins* extensible que permite a los desarrolladores definir, compartir y aplicar estándares de codificación personalizados. ESLint utiliza un analizador sintáctico (parser) para transformar el código fuente en un árbol de sintaxis abstracta (AST), sobre el cual aplica las reglas definidas, lo que permite una detección precisa de problemas potenciales y asegura la coherencia en bases de código de gran escala [20].

4.3.7. HTML y CSS

HTML

HTML es un lenguaje de marcado que se utiliza para crear la estructura y el contenido de las páginas web. Es el esqueleto de cualquier página web, definiendo los elementos básicos como textos, imágenes, enlaces, listas, formularios, tablas y otros tipos de contenido.

- Etiquetas HTML: HTML organiza el contenido en elementos, definidos por etiquetas. Estas etiquetas rodean o envuelven diferentes partes del contenido para especificar su propósito y apariencia. Por ejemplo, <h1> define un encabezado de primer nivel, <p> define un párrafo y <a> crea un enlace.
- Árbol DOM (Document Object Model): cuando un navegador carga una página HTML, convierte el contenido en un árbol de nodos llamado DOM. Cada elemento HTML se convierte en un nodo en este árbol, lo que permite a los desarrolladores acceder y manipular la estructura de la página a través de JavaScript, haciendo que las páginas sean dinámicas e interactivas.
- Atributos HTML: las etiquetas HTML pueden tener atributos que proporcionan información adicional sobre los elementos. Por ejemplo, la etiqueta <img> usa el atributo src para definir la ruta de una imagen y alt para proporcionar un texto alternativo.
- Semántica en HTML: HTML5 introdujo una serie de etiquetas semánticas como <header>, <footer>, <article> y <section>, que describen más claramente el propósito de los distintos bloques de contenido en una página web. Esto no solo mejora la accesibilidad, sino que también facilita a los motores de búsqueda comprender mejor la estructura de la página.

CSS

CSS es un lenguaje de estilo utilizado para controlar la presentación de los documentos HTML. Mientras que HTML define el contenido y la estructura, CSS se encarga de cómo esos elementos deben aparecer visualmente en la pantalla.

- Selectores y propiedades CSS: CSS utiliza selectores para apuntar a elementos específicos en el DOM y aplicarles estilos. Las propiedades CSS, como color, font-size, margin y padding, se utilizan para definir cómo deben verse estos elementos.
- Selectores básicos: incluyen selectores de tipo (h1, p), de clase (.miClase), de id (#miId) y de atributo ([type="text"]).
- Selectores combinados: Permiten apuntar a elementos más específicos mediante combinaciones de selectores, como selectores descendientes (div p) o selectores hermanos (h1 + p).
- Modelo de caja (Box Model): uno de los conceptos más importantes en CSS es el modelo de caja, que define cómo se espacian y dimensionan los elementos en la página. Cada elemento HTML se representa como una caja rectangular que consta de cuatro áreas: contenido, *padding* (relleno), *border* (borde) y *margin* (margen).
- *Layout* y diseño responsivo: CSS permite crear layouts complejos y responsivos utilizando técnicas como Flexbox y CSS Grid. Estos modelos de diseño permiten alinear, distribuir y organizar elementos en una página de manera flexible y adaptativa, lo que es crucial para la creación de sitios web que funcionen bien en diferentes dispositivos y tamaños de pantalla.
- *Media queries*: son reglas CSS que permiten aplicar estilos específicos en función del tamaño de la pantalla o del dispositivo, asegurando que el contenido sea accesible y estéticamente agradable en dispositivos móviles, tabletas y monitores de escritorio.
- Transiciones y animaciones: CSS también permite añadir transiciones suaves y animaciones a los elementos de la página, mejorando la experiencia del usuario y la interactividad sin la necesidad de JavaScript. Sus propiedades de animación permiten definir cómo un elemento debe cambiar de un estado a otro a lo largo del tiempo.

HTML y CSS son las piedras angulares del desarrollo web, permitiendo a los desarrolladores construir y diseñar sitios web que son tanto funcionales como visualmente atractivos. Comprender su funcionamiento y cómo interactúan es crucial para crear experiencias web modernas, responsivas y accesibles [21].

4.3.8. Vercel

Vercel es una plataforma de despliegue y *hosting* que está optimizada para aplicaciones web modernas, especialmente aquellas construidas con *frameworks* y bibliotecas JavaScript como Next.js, React y Vue.js. Vercel permite a los desarrolladores desplegar aplicaciones de manera rápida y eficiente, proporcionando un entorno serverless que maneja automáticamente la infraestructura necesaria.

Arquitectura y funcionamiento

Vercel se basa en una arquitectura serverless, lo que significa que las aplicaciones desplegadas en la plataforma no requieren la administración de servidores por parte del desarrollador. En lugar de ello, Vercel utiliza funciones serverless que se ejecutan en la nube bajo demanda. Este enfoque ofrece varias ventajas, como la escalabilidad automática, menor tiempo de inactividad y una reducción en los costos operativos.

- Despliegue continuo: Vercel integra de manera nativa con sistemas de control de versiones como GitHub, GitLab y Bitbucket. Cada vez que se realiza un cambio en el código, Vercel puede desplegar automáticamente una nueva versión de la aplicación. Este proceso asegura que los desarrolladores puedan trabajar de manera ágil, con despliegues rápidos y continuos.

- Previsualizaciones de despliegue: Vercel genera una URL única para cada solicitud de extracción (*pull request*), lo que permite a los equipos revisar los cambios en un entorno real antes de fusionarlos en la rama principal. Esto mejora significativamente la colaboración entre desarrolladores, diseñadores y otros *stakeholders*.
- Optimización del rendimiento: Vercel está diseñado para maximizar el rendimiento de las aplicaciones web. Implementa estrategias avanzadas de caché, optimización automática de imágenes y minimización de recursos, lo que resulta en tiempos de carga más rápidos y una mejor experiencia de usuario.
- *Edge network*: Vercel utiliza una red global de distribución de contenido (CDN) que asegura que las aplicaciones se sirvan desde el punto más cercano al usuario final. Esta red de borde reduce la latencia y mejora la velocidad de entrega, garantizando que las aplicaciones funcionen de manera eficiente a escala global.
- Soporte para *server-side rendering* (SSR) y *static site generation* (SSG): Vercel soporta tanto el renderizado del lado del servidor (SSR) como la generación de sitios estáticos (SSG), lo que lo hace ideal para aplicaciones que requieren SEO avanzado y tiempos de carga rápidos. Estos enfoques permiten que la aplicación sea renderizada previamente o en el servidor, entregando HTML optimizado al navegador del usuario.

Escalabilidad y flexibilidad

La escalabilidad de Vercel es uno de sus aspectos más destacados. Ya que se basa en funciones serverless y una red de distribución global, las aplicaciones que se despliegan en Vercel pueden llegar a manejar picos de tráfico sin necesidad de intervención manual. También ofrece un entorno flexible que soporta una amplia gama de lenguajes de programación y *frameworks*, permitiendo a los desarrolladores elegir las herramientas que mejor se adapten a sus necesidades.

4.3.9. Hostinger

Hostinger es una empresa global de servicios de internet que ofrece soluciones de hosting y registro de dominios. Fundada en 2004, Hostinger ha crecido rápidamente y se ha establecido como uno de los principales proveedores de servicios de alojamiento web, con millones de usuarios en todo el mundo.

Sus servicios principales incluyen:

- Registro de dominios: proporciona una amplia gama de extensiones de dominio (.com, .net, .online, etc.) con opciones de renovación automática y protección de privacidad WHOIS.
- Alojamiento web: ofrece planes de hosting compartido, VPS, *hosting* en la nube y soluciones especializadas para WordPress, con una excelente relación calidad-precio y soporte técnico en varios idiomas.
- Constructor de sitios web: incluye herramientas intuitivas de arrastrar y soltar con plantillas modernas. Soporta funciones de SEO y comercio electrónico, ideales para crear sitios web de manera rápida y sin conocimientos técnicos.
- Correo electrónico profesional: brinda opciones para correos electrónicos personalizados con el dominio de tu empresa, mejorando la imagen profesional.
- Certificados SSL y seguridad: ofrece certificados SSL gratuitos en algunos de sus planes para asegurar la comunicación. También incluye funciones de protección avanzada contra malware y respaldos automáticos.

- Servicios de comercio electrónico: facilita la creación y administración de tiendas en línea con integración de pasarelas de pago y opciones de envío, ayudando a los emprendedores a comenzar sus negocios digitales.
- Soporte técnico: Hostinger proporciona soporte al cliente 24/7 a través de chat en vivo, con un enfoque en resolver problemas rápidamente y de forma amigable.

Hostinger se distingue por su enfoque en la accesibilidad, combinando precios competitivos con un servicio confiable y de alta calidad. La empresa ha sido pionera en la adopción de nuevas tecnologías, asegurando una experiencia de usuario óptima y velocidades de carga rápidas. Su compromiso con la satisfacción del cliente y la innovación constante lo ha posicionado como una alternativa popular para usuarios individuales y empresas que buscan una solución integral para establecer y mantener su presencia en línea de manera eficiente.

Hostinger continúa expandiendo su infraestructura y mejorando sus servicios para adaptarse a las necesidades de sus clientes, consolidándose como uno de los líderes en la industria de *hosting* y servicios web [22].

4.3.10. Cloudflare

Cloudflare es una plataforma de servicios de red que ofrece una amplia gama de soluciones de rendimiento y seguridad para sitios web, aplicaciones y redes. Fundada en 2009, Cloudflare se ha convertido en una de las plataformas líderes en la industria de servicios de internet, conocida por su capacidad para proteger sitios web de ataques cibernéticos, mejorar el rendimiento de las páginas y facilitar la gestión de dominios.

Principales servicios de Cloudflare

Cloudflare proporciona varias funciones clave que permiten una gestión eficaz de los sitios web:

- Red de fistribución de contenido (CDN): Cloudflare cuenta con una red global de servidores que cachea contenido y lo entrega al usuario desde el servidor más cercano a su ubicación. Esto reduce el tiempo de carga de las páginas y mejora la experiencia del usuario.
- Protección DDoS: la plataforma ofrece protección avanzada contra ataques de denegación de servicio distribuida (DDoS), mitigando el riesgo de que el sitio se vea interrumpido por tráfico malicioso.
- Certificados SSL: Cloudflare permite la habilitación de HTTPS en el sitio mediante certificados SSL gratuitos, garantizando una conexión segura y cifrada entre el usuario y el servidor.
- Firewall de Aplicaciones Web (WAF): este *firewall* protege el sitio de amenazas comunes, como inyecciones SQL y ataques de scripts entre sitios (XSS), bloqueando solicitudes maliciosas antes de que lleguen al servidor.
- Gestión de DNS: Cloudflare proporciona un servicio de DNS rápido y confiable, permitiendo a los usuarios gestionar sus dominios y realizar cambios en registros DNS de manera sencilla.
- Optimización del rendimiento: la plataforma ofrece herramientas de optimización de rendimiento, como la compresión de archivos y el ajuste automático de imágenes, mejorando la velocidad de carga del sitio.

Cloudflare permite la gestión integral de un dominio propio mediante la combinación de seguridad avanzada y mejoras en el rendimiento, sin necesidad de hardware o *software* adicional. Esto lo convierte en una herramienta versátil y esencial para cualquier administrador de sitios web que busque maximizar la seguridad, la rapidez y la fiabilidad de su sitio[23].

4.3.11. ArcGIS

ArcGIS es una herramienta esencial en el campo de los sistemas de información geográfica, ofreciendo a usuarios y organizaciones la capacidad de visualizar, analizar y comprender datos geoespaciales en múltiples niveles. Su capacidad para integrar y analizar grandes volúmenes de datos geográficos es fundamental para la planificación, gestión de recursos y toma de decisiones en una variedad de campos.

Al combinar herramientas avanzadas de análisis espacial con la capacidad de compartir y colaborar en datos a través de plataformas en la nube, ArcGIS facilita la creación de soluciones geoespaciales que pueden ser aplicadas en tiempo real. Esto permite a las organizaciones responder de manera más efectiva a los desafíos globales relacionados con el medio ambiente, la infraestructura, la salud y la seguridad.

ArcGIS es una plataforma integral para la gestión y análisis de datos geoespaciales, utilizada en una amplia gama de industrias para mejorar la toma de decisiones basada en la ubicación. Con su suite de herramientas que abarca desde el análisis en escritorio hasta aplicaciones móviles y servicios en la nube, ArcGIS permite a los usuarios capturar, gestionar, analizar y compartir información geográfica de manera efectiva y eficiente. Su papel en la visualización y análisis de datos geográficos es crucial para la planificación, el desarrollo y la gestión sostenible de recursos a nivel global [24].

4.3.12. CI/CD

CI/CD (integración continua/despliegue continuo) es un conjunto de prácticas modernas en el desarrollo de *software* que automatizan y mejoran los procesos de integración de código, pruebas y despliegue de aplicaciones. Estas prácticas buscan aumentar la eficiencia y la calidad del desarrollo de *software*, al mismo tiempo que reducen los riesgos asociados con la entrega de *software*. CI/CD es fundamental en la metodología DevOps, que promueve la colaboración entre los equipos de desarrollo y operaciones.

Integración continua (*CI - continuous integration*)

La integración continua es una práctica en la que los desarrolladores integran cambios de código en un repositorio central con frecuencia, a menudo varias veces al día. Cada integración es validada mediante la ejecución automática de pruebas y construcciones (*builds*), lo que permite detectar errores en una etapa temprana del desarrollo.

El objetivo principal de CI es identificar y solucionar problemas de integración lo más pronto posible, evitando el ïnfierno de la integración", que ocurre cuando los desarrolladores integran grandes volúmenes de código no probado en la rama principal después de largos periodos de trabajo aislado.

Las herramientas de CI, como Jenkins, Travis CI, CircleCI, GitLab CI/CD y GitHub *Actions*, permiten la automatización de este proceso, ejecutando scripts que compilan el código, ejecutan pruebas unitarias y de integración y generan reportes sobre el estado de la integración.

Despliegue continuo (*CD - continuous deployment*)

El despliegue continuo es la práctica de automatizar el proceso de despliegue de una aplicación en diferentes entornos (desarrollo, pruebas, producción). En un flujo de trabajo de despliegue continuo, los cambios que pasan las pruebas automáticas en el entorno de integración se despliegan automáticamente a un entorno de producción o preproducción.

El objetivo de CD es reducir el tiempo que toma poner en producción nuevos cambios de código, facilitando la entrega de nuevas funcionalidades y mejoras a los usuarios de manera más rápida y frecuente.

CD minimiza el riesgo de despliegues fallidos y facilita la detección rápida de errores en producción ya que los cambios son pequeños y frecuentes. Herramientas como Kubernetes, Docker y AWS CodeDeploy son comúnmente utilizadas para orquestar y gestionar los despliegues automáticos.

Entrega continua (*continuous delivery*)

La entrega continua es una extensión de la integración continua que asegura que el código que se integra en la rama principal esté siempre listo para ser desplegado en producción. A diferencia del despliegue continuo, donde el código se despliega automáticamente, en la entrega continua, los cambios se despliegan manualmente pero con la certeza de que el código está siempre en un estado que puede ser desplegado.

La entrega continua proporciona un alto grado de control, permitiendo a los equipos decidir cuándo desplegar, pero con la confianza de que cualquier versión del código puede ser enviada a producción en cualquier momento.

El marco de CI/CD es fundamental en la entrega moderna de *software*, permitiendo a las organizaciones entregar *software* de alta calidad de manera rápida, eficiente y segura. A través de la integración y despliegue continuos, los equipos pueden responder más rápidamente a las necesidades del mercado, mejorar la colaboración entre departamentos y reducir los riesgos asociados con la integración y despliegue de código. Aunque su implementación puede presentar desafíos, los beneficios a largo plazo en términos de calidad, velocidad y eficiencia hacen que CI/CD sea una práctica esencial en cualquier organización que aspire a mantenerse competitiva en el mundo del desarrollo de *software* [6].

4.4. Marco conceptual de la agricultura inteligente

4.4.1. Definición y alcance de la agricultura inteligente

La Agricultura Inteligente utiliza tecnologías avanzadas como el internet de las cosas (IoT), la inteligencia artificial (AI) y los sistemas de información geográfica (GIS) para mejorar la eficiencia y la productividad de las prácticas agrícolas. Estas tecnologías facilitan una agricultura más precisa y controlada, lo que permite optimizar el uso de recursos y mejorar la sostenibilidad de las operaciones agrícolas [12].

4.4.2. Tecnologías fundamentales

- IoT y sensores: los dispositivos IoT y los sensores permiten la recopilación en tiempo real de datos críticos del campo, como condiciones del suelo, clima y salud de los cultivos, lo que es esencial para la toma de decisiones informada.

- Drones y robótica: los drones proporcionan imágenes aéreas que ayudan en la monitorización y gestión de grandes extensiones de tierra, mientras que la robótica juega un papel crucial en la automatización de tareas como la siembra y la cosecha.
- Análisis de datos e IA: las tecnologías de análisis de datos y AI interpretan grandes conjuntos de datos agrícolas para prever tendencias, optimizar las operaciones y aumentar la producción.

4.4.3. Beneficios de la agricultura inteligente:

Los principales beneficios incluyen aumentos significativos en la eficiencia y la producción, mejor gestión de los recursos naturales y reducción del impacto ambiental. Además, la capacidad de prever y mitigar problemas potenciales puede llevar a una mayor estabilidad en los rendimientos agrícolas [12].

4.4.4. Desafíos y Consideraciones

- Adopción tecnológica: los costos iniciales y la curva de aprendizaje pueden ser obstáculos significativos, especialmente en regiones menos desarrolladas.
- Cuestiones de seguridad y privacidad: la gestión de grandes volúmenes de datos agrícolas implica riesgos significativos en términos de seguridad y privacidad de los datos.
- Impacto social y económico: mientras que la agricultura inteligente puede incrementar la productividad, también plantea preguntas sobre el impacto en el empleo agrícola tradicional y las habilidades requeridas para la nueva era tecnológica.

4.4.5. Futuro de la agricultura inteligente

Explorar las tendencias futuras, como el aumento de la automatización y la integración de tecnologías más avanzadas, es crucial. Estas innovaciones continuarán transformando el paisaje agrícola, haciendo que la agricultura no solo sea más productiva sino también más resiliente a los desafíos globales como el cambio climático y el crecimiento poblacional [25].

4.5. Impacto ambiental y sostenibilidad

4.5.1. Reducción del uso de recursos

La agricultura inteligente está revolucionando el uso de recursos en la agricultura mediante la implementación de tecnologías avanzadas que permiten una gestión más eficiente del agua, la energía y los fertilizantes. Los sistemas de riego automatizados, por ejemplo, utilizan sensores de suelo para medir la humedad y ajustar el riego según las necesidades reales de los cultivos. Esto no solo reduce el consumo de agua, sino que también minimiza la cantidad de agua desperdiciada. Un estudio muestra que el uso de estas tecnologías puede reducir significativamente el uso de agua en la agricultura, lo que es crucial en áreas propensas a la sequía o donde el agua es un recurso escaso [26].

Además, la precisión en la aplicación de fertilizantes es otra área donde la agricultura inteligente está haciendo grandes avances. Los sensores y los sistemas de dosificación automatizados permiten aplicar la cantidad exacta de fertilizantes necesarios, basados en los datos del suelo y las condiciones de crecimiento de los cultivos. Esto no solo mejora la eficiencia del uso de fertilizantes, reduciendo

el exceso que puede llevar a la contaminación del suelo y del agua, sino que también optimiza la nutrición de las plantas, potencialmente aumentando los rendimientos de los cultivos [26].

En cuanto al consumo de energía, la agricultura inteligente contribuye a una reducción significativa mediante la optimización de las operaciones agrícolas y la automatización de procesos que tradicionalmente consumen mucha energía. Por ejemplo, la utilización de drones para la supervisión de cultivos y la aplicación de tratamientos puede disminuir la necesidad de vehículos y maquinaria pesada que consumen grandes cantidades de combustible. Estos avances no solo reducen los costos operativos para los agricultores, sino que también contribuyen a la reducción de las emisiones de gases de efecto invernadero asociadas con la agricultura [26].

4.5.2. Minimización del impacto ambiental

La minimización del impacto ambiental es uno de los beneficios más significativos de la agricultura inteligente. La utilización de tecnologías como sensores y sistemas de gestión automatizados permite una aplicación más precisa de pesticidas y fertilizantes, reduciendo así el escurrimiento hacia ríos y acuíferos. Esto es crucial para proteger los ecosistemas acuáticos y reducir la contaminación del agua. Además, al ajustar la cantidad de insumos agrícolas a lo estrictamente necesario, se minimiza la alteración del suelo y se fomenta su salud a largo plazo, lo que a su vez contribuye a la captura de carbono y la reducción de las emisiones de gases de efecto invernadero [9].

Por otro lado, la adopción de prácticas de agricultura inteligente también ayuda a reducir la deforestación y la degradación de la tierra al hacer un uso más eficiente de las tierras agrícolas existentes. Las técnicas avanzadas como el mapeo de rendimiento y la monitorización satelital permiten a los agricultores optimizar sus rendimientos sin la necesidad de expandir sus tierras cultivables. Esto no solo ayuda a preservar los hábitats naturales y la biodiversidad, sino que también asegura la sostenibilidad de las prácticas agrícolas para futuras generaciones [9].

4.5.3. Conservación de la biodiversidad

La agricultura inteligente desempeña un papel crucial en la conservación de la biodiversidad mediante la implementación de prácticas agrícolas que respetan y fomentan la diversidad biológica tanto en la flora como en la fauna. Una de estas prácticas es la agricultura de conservación, que se centra en tres principios fundamentales: mínima perturbación mecánica del suelo, cobertura orgánica permanente del suelo y diversificación de especies a través de secuencias de cultivos variados. Este enfoque no solo ayuda a mantener la salud del suelo y aumentar la eficiencia del uso del agua y nutrientes, sino que también mejora la producción de cultivos de manera sostenida, contribuyendo significativamente a la preservación de los ecosistemas locales [27].

Además, el Fondo Mundial para la Naturaleza (WWF) promueve enfoques agroecológicos que apoyan la producción de alimentos de manera que refuerza, en lugar de explotar, la naturaleza. Estos sistemas de producción agrícola son diseñados para ser positivos para la naturaleza, fomentando prácticas que aumentan la biodiversidad y que son sostenibles a largo plazo. La adopción de tales métodos puede jugar un papel vital en el cambio hacia sistemas de alimentos que son beneficiosos tanto para las personas como para el planeta, asegurando así un futuro más sostenible para la agricultura [27].

4.5.4. Mejora de salud de suelo

La mejora de la salud del suelo es una prioridad en la agricultura inteligente, donde diversas tecnologías y prácticas están diseñadas para optimizar la estructura del suelo y su capacidad para

retener nutrientes y agua. Un ejemplo significativo es el uso de prácticas de manejo que incluyen la aplicación de materiales orgánicos como el mulching, el mantenimiento de residuos de cultivos y la incorporación de cultivos de cobertura. Estas prácticas no solo aumentan el contenido de carbono orgánico del suelo, sino que también mejoran su estructura, lo cual es esencial para la salud a largo plazo del suelo. Además, la diversificación de especies mediante rotaciones de cultivos contribuye a una mayor biodiversidad subterránea, lo que puede mejorar la resiliencia del suelo y su productividad [28].

Otro avance tecnológico en la agricultura inteligente es el desarrollo de hidrogeles basados en cobre que capturan el nitrato residual de los fertilizantes y lo transforman en amoníaco, que luego puede reutilizarse. Esta tecnología no solo minimiza la contaminación por nitratos, que es un problema ambiental significativo, sino que también mejora la eficiencia de los fertilizantes al mantener los nutrientes disponibles para las plantas durante más tiempo. Además, este sistema innovador ha demostrado ser capaz de mejorar los rendimientos de los cultivos sin aumentar el uso de nitrógeno, lo cual es crucial para la sostenibilidad a largo plazo de las prácticas agrícolas [28].

4.5.5. Reducción de emisiones de gases de efecto invernadero

La adopción de prácticas de agricultura inteligente ha demostrado ser una estrategia efectiva para reducir las emisiones de gases de efecto invernadero en la agricultura, un sector que tradicionalmente ha sido un importante contribuyente a estas emisiones. Una de las aproximaciones más prometedoras es el uso de tecnologías y prácticas de agricultura eficientes en gases de efecto invernadero, que podrían reducir las emisiones hasta en un 20 % para el año 2050. Estas incluyen la mejora de la gestión del nitrógeno, prácticas de cultivo más eficientes y el uso de energías renovables, lo que no solo disminuye la dependencia de combustibles fósiles sino que también mejora la eficiencia de uso de los recursos en las granjas [29].

Además, la agricultura climáticamente inteligente (CSA) ha surgido como una solución integral que aborda simultáneamente la productividad agrícola, la resiliencia al cambio climático y la reducción de emisiones. Este enfoque incluye prácticas como la agricultura de conservación, agroforestería y la gestión precisa del agua y fertilizantes, todas las cuales contribuyen a un sistema alimentario más sostenible y menos perjudicial para el medio ambiente. Estas prácticas no solo ayudan a reducir las emisiones de metano y óxido nitroso, sino que también aumentan la captura de carbono en el suelo, fortaleciendo la resiliencia de los ecosistemas agrarios ante los cambios climáticos [29].

4.5.6. Sostenibilidad a largo plazo

A medida que la agricultura enfrenta desafíos sin precedentes debido al cambio climático y el crecimiento de la población, la sostenibilidad a largo plazo se convierte en una prioridad clave. La agricultura inteligente ofrece soluciones tecnológicamente avanzadas que no solo aumentan la productividad agrícola, sino que también garantizan la sostenibilidad ambiental y económica. El enfoque en prácticas como la agricultura de precisión y el uso de sistemas integrados de gestión de datos permite optimizar el uso de recursos y minimizar el impacto ambiental, asegurando así la viabilidad a largo plazo del sector agrícola.

La integración de tecnologías de agricultura inteligente también promueve la resiliencia de los sistemas agrícolas frente a condiciones climáticas adversas y la variabilidad estacional. Por ejemplo, el uso eficiente del agua a través de sistemas de riego inteligentes y la aplicación precisa de nutrientes ayudan a mejorar la salud del suelo y aumentar la productividad de los cultivos, al tiempo que se reduce la dependencia de insumos químicos y el riesgo de contaminación. Además, la adopción de prácticas de agricultura climáticamente inteligente (CSA) permite una triple ganancia: aumenta la productividad, mejora la resiliencia y reduce las emisiones de gases de efecto invernadero, contribu-

yendo así a los objetivos globales de desarrollo sostenible y al cumplimiento de los compromisos del Acuerdo de París [30].

Estos enfoques representan un cambio fundamental en cómo se concibe y se implementa la agricultura moderna, posicionando la sostenibilidad a largo plazo como un pilar central en la planificación y gestión agrícola. Al adoptar estas tecnologías y prácticas, el sector agrícola no solo puede enfrentar los desafíos actuales sino también prepararse para las demandas futuras, asegurando la alimentación y el bienestar de las generaciones venideras.

4.5.7. Resiliencia climática

La resiliencia climática en la agricultura inteligente es fundamental para asegurar la sostenibilidad a largo plazo del sector agrícola frente a los desafíos del cambio climático. La agricultura climáticamente inteligente (CSA) propone un enfoque integral que no solo busca aumentar la productividad y los ingresos agrícolas, sino también adaptar y fortalecer la resiliencia frente a las variaciones climáticas y reducir las emisiones de gases de efecto invernadero. Este enfoque aborda los objetivos de desarrollo sostenible y los compromisos del Acuerdo de París, destacando la importancia de prácticas agrícolas que respondan eficazmente al cambio climático [10].

Las estrategias de CSA incluyen la adopción de variedades de cultivos resistentes al clima, técnicas de agricultura de conservación, agroforestería, agricultura de precisión y estrategias mejoradas de gestión del agua y del ganado. Estas prácticas no solo contribuyen a una mayor productividad y sostenibilidad, sino que también fortalecen la capacidad de los sistemas agrícolas para manejar los impactos adversos del clima, como sequías, plagas y enfermedades. Al implementar estas medidas, la agricultura puede mejorar su adaptabilidad y capacidad de recuperación ante las fluctuaciones climáticas, lo que es crucial para mantener la seguridad alimentaria y la estabilidad económica de las comunidades rurales en todo el mundo [10].

4.5.8. Economía circular en agricultura

La economía circular en la agricultura representa una transición hacia prácticas que no solo buscan la eficiencia y la sostenibilidad, sino que también integran el reciclaje y la reutilización de recursos dentro de los sistemas agrícolas. Este enfoque ayuda a cerrar el ciclo de vida de los insumos agrícolas, reduciendo así el desperdicio y minimizando la extracción de nuevos recursos. Por ejemplo, la implementación de sistemas integrados de agricultura orgánica que siguen el concepto de 5R (reducción, reutilización, reciclaje, renovación y rechazo de insumos inorgánicos) puede mejorar significativamente la sostenibilidad de las prácticas agrícolas, aumentando la diversidad de cultivos y mejorando la interacción entre diferentes componentes del sistema agrícola [31].

Además, la adopción de prácticas de economía circular en la agricultura puede generar beneficios ambientales significativos, incluyendo la mejora de la salud del suelo y la reducción de las emisiones de gases de efecto invernadero. Estas prácticas no solo conservan los recursos naturales, sino que también ofrecen ventajas económicas, como la reducción de costos y el aumento de la rentabilidad para los agricultores. La transformación hacia una economía circular en la agricultura es fundamental para enfrentar los desafíos ambientales actuales y garantizar una producción alimentaria sostenible y resiliente al cambio climático [10].

4.6. Desafíos y limitaciones tecnológicas

La implementación de soluciones de inteligencia artificial (IA) en el sector agrícola presenta una serie de desafíos y limitaciones que pueden variar desde cuestiones técnicas hasta resistencias sociales y económicas. A continuación, se analizan detalladamente estos desafíos:

4.6.1. Resistencia al cambio

Uno de los principales obstáculos para la adopción de tecnologías avanzadas en agricultura es la resistencia al cambio por parte de los agricultores, especialmente en regiones rurales o menos desarrolladas. Esta resistencia puede estar motivada por la falta de familiaridad con las nuevas tecnologías, el coste de inversión inicial elevado y la percepción de que las nuevas tecnologías podrían complicar en lugar de simplificar las prácticas agrícolas. Además, existe un componente cultural y generacional ya que muchos agricultores han estado utilizando métodos tradicionales durante décadas [32].

4.6.2. Necesidad de datos de alta calidad

Las soluciones de IA dependen en gran medida de la disponibilidad de grandes volúmenes de datos de alta calidad para entrenar modelos predictivos eficaces. En la agricultura, esto puede incluir datos sobre patrones climáticos, tipos de suelo, ciclos de cultivo y salud de las plantas. Sin embargo, la recopilación de estos datos puede ser un desafío debido a la variabilidad espacial y temporal de los campos agrícolas, así como a la falta de infraestructura de datos en áreas rurales [33].

4.6.3. Interpretación y uso de los datos

Incluso cuando los datos están disponibles, la interpretación y el uso eficaz de estos para mejorar las decisiones agrícolas no es trivial. Los agricultores y los gestores de las granjas necesitan capacitación y herramientas para interpretar correctamente los resultados proporcionados por los algoritmos de IA. Esta barrera se complica aún más por la heterogeneidad de las operaciones agrícolas y las condiciones locales, lo que puede dificultar la aplicación generalizada de soluciones basadas en IA sin una considerable personalización.

4.6.4. Integración con sistemas agrícolas existentes

La integración de nuevas tecnologías de IA con los sistemas y procesos agrícolas existentes puede presentar desafíos técnicos significativos. Los sistemas de IA deben ser compatibles con los equipos y *software* agrícolas existentes, lo que a menudo requiere interfaces personalizadas y soluciones de *software* adaptativo. Además, la falta de estándares tecnológicos en la agricultura puede limitar la capacidad de los sistemas de IA para funcionar de manera efectiva a través de diferentes plataformas y equipos [34].

4.6.5. Costos de implementación y escalabilidad

Los costos iniciales de implementación de sistemas de IA pueden ser prohibitivos para pequeños y medianos productores. Además, mientras que las soluciones de IA pueden ser altamente efectivas en grandes operaciones agrícolas, su escalabilidad en pequeñas granjas o en regiones con recursos

limitados sigue siendo una cuestión abierta. La relación costo-beneficio de estas tecnologías debe evaluarse cuidadosamente en diferentes contextos agrícolas para garantizar una adopción más amplia [35].

4.6.6. Cuestiones de privacidad y seguridad de datos

Con el aumento de la digitalización en la agricultura, surge la preocupación por la privacidad y seguridad de los datos agrícolas. Los agricultores deben confiar en que su información está segura y que no será utilizada para propósitos contrarios a sus intereses. Las regulaciones sobre datos y su manejo juegan un papel crucial aquí y la falta de claridad o la inconsistencia en estas puede disuadir a los agricultores de adoptar soluciones basadas en IA [36].

Estos desafíos requieren un enfoque multifacético que incluye no solo el desarrollo tecnológico, sino también soporte en la implementación, capacitación para los usuarios y políticas que faciliten la integración de la IA en la agricultura tradicional. La colaboración entre desarrolladores tecnológicos, agricultores, investigadores y legisladores será esencial para superar estos obstáculos y aprovechar plenamente el potencial de la IA para transformar la agricultura hacia prácticas más sostenibles y productivas.

4.7. Conceptos claves extras

4.7.1. Servidores de nombres

Los servidores de nombres (*nameservers*) son servidores que funcionan como una especie de "guía telefónica"para los dominios de internet. Su función principal es traducir los nombres de dominio, como "dominio.com", en direcciones IP, que son las que los navegadores y otros servicios de internet necesitan para localizar y cargar el contenido del sitio web asociado a ese dominio.

¿Cómo funcionan los servidores de nombres?

Cuando un usuario escribe el nombre de un dominio en su navegador, la solicitud primero se dirige a los servidores de nombres para averiguar la dirección IP correspondiente a ese dominio. Los servidores de nombres contienen registros DNS que especifican esta información, permitiendo que la solicitud se redirija al servidor adecuado para cargar la página web.

Rol de los servidores de nombres en el proceso

- Redirigir el tráfico: al actualizar los servidores de nombres para que apunten a los de Cloudflare, indicas que Cloudflare será el encargado de gestionar el tráfico y la resolución de DNS para tu dominio. A partir de entonces, cualquier solicitud para "dominio.com"pasará por Cloudflare antes de llegar al servidor donde está alojado el sitio.
- Facilitar la gestión de DNS y seguridad: con los servidores de nombres de Cloudflare configurados, podrás aprovechar sus servicios de seguridad y rendimiento, como la protección contra ataques DDoS, la optimización del rendimiento y la administración simplificada de los registros DNS.
- Habilitar funciones adicionales: cambiar los servidores de nombres permite que Cloudflare actúe como un intermediario entre el usuario y el servidor de alojamiento (por ejemplo, Hostinger o

Vercel), brindando acceso a funcionalidades como el CDN, los certificados SSL y el firewall de aplicaciones.

En conclusión, al apuntar los servidores de nombres de tu dominio hacia los de Cloudflare, estamos delegando a esta plataforma el control sobre cómo se dirige y protege el tráfico hacia nuestro sitio, permitiéndonos acceder a una gama de herramientas avanzadas de gestión de dominios y seguridad [23].

4.7.2. API

Definición y concepto de API

Las API (interfaces de programación de aplicaciones) son mecanismos que permiten la comunicación entre dos componentes de *software* mediante un conjunto de definiciones y protocolos. Funcionan como un contrato de servicio entre aplicaciones, definiendo cómo se comunican entre sí a través de solicitudes y respuestas.

