

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Implementación Backend de una Plataforma Web, para la predicción de productividad de árboles frutales en predios agrícolas

Javier Felipe Lucero Díaz

javier.lucero@usm.cl

Lais San Martín Navarro
Profesora Guía

Rodrigo Pinochet Ponce
Profesor Correferente

Resumen: Chile es un país con fuerte actividad agrícola y una importante producción frutícola destinada tanto al mercado interno como a la exportación. Sin embargo, la planificación productiva aún depende de métodos tradicionales y de la experiencia empírica de los agricultores, lo que genera incertidumbre frente a factores climáticos y de gestión agronómica. Ante este escenario, se propone el desarrollo de AgroPredict, una plataforma web que permita predecir la productividad de árboles frutales en diferentes predios agrícolas. El presente trabajo se enfoca en la implementación del backend de dicha plataforma, cuyo objetivo es construir la base lógica y funcional del sistema: gestionar usuarios, integrar datos externos (como información climática mediante la API AccuWeather) y soportar la incorporación de modelos predictivos. La metodología utilizada fue incremental y modular, desarrollando y validando cada componente por etapas: autenticación, gestión de predicciones, conexión con API y pruebas con modelos simples. Como resultado, se obtuvo un backend funcional, seguro y escalable, capaz de conectar información climática y datos de usuarios en un flujo coherente. Se espera que esta solución sienta las bases para una herramienta moderna que mejore la toma de decisiones agrícolas y promueva el uso de tecnología en el sector frutícola chileno.

Palabras Clave: Modelo Predictivo, Agricultura de precisión, Predicción de productividad, Django.



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Implementación Backend de una Plataforma Web, para la predicción de productividad de árboles frutales en predios agrícolas

Nombre del candidato(a): Javier Felipe Lucero Díaz

Carrera / Grado: Ingeniería en Informática

Campus: Viña del Mar **Departamento:** Electrotecnia e Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Lais San Martín Navarro, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (**marcar una opción**):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 21/09/2026

Firma:

Estudiante o Candidato(a):

Fecha: 20/09/2026

Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

1 Introducción

1.1 Contexto y Antecedentes

Durante las últimas décadas, la agricultura chilena ha experimentado un proceso de modernización gradual impulsado por la demanda internacional y la necesidad de aumentar la competitividad. En este contexto, la fruticultura se ha consolidado como uno de los principales motores económicos del país, destacando cultivos como las uvas, manzanas, cerezas y arándanos, cuyos volúmenes de exportación posicionan a Chile entre los líderes mundiales del rubro.

No obstante, la productividad agrícola sigue enfrentando desafíos estructurales vinculados a la variabilidad climática, la escasez hídrica y la limitada adopción de tecnologías de análisis y predicción. En muchas explotaciones, especialmente medianas y pequeñas, la toma de decisiones productivas continúa basándose en la experiencia empírica, sin apoyo en herramientas digitales que integren datos ambientales y productivos. Esta falta de información procesada genera incertidumbre y dificulta anticipar rendimientos o ajustar estrategias de manejo.

El avance de la agricultura de precisión y el desarrollo de modelos predictivos abren nuevas oportunidades para optimizar la gestión agrícola. Sin embargo, la implementación de estas tecnologías requiere plataformas accesibles y adaptadas al contexto nacional. En este escenario, se plantea el desarrollo de AgroPredict, una solución tecnológica que busca combinar información climática, datos productivos y herramientas predictivas en un entorno web de fácil acceso, orientado a mejorar la planificación y sostenibilidad en los predios frutales chilenos.

1.2 Definición del problema

En la actualidad, una gran parte de los productores frutícolas en Chile toma decisiones sobre la planificación y manejo de sus predios basándose en la experiencia personal y en registros manuales. La falta de herramientas tecnológicas que integren información climática, productiva y predictiva provoca una brecha significativa en la capacidad de anticipar rendimientos, afectando tanto la eficiencia en la producción como la gestión de recursos.

La ausencia de sistemas centralizados genera que los datos de campo, como condiciones meteorológicas, estados fenológicos o históricos de producción, se almacenen de forma dispersa, dificultando su análisis y uso posterior. Además, muchos agricultores carecen de soluciones digitales accesibles que les permitan visualizar esta información de manera clara o generar predicciones personalizadas.

A nivel técnico, el problema también se manifiesta en la falta de una infraestructura backend robusta que gestione usuarios, datos y servicios externos de manera segura y escalable. Sin una base estructurada que permita conectar los diferentes módulos (usuarios, clima, y predicciones) resulta complejo desarrollar una plataforma confiable que pueda integrarse con modelos de inteligencia artificial en el futuro.

En consecuencia, la problemática principal radica en la inexistencia de una herramienta integral que permita a los agricultores acceder, gestionar y analizar información clave para predecir la productividad de sus cultivos de forma precisa, automatizada y adaptada al contexto agrícola chileno.

1.3 Breve descripción sobre la propuesta de solución

La solución propuesta para AgroPredict consiste en una plataforma web orientada a la predicción de productividad en cultivos frutales, cuyo propósito es mejorar la toma de decisiones en predios agrícolas mediante el uso de datos climáticos, históricos y productivos. La plataforma busca facilitar la planificación y gestión agrícola, reduciendo la incertidumbre en torno a los rendimientos esperados.

Para cumplir este objetivo, la solución contempla dos componentes principales. Por una parte, una base de datos relacional que permite gestionar la información del sistema, como usuarios, predicciones e historial. Por otra, un modelo predictivo basado en redes neuronales LSTM, encargado de procesar los datos históricos y climáticos para estimar la producción futura. Ambos componentes son integrados por el backend, permitiendo entregar una solución unificada al usuario.

La arquitectura propuesta considera una división modular, donde el backend, desarrollado en Django, actúa como el núcleo funcional del sistema. Este componente se encarga de gestionar usuarios, centralizar la información, integrar servicios externos y servir de base para la conexión con modelos predictivos desarrollados en Python.

Las principales responsabilidades del backend incluyen:

- Gestión de usuarios y autenticación segura, garantizando accesos personalizados y protección de datos.
- Integración de datos climáticos en tiempo real mediante la API de AccuWeather.
- Integración del modelo predictivo al flujo de la plataforma, asegurando su operación dentro del backend.
- Evaluación de un microservicio con FastAPI, orientado a alojar el modelo de predicción definitivo de forma independiente y escalable.

Límites del proyecto:

- Incluido en el desarrollo:
 - Implementación del backend en Django con los módulos de autenticación, gestión de predicciones e integración de la API AccuWeather.
 - Diseño de endpoints RESTful para comunicación con el frontend y servicios externos.
 - Integración y prueba de modelos predictivos básicos en el entorno backend.
- Fuera del alcance del desarrollo (realizados por otros integrantes del proyecto):
 - Diseño e implementación del frontend
 - Entrenamiento y validación del modelo predictivo principal.

Este enfoque busca entregar un backend sólido, escalable y seguro, que sirva como pilar técnico para la integración final de AgroPredict, permitiendo conectar información climática, datos productivos y predicciones en una sola plataforma.

1.4 Objetivos Generales y Específicos del Proyecto

El presente proyecto tiene como objetivo desarrollar una plataforma web denominada AgroPredict, destinada a apoyar la toma de decisiones de agricultores, productores frutícolas y profesionales del sector agrícola mediante la integración de información climática, productiva y predictiva. La propuesta busca proporcionar una herramienta moderna y accesible que permita predecir la productividad de árboles frutales en predios agrícolas, mejorando la planificación y gestión de recursos en el sector.

Para alcanzar este propósito general, se han definido los siguientes objetivos específicos, enfocados principalmente en la implementación del backend de la plataforma:

- Diseñar e implementar el backend del sistema en Django, estableciendo la estructura base para la comunicación con el frontend y los servicios externos.
- Configurar un sistema de autenticación y gestión de usuarios, que permita accesos seguros y personalizados según el tipo de usuario.
- Desarrollar la lógica de predicciones personalizadas, vinculando cada usuario con sus respectivos datos y resultados.
- Integrar la API AccuWeather para incorporar información climática actualizada y relevante en el proceso de predicción.
- Integrar la API de OpenAI que permita implementar un chat, al que el cliente le pueda hacer consultas.
- Probar modelos predictivos iniciales dentro del entorno backend, validando su funcionamiento y compatibilidad con la arquitectura del sistema.
- Explorar la factibilidad de integrar un microservicio con FastAPI, que aloje el modelo predictivo principal de forma independiente y escalable.
- Asegurar la integridad y coherencia de los datos, implementando mecanismos de validación y control de errores.
- Documentar el proceso de desarrollo y pruebas, dejando lineamientos claros para la futura integración completa del modelo predictivo.

Estos objetivos buscan garantizar que el sistema cuente con una base técnica sólida, modular y segura, capaz de escalar en futuras etapas hacia una plataforma predictiva completamente integrada y funcional.

1.5 Justificación de proyecto

El desarrollo de la plataforma AgroPredict responde a una necesidad concreta del sector agrícola chileno: contar con herramientas digitales que permitan anticipar la productividad de los cultivos frutales y optimizar el uso de los recursos disponibles. En un contexto donde las condiciones climáticas son cada vez más variables y la información suele manejarse de forma fragmentada, disponer de un sistema que integre datos climáticos, históricos y productivos se vuelve una ventaja estratégica para los agricultores.

Desde el punto de vista técnico, el proyecto se justifica por la necesidad de establecer una infraestructura backend sólida y escalable, capaz de centralizar la información, gestionar usuarios y comunicarse de manera eficiente con servicios externos. El uso de frameworks modernos como Django garantiza un entorno de desarrollo seguro y mantenible, mientras que la posible incorporación de FastAPI permitirá implementar un microservicio dedicado a los modelos predictivos, favoreciendo la modularidad y el rendimiento.

En términos operativos y económicos, la digitalización de los procesos agrícolas puede reducir pérdidas y mejorar la planificación productiva, permitiendo una gestión más eficiente del tiempo y de los insumos. Asimismo, el acceso a predicciones personalizadas contribuye a una toma de decisiones basada en evidencia, algo aún poco frecuente en el ámbito frutícola.

Finalmente, este proyecto tiene una relevancia social y ambiental significativa, ya que promueve la adopción de tecnologías accesibles que fortalecen la agricultura de precisión, mejorando la sostenibilidad del sector y fomentando la innovación en uno de los pilares productivos más importantes de Chile.

1.6 Metodología

El desarrollo del proyecto se abordará mediante un enfoque iterativo y flexible, que permita construir, evaluar y mejorar continuamente las funcionalidades del backend de AgroPredict. Considerando que se trata de un sistema en evolución, donde los requerimientos pueden ajustarse según las pruebas y resultados obtenidos, se optó por aplicar la metodología ágil Scrum como marco de trabajo principal.

Scrum facilita la planificación incremental mediante sprints o ciclos cortos de desarrollo, en los que se implementan y prueban componentes específicos del sistema, como la gestión de usuarios, la integración de la API de AccuWeather o el flujo de predicciones personalizadas. Cada ciclo comienza con una planificación detallada, donde se priorizan las tareas, y culmina con una revisión de resultados y una retrospectiva, en la que se identifican oportunidades de mejora.

