Documentación del Proyecto Final (APA visual)

Nombre del Autor

26 de noviembre de 2025

Índice

1. Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a aquellas personas que nos han acompañado y apoyado a lo largo de nuestra trayectoria académica. En especial, a nuestro docente y tutor Nicolás Escobar, quien ha sido un pilar fundamental en nuestra formación profesional. Sus observaciones, correcciones y sugerencias nos permitieron mejorar la calidad del proyecto y mantener una dirección clara en el proceso de desarrollo.

Agradecemos también a nuestras familias, por su apoyo constante y por facilitar las condiciones necesarias para llevar adelante este trabajo, especialmente durante los períodos de mayor dedicación. Su comprensión y acompañamiento hicieron posible sostener el ritmo de trabajo requerido.

A todas esas personas, nuestro más sincero agradecimiento.

2. Resumen

El sistema web integral desarrollado para el Instituto Nacional de Investigación Agropecuaria (INIA) surge como una solución tecnológica para modernizar y digitalizar los procesos de análisis de calidad de semillas. El proyecto, realizado como trabajo final de la carrera Tecnólogo Informático, aborda la necesidad de unificar y estandarizar procedimientos técnicos que anteriormente se gestionaban mediante planillas Excel. El sistema permite administrar lotes y diversos análisis — Germinación, Tetrazolio, PMS (Peso de Mil Semillas), DOSN (Determinación de Otras Semillas en Números) y Pureza Física — incorporando validaciones basadas en estándares internacionales, trazabilidad e historial técnico por lote, normalización de catálogos, importación de datos legados y generación de reportes especializados con exportación consolidada a Excel. La arquitectura se implementó mediante microservicios, con un backend en Java Spring Boot 3.5 expuesto como API REST y reforzado con seguridad por JWT y autenticación en dos factores (2FA). Se incorporaron notificaciones en tiempo real mediante WebSockets, paginación basada en cursores y pruebas automatizadas con JUnit/JaCoCo. El frontend fue desarrollado en Next.js 14 con TypeScript, ofrecien-

do una PWA con funcionamiento offline, persistencia local de formularios, validaciones dinámicas y experiencia responsiva en campo. El sistema resultante mejora la trazabilidad de los datos, la consistencia metodológica de los análisis y la eficiencia operativa institucional, estableciendo una base sólida para futuras ampliaciones y para la interoperabilidad con otros sistemas del INIA.

3. Introducción

En la última década, la digitalización ha transformado profundamente la gestión de información en sectores productivos y científicos, impulsando nuevos estándares de eficiencia, trazabilidad y precisión en los procesos operativos. En este contexto, el ámbito agropecuario no ha sido la excepción: la demanda de sistemas capaces de centralizar datos, reducir tareas manuales y garantizar la consistencia metodológica se ha vuelto fundamental para sostener la calidad de los análisis y la toma de decisiones institucionales. Previo a este proyecto, el proceso de análisis de calidad de semillas en el Instituto Nacional de Investigación Agropecuaria (INIA) presentaba desafíos significativos derivados del uso de registros dispersos, planillas Excel independientes y procedimientos manuales con alto riesgo de error. Los análisis técnicos —como Germinación, Tetrazolio, PMS, DOSN y Pureza Física— se realizaban sobre lotes de semillas cuyos datos se registraban de forma heterogénea, dificultando el seguimiento del historial completo de cada lote. Esta fragmentación generaba esfuerzos adicionales de normalización y verificación, con impacto en los tiempos de respuesta, en la trazabilidad de los resultados y en la comparabilidad histórica de la información técnica. Además, la ausencia de notificaciones oportunas y la falta de catálogos unificados limitaban la coordinación operativa entre equipos. Frente a esta necesidad, surgió el desarrollo del Sistema Web Integral INIA, una plataforma diseñada para modernizar, centralizar y estandarizar todo el ciclo de vida de la información vinculada a los análisis de calidad de semillas. El proyecto, realizado como trabajo final de la carrera Tecnólogo Informático, propone una solución que gestiona de manera estructurada los lotes, sus datos técnicos asociados, los contactos involucrados, las validaciones específicas de cada tipo de análisis, el historial completo de resultados y la generación automatizada de reportes. Asimismo, se incorporó la importación de información proveniente

de sistemas legados, preservando datos históricos relevantes y facilitando la transición tecnológica. En conjunto, esta solución tecnológica permitió profesionalizar la gestión de los análisis y de los lotes de semillas, reducir errores operativos, fortalecer la trazabilidad institucional y mejorar sustancialmente la eficiencia en el registro, control y procesamiento de información técnica. El sistema constituye una base sólida para futuras ampliaciones, nuevos módulos de interoperabilidad y una evolución continua hacia un ecosistema digital integral para el INIA.