Funcionamiento de las API

Las API operan bajo una arquitectura cliente-servidor:

- El cliente envía una solicitud al servidor.
- El servidor procesa la solicitud y devuelve una respuesta.

Este proceso se realiza mediante protocolos estandarizados, siendo HTTP el más común en las API web modernas.

Tipos de API

Según su arquitectura:

- API de SOAP: utilizan el protocolo simple de acceso a objetos.
- API de RPC: basadas en llamadas a procedimientos remotos.
- API de WebSocket: permiten comunicación bidireccional en tiempo real.
- API de REST: las más populares, basadas en transferencia de estado representacional.

Según su ámbito de uso:

- API privadas: uso interno en una empresa.
- API públicas: abiertas al público general.
- API de socios: acceso limitado a desarrolladores autorizados.
- API compuestas: combinan múltiples API para tareas complejas.

API REST

Las API REST (*Representational State Transfer*) son las más utilizadas en la web actual.

Características principales:

- Sin estado: no guardan datos del cliente entre solicitudes.
- Utilizan métodos HTTP estándar (GET, POST, PUT, DELETE, etc.).
- Intercambian datos en formatos como JSON o XML.

Beneficios de las API

- Integración: facilitan la conexión entre sistemas existentes.
- Innovación: permiten implementar nuevos servicios rápidamente.
- Ampliación: posibilitan el acceso a funcionalidades en múltiples plataformas.
- Facilidad de mantenimiento: permiten cambios internos sin afectar a otros sistemas.

4.7.3. FastAPI

FastAPI es un *framework* web de Python moderno, rápido y altamente eficiente, diseñado específicamente para la creación de API. Desarrollado por Sebastián Ramírez en 2018, FastAPI se ha ganado rápidamente una reputación en la comunidad de desarrolladores por su combinación única de velocidad, facilidad de uso y robustez. El *framework* se basa en Starlette para las funcionalidades web asíncronas y Pydantic para la validación de datos, lo que le permite ofrecer un rendimiento excepcional comparable a *frameworks* en NodeJS y Go.

Una de las características más destacadas de FastAPI es su aprovechamiento del sistema de tipos de Python. Utilizando anotaciones de tipo estándar de Python 3.6+, FastAPI puede realizar automáticamente la validación de datos, serialización y generación de documentación OpenAPI (anteriormente conocida como Swagger). Esto no solo mejora la seguridad y la fiabilidad del código, sino que también proporciona una experiencia de desarrollo superior con autocompletado en IDE y detección temprana de errores.

El *framework* adhiere estrictamente a los estándares OpenAPI y JSON Schema, lo que facilita la interoperabilidad con otras herramientas y servicios. La documentación automática generada por FastAPI es interactiva y permite a los desarrolladores y usuarios de la API probar *endpoints* directamente desde la interfaz web, acelerando significativamente el proceso de desarrollo y pruebas.

FastAPI también se destaca por su soporte nativo para operaciones asíncronas. Aprovechando las características async/await de Python, permite manejar un gran número de conexiones simultáneas con un uso eficiente de los recursos del sistema. Esto lo hace particularmente adecuado para aplicaciones que requieren alta concurrencia, como servicios en tiempo real o sistemas que manejan un gran volumen de solicitudes.

El sistema de inyección de dependencias de FastAPI merece una mención especial. Permite una gestión limpia y eficiente de las dependencias del proyecto, facilitando la implementación de patrones como la inversión de control (IoC) y mejorando la modularidad y la capacidad de prueba del código.

Además, FastAPI ofrece características avanzadas como el manejo de *websockets*, tareas en segundo plano y soporte para GraphQL, lo que lo convierte en una opción versátil para una amplia gama de

aplicaciones web modernas. Su integración con herramientas populares del ecosistema Python, como SQLAlchemy para ORM y Alembic para migraciones de bases de datos, permite a los desarrolladores construir aplicaciones completas y escalables con relativa facilidad.

En términos de seguridad, FastAPI incluye soporte integrado para autenticación y autorización, incluyendo OAuth2 con JWT tokens. También proporciona protección contra vulnerabilidades web comunes, como CORS (cross-origin resource sharing) y limitación de tasa de solicitudes.

La curva de aprendizaje relativamente suave de FastAPI, combinada con su potencia y flexibilidad, lo ha convertido en una elección popular tanto para desarrolladores experimentados como para principiantes en el desarrollo de API. Su enfoque en la productividad del desarrollador, sin comprometer el rendimiento o la escalabilidad, lo posiciona como una herramienta valiosa en el panorama actual del desarrollo web y de microservicios [37].

4.7.4. Google Lighthouse

Introducción a la optimización del rendimiento web

La optimización del rendimiento web es un aspecto crítico del desarrollo web moderno, centrado en mejorar la velocidad, la eficiencia y la experiencia general del usuario en los sitios web. A medida que Internet continúa evolucionando y las expectativas de los usuarios crecen, la importancia de ofrecer experiencias web rápidas, accesibles y de alta calidad se ha vuelto primordial.

El papel de Google Lighthouse

Google Lighthouse surge como una herramienta fundamental en el ecosistema de optimización del rendimiento web. Desarrollada por Google como un proyecto de código abierto, Lighthouse sirve como una herramienta de auditoría integral que evalúa los sitios web en múltiples dimensiones:

- Rendimiento
- Accesibilidad
- Mejores Prácticas
- Optimización para motores de búsqueda (SEO)

Al proporcionar información y recomendaciones en estas áreas, Lighthouse tiende un puente entre la implementación técnica y los principios de diseño centrados en el usuario.

Core web vitals y experiencia del usuario

Un aspecto central del rendimiento en Lighthouse son las Core Web Vitals, un conjunto de factores específicos que Google considera importantes en la experiencia general del usuario en una página web. Estos incluyen:

- *Largest contentful paint* (LCP): mide el rendimiento de carga
- *First input delay* (FID): mide la interactividad
- *Cumulative layout shift* (CLS): mide la estabilidad visual

Estas métricas proporcionan una base cuantificable para evaluar y mejorar la experiencia del usuario.

Accesibilidad y diseño inclusivo

La auditoría de accesibilidad de Lighthouse refleja la creciente importancia del diseño inclusivo en el desarrollo web. Al evaluar elementos como la descripción de botones y enlaces y el texto alternativo para imágenes, Lighthouse promueve prácticas que hacen que los sitios web sean utilizables por una gama más amplia de usuarios, incluidas las personas con discapacidades.

Mejores prácticas y seguridad web

La categoría de Mejores Prácticas en Lighthouse abarca una amplia gama de consideraciones técnicas, desde la seguridad (como el uso de HTTPS) hasta la optimización de imágenes y la prevención de vulnerabilidades. Este enfoque subraya la importancia de adoptar estándares modernos de desarrollo web para crear sitios robustos y seguros.

SEO técnico y visibilidad en línea

La auditoría SEO de Lighthouse se centra en los aspectos técnicos de la optimización para motores de búsqueda. Al evaluar factores como la adaptabilidad móvil, los datos estructurados y la indexabilidad, Lighthouse ayuda a los desarrolladores y propietarios de sitios web a mejorar la visibilidad de sus sitios en los resultados de búsqueda.

Integración con el ecosistema de desarrollo web

Lighthouse se integra perfectamente con las herramientas de desarrollo web existentes, como Chrome DevTools y Node.js. Esta integración facilita la incorporación de auditorías de rendimiento en los flujos de trabajo de desarrollo, promoviendo una cultura de mejora continua.

Google Lighthouse representa un enfoque holístico para la evaluación y mejora de sitios web. Al proporcionar métricas cuantificables y recomendaciones procesables en múltiples dimensiones de la calidad web, Lighthouse se ha convertido en una herramienta esencial para desarrolladores, diseñadores y profesionales del SEO. Su enfoque en el rendimiento, la accesibilidad y las mejores prácticas refleja las prioridades actuales en el desarrollo web. En un panorama digital en constante evolución, Lighthouse sirve como faro, guiando el camino hacia experiencias web más rápidas, accesibles y efectivas [38].

4.7.5. Linting

El uso de ESLint como herramienta de análisis de código estático se ha vuelto fundamental en el desarrollo moderno de JavaScript. Su configuración se realiza principalmente a través de un archivo .eslintrc que puede estar en formato JSON o JavaScript, donde las reglas pueden configurarse en tres niveles: desactivadas, advertencias o errores. Una de sus características más destacadas es la capacidad de extender configuraciones populares como las de Airbnb, Google o el JavaScript Standard Style, lo que facilita la adopción de estándares probados.

En cuanto a sus funcionalidades principales, ESLint sobresale en la detección de errores potenciales, identificando variables no utilizadas, problemas con promesas y uso inadecuado de declaraciones,

mientras que simultáneamente vigila aspectos de estilo como el uso consistente de comillas y el espaciado correcto. La integración con el entorno de desarrollo es otro punto fuerte ya que se puede incorporar fácilmente en editores populares como VS Code o WebStorm, además de funcionar perfectamente en *pipelines* de integración continua y *scripts* de npm.

Los beneficios que aporta al desarrollo son sustanciales: mejora la calidad del código, reduce errores en producción, mantiene la consistencia en equipos grandes y facilita la incorporación de nuevos desarrolladores al proyecto. En el aspecto más avanzado, ESLint ofrece soporte para TypeScript, permite la creación de reglas personalizadas mediante plugins y proporciona la flexibilidad de ignorar archivos o directorios específicos cuando sea necesario.

El proceso de corrección es igualmente robusto, ofreciendo tanto corrección automática a través del comando *–fix* como sugerencias en tiempo real, todo esto respaldado por una documentación detallada que incluye ejemplos de código correcto e incorrecto para cada regla [39].

4.7.6. Tobii

El *eye tracking* o seguimiento ocular es una técnica que permite registrar y analizar los movimientos oculares de una persona mientras observa o interactúa con un estímulo visual. En este contexto, los dispositivos Tobii eye tracker han revolucionado la forma en que se estudia el comportamiento visual humano, destacándose como una de las tecnologías más precisas y confiables en el mercado. Estos dispositivos utilizan tecnología basada en reflexión corneal e iluminación infrarroja para detectar y registrar la dirección de la mirada con una precisión notable.

La tecnología de Tobii funciona mediante un sistema de emisores infrarrojos que proyectan luz hacia los ojos del usuario, mientras que sensores especializados captan los reflejos de esta luz en la córnea y la pupila. Mediante algoritmos avanzados de procesamiento de imágenes y cálculos geométricos, el sistema puede determinar con exactitud el punto exacto donde se fija la mirada del usuario. Este proceso se realiza varias veces por segundo, permitiendo un seguimiento fluido y en tiempo real de los movimientos oculares.

En el ámbito de la investigación científica, los *eye trackers* de Tobii han demostrado ser herramientas invaluables para diversos campos de estudio. En psicología cognitiva, permiten analizar patrones de atención visual y procesos de toma de decisiones. En el campo del marketing y la publicidad, estos dispositivos facilitan la comprensión de cómo los consumidores interactúan visualmente con diferentes elementos publicitarios, empaques y diseños. En el contexto educativo, el eye tracking ayuda a comprender los procesos de lectura y aprendizaje, permitiendo desarrollar materiales didácticos más efectivos.

La implementación de esta tecnología ha evolucionado significativamente desde sus inicios. Los modernos *eye trackers* de Tobii son cada vez más compactos y menos intrusivos, permitiendo estudios en condiciones más naturales. Además, el *software* asociado ha mejorado considerablemente, ofreciendo análisis más sofisticados que incluyen mapas de calor, gráficos de recorrido visual y métricas cuantitativas detalladas. Estas características han expandido las posibilidades de investigación y han hecho que la tecnología sea más accesible para diferentes campos de aplicación.

La precisión y fiabilidad de los datos obtenidos mediante eye trackers Tobii dependen de varios factores técnicos y ambientales. La calibración adecuada del dispositivo, las condiciones de iluminación y la posición del participante son elementos cruciales para obtener resultados óptimos. Los investigadores deben considerar también las limitaciones inherentes a la tecnología, como la variabilidad entre participantes y las posibles interferencias causadas por el uso de lentes o condiciones oculares específicas.

La interpretación de los datos proporcionados por los eye trackers requiere un entendimiento profundo tanto de la tecnología como del contexto de investigación. Las métricas más comunes in-

cluyen fijaciones (momentos en que la mirada se detiene en un punto), sacadas (movimientos rápidos entre fijaciones) y tiempo de permanencia en áreas de interés. Estos datos, cuando se analizan en conjunto, proporcionan información valiosa sobre el comportamiento visual y los procesos cognitivos subyacentes [40].

CAPÍTULO 5

Antecedentes

El desarrollo de tecnologías avanzadas en la agricultura ha sido un factor clave para mejorar la productividad y sostenibilidad de los cultivos, especialmente en el contexto de la caña de azúcar. La integración de la inteligencia artificial (IA) en el monitoreo y gestión de cultivos ha permitido a los agricultores tomar decisiones informadas y optimizar el uso de recursos. A lo largo de los años, han surgido diversas soluciones tecnológicas que buscan abordar los desafíos en la agricultura, desde la detección de enfermedades hasta la optimización del rendimiento de los cultivos.

El proyecto AgroIntelligence se posiciona en este contexto como una iniciativa pionera, centrada en el desarrollo de un *dashboard* avanzado para el manejo y análisis de datos generados por modelos de IA dedicados a la detección de enfermedades en la caña de azúcar. Este proyecto no solo se enfoca en la creación de una plataforma para la visualización de datos, sino que también integra modelos de aprendizaje automático que permiten a los usuarios identificar y reaccionar rápidamente a posibles problemas en sus cultivos.

En la revisión de antecedentes, se identificaron diversas soluciones que han influido en el desarrollo del proyecto AgroIntelligence. Por ejemplo, la startup sudafricana Aerobotics se especializa en proporcionar análisis detallados de árboles y frutas mediante imágenes de drones e IA, permitiendo a los agricultores optimizar el rendimiento de sus cultivos a través del análisis de patrones de crecimiento en tiempo real [41]. De manera similar, la plataforma HEMAV utiliza técnicas de IA y aprendizaje automático para proporcionar a los agricultores datos específicos y accionables sobre la salud de los cultivos y la humedad del suelo, entre otros factores, lo que mejora significativamente la productividad y rentabilidad [42].

Otro referente es el proyecto *Smart farming robot for detecting environmental conditions in a greenhouse*, el cual emplea un robot autónomo con sensores inalámbricos para recopilar datos ambientales en invernaderos, utilizando algoritmos de aprendizaje no supervisado para identificar patrones de crecimiento inapropiados en los cultivos [12]. Además, aplicaciones como PlantVillage permiten a los agricultores tomar fotos de sus cultivos y obtener diagnósticos de posibles enfermedades, utilizando modelos entrenados con imágenes tomadas por drones [43].

Estos antecedentes subrayan la relevancia y necesidad de desarrollar herramientas tecnológicas que, como AgroIntelligence, permitan a los agricultores gestionar sus cultivos de manera más eficiente y efectiva, aprovechando los avances en inteligencia artificial y análisis de datos.

CAPÍTULO 6

Metodología

6.1. Descripción detallada del uso empleado a las herramientas elegidas para la construcción del panel de control

6.1.1. Github

El proceso comenzó con la configuración del control de versiones utilizando Git y la creación de un repositorio en GitHub para almacenar y gestionar el código fuente de la aplicación.

6.1.2. React

Para crear React en nuestra aplicación el primer paso fue instalar React en nuestro proyecto, para ello utilizamos el comando:

- npm create vite@latest

El cual nos permitió crear un proyecto desde cero, en el cual definimos React como el marco de trabajo y Vite como la herramienta de desarrollo de la cual también se hablará a detalle más adelante, esta opción nos permitió definir todos los parámetros deseados desde el inicio del desarrollo.

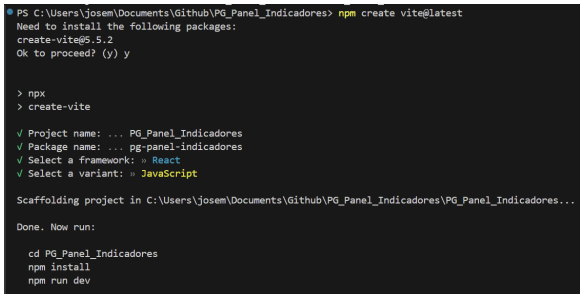


Figura 1. Vite - creación de proyecto

Durante la ejecución del comando fue necesario elegir una plantilla para el proyecto creado, en la figura mostrada se muestran todas las opciones posibles para la creación de un proyecto con Vite y una de ella es elegir React, seleccionar un lenguaje principal de programación, en nuestro caso JavaScript.

Una vez configurado el entorno inicial, se procedió con las configuraciones específicas del proyecto en relación con React. Un aspecto clave fue la separación del código y las configuraciones del proyecto en distintas carpetas, para mantener una estructura ordenada y escalable. Para esto, se creó la carpeta "src", la cual es comúnmente utilizada para alojar el código fuente en la mayoría de los proyectos React.

Dentro de la carpeta "src", el código se distribuyó en diferentes secciones para facilitar la organización y el mantenimiento. En la raíz de esta carpeta, se encuentra el archivo .ªpp.jsx", que define el componente principal de la aplicación y el archivo "main.jsx", que se encarga de montar la aplicación en el DOM, utilizando la función ReactDOM.render".

Para la creación de las distintas páginas de la aplicación, se decidió agruparlas en una carpeta llamada *pages*. Esta carpeta contiene todas las páginas individuales, desde la página de inicio (Home) hasta las páginas dedicadas a los modelos de análisis y a la API. Esta separación permite un desarrollo modular, facilitando tanto la navegación como el mantenimiento del código.

Además, se creó una carpeta específica para los componentes reutilizables, denominada *components*. En esta carpeta, se encuentran elementos clave como la barra de navegación *navbar* y el pie de página *footer*, los cuales se combinan posteriormente en un *layout* general que se aplica al resto de las plantillas. Esta organización facilita la reutilización de código y mantiene una consistencia visual en toda la aplicación. Se tienen también carpetas de componentes específicos para cada página individual.

6.1.3. Vite

Vite es una herramienta de desarrollo seleccionada para este proyecto debido a su capacidad para proporcionar un entorno de desarrollo ágil y eficiente. Una de las principales ventajas de Vite es su rapidez ya que utiliza un servidor de desarrollo optimizado y un sistema de recarga en caliente que permite ver los cambios en tiempo real sin necesidad de recargar toda la página. Esto es particularmente útil en un proyecto React, donde la interacción dinámica y la actualización rápida de la interfaz de usuario son esenciales para un flujo de trabajo eficiente.

Además de mejorar la experiencia de desarrollo, Vite también ofrece un sistema de empaquetado y optimización de código listo para producción, lo cual es crucial al momento de desplegar la aplicación. La configuración de Vite se maneja a través del archivo "vite.config.js", el cual permite personalizar diversos aspectos del proyecto, como la definición de alias para rutas, la configuración de *plugins* y la optimización de la compilación. En este archivo, se pueden ajustar parámetros específicos para mejorar tanto el rendimiento del desarrollo como el del producto final.

Por ejemplo, en "vite.config.js"se pueden definir configuraciones relacionadas con la estructura de la carpeta de salida *output*, la integración con otras herramientas o cuadros de trabajo y ajustes para manejar archivos estáticos. Esta flexibilidad hace que Vite no solo sea una herramienta rápida y eficiente durante el desarrollo, sino también robusta y personalizable para adaptarse a las necesidades específicas del proyecto.

6.1.4. Node.js

Node.js juega un papel fundamental en el desarrollo de este proyecto ya que proporciona el entorno de ejecución necesario para ejecutar JavaScript fuera del navegador, lo que permite tanto el desarrollo como la compilación eficiente de la aplicación. En este proyecto, Node.js es esencial para gestionar las dependencias, ejecutar *scripts* de desarrollo y compilar la aplicación para producción.

Uno de los aspectos más importantes de Node.js en este contexto es su integración con herramientas como Vite y ESLint. En el archivo "package.json", se pueden observar varios scripts que dependen de Node.js para ejecutar comandos clave en el proyecto. Por ejemplo, el comando "dev": "vite.ejecuta el servidor de desarrollo de Vite, mientras que "build": "vite buildçompila la aplicación para producción. Además, se utiliza ESLint para mantener la calidad del código, ejecutando el *script* "lint", que verifica y corrige problemas de estilo y errores en los archivos JavaScript y JSX.

En cuanto a las dependencias, se incluyen varias librerías esenciales para la funcionalidad del proyecto. Entre las principales se encuentran:

- @fortawesome/fontawesome-svg-core, @fortawesome/free-brands-svg-icons, @fortawesome/free-solid-svg-icons y @fortawesome/react-fontawesome: Estas librerías proporcionan íconos y recursos visuales para mejorar la interfaz de usuario, integrando Font Awesome con React de manera eficiente.
- Bootstrap y react-bootstrap: Estas librerías ofrecen una solución rápida y eficaz para el diseño responsivo, utilizando componentes preconstruidos que facilitan la creación de una interfaz de usuario atractiva y funcional.
- react, react-dom y react-router-dom: Son los pilares del desarrollo en React, permitiendo la creación de componentes, la interacción con el DOM y la gestión de rutas en la aplicación.

En cuanto a las dependencias de desarrollo, se destacan herramientas como ESLint y los *plugins* relacionados con React, que ayudan a mantener la calidad del código, asegurando que se sigan las mejores prácticas. El plugin de React para Vite (vitejs/plugin-react) es también esencial para optimizar la integración de React con Vite, mejorando el rendimiento del proyecto.

6.1.5. Html y CSS

HTML y CSS son fundamentales en la estructura y presentación visual de la aplicación, proporcionando la base sobre la cual se construye y estiliza la interfaz de usuario. Aunque el uso de React permite una mayor modularidad y dinamismo en el desarrollo de la aplicación, HTML y CSS siguen siendo esenciales para definir la estructura del contenido y el estilo visual que los usuarios experimentan al interactuar con el panel de control.

HTML

En el contexto de este proyecto, HTML se utiliza principalmente dentro de los componentes de React para estructurar el contenido de las páginas. Aunque React abstrae parte de la sintaxis HTML a través de JSX (JavaScript XML), la esencia del marcado HTML sigue siendo la base para la creación de elementos en la interfaz. Cada componente de React incluye su propio marcado HTML en JSX, definiendo la disposición de elementos como encabezados, párrafos, tablas, formularios y botones, entre otros.

Por ejemplo, en la creación de páginas individuales y componentes reutilizables, se utiliza HTML semántico para asegurar que la estructura del contenido sea clara y accesible. Esto incluye el uso

adecuado de etiquetas como <header>, <nav>, <main> y <footer>, lo cual no solo mejora la accesibilidad, sino también la optimización para motores de búsqueda (SEO).

CSS

El diseño visual de la aplicación se gestiona mediante CSS, asegurando que la interfaz sea atractiva, coherente y responsiva en diferentes dispositivos. En este proyecto, se hizo uso de CSS tanto de manera directa como a través de la integración con marcos de trabajo como Bootstrap, que facilita la creación de un diseño responsivo con una amplia variedad de componentes predefinidos.

Para personalizar aún más el diseño y adaptarlo a las necesidades específicas del proyecto, se utilizó CSS personalizado. Esto incluyó la definición de estilos específicos para la paleta de colores, tipografías, márgenes y disposición de elementos en la pantalla. También se emplearon técnicas de diseño responsivo, como el uso de *media queries*, para asegurar que la aplicación se visualice correctamente en dispositivos de diferentes tamaños, desde computadoras de escritorio hasta teléfonos móviles.

Además, el CSS se organizó de manera modular para facilitar su mantenimiento. Cada componente o página tiene sus propios estilos, evitando conflictos y permitiendo una mayor flexibilidad en el diseño. Esto se logró tanto mediante hojas de estilo independientes como mediante el uso de CSS en línea en los archivos JSX cuando fue necesario para estilos específicos.

Integración con bootstrap

El uso de Bootstrap y su variante para React, react-bootstrap, permitió una implementación más ágil de los estilos base y componentes como botones, formularios y *layouts*. Esto no solo aceleró el desarrollo, sino que también garantizó que la interfaz tuviera un diseño coherente y estéticamente agradable desde el inicio.

6.1.6. Vercel

Una vez que la aplicación estuvo configurada y probada localmente, se procedió con su despliegue en Vercel siguiendo estos pasos:

- Integración con Vercel: se accedió a la plataforma de Vercel y se inició sesión con una cuenta de GitHub. Esto permitió la integración directa con el repositorio donde estaba alojado el proyecto. En Vercel, se seleccionó la opción *New Project* y se eligió el repositorio de GitHub que contenía la aplicación Vite con React.
- Configuración del proyecto en Vercel: durante el proceso de configuración, Vercel detectó automáticamente que el proyecto estaba basado en Vite y aplicó la configuración predeterminada, lo que incluyó los comandos de construcción (npm run build) y el directorio de salida (dist).
- Despliegue automático: Vercel realizó la construcción de la aplicación utilizando su entorno serverless, generando una versión optimizada de la aplicación React. Una vez completado el proceso de construcción, Vercel desplegó automáticamente la aplicación en su red de distribución global (CDN), haciendo que la aplicación estuviera disponible públicamente a través de una URL proporcionada por Vercel.
- Previsualización de despliegue: para cada commit o pull request en el repositorio de GitHub, Vercel generó una URL de previsualización que permitió revisar los cambios antes de fusionarlos

a la rama principal. Esta característica fue esencial para mantener un flujo de trabajo ágil y colaborativo, asegurando que solo se desplegaran en producción cambios verificados.

6.1.7. Hostinger y Cloudfare

1. Preparación y Requisitos Previos

- Cuenta en Hostinger: necesitamos una cuenta en Hostinger para la creación y gestión del dominio.
- Cuenta en Cloudflare: con nuestra cuenta de Cloudfare nos registramos como primeros usuarios.

2. Agregar el dominio a Cloudflare

- Iniciar el proceso de adición de dominio: en el panel de Cloudflare, selecciona *Add a Site* e ingresa el nombre de nuestro dominio, el que usaremos será "agrointelligence.online".
- Seleccionar un plan: Cloudflare nos ofrece varios planes. empezamos por el plan gratuito, que incluye características esenciales de seguridad y rendimiento.
- Escaneo de DNS: Cloudflare escaneará automáticamente los registros DNS actuales de nuestro dominio.

3. Actualización de los servidores de nombres en Hostinger

- Obtener los Nameservers de Cloudflare: después de confirmar los registros DNS, Cloudflare nos proporciona un par de servidores de nombres (nameservers) personalizados, como "ns1.cloudflare.com" y "ns2.cloudflare.com".
- Iniciar sesión en Hostinger: accedemos a la cuenta de Hostinger y nos movemos a la sección de "Dominios".
- Actualizar los Nameservers: seleccionamos el dominio que quieres gestionar y busca la opción para cambiar los servidores de nombres. Reemplaza los actuales con los que proporciona Cloudflare.
- Guardar los cambios: confirmamos y guardamos los cambios en Hostinger. La actualización de los servidores de nombres puede tardar de unos minutos a unas horas en propagarse completamente.

4. Verificación y configuración final en Cloudflare

- Verificar la configuración: una vez que la propagación se complete, regresamos al panel de Cloudflare y selecciona "Check Nameservers" para confirmar que la actualización ha sido exitosa.
- Configurar SSL y seguridad: en la sección de "SSL/TLS" de Cloudflare, activamos SSL en modo "Flexible" o "Full" para habilitar HTTPS en nuestro sitio web.
- Configurar reglas de seguridad adicionales: ajustamos el *Firewall* y otras reglas de seguridad para proteger tu sitio contra amenazas y optimizar el rendimiento según tus necesidades.

Con esos pasos nos aseguramos de una configuración correcta del dominio que se usará en nuestro panel de control, asegurándonos que tenga desde certificados de seguridad, hasta disponibilidad en todo momento.

6.1.8. ArcGIS

ArcGIS es una plataforma robusta y ampliamente reconocida por su capacidad para crear mapas detallados y realizar análisis geoespaciales avanzados. En el contexto de este proyecto, ArcGIS se evaluó como una solución alternativa para el desarrollo del panel de control, con un enfoque específico en la implementación de mapas interactivos y la visualización de datos geoespaciales relacionados con la producción de caña de azúcar.

Una de las principales ventajas de ArcGIS es su habilidad para manejar grandes volúmenes de datos geográficos con alta precisión, lo que lo convierte en una opción ideal para proyectos que requieren un análisis espacial detallado. En este caso, donde la ubicación de los cultivos y la representación de datos asociados, como el clima y la topografía, son cruciales, ArcGIS ofrecía herramientas avanzadas para la creación de capas de información, análisis de proximidad y la integración de datos satelitales.

Además, ArcGIS se destaca por su capacidad para realizar análisis geoespaciales complejos, lo que permitiría al panel de control proporcionar información adicional valiosa a los usuarios, como la identificación de áreas de riesgo potencial o la optimización de rutas de transporte de la caña. La plataforma también ofrece una interfaz intuitiva para la creación de mapas interactivos, lo que facilita la visualización de datos de manera clara y accesible, mejorando la experiencia del usuario final.

El panel desarrollado en ArcGIS replicaría las funcionalidades clave presentes en la aplicación de React, incluyendo la visualización de datos geoespaciales, la representación de modelos predictivos y de análisis de enfermedades y la interacción con los mapas para obtener información relevante sobre las áreas de cultivo. Una de las principales ventajas de utilizar ArcGIS en este contexto es la integración directa con los sistemas y flujos de trabajo existentes en el Ingenio Pantaleón. Esto permitiría una transición fluida y menos disruptiva para los usuarios ya familiarizados con esta herramienta.

Además, el uso de ArcGIS aprovecharía la infraestructura de datos geoespaciales ya existente en el ingenio, facilitando la actualización y el mantenimiento continuo del panel con los datos más recientes. Las funcionalidades avanzadas de análisis geoespacial de ArcGIS también enriquecerían el contenido del panel, permitiendo realizar análisis de proximidad, evaluar patrones espaciales complejos y generar informes detallados directamente desde la plataforma.

6.2. Aspectos de interacción humano-computador en el diseño del panel de control

6.2.1. Teoría del color y su aplicación

La selección de colores en el diseño del panel de control se fundamentó en la teoría del color y en el análisis de contraste, asegurando que la combinación de colores no solo sea visualmente atractiva, sino también accesible y funcional. En la imagen adjunta, se utilizó el color de texto "#000000"(negro) sobre un fondo de "#FE721B"(naranja), lo que da como resultado un contraste de 7.64, clasificado como "muy bueno"según la herramienta de verificación de contraste utilizada, que fue *Coolors Contrast Checker* [44]. Este alto contraste garantiza que el texto sea legible tanto para pequeños como grandes tamaños, cumpliendo con las pautas de accesibilidad recomendadas para la web. Lo anterior se puede ver en la siguiente figura.

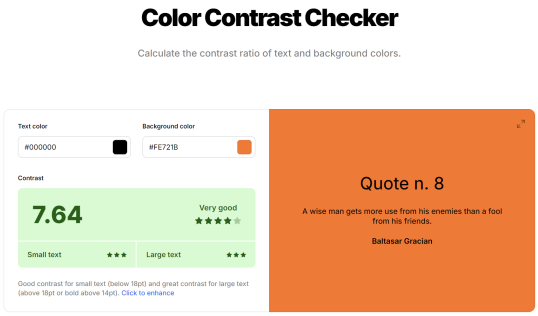


Figura 2. Coolors - teoría del color

Además, la selección de colores del panel de control se inspiró en la identidad visual del Ingenio Pantaleón, cuya página principal utiliza una paleta de colores que incluye tonos similares al "#FE7018"(un tono de naranja) junto con blanco "#FFFFFFz negro "#000000". Este enfoque asegura que el diseño del panel mantenga una coherencia visual con la identidad corporativa del Ingenio, lo que es clave para fortalecer la marca y ofrecer una experiencia de usuario consistente.

La página del panel se basa en tres colores principales:

- "#FE7018"(naranja)
- "#FFFFFF"(blanco)
- "#000000"(negro)

El naranja actúa como el color predominante, atrayendo la atención y destacando elementos interactivos, mientras que el blanco se utiliza como fondo para mantener la claridad y el orden visual. El negro, por su parte, se emplea para el texto principal y elementos destacados, garantizando legibilidad y contraste.

Para los párrafos y textos secundarios, se optó por un tono de gris oscuro "#464646", que proporciona un contraste adecuado sin ser tan dominante como el negro, ayudando a mantener la jerarquía visual y haciendo que la lectura sea más cómoda en bloques de texto más largos.

En resumen, la combinación de colores utilizada en el diseño del panel de control está cuidadosamente seleccionada para lograr un equilibrio entre accesibilidad, coherencia visual con la marca del Ingenio Pantaleón y una experiencia de usuario agradable.

6.2.2. Principios de diseño de interfaces de usuario

En el diseño del panel de control, se aplicaron varios principios fundamentales de diseño de interfaces de usuario para garantizar una experiencia óptima y eficiente para los usuarios del Ingenio. Estos principios guiaron las decisiones de diseño y la implementación de las funcionalidades del panel:

1. Consistencia: se mantuvo una coherencia visual y funcional en todo el panel de control. Esto incluye el uso consistente de la paleta de colores mencionada anteriormente, estilos de botones, tipografía y disposición de elementos. La consistencia ayuda a los usuarios a familiarizarse rápidamente con la interfaz y reduce la curva de aprendizaje.
2. Jerarquía visual: se implementó una clara jerarquía visual para guiar la atención del usuario hacia la información más importante. Esto se logró mediante el uso de diferentes tamaños de

texto, colores de énfasis para elementos clave y una disposición estratégica de los componentes en la pantalla.

3. Retroalimentación: se incorporaron elementos de retroalimentación visual para las acciones del usuario. Por ejemplo, los botones cambian de estado al ser presionados y se muestran indicadores de carga durante procesos que requieren tiempo. Esto ayuda a los usuarios a entender que sus acciones han sido registradas por el sistema.

4. Simplicidad: se adoptó un enfoque minimalista en el diseño, evitando la sobrecarga de información y elementos visuales. Cada página y componente se diseñó con un propósito claro, facilitando la navegación y reduciendo la carga cognitiva del usuario.

5. Accesibilidad: además del contraste de color mencionado anteriormente, se implementaron otras características de accesibilidad, como textos alternativos para imágenes y una estructura de navegación clara para usuarios de lectores de pantalla.

6. Diseño responsivo: el panel se diseñó para ser completamente responsivo, adaptándose a diferentes tamaños de pantalla y dispositivos. Esto asegura una experiencia consistente tanto en computadoras de escritorio como en dispositivos móviles.

7. Agrupación lógica: la información y las funcionalidades relacionadas se agruparon de manera lógica. Por ejemplo, los controles para los diferentes modelos de análisis se ubicaron juntos, facilitando su uso y comprensión.

8. Señalización: se utilizaron elementos de diseño que sugieren cómo interactuar con la interfaz. Por ejemplo, los botones tienen un aspecto presionable y los elementos interactivos cambian de estilo al pasar el cursor sobre ellos.

9. Prevención de errores: se implementaron mecanismos para prevenir errores del usuario, como validaciones en formularios y confirmaciones para acciones irreversibles.

10. Eficiencia: se optimizó la interfaz para minimizar el número de clics o interacciones necesarias para realizar tareas comunes. Esto incluye el uso de atajos y la priorización de las funciones más utilizadas.

11. Flexibilidad y control: se proporcionó a los usuarios flexibilidad en cómo interactúan con los datos, permitiéndoles personalizar vistas, filtrar información y exportar datos según sus necesidades.

La aplicación de estos principios resultó en una interfaz de usuario intuitiva, eficiente y agradable, que permite a los usuarios del Ingenio Pantaleón acceder y analizar la información de producción de caña de azúcar de manera efectiva. El diseño centrado en el usuario no solo mejora la experiencia general, sino que también aumenta la productividad y reduce la probabilidad de errores en la interpretación de datos críticos para la toma de decisiones.