Durante el desarrollo, se realizarán reuniones diarias breves para evaluar avances y resolver obstáculos técnicos. Este proceso permite que el backend evolucione de manera controlada y que las funcionalidades se integren gradualmente, manteniendo siempre un producto funcional al cierre de cada sprint.

Para garantizar la calidad del software, se emplearán herramientas como Postman para la verificación de endpoints, y pytest para pruebas unitarias y de integración. Asimismo, se documentarán los avances de cada sprint, asegurando la trazabilidad del desarrollo y la coherencia del sistema.

Este enfoque metodológico asegura una implementación progresiva, verificable y alineada con los objetivos del proyecto, resultando en un backend seguro, modular y escalable, preparado para la integración futura de modelos predictivos avanzados.

1.7 Breve descripción de la organización del informe en capítulos

El presente documento está estructurado en capítulos que presentan de manera ordenada el desarrollo del proyecto AgroPredict, desde su planteamiento inicial hasta las conclusiones. A continuación, se describe brevemente el contenido de cada capítulo:

- **Capítulo 1: Introducción**
En este capítulo se contextualiza el proyecto dentro del ámbito agrícola chileno, abordando los antecedentes, la definición del problema y la propuesta de solución. Además, se establecen los objetivos generales y específicos, la justificación del trabajo y la metodología utilizada en el desarrollo del backend de la plataforma.

- **Capítulo 2: Marco Teórico**
Este capítulo reúne los fundamentos técnicos y teóricos que sustentan el desarrollo del sistema. Se analizan las tecnologías seleccionadas, los frameworks utilizados (principalmente Django y FastAPI), los principios del desarrollo backend y la integración de APIs externas. También se abordan las bases conceptuales de la arquitectura web empleada.
- **Capítulo 3: Diseño e Implementación del Backend**
Aquí se detallan las etapas de construcción del sistema, incluyendo la estructura de la arquitectura, los módulos desarrollados, la gestión de usuarios, la integración de servicios externos y las pruebas de validación realizadas durante el proceso de desarrollo.
- **Capítulo 4: Conclusiones y Trabajo Futuro**
Este capítulo presenta las conclusiones generales del proyecto, los resultados obtenidos y las principales dificultades enfrentadas. Finalmente, se proponen mejoras y líneas de trabajo futuras orientadas a la integración completa del modelo predictivo en la plataforma AgroPredict.

2 Marco Teórico

En este capítulo se presentan los fundamentos teóricos que sustentan el desarrollo del backend de la plataforma AgroPredict, abordando las bases conceptuales y tecnológicas que orientan su implementación. El propósito de este marco es establecer una visión general sobre las arquitecturas de software utilizadas, el rol del backend en los sistemas web modernos y las herramientas seleccionadas para su desarrollo.

La evolución de las arquitecturas de software ha permitido incorporar nuevos enfoques, como las arquitecturas modulares y orientadas a servicios, ampliando las alternativas disponibles para el desarrollo de aplicaciones. Si bien las arquitecturas monolíticas continúan siendo una opción válida en muchos proyectos, estos nuevos enfoques privilegian aspectos como la escalabilidad, la modularidad y el desacoplamiento de componentes. En este contexto, el backend cumple una función central, al encargarse de procesar la lógica del sistema, gestionar la información y permitir la comunicación con servicios externos mediante interfaces bien definidas.

Para el caso de AgroPredict, el backend se desarrolla utilizando el framework Django, seleccionado por su capacidad para estructurar aplicaciones web seguras, mantener una organización clara del código y facilitar la gestión de bases de datos relacionales. Asimismo, el diseño de la solución contempla la posibilidad de incorporar mecanismos que favorezcan la escalabilidad y la evolución del sistema en futuras etapas de desarrollo.

Asimismo, se utilizan APIs RESTful como medio de comunicación entre los componentes del sistema, garantizando interoperabilidad, eficiencia y facilidad de mantenimiento. Junto con ello, se aplican metodologías ágiles, específicamente Scrum, que permiten un desarrollo iterativo, con entregas incrementales y retroalimentación continua.

Este marco teórico proporciona las bases necesarias para comprender las decisiones técnicas y metodológicas adoptadas durante el desarrollo del backend, sirviendo como referencia para el diseño e implementación detallada que se aborda en los capítulos siguientes.

2.1 Fundamentos del desarrollo backend

El desarrollo backend constituye un componente esencial en los sistemas de software modernos, ya que se encarga de procesar la lógica de negocio, administrar los datos y coordinar la comunicación entre las distintas partes que componen una aplicación. A diferencia del frontend, orientado a la interacción visual con el usuario, el backend trabaja en el servidor, asegurando el funcionamiento interno del sistema y la integridad de la información procesada.

En el caso de AgroPredict, el backend cumple la función de gestionar usuarios, almacenar registros de predicciones y conectar la plataforma con fuentes de datos externas, como la API de AccuWeather, que proporciona información climática relevante para las estimaciones agrícolas. Este componente también será responsable de establecer la comunicación con el modelo predictivo de productividad frutal, ya sea de forma directa o mediante un microservicio.

La adopción de un backend desarrollado con Django responde a las necesidades específicas del proyecto, particularmente la integración del modelo predictivo, el manejo de datos y la comunicación con servicios externos.

En AgroPredict se optó por una arquitectura monolítica, debido a su simplicidad de implementación, facilidad de mantenimiento y adecuada integración con los distintos componentes del sistema. No obstante, la arquitectura fue diseñada considerando una futura evolución hacia un enfoque basado en microservicios, permitiendo desacoplar componentes como el modelo predictivo cuando el crecimiento de la aplicación o los requerimientos del sistema lo hagan necesario.

El backend de AgroPredict se desarrolla sobre el framework Django, que ofrece una estructura modular, herramientas de seguridad integradas y un ORM que facilita la gestión de bases de datos. Además, se contempla la posibilidad de utilizar FastAPI como microservicio independiente para alojar el modelo predictivo, optimizando el procesamiento y favoreciendo la escalabilidad del sistema.

Desde el punto de vista metodológico, el desarrollo sigue un enfoque ágil basado en Scrum, lo que permite organizar el trabajo en sprints cortos y entregar avances funcionales de forma iterativa. Cada iteración contempla fases de planificación, desarrollo, pruebas y revisión, fomentando la retroalimentación constante y la mejora continua.

En el plano técnico, se emplean herramientas como Postman para la verificación de endpoints, pytest para las pruebas unitarias, y SQLite como base de datos inicial, con la posibilidad de migrar a PostgreSQL en versiones futuras. Este conjunto tecnológico permite garantizar la estabilidad, seguridad y eficiencia del backend, estableciendo una base sólida para la futura integración del modelo de predicción de productividad.

En conjunto, este enfoque combina principios de modularidad, metodologías ágiles y tecnologías modernas, asegurando que el backend de AgroPredict sea adaptable, mantenible y preparado para responder a las necesidades futuras del sector agrícola.

2.2 Arquitectura de software

La arquitectura definida para el desarrollo del backend de AgroPredict fue diseñada como una solución personalizada que prioriza la eficiencia, la modularidad y la escalabilidad. El objetivo principal es permitir la gestión adecuada de las peticiones REST, además de contemplar la futura incorporación de servicios en tiempo real mediante tecnologías como WebSockets.

Si bien el sistema no se adhiere estrictamente a un único patrón arquitectónico formal, su estructura se basa en la arquitectura MTV proporcionada por Django e integra además patrones de diseño que favorecen la organización del código y el mantenimiento a largo plazo.

- Patrón Facade (Fachada):

Este patrón se aplica para reducir la complejidad de las interacciones entre los distintos módulos del sistema. A través de la creación de interfaces simplificadas, se facilita la comunicación entre los componentes encargados de la lógica de negocio, la gestión de datos y los servicios externos (como la API de AccuWeather o el modelo predictivo). De esta forma, se logra un sistema más ordenado y menos propenso a errores por dependencias internas.

- Patrón MTV (Model-Template-View):

Este patrón corresponde a la arquitectura utilizada por el framework Django y organiza la aplicación en tres componentes principales: Model, encargado de representar y gestionar los datos; View, responsable de procesar las solicitudes y contener la lógica de negocio; y Template, destinado a la presentación de la información al usuario. Esta separación de responsabilidades facilita el mantenimiento del sistema, mejora la organización del código y permite desarrollar cada componente de forma independiente.

La arquitectura también mantiene compatibilidad con una estructura cliente-servidor tradicional, en la cual el backend actúa como punto central de procesamiento de datos, gestionando solicitudes provenientes de las interfaces web y móviles. Este diseño no solo optimiza la respuesta del sistema, sino que además facilita la incorporación de nuevos servicios en fases posteriores del proyecto.

Finalmente, se proyecta la incorporación de módulos de comunicación en tiempo real que complementen las peticiones REST actuales, donde en futuras versiones podría incorporarse la lógica del patrón Observer para gestionar eventos y notificaciones de forma desacoplada entre los distintos componentes del sistema. Esta flexibilidad permitirá a AgroPredict evolucionar sin necesidad de rediseñar por completo su arquitectura base, garantizando así la continuidad y la eficiencia del sistema a largo plazo.

2.3 Tecnologías y frameworks utilizados

El desarrollo del sistema backend de AgroPredict se apoya en un conjunto de tecnologías modernas seleccionadas por su fiabilidad, facilidad de integración y adaptabilidad a futuras expansiones. Estas herramientas garantizan que la plataforma sea segura, eficiente y capaz de escalar a medida que el proyecto evoluciona.

El framework principal utilizado es Django, desarrollado en Python, el cual proporciona una estructura sólida para el desarrollo de aplicaciones web. Django destaca por su enfoque modular, que permite una clara separación entre la lógica del negocio, la capa de presentación y la gestión de datos. Además, incluye componentes integrados para la autenticación de usuarios, la administración de permisos y la protección frente a vulnerabilidades comunes como la inyección SQL o los ataques XSS.

El lenguaje de programación Python fue elegido por su versatilidad y amplia adopción en el ámbito del desarrollo web y la ciencia de datos. Su sintaxis clara y su extenso ecosistema de librerías facilitan la futura integración de modelos de aprendizaje automático, una parte esencial del proyecto orientada a la predicción de productividad de árboles frutales.

Para la gestión de datos, se utiliza SQLite durante las etapas iniciales del desarrollo por su simplicidad y bajo costo de implementación. No obstante, se proyecta la migración a PostgreSQL en una fase posterior, dado su mejor rendimiento y capacidad para manejar consultas más complejas y volúmenes de datos mayores.

En cuanto a la obtención de información externa, el sistema se integra con la API de AccuWeather, que proporciona datos meteorológicos actualizados. Esta integración se realiza mediante APIs RESTful, que permiten una comunicación eficiente y estandarizada entre el backend y los servicios externos, garantizando la interoperabilidad del sistema.

Además del consumo de información meteorológica, el sistema incorpora una API de inteligencia artificial basada en los servicios de OpenAI, destinada a habilitar un módulo de asistencia conversacional para los usuarios. Esta integración permite ofrecer un chatbot capaz de responder consultas, entregar recomendaciones y apoyar al usuario dentro de la plataforma. La comunicación con OpenAI se implementa igualmente mediante APIs RESTful, lo que facilita un intercambio de datos seguro y estandarizado, manteniendo la coherencia tecnológica con el resto de los servicios externos utilizados por el sistema.