4. Objetivos Planteados y Resultados Esperados

El principal objetivo del proyecto fue analizar, diseñar y desarrollar un sistema integral para la gestión de análisis de semillas realizados por el Instituto Nacional de Investigación Agropecuaria, orientado a la optimización del proceso actual de registro, edición, consulta y trazabilidad de las muestras (lotes) ingresadas para su análisis. Este sistema busca centralizar y modernizar la operativa ya existente, brindando tecnologías que faciliten la agilizar tareas, reducir errores y mejorar la disponibilidad de información para todos los posibles usuarios. Durante el proceso de construcción del sistema se evaluaron diferentes alternativas tecnológicas para su implementación, optando por Java 21 junto con Springboot 3.5 y Spring Data JPA para el backend REST. Mientras que en el frontend se consideró React con Tailwind CSS 4.1. Asimismo se buscó la integración de módulos complementarios, como notificaciones, reportes y la exportación e importación de archivos formato xlsx.

4.1. Objetivos específicos

- Un sistema centralizado y estable para gestionar todo el ciclo de análisis.
- Mayor trazabilidad y transparencia en el seguimiento de lotes.
- Reducción de errores mediante automatización y validaciones.
- Disminución del tiempo dedicado a tareas administrativas.

- Operativa más moderna, simple e intuitiva.

5. Estado del Arte

En los últimos años, la transformación digital se ha vuelto una realidad en prácticamente todas las organizaciones, tanto públicas como privadas. Este proceso busca mejorar la forma en que se gestionan los datos y se llevan adelante las tareas diarias. En el sector agropecuario, especialmente, la digitalización ha demostrado ser clave para lograr una mayor trazabilidad, precisión en los registros y eficiencia operativa. Por este motivo, cada vez más instituciones están dejando atrás los documentos en papel, las planillas de Excel y los procedimientos dispersos, y están migrando hacia plataformas web que centralizan y organizan la información de forma más segura y confiable. Esta tendencia está ampliamente respaldada por estudios académicos y por la práctica común de centros de investigación y laboratorios en todo el mundo. El proyecto desarrollado para el Instituto Nacional de Investigación Agropecuaria (INIA) se inscribe en esta línea de evolución tecnológica. La institución necesita una herramienta digital que permita gestionar sus análisis de forma integral: registrar y consultar muestras (lotes), administrar configuraciones, controlar distintos roles de usuario, generar reportes, exportar información y también incorporar datos históricos provenientes de sistemas anteriores. Este Estado del Arte presenta el marco conceptual, tecnológico y comparativo que justifica la solución propuesta y contextualiza su diseño dentro de las tendencias actuales de la industria del software.

5.1. Marco Teórico

En estos últimos años las organizaciones adoptaron hojas de cálculo como herramienta base para almacenar información gracias a su accesibilidad, bajo costo, flexibilidad y facilidad de uso. Sin embargo, a medida que los procesos se vuelven más complejos y los volúmenes de datos crecen, estos mecanismos se vuelven insuficientes y arriesgados para labores críticas. Las limitaciones más comunes incluyen:

- **Escalabilidad restringida:** Las hojas de cálculo no están diseñadas para manejar grandes

volúmenes de datos ni crecer de forma sostenible. A medida que aumentan los registros, las operaciones se vuelven lentas, propensas a fallos y difíciles de mantener.

- **Colaboración limitada:** La funcionalidad de edición simultánea que es ofrecida por la mayoría de las aplicaciones de gestión de hojas de cálculo es vulnerable a conflictos, sobreescrituras y pérdida de información.
- **Falta de trazabilidad y mecanismos de auditoría:** Resulta difícil rastrear cambios, identificar responsables y asegurar integridad de datos.
- **Integración deficiente con sistemas externos:** La conexión con APIs, bases de datos, servicios externos u otros sistemas institucionales resulta limitada.