Para evaluar y optimizar el rendimiento técnico y la calidad general del panel de control, se utilizó Google Lighthouse, una herramienta de auditoría automatizada y de código abierto desarrollada por Google. Lighthouse se empleó para analizar varios aspectos críticos de la aplicación web, incluyendo el rendimiento, la accesibilidad, las mejores prácticas y el SEO (Optimización para motores de búsqueda). Esta herramienta proporciona puntuaciones numéricas y recomendaciones detalladas para cada categoría, lo que permitió identificar áreas de mejora y optimizar la experiencia del usuario. La importancia de utilizar Lighthouse radica en su capacidad para ofrecer una visión objetiva y cuantificable de la calidad técnica de la aplicación. Al mejorar las métricas proporcionadas por Lighthouse, se asegura que el panel de control no solo sea funcional y atractivo visualmente, sino también rápido, accesible y optimizado para diferentes dispositivos y conexiones de red. Esto se traduce en una mejor experiencia de usuario, mayor eficiencia en el uso de recursos del sistema y una base sólida para futuras mejoras y mantenimiento del panel de control.

6.2.3. Accesibilidad en el diseño de interfaces

La accesibilidad es un aspecto fundamental en el diseño de interfaces modernas, garantizando que el panel de control sea utilizable por la mayor cantidad de personas posible, independientemente de sus capacidades o discapacidades. En el desarrollo de este proyecto, se prestó especial atención a los siguientes aspectos de accesibilidad:

1. Contraste de color: como se mencionó anteriormente, se utilizó la herramienta Coolors Contrast Checker para asegurar que el contraste entre el texto y el fondo cumpliera con los estándares de accesibilidad WCAG 2.1 [45]. Esto garantiza que el contenido sea legible para personas con diferentes niveles de visión.

2. Estructura semántica: se implementó una estructura HTML semántica, utilizando elementos como <header>, <nav>, <main> y <footer> para proporcionar una jerarquía clara del contenido. Esto beneficia a los usuarios de lectores de pantalla y mejora la navegación general del sitio.

3. Textos alternativos: todas las imágenes e iconos relevantes en el panel de control se acompañaron de textos alternativos descriptivos. Esto permite que los usuarios que dependen de lectores de pantalla comprendan el contenido visual.

4. Etiquetas de formulario: todos los campos de formulario se asociaron correctamente con sus etiquetas, facilitando su uso tanto para usuarios videntes como para aquellos que utilizan tecnologías de asistencia.

5. Tamaños de fuente adaptables: se implementó un sistema de tamaños de fuente que permite a los usuarios ajustar el tamaño del texto sin romper el diseño, mejorando la legibilidad para usuarios con dificultades visuales.

6. Mensajes de error claros: se diseñaron mensajes de error y validación de formularios que son claros, específicos y fácilmente perceptibles, ayudando a todos los usuarios a corregir errores de entrada.

7. Evitar dependencia del color: aunque se utilizó el color para mejorar la experiencia visual, se aseguró que la información crítica no dependiera únicamente del color, utilizando también formas, textos y patrones para transmitir información.

8. Contenido multimedia: para cualquier contenido multimedia incluido en el panel, se proporcionaron alternativas textuales o subtítulos cuando fue apropiado.

La implementación de estas prácticas de accesibilidad no solo cumple con las normativas legales y éticas, sino que también mejora la usabilidad general del panel de control para todos los usuarios. Al diseñar con accesibilidad en mente, se creó una interfaz más inclusiva y fácil de usar, beneficiando a una amplia gama de usuarios, incluyendo aquellos con discapacidades temporales o situacionales.

6.2.4. Usabilidad y experiencia del usuario

La usabilidad y experiencia del usuario (UX) fueron consideraciones clave en cada fase del desarrollo del panel de indicadores. El objetivo principal fue crear una interfaz intuitiva y eficiente, que permita a los usuarios interactuar con el panel de manera fluida y sin obstáculos, optimizando su productividad y satisfacción general.

Minimización del número de clics

Desde el inicio, se prestó especial atención al número de clics necesarios para acceder a las diferentes secciones del panel. La estructura de navegación fue diseñada para que las subsecciones se encuentren a solo uno o dos clics de distancia, lo que reduce el esfuerzo cognitivo y mejora la eficiencia del usuario. Esta estrategia sigue el principio de no hacer pensar (*Don't Make Me Think*) de la guía realizada por Steve Krug, que promueve interfaces claras y directas, facilitando el acceso rápido a la información relevante.

Facilidades de navegación

Para mejorar aún más la navegación, se implementaron menús claros y accesibles, con etiquetas descriptivas y consistentes que permiten al usuario comprender inmediatamente qué contenido se encuentra en cada sección. También se integró un sistema de búsqueda que permite a los usuarios encontrar información específica sin tener que navegar manualmente por cada sección, ahorrando tiempo y esfuerzo.

Tutoriales interactivos

Además, para mejorar la facilidad de uso, se desarrollaron tutoriales interactivos en forma de desplegables que guían al usuario paso a paso a través de las funcionalidades clave del panel. Estos tutoriales no solo explican cómo utilizar los modelos y las herramientas disponibles, sino que también actúan como una especie de introducción asistida para nuevos usuarios, lo que facilita la curva de aprendizaje y asegura que incluso aquellos con menos experiencia técnica puedan utilizar el panel de manera efectiva.

Diseño responsivo

Otro aspecto crítico de la experiencia del usuario fue asegurar que el panel fuera completamente responsivo, adaptándose a una variedad de dispositivos, desde computadoras de escritorio hasta teléfonos móviles. Esto se logró utilizando técnicas de diseño responsivo y marcos de trabajo como Bootstrap, garantizando que la interfaz se mantenga funcional y atractiva en pantallas de diferentes tamaños. De esta manera, los usuarios pueden acceder al panel en cualquier momento y desde cualquier lugar, sin sacrificar la usabilidad.

Linting

La etapa de depuración de código o *Linting* se realiza con cada archivo del proyecto, tanto mientras se desarrolla como una vez conlcuído el proyecto. La depuración de código nos ayuda a mantener el desarrollo lo más limpio posible y cuando hay errores nos muestra en consola los errores y si el código está limpio de igual manera nos dice.

Retroalimentación visual y accesibilidad

Para mejorar la retroalimentación visual, se implementaron indicaciones claras en la interfaz, como animaciones sutiles en los botones y cambios de color cuando se pasa el cursor sobre ellos, lo que proporciona a los usuarios una confirmación visual inmediata de sus acciones. Este enfoque ayuda a evitar errores y mejora la percepción de control por parte del usuario.

6.3. Detalles sobre la elaboración del panel de control y sus componentes

6.3.1. Protocolo de pruebas de usuario y evaluación del prototipo

El desarrollo del panel de control AgroIntelligence incluyó un riguroso protocolo de pruebas de usuario para garantizar que la interfaz y sus componentes cumplieran con las expectativas y necesidades de los usuarios finales. El proceso de evaluación se basó en un enfoque iterativo, donde se realizaron varias rondas de pruebas con usuarios representativos del público objetivo.

Metodología de pruebas de usuario

1. Selección de participantes: se seleccionaron usuarios con diferentes niveles de experiencia en tecnología, incluyendo tanto personal técnico del Ingenio Pantaleón como agricultores y administradores de cultivos. Esta diversidad de usuarios permitió obtener un amplio rango de retroalimentación y asegurarse de que el panel fuera accesible para todos los perfiles.

2. Escenarios de uso: durante las pruebas, se definieron escenarios de uso específicos que los participantes debían completar utilizando el prototipo del panel de control. Estos escenarios incluyeron tareas como la navegación entre secciones (Inicio, API, Modelos, Mapa), la carga y análisis de imágenes y la consulta de datos a través de los diferentes filtros disponibles en el mapa.

3. Recopilación de retroalimentación: se utilizaron herramientas como encuestas y entrevistas semiestructuradas para recopilar el retroalimentación de los usuarios sobre su experiencia con el prototipo. Los participantes proporcionaron información sobre la facilidad de uso, la claridad de la interfaz y cualquier dificultad que enfrentaron durante la realización de las tareas asignadas.

4. Observación directa: además de las encuestas, se llevó a cabo la observación directa de los usuarios mientras interactuaban con el panel. Esta observación permitió identificar problemas de usabilidad que los usuarios podrían no haber mencionado explícitamente, como dificultades para encontrar ciertas funciones o confusión con la terminología utilizada en el panel.

Evaluación del prototipo

El prototipo se evaluó en función de varios criterios clave:

1. Facilidad de navegación: se evaluó la claridad del diseño de navegación y la facilidad con la que los usuarios podían moverse entre las diferentes secciones del panel, como Inicio, API, Modelos y Mapa. La estructura del prototipo permitió un acceso rápido y directo a las funcionalidades principales, lo que se reflejó en el retroalimentación positivo de los usuarios.

2. Eficiencia del flujo de trabajo: el flujo de trabajo dentro del panel, especialmente en tareas como la carga de imágenes y el análisis de enfermedades, fue un punto central de la evaluación. Se aseguró que las tareas pudieran completarse de manera eficiente y que el número de pasos necesarios fuera mínimo.

3. Claridad visual y consistencia: la evaluación también se centró en la claridad visual del diseño. Se revisaron elementos como la disposición de los carruseles, la organización de los modelos y el uso de filtros en el mapa. La consistencia en la presentación de la información y el uso adecuado de la teoría del color (como se describe en la sección anterior) fueron fundamentales para crear una interfaz coherente y fácil de entender.

4. Accesibilidad y retroalimentación visual: el prototipo fue evaluado en términos de accesibilidad, asegurando que todos los elementos de la interfaz fueran claros y utilizables. Se verificó el contraste de colores y la disposición de los elementos para garantizar que cumplieran con las pautas de accesibilidad. La retroalimentación visual, como el resaltado de elementos interactivos y la confirmación de acciones, también se evaluó positivamente.

6.3.2. Home con elementos generales

La sección de Inicio del panel de control es el punto de entrada principal y está diseñada para facilitar la navegación y acceso a las funcionalidades clave. Se organiza en tres secciones principales:

1. Carrusel de contenidos: un carrusel dinámico presenta de manera atractiva las características principales del panel, como el modelo de detección de enfermedades, la predicción del rendimiento de caña y el API. El carrusel utiliza transiciones suaves y controles intuitivos para una navegación fácil.
2. Sección de explicación: ofrece una descripción clara del propósito y funcionalidades del panel, ayudando a los usuarios a comprender cómo utilizar las herramientas disponibles. La información se presenta de manera concisa y accesible.
3. Cuadros de acceso rápido: Tres cuadros dirigen a los usuarios a las herramientas clave del panel:
   - Modelo de detección de enfermedades: para cargar imágenes y analizar enfermedades.
   - Predicción del TCH: para estimar el rendimiento de toneladas caña por hectárea.
   - API: para acceder a la documentación y herramientas de integración.

6.3.3. Modelo de detección de enfermedades

Para esta sección se va a contar con 3 secciones principales el carrusel, la sección para subida de imagen y análisis de la misma y una sección de tutorial para entender el funcionamiento de la pagina La sección modelo de detección de enfermedades del panel de control está diseñada para proporcionar una funcionalidad completa y accesible para la identificación de enfermedades en las plantas de caña de azúcar. Se organiza en tres partes principales:

1. Carrusel: esta área presenta un carrusel informativo que destaca las capacidades del modelo de detección de enfermedades. A través de imágenes y descripciones breves, el carrusel proporciona una visión general de cómo el modelo puede ayudar en la identificación de enfermedades, mostrando ejemplos de resultados y beneficios del análisis.
2. Sección de subida y análisis de Imágenes: la parte central de esta sección permite a los usuarios cargar imágenes de las plantas para su análisis. Aquí, los usuarios pueden subir imágenes y recibir resultados del modelo sobre la presencia de enfermedades. Esta sección está diseñada para ser intuitiva, con una interfaz clara para seleccionar archivos y recibir retroalimentación inmediata sobre las condiciones detectadas.
3. Sección de tutorial: para facilitar la comprensión del uso del modelo, se incluye un tutorial interactivo. Este tutorial guía a los usuarios a través del proceso de carga de imágenes y la interpretación de los resultados. Se ofrece en forma de instrucciones desplegables o mensajes emergentes que explican paso a paso cómo utilizar la herramienta y cómo interpretar los diagnósticos proporcionados.

6.3.4. Modelo predictivo de toneladas de caña por hectárea (TCH)

La sección modelo predictivo de toneladas de caña por hectárea (TCH) está diseñada para proporcionar a los usuarios una visión clara y detallada de las predicciones sobre el rendimiento de la caña de azúcar. Se estructura en dos partes principales:

1. Carrusel general: esta área muestra un carrusel informativo que presenta las principales características y funcionalidades del modelo predictivo. A través del carrusel, los usuarios pueden visualizar de manera dinámica ejemplos de predicciones y obtener una visión general de cómo el modelo estima el rendimiento de la caña por hectárea. El carrusel está diseñado para ser interactivo, permitiendo a los usuarios explorar diferentes aspectos del modelo y entender su aplicación en la gestión de cultivos.

Organizándolo de esta manera se podrá acceder rápidamente a la información relevante y obtener una comprensión profunda de las predicciones del TCH, facilitando una toma de decisiones más informada sobre la gestión de los cultivos.

Mapa interactivo

La sección del mapa interactivo está diseñada para ofrecer a los usuarios una herramienta visual poderosa para explorar y analizar datos geoespaciales relacionados con los cultivos de caña de azúcar. Esta sección se organiza en dos funcionalidades principales:

1. Interacción global con el mapa: los usuarios pueden navegar y explorar el mapa a nivel global, lo que les permite obtener una visión general de la distribución de los cultivos y otras características geoespaciales relevantes. Esta funcionalidad incluye herramientas de zoom y desplazamiento que facilitan la exploración del mapa a diferentes niveles de detalle y en diversas áreas geográficas.
2. Selección de parcelas específicas: además de la vista global, los usuarios tienen la opción de seleccionar parcelas específicas en el mapa. Esta funcionalidad permite a los usuarios hacer clic en parcelas individuales para obtener información detallada sobre cada una, como datos de rendimiento, condiciones del suelo y otros parámetros relevantes. La selección de parcelas específicas facilita el análisis detallado y la toma de decisiones basadas en datos concretos de áreas específicas del cultivo.

Esta estructura del mapa interactivo permite a los usuarios combinar una visión global con detalles específicos, mejorando la capacidad de análisis y la gestión de los cultivos de caña de azúcar.

6.3.5. API

Se va a usar el API REST en varias secciones, la página donde se verán la documentación del API, que va a ser un *embed* del API creada en FastAPI en otro módulo. Y el propio uso de la API en las diferentes secciones de la página, como se detalla a continuación.

La metodología para la integración de API REST en el panel de control React se enfoca en el uso de métodos GET y POST, diseñados para facilitar la interacción y el manejo de datos relacionados con la detección de enfermedades en cultivos. La estructura se organiza en dos funcionalidades principales:

Carga y análisis de imágenes (Método POST)

Esta sección permite a los usuarios cargar imágenes de cultivos para analizarlas y detectar posibles enfermedades. El método POST es fundamental aquí ya que facilita la carga y envío de datos de imagen hacia la API para su procesamiento. La estructura de esta funcionalidad incluye:

1. Carga de imágenes: los usuarios pueden seleccionar y cargar imágenes directamente desde sus dispositivos. Al activar esta opción, el sistema convierte la imagen en un formato adecuado para el análisis y la envía a la API mediante una solicitud POST.
2. Análisis de enfermedades: una vez recibida la imagen, la API procesa la información y devuelve una predicción sobre posibles enfermedades presentes en los cultivos. Esta predicción se actualiza dinámicamente en el panel de control, permitiendo a los usuarios visualizar los resultados de manera inmediata y realizar ajustes basados en los datos obtenidos.

Esta estructura permite a los usuarios realizar un análisis en tiempo real y tomar decisiones informadas sobre el estado de los cultivos, con la ayuda de datos precisos obtenidos a través de la API.

Consulta de resultados de análisis (Método GET)

La sección de consulta de resultados utiliza el método GET para recuperar y presentar los datos históricos de análisis de enfermedades en los cultivos. Este método facilita el acceso a información almacenada en la API, permitiendo una revisión detallada de los resultados. Las funcionalidades de esta sección incluyen:

1. Visualización de datos históricos: a través de solicitudes GET, los usuarios pueden acceder a registros previos de análisis realizados en los cultivos. Estos datos incluyen información detallada sobre el tipo de enfermedad detectada y el nivel de precisión de cada análisis, lo cual proporciona una base sólida para la evaluación de tendencias y patrones.
2. Análisis comparativo de resultados: la sección permite comparar diferentes modelos obtenidos en distintos momentos, ofreciendo una visión general sobre la evolución de enfermedades en los cultivos. Esto se organiza en una serie de tabs que agrupan los datos de manera coherente, facilitando la interpretación y el análisis comparativo.

Con esta estructura, los usuarios pueden explorar tanto los resultados actuales como los datos históricos, optimizando así la capacidad de planificación y respuesta ante problemas sanitarios en sus cultivos.

6.4. Pruebas de usuario

Es importante resaltar que para las pruebas de usuario se hizo uso de una ecuación para sacar tamaños de muestra que toma en cuenta el nivel de confianza que queremos, la proporción de éxito de la población, un margen de error tolerado, la población total y lo que calculamos que es el tamaño de la muestra, un desglose más detallado se presenta a continuación donde hacemos referencia a la ecuación siguiente:

$$n = \frac{Z^2 \times p \times (1-p)}{e^2} \times \frac{1}{1 + \frac{Z^2 \times p \times (1-p)}{e^2 N}} \tag{6.1}$$

Figura 3. Fórmula para el cálculo del tamaño de la muestra

Seguimos el uso de esa ecuación debido a la página de SurveyMonkey que nos detalla el uso de esta ecuación y sus motivos para usarla para calcular tamaños de muestra [46].

- Z: es el valor correspondiente al nivel de confianza. Este valor se relaciona con la probabilidad de que la estimación sea correcta. Para un nivel de confianza del 95 %, el valor de Z es aproximadamente 1.96. Para un nivel de confianza del 99 %, Z es aproximadamente 2.58.
- p: proporción esperada o proporción de éxito en la población. Si no se tiene un valor preciso, se suele utilizar 0.5 (que representa el escenario más conservador, donde la probabilidad de éxito y fracaso es igual).
- e: margen de error tolerado. Es el rango dentro del cual se espera que caiga la estimación. Por ejemplo, si se permite un error del 5 %, se usa e = 0.05.
- N: tamaño de la población total. Es el número de individuos o elementos en la población de interés.
- n: tamaño de la muestra, que es lo que estamos calculando.

6.4.1. Formulario acerca del diseño

El Formulario Acerca del Diseño se utiliza para recolectar opiniones detalladas de los usuarios sobre el diseño visual y funcional del panel de control. Este formulario tiene como objetivo evaluar la efectividad del diseño en términos de estética, usabilidad y satisfacción general del usuario.

- Objetivo: evaluar la percepción del diseño por parte de los usuarios finales y recoger retroalimentación sobre la apariencia visual, la disposición de los elementos y la coherencia con las expectativas y necesidades de los usuarios.
- Contenido: el formulario incluye preguntas específicas sobre varios aspectos del diseño, como:
  - Estética visual: evaluación de la apariencia general del panel, incluyendo el uso de colores, tipografía y gráficos.
  - Organización de contenidos: opiniones sobre la disposición de los elementos en la interfaz, la claridad de la información presentada y la facilidad para encontrar funcionalidades clave.
  - Satisfacción general: preguntas sobre la satisfacción global con el diseño y cualquier aspecto que los usuarios consideren necesario mejorar.
- Método de evaluación: los usuarios completan el formulario después de interactuar con el panel, proporcionando retroalimentación que se analiza para identificar áreas de mejora y ajustar el diseño según las necesidades reales de los usuarios.

6.4.2. Pruebas de navegación

Las pruebas de navegación se centran en evaluar la facilidad con la que los usuarios pueden moverse a través del panel de control y acceder a las diferentes funcionalidades disponibles. El objetivo es asegurar que la estructura de navegación sea intuitiva y eficiente.

- Objetivo: garantizar que los usuarios puedan encontrar y utilizar las diferentes secciones del panel sin dificultad, minimizando el tiempo necesario para realizar tareas y evitar frustraciones.
- Metodología:
  - Escenarios de navegación: los usuarios realizaron tareas específicas, como acceder a secciones particulares del panel, usar herramientas de análisis y generar informes. Se observan los pasos que siguen para completar cada tarea y se registran los tiempos y problemas encontrados.
  - Registro de problemas: se identificaron y documentaron los problemas de navegación, como enlaces rotos, rutas confusas o funcionalidades difíciles de encontrar.
  - Retroalimentación en tiempo real: los usuarios proporcionaron comentarios inmediatos sobre su experiencia de navegación, incluyendo cualquier dificultad que hayan tenido y sugerencias para mejorar la fluidez del proceso.
- Evaluación: se analizan los resultados de las pruebas para determinar la efectividad de la estructura de navegación. Se realizan ajustes basados en el retroalimentación recibido para mejorar la experiencia del usuario, asegurando que la navegación sea lógica y que los usuarios puedan realizar sus tareas de manera eficiente.

Para llevar a cabo estas pruebas se hará uso de la Tecnología que sigue el movimiento de ojos Tobii y se subirán los resultados en Youtube para almacenar los videos.

6.4.3. Pruebas unitarias

Por último, realizaremos pruebas unitarias con la librería Vitest, especializada para hacer pruebas unitarias en entornos de Vite. Para lograr esto debemos seguir una serie de pasos que se detallan a continuación.

- Vamos a configurar el archivo "vite.config.js"para decirle que vamos a usar Vitest para las pruebas unitarias.
- Luego vamos a crear un archivo nuevo llamado "setupTests.js.ªdentro de la carpeta "src"que nos va a ayudar a gestionar las librerías a usar.
- Crearemos también una carpeta de *mocks* para simular los entornos de nuestras secciones en las pruebas.
- Vamos a crear las diferentes pruebas unitarias para probar elementos dentro de nuestra página.

Con este flujo nos aseguramos de probar puntos claves de la aplicación, que nos dirán si fueron exitosas o no las pruebas unitarias.

CAPÍTULO 7

Resultados

7.1. Proceso de diseño e implementación

7.1.1. Prototipado y retroalimentación

Para esta sección se tomaron en cuenta prototipos realizados en la herramienta de Figma. Se realizaron un total de 4 secciones en el prototipo que posteriormente se iban a convertir en las secciones de la aplicación web al momento de programarse.

La primera sección del prototipo abarcaba el Inicio de la aplicación, donde se tenía la vista de un carrusel, una explicación de la página y cuadros que redirigirían a las diferentes secciones del proyecto como se puede observar en la siguiente figura.

Figura 4. Prototipo - página de inicio

Luego se realizó una segunda vista que representaría al modelo de detección de enfermedades que contemplaba de igual manera una sección para el carrusel y posterior a esto la opción para poder subir una imagen para analizarla y saber que enfermedad se le detectó a la planta en ese momento como se puede ver a continuación.

Figura 5. Prototipo - página de detección de enfermedades

Se había contemplado una versión con historial de plantas analizadas, pero esta sección se descartó y el producto final incluye un tutorial sobre el uso del modelo en la página y la sección donde se despliegan los resultados.

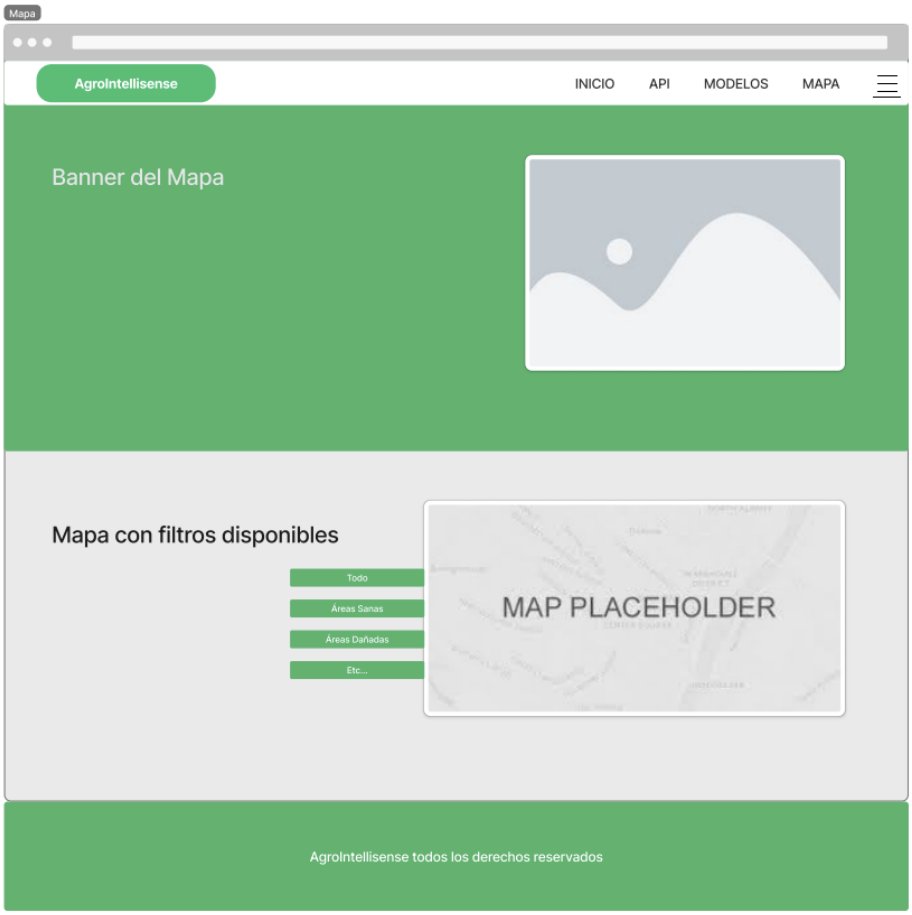


Figura 6. Prototipo - página de predicción de TCH

Como siguiente sección, la figura anterior nos muestra el panel de indicador del mapa donde se evaluaría las toneladas de caña por hectárea (TCH), esta sección en el prototipo mostraba un carrusel como las secciones anteriores y un mapa que ocupaba la mitad de pantalla para los distintos filtros que se le aplicasen. La versión final por retroalimentación obtenida en el formulario indujo a un cambió en esta sección la cual se convirtió en dos pestañas, una para la visualización del mapa y otra para las estadísticas del modelo desplegadas por medio de pestañas dentro de la misma página.

La última sección analizada por el prototipo fue el API, esta mostraba la misma estructura que la anterior con carrusel y abajo de esta una opción para descargar datos y una imagen. Este diseño terminó cambiando el contenido por retroalimentación del usuario. Esta sección la podemos ver reflejada en la siguiente figura.

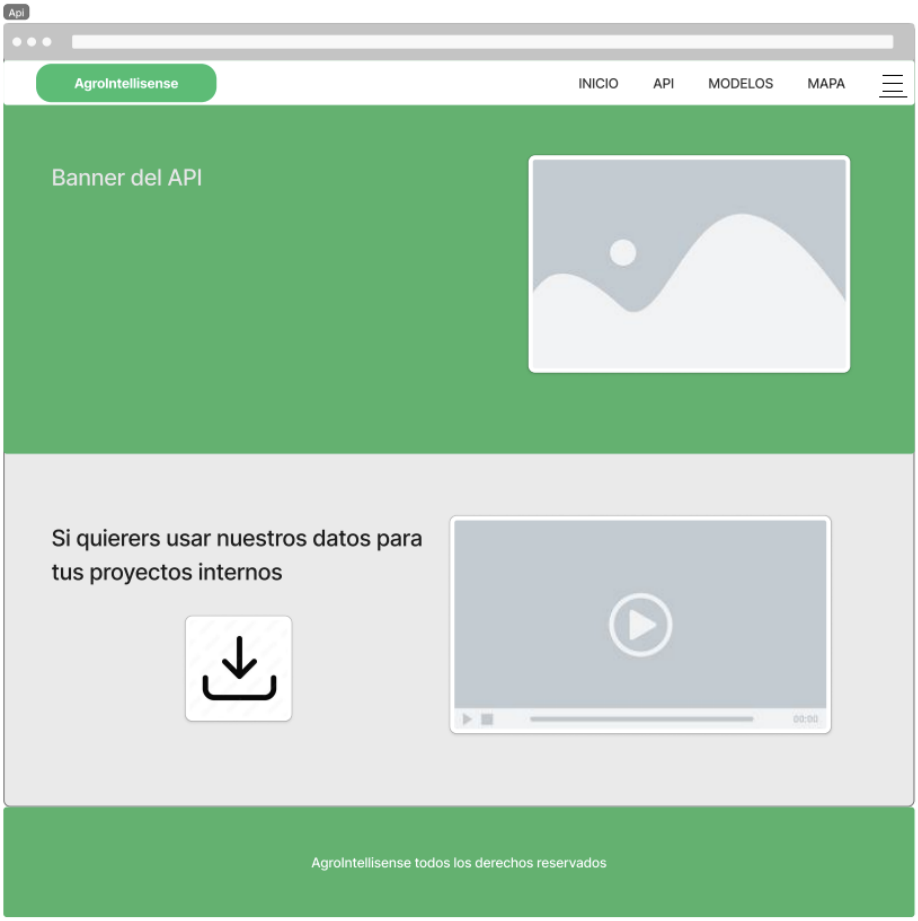


Figura 7. Prototipo - página de API

7.1.2. Selección del tamaño de muestra

Como resultado final del panel de control, se espera un uso aproximado de 50 usuarios. Con esto en mente, hacemos uso de la ecuación 3 mencionada anteriormente en la metodología y se obtiene el resultado que para un tamaño de población de 50 personas, con un nivel de confianza del 95 % con un margen de error del 5 %, el tamaño de la muestra debe ser de 45 personas.

Con el tamaño de la muestra en mente, se decidió dividir a la muestra en dos grupos para las siguientes fases de prueba del prototipado y pruebas de navegación. Por la dificultad que conlleva hacer una prueba de navegación y que se deben realizar de manera presencial se concluyó que 40 personas de la muestra debían hacer las pruebas sobre el prototipo, que se detalla como se hicieron en la siguiente sección y para las pruebas de navegación se busco la prueba de 5 personas de diferentes habilidades técnicas para llevar a cabo la prueba como se detalla en siguientes secciones.

Pruebas de usuario con retroalimentación

El formulario realizado exponía directamente las diferentes secciones que iba a tener el panel de control. El Formulario se puede ver en 53, este tenía preguntas generales sobre el aspecto de las secciones, si se consideraba correcta la distribución de los elementos de la sección y si cambiarían algo en específico. De igual manera podemos ver las respuestas a este formulario en la sección de

anexos con la siguiente referencia 54.

Tomando en cuenta los resultados de la encuesta, se realizaron los siguientes cambios que hacen que el panel de control sea más fácil de entender y más fácil de usar.

La primera sección que recibió cambios por el formulario fue la Figura 4 del inicio, donde se decidió dejar solo los cuadros a las diferentes secciones y se eliminaron las gráficas. Los comentarios recibidos sobre esta sección indicaban que se miraría mejor sin la gráfica de pie en el inicio y dejando solo los cuadros de secciones.

Por parte de la Figura 5 que es la detección de enfermedades de la planta de caña de azúcar, se decidió remover el historial de las imágenes analizadas y se dejó solo la subida de imagen, los resultados del análisis y un tutorial que explica el funcionamiento de la página de acuerdo a los comentarios.

La sección que recibió los cambios más significativos fue la pestaña del modelo predictivo como está en la Figura 6. Esta sección gracias al formulario se decidió dividir en dos nuevas vistas. Una donde estaría el mapa interactivo y la segunda donde estarían las estadísticas del modelo. Se decidió actuar de esta manera para simplificar la navegación del usuario y hacer más fácil el entendimiento. Se dicidió de esta manera ya que la retroalimentación nos indicó que sería bueno tener mas secciones en esta misma.

Por último, de la Figura 7, que es la sección del API, con la retroalimentación obtenida se decidió agregar una sección para la explicación del uso de datos y luego la opción de descarga nada más.

7.1.3. Github para almacenamiento de código

Creación del Repositorio en GitHub

Configuración Inicial del Repositorio:

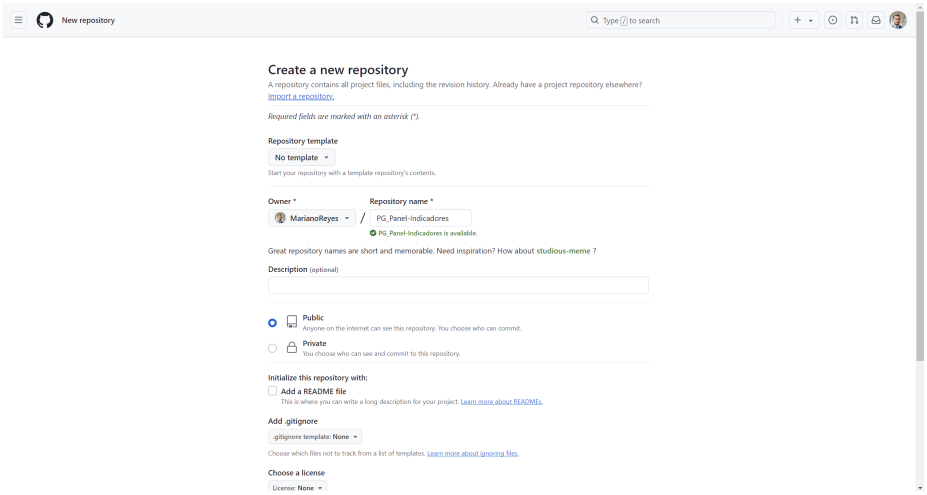


Figura 8. Github - creación de repositorio

En la Figura anterior se muestra el proceso de creación del repositorio PG_Panel_Indicadores. El nombre del repositorio fue definido como PG_Panel_Indicadores y se seleccionó al usuario "MarianoReyes", que es mi usuario de Github, como propietario del repositorio.

Se optó por crear el repositorio como público, lo que significa que cualquier persona puede acceder y visualizar el contenido del repositorio. Esta decisión permite compartir el proyecto con una comunidad más amplia, facilitando la colaboración y la retroalimentación.

No se añadió un archivo README.md.º ".gitignore"durante la creación inicial, lo que me permitió personalizar estos archivos después de que el repositorio haya sido creado.

Después de la creación del repositorio, se añadieron los primeros archivos y se realizaron los primeros çommits", como se observa en el resto de figuras. Se organizaron las carpetas y archivos esenciales para el proyecto, como "src", "publicz los archivos de configuración (vite.config.js, package.json, etc...).

Estado actual del repositorio

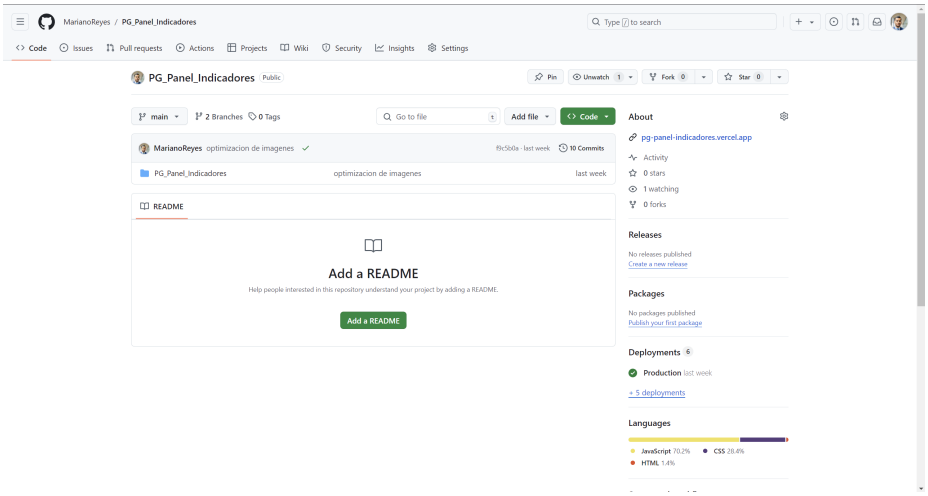


Figura 9. Github - panel de control del repositorio

En la figura anterior se muestra la página principal del repositorio una vez creado. Y la siguiente figura muestra la estructura del directorio del repositorio en GitHub, donde se pueden ver los archivos y carpetas clave del proyecto, como src, public y archivos de configuración como .gitignore y vite.config.js.