En materia de seguridad, el backend aprovecha los mecanismos nativos de autenticación y autorización de Django, permitiendo gestionar sesiones y restringir el acceso a ciertas funcionalidades según los roles de usuario. Además, se contempla la implementación futura de JSON Web Tokens (JWT) para fortalecer el control de acceso y facilitar la comunicación segura con otros servicios o microservicios.

Para verificar el correcto funcionamiento del sistema, se utilizan herramientas como Postman, que permite realizar pruebas de endpoints y validar la comunicación entre el cliente y el servidor, y pytest, empleado para realizar pruebas unitarias básicas y asegurar la consistencia de las funciones principales del backend.

Finalmente, aunque Docker aún no se ha implementado en el entorno actual, se planea su uso en etapas futuras del proyecto para el despliegue del sistema en contenedores. Esta tecnología permitirá asegurar la consistencia entre los entornos de desarrollo,

prueba y producción, mejorando la portabilidad y simplificando la administración de dependencias.

En conjunto, el uso de Django, Python, PostgreSQL, AccuWeather y OpenAI API, Postman y pytest proporciona una base tecnológica sólida y coherente con los objetivos del proyecto, asegurando que AgroPredict cuente con un backend escalable, seguro y preparado para su futura integración con modelos predictivos y servicios externos.

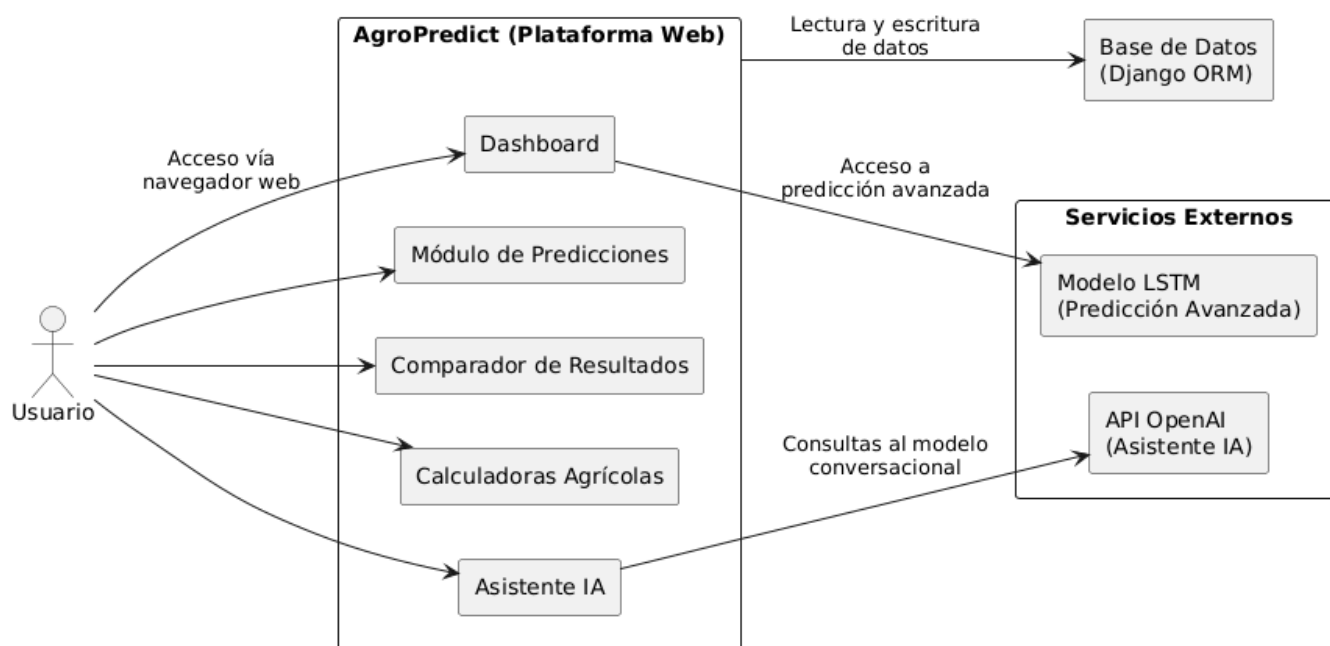
3 Diseño e Implementación

3.1 Visión General de la Solución

AgroPredict fue diseñada como una plataforma web orientada a apoyar la toma de decisiones agrícolas mediante la generación de predicciones productivas, económicas y de consumo hídrico. La solución se desarrolló completamente sobre Django, permitiendo integrar en un solo entorno la capa de datos, la lógica de negocio y las interfaces web. A través de este enfoque, el sistema ofrece distintas herramientas, como creación de predicciones, comparaciones, calculadoras y un asistente IA; que pueden utilizarse de manera flexible según las necesidades del agricultor.

El objetivo del diseño fue construir una aplicación modular, extensible y fácil de mantener, donde cada componente cumpla un rol claro dentro del flujo general del sistema. Para ello, se definió una estructura basada en aplicaciones internas de Django (apps), separación lógica de views y modelos, y el uso de variables de entorno para asegurar las integraciones con servicios externos como AccuWeather y OpenAI. Si bien el proyecto opera actualmente sobre SQLite para fines académicos, su diseño permite una futura migración hacia motores más robustos como PostgreSQL sin cambios significativos en la lógica de negocio.

La Figura 1 presenta una visión general de la arquitectura de AgroPredict, mostrando la interacción entre el usuario, los principales módulos de la plataforma, la base de datos y los servicios externos integrados. En ella se observa cómo el backend centraliza el acceso a la información y coordina la comunicación con el modelo LSTM y la API de OpenAI, permitiendo entregar las distintas funcionalidades de la aplicación de forma integrada.

**Figura 1.** Visión General de la Solución

Fuente: Elaboración propia

3.2 Arquitectura del Sistema

La arquitectura de AgroPredict se basa en un único backend Django que integra todas las capas del sistema, desde la presentación hasta el acceso a datos. Django opera bajo la arquitectura MTV (Model-Template-View), donde los modelos representan la estructura de datos, las vistas contienen la lógica de negocio y los templates generan las interfaces web. Este enfoque permite mantener una separación clara entre responsabilidades, a la vez que facilita incorporar nuevas funcionalidades sin alterar la base del sistema.

Desde una perspectiva funcional, la arquitectura está compuesta por los siguientes elementos:

- a) Capa de presentación (frontend en Django):
Generada mediante templates HTML integrados en Django. Aquí se construyen las vistas del Dashboard, Nueva Predicción, Historial, Comparar, Calculadoras y el Asistente IA.
El frontend y el backend se ejecutan en el mismo servidor, lo que simplifica el despliegue y la comunicación interna del sistema.
- b) Lógica de negocio y procesamiento interno:
Implementada en las views y servicios de Django. Aquí se ejecutan:
 - Cálculos de producción
 - Análisis económico
 - Cálculo de consumo de agua

- Integración con el modelo LSTM (Long Short-Term Memory)
 - Control de credenciales y sesiones
 - Peticiones a APIs externas
- c) Capa de datos y ORM (Object-Relational Mapping) de Django
Django gestiona una base de datos relacional donde se almacenan predicciones, usuarios, tipos de árbol, comunas, datos climáticos y análisis.
La ORM permite trabajar con datos mediante modelos Python en lugar de SQL, facilitando la mantenibilidad y escalabilidad del sistema.
- d) Servicios externos integrados
- AccuWeather API: entrega datos climáticos por comuna.
 - OpenAI API: habilita el asistente conversacional del sistema
- Ambos servicios se consumen mediante peticiones HTTP desde Django utilizando llaves de acceso en variables de entorno.
- e) Módulo LSTM (externo):
El modelo LSTM —desarrollado por un integrante del equipo— se encuentra desacoplado del flujo principal y es consultado desde el Dashboard como herramienta avanzada.
En trabajo futuro se proyecta convertirlo en un microservicio independiente mediante FastAPI.

A continuación, en la figura 2, se presenta arquitectura general implementada en AgroPredict, donde se observa la interacción entre los principales componentes del sistema. El diagrama muestra cómo el frontend y el backend, ambos desarrollados en Django, se comunican para procesar las solicitudes del usuario, además de la integración con la base de datos mediante el ORM de Django, el modelo predictivo LSTM y los servicios externos utilizados por la plataforma, como las APIs de AccuWeather y OpenAI. Esta representación permite comprender el flujo general de información y la relación entre los distintos componentes que conforman la solución.

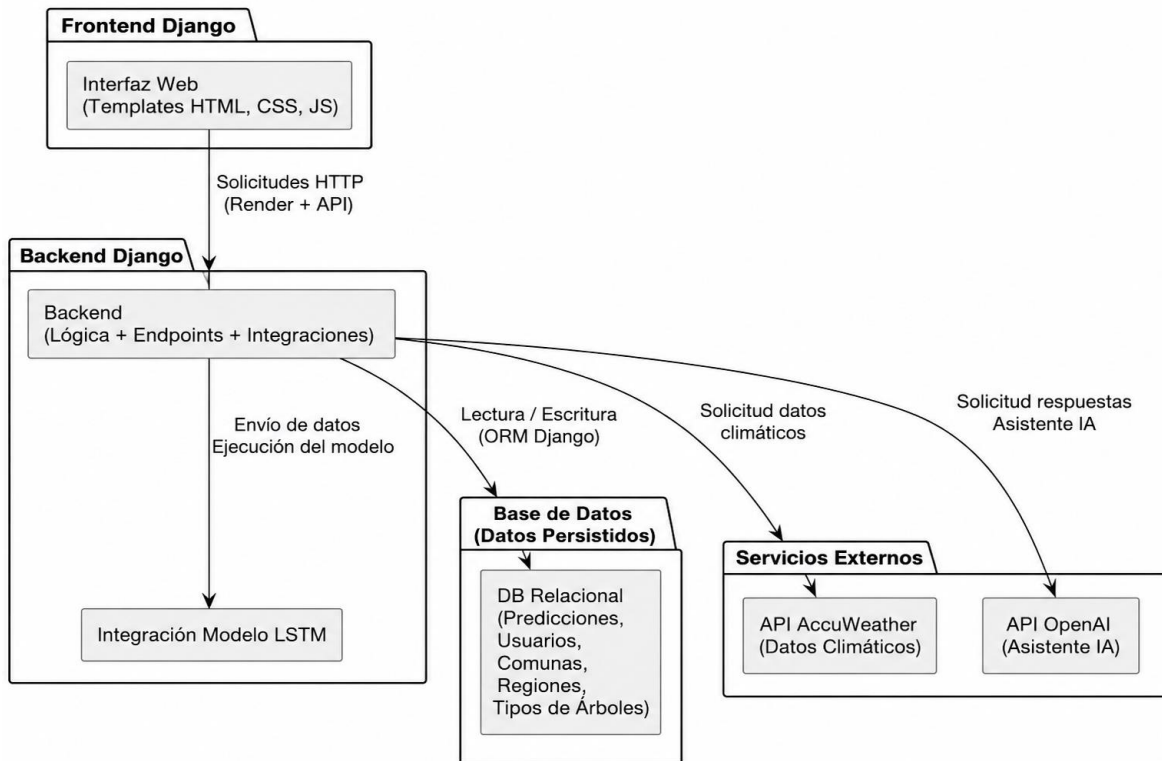


Figura 2: Arquitectura general del sistema AgroPredict

Fuente: Elaboración propia

3.3 Diseño de la Base de Datos

El diseño de la base de datos de AgroPredict se desarrolló utilizando la ORM (Object-Relational Mapping) de Django, lo que permitió estructurar el modelo de datos mediante clases Python, las cuales posteriormente son traducidas por la ORM a la estructura correspondiente en la base de datos. Esta aproximación facilita la mantenibilidad del sistema, asegura consistencia en los tipos de datos y simplifica la portabilidad hacia otros motores de base de datos como PostgreSQL. Aunque el proyecto se ejecuta actualmente sobre SQLite por razones de simplicidad académica, el diseño permite una migración sin modificaciones significativas en la lógica del sistema.

