AgroSense: Sistema Inteligente de Monitoreo Agrícola

Jhoan Alexander Molina Gomez (192490)

Adrian Camilo Rincon Ascanio (192531)

Isaac David García Vesga (192535)

Docente: Jesús Guerrero

Universidad Francisco de Paula Santander Seccional Ocaña

Programación Orientada a Objetos

10 de diciembre de 2025

Introducción

La agricultura moderna enfrenta desafíos significativos relacionados con la eficiencia en el uso de recursos y la optimización de la producción. En un contexto donde el agua se ha convertido en un recurso cada vez más escaso y la demanda de alimentos continúa en aumento, resulta imperativo desarrollar soluciones tecnológicas que permitan a los agricultores tomar decisiones informadas basadas en datos objetivos y en tiempo real.

AgroSense surge como respuesta a esta necesidad, presentándose como un sistema inteligente de monitoreo agrícola diseñado específicamente para pequeñas y medianas explotaciones. El sistema integra tecnología de sensores IoT con una plataforma de software desarrollada en Java, utilizando principios de programación orientada a objetos para garantizar un código mantenible, escalable y robusto.

Este proyecto representa la culminación de los conocimientos adquiridos en el curso de Programación Orientada a Objetos, aplicando conceptos fundamentales como encapsulamiento, herencia, polimorfismo y abstracción en la resolución de un problema real del sector agrícola. A través de AgroSense, se demuestra cómo la tecnología puede democratizarse y llegar a sectores tradicionalmente alejados de la innovación digital, proporcionando herramientas accesibles que mejoran la productividad y sostenibilidad de las prácticas agrícolas.

Objetivos

Objetivo General

Diseñar e implementar un sistema de monitoreo agrícola basado en programación orientada a objetos que permita optimizar el uso del agua, mejorar la productividad y facilitar la toma de decisiones de los agricultores mediante el análisis de datos en tiempo real provenientes de sensores IoT.

Objetivos Específicos

1. Desarrollar una arquitectura de software orientada a objetos que modele de manera efectiva los componentes del sistema agrícola (lotes, sensores, mediciones, alertas y recomendaciones).
2. Implementar una interfaz gráfica de usuario utilizando JavaFX que permita visualizar de manera intuitiva el estado de los cultivos y las condiciones ambientales de cada lote.
3. Diseñar e implementar un sistema de alertas inteligentes que notifique al agricultor sobre condiciones críticas detectadas por los sensores de humedad y temperatura.
4. Crear un módulo de recomendaciones que proporcione consejos prácticos basados en el análisis de los datos recopilados, promoviendo el uso eficiente de recursos hídricos.
5. Implementar un sistema de persistencia de datos que permita guardar y recuperar la información del sistema entre sesiones de uso.
6. Validar el funcionamiento del sistema mediante pruebas unitarias y de integración que garanticen la calidad y confiabilidad del software.

Justificación

Análisis del Problema

La agricultura de pequeña escala enfrenta múltiples desafíos debido al uso de métodos tradicionales y la falta de acceso a tecnología moderna. Muchos agricultores continúan realizando el riego y la aplicación de insumos de manera manual, basándose únicamente en su experiencia y observación visual, sin contar con datos objetivos sobre las condiciones reales del suelo y el ambiente.

Árbol de Problemas

Problema Central: Gestión ineficiente de recursos en la agricultura de pequeña escala

Causas Identificadas:

- Falta de información objetiva sobre las condiciones del suelo y ambiente
- Ausencia de sistemas de monitoreo en tiempo real
- Limitado acceso a tecnología agrícola moderna
- Dependencia exclusiva de métodos tradicionales y experiencia empírica
- Carencia de alertas tempranas sobre condiciones críticas

Efectos Resultantes:

- Uso ineficiente del agua: se riega sin medir la necesidad real del cultivo, generando desperdicio de un recurso cada vez más escaso
- Baja productividad: el estrés hídrico y las plagas se detectan tardíamente, disminuyendo significativamente el rendimiento de los cultivos