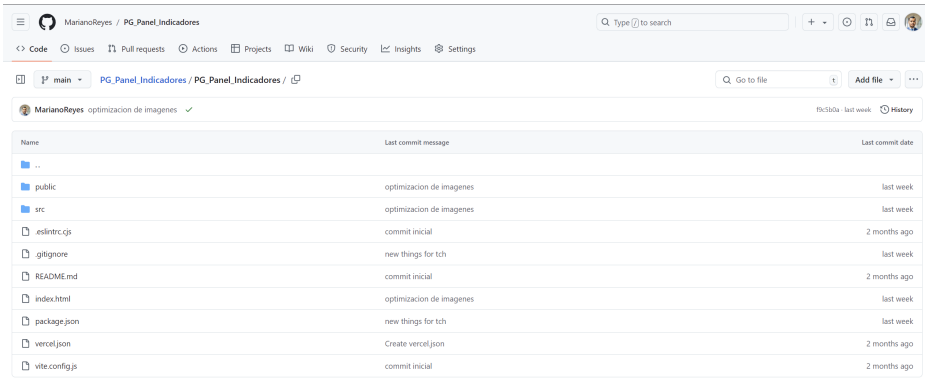


Figura 10. Github - estructura del repositorio

Cada archivo y carpeta tiene asociado un mensaje de çommit"que describe el cambio más reciente realizado, lo que facilita el seguimiento del historial de cambios y la colaboración en el proyecto.

Se observan varios despliegues automáticos, como se muestra en la segunda figura. Esto indica que el repositorio está vinculado con un servicio de despliegue continuo, el cual es Vercel, que despliega automáticamente la última versión del proyecto cada vez que se realiza un çommit.[en] la rama principal.

7.1.4. React para el desarrollo principal de la aplicación

React es el marco de trabajo usado a lo largo del desarrollo del panel de control, React siendo una biblioteca de JavaScript nos ayudó a crear una página web dinámica y reactiva.

La principal fortaleza que nos otorga React es el poder hacer uso de componentes que se pueden sumar para ir armando una aplicación web, tal y como fue el caso de nuestro panel de control. El siguiente es un ejemplo de un componente de React que sirvió como barra de navegación para el panel de control llamado "Header.jsx.

```
import { useState, useEffect } from 'react';
import { Container, Row, Col, Nav, Navbar, NavDropdown, Modal, Button } from 'react-bootstrap';
import './Header.css';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faFacebook, faTwitter, faTiktok } from '@fortawesome/free-brands-svg-icons';
import { faUser, faSignOutAlt } from '@fortawesome/free-solid-svg-icons';

const Header = ({ isAuthenticated, onLogin, onLogout }) => {
  const [isSticky, setIsSticky] = useState(false);
  const [showLogoutModal, setShowLogoutModal] = useState(false);

  const handleScroll = () => {
    setIsSticky(window.scrollY > 50);
  };

  const handleLogout = () => {
    setShowLogoutModal(true);
  };

  const confirmLogout = () => {
    setShowLogoutModal(false);
    onLogout();
  };

  const cancelLogout = () => {
    setShowLogoutModal(false);
  };

  useEffect(() => {
    window.addEventevento de escucha('scroll', handleScroll);
    return () => {
      window.removeEventevento de escucha('scroll', handleScroll);
    };
  }, []);

  return (
    <>
      <header className={`header ${isSticky ? 'sticky' : ''}`}>
        <Navbar expand="lg" className="navbar">
          <Container>
            <Row className="align-items-center w-100">
              {/* Logo y Boton del menu */}
              <Col xs={6} md={3} className="d-flex align-items-center order-md-1">
                <img
```

```
                    src="/Logo-Pantaleon.webp"
                    alt="Brand Image"
                    className={`brand-image ${isSticky ? '' : 'white-filter'}`}
                  />
                </Col>

                <Col xs={6} className="d-flex justify-content-end d-lg-none order-2">
                  <Navbar.Toggle aria-controls="basic-navbar-nav" />
                </Col>

                {/* Redes Sociales en moviles */}
                <Col xs={12} className="d-flex justify-content-center mt-2 mb-2 order-3
                     d-lg-none redes-container">
                  <div className={`redes ${isSticky ? 'redes-sticky' : ''}`}>
                    <a href="#"><FontAwesomeIcon icon={faFacebook} /></a>
                    <a href="#"><FontAwesomeIcon icon={faTwitter} /></a>
                    <a href="#"><FontAwesomeIcon icon={faTiktok} /></a>
                    <a
                      href="#"
                      onClick={(e) => {
                        e.preventDefault();
                        isAuthenticated ? handleLogout() : window.location.href = '/
                            login';
                      }}
                      className="ml-2"
                      title={isAuthenticated ? 'Cerrar Sesion' : 'Iniciar Sesion'}
                    >
                      <FontAwesomeIcon icon={isAuthenticated ? faSignOutAlt : faUser}
                          />
                    </a>
                  </div>
                </Col>

                {/* Menu de Navegacion */}
                <Col xs={12} md={6} className="order-4 order-md-2">
                  <Navbar.Collapse id="basic-navbar-nav">
                    <Nav className="w-100 justify-content-lg-center">
                      <Nav.Link href="/">Inicio</Nav.Link>
                      <NavDropdown title="Modelos" id="basic-nav-dropdown">
                        <NavDropdown.Item href="/deteccion_enfermedades">Deteccion de
                            Enfermedades</NavDropdown.Item>
                        <NavDropdown.Item href="/prediccion_tch">Prediccion TCH</
                            NavDropdown.Item>
                      </NavDropdown>
                      <Nav.Link href="/api">API</Nav.Link>
                      <Nav.Link href="/contact">Contacto</Nav.Link>
                    </Nav>
                  </Navbar.Collapse>
                </Col>

                {/* Redes Sociales en pantallas grandes */}
                <Col md={3} className="d-none d-lg-flex justify-content-end order-md-3
                    redes-container">
                  <div className={`redes ${isSticky ? 'redes-sticky' : ''}`}>
                    <a href="#"><FontAwesomeIcon icon={faFacebook} /></a>
                    <a href="#"><FontAwesomeIcon icon={faTwitter} /></a>
                    <a href="#"><FontAwesomeIcon icon={faTiktok} /></a>
                    <a
                      href="#"
                      onClick={(e) => {
                        e.preventDefault();
                        isAuthenticated ? handleLogout() : window.location.href = '/
                            login';
                      }}
                      className="ml-2"
                      title={isAuthenticated ? 'Cerrar Sesion' : 'Iniciar Sesion'}
                    >
                      <FontAwesomeIcon icon={isAuthenticated ? faSignOutAlt : faUser}
```

```
                        />
                  </a>
                </div>
              </Col>
            </Row>
          </Container>
        </Navbar>
      </header>

      {/* Modal de Confirmacion de Cerrar Sesion */}
      <Modal show={showLogoutModal} onHide={cancelLogout}>
        <Modal.Header closeButton>
          <Modal.Title>Confirmar Cierre de Sesion</Modal.Title>
        </Modal.Header>
        <Modal.Body>Estas seguro de que deseas cerrar sesion?</Modal.Body>
        <Modal.Footer>
          <Button variant="secondary" onClick={cancelLogout}>
            Cancelar
          </Button>
          <Button
            onClick={confirmLogout}
            style={{ backgroundColor: '#fe7018', borderColor: '#fe7018' }}
          >
            Cerrar Sesion
          </Button>
        </Modal.Footer>
      </Modal>
    </>
  );
};

export default Header;
```

Se van a identificar puntos clave de este componente realizado en React.

```
import React, { useState, useEffect } from 'react';
import { Container, Row, Col, Nav, Navbar, NavDropdown } from 'react-bootstrap';
import './Header.css';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faFacebook, faTwitter, faTiktok } from '@fortawesome/free-brands-svg-icons';
```

Esta primera sección es una opcional de los componentes, la sección de importaciones. Esta sección es usada para definir que librerías y estilos vamos a usar en el componente, donde se implementaban las diferentes librerías y estilos dependiendo del componente en cuestión.

```
const Header = ({ isAuthenticated, onLogin, onLogout }) => {

  /* Codigo de JavaScript */

  return (

  /* Componente de HTML */

  );
}

export default Header;
```

Esa estructura es común en todos los componentes, la definición de la función, en este caso el nombre del componente, una sección que nos permite usar código de JavaScript, una sección que nos va a permitir usar la sintaxis de HTML para crear la estructura de la página y para que funcione el componente se debe exportar al final del mismo.

Sumado a esto se le pasan los estados de inició de sesión que nos van a decir si el usuario esta

autorizado o no.

```
const [isSticky, setIsSticky] = useState(false);

const handleScroll = () => {
  if (window.scrollY > 50) {
    setIsSticky(true);
  } else {
    setIsSticky(false);
  }
};

useEffect(() => {
  window.addEventevento de escucha('scroll', handleScroll);
  return () => {
    window.removeEventevento de escucha('scroll', handleScroll);
  };
}, []);
```

Aquí es donde radica la genialidad de React, el poder hacer uso de 'Hooks' como se conocen en el entorno de React que nos van a ayudar a poder gestionar el estado y los efectos secundarios en el componente como se profundizó en el marco teórico.

En este caso usamos para el UseState:

- isSticky: es una variable de estado que indica si la barra de navegación debe ser pegajosa (sticky) al hacer scroll en la página.
- setIsSticky: es la función que se utiliza para actualizar el valor de isSticky.

Por parte del UseEffect:

- handleScroll: es una función que se ejecuta cada vez que el usuario hace scroll en la página. Cambia el estado de isSticky dependiendo de la posición del scroll.
- useEffect: este hook se utiliza para agregar un evento de escucha al evento scroll cuando el componente se monta y para limpiar ese evento de escucha cuando el componente se desmonta. Esto asegura que el evento de scroll se maneje correctamente mientras el componente está en el DOM.

```
const handleLogout = () => {
  setShowLogoutModal(true);
};

const confirmLogout = () => {
  setShowLogoutModal(false);
  onLogout();
};

const cancelLogout = () => {
  setShowLogoutModal(false);
};
```

Son funciones pequeñas que nos ayudan a verificar el estado de sesión y como manejar en caso se dese un cambio.

```
return (
  <>
    <header className={`header ${isSticky ? 'sticky' : ''}`}>
      <Navbar expand="lg" className="navbar">
        <Container>
```

```
            <Row className="align-items-center w-100">
              {/* Logo y Boton del menu */}
              <Col xs={6} md={3} className="d-flex align-items-center order-md-1">
                <img
                  src="/Logo-Pantaleon.webp"
                  alt="Brand Image"
                  className={`brand-image ${isSticky ? '' : 'white-filter'}`}
                />
              </Col>

              <Col xs={6} className="d-flex justify-content-end d-lg-none order-2">
                <Navbar.Toggle aria-controls="basic-navbar-nav" />
              </Col>

              {/* Redes Sociales en moviles */}
              <Col xs={12} className="d-flex justify-content-center mt-2 mb-2 order-3 d-lg-none redes-container">
                <div className={`redes ${isSticky ? 'redes-sticky' : ''}`}>
                  <a href="#"><FontAwesomeIcon icon={faFacebook} /></a>
                  <a href="#"><FontAwesomeIcon icon={faTwitter} /></a>
                  <a href="#"><FontAwesomeIcon icon={faTiktok} /></a>
                  <a
                    href="#"
                    onClick={(e) => {
                      e.preventDefault();
                      isAuthenticated ? handleLogout() : window.location.href = '/login';
                    }}
                    className="ml-2"
                    title={isAuthenticated ? 'Cerrar Sesion' : 'Iniciar Sesion'}
                  >
                    <FontAwesomeIcon icon={isAuthenticated ? faSignOutAlt : faUser} />
                  </a>
                </div>
              </Col>

              {/* Menu de Navegacion */}
              <Col xs={12} md={6} className="order-4 order-md-2">
                <Navbar.Collapse id="basic-navbar-nav">
                  <Nav className="w-100 justify-content-lg-center">
                    <Nav.Link href="/">Inicio</Nav.Link>
                    <NavDropdown title="Modelos" id="basic-nav-dropdown">
                      <NavDropdown.Item href="/deteccion_enfermedades">Deteccion de Enfermedades</NavDropdown.Item>
                      <NavDropdown.Item href="/prediccion_tch">Prediccion TCH</NavDropdown.Item>
                    </NavDropdown>
                    <Nav.Link href="/api">API</Nav.Link>
                    <Nav.Link href="/contact">Contacto</Nav.Link>
                  </Nav>
                </Navbar.Collapse>
              </Col>

              {/* Redes Sociales en pantallas grandes */}
              <Col md={3} className="d-none d-lg-flex justify-content-end order-md-3 redes-container">
                <div className={`redes ${isSticky ? 'redes-sticky' : ''}`}>
                  <a href="#"><FontAwesomeIcon icon={faFacebook} /></a>
                  <a href="#"><FontAwesomeIcon icon={faTwitter} /></a>
                  <a href="#"><FontAwesomeIcon icon={faTiktok} /></a>
                  <a
                    href="#"
                    onClick={(e) => {
                      e.preventDefault();
                      isAuthenticated ? handleLogout() : window.location.href = '/login';
                    }}
```

```
                    className="ml-2"
                    title={isAuthenticated ? 'Cerrar Sesion' : 'Iniciar Sesion'}
                  >
                    <FontAwesomeIcon icon={isAuthenticated ? faSignOutAlt : faUser}
                        />
                  </a>
                </div>
              </Col>
            </Row>
          </Container>
        </Navbar>
      </header>

      {/* Modal de Confirmacion de Cerrar Sesion */}
      <Modal show={showLogoutModal} onHide={cancelLogout}>
        <Modal.Header closeButton>
          <Modal.Title>Confirmar Cierre de Sesion</Modal.Title>
        </Modal.Header>
        <Modal.Body>Estas seguro de que deseas cerrar sesion?</Modal.Body>
        <Modal.Footer>
          <Button variant="secondary" onClick={cancelLogout}>
            Cancelar
          </Button>
          <Button
            onClick={confirmLogout}
            style={{ backgroundColor: '#fe7018', borderColor: '#fe7018' }}
          >
            Cerrar Sesion
          </Button>
        </Modal.Footer>
      </Modal>
    </>
  );
```

En esta sección se trabaja como si fuera una página HTML con la peculiaridad que solo puede haber un contenedor, es decir, un padre que contenga a varios hijos que conformen un componente.

Se pueden usar variables dentro de esta sección también gracias a ser un proyecto de React, que permite este tipo de sintaxis usada en este caso para el manejo de sesión.

Esta estructura de componentes se utilizó a lo largo de todo el panel de control, esto nos ayuda a hacer la aplicación fácil de recorrer para cualquier cambio requerido y muestra la relevancia de React para el desarrollo de aplicaciones web como fue nuestro caso.

7.1.5. Vite para el mantenimiento del estado global de la aplicación

Vite, como fue hablado en el marco teórico, es una herramienta de construcción y servidor de desarrollo que es extremadamente rápida. Nos permitió elegir varias configuraciones iniciales como podemos ver en la siguiente Figura 1.

Vamos a ahondar en las secciones de la creación de un app con Vite para nuestro proyecto.

- npm create vite@latest

Aquí estamos creando una aplicación de Vite, donde le estamos especificando que queremos la última versión del paquete la cual es 5.5.2. Una vez hecho esto se debe otorgarle un nombre al Proyecto, el cual es el mismo que el repositorio para términos de uso. Luego seleccionamos el marco de trabajo de React y por último la variante de Javascript para el proyecto.

- cd pg-panel-indeicadores

Aquí navegamos a la carpeta para poder trabajar en el proyecto y luego le damos a instalar los paquetes del proyecto con el siguiente comando.

- npm install

Una vez creado el proyecto, la estructura básica quedó de la siguiente manera:

```
pg-panel-indeicadores/
---- index.html
---- package.json
---- vite.config.js
---- public/
-------- favicon.ico
---- src/
-------- main.jsx
-------- App.jsx
-------- assets/
```

- index.html: es el punto de entrada principal de la aplicación. Vite inyecta automáticamente los scripts de JavaScript aquí.
- vite.config.js: archivo de configuración donde ajustamos diversas opciones de Vite.
- src/: contiene todos los archivos fuente de la aplicación, incluyendo componentes, estilos, etc.

Por último, se corre el siguiente comando para un ambiente en tiempo real donde se van revisando los cambios en tiempo real de la aplicación.

- npm run dev

Concluyendo, Vite ayudó a tener un ambiente para correr la aplicación de manera rápida y eficiente, ayudando de esta manera al desarrollo del proyecto.

7.1.6. Node JS para un entorno óptimo

Node.js fue crucial en el entorno de desarrollo de React ya que se utilizó para ejecutar la herramienta de Vite (para empaquetar y servir los archivos de la aplicación) y para tener un servidor de React con SSR (Server-Side Rendering). Node.js nos ayudó a servir aplicaciones React con renderizado del lado del servidor. Esto probó ser útil para mejorar el rendimiento y SEO de la aplicación al enviar HTML pre-renderizado al cliente.

Vite se complementó con Node.js para ofrecer un entorno de desarrollo ligero y rápido. Vite usando Node.js fue usado para servir la aplicación durante el desarrollo, compilar el código y realizar tareas como *hot module replacement* (HMR). Vite usa Node.js en el proceso de construcción (build), para transformar y optimizar el código antes de desplegarlo en producción.

Se usaron varias dependencias para el desarrollo del panel de control como se detallan a continuación.

Dependencies

- `@fortawesome/fontawesome-svg-core` ^6.6.0
- `@fortawesome/free-brands-svg-icons` ^6.6.0
- `@fortawesome/free-solid-svg-icons` ^6.6.0
- `@fortawesome/react-fontawesome` ^0.2.2
- `@tensorflow/tfjs` ^4.20.0
- `bootstrap` ^5.3.3
- `leaflet` ^1.9.4
- `react` ^18.3.1
- `react-bootstrap` ^2.10.4
- `react-dom` ^18.3.1
- `react-joyride` ^2.8.2
- `react-leaflet` ^4.2.1
- `react-loader-spinner` ^6.1.6
- `react-router-dom` ^6.25.0

Las dependencias incluyen bibliotecas esenciales como react, react-dom y bootstrap, que son necesarias para construir y renderizar la interfaz de usuario. Estas dependencias son fundamentales para que la aplicación funcione correctamente en producción, asegurando que todos los componentes clave estén disponibles y operativos. Estas bibliotecas son gestionadas a través de Node.js, que permite instalarlas y mantenerlas actualizadas mediante gestores de paquetes como npm o yarn.

DevDependencies

- `@testing-library/react` ^16.0.1
- `@types/react` ^18.3.3
- `@types/react-dom` ^18.3.0
- `@vitejs/plugin-react` ^4.3.1
- `eslint` ^8.57.0
- `eslint-plugin-react` ^7.34.3
- `eslint-plugin-react-hooks` ^4.6.2
- `eslint-plugin-react-refresh` ^0.4.7
- `vitest` ^2.0.5
- `vite` ^5.3.4

Las dependencias de desarrollador contienen herramientas como eslint, vite y @vitejs/plugin-react, que son utilizadas durante el desarrollo del proyecto. Estas herramientas facilitan la compilación, la verificación de la calidad del código y otros procesos de desarrollo que no son necesarios en el entorno de producción. Al igual que las dependencias, las dependencias de desarrollador se manejan a través de Node.js, permitiendo a los desarrolladores trabajar de manera eficiente y asegurar que el código esté bien estructurado y optimizado antes de ser desplegado.

Otra dependencia importante de desarrollador es la librería de Jesy que nos ayudó a hacer pruebas unitarias en nuestro código.

7.1.7. Bootstrap para un diseño responsivo y organizado de la aplicación

Para esta sección se tiene que tener un entendimiento de cómo funciona Bootstrap y cómo se implementó en el panel de control.

Bootstrap es un paquete de diseño front-end que nos proporcionó un conjunto de herramientas y estilos predefinidos para crear interfaces web responsivas y visualmente atractivas. En el contexto de una aplicación React, Bootstrap se utiliza para estructurar el layout, crear componentes UI consistentes y asegurar que la aplicación se vea bien en diferentes tamaños de pantalla.

En el panel de control, Bootstrap se implementó en el componente de Header explicado anteriormente de la siguiente manera:

```
import { Container, Row, Col, Nav, Navbar, NavDropdown } from 'react-bootstrap';
```

Esta línea de código importa componentes específicos de React-Bootstrap, una librería que proporciona componentes de Bootstrap reimplementados para React. Estos componentes permiten utilizar la funcionalidad y el estilo de Bootstrap de una manera que se integra perfectamente con el paradigma de componentes de React. Cabe resaltar que estas importaciones no son todas las que tiene la librería de react'bootstrap, pero si son las usadas en este proyecto mayoritariamente

A continuación, se detalla cómo se utilizan estos componentes y otras características de Bootstrap en el Header:

Estructura del *Header*

El componente utiliza la estructura de cuadrícula de Bootstrap para organizar su contenido:

```
<Container>
  <Row className="w-100">
    <Col xs={6} md={3} className="order-md-1">
      {/* Logo */}
    </Col>
    <Col xs={6} md={3} className="order-md-3">
      {/* Redes sociales */}
    </Col>
    <Col xs={12} md={6} className="order-md-2">
      {/* Navegacion */}
    </Col>
  </Row>
</Container>
```

Esta estructura aprovecha el sistema de rejilla responsiva de Bootstrap, utilizando diferentes tamaños de columna para dispositivos móviles (xs) y medianos (md), así como clases de orden para reorganizar el contenido en diferentes tamaños de pantalla.

Navbar responsivo

El componente utiliza el componente Navbar de react-bootstrap para crear una barra de navegación responsiva:

```
<Navbar expand="lg" className="navbar">
  {/* ... */}
  <Navbar.Toggle aria-controls="basic-navbar-nav" />
  <Navbar.Collapse id="basic-navbar-nav">
    {/* ... */}
  </Navbar.Collapse>
</Navbar>
```

Esto crea una barra de navegación que se colapsa en un menú de hamburguesa en pantallas más pequeñas, una característica típica de Bootstrap.

Menú desplegable

El componente utiliza NavDropdown para crear un menú desplegable:

```
<NavDropdown title="Modelos" id="basic-nav-dropdown">
  <NavDropdown.Item href="/deteccion_enfermedades">Deteccion de Enfermedades</
      NavDropdown.Item>
  <NavDropdown.Item href="/prediccion_tch">Prediccion TCH</NavDropdown.Item>
</NavDropdown>
```

Este es otro componente típico de Bootstrap, adaptado para su uso en React que fue usado para los diferentes enlaces que el menú provee al usuario.

Clases de utilidad

El código también hace uso de las clases de utilidad de Bootstrap para alineación y espaciado en reiteradas ocasiones a lo largo de desarrollo:

- `w-100` para ancho completo
- `d-flex` para crear contenedores flexibles
- `justify-content-end` para alinear contenido al final del contenedor
- `ml-auto` para margen izquierdo automático

Personalización

Es importante recalcar que react-bootstrap es un complemento muy poderoso para cualquier aplicación web y que fue usado de manera exhaustiva a lo largo del desarrollo del panel de control, a pesar de solo haber mostrado un componente que hace uso de esta librería para demostrar su uso. Se debe usar en acompañamiento con el resto de componentes para poder sacarle su máximo provecho que nos permite un ambiente responsivo a lo largo de toda la aplicación.

7.1.8. Vercel para publicar la aplicación web

El despliegue de la aplicación PG_Panel_Indicadores que se desarrolló con Vite y React se realizó utilizando Vercel ya que facilita la implementación y el manejo de aplicaciones web modernas. A continuación, se detallan los resultados de esta implementación, destacando las ventajas de utilizar Vercel en comparación con otras plataformas como Netlify.

Importación del Repositorio desde GitHub

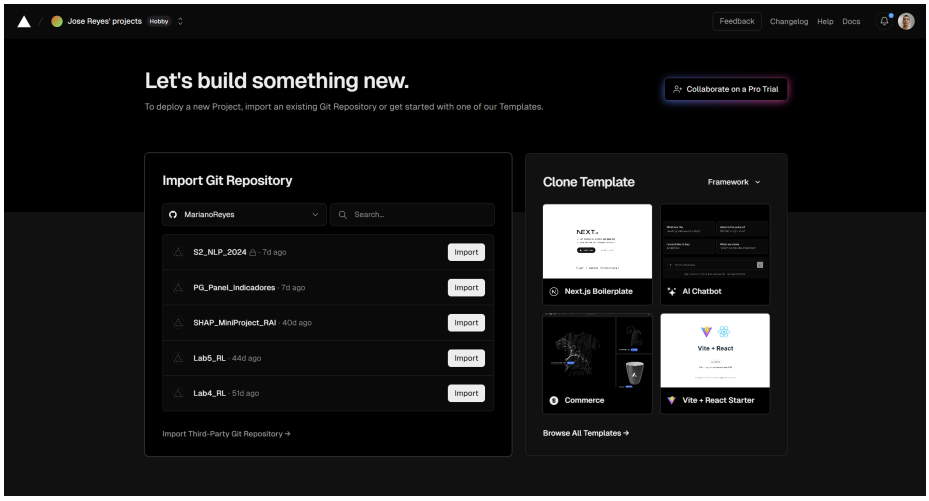


Figura 11. Vercel - pantalla de inicio

En la figura anterior, se muestra la pantalla de inicio de Vercel donde se seleccionó el repositorio PG_Panel_Indicadores desde la cuenta de GitHub asociada a mi usuario (MarianoReyes). Vercel nos ofreció una integración nativa con GitHub, lo que facilitó la importación de repositorios para su despliegue. Esta integración permite que cualquier cambio en el repositorio desencadene automáticamente un nuevo despliegue en Vercel, asegurando de esta manera que la aplicación esté siempre actualizada con la última versión del código.

Configuración del proyecto en Vercel

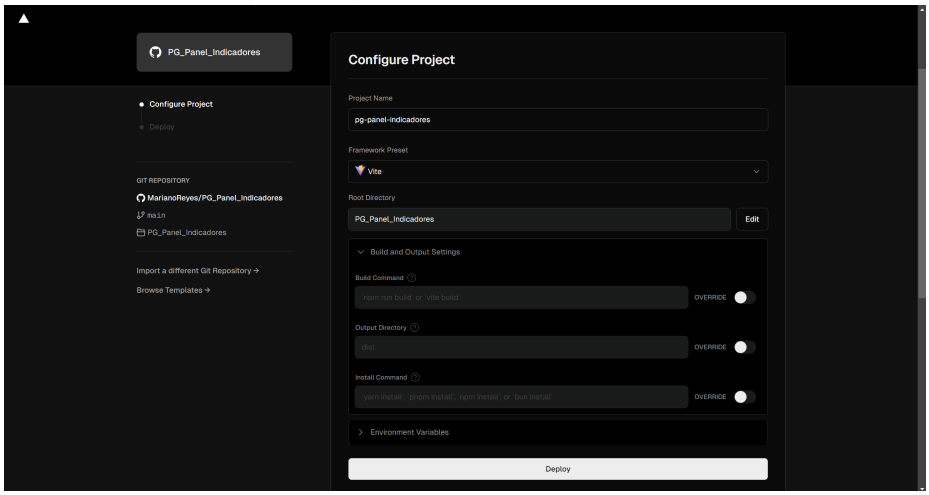


Figura 12. Vercel - pantalla de configuración

Se muestra la pantalla de configuración del proyecto dentro de Vercel. Aquí, se configuró el marco de trabajo como Vite, lo que es detectado automáticamente por Vercel. Los comandos de construcción (build) y las configuraciones de directorio se aplican por defecto, pero se pueden personalizar si es necesario. Este proceso de configuración es extremadamente sencillo y rápido, lo que es una ventaja significativa de Vercel sobre otras plataformas.

Despliegue y gestión del proyecto

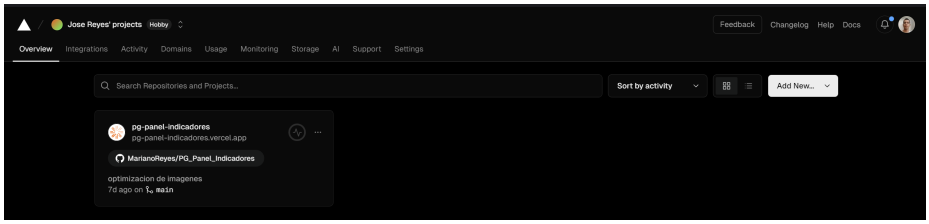


Figura 13. Vercel - aplicación desplegada

Se observa el panel de control de Vercel después de que la aplicación ha sido desplegada. Aquí se pueden ver las actividades recientes, como el despliegue de la optimización de imágenes. Vercel nos proporcionó un entorno centralizado para gestionar todos los aspectos del despliegue, incluso incluye logs de construcción y la gestión de dominios personalizados, todo dentro de una interfaz de usuario intuitiva.

Despliegue en producción

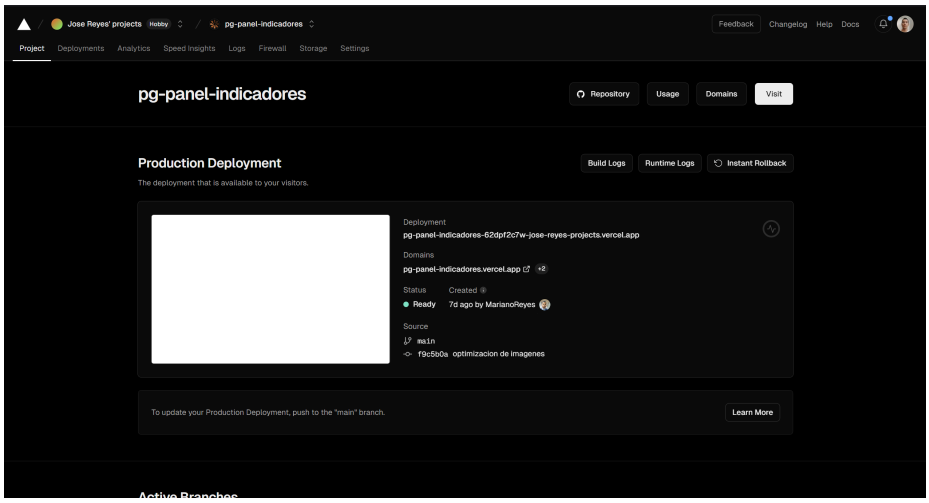


Figura 14. Vercel - producción

La figura anterior muestra la pantalla de despliegue en producción de la aplicación. Vercel asigna automáticamente un dominio (en este caso, pg-panel-indicadores.vercel.app) donde la aplicación está disponible para los usuarios finales.

Además, cualquier cambio en la rama "main"del repositorio de GitHub automáticamente desencadena un nuevo despliegue, asegurando que la versión en producción esté siempre actualizada. El estado de despliegue muestra que la aplicación está lista y que fue desplegada exitosamente por mi usuario "MarianoReyes", con la capacidad de realizar un rollbackïnstantáneo si es necesario.

Un punto fuerte de este tipo de soluciones para desplegar aplicaciones web es que dejan agregar dominios personalizados como se hablará próximamente.

Comparación con Netlify

Aunque Netlify también es una plataforma popular para el despliegue de aplicaciones web, Vercel nos ofreció ciertas ventajas específicas tratándose de proyectos creados con Vite. Vercel detecta automáticamente la configuración de Vite y simplifica el proceso de despliegue, eliminando la necesidad de configuraciones adicionales que pueden ser requeridas en Netlify. Además, la integración con GitHub es más fluida en Vercel, proporcionando un flujo de trabajo más eficiente para la integración continua y despliegue continuo (CI/CD).

7.1.9. Hostinger como proveedor de dominios y Cloudflare para la gestión de dominios

Sobre la creación del dominio, se creo en Hostinger como se mencionó en la metodología. La elección de este proveedor de dominios sobre otros como GoDaddy, es que prácticamente todos los proveedores de dominios ofrecen las mismas herramientas y la elección de uno u otro no va a afectar el resultado, como lo pudo haber sido el caso de elegir Netlify sobre Vercel, discutido anteriormente.

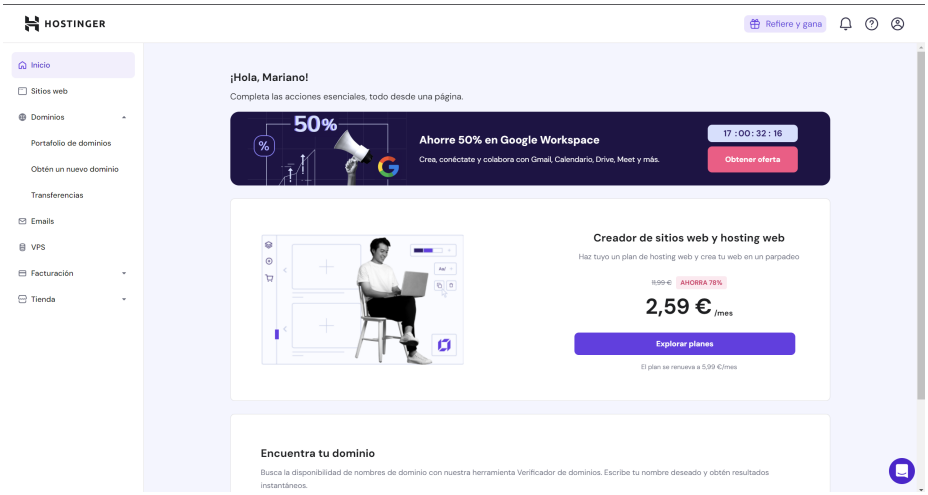


Figura 15. Hostinger - inicio

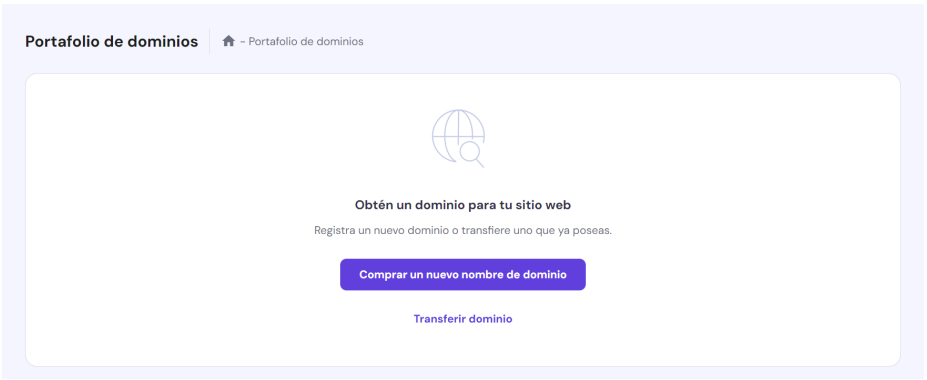


Figura 16. Hostinger - portafolio de dominios

Figura 17. Hostinger - disponibilidad de dominio

Una vez en la página de inicio de Hostinger, Figura 15, se puede acceder a una compra de dominio. Bajo la sección de "Portafolio de Dominios"podemos comprar uno nuevo como se puede ver en la Figura 16, para luego ver la disponibilidad del dominio reflejado en la Figura 17. Donde se seleccionó el dominio .ªgrointelligence.online"para usar a lo largo de la aplicación.