La base de datos se estructura en torno a un conjunto de entidades que representan los elementos esenciales del dominio agrícola: regiones, comunas, tipos de árboles, usuarios y predicciones. Estas tablas se complementan con entidades destinadas al almacenamiento de datos climáticos, información histórica del predio y análisis asociados a las predicciones, permitiendo mantener un registro de la información utilizada por la plataforma. El modelo no incluye tablas asociadas al módulo LSTM, ya que dicho componente funciona como un servicio externo y sus resultados no se almacenan en la base de datos relacional.

3.3.1 Modelo de Datos General

El modelo de datos general de AgroPredict se representa mediante un diagrama de clases UML, el cual describe las principales entidades del sistema, sus atributos y las relaciones existentes entre ellas. Estas clases fueron definidas en Python utilizando el ORM de Django, herramienta que posteriormente se encarga de convertirlas en su equivalente dentro de una base de datos relacional. Este proceso de conversión se explica con mayor detalle en la Sección 3.3.2. Las principales relaciones del modelo son las siguientes:

- a) Región → Comuna(1:N):
Cada región contiene múltiples comunas.
Cada comuna pertenece a una sola región.
Esta relación permite asociar cada predicción a una ubicación geográfica específica.
- b) Usuario → Predicción(1:N):
Un usuario registrado (agricultor o administrador de la plataforma) puede generar múltiples predicciones dentro del sistema.
Las predicciones quedan asociadas a la cuenta del usuario, lo que permite mantener un historial personal.
- c) Tipo de Árbol → Predicción (1:N):
Cada predicción se realiza sobre un tipo de árbol determinado (por ejemplo, palto, manzano o naranjo).
Esta entidad almacena:
 - Rendimiento base por hectárea (toneladas por hectárea).
 - Precio promedio de venta por tonelada.Estos valores sirven como parámetros base para los cálculos productivos y económicos.
- d) Predicción → Datos Predio Histórico (1:N):
Permite almacenar años previos y rendimientos reales asociados a un predio, enriqueciendo el sistema para futuros modelos o análisis longitudinales.
- e) Predicción → Análisis de Predicción (1:1)
Cada predicción puede asociarse a un análisis complementario que clasifica la rentabilidad y genera recomendaciones en función de los resultados del análisis económico proyectado a cinco años.
Esto se almacena en una entidad independiente para mantener el modelo modular y registrar la fecha en que se realizó el análisis.

Con el propósito de visualizar la estructura del modelo de datos descrito anteriormente, a continuación, se presenta el diagrama de clases UML correspondiente a las principales entidades de AgroPredict. En él se representan las clases definidas mediante el ORM de Django, sus atributos más relevantes y las relaciones existentes entre ellas.

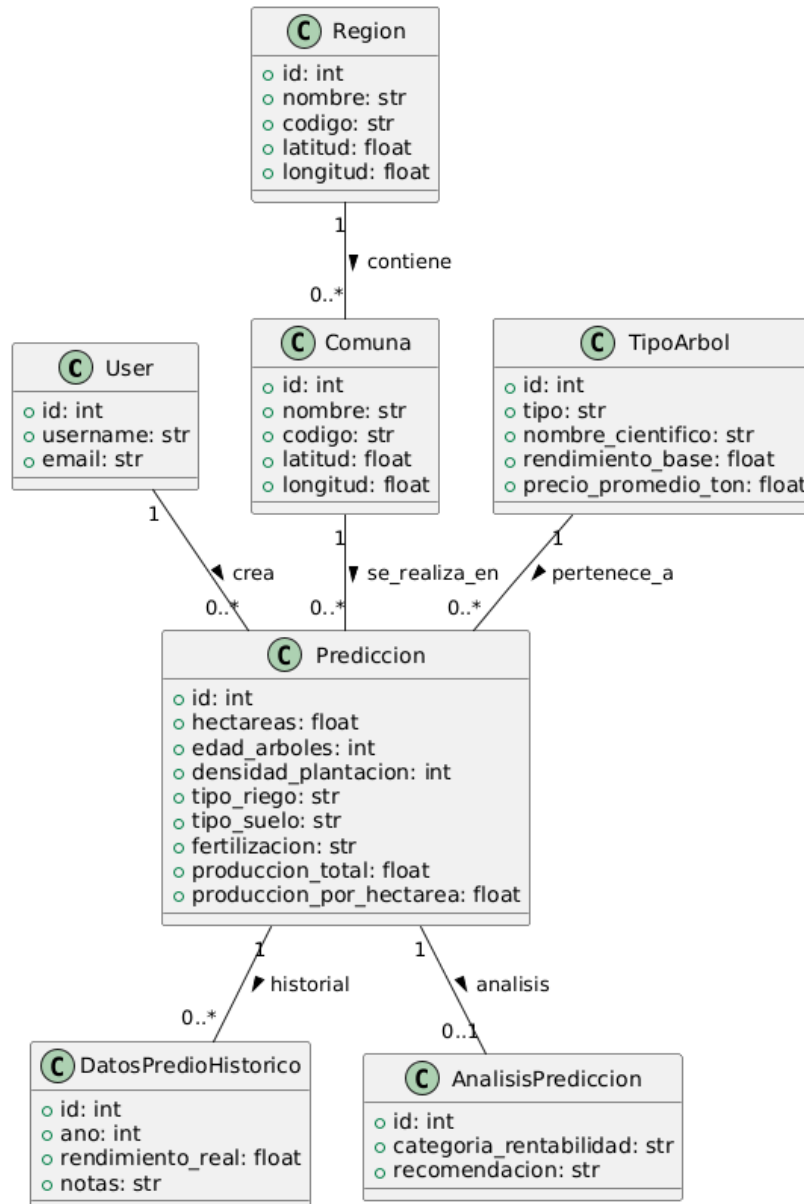


Figura 3: Diagrama de clases de las entidades principales del sistema.

Fuente: Elaboración Propia

Con el fin de complementar el diagrama de clases, a continuación, se presenta la Figura 4, que describe brevemente los principales atributos del modelo de datos.

Entidad	Atributo	Descripción
Todas	id	Identificador único generado automáticamente por Django (clave primaria).
Región	nombre	Nombre de la región.
	codigo	Código asociado a la región como identificador adicional.
Comuna	nombre	Nombre de la comuna.
	codigo	Código asociado a la comuna como identificador adicional.
TipoArbol	tipo	Nombre común de la especie (palto, manzano, etc.).
	nombre_cientifico	Nombre científico de la especie.
	rendimiento_base	Rendimiento promedio (t/ha).
	precio_promedio_ton	Precio promedio de venta por tonelada (CLP).
	costo_plantacion_hectarea	Costo estimado de plantación por hectárea.
	costo_mantenimiento_anual	Costo anual de mantenimiento por hectárea.
Predicción	consumo_agua_m3_ton	Consumo estimado de agua (m³/ton).
	hectareas	Superficie del cultivo (ha).
	edad_arboles	Edad promedio de los árboles (años).
	densidad_plantacion	Cantidad de árboles por hectárea.
	produccion_total	Producción total estimada (toneladas).
	produccion_por_hectarea	Producción estimada por hectárea (t/ha).
DatosPredioHistorico	ano	Año del registro histórico.
	rendimiento_real	Producción real registrada.
AnalisisPrediccion	categoria_rentabilidad	Clasificación de la rentabilidad.
	recomendacion	Recomendación generada por el análisis.

Figura 4. Descripción de los principales atributos del modelo de datos.

Fuente: Elaboración Propia

3.3.2 Definición del Modelo mediante la ORM de Django

Una vez definido el modelo de datos general, su implementación se realizó utilizando la ORM (Object-Relational Mapping) de Django. La ORM permite definir la estructura de la base de datos directamente desde clases Python, evitando la necesidad de escribir consultas SQL manuales y proporcionando una capa de abstracción que facilita la mantenibilidad del sistema. Cada clase definida en el archivo models.py corresponde a una tabla en la base de datos, mientras que cada atributo de la clase representa una columna. Las relaciones entre tablas se establecen mediante campos tales como ForeignKey, OneToOneField y ManyToManyField, que Django traduce automáticamente a claves foráneas en el esquema relacional.

A continuación, se presenta un ejemplo representativo del modelo Prediccion, el cual ilustra cómo el modelo de datos descrito anteriormente es implementado mediante la ORM de Django.

```
class Prediccion(models.Model):
    usuario = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
```

```
        related_name='predicciones'
    )
    tipo_arbol = models.ForeignKey(
        TipoArbol,
        on_delete=models.CASCADE
    )
    comuna = models.ForeignKey(
        Comuna,
        on_delete=models.CASCADE
    )

    hectareas = models.FloatField()
    edad_arboles = models.IntegerField()
    densidad_plantacion = models.IntegerField()
```

En este fragmento se observa cómo la ORM implementa las relaciones definidas en el modelo de datos mediante el uso de `ForeignKey`, además de establecer los atributos propios de la entidad `Prediccion` utilizando tipos de datos como `FloatField` e `IntegerField`.

El parámetro `on_delete=models.CASCADE` asegura que, si una entidad referenciada es eliminada, las predicciones asociadas también lo sean, manteniendo la integridad del esquema. Por su parte, el atributo `related_name='predicciones'` habilita consultas inversas como:

```
usuario.predicciones.all()
```

Finalmente, la definición de campos como `FloatField` o `IntegerField` establece los tipos de datos de manera explícita, de modo que la ORM genera automáticamente las sentencias SQL necesarias para la creación y mantenimiento de la tabla correspondiente.

Este enfoque permite que el modelo de datos sea claro, extensible y desacoplado del motor de base de datos, facilitando futuras migraciones hacia sistemas más robustos como PostgreSQL sin necesidad de alterar la lógica interna de la aplicación.

3.3.3 Migraciones y Control de Esquema

Django administra la evolución del esquema de la base de datos mediante un sistema de migraciones, que actúa como un historial estructurado de cambios. Cada vez que se modifica un modelo —ya sea por la creación de una nueva entidad, la adición de un campo o la actualización de una relación— se genera un archivo de migración que describe esa modificación de forma incremental. Esto permite mantener coherencia entre distintos entornos y evita la necesidad de escribir SQL manual.