5.2. La Transición Hacia Sistemas Web

Las planillas suelen funcionar bien en etapas iniciales, pero a medida que crecen los volúmenes de información o los procesos se vuelven más complejos, es común que se busquen alternativas web que permitan centralizar la información y manejarla de manera más eficiente. Los sistemas modernos ofrecen:

- Acceso multiplataforma.
- Escalabilidad horizontal mediante arquitecturas distribuidas.
- Integración nativa a través de APIs REST.
- Validaciones automáticas a nivel de negocio.
- Auditorías completas de operaciones.
- Estandarización de procesos y flujos de trabajo.

Este enfoque tecnológico es el más utilizado en laboratorios, centros de investigación y organizaciones científicas para gestionar muestras, controles de calidad, análisis y trazabilidad de procesos experimentales.

5.3. Tecnologías

El proyecto adopta tecnologías modernas, robustas y ampliamente utilizadas en la industria. La selección busca equilibrio entre estabilidad, madurez, facilidad de mantenimiento y alineación con estándares profesionales.

5.3.1. Frontend: React + TypeScript

React es uno de los frameworks más utilizados globalmente para construir interfaces de usuario debido a:

- Su modelo de componentes reutilizables
- Su alto rendimiento mediante virtual DOM
- Un ecosistema amplio de librerías
- Facilidad para construir SPA (Single Page Applications)

El uso de TypeScript aporta tipado estático, reduce errores y mejora la mantenibilidad del código.

5.3.2. Backend: Java + Spring Boot

Spring Boot es estándar en el desarrollo empresarial gracias a:

- Integración nativa con Spring Security
- Soporte simplificado para APIs REST
- Inyección de dependencias y modularidad
- Comunidad madura y documentación extensa

Java 21, como versión LTS, asegura estabilidad a largo plazo.

5.3.3. Base de Datos: PostgreSQL

Elegida por ser:

- Open source
- Altamente confiable
- Compatible con operaciones complejas
- Escalable y sólida para manejo de datos institucionales

5.3.4. Infraestructura y Arquitectura

- Modelo cliente-servidor (frontend Next.js – backend Spring Boot)
- Arquitectura en tres capas (presentación – servicios – datos)
- Uso de Docker para portabilidad y despliegue
- Estructura modular por dominios (análisis, seguridad, notificaciones, usuarios)
- Comunicación vía REST y WebSocket (notificaciones en tiempo real)

5.4. Soluciones Similares

En el mercado existen herramientas orientadas a gestionar información estructurada, colaborar en equipos y reemplazar flujos basados en planillas. Aunque ninguna se adapta exactamente a los procesos complejos del INIA, sirven como referencia sobre cómo la industria resuelve problemas similares.

5.4.1. Airtable

Airtable combina conceptos de base de datos con la interfaz amigable de una hoja de cálculo. Es una plataforma low-code orientada a pequeños proyectos y equipos que necesitan digitalizar procesos sin desarrollar software propio. Ventajas:

- Interfaz simple y accesible
- Colaboración en tiempo real
- APIs integradas
- Automatizaciones básicas

Limitaciones:

- No escala para flujos complejos
- Restricciones para reglas de negocio avanzadas
- Dependencia de licencias externas

5.4.2. Smartsheet

Smartsheet ofrece una experiencia similar a Excel, pero con mayor control, trazabilidad y herramientas para flujos de trabajo. Ventajas:

- Gestión de proyectos y procesos
- Reportes avanzados
- Automatizaciones integradas

Limitaciones:

- Alto costo según uso
- Menor flexibilidad frente a un desarrollo a medida
- Dependencia del ecosistema propietario

5.4.3. Odoo

Odoo es un ERP (Planificador de Recursos Empresariales) modular que integra distintos dominios empresariales. Ventajas:

- Gran variedad de módulos
- Comunidad activa
- Escalabilidad para múltiples áreas

Limitaciones:

- Instalación y configuración complejas
- Sobrecarga funcional para proyectos específicos
- Dificultad de adaptación a procesos científicos y de laboratorio

5.5. Conclusión del análisis comparativo

Si bien las herramientas ofrecen funcionalidades útiles, no cumplen de forma precisa con los requerimientos del Instituto Nacional de Investigación Agropecuaria, especialmente en lo relativo a:

- Procesos de laboratorio
- Gestión de análisis y múltiples roles
- Validaciones específicas
- Importación de datos legados
- Adaptación a flujos técnicos

Por ello, un sistema a medida es la opción más adecuada para atender las necesidades reales y garantizar la adaptación total a los procesos internos y ofrecer la flexibilidad necesaria para futuros cambios o expansiones.