Como punto importante, se debe saber que el dominio no solo es usado para el Panel de Control que se creó, sino que también sirve para otro de los módulos del proyecto, específicamente el API donde se tiene a los diferentes modelos. Con ese contexto es que se usó Cloudfare para el manejo de sitios web (nuestro dominio), para no solo un manejo del Panel de Control, sino también de nuestra API de datos.

Para el manejo de nuestro dominio, asignamos los "Servidores de Nombres.ºtorgados por Cloudfare en Hostinger. Esto para saber que nuestro dominio esta siendo manejado por Cloudfare, esto lo podemos ver en la siguiente figura.

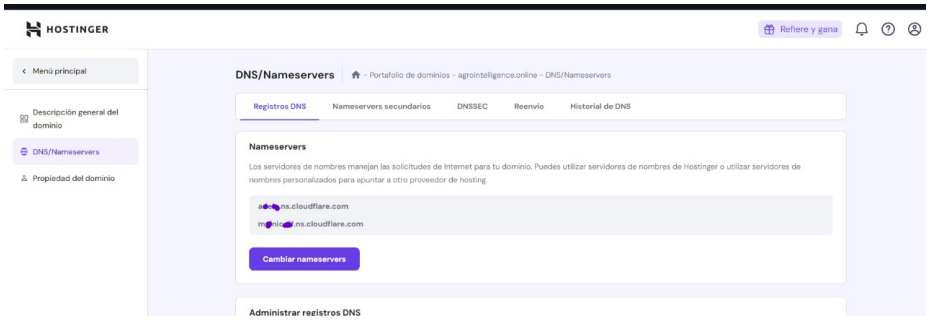


Figura 18. Hostinger - servidores de nombres

Una vez configurado Cloudfare para manejar nuestro sitio, procedimos a conectar nuestro panel de control alojado en Vercel bajo el dominio "pg-panel-indicadores.vercel.app.ª nuestro nuevo dominio. Seleccionamos la opción de añadir dominio donde pondremos "panel.agrointelligence.onlineüna vez lo terminemos de configurar en Cloudfare, para esta configuración se siguieron tres pasos y las siguientes figuras:

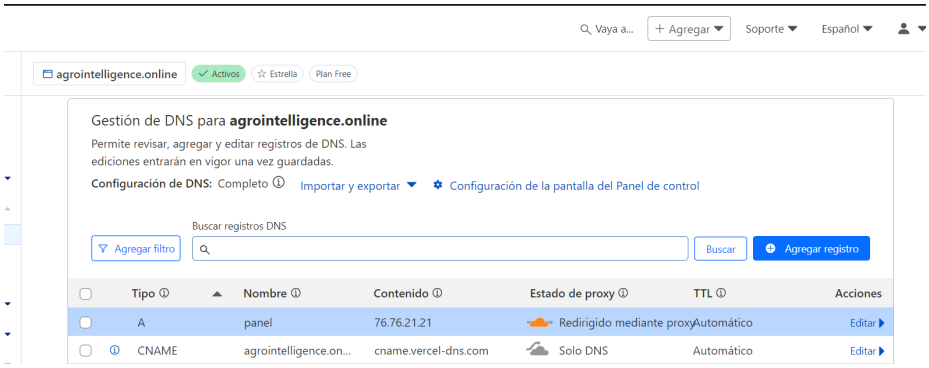


Figura 19. Cloudfare - configuración del registro CNAME

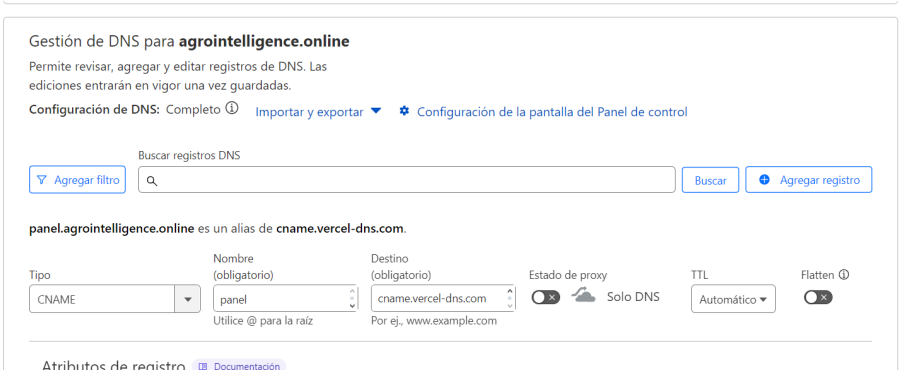


Figura 20. Cloudfare - configuración del registro A

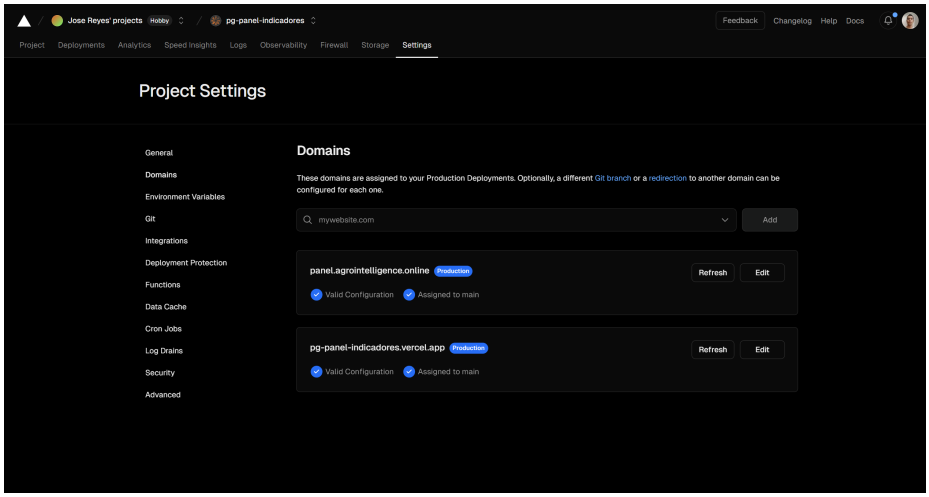


Figura 21. Vercel - configuración del dominio

- Añadimos un registro ÇNAME"para que apunte a los servidores de vercel bajo el enlace çname.vercel-dns.com"proporcionado por vercel, la Figura 19 nos muestra este proceso.
- Luego añadimos un registro .Açomo se puede ver en la Figura 20, para la selección de nuestro subdominio que apunte a la ip de Vercel "76.76.21.21", también dado por Vercel para apuntar a su servidor.
- Por último añadimos el nuevo dominio a nuestra página de vercel, Figura 21, donde gracias a Cloudfare se va a configurar el certificado de seguridad automáticamente para nuestra página.

Con estos pasos nos aseguramos de tener un dominio y un gestor de dominios en la nube, lo cual nos da disponibilidad en todo momento para tener nuestro Panel de Control activo 24/7.

7.1.10. ArcGIS como solución alterna

El desarrollo de un portal web dedicado a las soluciones propuestas en secciones anteriores, accesible en https://arcgismaps.pantaleon.com/portal/apps/sites/#/agrointelligense-uvg, demuestra el potencial de las herramientas de ArcGIS para transformar datos complejos en información accesible

y visualmente impactante. Este sitio web se ha convertido en una plataforma crucial para la presentación de dos modelos fundamentales: uno enfocado en la detección de enfermedades en plantas de caña de azúcar y otro en la predicción de TCH (Toneladas de Caña por Hectárea).

La creación de este portal fue posible gracias a la versatilidad del generador de sitios de ArcGIS, que ofrece una gama de widgets prediseñados. Estos componentes modulares permiten la integración fluida de mapas interactivos, gráficos dinámicos y paneles de control personalizados, adaptados específicamente a las necesidades de visualización de los modelos agrícolas en cuestión.

La interfaz resultante no solo facilita la presentación de datos complejos de manera intuitiva, sino que también permite a los usuarios interactuar con la información de formas significativas. Los agricultores, agrónomos y tomadores de decisiones pueden ahora acceder a predicciones precisas y detectar potenciales brotes de enfermedades con una facilidad sin precedentes, todo desde una única plataforma web.

Con respecto al panel de control resultante de usar ArcGIS como solución alterna podemos ver desde el manejo de sitios, hasta la misma creación de la página por completo. Con respecto al manejo del sitio, podemos ver la siguiente figura.

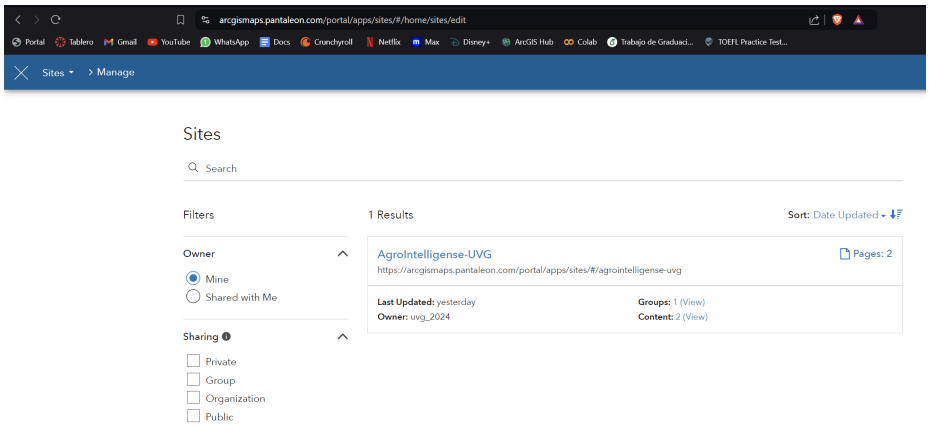


Figura 22. ArcGIS - *enterprise* inicio

Nos muestra un panel de control simple e intuitivo para mostrar los diferentes sitios que tenemos disponibles. El usado para nuestro proyecto es .ªgroIntelligense-UVG.ªparece como única opción debido a que el usuario que se nos otorgó adentro del sistema solo cuenta con ese sitio creado.

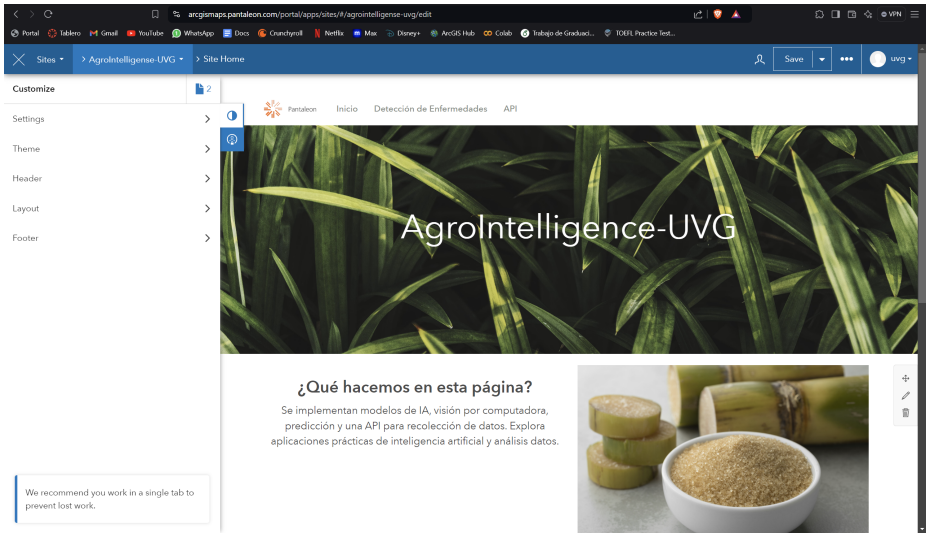


Figura 23. ArcGIS - edición de página 1

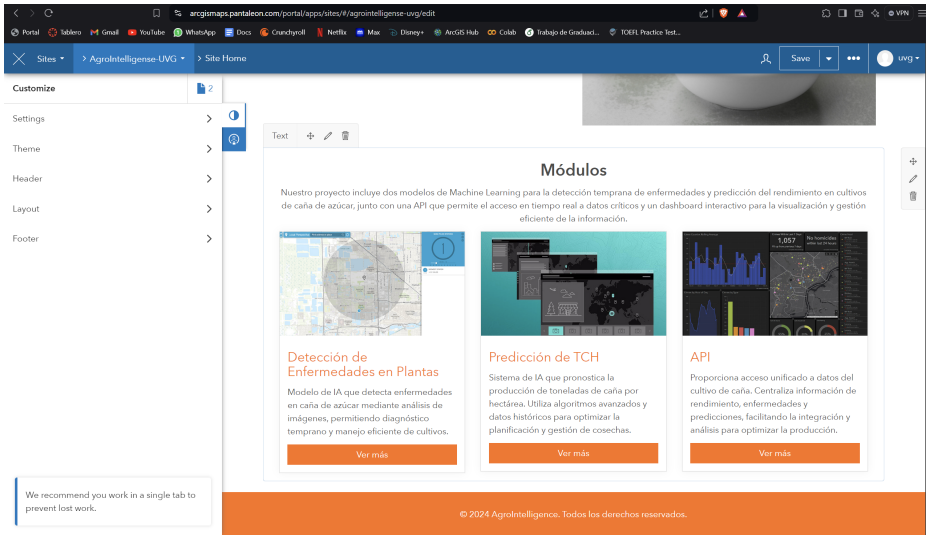


Figura 24. ArcGIS - edición de página 2

Una vez entramos en el sitio se realizó una página de inicio como podemos ver en las Figuras 23 y 24 esta página funciona con el objetivo de ser igual al panel de control realizado con React, el cual imita tanto su funcionalidad como su estilo, permitiéndonos así tener un panel de control realizado con ArcGIS como solución alterna para nuestro panel de control realizado con la tecnología de React.

7.2. Integración de sistemas y pruebas finales

Como fue mencionado anteriormente se trabajaron dos soluciones. Una en React y otra en ARC-Gis, esto para cumplir con la tecnología que se usa en el ingenio por parte de ARCGis y para demostrar como una herramienta como React puede potenciar los resultados obtenidos.

7.2.1. Código final React

Estructura final del proyecto

A continuación se detalla la estructura final del proyecto de la cual se resalta la moduralidad para el manejo de componentes y la construcción de la página.

```
pg-panel-indeicadores/
---- index.html
---- package.json
---- vite.config.js
---- vercel.json
---- README.md
---- .eslint.cjs
---- \_\_tests\_\_
-------- /* Pruebas Unitarias */
---- public/
-------- /* Recursos del Proyecto */
---- src/
-------- Components/
------------ /* Componentes para la construccion de Paginas */
-------- pages/
------------ ApiPage.jsx
------------ DeteccionEnfermedadesPage.jsx
------------ PrediccionTchPage.jsx
------------ HomePage.jsx
------------ /* Estilos de las Paginas */
-------- App.jsx
-------- App.css
-------- Main.jsx
-------- Main.css
```

Main.jsx

```jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'
import 'bootstrap/dist/css/bootstrap.min.css';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

Esta página nos muestra como se inicializa la aplicación de React, como a partir de un elemento Root, que siempre está presente en el index.html de la plantilla se muestra se llama para asignarle el dominio a React desde ese punto.

Posteriormente se llama al componente App, donde va a ir la totalidad del desarrollo para mostrar el Panel de Control.

App.jsx

```jsx
import { useState, useEffect } from 'react';
import { BrowserRouter as Router, Route, Routes, Navigate } from 'react-router-dom';
import Layout from './components/Layout';
import HomePage from './pages/HomePage';
```

```
import ContactPage from './pages/ContactPage';
import ApiPage from './pages/ApiPage';
import DeteccionEnfermedadesPage from './pages/DeteccionEnfermedadesPage';
import PrediccionTchPage from './pages/PrediccionTchPage';
import LoginPage from './pages/LoginPage';

function App() {
  // Inicializa el estado directamente desde localStorage
  const [isAuthenticated, setIsAuthenticated] = useState(() => {
    return localStorage.getItem('isAuthenticated') === 'true';
  });

  useEffect(() => {
    console.log("Estado de autenticacion al cargar App:", isAuthenticated);
  }, [isAuthenticated]);

  const handleLogin = () => {
    setIsAuthenticated(true);
    localStorage.setItem('isAuthenticated', 'true');
    console.log("Inicio sesion, isAuthenticated:", true);
  };

  const handleLogout = () => {
    setIsAuthenticated(false);
    localStorage.removeItem('isAuthenticated');
    console.log("Cerro sesion, isAuthenticated:", false);
  };

  return (
    <Router>
      <Layout isAuthenticated={isAuthenticated} onLogin={handleLogin} onLogout={
          handleLogout}>
        <Routes>
          <Route path="/" element={<HomePage />} />
          <Route path="/contact" element={<ContactPage />} />
          <Route path="/login" element={<LoginPage onLogin={handleLogin} />} />
          <Route
            path="/api"
            element={
              isAuthenticated ? <ApiPage /> : <Navigate to="/login" replace />
            }
          />
          <Route
            path="/deteccion_enfermedades"
            element={
              isAuthenticated ? <DeteccionEnfermedadesPage /> : <Navigate to="/login"
                   replace />
            }
          />
          <Route
            path="/prediccion_tch"
            element={
              isAuthenticated ? <PrediccionTchPage /> : <Navigate to="/login" replace
                   />
            }
          />
        </Routes>
      </Layout>
    </Router>
  );
}

export default App;
```

Lo más importante de esta sección radica en el manejo de rutas para la aplicación. El tráfico de la misma se va a construir desde este punto. Hacemos uso de la librería react-router-dom para poder

tener diferentes direcciones adentro de la misma página.

Para las rutas en si, asignamos diferentes rutas para cada página que tendremos dentro de nuestra aplicación. /.[es] asignado para el Inicio de nuestra aplicación. Luego tenemos /contact"para la página de Contacto. /api.[es] usada para la sección de consumo de datos. /deteccion_enfermedades.[es] usada para el primer modelo de detección de enfermedades. Por último, tenemos /prediccion_tch"que nos sirve para el mapa interactivo con la predicción de las toneladas de caña de azúcar por hectárea.

Por otro lado se hace uso de un componente más para verificar el inicio de sesión en el panel de control. "LoginPage.jsx"nos sirve para este caso del cual se hablará más adelante en resultados, pero como puntos importantes es que restringe el acceso a las diferentes páginas dependiendo si se está autenticado o no.

Otro punto relevante de esta sección es el uso del componente "Layout".

```jsx
import Header from './Header';
import Footer from './Footer';
import './Layout.css';

const Layout = ({ children, isAuthenticated, onLogin, onLogout }) => {
  return (
    <div className="layout-container">
      <Header isAuthenticated={isAuthenticated} onLogin={onLogin} onLogout={onLogout}
          />
      <main className="main-content">{children}</main>
      <Footer />
    </div>
  );
};

export default Layout;
```

Este es usado como el esqueleto de todas las páginas. Tenemos dentro de este componente dos componentes más: *Header* y *Footer* los cuales son usados para crear el menú de navegación de la página y el cierre de la misma.

Se pasan como parámetros las funciones que nos ayudarán a verificar el estado de sesión en la aplicación.

```jsx
import { useState, useEffect } from 'react';
import { Container, Row, Col, Nav, Navbar, NavDropdown } from 'react-bootstrap';
import './Header.css';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faFacebook, faTwitter, faTiktok } from '@fortawesome/free-brands-svg-icons';

const Header = () => {
  const [isSticky, setIsSticky] = useState(false);

  const handleScroll = () => {
    if (window.scrollY > 50) {
      setIsSticky(true);
    } else {
      setIsSticky(false);
    }
  };

  useEffect(() => {
    window.addEventevento de escucha('scroll', handleScroll);
    return () => {
      window.removeEventevento de escucha('scroll', handleScroll);
    };
  }, []);

  return (
```

```
    <header className={`header ${isSticky ? 'sticky' : ''}`}>
      <Navbar expand="lg" className="navbar">
        <Container>
          <Row className="w-100">
            <Col xs={6} md={3} className="order-md-1">
              <img
                src="/Logo-Pantaleon.webp"
                alt="Brand Image"
                className={`brand-image ${isSticky ? '' : 'white-filter'}`}
              />
            </Col>
            <Col xs={6} md={3} className="order-md-3 d-flex justify-content-end">
              <div className={`redes ${isSticky ? 'redes-sticky' : ''}`}>
                <a href="#"><FontAwesomeIcon icon={faFacebook} /></a>
                <a href="#"><FontAwesomeIcon icon={faTwitter} /></a>
                <a href="#"><FontAwesomeIcon icon={faTiktok} /></a>
              </div>
            </Col>
            <Col xs={12} md={6} className="order-md-2 d-flex justify-content-end">
              <Navbar.Toggle aria-controls="basic-navbar-nav" />
              <Navbar.Collapse id="basic-navbar-nav" className="justify-content-end">
                <Nav className="ml-auto">
                  <Nav.Link href="/">Inicio</Nav.Link>
                  <Nav.Link href="/api">API</Nav.Link>
                  <NavDropdown title="Modelos" id="basic-nav-dropdown">
                    <NavDropdown.Item href="/deteccion_enfermedades">Deteccion de
                        Enfermedades</NavDropdown.Item>
                    <NavDropdown.Item href="/prediccion_tch">Prediccion TCH</
                        NavDropdown.Item>
                  </NavDropdown>
                  <Nav.Link href="/contact">Contacto</Nav.Link>
                </Nav>
              </Navbar.Collapse>
            </Col>
          </Row>
        </Container>
      </Navbar>
    </header>
  );
};

export default Header;
```

La explicación de este componente se detallo en secciones anteriores pero demuestra varias herramientas empleadas por React para hacer un diseño limpio y conciso.

```
import { Container } from 'react-bootstrap';
import './Layout.css';

const Footer = () => {
  return (
    <footer className="bg-orange text-black text-center py-3">
      <Container>
        <p className='text-black mb-0 py-4'>&copy; 2024 AgroIntelligence. Todos los
            derechos reservados.</p>
      </Container>
    </footer>
  );
};

export default Footer;
```

El *Footer* es más simple en funcionalidad pero nos ayuda a darle limpieza en general a la página.

HomePage.jsx

```
import Slider from '../components/home/Slider';
import Explanation from '../components/home/Explanation';
import SectionBoxes from '../components/home/SectionBoxes';
import './Main.css';

const HomePage = () => {
  const slides = [
    {
      image: '/slider2.webp',
      title: 'Modelo para la deteccion de enfermedades en la cana',
      description: 'Con datos recogidos en campo, mostramos las enfermedades que
          afectan'
    },
    {
      image: '/slider4.webp',
      title: 'Modelo para predecir el TCH',
      description: 'Con datos historicos, predecimos como seran las plantaciones
          futuras'
    },
    {
      image: '/slider1.webp',
      title: 'API para el consumo de datos',
      description: 'Todos los datos centralizados en una sola solucion'
    }
  ];

  return (
    <div>
      <Slider slides={slides} />
      <Explanation />
      <SectionBoxes />
    </div>
  );
};

export default HomePage;
}
```

La parte principal del proyecto es la página de inicio. Es la más importante en términos de experiencia de usuario ya que va a captar o a perder la atención del mismo en los primeros segundos de la navegación. Para esto evitar lo último, se optó por un estilo simplista a lo largo de la página, que vaya directo al grano y no pierda la atención del usuario.

Se hace uso de la modularidad para dividir de una manera más organizada las secciones de la página. De primero se hace un carrusel con las imagenes y textos que le damos al componente de Slider, este componente se ve de la siguiente manera:

```
import { Carousel } from 'react-bootstrap';
import './Slider.css';

const Slider = ({ slides }) => {
  return (
    <Carousel className="custom-carousel">
      {slides.map((slide, index) => (
        <Carousel.Item key={index}>
          <img
            className="d-block w-100"
            src={slide.image}
            alt={`Slide ${index + 1}`}
          />
          <Carousel.Caption>
            <h3>{slide.title}</h3>
            <p>{slide.description}</p>
          </Carousel.Caption>
```

```
        </Carousel.Item>
      ))}
    </Carousel>
  );
};

export default Slider;
```

Lo más importante a destacar es como hacemos uso del componente externo de *Carousel* proporcionado por react-bootstrap"que nos permite hacer un carrusel de forma simplificada.

Luego con el flujo de la página de inicio, se trabajó una sección de explicación.

```
import { Container, Row, Col} from 'react-bootstrap';
import './Explanation.css'

const Explanation = () => {
  return (
    <Container>
        <Row className='my-5 py-5'>
            <Col className='centered' sm={12} xl={6}>
                <h2>Que hacemos en esta pagina?</h2>
                <p>
                Se implementan modelos de IA, vision por computadora,
                prediccion y una API para recoleccion de datos.
                Explora aplicaciones practicas de inteligencia artificial y analisis
                    de datos.
                </p>
            </Col>
            <Col sm={12} xl={6}>
                <video className='iframe' src="/sugar_cane.mp4" autoPlay muted loop
                    ></video>
            </Col>
        </Row>
    </Container>
  );
};

export default Explanation;
```

El objetivo principal de este componente es el de otorgar una sección que facilite la comprensión de la página al usuario en caso no quede del todo claro que se hace en la página con el carrusel.

Y por último tenemos una sección de cuadros que van a redirigir a cada una de las subpáginas de nuestro proyecto.

```
import { Container, Row, Col } from 'react-bootstrap';
import Box from './Box';
import './SectionBoxes.css';

const SectionBoxes = () => {
  return (
    <div className="section-boxes">
      <Container>
        <Row className='my-5 py-5'>
          <Col sm={12} xl={4}>
            <Box
            title="Deteccion de Enfermedades en Plantas"
            text="Modelo de IA que detecta enfermedades en cana de azucar mediante
            analisis de imagenes, permitiendo diagnostico temprano y manejo eficiente
                de cultivos."
            link="/deteccion_enfermedades"/>
          </Col>
          <Col sm={12} xl={4}>
            <Box
```

```
            title="Prediccion de TCH"
            text="Sistema de IA que pronostica la produccion de
            toneladas de cana por hectarea. Utiliza algoritmos avanzados y datos
            historicos para optimizar la planificacion y gestion de cosechas."
            link="/prediccion_tch"/>
          </Col>
          <Col sm={12} xl={4}>
            <Box
            title="API"
            text="Proporciona acceso unificado a datos del cultivo de cana.
            Centraliza informacion de rendimiento, enfermedades y predicciones,
            facilitando la integracion y analisis para optimizar la produccion."
            link="/api"/>
          </Col>
        </Row>
      </Container>
    </div>
  );
};

export default SectionBoxes;
```

En este componente hacemos uso de un subcomponente que nos ayudó a crear cajas con estilos personalizados para hacer uso de los diferentes elementos del panel de control.

```
import { Card } from 'react-bootstrap';
import './Box.css';

const Box = ({ title, text, link }) => {
  return (
    <Card className="w-100 custom-card">
      <Card.Body>
        <Card.Title className="card-title">{title}</Card.Title>
        <Card.Text className="card-text">{text}</Card.Text>
        <a className='btn' href={link}>Ver mas</a>
      </Card.Body>
    </Card>
  );
};

export default Box;
```

Con algo tan sencillo como el código anterior hacemos un componente reutilizable a lo largo de la página que nos ahorra tiempo a la larga.

En conclusión para la página de *HomePage*, tenemos como el uso de modularidad y reciclaje de código nos permite hacer una sección bastante limpia que podemos ver reflejado en la figura a continuación.

Figura 25. React - página de inicio

LoginPage.jsx

Se implementó una página de inicio de sesión para restringir el uso de las diferentes páginas adentro del panel de control ya que es información que solo se debería desplegar a usuarios autorizados.

```
import { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import { Modal, Button } from 'react-bootstrap';
import './LoginPage.css';
```

```
5
6  function LoginPage({ onLogin }) {
7    const [username, setUsername] = useState('');
8    const [password, setPassword] = useState('');
9    const [showErrorModal, setShowErrorModal] = useState(false);
10   const [errorMessage, setErrorMessage] = useState('');
11   const navigate = useNavigate();
12
13   // Redireccion automatica si el usuario ya estq autenticado
14   useEffect(() => {
15     if (localStorage.getItem('isAuthenticated') === 'true') {
16       navigate('/'); // Redirige a la pagina principal
17     }
18   }, [navigate]);
19
20   const handleLogin = async (e) => {
21     e.preventDefault();
22
23     // Datos de inicio de sesion
24     const loginData = {
25       email: username,
26       password: password
27     };
28
29     try {
30       const response = await fetch('https://api.agrointelligence.online/users/login',
             {
31         method: 'POST',
32         headers: {
33           'Content-Type': 'application/json'
34         },
35         body: JSON.stringify(loginData)
36       });
37
38       if (response.ok) {
39         const data = await response.json();
40
41         localStorage.setItem('token', data.token);
42         localStorage.setItem('isAuthenticated', 'true');
43
44         onLogin(); // Actualiza el estado de autenticacion en App
45         navigate('/'); // Redirige a la pagina principal
46       } else {
47         // Muestra el modal de error si las credenciales son incorrectas
48         setErrorMessage('Credenciales incorrectas. Intentalo de nuevo.');
49         setShowErrorModal(true);
50       }
51     } catch (error) {
52       console.error('Error al iniciar sesion:', error);
53       setErrorMessage('Hubo un error al intentar iniciar sesion. Por favor, intenta
             nuevamente.');
54       setShowErrorModal(true);
55     }
56   };
57
58   const handleCloseErrorModal = () => setShowErrorModal(false);
59
60   return (
61     <div className="login-container">
62       <div className="login-form">
63         <h2>Iniciar Sesion</h2>
64         <form onSubmit={handleLogin}>
65           <div>
66             <label>Correo:</label>
67             <input
68               type="text"
69               value={username}
70               onChange={(e) => setUsername(e.target.value)}
```

```
              required
            />
          </div>
          <div>
            <label>Contrasena:</label>
            <input
              type="password"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              required
            />
          </div>
          <button type="submit">Ingresar</button>
        </form>
      </div>

      {/* Modal de Error */}
      <Modal show={showErrorModal} onHide={handleCloseErrorModal}>
        <Modal.Header closeButton>
          <Modal.Title>Error de Inicio de Sesion</Modal.Title>
        </Modal.Header>
        <Modal.Body>{errorMessage}</Modal.Body>
        <Modal.Footer>
          <Button variant="secondary" onClick={handleCloseErrorModal}>
            Cerrar
          </Button>
        </Modal.Footer>
      </Modal>
    </div>
  );
}

export default LoginPage;
```

El código anterior nos muestra el componente de inicio de sesión. Este refleja el uso nuevamente del API para verificar si el usuario ingresado, esta autentificado o no por la plataforma.

Para hacer esto la principal función de "handleLogin"nos ayuda:

```
  const handleLogin = async (e) => {
    e.preventDefault();

    // Datos de inicio de sesion
    const loginData = {
      email: username,
      password: password
    };

    try {
      const response = await fetch('https://api.agrointelligence.online/users/login',
          {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify(loginData)
      });

      if (response.ok) {
        const data = await response.json();

        localStorage.setItem('token', data.token);
        localStorage.setItem('isAuthenticated', 'true');

        onLogin(); // Actualiza el estado de autenticacion en App
        navigate('/'); // Redirige a la pagina principal
      } else {
```

```
28          // Muestra el modal de error si las credenciales son incorrectas
29          setErrorMessage('Credenciales incorrectas. Intentalo de nuevo.');
30          setShowErrorModal(true);
31        }
32      } catch (error) {
33        console.error('Error al iniciar sesion:', error);
34        setErrorMessage('Hubo un error al intentar iniciar sesion. Por favor, intenta
              nuevamente.');
35        setShowErrorModal(true);
36      }
37    };
```

Con el correo y contraseña del usuario podemos usarlos para verificar en el API si es un usuario correcto, la figura siguiente nos sirve de referencia.

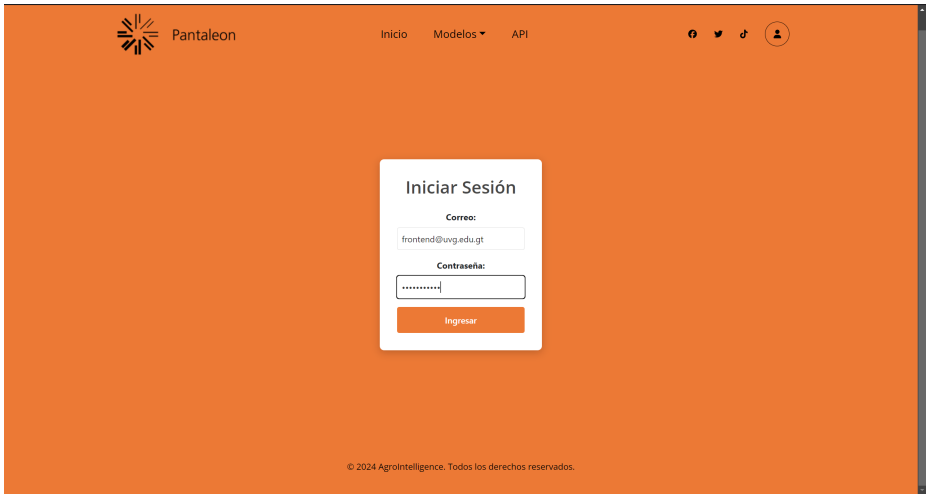


Figura 26. React - página de inicio de sesión

El método es un POST ya que estamos mandando un cuerpo y esperamos una respuesta del servidor.

Si se encuentra respuesta, verificamos si es la que queremos para el inicio de sesión y luego ya redirigimos a la página de inicio. Si no es lo que esperamos, configuramos un mensaje de error y lo mostramos al usuario como se muestra en la figura siguiente.

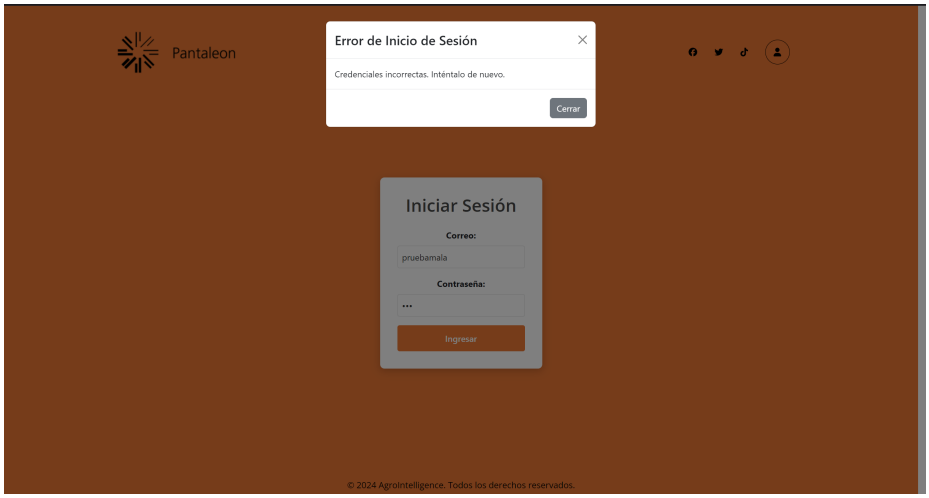


Figura 27. React - página de inicio de sesión con error

DeteccionEnfermedadesPage.jsx

Para la página de detección de enfermedades se realizó una implementación con el API REST, de método POST que aceptaba una imagen de planta de caña de azúcar y devuelve la predicción sobre que enfermedad tiene la planta.