- Pérdidas económicas: la aplicación inadecuada de recursos resulta en costos innecesarios y menor rentabilidad
- Impacto ambiental negativo: el uso excesivo de agua y fertilizantes afecta la sostenibilidad de las prácticas agrícolas
- Toma de decisiones basada en intuición: sin datos concretos, las decisiones críticas dependen únicamente de la experiencia, aumentando el riesgo de error

Solución Propuesta

AgroSense aborda directamente estas problemáticas mediante la implementación de un sistema que visualiza las condiciones de cultivo en tiempo real, genera alertas oportunas y proporciona recomendaciones claras y prácticas. El sistema ofrece los siguientes beneficios:

- Visualización sencilla e intuitiva del estado de cada lote
- Alertas oportunas sobre necesidades de riego o cuidados especiales
- Recomendaciones prácticas basadas en datos objetivos
- Ahorro significativo de agua y recursos mediante riego inteligente
- Mejor rendimiento de los cultivos al evitar pérdidas por detección tardía
- Tecnología accesible que no requiere conocimientos técnicos complejos

Ciclo de Vida del Desarrollo de Software (SDLC)

El desarrollo de AgroSense siguió un enfoque iterativo e incremental, adaptando las mejores prácticas del desarrollo de software a las necesidades específicas del proyecto. A continuación, se describen las fases principales del ciclo de vida implementado:

1. Análisis de Requisitos

En esta fase inicial se identificaron las necesidades del sector agrícola y se definieron los requisitos funcionales y no funcionales del sistema.

Requisitos Funcionales:

- Gestión de lotes de cultivo con información detallada (ID, nombre, tipo de cultivo, área, fecha de siembra, etapa de crecimiento)
- Registro y monitoreo de sensores de humedad y temperatura asociados a cada lote
- Generación automática de mediciones simuladas para demostración del sistema
- Sistema de alertas que detecte condiciones críticas (humedad baja, temperatura extrema)
- Módulo de recomendaciones personalizadas basadas en el estado de cada lote
- Persistencia de datos para mantener la información entre sesiones
- Interfaz gráfica intuitiva para visualización y gestión de toda la información

Requisitos No Funcionales:

- Usabilidad: interfaz intuitiva accesible para usuarios sin conocimientos técnicos avanzados
- Mantenibilidad: código bien estructurado siguiendo principios SOLID y patrones de diseño
- Escalabilidad: arquitectura que permita agregar nuevos tipos de sensores y funcionalidades
- Rendimiento: respuesta rápida en la visualización de datos y generación de alertas
- Confiabilidad: validación de datos y manejo robusto de errores

2. Diseño del Sistema

El diseño de AgroSense se basó en principios de programación orientada a objetos, creando una arquitectura modular y bien estructurada.

Arquitectura del Sistema:

El sistema se organizó en tres capas principales:

- Capa de Modelo (com.agrosense.model): Contiene las clases de dominio que representan las entidades del sistema (Lote, Sensor, SensorHumedad, SensorTemperatura, Medicion, Alerta, Recomendacion)
- Capa de Servicio (com.agrosense.service): Implementa la lógica de negocio mediante servicios especializados (GestorLotes, SensorService, AlertaService, RecomendacionService, EstadisticasService, ToonPersistenceService, DatosDemoService)
- Capa de Presentación (com.agrosense.ui): Maneja la interfaz de usuario tanto en consola (ConsoleUI) como en interfaz gráfica (AgroSenseFX con JavaFX)

Patrones de Diseño Aplicados:

- Herencia: Jerarquía de sensores con clase base Sensor y especializaciones SensorHumedad y SensorTemperatura
- Encapsulamiento: Atributos privados con getters/setters apropiados en todas las clases del modelo
- Separación de Responsabilidades: Cada clase de servicio tiene una responsabilidad única y bien definida
- Inyección de Dependencias: Los servicios se pasan como parámetros donde se necesitan, facilitando las pruebas

3. Implementación

La implementación se realizó utilizando Java 17 como lenguaje principal, aprovechando sus características modernas y robustas.