Las migraciones se generan y aplican mediante los comandos:

```
python manage.py makemigrations
python manage.py migrate
```

El primero detecta cambios en los modelos y crea los archivos correspondientes; el segundo los aplica al motor de base de datos. Este mecanismo facilita el versionamiento del esquema y asegura que la estructura pueda reproducirse de manera consistente a lo largo del desarrollo del proyecto.

3.3.4 Visualización del Modelo en el Administrador Django

El panel de administración de Django constituye una herramienta central para verificar el funcionamiento del modelo de datos. Desde esta interfaz es posible revisar registros, editar información y observar cómo se representan las relaciones entre entidades dentro de la base de datos.

En AgroPredict, el administrador permite visualizar tablas como Región, Comuna, TipoArbol y Predicciones. Esto facilita comprobar que las claves foráneas se comportan correctamente y que los datos asociados a cada modelo se almacenan y consultan como se espera. Además, el panel incorpora funcionalidades de búsqueda y filtros por estado, tipo de árbol, región y fecha de creación, lo que facilita la gestión de un gran volumen de predicciones. Asimismo, sirve como apoyo para pruebas funcionales durante el desarrollo, confirmando que el sistema está interactuando adecuadamente con la base de datos.

☐	ID	TIPO ARBOL	COMUNA	USUARIO	1 ▾	HECTAREAS	PRODUCCIÓN POR HECTÁREA (t/ha)	CONFIABILIDAD	ESTADO	FECHA CREACION	2 ▾
☐	16	Almendro	Melipilla, Region Metropolitana	admin		23,0	1,78	74	Completada	24 de octubre de 2025 a las 01:06	
☐	15	Limonero	Melipilla, Region Metropolitana	admin		123,0	4,92	86	Completada	23 de octubre de 2025 a las 20:54	
☐	14	Cerezo	Las Condes, Region Metropolitana	admin		3,3	6,65	88	Completada	7 de octubre de 2025 a las 20:43	
☐	13	Cerezo	Melipilla, Region Metropolitana	admin		44,0	1,98	72	Completada	7 de octubre de 2025 a las 20:41	
☐	11	Limonero	Maipu, Region Metropolitana	admin		5,5	12,75	90	Completada	7 de octubre de 2025 a las 20:11	

Figura 5: Listado de predicciones en el administrador de Django

Fuente: Captura al panel de administración de Django

☐	NOMBRE	CODIGO
☐	Region Metropolitana	RM
☐	Region de Antofagasta	AN
☐	Region de Arica y Parinacota	AP
☐	Region de Atacama	AT
☐	Region de Aysen	AI
☐	Region de Coquimbo	CO
☐	Region de La Araucania	AR
☐	Region de Los Lagos	LL
☐	Region de Los Rios	LR
☐	Region de Magallanes	MG
☐	Region de OHiggins	OH
☐	Region de Tarapaca	TA
☐	Region de Valparaiso	VA
☐	Region del Biobio	BB
☐	Region del Maule	MA

Figura 6: Listado de regiones en el administrador de Django

Fuente: Captura al panel de administración de Django

Las capturas del panel de administración permiten verificar cómo se almacenan las entidades del sistema y cómo se visualizan sus relaciones. En las imágenes se aprecia el listado de predicciones y la estructura de regiones registradas, lo que confirma que el modelo de datos está funcionando correctamente dentro de la base de datos.

3.3.5 Consideraciones sobre el Motor de Base de Datos

AgroPredict utiliza SQLite como motor de base de datos durante el desarrollo, debido a su facilidad de uso y a que no requiere un servidor dedicado. Este motor es suficiente para entornos académicos y prototipos, permitiendo ejecutar el proyecto con una configuración mínima.

No obstante, para una implementación en producción resultaría más adecuado un motor como PostgreSQL, que ofrece mayor rendimiento ante múltiples usuarios, mejor manejo de transacciones y soporte para tipos de datos avanzados. Gracias a la ORM de Django, la transición entre motores puede realizarse sin cambios significativos en el código, lo que permite que la solución escale de forma natural si se requiere en el futuro.

3.4 Implementación de Funcionalidades del Sistema

Esta sección describe las principales funcionalidades implementadas en la plataforma AgroPredict, detallando su propósito dentro del sistema y la forma en que interactúan con la lógica de negocio y la base de datos. Cada módulo fue desarrollado utilizando el framework Django y se integra de manera coherente dentro del flujo general de la aplicación. Para cada funcionalidad se presentan descripciones, imágenes de la interfaz y, cuando corresponde, fragmentos de código que ilustran su funcionamiento interno. Estas funcionalidades conforman el conjunto de herramientas que permiten al usuario

generar predicciones, consultar resultados, comparar escenarios productivos y acceder a utilidades adicionales de apoyo agrícola

3.4.1 Interfaz Principal y Navegación General

La interfaz principal de AgroPredict se organiza en torno a un dashboard que funciona como punto de entrada para todas las herramientas del sistema. Desde esta vista inicial, el usuario puede acceder de manera rápida a la creación de una nueva predicción, revisar su historial, utilizar las calculadoras agrícolas o interactuar con el asistente de inteligencia artificial. Esta estructura de navegación se diseñó con el objetivo de mantener un flujo simple e intuitivo, permitiendo que el usuario pueda explorar las distintas funciones sin necesidad de recorrer rutas complejas.

El dashboard también presenta un resumen de las actividades recientes del usuario, incluyendo sus últimas predicciones y métricas generales del sistema. Esta información cumple la función de contextualizar el uso de la aplicación, facilitando su retorno a tareas previamente realizadas. La organización visual se apoyó en elementos claros y minimalistas, de manera que las funciones esenciales se mantuvieran siempre accesibles desde la pantalla principal.

A continuación, se presenta la Figura 7, correspondiente al dashboard principal de AgroPredict. Esta interfaz reúne indicadores generales, gráficos y accesos a los distintos módulos del sistema, proporcionando una visión resumida de la información generada por la plataforma.

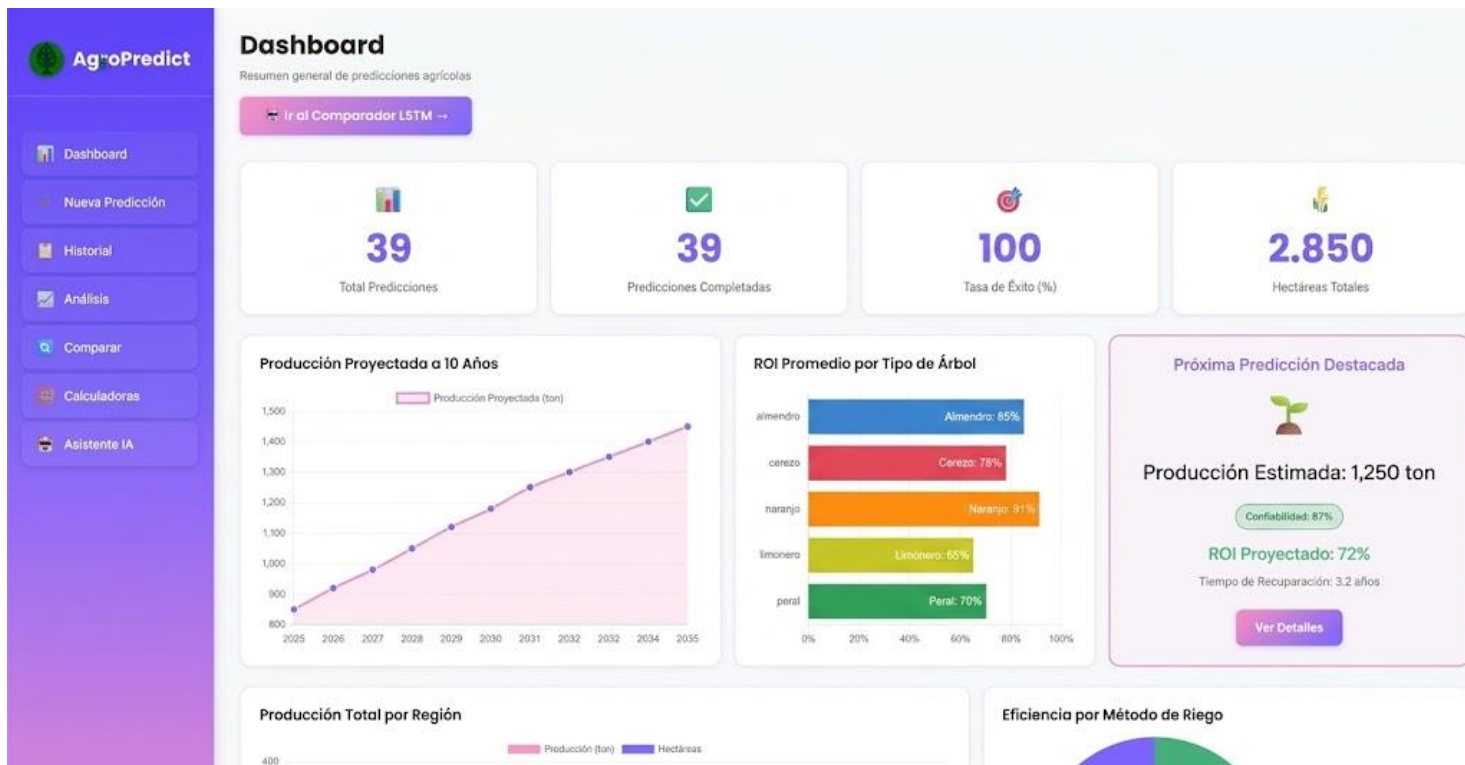


Figura 7: Vista general del dashboard de Agropredict

Fuente: Captura de pantalla de la plataforma Agropredict

En la parte central del dashboard se muestra un gráfico con la proyección de producción total del cultivo, expresada en toneladas, para un horizonte de diez años, obtenida a partir del modelo predictivo integrado en la plataforma.

3.4.2 Integración del Modelo LSTM en el Sistema

El proyecto AgroPredict incorpora un modelo predictivo basado en redes neuronales LSTM, desarrollado por un integrante del equipo, cuyo objetivo es estimar la producción agrícola de paltas a partir de datos históricos. El modelo fue entrenado utilizando datos climáticos de la región, combinados con información histórica de producción, permitiendo analizar series temporales de aproximadamente 10 años hacia atrás y generar estimaciones de producción para los 10 años siguientes, entregando proyecciones de largo plazo.

Desde el punto de vista del sistema, el modelo LSTM forma parte de la arquitectura de AgroPredict y se integra al proceso de predicción mediante un módulo accesible desde el dashboard de la aplicación. Esto permite al usuario consultar estimaciones avanzadas sin interferir con el flujo estándar de creación de predicciones, manteniendo separadas las funcionalidades del sistema y evitando acoplar el uso del modelo a los formularios principales.

El modelo se apoya en un conjunto reducido de archivos persistidos dentro del proyecto. El modelo entrenado (.keras) contiene la red neuronal utilizada para realizar la inferencia y generar las predicciones. Los escaladores de entrada y salida (.pkl) permiten normalizar los datos antes de ser procesados por la red neuronal y desescalar los

resultados obtenidos, de modo que las predicciones puedan presentarse en valores reales e interpretables. Por su parte, el dataset histórico cronológico (.csv) almacena los datos climáticos y productivos utilizados como referencia para la generación de las estimaciones.