```
import { useState, useEffect } from 'react';
import Joyride, { EVENTS, STATUS } from 'react-joyride';
import { Container, Row, Col, Button } from 'react-bootstrap';
import Slider from '../components/home/Slider';
import './DeteccionEnfermedadesPage.css';

const DeteccionEnfermedadesPage = () => {
  const [prediction, setPrediction] = useState(null);
  const [image, setImage] = useState(null);
  const [runTour, setRunTour] = useState(false);
  const [tourKey, setTourKey] = useState(0);

  useEffect(() => {
    setRunTour(true);
    setTourKey(prevKey => prevKey + 1);
  }, []);

  const slides = [
    {
      image: '/slider2.webp',
      title: 'Modelo para la deteccion de enfermedades en la cana',
      description: 'Con datos recogidos en campo, mostramos las enfermedades que
          afectan'
    }
  ];

  const loadImage = (e) => {
    const file = e.target.files[0];
    const reader = new FileReader();
    reader.onload = () => {
      setImage(reader.result);
    };
    reader.readAsDataURL(file);
  };

  const enfermedadDescriptions = {
    "Chinche salivosa": "...",
```

```
    "Roya naranja": "...",
    "Roya purpura": "...",
    "Clorosis": "...",
    "Hoja Sana": "..."
  };


  const handlePredict = async () => {
    if (!image) {
      alert("Por favor, sube una imagen antes de detectar enfermedades.");
      return;
    }

    try {
      // Convertir base64 a blob para enviar como archivo jpg
      const byteString = atob(image.split(',')[1]);
      const arrayBuffer = new ArrayBuffer(byteString.length);
      const uint8Array = new Uint8Array(arrayBuffer);
      for (let i = 0; i < byteString.length; i++) {
        uint8Array[i] = byteString.charCodeAt(i);
      }
      const blob = new Blob([uint8Array], { type: 'image/jpeg' });

      // Crear un FormData y agregar el archivo con clave 'file'
      const formData = new FormData();
      formData.append('file', blob, 'planta.jpg'); // cambiando la clave a 'file'

      const response = await fetch("https://api.agrointelligence.online/models/
          eagawesome/predict", {
        method: "POST",
        body: formData
      });

      const data = await response.json();
      setPrediction(data.prediction);
    } catch (error) {
      console.error("Error al realizar la prediccion:", error);
    }
  };


  const steps = [
    {
      target: '.upload-image',
      content: 'Aqui puedes subir la imagen de la cana de azucar que deseas analizar.
          ',
    },
    {
      target: '.button-page',
      content: 'Haz clic en este boton para comenzar la deteccion de enfermedades.',
    }
  ];

  const handleJoyrideCallback = (data) => {
    const { type, status } = data;

    if ([STATUS.FINISHED, STATUS.SKIPPED].includes(status)) {
      setRunTour(false);
    }

    if (type === EVENTS.STEP_AFTER || type === EVENTS.TARGET_NOT_FOUND) {
      window.scrollTo({ top: document.body.scrollHeight, behavior: 'smooth' });
    }
  };

  return (
    <div>
      <Joyride
```

```
        steps={steps}
        run={runTour}
        continuous={true}
        showSkipButton={true}
        showProgress={true}
        scrollToFirstStep={true}
        key={tourKey}
        callback={handleJoyrideCallback}
        styles={{
          options: {
            zIndex: 10000,
            primaryColor: '#fe7018',
            textColor: '#333',
            arrowColor: '#fe7018',
            spotlightShadow: '0 0 15px rgba(254, 112, 24, 0.4)'
          },
          buttonClose: {
            color: '#fe7018',
          },
          buttonNext: {
            backgroundColor: '#fe7018',
          },
          buttonBack: {
            marginRight: 10,
            color: '#fe7018',
          }
        }}
      />
      <Slider slides={slides} />
      <Container>
        <Row className="my-5 justify-content-center">
          <Col sm={12} xl={8}>
            <p className="model-description">
              Modelo de IA que detecta enfermedades en cana de azucar mediante
              analisis de imagenes, permitiendo diagnostico temprano y manejo
                  eficiente de cultivos.
            </p>
            <Row className='my-5'>
              <Col sm={12} xl={4}>
                <label htmlFor="upload" className='upload-image button-page-2'>
                  Cargar imagen
                  <input
                    id="upload"
                    type="file"
                    onChange={loadImage}
                    style={{ display: 'none' }}
                  />
                </label>
              </Col>
              <Col sm={12} xl={4} >
                <Button className="button-page" onClick={handlePredict}>Detectar
                    Enfermedad</Button>
              </Col>
              <Col sm={12} xl={4} >
                {image && <img src={image} alt="Uploaded" style={{ maxWidth: '100%',
                    height: '400px' }} />}
              </Col>
            </Row>
            <Row className='my-5'>
              <Col sm={12} xl={12}>
                {prediction && <p>Prediccion: {JSON.stringify(prediction)}</p>}
                <p>{enfermedadDescriptions[prediction]}</p>
              </Col>
            </Row>
          </Col>
        </Row>
      </Container>
    </div>
```

```
  );
};

export default DeteccionEnfermedadesPage;
```

De esta sección podemos resaltar ciertos puntos importantes que nos ayudaron a implementar el modelo en el panel de control, sobretodo la función de "HandlePredict"que se comunica directamente con el API para poder hacer la predicción.

```
  const handlePredict = async () => {
    if (!image) {
      alert("Por favor, sube una imagen antes de detectar enfermedades.");
      return;
    }

    try {
      // Convertir base64 a blob para enviar como archivo jpg
      const byteString = atob(image.split(',')[1]);
      const arrayBuffer = new ArrayBuffer(byteString.length);
      const uint8Array = new Uint8Array(arrayBuffer);
      for (let i = 0; i < byteString.length; i++) {
        uint8Array[i] = byteString.charCodeAt(i);
      }
      const blob = new Blob([uint8Array], { type: 'image/jpeg' });

      // Crear un FormData y agregar el archivo con clave 'file'
      const formData = new FormData();
      formData.append('file', blob, 'planta.jpg'); // cambiando la clave a 'file'

      const response = await fetch("https://api.agrointelligence.online/models/
          eagawesome/predict", {
        method: "POST",
        body: formData
      });

      const data = await response.json();
      setPrediction(data.prediction);
    } catch (error) {
      console.error("Error al realizar la prediccion:", error);
    }
  };
```

En un inicio revisamos la imagen, si hay algo para analizar o no. Una vez eso debemos empaquetar la imagen de base64 a jpg para poder ser usada por el modelo. Guardamos todo en un formulario de data "formData"que le vamos a mandar al API y esperamos la respuesta. Una vez obtenida la respuesta la guardamos en nuestras variables para desplegar los resultados.

Figura 28. React - página de detección de enfermedades

La figura anterior nos demuestra este uso, el cual aparte de la detección tenemos un pequeño tutorial que nos enseña a usar la sección. Subimos una imagen en el botón de subir y presionamos detectar enfermedad para hacer la detección.

Con este flujo simple nos aseguramos que sea fácil para el usuario el detectar enfermedades en sus plantas que quieran analizar.

PrediccionTchPage.jsx

El resultado de esta sección lo podemos ver reflejado en la figura siguiente.

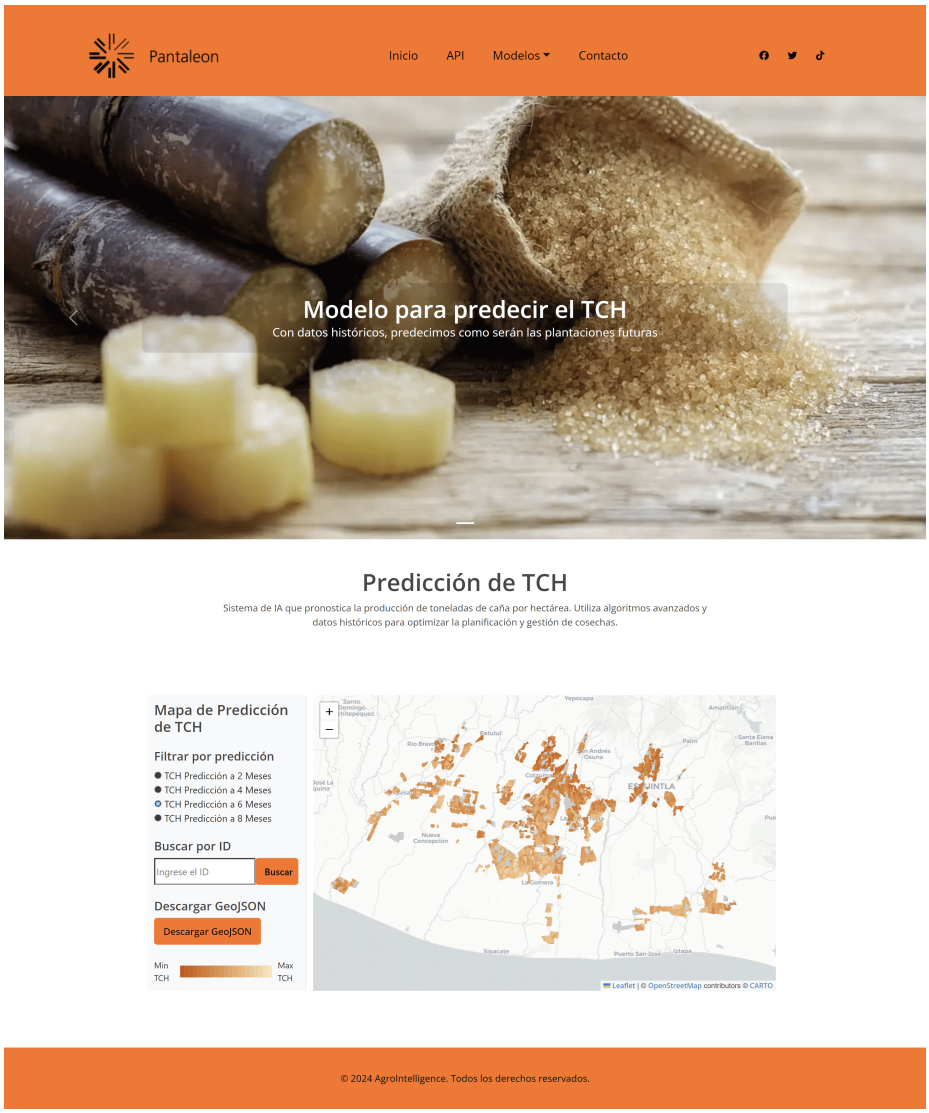


Figura 29. React - Página de Predicción de TCH

Donde podemos ver un carrusel similar al de la página de inicio y en la parte de abajo un mapa interactivo que nos va a ayudar a conocer las predicciones obtenidas de TCH en los diferentes cuadros del terreno del ingenio.

```
import { Container, Row, Col} from 'react-bootstrap';
import Slider from '../components/home/Slider';
import 'leaflet/dist/leaflet.css';
import TchMap from '../components/tch/TchMap';

const PrediccionTchPage = () => {
  const slides = [
    {
      image: '/slider4.webp',
      title: 'Modelo para predecir el TCH',
      description: 'Con datos historicos, predecimos como seran las plantaciones
          futuras',
    },
  ];

  return (
```

```
    <div>
      <Slider slides={slides} />
      <Container>
        <Row className="my-5 justify-content-center">
          <Col className="centered" sm={12} xl={8}>
            <h1>Prediccion de TCH</h1>
            <p>
              Sistema de IA que pronostica la produccion de toneladas de cana por
                  hectarea. Utiliza algoritmos avanzados y datos historicos para
                  optimizar la planificacion y gestion de cosechas.
            </p>
          </Col>
        </Row>
        <Row className="my-5 justify-content-center">
          <Col className="centered" sm={11} xl={10}>
            <TchMap />
          </Col>
        </Row>
      </Container>
    </div>
  );
};

export default PrediccionTchPage;
```

Volvemos a hacer uso de nuestro componente de "Slider"para tener un carrusel personalizado en la subpágina. Luego explicamos un poco de lo que se hace en la sección y por último tenemos nuestro componente de "TchMap"donde ocurre todo el procesamiento de datos y despliegue del mapa, con un menú lateral que nos permite interactuar con el mapa.

El código completo de esta sección lo podemos ver en el Anexo 55, en este podemos resaltar puntos claves para el desarrollo que nos ayudó a crear un mapa interactivo explicando las funciones usadas a lo largo del componente para el funcionamiento del mismo.

- getColorFromTCH(value, minTCH, maxTCH)

Usada para definir un rango de colores para el mapa, desde un naranja con toques de marrón para los terrenos con bajo TCH, hasta un Naranja bastante claro para los terrenos con un TCH mayor. Con "value"nos aseguramos que se tenga un TCH y los otros dos parámetros nos sirven para calcular la paleta de color para los terrenos.

- CenterPolygon(feature)

Con esta función nos aseguramos de centrar el mapa a un terreno en específico dependiendo de la búsqueda realizada por el usuario.

```
useEffect(() => {
    const fetchGeoData = async () => {
      try {
        const response = await fetch('URLLLLLLLLLLLLLLL');
        const data = await response.json();

        // Transformar las coordenadas de cada feature en el GeoJSON
        const transformedData = {
          ...data,
          features: data.features.map((feature) => ({
            ...feature,
            geometry: {
              ...feature.geometry,
              coordinates: convertUTMToLatLng(feature.geometry.coordinates),
```

```
            },
          })),
        };

        // Obtener el valor minimo y maximo del predictor seleccionado
        const tchValues = data.features
          .map((feature) => parseFloat(feature.properties[selectedTCH]))
          .filter((value) => !isNaN(value));

        setMinTCH(Math.min(...tchValues));
        setMaxTCH(Math.max(...tchValues));

        setGeoData(transformedData);
        setLoading(false);

        // Crear un Blob y URL para descargar el GeoJSON transformado, intercambiando
             latitud y longitud
        const swappedData = {
          ...transformedData,
          features: transformedData.features.map((feature) => ({
            ...feature,
            geometry: {
              ...feature.geometry,
              coordinates: swapLatLngInCoordinates(feature.geometry.coordinates),
            },
          })),
        };

        const blob = new Blob([JSON.stringify(swappedData)], {
          type: 'application/json',
        });
        const url = URL.createObjectURL(blob);
        setDownloadUrl(url);
      } catch (error) {
        console.error('Error al cargar los datos GeoJSON:', error);
        setLoading(false);
      }
    };

    fetchGeoData();

    // Limpiar el URL anterior al cambiar el predictor
    return () => {
      if (downloadUrl) {
        URL.revokeObjectURL(downloadUrl);
      }
    };
  }, [selectedTCH]); // Ejecuta nuevamente cuando el predictor cambia
```

El ÜseEffect"de esta sección es lo más relevante del código. Aquí tenemos la lógica para ir cambiando de capas del mapa para mostrar los diferentes valores de TCH a lo largo de los meses. Nos aseguramos de cambiar los valores del mapa para su interacción, obtener sus valores máximo y mínimo, crear un archivo disponible para la descarga y escuchar a cualquier cambio que ocurra.

Es importante recalcar como los datos son extraídos de un URL externo donde se almacenan los resultados del modelo de predicciones en tiempo real.

Luego de eso nos enfocamos de darle estilo a cada una de las parcelas en las funciones:

- onEachFeature(feature, layer)
- style(feature)

Donde le damos el estilo a las parcelas tanto es su estado natural como al pasar el *mouse* por

encima.

- handleSearch()

Por último, esa función sin parámetros nos sirve para manejar la búsqueda de parcelas singulares por medio del id de las mismas, que se implementó en el menú lateral del mapa.

```
<MapContainer
    center={[14.305, -90.785]}
    zoom={9}
    style={{ height: '500px', width: '100%' }}
  >
    {selectedFeature && <CenterPolygon feature={selectedFeature} />}
    <TileLayer
      url="https://{s}.basemaps.cartocdn.com/light_all/{z}/{x}/{y}{r}.png"
      attribution='&copy; <a href="https://www.openstreetmap.org/copyright">
          OpenStreetMap</a> contributors &copy; <a href="https://carto.com/
          attributions">CARTO</a>'
    />
    {geoData && minTCH !== null && maxTCH !== null && (
      <GeoJSON
        data={geoData}
        style={style}
        onEachFeature={onEachFeature}
      />
    )}
</MapContainer>
```

Otro punto importante a recalcar de esta sección es el elemento del mapa como tal, usamos la librería de react-leaflet.[es]especializada en generación de mapas. Usamos uno de sus estilos base en colaboración con Carto que es una base en escala de grises y por encima le colocamos nuestras capas de datos con el GeoJson que obtuvimos anteriormente por medio del url de consumo de datos.

ApiPage.jsx

La sección de API queda bastante simplificada, el siguiente código lo muestra:

```
import { Container, Row, Col, Accordion } from 'react-bootstrap';
import Slider from '../components/home/Slider';
import './ApiPage.css';


const ApiPage = () => {
  const slides = [
    {
      image: '/slider1.webp',
      title: 'API para el consumo de datos',
      description: 'Todos los datos centralizados en una sola solucion'
    }
  ];

  return (
    <div>
      <Slider slides={slides} />
      <Container>
        <Row className="my-5 justify-content-center">
          <Col className="centered" sm={12} xl={8}>
            <p>
              Proporciona acceso unificado a datos del cultivo de cana.
              Centraliza informacion de rendimiento, enfermedades y predicciones,
              facilitando la integracion y analisis para optimizar la produccion.
            </p>
```

```
          </Col>
        </Row>
        <Row className="my-5 justify-content-center">
          <Col sm={12} xl={8}>
            <embed className='w-100 api' src="https://api.agrointelligence.online/
                docs"></embed>
          </Col>
        </Row>
      </Container>
    </div>
  );
};

