Sistema de Gestión de Laboratorio de Semillas (SGLS)

Nadia Gorría, Nahuel Perdomo & Milagros Vairo.

Tutor: Nicolás Escobar

Proyecto Final de la Carrera Tecnólogo en Informática

Diciembre 2025

Universidad Tecnológica del Uruguay

Sede: San José de Mayo

UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Tabla de contenidos

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.

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 en Informática, aborda la necesidad de unificar y estandarizar procedimientos técnicos que anteriormente se gestionaban mediante planillas Excel.

El sistema permite administrar lotes y diversos análisis, como Germinación, Tetrazolio, PMS (Peso de Mil Semillas), DOSN (Determinación de Otras Semillas en Número) 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 (RFC 7519) 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, ofreciendo una PWA (Progressive Web App) optimizada y con un buen rendimiento.

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.

1. 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[14]. 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[15, 6]. 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 en Informática, 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.

2. 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 INIA, 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. [14, 15, 6]

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.

Durante la selección de la pila tecnológica se tuvieron en cuenta las referencias y documentación de los proyectos y herramientas usadas [29, 35], así como la necesidad de contar con buenas prácticas y herramientas de testing y cobertura (por ejemplo JUnit y JaCoCo) para garantizar calidad y confiabilidad del software [18, 5].

2.1. 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.

3. Marco Teórico

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.

[14, 15, 6]

El proyecto desarrollado para el 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 capítulo 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.

[14]

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.

3.1. 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.

[15, 6] 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.

3.2. 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.

3.2.1. 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.

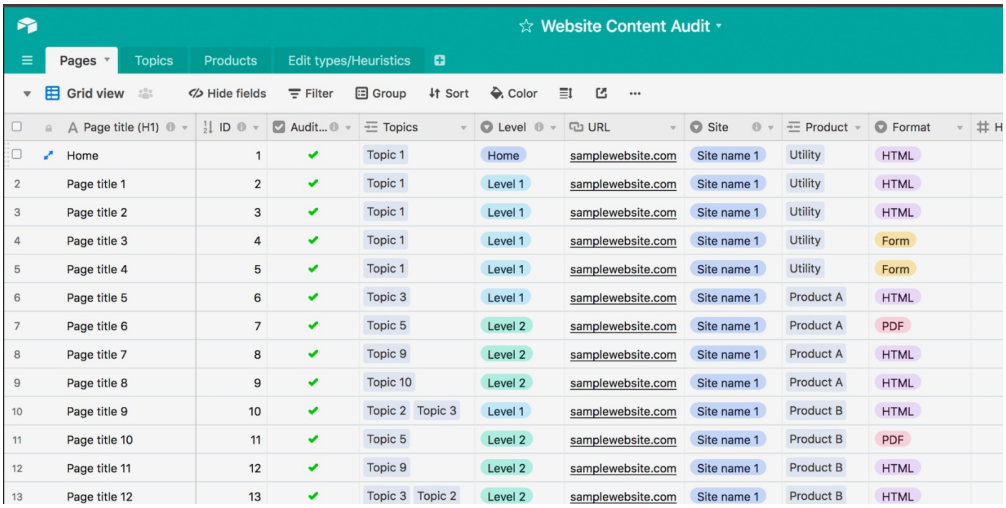
Ventajas:

- 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

[1]



	Page title (H1)	ID	Audit...	Topics	Level	URL	Site	Product	Format
	Home	1	✔	Topic 1	Home	samplewebsite.com	Site name 1	Utility	HTML
2	Page title 1	2	✔	Topic 1	Level 1	samplewebsite.com	Site name 1	Utility	HTML
3	Page title 2	3	✔	Topic 1	Level 1	samplewebsite.com	Site name 1	Utility	HTML
4	Page title 3	4	✔	Topic 1	Level 1	samplewebsite.com	Site name 1	Utility	Form
5	Page title 4	5	✔	Topic 1	Level 1	samplewebsite.com	Site name 1	Utility	Form
6	Page title 5	6	✔	Topic 3	Level 1	samplewebsite.com	Site name 1	Product A	HTML
7	Page title 6	7	✔	Topic 5	Level 2	samplewebsite.com	Site name 1	Product A	PDF
8	Page title 7	8	✔	Topic 9	Level 2	samplewebsite.com	Site name 1	Product A	HTML
9	Page title 8	9	✔	Topic 10	Level 2	samplewebsite.com	Site name 1	Product A	HTML
10	Page title 9	10	✔	Topic 2, Topic 3	Level 1	samplewebsite.com	Site name 1	Product B	HTML
11	Page title 10	11	✔	Topic 5	Level 2	samplewebsite.com	Site name 1	Product B	PDF
12	Page title 11	12	✔	Topic 9	Level 2	samplewebsite.com	Site name 1	Product B	HTML
13	Page title 12	13	✔	Topic 3, Topic 2	Level 2	samplewebsite.com	Site name 1	Product B	HTML

Figura 1: Ejemplo: vista de contenido en Airtable (tabla/registro).