6. Análisis del Problema

Para poder desarrollar el sistema web del INIA fue necesario entender a fondo cuáles eran los problemas reales en el trabajo diario del laboratorio. Durante mucho tiempo los análisis se registraban en planillas haciendo difícil mantener un control claro y asegurar la uniformidad de la información. Este diagnóstico inicial fue clave para diseñar una solución que realmente mejorase la forma en que los análisis se gestionan y resultase útil para el cliente. A partir de este análisis, se estudiaron cuidadosamente los procesos actuales, se conversó con los usuarios y se documentaron sus necesidades. Esto permitió definir cómo debía funcionar el sistema, qué información era importante, quiénes interactúan con él y qué herramientas debían construirse para hacer su trabajo más sencillo y seguro. Con esta información se estableció el alcance del proyecto y se identificaron los puntos críticos a resolver en cuanto a diseño y desarrollo.

6.1. Vista del Modelo de Dominio

El modelo de dominio reúne todas las entidades del sistema y muestra cómo se relacionan todas entre sí. Tener esta vista clara fue fundamental para ordenar las ideas, entender el flujo de la información y asegurarse de que el sistema reflejase fielmente la forma en que se trabaja en el laboratorio.

6.2. Definición de Casos de Uso

Los casos de uso ayudan a describir qué puede hacer cada usuario y cómo se relacionan las acciones con las funcionalidades disponibles en el sistema. A partir de ellos se identificaron tareas clave como crear y gestionar lotes y análisis, administrar catálogos, generar reportes y cargar datos históricos.

6.3. Actores

Los actores son los elementos externos al sistema, ya sean usuarios o sistemas, que interactúan con la plataforma para llevar a cabo determinadas tareas. En este proyecto se definieron tres actores centrales: **Administrador:**

- Administra catálogos, lista de contactos y aprobación de usuarios.
- Gestiona análisis y lotes.
- Importa datos históricos.
- Otorga y revoca permisos.
- Exporta reportes en formato xlsx.
- Supervisa y valida las acciones realizadas por los Analistas.
- Observa reportes.
- Tiene acceso total a todos los módulos del sistema.

Analista:

- Registra y edita lotes y análisis.
- Carga y modifica información técnica.
- Requiere la aprobación del Administrador para confirmar los resultados de un análisis.
- Exporta reportes en formato xlsx.
- Observa reportes.

Observador:

- Solo puede visualizar información.

- Accede a lotes, análisis, resultados y reportes.
- No posee permisos para modificar datos.

6.4. Diagrama de Casos de Uso

El siguiente diagrama muestra los principales casos de uso del sistema:

6.5. Vista del Modelo de Diseño

El diseño propuesto para la arquitectura del sistema busca mantener un equilibrio claro entre simplicidad, organización y capacidad de crecimiento. La estructura en capas responde a la necesidad de separar responsabilidades y asegurar que cada parte del sistema pueda evolucionar sin generar impacto innecesario en las demás. Este enfoque no solo mejora la mantenibilidad, sino que también facilita la incorporación de nuevas funcionalidades en el futuro. La organización del sistema se basa en una arquitectura por capas que incluye la capa de Cliente, donde se ubica la aplicación web, incluyendo sus capacidades de PWA, y la capa de Presentación, encargada de gestionar la interfaz y la comunicación inicial con el backend. A esto se suma la capa de Seguridad, que centraliza los mecanismos de autenticación, autorización y protección de datos. Por su parte, la capa de Aplicación, implementada con Spring Boot, contiene toda la lógica del negocio y se encarga del flujo de información hacia la capa de Datos, responsable del acceso y persistencia. En esta arquitectura, los servicios adicionales que utiliza el sistema, como el envío de correos o la verificación en dos pasos, se integran directamente dentro de la capa de Aplicación. Esto evita complejidades innecesarias en la representación del diseño y mantiene el diagrama coherente y simple, sin dejar de reflejar el funcionamiento real del sistema.

6.6. Descripción de la Arquitectura del Sistema

La arquitectura propuesta organiza el sistema en capas bien definidas, lo que permite mantener una estructura ordenada, escalable y fácil de mantener. Cada capa cumple un rol específico dentro del flujo general de la aplicación.

6.6.1. Capa de Cliente (SPA/PWA)

El frontend funciona como una aplicación web moderna basada en Next.js y diseñada como SPA con capacidades PWA. Desde el navegador, los usuarios interactúan mediante una interfaz rápida, responsiva y adaptable a distintos dispositivos. La aplicación soporta instalación como app y toda la comunicación con el backend se realiza a través de REST.