Tecnologías Utilizadas:

- Java 17: Lenguaje de programación principal
- JavaFX 17.0.6: Framework para la interfaz gráfica de usuario
- Maven: Gestión de dependencias y construcción del proyecto
- JUnit 5: Framework para pruebas unitarias
- Git: Control de versiones del código fuente

Componentes Principales Implementados:

- Modelo de Datos: 7 clases principales (Lote, Sensor, SensorHumedad, SensorTemperatura, Medicion, Alerta, Recomendacion)
- Servicios de Negocio: 7 servicios especializados que implementan toda la lógica del sistema
- Interfaz Gráfica: Aplicación JavaFX con 4 pestañas principales (Lotes, Sensores, Monitoreo, Alertas)
- Sistema de Persistencia: Serialización de objetos para guardar y recuperar datos
- Base de Datos de Cultivos: Utilidad para validar nombres de cultivos
- Generación de Datos Demo: Servicio para crear datos de ejemplo y facilitar la demostración

4. Pruebas y Validación

Se implementó una estrategia de pruebas multinivel para garantizar la calidad y confiabilidad del sistema.

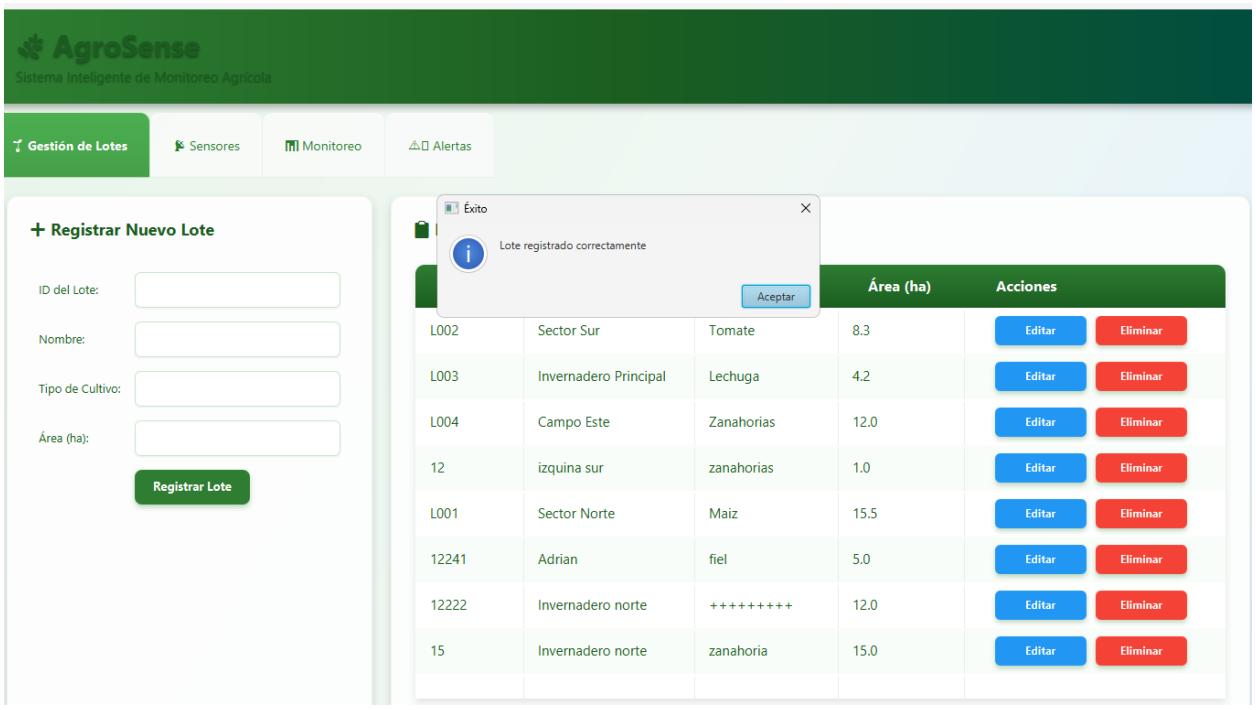
Tipos de Pruebas Realizadas:

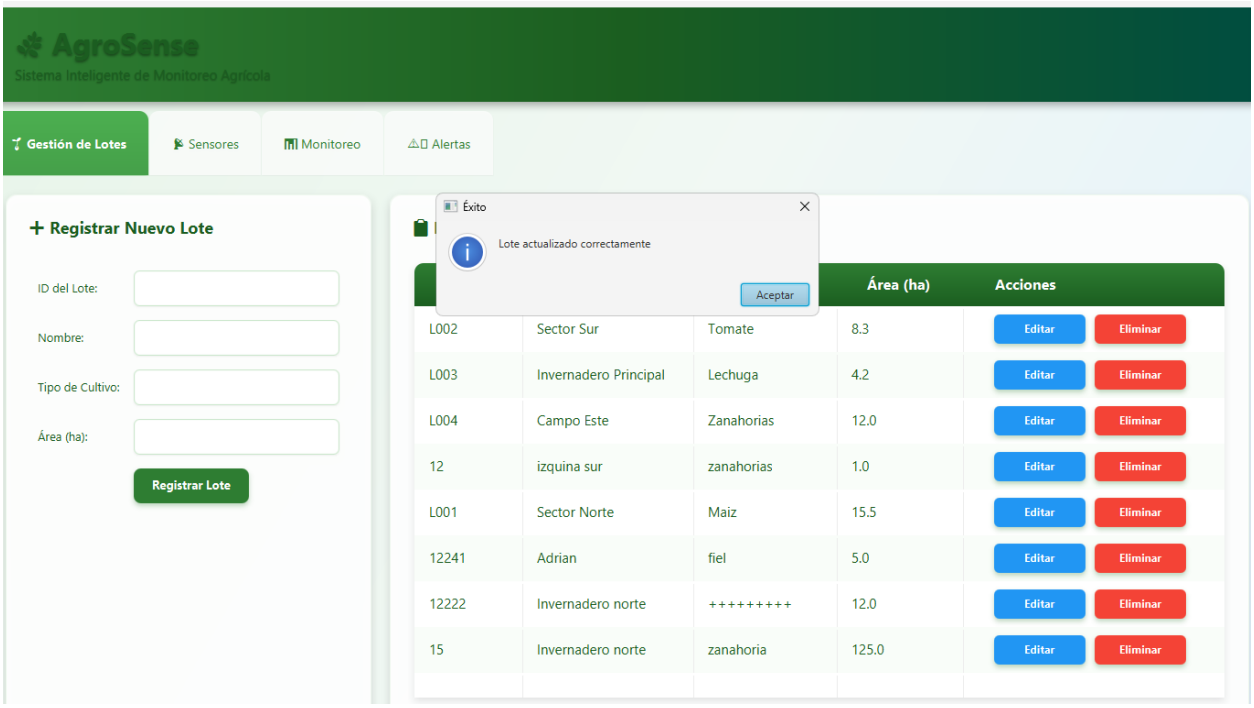
- Pruebas Unitarias: Validación de la lógica de cada clase del modelo, verificando constructores, getters, setters y métodos de negocio
- Pruebas de Integración: Verificación de la correcta interacción entre servicios y componentes del sistema
- Pruebas de Interfaz: Validación manual de todas las funcionalidades de la interfaz gráfica JavaFX
- Pruebas de Persistencia: Verificación de que los datos se guardan y recuperan correctamente entre sesiones
- Pruebas de Validación: Comprobación de que el sistema maneja correctamente entradas inválidas y casos extremos