Para gestionar estos archivos y facilitar su uso dentro del backend, se implementó una clase de servicio específica (`lstm_predictor.py`). Este archivo actúa como una capa de integración, encargada de centralizar la carga del modelo, los escaladores y los datos históricos, así como de preparar los datos de entrada, ejecutar la inferencia y retornar los resultados al sistema. El servicio recibe los datos ingresados por el usuario, aplica el preprocesamiento necesario, ejecuta el modelo LSTM y entrega las predicciones de forma estructurada al backend de Django.

Gracias a que tanto Django como el modelo LSTM están desarrollados en Python, la integración se realizó de manera directa, incorporando los archivos correspondientes al proyecto y ejecutándolos desde el entorno de la aplicación. Este enfoque permitió una implementación sencilla y mantenible, manteniendo la lógica del modelo encapsulada y evitando que las vistas del sistema interactúen directamente con los detalles internos del LSTM. Asimismo, esta estructura deja abierta la posibilidad de evolucionar hacia una arquitectura más desacoplada, donde el modelo pueda operar como un servicio independiente en el futuro.

3.4.3 Gestión de Predicciones

La gestión de predicciones constituye el núcleo funcional de AgroPredict y corresponde al proceso mediante el cual el usuario ingresa los parámetros de su cultivo para obtener estimaciones productivas, económicas e hídricas. Este módulo combina la interacción directa con el usuario, a través de un formulario web, con la lógica de cálculo implementada en el backend mediante las vistas y modelos de Django.

El flujo se inicia cuando el usuario accede al formulario de “Nueva Predicción” y registra información relevante del predio, como el tipo de árbol, la comuna, las hectáreas disponibles, la edad de los árboles y las condiciones de riego, suelo y fertilización. Una vez enviado el formulario, los datos son procesados por el servidor, donde se aplican validaciones y reglas de negocio para asegurar consistencia y evitar valores fuera de rango.

Posteriormente, el backend ejecuta los cálculos asociados al modelo productivo definido en la aplicación. Entre ellos se encuentran el rendimiento estimado por hectárea, la producción total, el consumo de agua según la especie seleccionada y un análisis económico basado en costos e ingresos proyectados. Estos resultados se almacenan en la base de datos y quedan disponibles para futuras consultas.

Finalmente, el sistema presenta al usuario una visualización de la predicción generada, permitiendo revisar los valores estimados y comprender su impacto productivo. Cada predicción queda registrada en el historial del usuario, lo que facilita comparar decisiones, observar cambios en distintos escenarios y mantener un seguimiento ordenado de su actividad dentro de la plataforma.

Para complementar la explicación anterior, se presenta a continuación un diagrama de flujo que resume las etapas centrales del proceso de predicción dentro del sistema.

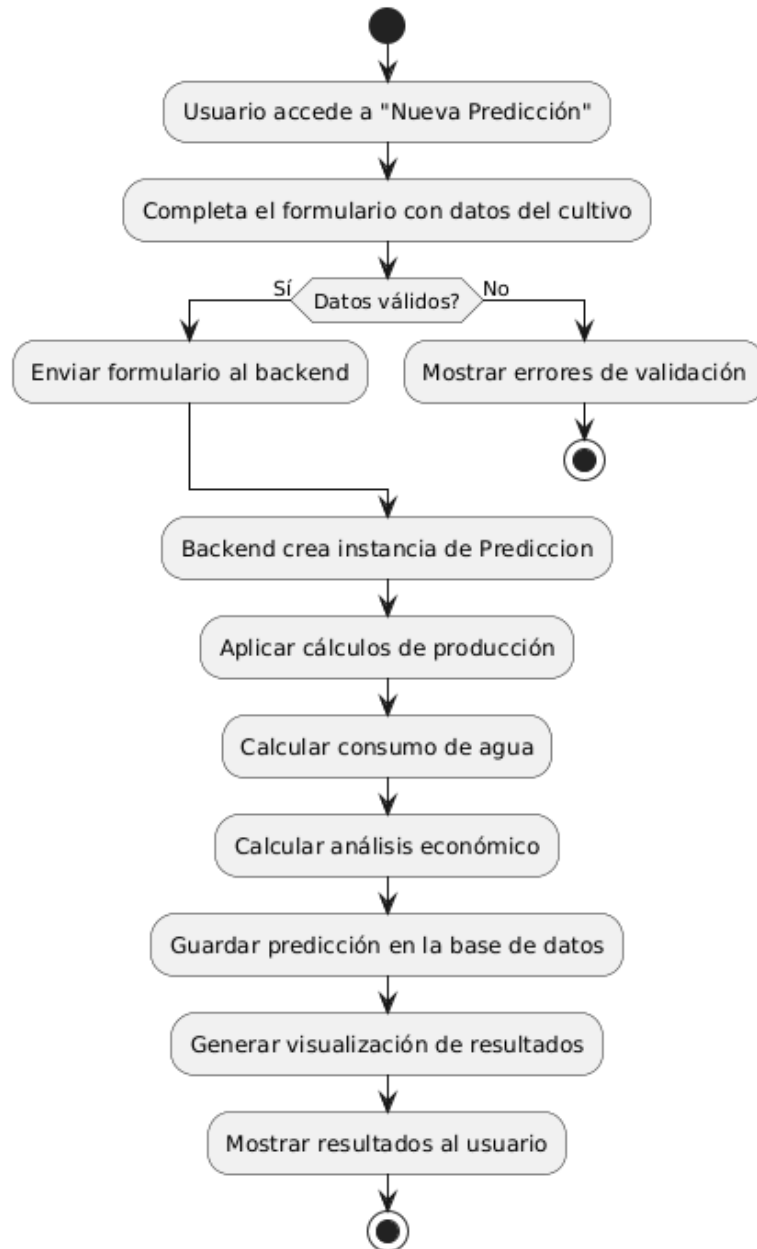


Figura 8: Flujo básico del proceso de predicción en Agropredict

Fuente: Elaboración propia

3.4.4 Herramientas de Apoyo

Las herramientas de apoyo de AgroPredict consisten en una serie de calculadoras que permiten obtener estimaciones rápidas relacionadas con producción, consumo de agua y rentabilidad. Estas funciones se basan principalmente en fórmulas matemáticas simples, diseñadas para entregar resultados inmediatos sin necesidad de generar una predicción completa.

Las fórmulas empleadas se fundamentan en valores promedio y referencias provenientes de estudios científicos y organismos agrícolas reconocidos, lo que otorga validez a los cálculos entregados. Aunque sus resultados no reemplazan un análisis detallado, estas herramientas ofrecen una orientación inicial útil para que los usuarios puedan evaluar escenarios preliminares o complementar sus decisiones productivas.

A continuación, se presenta el código correspondiente a la calculadora de ROI, la cual ejemplifica la implementación de una de las herramientas de apoyo desarrolladas en AgroPredict. Esta calculadora utiliza una fórmula matemática simple para estimar la rentabilidad y el período de recuperación de una inversión agrícola a partir de los datos ingresados por el usuario.

```
def calculadora_roi(request):
    resultado = None
    if request.method == 'POST':
        inversion = float(request.POST.get('inversion', 0))
        beneficio = float(request.POST.get('beneficio_anual', 0))

        roi = (beneficio / inversion) * 100 if inversion > 0 else 0
        payback = inversion / beneficio if beneficio > 0 else None

        resultado = {
            'roi': round(roi, 1),
            'payback': round(payback, 1) if payback else None
        }

    return render(request, 'calculadoras/roi.html', {'resultado':
resultado})
```

El fragmento presentado corresponde a la vista de Django encargada de procesar los datos ingresados por el usuario. A partir de los valores de inversión y beneficio anual, el sistema calcula el ROI y el período de recuperación mediante operaciones matemáticas básicas, para posteriormente enviar los resultados a la interfaz y mostrarlos al usuario. Todas las calculadoras del módulo siguen una lógica similar, adaptando únicamente las fórmulas empleadas según el indicador agrícola correspondiente.

3.4.5 Asistente IA

El asistente conversacional integrado en AgroPredict permite a los usuarios realizar consultas en lenguaje natural con el fin de aclarar dudas, interpretar resultados o solicitar explicaciones relacionadas con los cálculos agrícolas utilizados en la plataforma. Esta herramienta mejora la accesibilidad del sistema al ofrecer respuestas inmediatas y

contextualizadas, especialmente útiles para usuarios que no están familiarizados con conceptos técnicos.

Para habilitar esta funcionalidad fue necesario utilizar una API de inteligencia artificial proporcionada por OpenAI. El sistema envía cada consulta del usuario a esta API junto con los parámetros correspondientes, y el modelo procesa el texto y devuelve una respuesta generada dinámicamente. Con el fin de utilizar este servicio, fue necesario crear una cuenta en OpenAI y obtener una clave de acceso (API Key), la cual se almacena de forma segura mediante variables de entorno dentro de la aplicación.

El uso de esta API implica un consumo de tokens, que representan la cantidad de texto procesado en cada interacción. Tanto el mensaje enviado como la respuesta generada contribuyen al total de tokens utilizados en una consulta. Dado que este consumo está asociado a un costo económico, fue necesario cargar un monto inicial en dólares para habilitar el funcionamiento del asistente durante el desarrollo y las pruebas. Este mecanismo de cobro permite controlar el uso del servicio y garantiza que su integración sea sostenible.

Aunque el asistente no participa en los cálculos internos del sistema, sí cumple un rol complementario al proporcionar apoyo, explicaciones y orientación general al usuario. De este modo, la plataforma combina herramientas matemáticas tradicionales con capacidades modernas de procesamiento de lenguaje natural, ofreciendo una experiencia más completa y flexible.

Para implementar el Asistente IA dentro de AgroPredict fue necesario definir una función encargada de recibir las consultas de los usuarios y comunicarla con el modelo de lenguaje seleccionado. Esta función actúa como punto central de interacción entre la aplicación y el servicio de inteligencia artificial, gestionando tanto la recepción del mensaje como la generación de la respuesta. En la siguiente imagen se muestra el fragmento de código utilizado para realizar esta operación

```
client = OpenAI(api_key=settings.OPENAI_API_KEY)

def ia_consulta(request):
    pregunta = request.GET.get("q") or request.POST.get("q")

    if not pregunta:
        return JsonResponse({"error": "Debe enviar parámetro 'q'."},
                             status=400)

    try:
        chat = client.chat.completions.create(
            model="gpt-4o-mini",
            messages=[{"role": "user", "content": pregunta}]
        )
        respuesta = chat.choices[0].message.content
        return JsonResponse({"respuesta": respuesta})
    except Exception as e:
        return JsonResponse({"error": str(e)}, status=500)
```

El fragmento anterior muestra la función utilizada por AgroPredict para procesar consultas del Asistente IA. En esta implementación, el sistema recibe una pregunta

enviada por el usuario y la envía a un modelo de lenguaje mediante la API de OpenAI, utilizando la clave de acceso configurada en las variables de entorno. Tal como se aprecia en el código, se definió el uso del modelo gpt-4o-mini, seleccionado por su equilibrio entre costo, velocidad y capacidad de respuesta. El modelo genera una salida en lenguaje natural y esta es devuelta al usuario en formato JSON. Esta estructura permite mantener el procesamiento centralizado en el backend y garantiza un control directo sobre el uso de tokens, el manejo de errores y la interacción con el servicio de inteligencia artificial.