6.6.2. Capa de Seguridad

Incluye los mecanismos que protegen el acceso al sistema. Se utiliza autenticación basada en JWT almacenado en cookies seguras y un sistema de doble factor (2FA) mediante códigos TOTP compatibles con Google Authenticator. La autorización se gestiona con Spring Security mediante roles como ADMIN, ANALISTA y OBSERVADOR. El frontend también aporta seguridad con middleware que controla el acceso a rutas protegidas. Además, existe un sistema de notificaciones por correo para avisos de seguridad y recuperación de cuentas.

6.6.3. Capa de Presentación (Frontend)

Esta capa engloba la lógica de presentación, los componentes visuales, la validación de formularios y el manejo de estado. Se utilizan herramientas como React Query, React Hook Form, Zod, Radix UI, shadcn/ui y Tailwind. El frontend organiza sus rutas y funcionalidades en módulos bien definidos: autenticación, administración, análisis, reportes, perfil, notificaciones, etc. También ofrece integración con WebSockets para recibir actualizaciones y notificaciones automáticamente e incluye funcionalidades como dashboards interactivos, listados, formularios complejos e instalación como PWA.

6.6.4. Capa de Aplicación (Backend)

El backend está construido con Spring Boot siguiendo el patrón MVC y estructurado en Controllers, Services y Repositories. Los Controllers exponen las APIs, los Services contienen la lógica de negocio y los Repositories gestionan el acceso a la base de datos. Aquí se manejan análisis de

semillas, validaciones, reportes, importación de datos históricos, seguridad, notificaciones y todo el flujo del sistema. También se implementa un canal WebSocket con STOMP para notificaciones en tiempo real, manejo de transacciones y un sistema global de manejo de excepciones.

6.6.5. Capa de Datos

La persistencia está implementada con Spring Data JPA e Hibernate, usando PostgreSQL como base de datos. La base está diseñada con relaciones bien definidas e integridad referencial.

6.7. Descomposición en subsistemas

El sistema se organiza en cinco subsistemas principales que trabajan en conjunto para ofrecer una plataforma robusta, segura y orientada a la experiencia del usuario. Cada uno cumple un rol específico dentro de la solución, y en conjunto conforman una arquitectura coherente y fácil de mantener. A continuación, se detalla cada subsistema y los elementos que lo componen.

6.7.1. Subsistema de Cliente

Representa lo que usa el usuario en su navegador o como PWA instalada.

- Renderiza la interfaz y maneja la experiencia visual e interacción.
- Funciona incluso con conexión inestable
- Administra cookies de sesión y comunicación con el backend vía HTTPS y WebSockets.
- Se encarga de notificaciones push y la instalación como app en dispositivos.

6.7.2. Subsistema de Seguridad

- Gestiona la autenticación con JWT y Cookies HttpOnly.
- Implementa 2FA con TOTP y manejo de dispositivos confiables.
- Controla los permisos según roles: ADMIN, ANALISTA, OBSERVADOR.

- Administra contraseñas, recuperación de acceso y validaciones.
- Incluye middleware que protege rutas y verifica sesiones en el frontend.
- Genera alertas por eventos críticos de seguridad (cambios de clave).

6.7.3. Subsistema de Presentación (Frontend)

- SPA/PWA construida con componentes reutilizables.
- Renderiza pantallas, formularios, tablas, flujos y navegación interna.
- Maneja el estado global de la aplicación.
- Canaliza toda la comunicación hacia el backend mediante servicios API.
- Recibe actualizaciones en tiempo real mediante WebSockets.
- Aplica reglas de visibilidad según rol del usuario.

6.7.4. Subsistema de Aplicación (Backend)

- Contiene la lógica de negocio principal del sistema.
- Expone endpoints REST desde los controllers.
- Ejecuta reglas de negocio, validaciones e integraciones internas.
- Administra transacciones y operaciones complejas de múltiples pasos.
- Gestiona módulos como catálogos, reportes, usuarios, análisis, lotes, etc.
- Envía notificaciones en tiempo real al frontend.
- Devuelve respuestas seguras, consistentes y auditables.

6.7.5. Subsistema de Datos

- Gestiona el acceso a la base de datos PostgreSQL.
- Define las entidades del dominio y su estructura.
- Implementa repositorios JPA para consultas y persistencia.
- Garantiza integridad, consistencia y trazabilidad de la información.
- Registra auditoría automática de creación y modificación de registros.