export default ApiPage;
```

Importamos las librerías utilizadas, luego reutilizamos el componente de *Slider* para colocar nuestro carrusel para la sección, colocamos un párrafo de descripción y por último colocamos un *embed* hacia el API que se usó para la aplicación.

FastApi provee una interfaz gráfica bastante pulida, para un entorno que se integra bastante bien con el resto de la página. Se realizó de esta manera ya que se tiene un mejor entendimiento de que documentos existen en el API así y se facilita su uso en el resto de secciones. El resultado de esta página la podemos ver en la siguiente figura.

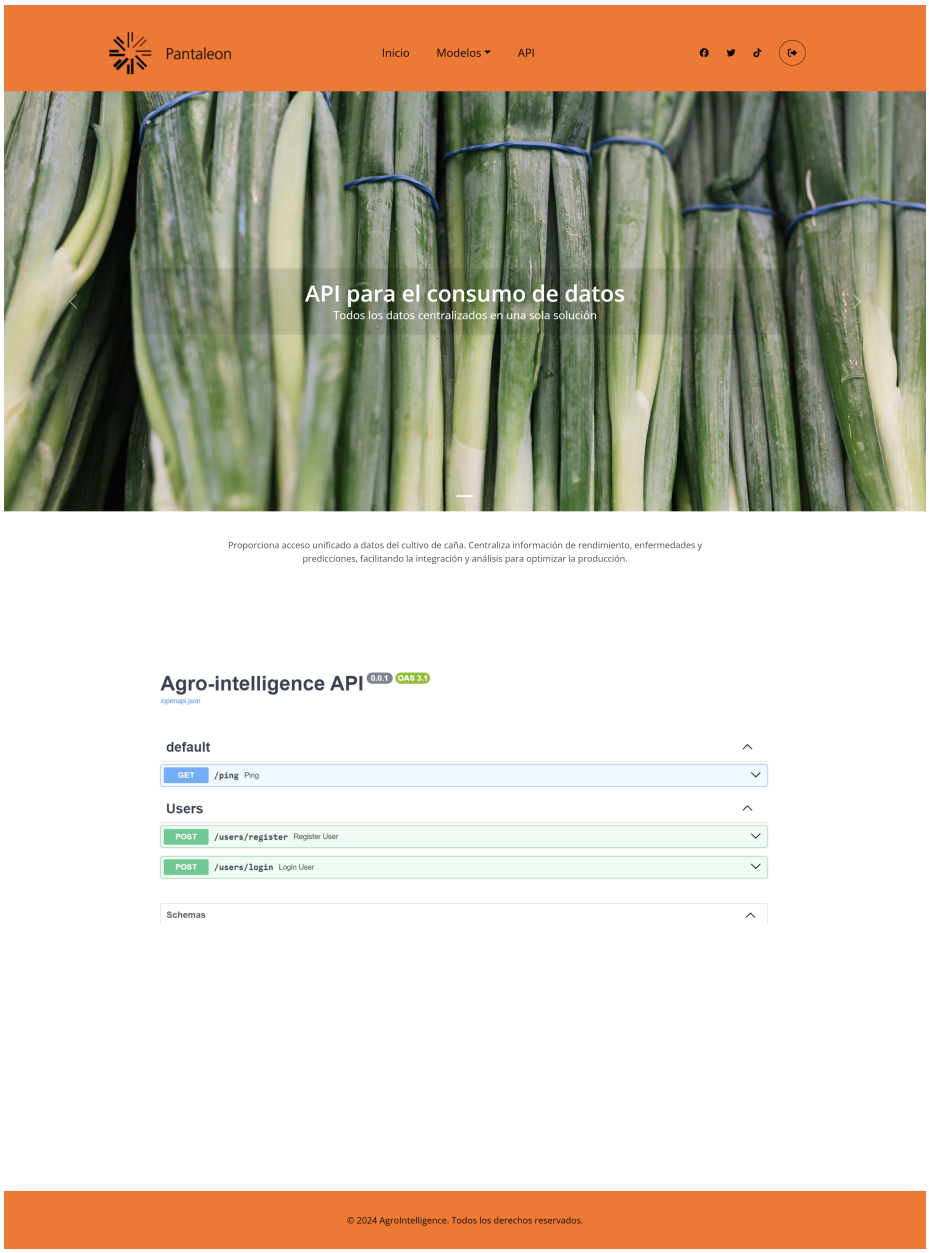


Figura 30. React - página de API REST

Variables de entorno

Como siguiente punto acerca del código en React, para mantener las buenas prácticas, se desarrolló un archivo ".env"que tiene nuestras variables de entorno definidas.

- VITE_URL_LOGIN=https://api.agrointelligence.online/users/login
- VITE_URL_DETECCION_ENFERMEDADES=https: //api.agrointelligence.online/ml/plague-net/predict
- VITE_URL_PREDICCION_TCH=https: //raw.githubusercontent.com/Jack200133 /tcha-pi/master/data/outputgisv6.geojson

Esto con el objetivo de dejar un acceso fácil a los urls de los modelos y que sean fácil de modificar de ser necesario.

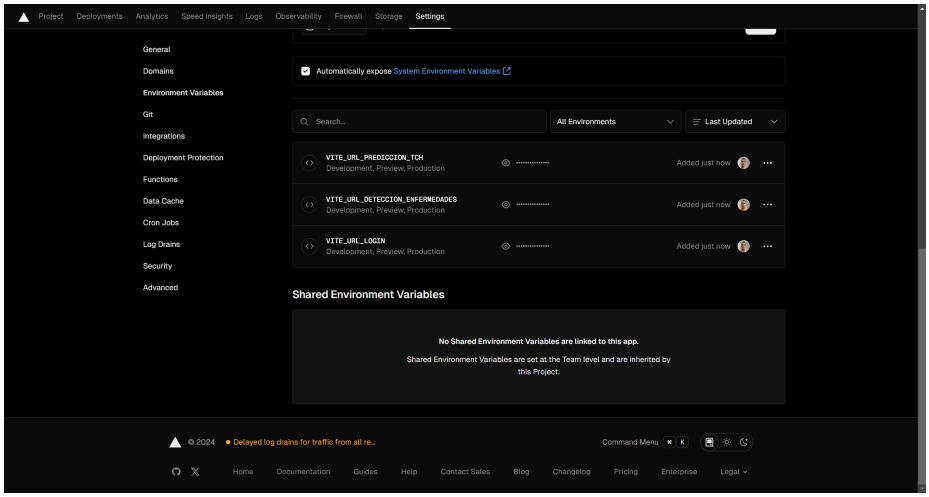


Figura 31. Variables de entorno en vercel

Luego, como se puede ver en la figura anterior, para que Vercel pueda usar esas variables de entorno se deben configurar en el panel de control de Vercel y de esa manera quedan disponibles para el proyecto. De esa manera dividimos variables de entorno de producción y de desarrollo.

Diseño responsivo

Para asegurarnos de tener una aplicación responsiva se uso Boostrap como marco de trabajo responsivo. Con este marco nos aseguramos que la vista de la aplicación se mantenga tanto en un despliegue de pantalla amplio, como Laptop por ejemplo. Y también nos sirve para desplegar la aplicación desde el teléfono, simulando ser una aplicación nativa para celular de igual manera.

Figura 32. Boostrap - diseño responsivo 1

Figura 33. Boostrap - diseño responsivo 2

Las figuras anteriores nos muestran el inicio de la página en modo responsivo, lo cual nos muestra un despliegue diferente a como se muestra en pantalla grande como se ve en la Figura 25.

Para asegurarnos de un despliegue adecuado en pantallas más pequeñas se hizo uso de las siguientes clases:

- order
- order-md-1

Entre estas y otras clases explicadas anteriormente nos aseguramos de tener una aplicación responsiva en todo momento, incluso en el despliegue de cuadros dentro de la aplicación como se muestra en la figura siguiente.

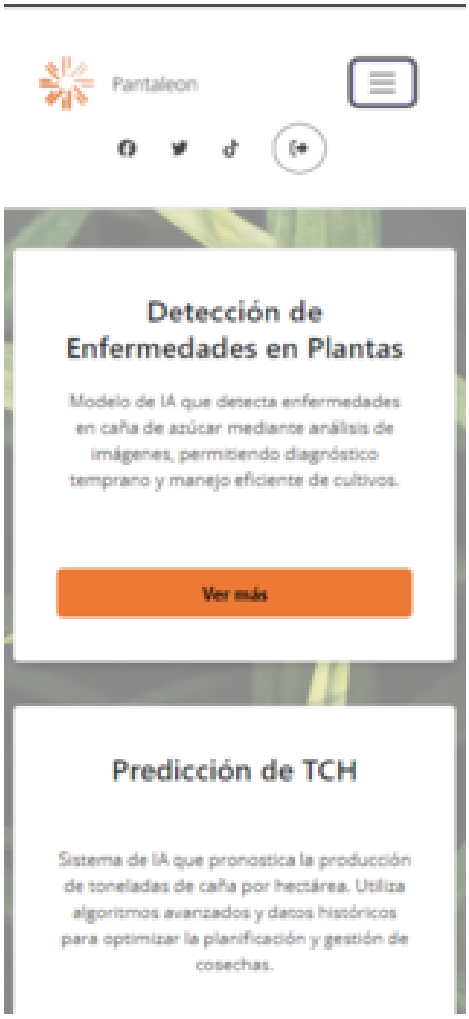


Figura 34. Boostrap - diseño responsivo 3

Optimización de página

Se usó la herramienta de LightHouse proporcionada por Google para el reporte final de la aplicación en términos de rendimiento, accesibilidad, mejores prácticas y SEO, la figura siguiente nos lo muestra más a detalle.

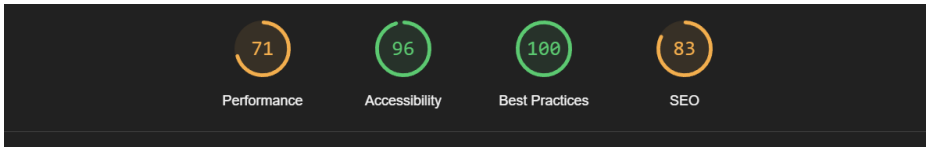


Figura 35. Lighthouse - sección principal

Figura 36. Lighthouse - desempeño general

La primera sección, la figura anterior, nos muestra un desempeño general de la página por arriba del 70 %, lo cual se encuentra dentro del rango esperado debido a la carga alta de datos para la sección de predicción de TCH. De igual manera este puntaje se puede mejorar si optimizamos aún más las imágenes usadas en el panel de control ya que se puede ahorrar espacio con imágenes más pequeñas. Cabe resaltar que se hizo varias optimizaciones en las mismas con herramientas para bajar peso de imagen, convertir a un formato más óptimo para web como ".webp.[ei]ncluso volver a definir el tamaño de las imágenes.

Como siguiente punto, la accesibilidad de la página nos retorno un 96 % de puntaje lo cual es prácticamente perfecto con pequeños detalles a mejorar en la redirección de páginas, la siguiente figura nos muestra estos resultados.

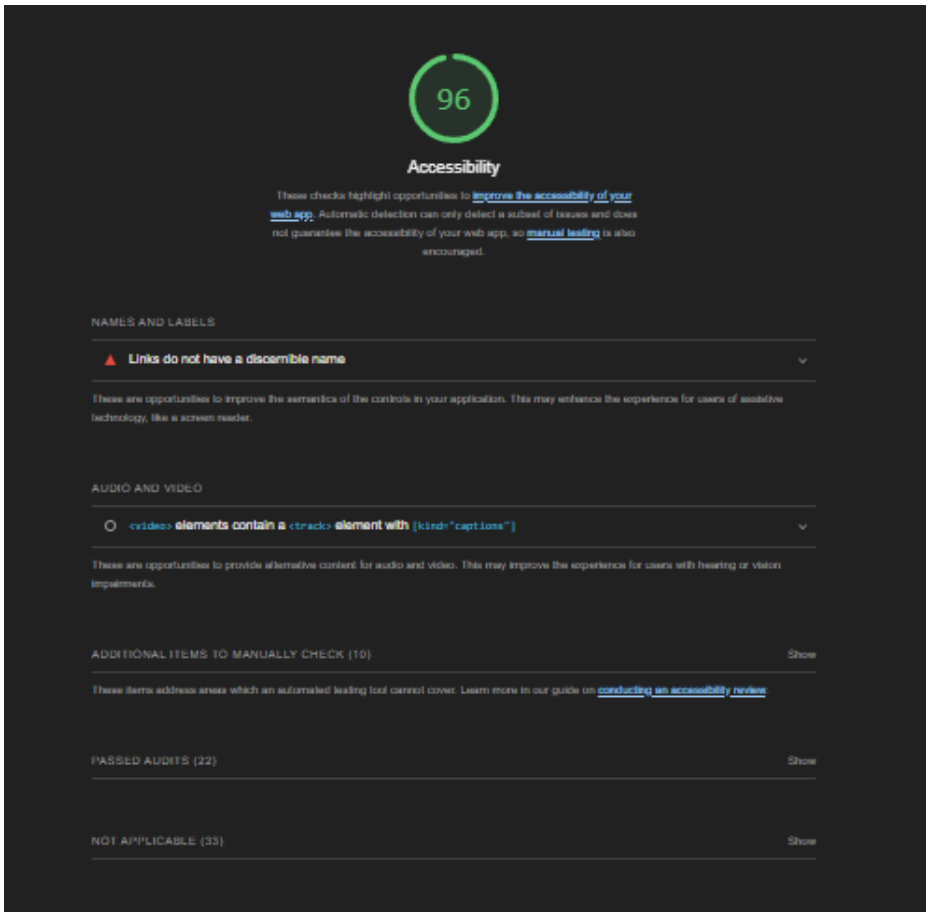


Figura 37. Lighthouse - accesibilidad

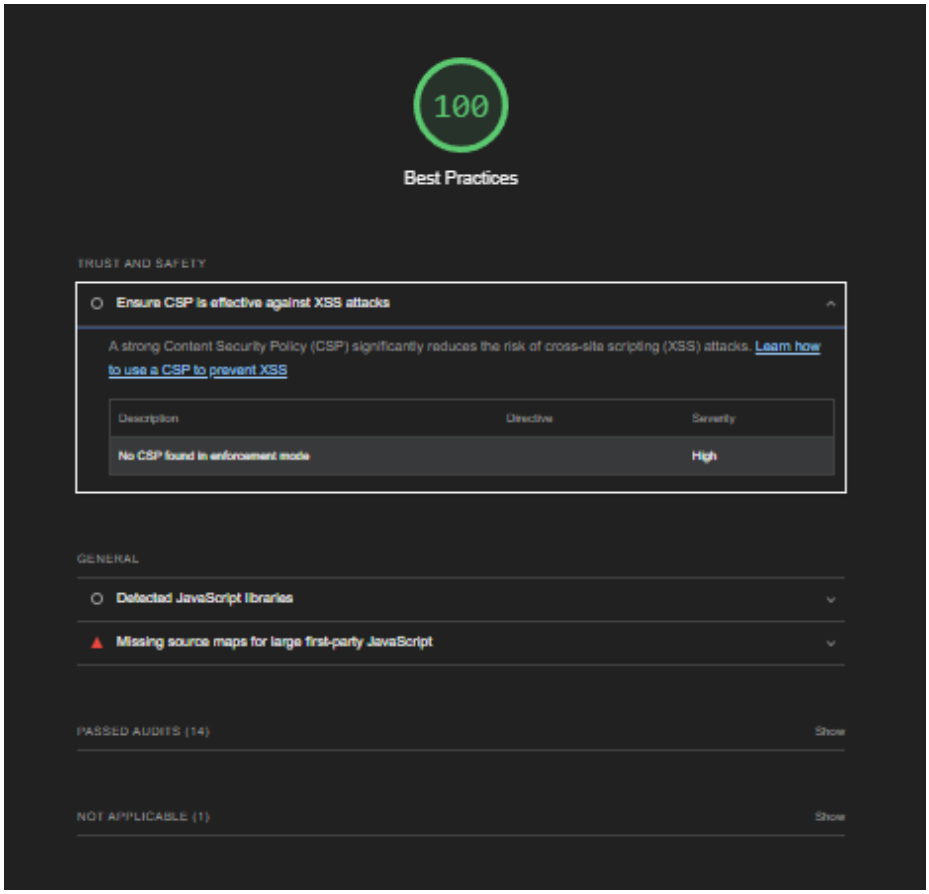


Figura 38. Lighthouse - mejores prácticas

En el ámbito de las mejores prácticas, la Figura 38 nos devuelve un puntaje perfecto de 100 %, esto principalmente debido a que React y sobre todo Vite son marcos de trabajo altamente optimizados para funcionar de una manera adecuada en navegadores web.

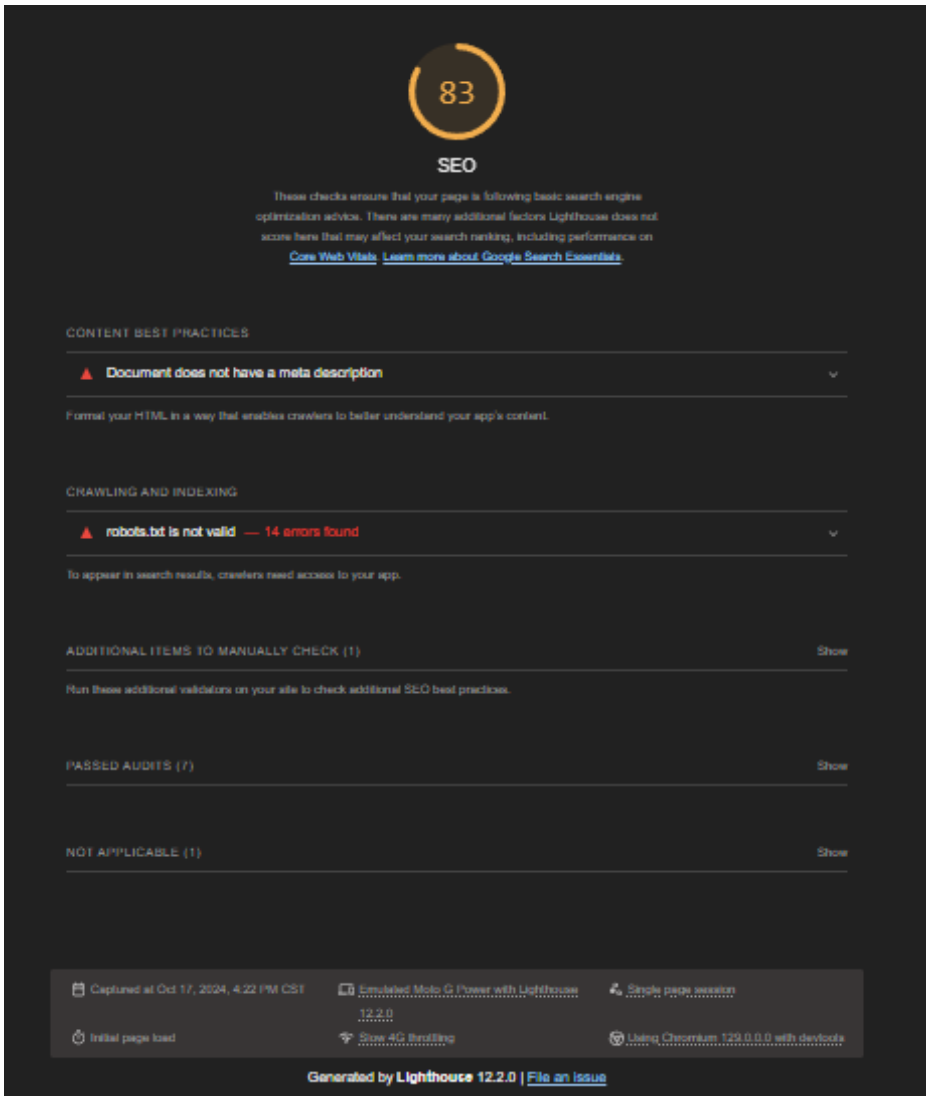


Figura 39. Lighthouse - *Search Engine Optimization* (SEO)

Por último, el SEO reflejado en la Figura 39 nos da un puntaje por arriba del 80 % el cual se podría mejorar añadiendo una descripción meta a la página y otros pequeños detalles que son más del lado de React, que de nuestra aplicación.

En conclusión, Lighthouse nos otorgó un ambiente adecuado para la evaluación de la página, que es recomendable usar en todo momento para depurar la página como herramienta de depuración para mejorar las diferentes métricas de la misma.

Linting

Los resultados de esta sección son breves, Vite viene con la librería de eslint instalada por defecto. eslint es nuestro depurador de código usado del cual se habló anteriormente.

- npm run lint

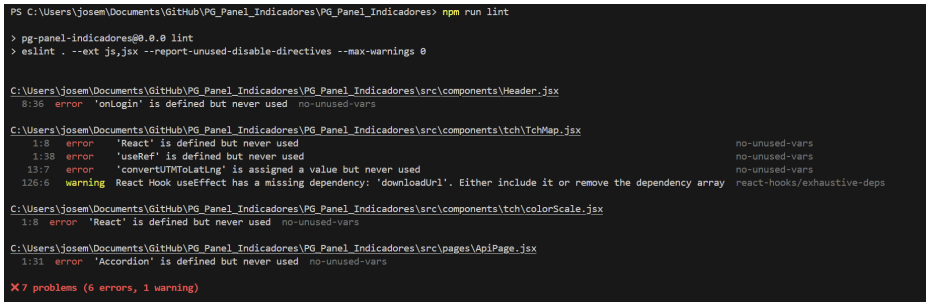


Figura 40. Linting - error de ejecución

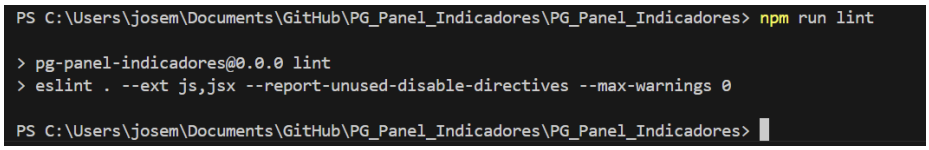


Figura 41. Linting - ejecución exitosa

El comando anterior es usado para depurar el código. La Figura 40 nos demuestra el comando en su forma de error, cuando tenemos código por corregir aún. Luego de corregir el código se nos despliega un mensaje de éxito que nos dice que el código está bien, como lo podemos ver en la Figura 41.

7.2.2. Resultado final ArcGIS

Al igual que en React esta solución se divide en diferentes páginas que llevan soluciones distintas a cada uno de los aspectos a tratar. En secciones anteriores se habló sobre la creación del sitio.

Página de inicio

Figura 42. ArcGIS - página de inicio

Como se puede ver en la Figura 42, se nos muestra una vista muy similar a la creada con React, lo cual era el objetivo. Con esto nos aseguramos una uniformidad entre plataformas para demostrar que se pueden lograr resultados muy parecidos, a pesar de ser tecnologías muy diferentes.

En el caso del Inicio, tenemos de igual manera un *Slider* para mostrar el título de la página. Una sección explicativa acerca de la funcionalidad de la página y Una sección de cuadros para las diferentes secciones de la página.

Es importante recalcar que ArcGIS como herramienta de desarrollo web, nos otorga un entorno *No-Code* con opción de *Low-Code* en ciertas ocasiones, es decir, la programación involucrada en el desarrollo del panel de control es prácticamente inexistente, sino que en su lugar tenemos un entorno muy parecido a "Wordpress"para las diferentes secciones de la página, esto lo podemos ver en la figura siguiente.

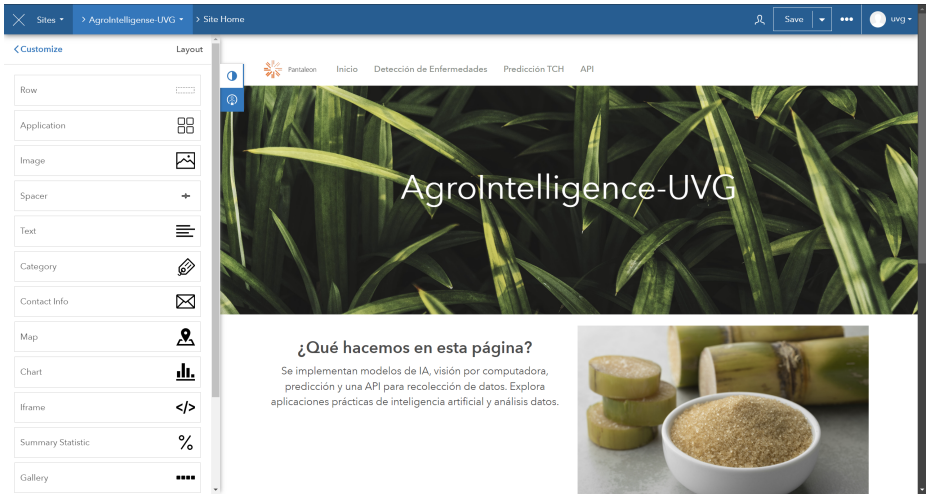


Figura 43. ArcGIS - página de inicio en edición

Página de detección de enfermedades

Para esta sección, se implementó una versión simplificada de la página con React, con la misma funcionalidad de subir imagen y detectar la plaga. Referencia de esto lo podemos ver en la figura a continuación.

Figura 44. ArcGIS - página de detección de enfermedades

Siguiendo el estilo de la página se despliega el resultado de la predicción con un pequeño párrafo de descripción y guías de acción con respecto de la enfermedad.

Página de predicción de TCH

Considero que esta sección nos muestra la fortaleza que tiene ArcGIS sobre un enfoque más convencional que React. Nos permitió crear un mapa mucho más robusto que el propio de Leaflet, otorgándonos un *software* especializado para la creación de entornos geo-espaciales.

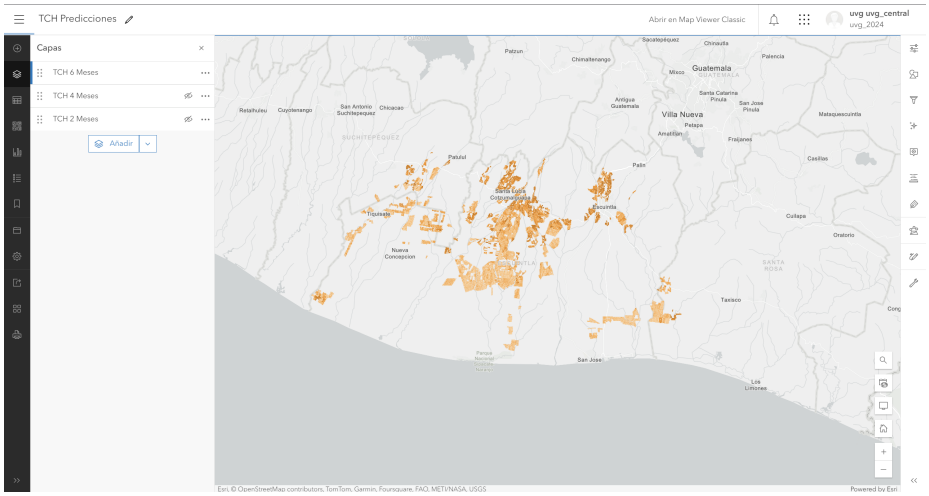


Figura 45. ArcGIS - página de mapa para predicción de TCH

Haciendo referencia a la figura anterior, podemos ver como es la vista de un creador de mapas especializado, este mapa se usará luego en una aplicación y en un Panel de Control tipo "Power Bi"para mostrarlos en la página principal de predicción de TCH luego. Como punto importante es que el mapa consume datos a partir de una URL en linea, con el objetivo de poder ser actualizable en tiempo real si así se requiere.

Luego se procedió a hacer una aplicación en torno al mapa, esto con el objetivo de desplegar el mapa creado anteriormente de una manera más llamativa, lo podemos ver en la figura que viene a continuación.

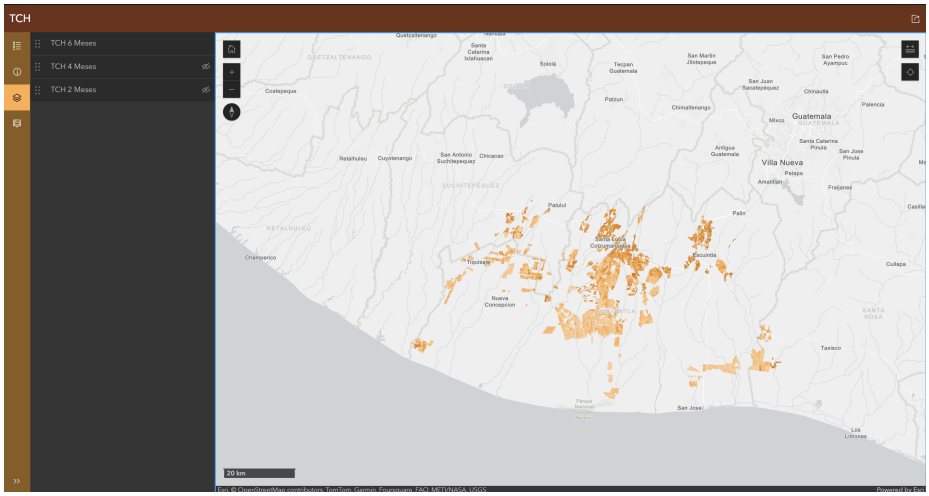


Figura 46. ArcGIS - página de aplicación de mapa para predicción de TCH

Podemos ver en la imagen colores llamativos y un mapa que podemos usar para interactuar con el y ver los diferentes valores del mismo.

Como siguiente punto, la creación de un Panel de Control es la herramienta que hace más llamativa esta solución en ArcGIS en comparación al mapa creado en React.

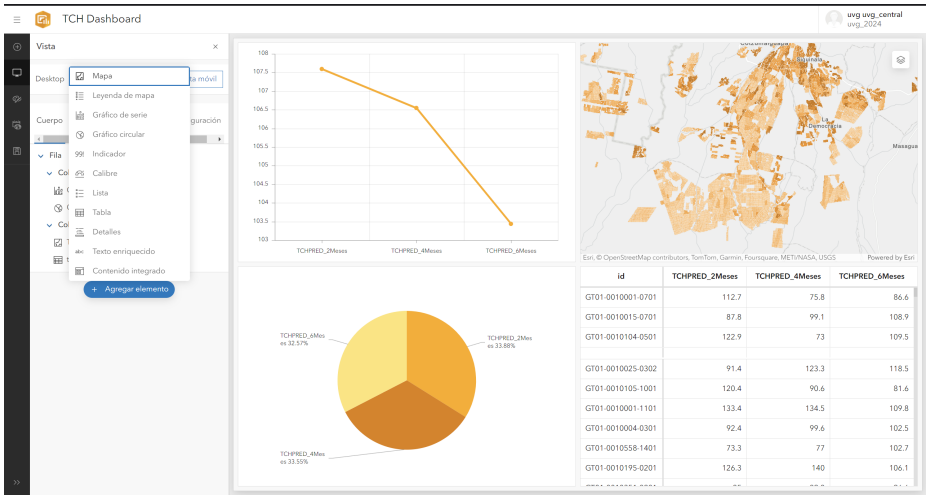


Figura 47. ArcGIS - página de panel de control de mapa para predicción de TCH

La figura anterior nos demuestra un poco del potencial de ArcGIS como marco de trabajo para realizar cuadros de mando para los mapas.

Con todo esto unificado, logramos conseguir una visión bastante profesional sobre el mapa que tenemos acerca de las predicciones de TCH esto lo podemos observar en la figura siguiente.

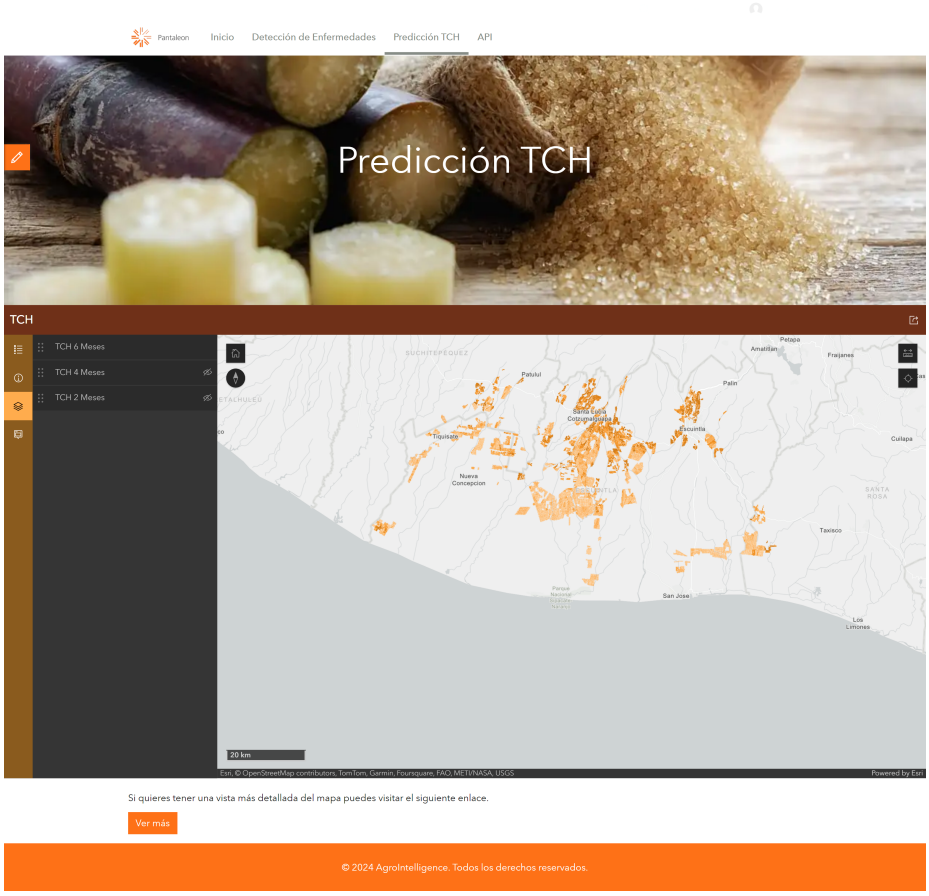


Figura 48. ArcGIS - página predicción de TCH

Nos muestra el *Slider* de la sección y luego la aplicación del mapa que nos permite interactuar con el mismo y seleccionar las diferentes capas.

Posteriormente se colocó una vista más como un cuadro de mando donde tenemos al mapa y secciones estadísticas, como gráficas de dispersión, de pie y la tabla de datos. Todo esto lo podemos ver reflejado en la siguiente figura.

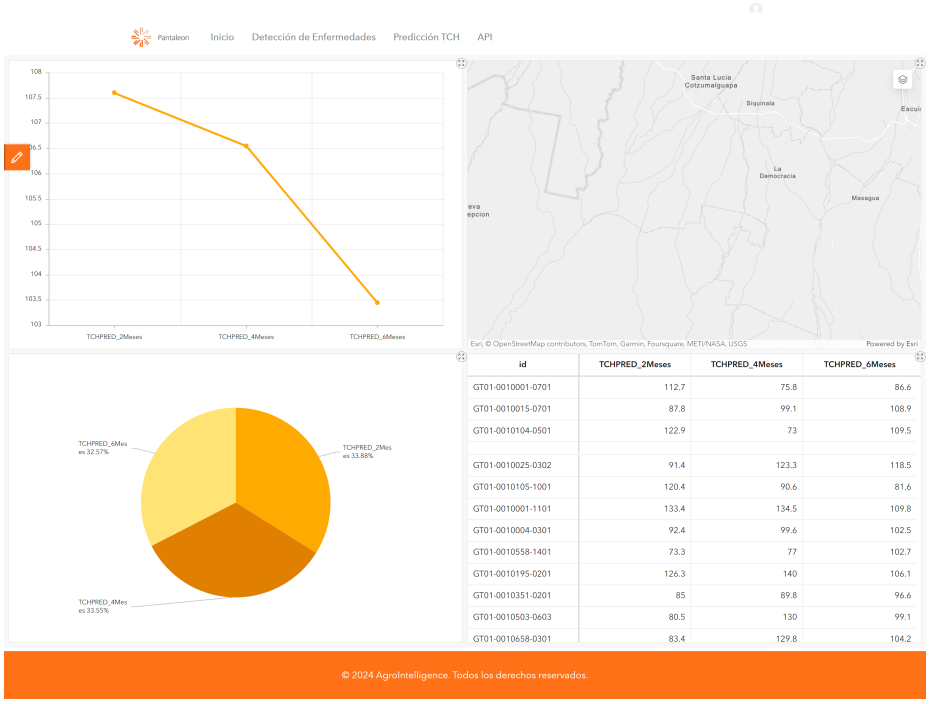


Figura 49. ArcGIS - ppágina panel de control predicción de TCH

Página de API

Para la sección del API en el Portal de ArcGIS se usó la misma solución que en el portal de React, un *embed* hacia la documentación del API creada por FastAPI, debido a que se mezcla bien con la página y le da un toque profesional a la misma. El resultado de esto lo podemos ver en la figura siguiente.

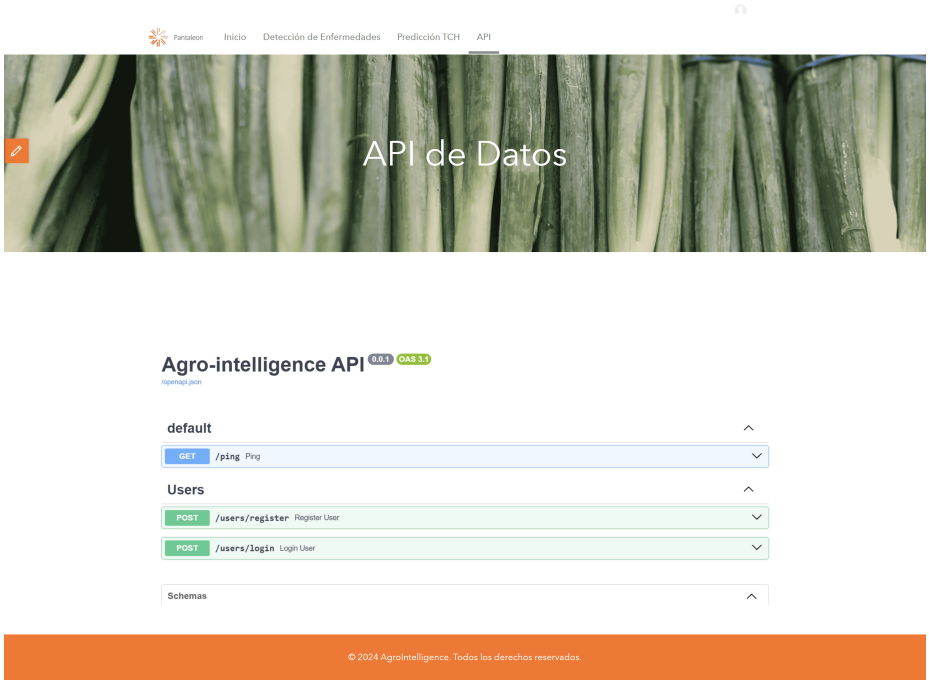


Figura 50. ArcGIS - página de API

7.2.3. Pruebas de navegación

Regresando al tema de la muestra y como la complementamos, luego de realizado el panel de control se realizaron 5 pruebas de navegación para probar las funcionalidades del panel de control de React, estas pruebas nos dieron diferentes puntos de retroalimentación que se implementaron en el producto final.

Las pruebas de navegación se realizaron con el dispositivo que sirve para capturar el movimiento de ojos en la pantalla llamado "Tobiiçomo se mencionó anteriormente. Este nos ayudó para poder observar los comportamientos de los diferentes usuarios en nuestra página, donde se creaba un mapa de calor en las secciones donde el usuario postraba su vista. A continuación se muestra un ejemplo de como se mira la navegación de la página usando "Tobii"para la detección de movimiento de ojos.

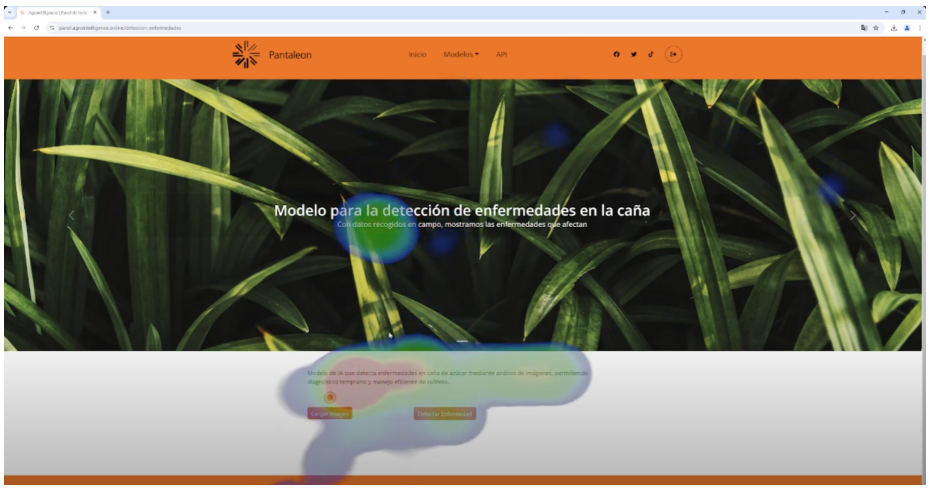


Figura 51. Tobii - ejemplo de uso

Se realizaron 5 pruebas de navegación con diferentes usuarios. 4 de ellos navegaron por la solución de React y 1 navegó tanto por la React como la de ArcGIS. El objetivo de las pruebas era cumplir una serie de objetivos en los paneles de control para demostrar el uso de los mismos.

Todas las pruebas de navegación se subieron a Youtube para mantenerlas guardadas. La primera de estas pruebas que se puede ver en el Anexo 56 nos demostró que la página cumple con los requerimientos para que la página se considere funcional. Podemos ver en un inicio como se fija en la barra de navegación, la cual al tener un color llamativo y textos grandes capta más la atención de la persona. Luego se fija en titulares y para cumplir las tareas realmente ignora los textos de descripción iniciales ya que son poco relevantes para cumplirlas. Se puede ver también como se ignora en cierta manera el botón de inicio de sesión en un inicio y se busca los modelos navegando por la página en lugar de acceder a ellos por medio del menú. Esto lo logramos solucionar gracias a tener las rutas protegidas de las diferentes secciones detrás del inicio de sesión forzado si no se tiene el usuario ingresado. Luego se procedió con el resto de las tareas sin mayor comentario más que resaltar que era intuitivo en todo momento que hacer lo cual facilitó el uso de la aplicación. Cabe resaltar que la primera persona no es muy adepta a la tecnología, por lo que completar las tareas de la manera fácil como lo hizo es gracias a la disposición de la página.

Como punto interesante de la segunda persona en realizar las pruebas de navegación es que la navegación a las diferentes secciones las realizó desde el menú principal, sin navegar por la página, lo cual nos beneficia porque se puede llegar a las diferentes secciones del panel de control sin realizar más de 3 çlicks". Gracias a lo anterior se redujo el tiempo de finalización de las tareas en casi un minuto y no tuvo mayor complicación para completarlas. El enlace a dicha prueba se encuentra en el Anexo 57.

La tercera persona, con prueba en el Anexo 58, tuvo un poco más de problemas para encontrar las secciones donde se debían realizar las diferentes tareas, esto se pudo deber más que todo a que las tareas no estaban bien definidas y de igual manera la persona no estaba muy familiarizada con la tecnología. La persona se tomó el tiempo de revisar las diferentes secciones del panel de control, pero una vez entendido el funcionamiento logró realizar las tareas sin contratiempos.

Por parte de la cuarta persona, esta ya era cercana al proyecto, por lo que sabía como navegar por el panel de control. La persona, al usar lentes para mejorar su visión, se ve como si estuviese viendo más abajo de lo que realmente está viendo, pero en términos de probar la navegación no falla realmente la disposición de la página dejándonos a entender que un flujo general se está siguiendo. Se puede observar el video en el Anexo 59

Por último, se realizó la prueba más exhaustiva con la última persona. Se evaluaron tanto el panel de control realizado con React, como el realizado en ArcGIS. Esta persona es experta en Q/A por lo que su prueba era de las que más importaban los resultados. Como punto interesante, esta fue la única persona que inicio sesión directamente desde el botón de usuario, por lo que se demostró que su implementación fue correcta y estaba bien entendido. Luego tomó en un tiempo en comprender la tarea 1, confundiendo la localización de la imagen, pero una vez ahí logró procesar rápido la información. Puntos importantes de esta prueba es que su navegación es mucho más rápida comparada con el resto, lo que nos da a entender que en el modelo de detección de enfermedades hace falta un *loader* para mostrar que el modelo está trabajando, solo se está llevando un poco más de tiempo en responder. Este fue el denominador común en toda la prueba, donde debido a su velocidad de navegación, ciertas funcionalidades no se mostraban por completo en un inicio lo cual la frustraba y pensaba que la página tenía más de algún error. Podemos ver la prueba completa en el Anexo 60

Las pruebas de navegación realizadas en la aplicación proporcionaron valiosos puntos de mejora, las cuales nos permitieron optimizar tanto la experiencia de usuario como la funcionalidad general del sistema. Uno de los principales cambios fue hacer más visible el botón de inicio de sesión en el menú de navegación. Este ajuste surgió de la retroalimentación de los usuarios, quienes consideraban que el acceso al inicio de sesión no era lo suficientemente evidente, lo que podía retrasar la autenticación. Al hacer este cambio, se busca fomentar una interacción más directa y fluida con la aplicación, permitiendo a los usuarios acceder de manera rápida a sus funciones personalizadas y mejorar la retención.

Además, se implementó un *loader* en la sección de detección de enfermedades. Este componente fue fundamental para mantener la comunicación visual con los usuarios, evitando que se enfrenten a una pantalla en blanco mientras el modelo de detección procesa la imagen. Al mejorar este punto en la interfaz de usuario, nos aseguramos que los usuarios comprendan que el sistema está en funcionamiento, lo que ayuda a reducir la ansiedad y mejora la percepción de eficiencia en la aplicación. Esto también permite a los usuarios apreciar el avance en el procesamiento de sus datos, lo cual es particularmente importante en aplicaciones que requieren un tiempo de espera considerable para el análisis y la predicción.

En términos generales, las pruebas fueron exitosas y cumplieron con los objetivos de usabilidad y funcionalidad planteados en el diseño inicial. Los resultados indicaron una navegación intuitiva y accesible a través del panel de control, con comentarios positivos tanto en entrevistas verbales como en videos grabados de las interacciones de los usuarios.

7.2.4. Pruebas unitarias

Vamos a hablar sobre la configuración para empezar.

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig({
  plugins: [react()],
  test: {
    environment: 'jsdom',
    setupFiles: './src/setupTests.js',
    globals: true,
  },
})
```

El código anterior es del archivo de configuración de Vite, "vite.config.js", el cual nos sirve para especificar el ambiente de pruebas que se usará en la aplicación, donde se especifican los "setupFiles.[en] el archivo siguiente.

```
// src/setupTests.js
import { vi } from 'vitest';
import '@testing-library/jest-dom';
import 'whatwg-fetch';
import { server } from './mocks/server';

// Mock de URL.createObjectURL para Vitest
globalThis.URL.createObjectURL = vi.fn(() => 'mocked-url');

// Mock de URL.revokeObjectURL
globalThis.URL.revokeObjectURL = vi.fn();

// Configuracion de MSW
beforeAll(() => server.listen());
afterEach(() => server.resetHandlers());
afterAll(() => server.close());
```

Se nos explica más que todo la configuración general de las pruebas para usar nuestras diferentes pruebas unitarias creadas.

Los *mocks* son elementos que nos ayudan a simular elementos de nuestra página web. Por ejemplo, un *mock* puede ir desde un elemento HTML como el carrusel de inicio hasta la respuesta que esperamos al usar el API, como se puede ver a continuación en el archivo "mocks/handlers.js".

```
// src/mocks/handlers.js
import { rest } from 'msw';

export const handlers = [
  rest.get(import.meta.env.VITE_URL_PREDICCION_TCH, (req, res, ctx) => {
    // Devuelve una respuesta mock del API
    const mockData = {
      type: 'FeatureCollection',
      features: [
        {
          type: 'Feature',
          properties: {
            id: '123',
            TCHPRED_6Meses: '50',
          },
          geometry: {
            type: 'Polygon',
            coordinates: [
              [
                [-90.785, 14.305],
                [-90.780, 14.310],
                [-90.775, 14.305],
                [-90.785, 14.305],
              ],
            ],
          },
        },
      ],
    };
    return res(ctx.status(200), ctx.json(mockData));
  }),
];
```

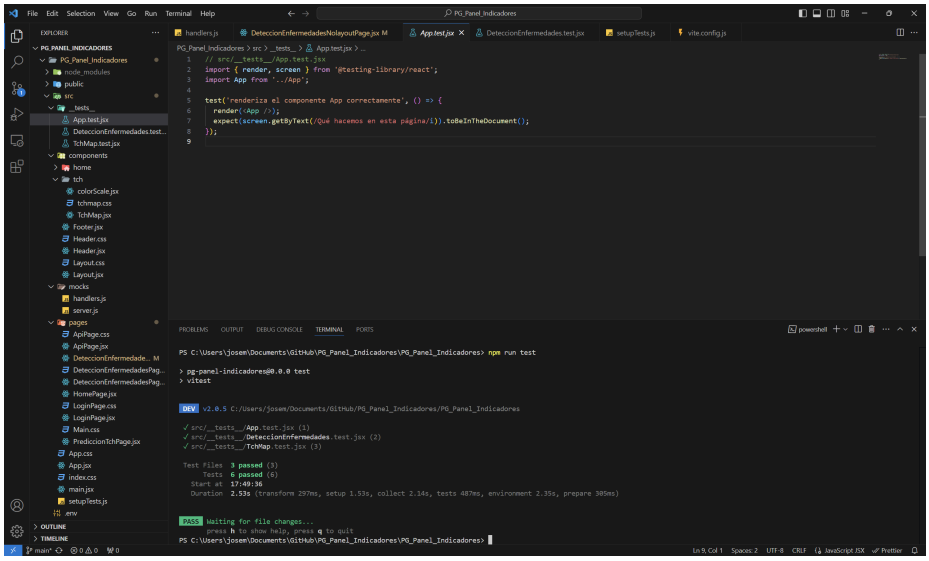


Figura 52. Vitest - pruebas unitarias

La figura anterior nos muestra el resultado de correr las pruebas unitarias. Se crearon tres archivos de pruebas unitarias, con sub-pruebas en cada uno de ellos. El primero de ellos, .ªpp.test.jsx"se encarga de verificar la carga de la aplicación.

```jsx
// src/__tests__/App.test.jsx
import { render, screen } from '@testing-library/react';
import App from '../App';

test('renderiza el componente App correctamente', () => {
  render(<App />);
  expect(screen.getByText(/Que hacemos en esta pagina/i)).toBeInTheDocument();
});
```

Como se ve anteriormente vamos a importar el archivo que queremos probar, importamos la librería para pruebas unitarias y luego hacemos el test. Este test es tan simple como verificar un texto de la página una vez renderizado, pero nos sirve como estructura para el resto de pruebas unitarias.

```jsx
// src/__tests__/DeteccionEnfermedades.test.jsx
import { render, screen, fireEvent, waitFor, act } from '@testing-library/react';
import DeteccionEnfermedadesPage from '../pages/DeteccionEnfermedadesPage';
import { server } from '../mocks/server';
import '@testing-library/jest-dom';

// Mock de FileReader y variable de entorno
beforeAll(() => {
  global.FileReader = class {
    readAsDataURL(file) {
      this.onload({ target: { result: '-image-data' } })
          ;
    }
  };
  import.meta.env.VITE_URL_DETECCION_ENFERMEDADES = 'http://localhost/mock-url';
});

describe('DeteccionEnfermedadesPage', () => {
  beforeAll(() => server.listen());
  afterEach(() => server.resetHandlers());
  afterAll(() => server.close());

  test('se renderiza correctamente', async  () => {
```

```
    await act(async () => {
      render(<DeteccionEnfermedadesPage />);
    });
    expect(screen.getByText(/Modelo para la deteccion de enfermedades en la cana/i)).
        toBeInTheDocument();
  });

  test('muestra error cuando no se ha subido imagen y se hace clic en "Detectar
      Enfermedad"', async () => {
    await act(async () => {
      render(<DeteccionEnfermedadesPage />);
    });
    const detectButton = screen.getByText('Detectar Enfermedad');
    fireEvent.click(detectButton);
    expect(screen.getByText(/Por favor, sube una imagen antes de detectar
        enfermedades/i)).toBeInTheDocument();
  });
});
```

El código anterior pertenece a "DeteccionEnfermedades.test.jsx"donde hacemos dos pruebas, uno para verificar que la página cargue correctamente. Y el segundo que verifica cuando no se ha subido una imagen y querramos detectar la enfermedad.

```
// src/components/tch/TchMap.test.jsx
import { render, screen, waitFor, fireEvent } from '@testing-library/react';
import TchMap from '../components/tch/TchMap';
import '@testing-library/jest-dom';
import { vi } from 'vitest';
import React from 'react';

// Mock de react-leaflet
vi.mock('react-leaflet', async () => {
  const actual = await vi.importActual('react-leaflet');
  return {
    ...actual,
    MapContainer: ({ children }) => <div data-testid="map-container">{children}</div
        >,
    TileLayer: () => <div data-testid="tile-layer" />,
    GeoJSON: () => <div data-testid="geojson" />,
    useMap: () => ({
      fitBounds: vi.fn(),
      getZoom: vi.fn().mockReturnValue(10),
      setZoom: vi.fn(),
    }),
  };
});

// Mock de leaflet
vi.mock('leaflet', () => {
  return {
    __esModule: true,
    default: {},
    latLngBounds: () => ({
      extend: () => {},
    }),
    map: () => ({
      fitBounds: () => {},
      getZoom: () => 10,
      setZoom: () => {},
      remove: () => {},
    }),
    tileLayer: () => ({}),
    geoJSON: () => ({
      addTo: () => {},
    }),
  };
});
```

```
44
45 describe('TchMap Component', () => {
46   test('renderiza el mapa', () => {
47     render(<TchMap />);
48     expect(screen.getByLabelText(/loading/i)).toBeInTheDocument();
49   });
50
51   test('verifica el mapa renderizado', async () => {
52     render(<TchMap />);
53     await waitFor(() => {
54       expect(screen.getByText(/Mapa de Prediccion de TCH/i)).toBeInTheDocument();
55     });
56   });
57
58   test('error de modal al no encontrar id', async () => {
59     render(<TchMap />);
60     await waitFor(() => {
61       expect(screen.getByText(/Mapa de Prediccion de TCH/i)).toBeInTheDocument();
62     });
63
64     const input = screen.getByPlaceholderText(/Ingrese el ID/i);
65     fireEvent.change(input, { target: { value: '999' } });
66
67     // Utiliza getByRole para seleccionar el boton
68     const searchButton = screen.getByRole('button', { name: /Buscar/i });
69     fireEvent.click(searchButton);
70
71     await waitFor(() => {
72       expect(screen.getByText(/No se encontro un poligono con el ID proporcionado/i))
           .toBeInTheDocument();
73     });
74   });
75 });
```

Por último probamos el mapa de predicción de TCH en el archivo "TchMap.test.js"que consiste de tres pruebas. Verificar el renderizado correcto de la página, luego del mapa que tarda un poco más y por último vamos a verificar el mensaje de error que aparece cuando encontramos un id en la búsqueda.