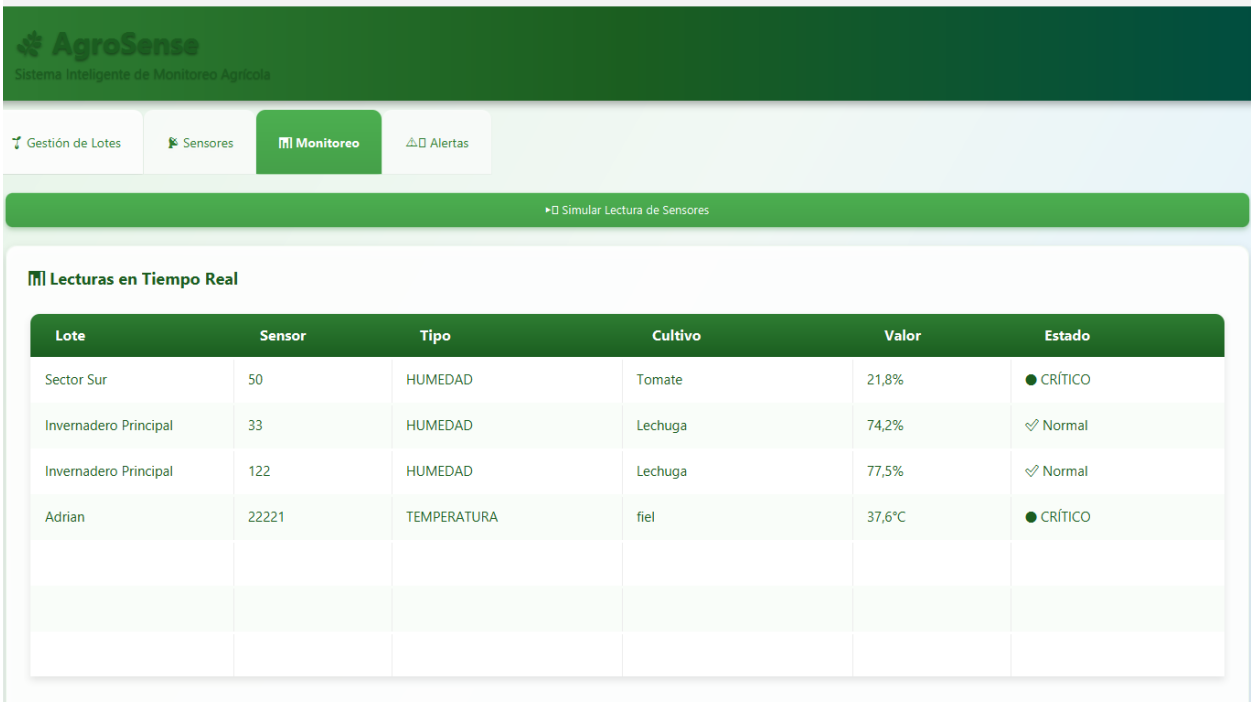
Evidencias de Funcionamiento:

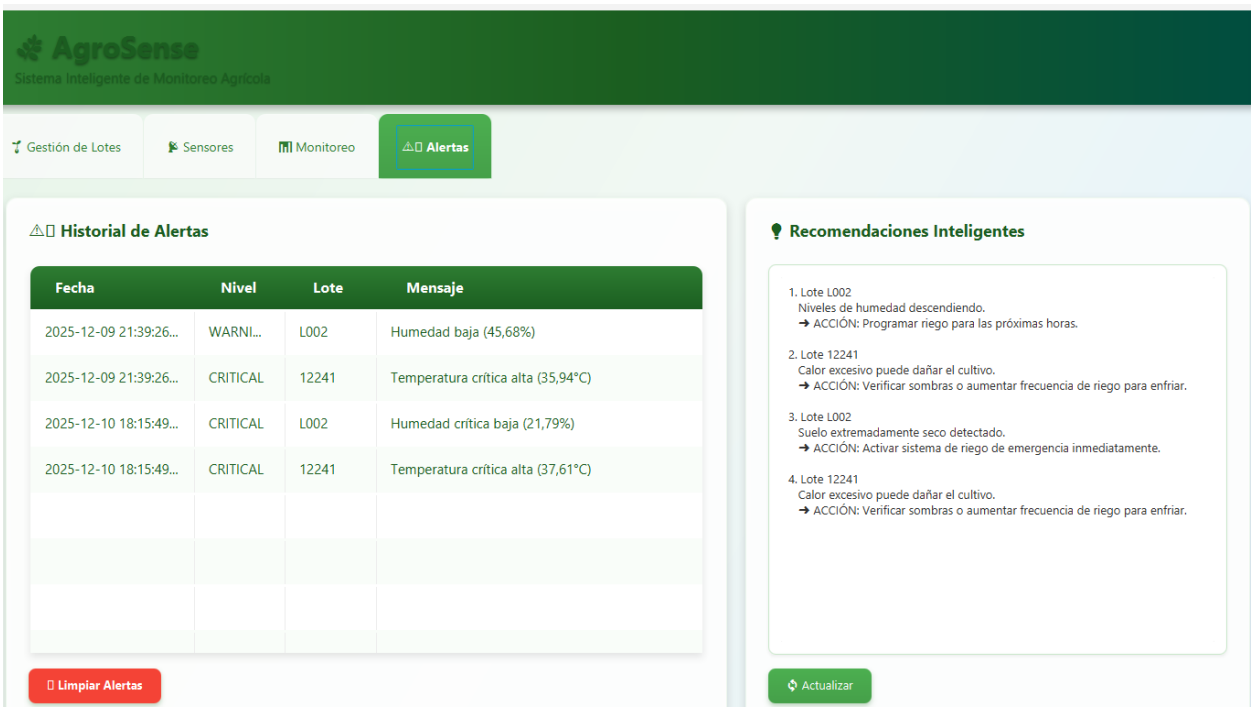
El sistema ha sido probado exhaustivamente, demostrando las siguientes capacidades:

- Gestión completa de lotes: crear, editar, visualizar y eliminar lotes de cultivo con toda su información asociada
- Administración de sensores: agregar, configurar y eliminar sensores de diferentes tipos en cada lote
- Monitoreo en tiempo real: visualización de mediciones actuales de todos los sensores activos
- Sistema de alertas funcional: detección y notificación de condiciones críticas basadas en umbrales configurables
- Recomendaciones inteligentes: generación de consejos prácticos basados en el análisis de datos
- Persistencia confiable: los datos se mantienen entre sesiones de uso del sistema




AgroSense
Sistema Inteligente de Monitoreo Agrícola
Gestión de Lotes
Sensores
Monitoreo
Alertas
+ Registrar Nuevo Lote
ID del Lote:
Nombre:
Tipo de Cultivo:
Área (ha):
Registrar Lote
Éxito
Lote actualizado correctamente
Aceptar
Área (ha)
Acciones
L002 Sector Sur Tomate 8.3 Editar Eliminar
L003 Invernadero Principal Lechuga 4.2 Editar Eliminar
L004 Campo Este Zanahorias 12.0 Editar Eliminar
12 izquina sur zanahorias 1.0 Editar Eliminar
L001 Sector Norte Maiz 15.5 Editar Eliminar
12241 Adrian fiel 5.0 Editar Eliminar
12222 Invernadero norte +++++++++ 12.0 Editar Eliminar
15 Invernadero norte zanahoria 125.0 Editar Eliminar


AgroSense
Sistema Inteligente de Monitoreo Agrícola
Gestión de Lotes
Sensores
Monitoreo
Alertas
Simular Lectura de Sensores
Lecturas en Tiempo Real
Lote Sensor Tipo Cultivo Valor Estado
Sector Sur 50 HUMEDAD Tomate 21,8% CRÍTICO
Invernadero Principal 33 HUMEDAD Lechuga 74,2% Normal
Invernadero Principal 122 HUMEDAD Lechuga 77,5% Normal
Adrian 22221 TEMPERATURA fiel 37,6°C CRÍTICO



5. Despliegue y Distribución

El proyecto está configurado para facilitar su distribución y ejecución en diferentes entornos.

Métodos de Ejecución:

- Ejecución con Maven: mvn clean javafx:run para ejecutar la interfaz gráfica
- Ejecución del JAR: Generación de un archivo JAR ejecutable mediante Maven Shade Plugin
- Modo Consola: Opción de ejecutar el sistema en modo consola para entornos sin interfaz gráfica
- Repositorio Git: Código fuente disponible en GitHub para clonación y colaboración

6. Mantenimiento y Evolución

El diseño modular y la documentación del código facilitan el mantenimiento futuro y la incorporación de nuevas funcionalidades.