3.5 Flujo General de la Aplicación

El flujo general de AgroPredict fue diseñado para ofrecer una navegación intuitiva y permitir que los usuarios accedan libremente a las distintas herramientas disponibles dentro del sistema. Una vez que el usuario inicia sesión, es dirigido al dashboard principal, donde se presentan indicadores generales, accesos rápidos y las últimas predicciones registradas. Desde esta vista central, el usuario puede desplazarse hacia cualquiera de los módulos de la aplicación, manteniendo siempre una estructura de navegación simple y consistente.

El flujo no es lineal ni restringido; cada módulo es independiente y puede utilizarse según la necesidad del usuario. No obstante, existe un camino básico que representa la experiencia típica: el usuario inicia sesión, accede al dashboard, crea o revisa predicciones, utiliza las calculadoras para estimaciones adicionales y, si lo requiere, consulta el Asistente IA para obtener aclaraciones o explicaciones. Este enfoque flexible permite que tanto usuarios experimentados como principiantes interactúen con la plataforma sin barreras ni dependencias innecesarias.

A continuación, se presenta un diagrama que resume el flujo general de navegación dentro de la aplicación.

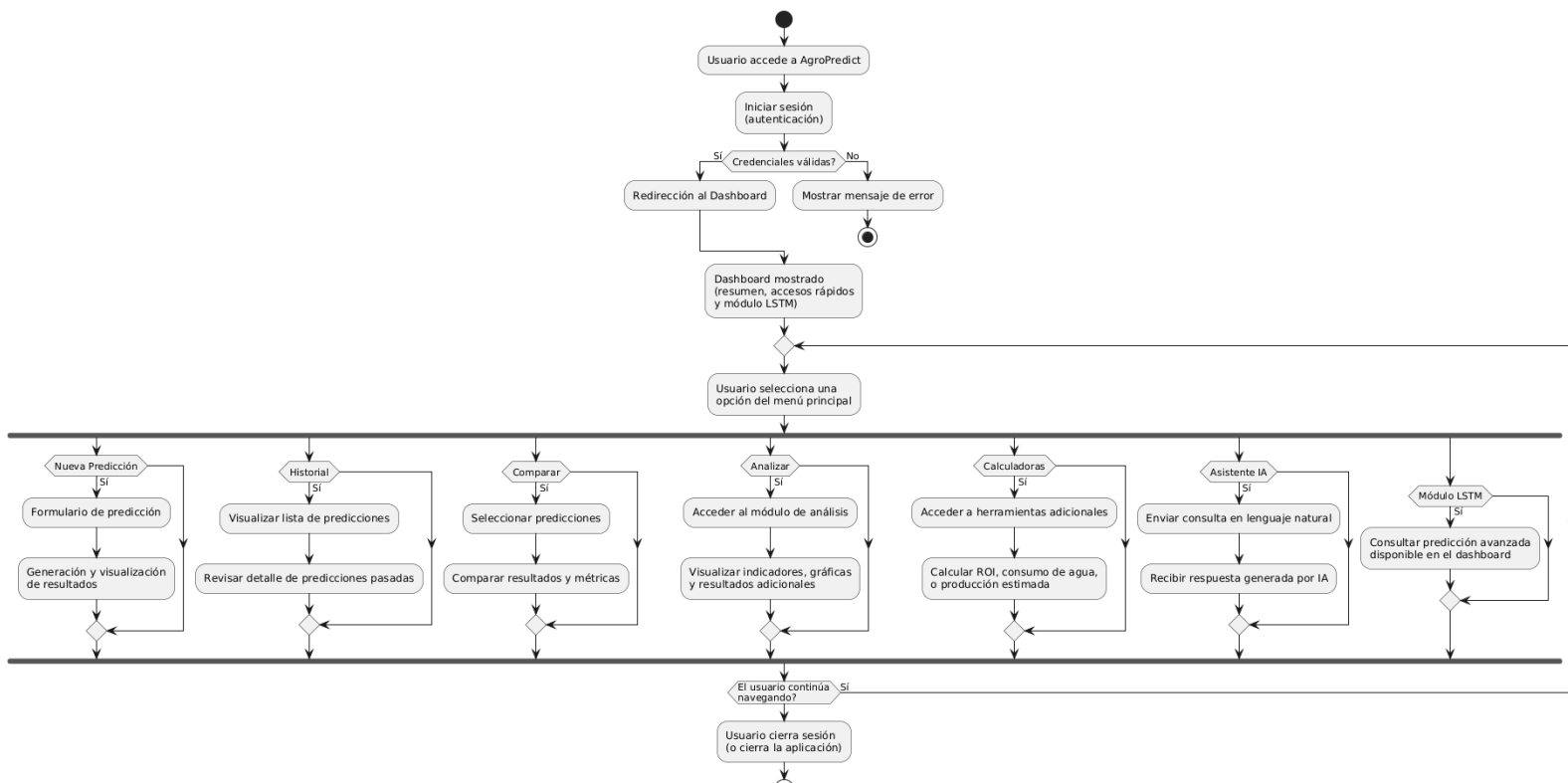


Figura 9: Diagrama de flujo general de Agropredict

Fuente: Elaboración propia

3.6 Decisiones de Diseño y Justificación Técnica

Durante el desarrollo de AgroPredict se tomaron diversas decisiones técnicas orientadas a equilibrar simplicidad, claridad y proyección futura del sistema. Estas decisiones estuvieron condicionadas por el carácter académico del proyecto, el enfoque principalmente backend de la solución y la necesidad de construir una plataforma funcional, extensible y comprensible.

3.6.1 Elección del Framework Django

Durante la etapa inicial del proyecto se evaluaron distintas alternativas para la implementación del backend. En un comienzo, se consideró el uso del framework Laravel, dado que era la tecnología empleada por la empresa colaboradora y contaba con una estructura madura para el desarrollo de aplicaciones web. Sin embargo, tras analizar los requerimientos específicos de AgroPredict, se optó por utilizar Django como framework principal.

Una de las razones fundamentales para esta decisión fue el uso de Python como lenguaje base, lo cual permitió una integración más directa y coherente con los componentes de análisis de datos y las funcionalidades de inteligencia artificial contempladas en el proyecto. Django facilita el desarrollo de aplicaciones orientadas al procesamiento de

información, ofreciendo un ecosistema robusto para el manejo de modelos, cálculos y lógica de negocio.

Adicionalmente, Django proporciona un conjunto de herramientas integradas que aceleran el desarrollo, tales como el sistema de autenticación, el panel de administración y el manejo de formularios y validaciones. Estas características permitieron centrar los esfuerzos en la implementación de las funcionalidades principales del sistema, reduciendo la complejidad asociada a configuraciones externas.

Finalmente, la estructura basada en el patrón Modelo–Vista–Template (MVT) contribuyó a una organización clara del código y a una separación adecuada de responsabilidades, alineándose con los objetivos académicos del proyecto y facilitando su mantenimiento y extensión futura.

3.6.2 Uso de ORM y Base de Datos Relacional

El uso de la ORM de Django permitió definir el modelo de datos directamente en código Python, evitando la escritura manual de consultas SQL y reduciendo errores asociados al manejo directo de la base de datos. Esta abstracción facilitó la creación de relaciones entre entidades, el control de integridad referencial y la evolución progresiva del esquema mediante migraciones.

La elección de una base de datos relacional responde a la naturaleza estructurada de la información manejada por AgroPredict, donde existen relaciones claras entre usuarios, predicciones, comunas, tipos de árboles y análisis asociados. Asimismo, el uso de SQLite durante el desarrollo permitió mantener una configuración simple, sin impedir una migración futura hacia motores más robustos como PostgreSQL.

3.6.3 Rol del Backend dentro de una Solución Integral

El backend de AgroPredict fue concebido como el componente encargado de articular y soportar el funcionamiento global del sistema, integrándose con los distintos módulos desarrollados a lo largo del proyecto. Su rol principal es gestionar la lógica de negocio, el procesamiento de datos y la comunicación entre la interfaz de usuario, los modelos predictivos y los servicios externos utilizados por la plataforma.

El diseño del backend se realizó considerando la coexistencia y complementariedad con otros componentes clave del proyecto, tales como la interfaz de usuario, el modelo de inteligencia artificial y la arquitectura general del sistema. De esta forma, el backend actúa como un punto de coordinación que permite que cada módulo cumpla su función específica de manera coherente y consistente.

Este enfoque permitió que los distintos integrantes del equipo trabajaran de forma paralela en sus respectivas áreas, facilitando la integración progresiva de funcionalidades y asegurando que los resultados generados por los modelos predictivos y las herramientas de análisis pudieran ser correctamente almacenados, procesados y visualizados dentro de la aplicación.

Finalmente, la estructuración del backend buscó mantener una base sólida y extensible, capaz de adaptarse a futuras mejoras sin afectar el trabajo realizado en los demás componentes del sistema, reforzando así el carácter colaborativo y modular del proyecto.

3.6.4 Integración de Inteligencia Artificial como Herramienta de Apoyo

La incorporación del asistente de inteligencia artificial se diseñó como un componente de apoyo al sistema, orientado a mejorar la experiencia del usuario y facilitar la interpretación de resultados. En lugar de integrar modelos complejos dentro del backend, se optó por utilizar un servicio externo especializado, lo que permitió mantener una arquitectura liviana y modular.

Esta decisión también permitió explorar el uso práctico de tecnologías de inteligencia artificial modernas dentro de un contexto agrícola, sin interferir con la lógica central del sistema. De esta forma, AgroPredict combina herramientas matemáticas tradicionales con capacidades de procesamiento de lenguaje natural, aportando valor adicional sin aumentar innecesariamente la complejidad del proyecto.

3.7 Pruebas y Validación

Con el propósito de verificar el correcto funcionamiento de AgroPredict, se realizaron distintas pruebas durante el desarrollo, permitiendo comprobar el cumplimiento de las funcionalidades implementadas y la correcta interacción entre los distintos componentes de la plataforma. Estas pruebas incluyeron pruebas unitarias, pruebas de APIs y pruebas de integración, orientadas a validar el comportamiento del sistema en diferentes escenarios de uso.

3.7.1

Como apoyo al proceso de desarrollo, se utilizaron pruebas unitarias mediante el framework Pytest, con el objetivo de verificar el correcto funcionamiento de componentes específicos del backend. Estas pruebas permitieron validar el comportamiento esperado de funciones y módulos implementados, facilitando la detección temprana de errores durante el desarrollo y contribuyendo a mantener la estabilidad del sistema a medida que se incorporaban nuevas funcionalidades.

pytest-report.html

Report generated on 16-Sep-2026 at 06:38:01 by [pytest-html](#) v4.2.0

Environment

Python	3.13.15
Platform	Windows-10-10.0.19045-SP0
Packages	• pytest: 9.1.1 • pluggy: 1.6.0
Plugins	• anyio: 4.15.1 • django: 4.14.0 • html: 4.2.0 • metadata: 3.1.1

Summary

3 tests took 00:00:03.

(Un)check the boxes to filter the results.