7. Implementación

7.1. Diagrama de Deployment de UML

El diagrama de Deployment permitió mostrar de manera clara cómo quedó distribuida la arquitectura física del sistema una vez implementado. Allí se identificaron los distintos nodos de hardware y software que intervinieron en la solución, así como la forma en que se organizaron los componentes dentro de la infraestructura disponible. El sistema se desplegó en un servidor virtual de Amazon Web Services (AWS), utilizando una instancia EC2 configurada para ejecutar contenedores Docker. En este entorno se alojaron los servicios del frontend, el backend y la base de datos, los cuales se comunicaron a través de una red interna propia del host. El diagrama también reflejó la forma en que los usuarios accedieron a la aplicación, ya fuera desde un navegador web o desde la versión móvil como PWA. En ambos casos, la conexión se estableció a través de Internet hacia la dirección pública del servidor EC2. Gracias a esta representación fue posible visualizar de manera sintética cómo se relacionaron los distintos elementos de la solución y cómo se organizó el entorno de despliegue, lo que sirvió como punto de partida para el análisis detallado de las tecnologías utilizadas en el backend y el frontend.

7.2. Tecnologías a utilizar

Las tecnologías empleadas en el proyecto se seleccionaron en función de los requerimientos funcionales, el rendimiento esperado y la necesidad de contar con una arquitectura flexible y mantenible. Se optó por una combinación de herramientas modernas y ampliamente utilizadas en la industria, lo que permitió asegurar un entorno estable durante el desarrollo y el despliegue. La infraestructura se apoyó en contenedores Docker ejecutados sobre una instancia EC2 de AWS, lo que facilitó la portabilidad, el aislamiento y la gestión individual de cada servicio del sistema. Además, se adoptaron tecnologías que ofrecieron buen soporte para el trabajo con APIs, gestión de datos, desarrollo de interfaces web y compatibilidad con PWA. Esta selección permitió construir una solución sólida y coherente, manteniendo una estructura clara entre la capa de presentación, la capa de negocio y la capa de persistencia.

7.3. Frontend

El frontend se construyó con Next.js y TypeScript, una combinación que ofreció un entorno estable para desarrollar interfaces modernas y con buen rendimiento. Next.js permitió implementar renderizado híbrido (SSR/CSR) y optimizaciones automáticas que mejoraron la experiencia del usuario, mientras que TypeScript aportó tipado estático, ayudando a reducir errores y mantener el código más organizado. La aplicación también se adaptó como Progressive Web App (PWA), lo que permitió que pudiera instalarse en dispositivos móviles y funcionar con mayor fluidez en entornos donde la conectividad podía ser limitada. Al igual que los demás componentes, el frontend se ejecutó dentro de un contenedor Docker, lo que facilitó su despliegue y mantuvo la coherencia del entorno de ejecución.

7.4. Backend

El backend se desarrolló utilizando Java junto con el framework Spring Boot, una tecnología que brindó un conjunto amplio de herramientas para construir servicios web robustos y escalables. Spring Boot permitió organizar la lógica de negocio mediante controladores, servicios y reposito-

rios, además de facilitar la configuración de la API mediante estándares REST. Esta elección favoreció la claridad en la estructura del proyecto y simplificó la integración con el resto de los componentes del sistema. La comunicación con la base de datos se realizó a través de JPA/Hibernate, lo que permitió manejar las entidades del dominio de forma eficiente y reducir la carga de escribir consultas SQL manuales. El servicio funcionó dentro de un contenedor Docker, lo que facilitó su despliegue en la instancia EC2 y garantiza un entorno consistente durante todo el ciclo de vida del proyecto.

7.5. Horas Dedicadas

A continuación se detallarán los registros de horas por etapa del proyecto. Estos fueron gestionados utilizando un proyecto en Toggl.

7.5.1. Etapa 1: Documentación y Análisis

Período: Desde el 14 de agosto al 08 de septiembre de 2025. La primera etapa se enfocó principalmente en recolectar la información pertinente con las clientes y documentar las características inicialmente establecidas para comenzar con el desarrollo adecuado de la aplicación web. Un total de 60 horas fueron dedicadas a esta fase.

7.5.2. Etapa 2: Desarrollo

Período: Desde el 08 de septiembre al 08 de noviembre de 2025. La etapa de desarrollo se dedicó enteramente al desarrollo completo de la aplicación web, cubriendo los requerimientos solicitados y validando con las clientes. Un total de 553 horas fueron dedicadas a esta fase.