Mejoras Futuras Planificadas:

- Integración con sensores IoT reales mediante protocolos MQTT o HTTP
- Implementación de base de datos relacional (PostgreSQL o MySQL) para mayor escalabilidad
- Desarrollo de aplicación móvil para Android/iOS
- Incorporación de análisis predictivo mediante machine learning
- Sistema de notificaciones push para alertas críticas
- Generación de reportes y estadísticas avanzadas en PDF
- Soporte multi-usuario con autenticación y roles

Conclusiones

1. El desarrollo de AgroSense ha demostrado exitosamente la aplicabilidad de los principios de programación orientada a objetos en la resolución de problemas reales del sector agrícola. La arquitectura modular implementada permite una fácil extensión y mantenimiento del sistema, cumpliendo con los objetivos académicos del curso.

2. La implementación de una jerarquía de clases bien diseñada, con separación clara de responsabilidades entre modelo, servicio y presentación, ha resultado en un código limpio, mantenible y escalable. Los conceptos de encapsulamiento, herencia y polimorfismo se aplicaron de manera efectiva en componentes clave como la jerarquía de sensores.

3. El sistema de alertas y recomendaciones implementado demuestra cómo el análisis de datos en tiempo real puede traducirse en información accionable para los agricultores. La capacidad de detectar condiciones críticas y proporcionar consejos prácticos representa un valor tangible para los usuarios finales.

4. La interfaz gráfica desarrollada con JavaFX logra un balance entre funcionalidad y usabilidad, presentando información compleja de manera intuitiva y accesible. La organización en pestañas y el uso de tablas facilitan la gestión de múltiples lotes y sensores simultáneamente.

5. El proyecto ha evidenciado la importancia de las pruebas en el desarrollo de software confiable. Las pruebas unitarias implementadas garantizan que los componentes individuales funcionen correctamente, mientras que las pruebas de integración validan la interacción entre diferentes partes del sistema.

6. AgroSense representa un primer paso hacia la democratización de la tecnología agrícola, demostrando que soluciones innovadoras pueden desarrollarse con herramientas

accesibles y conocimientos fundamentales de programación. El sistema sienta las bases para futuras mejoras que podrían incluir integración con hardware real y análisis predictivo.

7. La experiencia de desarrollo ha reforzado la comprensión de conceptos clave como el diseño de APIs, la gestión de estado, la persistencia de datos y la creación de interfaces de usuario. Estos conocimientos son fundamentales para el desarrollo profesional de software en cualquier dominio.

8. El uso de herramientas modernas como Maven para gestión de dependencias, Git para control de versiones y JUnit para pruebas, ha proporcionado experiencia práctica con el ecosistema de desarrollo Java profesional, preparando para proyectos de mayor envergadura.

Referencias

Nota: Las referencias están formateadas según las normas APA 7ma edición, con sangría francesa y orden alfabético.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley Professional.

Goadrich, M. H., & Rogers, M. P. (2011). Smart smartphone development: iOS versus Android. Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, 607-612. https://doi.org/10.1145/1953163.1953330

Horstmann, C. S. (2019). Core Java Volume I: Fundamentals (11th ed.). Prentice Hall.

Martin, R. C. (2017). Clean architecture: A craftsman's guide to software structure and design. Prentice Hall.

Oracle Corporation. (2023). JavaFX Documentation. https://openjfx.io/javadoc/17/

Pressman, R. S., & Maxim, B. R. (2020). Ingeniería del software: Un enfoque práctico (9a ed.). McGraw-Hill Education.

Schildt, H. (2022). Java: The complete reference (12th ed.). McGraw-Hill Education.

Sierra, K., & Bates, B. (2005). Head First Java (2nd ed.). O'Reilly Media.

Sommerville, I. (2016). Software engineering (10th ed.). Pearson Education.