0 Failed	3 Passed	0 Skipped	0 Expected failures	0 Unexpected passes	0 Errors	0 Reruns	0 Retried	Show all details	/	Hide all details
Result	Test							Duration		Links
Passed	tests/test_ia.py::test_obtener_datos_clima_devuelve_diccionario							2 ms		
Passed	tests/test_ia.py::test_ia_consulta_sin_parametro_q							61 ms		
Passed	tests/test_ia.py::test_ia_consulta_post_envia_pregunta							618 ms		

Figura 10: Resultado de la ejecución de pruebas mediante pytest

Fuente: Captura del reporte en HTML



La Figura 10 muestra el informe HTML generado mediante Pytest. Se ejecutaron tres pruebas automatizadas correspondientes a la validación de datos climáticos y al endpoint de consulta de inteligencia artificial. Todas las pruebas fueron aprobadas, sin errores funcionales durante la ejecución.

3.7.2

Para validar el funcionamiento de los servicios implementados en el backend, se realizaron pruebas utilizando la herramienta Postman. En particular, se verificó el correcto funcionamiento del endpoint correspondiente al asistente de inteligencia artificial, comprobando la recepción de solicitudes, el procesamiento de las consultas y la generación de respuestas por parte del sistema.

A continuación, la Figura 10 presenta una prueba realizada mediante Postman al endpoint del asistente de inteligencia artificial, donde se observa una solicitud procesada correctamente (HTTP 200 OK) y la respuesta generada por el sistema en formato JSON.

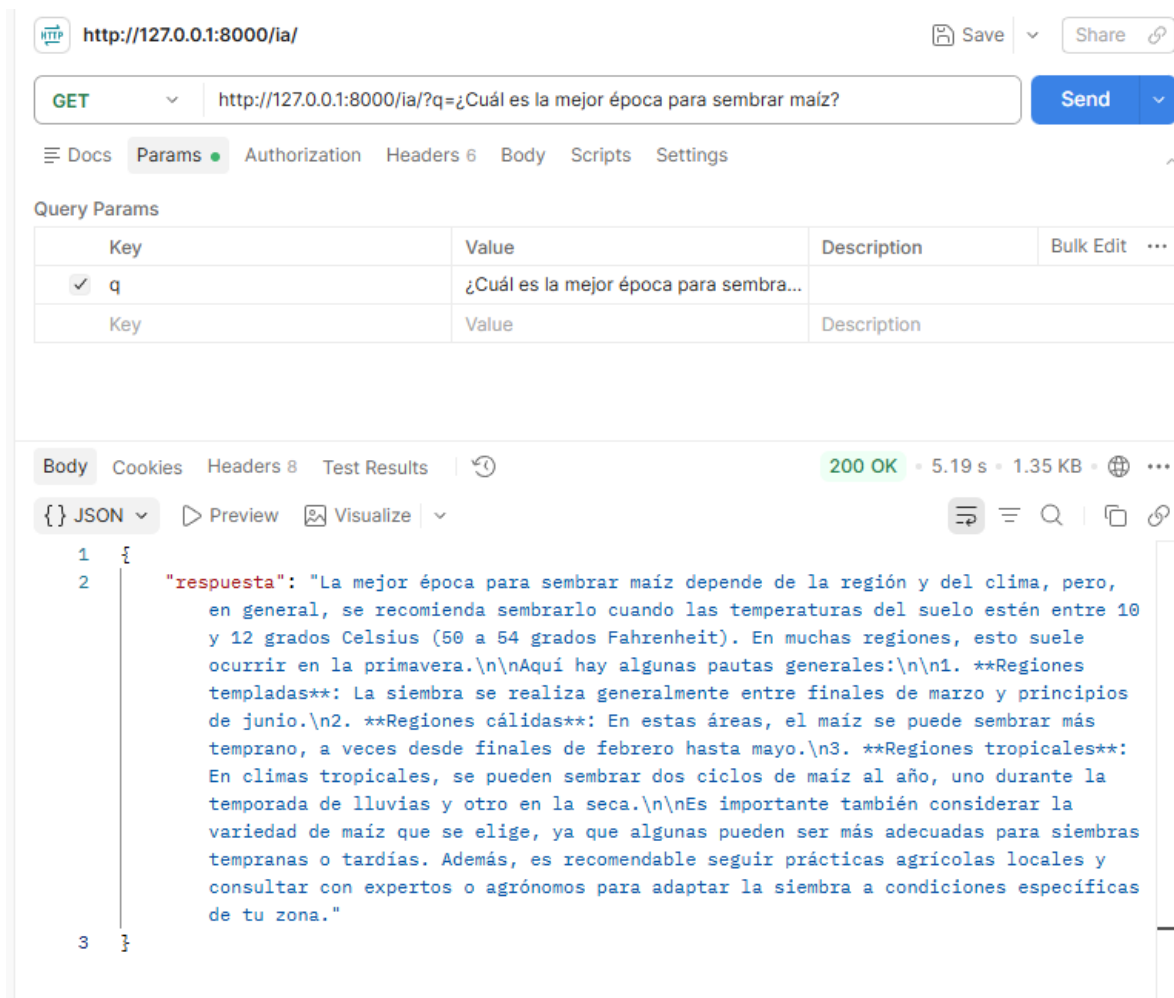


Figura 11: Prueba del endpoint del asistente de inteligencia artificial utilizando Postman.

Fuente: Captura de la prueba en postman

3.7.3

Además de las pruebas unitarias y de API, se consideró la realización de pruebas de integración para verificar la comunicación entre los distintos componentes de AgroPredict. Estas pruebas permitirían comprobar la interacción entre Django, la base de datos SQLite, el modelo predictivo LSTM y la API de OpenAI. Sin embargo, en la versión evaluada, la validación automatizada se concentró en pruebas unitarias y de API, mientras que la integración general fue comprobada mediante la ejecución manual de la aplicación y sus funcionalidades principales.

3.7.4

Finalmente, se validó el funcionamiento general de la plataforma mediante distintos escenarios de uso, verificando la creación de nuevas predicciones, el almacenamiento de la información en la base de datos, la visualización del historial de predicciones, el funcionamiento de las calculadoras de apoyo y la correcta generación de respuestas mediante el asistente de inteligencia artificial.

Las pruebas realizadas permitieron comprobar el correcto funcionamiento de las funcionalidades implementadas y la adecuada integración entre los distintos componentes del sistema, cumpliendo con los objetivos establecidos para el desarrollo de la plataforma.

3.8 Proyección de la Arquitectura y Escalabilidad

La arquitectura de AgroPredict fue diseñada considerando no solo los requerimientos actuales del proyecto, sino también su posible evolución a futuro. En este sentido, una de las principales proyecciones técnicas corresponde a la separación del modelo predictivo basado en LSTM en un microservicio independiente, utilizando tecnologías como FastAPI.

La implementación del modelo de inteligencia artificial como un servicio desacoplado permitiría mejorar la escalabilidad y mantenibilidad del sistema, facilitando la ejecución de predicciones de manera asincrónica y el despliegue independiente del componente de análisis. Esta aproximación también permitiría incorporar nuevas versiones del modelo o incluso distintos modelos predictivos sin afectar el funcionamiento del backend principal.

Asimismo, el uso de FastAPI para la exposición de endpoints especializados permitiría optimizar el rendimiento de las solicitudes relacionadas con predicción y análisis, además de habilitar una integración más flexible con otras aplicaciones o servicios externos. Este enfoque arquitectónico resulta coherente con una evolución natural del sistema hacia una arquitectura orientada a servicios.

De esta forma, AgroPredict mantiene una base sólida en su implementación actual, pero deja abierta la posibilidad de crecimiento y adaptación, alineándose con buenas prácticas de diseño de software y arquitecturas modernas.

4 Conclusiones

El desarrollo de AgroPredict permitió consolidar una solución tecnológica orientada a apoyar la toma de decisiones en el ámbito agrícola mediante la integración de herramientas de predicción, análisis productivo y modelos de inteligencia artificial. A lo largo del proyecto se logró diseñar e implementar una plataforma web completamente funcional, sustentada en el framework Django, que centraliza los principales procesos agrícolas vinculados a la estimación de rendimiento, el análisis económico y la interpretación de datos climáticos. Esta integración permitió ofrecer una experiencia consistente y accesible, donde el usuario puede navegar libremente entre las distintas funcionalidades, crear predicciones personalizadas y obtener información útil para la gestión de sus cultivos.

La utilización de una base de datos relacional administrada mediante la ORM de Django permitió estructurar el dominio de manera coherente, facilitando la manipulación de datos y manteniendo la escalabilidad del sistema. Asimismo, la incorporación de servicios externos como la API de AccuWeather y la API de OpenAI demostró el potencial de integrar fuentes de información en tiempo real y capacidades avanzadas de IA dentro de un entorno práctico orientado al usuario. El módulo basado en LSTM, aunque aún en evolución, permitió explorar técnicas modernas de aprendizaje profundo aplicadas a la predicción agrícola, ampliando las posibilidades del sistema y otorgando un valor agregado al proyecto.

En conjunto, AgroPredict constituye un ejemplo de cómo las tecnologías web, los modelos predictivos y los servicios de inteligencia artificial pueden converger para ofrecer herramientas claras y confiables a quienes trabajan en el sector agrícola. Si bien la plataforma cumple con los objetivos planteados y demuestra su utilidad en un contexto académico, también abre la puerta a mejoras futuras que pueden fortalecer su precisión, su escalabilidad y su capacidad de integrarse con otros sistemas. Este proyecto evidencia tanto el potencial como la necesidad de herramientas digitales accesibles para acompañar a los agricultores en sus decisiones productivas, contribuyendo a optimizar el uso de recursos y a mejorar la planificación agrícola en escenarios cada vez más complejos.



5 Agradecimientos

Quiero agradecer en primer lugar a mis profesores guías, por el tiempo, la orientación y la disposición para apoyar este trabajo, especialmente en los momentos en que fue necesario replantear ideas o corregir el rumbo del proyecto.

También quiero agradecer a mis compañeros de AgroPredict, con quienes trabajé durante todo el desarrollo del proyecto. El intercambio de ideas, el apoyo mutuo y el trabajo en equipo fueron fundamentales para poder concretar esta propuesta.

Finalmente, agradezco a mi familia por el apoyo constante a lo largo de toda mi formación académica, por la confianza, la motivación y la ayuda en cada etapa de este camino. Su apoyo fue clave no solo para la realización de esta tesina, sino también para mi desarrollo personal y profesional. Asimismo, agradezco a mis amigos por el apoyo y la motivación durante este proceso.

6 Referencias

- Django Software Foundation. (2025). Documentación oficial de Django. <https://docs.djangoproject.com/>
- OpenAI. (2025). Documentación de la API de OpenAI. <https://platform.openai.com/docs/>
- Python Software Foundation. (2025). Documentación oficial de Python. <https://docs.python.org/es/3/>
- TensorFlow. (2025). TensorFlow Documentation. <https://www.tensorflow.org/>
- Postman, Inc. (2025). Postman Learning Center. <https://learning.postman.com/>
- Krekel, H., Oliveira, B., Pfannschmidt, R., & Bruynooghe, F. (2025). pytest documentation. <https://docs.pytest.org/>
- Schwaber, K., & Sutherland, J. (2020). La Guía Scrum. <https://scrumguides.org/es/>
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.