7.5.3. Etapa 3: Testing, Documentación Final y Presentación

Período: Desde el 08 de noviembre al 30 de noviembre de 2025. La etapa final del proyecto se enfocó en la realización de pruebas funcionales y de usabilidad, así como en la corrección de

errores detectados. Además, se completó la documentación técnica y se preparó la presentación final del proyecto. Un total de ?? horas fueron dedicadas a esta fase.

8. Gestión de Proyecto

Para la ejecución de este proyecto no se estructuraron roles específicos para cada integrante del equipo. Si bien cada miembro tenía mayor afinidad con determinadas áreas tecnológicas, se decidió adoptar una dinámica de trabajo colaborativa en la que todos participaron tanto en el desarrollo del frontend como el backend, así como en tareas vinculadas a documentación, testing y despliegue. Para la organización del trabajó se creó un proyecto en Notion, donde se registraron las tareas principales, los responsables y las estimaciones de tiempo para cada tarea. El principal medio de comunicación en línea fue Google Meet, utilizado para reuniones virtuales, donde a su vez se llevaron a cabo reuniones periódicas con el cliente. Además, se creó un grupo de WhatsApp para coordinar rápidamente temas urgentes, notificaciones internas y actualizaciones cortas entre los integrantes del equipo, lo que permitió una comunicación ágil y continua.

8.1. Entorno de Desarrollo

El desarrollo del sistema se realizó en un entorno orientado a aplicaciones web modernas, priorizando la reproducibilidad, la trazabilidad y la correcta separación de responsabilidades entre los distintos componentes. La arquitectura full-stack del proyecto se implementó utilizando tecnologías ampliamente adoptadas en la industria, con el objetivo de asegurar buenas prácticas de ingeniería, facilitar el mantenimiento y permitir la futura escalabilidad del sistema. Para el desarrollo del software se utilizaron dos entornos de desarrollo integrados (IDE):

- Visual Studio Code, empleado principalmente para el frontend y configurado con extensiones específicas para TypeScript, React, Docker y herramientas de control de versiones.
- IntelliJ IDEA, utilizado para el backend en Java, aprovechando su soporte avanzado para Spring Boot, gestión de dependencias con Maven y depuración integrada.

La gestión del código fuente se realizó utilizando Git como sistema de control de versiones y GitHub como plataforma principal de alojamiento, revisión y seguimiento del proyecto. GitHub permitió manejar ramas, realizar revisiones de código, gestionar issues y mantener un flujo de trabajo organizado basado en buenas prácticas de versionado. Esta combinación permitió a todos los integrantes del equipo trabajar de forma eficiente y con herramientas adecuadas para cada tecnología.

8.2. Tecnologías utilizadas

8.2.1. Cliente

El cliente del sistema fue diseñado como una aplicación web dinámica utilizando Next.js 14 y React 18, tecnologías ampliamente adoptadas en entornos productivos debido a su eficiencia, modularidad y soporte comunitario. La estructura se organizó siguiendo las convenciones del App Router de Next.js, lo cual permitió una clara separación entre páginas, componentes y módulos de lógica, promoviendo escalabilidad y mantenibilidad. Para el desarrollo de la interfaz de usuario se utilizaron Tailwind CSS, Radix UI y Shadcn. Estas tecnologías permitieron crear una experiencia visual coherente y accesible, con componentes reutilizables y mantenibles, alineados con estándares modernos de diseño. La comunicación entre el frontend y el backend se realizó mediante APIs REST para la mayor parte de las operaciones, mientras que las funcionalidades en tiempo real se implementaron a través de WebSocket, permitiendo la recepción de notificaciones instantáneas y mejorando la experiencia interactiva del usuario. El aseguramiento de calidad del frontend incluyó la ejecución de pruebas automatizadas mediante Jest y React Testing Library, herramientas ampliamente utilizadas para validar el comportamiento de componentes y flujos de interacción.

8.2.2. Servidor

El backend se implementó utilizando Java 21 con Spring Boot 3.5, aplicando una arquitectura en capas que divide la lógica de negocio, los servicios, los controladores web, los repositorios de datos y las configuraciones transversales. Esta estructura facilita el mantenimiento, reduce el

acoplamiento y mejora la estabilidad del sistema.