Con estas pruebas unitarias verificamos acciones simples en la página pero sentamos la base para la implementación de pruebas más avanzadas en un futuro y nos aseguramos de incluir el concepto en el desarrollo del panel de control.

CAPÍTULO 8

Discusión de resultados

8.1. Comparación de los resultados con las expectativas y objetivos

8.1.1. Logro del objetivo general

Se logró desarrollar una interfaz de usuario intuitiva para el panel de control interactivo y la aplicación web para facilitar a los agricultores y gestores de cultivos el acceso y la comprensión de los análisis de datos y recomendaciones ofrecidos por el sistema.

Para el éxito del objetivo general se desarrollaron dos soluciones, una aplicación en Panel de Control en React y un Panel de Control en el marco de trabajo ArcGIS. Siendo el panel de control con React la principal de las soluciones.

Se desarrolló de esta manera debido a que el Ingenio Pantaleón tiene todas sus páginas web alojadas en la tecnología ArcGIS, que como se ha explicado reiteradamente es un motor bastante potente para el desarrollo de cuadros de mando que tienen capas de mapas involucradas. Este fue el caso para la sección de Predicción de Toneladas de Caña por Hectarea que desplegaba los resultados en un mapa.

Hubiese sido la solución principal ArcGIS si no solo se tratara de un mapa, también se involucraban más secciones en el proyecto, por lo que una aplicación en React terminó siendo la propuesta final. Dándole al Ingenio una herramienta extra que les permite tanto la gestión de los resultados del modelo de predicción, como también la posibilidad de detectar enfermedades en la planta de caña de azúcar con un modelo por visión computacional que no hubiese sido pposible desde el marco de trabajo de ArcGIS.

8.1.2. Logro de los objetivos específicos

Validación con usuarios

Se logró validar la eficacia de la interfaz de usuario diseñada mediante pruebas con usuarios finales, para así asegurar que los agricultores y gestores de cultivos puedan utilizar efectivamente los análisis de datos y recomendaciones dadas.

Para el logro de este objetivo se llevaron a cabo las pruebas de navegación hechas a usuarios de interés, donde se navegaba a trabes de la página para cumplir con objetivos dados a manera de pruebas unitarias.

Implementación de modelo clasificatorio

Se logró implementar modelos de inteligencia artificial que procesan imágenes y datos sensoriales por medio del API con el Data Center para identificar y clasificar eficazmente enfermedades en cultivos de caña de azúcar.

Se implementó el modelo de detección de enfermedades realizado en otro módulo, el cual se encargaba de recibir una imagen de una planta caña de azúcar, que retornaba una clasificación de enfermedad en caso tuviera una y si no se clasificaba como hoja sana.

Para conseguir cumplir el objetivo se realizó la comunicación exitosa con el API para mandar y recibir solicitudes de este modelo.

Desarrollo del modelo predictivo

Se desarrolló un panel de control interactivo para indicadores definidos para poder visualizar en tiempo real datos relevantes antes de fin de año.

Con ayuda de herramientas geoespaciales como Leaflet se logró mostrar un mapa interactivo con las parcelas de producción de caña de azúcar, donde cada una mostraba un valor relacionado con su predicción de caña de azúcar para la zafra actual en diferentes modelos realizados de 2 meses, 4 meses, 6 meses y 8 meses, con el objetivo de mostrar la producción futura en la actualidad.

Se implementó con éxito consumiendo un archivo geojson alojado en linea, para siempre tener disponibilidad de datos actualizados (el módelo no se actualiza con regularidad necesariamente). Una vez consumido el archivo, se extraían las propiedades y la geometría de las parcelas y se procedían a implementar en el mapa para su despliegue.

De igual manera, se colocaron filtros, campo de búsqueda de parcela y una opción para la descarga de los datos, para asegurarnos que fuera lo más amigable para el usuario y facilitara su uso.

Funcionalidades específicas

Se implementaron funcionalidades en el panel de control que presenten análisis de datos y recomendaciones de manera visualmente intuitiva y fácilmente comprensible, permitiendo a los agricultores y gestores realizar decisiones informadas para el manejo óptimo de sus cultivos.

Retomando los temas hablados en los dos objetivos anteriores, se logró crear un panel de control simplificado para que el usuario pudiera navegar en el de manera fácil, con secciones divididas para su comprensión donde se pudiera interactuar y recibir respuestas en base a las acciones tomadas.

Retroalimentación

Se logró modificar la interfaz y funcionalidades del sistema basándose en la retroalimentación recibida para mejorar la usabilidad y eficacia del panel de control y la aplicación web.

Para este punto la retroalimentación ocurrió en todo momento, no solo al final del desarrollo. Al inicio del mismo, se llevaron a cabo encuestas sobre las funcionalidades que iban a estar presentes en la aplicación final, se hicieron cambios sobre los comentarios recibidos, luego se llevó a cabo el desarrollo del panel y al terminarlo se recibió mas retroalimentación que llevó a la mejora del producto final.

Al ser un desarrollo del cual se tenía resultados en todo momento, la retroalimentación fue clave para cumplir con las expectativas de los usuarios en todo momento. Esto contaba tanto para el panel de control implementado con React, como el realizado en ArcGIS.

Aplicación en ArcGIS

Por último, se implementó una aplicación especializada de ArcGIS como complemento del sistema principal, adaptada a las necesidades del Ingenio Pantaleón, con el fin de mejorar la precisión del mapeo de cultivos y optimizar la gestión de recursos agrícolas.

Como solución alterna tuvo tanto sus fortalezas como debilidades. La principal fortaleza era el manejo de datos geoespaciales que nos beneficiamos de la herramienta para crear una aplicación especializada con gráficas y datos que nos permitieran interactuar con el mapa, el cual al estar optimizado para este tipo de trabajos respondía mucho más rápido en la carga de los datos.

Sus debilidades no eran evidentes de todos modos, estas solo eran limitadas, es decir, el modelo de detección de enfermedades y la API eran complemento para este panel sin ser protagonistas.

8.2. Impacto del panel de control en la gestión de cultivos de caña de azúcar

El desarrollo e implementación del panel de control ha tenido un impacto significativo en la gestión de cultivos de caña de azúcar, mejorando diversos aspectos del proceso agrícola:

La toma de decisiones informada ha experimentado una mejora sustancial gracias al panel de control. Este proporciona análisis de datos y recomendaciones de manera visualmente intuitiva y fácilmente comprensible, permitiendo a agricultores y gestores realizar decisiones más acertadas para el manejo óptimo de sus cultivos. La presentación clara y accesible de información compleja facilita una gestión más eficiente y estratégica de los recursos agrícolas.

La predicción de rendimientos se ha convertido en una herramienta valiosa para la planificación agrícola. El panel de control incorpora un mapa interactivo que muestra predicciones de producción de caña de azúcar para la zafra actual, con modelos predictivos a 2, 4, 6 y 8 meses. Esta funcionalidad ofrece una visión a futuro de la producción, permitiendo a los agricultores anticipar y prepararse para las fluctuaciones en el rendimiento de los cultivos.

Continuando, la detección temprana de enfermedades ha dado un salto cualitativo con la integración de un modelo de inteligencia artificial en el panel. Este sistema procesa imágenes y datos sensoriales para identificar y clasificar eficazmente enfermedades en cultivos de caña de azúcar. La capacidad de realizar un diagnóstico rápido y preciso permite intervenciones tempranas, reduciendo potencialmente las pérdidas de cultivos y optimizando el uso de tratamientos.

La optimización de recursos se ha visto favorecida por la aplicación complementaria desarrollada en ArcGIS. Esta herramienta mejora la precisión del mapeo de cultivos y facilita una gestión más eficiente de los recursos agrícolas. La capacidad de visualizar y analizar datos geoespaciales de manera precisa permite una distribución más efectiva de insumos y esfuerzos de cultivo.

De igual manera, la accesibilidad de la información ha mejorado significativamente con la interfaz de usuario intuitiva del panel de control. Las funcionalidades como filtros, búsqueda de parcelas y opciones de descarga de datos mejoran la usabilidad, permitiendo a los usuarios acceder y comprender rápidamente análisis de datos complejos. Esta facilidad de acceso democratiza la información, permitiendo que más personal involucrado en la gestión de cultivos pueda beneficiarse de los datos disponibles.

El monitoreo en tiempo real se ha convertido en una realidad gracias al panel de control. La visualización de indicadores relevantes en tiempo real, junto con la actualización constante de datos mediante la conexión con archivos geojson en línea, permite a los gestores mantenerse al tanto de las condiciones actuales de los cultivos. Esta capacidad de seguimiento continuo facilita respuestas rápidas a cambios en las condiciones de cultivo.

La adaptabilidad y mejora continua han sido características clave en el desarrollo del panel de control. El proceso de desarrollo incluyó retroalimentación constante de los usuarios, con modificaciones basadas en pruebas de campo y comentarios de un grupo piloto de agricultores. Esta aproximación iterativa asegura que el panel de control evolucione continuamente para satisfacer las necesidades cambiantes de los usuarios y adaptarse a nuevas tecnologías o métodos agrícolas.

También, la integración tecnológica lograda con el panel de control representa un avance significativo en la gestión agrícola. La combinación de soluciones (React y ArcGIS) se adapta a las necesidades específicas del Ingenio Pantaleón, aprovechando las fortalezas de cada tecnología. El uso de tecnologías geoespaciales permite una gestión más precisa de las parcelas, mientras que la flexibilidad de React facilita la implementación de funcionalidades avanzadas como la detección de enfermedades mediante visión por computadora.

En conjunto, el impacto del panel de control se traduce en una gestión más eficiente, precisa y proactiva de los cultivos de caña de azúcar. Esta mejora en la gestión tiene el potencial de aumentar significativamente la productividad y sostenibilidad de las operaciones agrícolas, proporcionando a los agricultores y gestores las herramientas necesarias para enfrentar los desafíos de la agricultura moderna.

CAPÍTULO 9

Conclusiones

El desarrollo e implementación del panel de control interactivo para la gestión de cultivos de caña de azúcar ha demostrado ser un proyecto exitoso y de gran impacto en la optimización de procesos agrícolas. A partir de los resultados obtenidos y el análisis del impacto generado, se pueden extraer las siguientes conclusiones:

- Se logró desarrollar con éxito una interfaz de usuario intuitiva que facilita el acceso y comprensión de análisis de datos complejos para agricultores y gestores de cultivos. La implementación de dos soluciones complementarias - un panel de control en React y otro en ArcGIS - demuestra la capacidad de adaptación a las necesidades específicas del Ingenio Pantaleón, superando las expectativas iniciales del proyecto.
- La combinación de diferentes tecnologías, como React para la interfaz principal y ArcGIS para el manejo de datos geoespaciales, ha resultado en una solución robusta y versátil. Esta integración permite abordar diversos aspectos de la gestión agrícola, desde la predicción de rendimientos hasta la detección de enfermedades, en una única plataforma unificada.
- El panel de control ha demostrado ser una herramienta valiosa para la toma de decisiones informadas en la gestión de cultivos. La presentación visual intuitiva de datos complejos y las funcionalidades de predicción permiten a los usuarios anticipar escenarios y optimizar sus estrategias de cultivo.
- La implementación de modelos de inteligencia artificial para la detección de enfermedades y la predicción de rendimientos ha elevado significativamente la capacidad de gestión proactiva de los cultivos. Esto se traduce en una potencial reducción de pérdidas y un aumento en la productividad general.
- El enfoque iterativo adoptado durante el desarrollo, que incluyó pruebas de campo y retroalimentación constante de los usuarios, ha resultado en un producto final altamente adaptado a las necesidades reales de los agricultores y gestores. Este proceso de mejora continua sienta las bases para futuras actualizaciones y expansiones del sistema.
- El proyecto representa un paso significativo hacia la implementación de prácticas de agricultura de precisión en el cultivo de caña de azúcar. La capacidad de monitoreo en tiempo real y análisis predictivo permite una gestión más precisa y eficiente de los recursos agrícolas.

- Aunque desarrollado específicamente para el Ingenio Pantaleón, el sistema demuestra un alto potencial de escalabilidad y adaptación a otros contextos agrícolas. La arquitectura flexible y modular del panel de control permite su extensión a otros tipos de cultivos o regiones geográficas.

En conclusión, el desarrollo de este panel de control interactivo no solo ha cumplido con los objetivos propuestos inicialmente, sino que ha superado las expectativas en términos de funcionalidad e impacto. Representa un avance significativo en la aplicación de tecnologías de la información y la inteligencia artificial en el sector agrícola, específicamente en el cultivo de caña de azúcar.

CAPÍTULO 10

Recomendaciones

Basándonos en la experiencia adquirida durante el desarrollo del panel de control interactivo para la gestión de cultivos de caña de azúcar, se presentan las siguientes recomendaciones para futuros desarrollos y mejoras del sistema:

Configuración adecuada del entorno de React:

- Utilizar herramientas como Create React App para configurar rápidamente un entorno de desarrollo robusto.
- Implementar un sistema de control de versiones (como Git) desde el inicio del proyecto.
- Configurar linters (ESLint) y formateadores de código (Prettier) para mantener un estilo de código consistente.
- Utilizar gestores de paquetes como npm para manejar las dependencias del proyecto.

Modularización de archivos

- Adoptar un enfoque de componentes reutilizables para maximizar la eficiencia del código.
- Implementar una arquitectura basada en servicios para separar la lógica de negocio de la interfaz de usuario.
- Utilizar hooks personalizados de React para encapsular lógica compleja y reutilizable.

Estructura ordenada de archivos

- Organizar los archivos por funcionalidad o tipo (componentes, pages, etc.).
- Utilizar una convención de nombres clara y consistente para todos los archivos y carpetas.
- Implementar un sistema de importación de módulos que facilite la navegación y el mantenimiento del código.

Limpieza de datos adecuada para el manejo de mapas

- Implementar funciones de validación y limpieza de datos antes de su uso en componentes de mapas.
- Utilizar bibliotecas como Turf.js para el procesamiento y manipulación eficiente de datos geoespaciales.
- Herramientas como Leaflet suelen ser el estándar para la creación de mapas en aplicaciones web.
- Considerar la implementación de un sistema de caché para datos geoespaciales frecuentemente utilizados.

Optimización de recursos

- Imágenes en formato ".webp"suelen ser más eficientes para aplicaciones web.
- Hacer uso de "loaders"para la carga de componentes pesados.

Uso de herramientas externas para validación

- Utilizar Coolors o Adobe Color para la selección y validación de paletas de colores accesibles.
- Implementar pruebas de rendimiento regulares utilizando Google Lighthouse o WebPageTest.

Experiencia de usuario (UX)

- Realizar pruebas de usabilidad periódicas con usuarios finales para identificar áreas de mejora.
- Implementar un sistema de recopilación de feedback dentro de la aplicación para captación continua de sugerencias de los usuarios.

Adaptabilidad a diferentes dispositivos

- Asegurar que el diseño sea completamente responsivo para su uso en dispositivos móviles y tabletas.
- Considerar el desarrollo de una aplicación móvil nativa para funcionalidades clave que requieran acceso en el campo.

Seguridad de Datos

- A pesar que se tienen procesos de autentificación en la aplicación desde el «Front-End», se debe implementar un proceso de autentificación desde el API para una aplicación más robusta.
- Proteger los puntos de enlace del API para que no sean accesibles por cualquier usuario.

Optimización de recursos en el uso de modelos de IA

- Implementar técnicas de compresión de modelos, como la cuantización, para reducir el tamaño y la carga computacional de los modelos de IA, disminuyendo así el consumo energético.
- Considerar el uso de modelos ligeros o versiones optimizadas de redes neuronales, que ofrecen un balance entre precisión y eficiencia energética.
- Aprovechar los sistemas de ejecución en la nube con energía renovable, lo que permite un uso más sostenible de los recursos computacionales.
- Realizar un monitoreo regular del consumo energético y optimizar el pipeline de inferencia, activando instancias de procesamiento solo cuando sea necesario para reducir el uso continuo de recursos.
- Integrar un sistema de apagado automático para instancias de procesamiento en la nube cuando no se esté ejecutando ninguna tarea de IA activa, disminuyendo el tiempo de actividad innecesario y el impacto ambiental.

Estas recomendaciones buscan no solo mejorar la calidad técnica y la eficiencia del panel de control, sino también asegurar su escalabilidad, mantenibilidad y relevancia a largo plazo en el contexto de la agricultura de precisión.

CAPÍTULO 11

Bibliografía

Bibliografía

[1] Pantaleón. Pantaleón. https://www.pantaleon.com/, 2016.

[2] Alliance Bioversity International CIAT. Artificial intelligence: How could it transform agriculture? https://alliancebioversityciat.org/stories/artificial-intelligence-agriculture, 2022.

[3] IDAP Blog. Ai in agriculture: Examples, benefits, challenges [latest data] - idap blog. https://idapgroup.com/blog/artificial-intelligence-in-agriculture/, 2021.

[4] FOQUM. Procesamiento de imágenes | foqum. https://foqum.io/blog/termino/procesamiento-de-imagenes/#:~:text=El%20procesamiento%20de%20im%C3%A1genes%20en,%2C%20superresoluci%C3%B3n%2C%20y%20muchas%20otras., 2024.

[5] Google Cloud. ¿qué son las analíticas predictivas y cómo funcionan? | google cloud. https://cloud.google.com/learn/what-is-predictive-analytics?hl=es#:~:text=Definici%C3%B3n%20de%20an%C3%A1lisis%20predictivo,-Las%20anal%C3%ADticas%20predictivas&text=El%20proceso%20utiliza%20an%C3%A1lisis%20de,puedan%20predecir%20el%20comportamiento%20futuro., 2024.

[6] Red Hat. La integración y la distribución continuas (ci/cd). https://www.redhat.com/es/topics/devops/what-is-ci-cd, 2022.

[7] Yiannis Ampatzidis. Applications of artificial intelligence for precision agriculture. *EDIS*, 2018(6), 2018.

[8] Maitiniyazi Maimaitijiang, Vasit Sagan, Paheding Sidike, Abdoul-Moubarak Daloye, Hasanjan Erkbol, and Felix B. Fritschi. Crop monitoring using satellite/uav data fusion and machine learning. *Remote Sensing*, 12(9):1357–1357, 2020.

[9] World Bank. Impacto ambiental y sostenibilidad en la agricultura inteligente. https://www.worldbank.org/en/topic/climate-smart-agriculture, 2021.

[10] World Bank. Resiliencia climática a través de la agricultura inteligente. https://www.worldbank.org/en/topic/climate-smart-agriculture, 2021.

[11] J. P. Ponce Pivaral. *Aplicación para el conteo y clasificación de huevos de mosquito con inteligencia artificial*. PhD thesis, Universidad del Valle de Guatemala, Facultad de Ingeniería, 2023.

[12] P. D. Rosero-Montalvo, C. A. Gordillo-Gordillo, and W. Hernandez. Smart farming robot for detecting environmental conditions in a greenhouse. *IEEE Access*, 11:57843–57853, 2023.

[13] Jiva.ag. How artificial intelligence can be used in agriculture. https://www.jiva.ag/blog/how-artificial-intelligence-can-be-used-in-agriculture, 2023.

[14] MongoDB. Cassandra vs mongodb comparison | mongodb. https://www.mongodb.com/resources/compare/cassandra-vs-mongodb, 2024.

[15] Github. Acerca de github y git. https://docs.github.com/es/get-started/start-your-journey/about-github-and-git, 2024.

[16] React. React. https://es.react.dev/, 2015.

[17] Vite. Vite - next generation frontend tooling. https://vitejs.dev/, 2020.

[18] Node.js. Node.js — run javascript everywhere. https://nodejs.org/en, 2024.

[19] Bootstrap. Bootstrap. https://getbootstrap.com/, 2024.

[20] ESLint. Eslint. https://eslint.org/docs/latest/, 2023.

[21] W3C. Html & css - the building blocks of the web. https://www.w3.org/standards/webdesign/htmlcss.html, 1999.

[22] Gustavo B. ¿qué es el hosting web y cómo funciona? https://www.hostinger.es/tutoriales/que-es-un-hosting, 2024.

[23] Cloudflare. Cloudflare registrar. https://www.cloudflare.com/es-es/products/registrar/, 2024.

[24] Sigsa. ¿qué es arcgis? | plataforma de mapeo y analítica. https://www.sigsa.info/es-mx/arcgis/about-arcgis/overview, 2024.

[25] IBM. Ibm - united states. https://www.ibm.com/us-en, 2024.

[26] IEEE JAS. Reducción del uso de recursos en la agricultura moderna. https://ieeexplore.ieee.org/document/9278790, 2024.

[27] FAO. Conservación de la biodiversidad en sistemas agrícolas. https://www.fao.org/conservation-agriculture/en/, 2024.

[28] National Academies Press. Mejora de la salud del suelo en la agricultura sostenible. https://www.nap.edu/catalog/12455/emerging-technologies-to-benefit-farmers, 2009.

[29] World Bank. Reducción de emisiones de gases de efecto invernadero en la agricultura. https://www.worldbank.org/en/topic/climate-smart-agriculture, 2021.

[30] IBM. Sostenibilidad a largo plazo en la agricultura inteligente. https://www.ibm.com/topics/smart-farming, 2023.

[31] Frontiers in Sustainable Food Systems. Economía circular en la agricultura. https://doi.org/10.3389/fsufs.2023.1170380, 2023.

[32] FAO. Resistencia al cambio en la adopción de tecnologías agrícolas. https://www.fao.org/conservation-agriculture/en/, 2024.

[33] IEEE JAS. Necesidad de datos de alta calidad en la agricultura inteligente. https://ieeexplore.ieee.org/document/9278790, 2024.

[34] MDPI. Integración de nuevas tecnologías con sistemas agrícolas existentes. https://www.mdpi.com/journal/sensors, 2024.

[35] McKinsey & Company. Costos de implementación y escalabilidad de tecnologías en la agricultura. https://www.mckinsey.com/industries/agriculture/our-insights/agricultures-connected-future, 2020.

[36] MDPI. Cuestiones de privacidad y seguridad de datos en la agricultura digital. https://www.mdpi.com/journal/sensors, 2024.

[37] FastAPI. Fastapi. https://fastapi.tiangolo.com/, 2024.

[38] Tushar Pol. Google lighthouse: Qué es y cómo utilizarlo. https://es.semrush.com/blog/como-utilizar-google-lighthouse/, 2023.

[39] Will Adamas. Why you should add eslint to a react vite project. https://dev.to/bushblade/add-eslint-to-a-react-vite-project-4pib, 2023.

[40] Tobii. Tobii. https://www.tobii.com/, 2023.

[41] Aerobotics. About us. https://www.aerobotics.com/about-us?locale=es, 2024.

[42] HEMAV. Plataforma de inteligencia agrícola. https://hemav.com/, 2024.

[43] PlantVillage. Plantvillage. https://plantvillage.psu.edu/, 2024.

[44] Coolors. Coolors. https://coolors.co/, 2024.

[45] W3.org. Web content accessibility guidelines (wcag) 2.1. https://www.w3.org/TR/WCAG21/, 2023.

[46] SurveyMonkey. Calculadora del tamaño de muestra. https://es.surveymonkey.com/mp/sample-size-calculator/, 2023.

CAPÍTULO 12

Anexos

Formulario de google

Figura 53. Acceso al formulario para el prototipo del proyecto

Resultados del formulario de google

Figura 54. Respuestas del formulario para el prototipo del proyecto

Repositorio del proyecto

Figura 55. Repositorio del proyecto en React

Prueba de navegación 1

Figura 56. Prueba de navegación 1

Prueba de navegación 2

Figura 57. Prueba de navegación 2

Figura 61. Prueba de navegación 1 - IRL

Prueba de navegación 3

Figura 58. Prueba de navegación 3

Prueba de navegación 4

Figura 59. Prueba de navegación 4

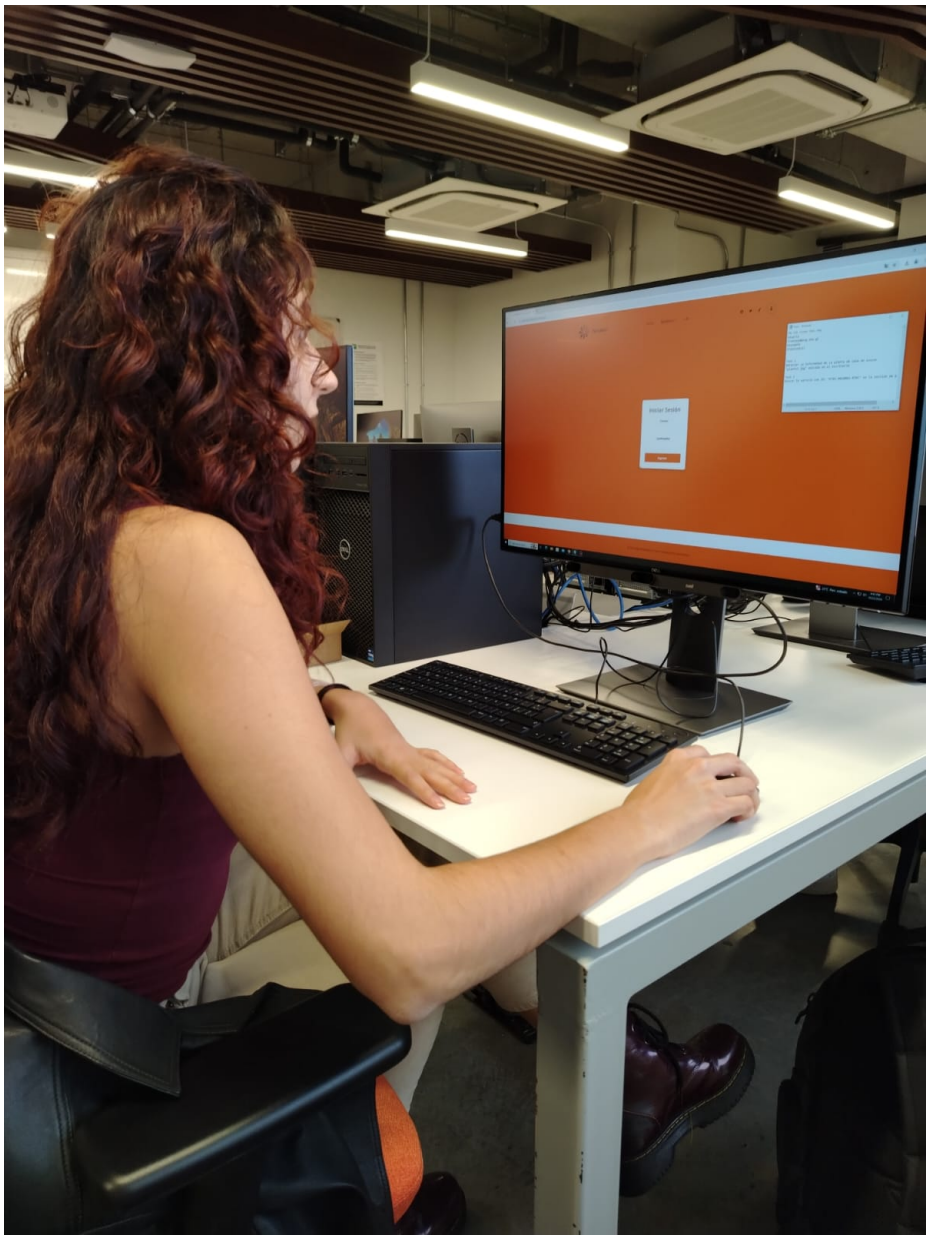


Figura 62. Prueba de navegación 2 - IRL

Prueba de navegación 5

Figura 60. Prueba de navegación 5

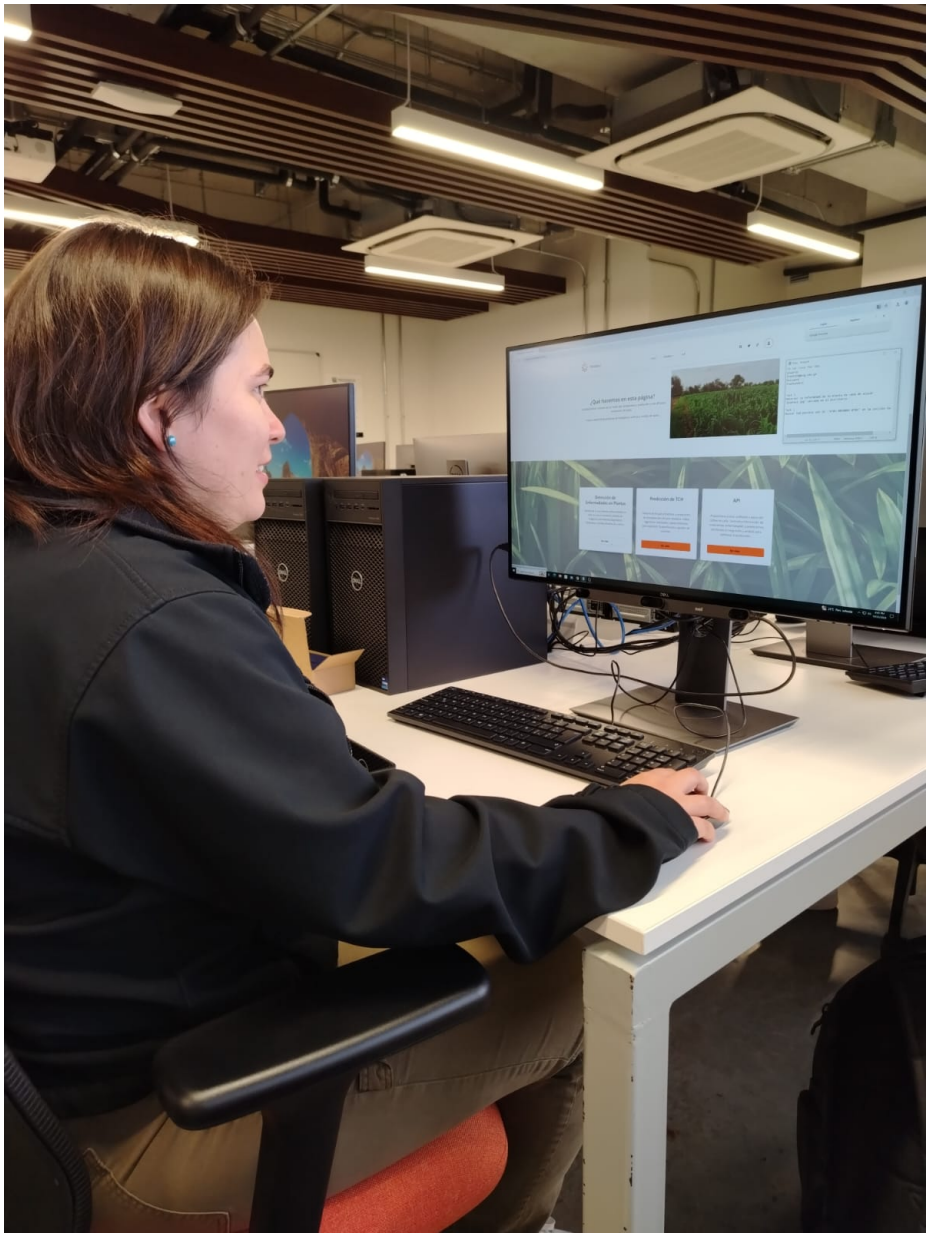


Figura 63. Prueba de navegación 3 - IRL

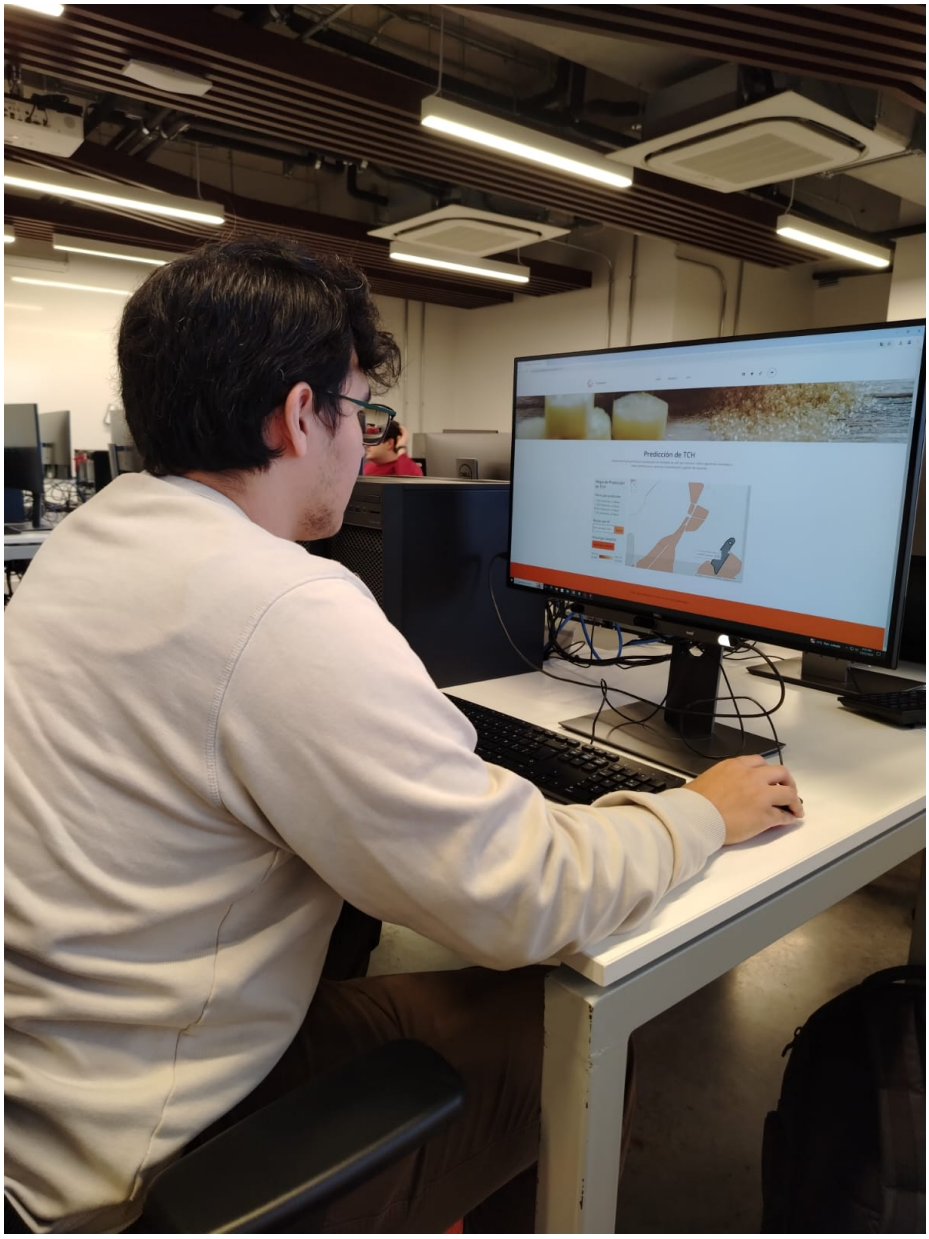


Figura 64. Prueba de navegación 4 - IRL