8.2.3. Servicios y API

La API del sistema fue documentada utilizando OpenAPI/Swagger, permitiendo que tanto desarrolladores como actores externos comprendan las especificaciones de cada endpoint, formatos de datos, tipos de respuestas y códigos de error. El backend expone servicios REST y un canal de comunicación en tiempo real mediante WebSocket (STOMP), empleado para enviar notificaciones y eventos relevantes sin necesidad de realizar consultas repetitivas al servidor.

8.2.4. Persistencia y Base de Datos

La aplicación utiliza PostgreSQL 15, gestionado mediante Spring Data JPA, lo cual permitió automatizar tareas de persistencia y reducir la necesidad de escribir consultas SQL manuales. Durante el proceso de pruebas se utilizó H2, una base de datos en memoria que permite ejecutar los tests de forma rápida, aislada y sin requerir infraestructura externa.

8.2.5. Seguridad

La seguridad del sistema se abordó mediante:

- Spring Security para la protección de endpoints y control de roles.
- JWT para la gestión de sesiones sin estado.
- Autenticación en dos pasos (2FA) mediante códigos TOTP, fortaleciendo la seguridad del acceso.

Este conjunto de herramientas permitió cumplir buenas prácticas de seguridad alineadas con estándares modernos.

8.2.6. Herramientas de Construcción, Testing y DevOps

El backend fue compilado y gestionado mediante Maven, lo que permitió automatizar tareas de instalación, test y empaquetado. La calidad del código se reforzó mediante pruebas unitarias utilizando JUnit 5 y Mockito, enfocadas en validar la lógica de negocio, la interacción entre módulos y el correcto comportamiento de los componentes críticos. Finalmente, el uso de Docker y Docker Compose permitió ejecutar el proyecto de forma completa (frontend, backend y base de datos) mediante entornos aislados y reproducibles, asegurando coherencia en las diferentes etapas del desarrollo.

9. Problemas Encontrados

9.1. Reestructuración del Módulo de Germinación

Durante la etapa final del desarrollo se identificó un problema crítico en el módulo de germinación, que obligó a replantear su estructura, lógica interna y funcionamiento general. Este inconveniente surgió a raíz de una discrepancia significativa entre los requisitos originalmente planteados y las necesidades reales del laboratorio. La dificultad se originó en la documentación inicial proporcionada por el cliente, compuesta por hojas de cálculo con información incompleta, las cuales eran utilizadas por el equipo para realizar la gestión de forma manual. A partir de la referencia el equipo asumió que todas las fechas de conteo eran comunes a todo el análisis de germinación y que las diferentes configuraciones de días de prefrío ingresadas en el mismo análisis no influían en las mismas, entre otras. Estas suposiciones llevaron a diseñar el módulo bajo una estructura que no reflejaba la complejidad real del proceso.

9.2. Cambio de Requisitos

Pocos días antes de la finalización de la etapa de desarrollo, el cliente aclaró que el funcionamiento real del análisis de germinación difería de lo inicialmente interpretado. Los requisitos correctos incluían configuraciones múltiples con fechas de conteos independientes, restringidas

por los días de prefrío y con validaciones específicas adicionales. Este cambio presentó un ajuste conceptual profundo respecto a la estructura inicial del módulo, y su corrección requirió una reingeniería completa del módulo, afectando múltiples áreas del sistema. Entre las acciones necesarias se incluyó el rediseño de la estructura de los datos, reescritura de la lógica del sistema y modificación de interfaces. A pesar de las restricciones de tiempo, el equipo logró implementar correctamente la nueva estructura, la cual resultó ser fundamental para asegurar que el módulo cumpliera adecuadamente con los cambios solicitados por el cliente.

10. Testing de la Aplicación Desarrollada

10.1. Estrategia de Pruebas

10.2. Pruebas Unitarias

10.3. Pruebas de Integración

10.4. Pruebas End-to-End (E2E)

10.5. Automatización y CI

10.6. Resultados y Cobertura

11. La Solución Desarrollada

11.1. Objetivos de la Solución

11.1.1. Gestión de Lotes de Semillas

11.1.2. Análisis de Calidad

11.1.3. Pureza Física

11.1.4. Germinación

11.1.5. DOSN (Determinación de Otras Semillas en Número)

11.1.6. PMS (Peso de Mil Semillas)

11.1.7. Tetrazolio

11.1.8. Flujo de Trabajo del Laboratorio

11.1.9. Reportes y Exportaciones

12. Conclusiones

13. Trabajos a Futuro

13.1. Módulo de auditorías y control de calidad