3.2.2. 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

[28]

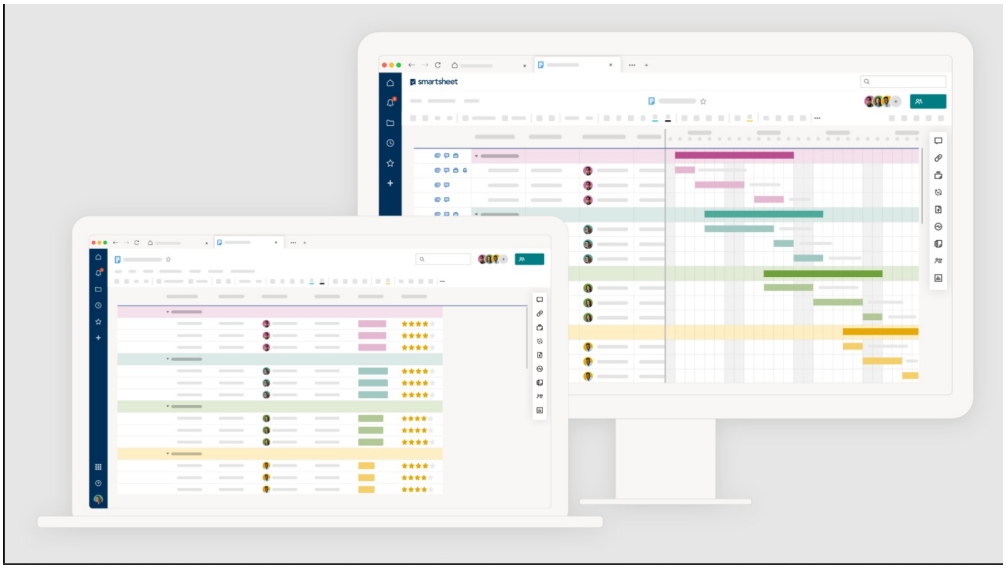


Figura 2: Ejemplo: interfaz de Smartsheet mostrando una vista tipo hoja de cálculo y reportes.

4. 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. [14, 15, 6]

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. [14]

4.1. 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 y asegurarse de que el sistema reflejase fielmente la forma en que se trabaja en el laboratorio.

4.2. 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.

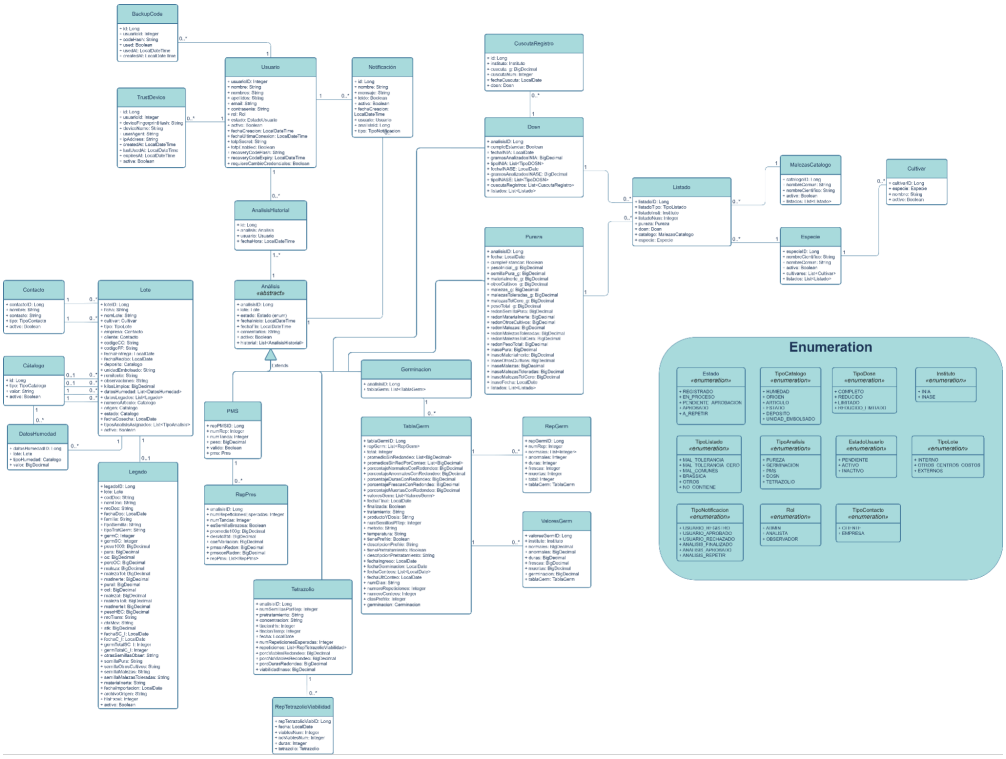


Figura 3: Modelo de dominio: entidades y relaciones principales.

4.3. 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.

4.4. Diagrama de Casos de Uso

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

4.5. 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.

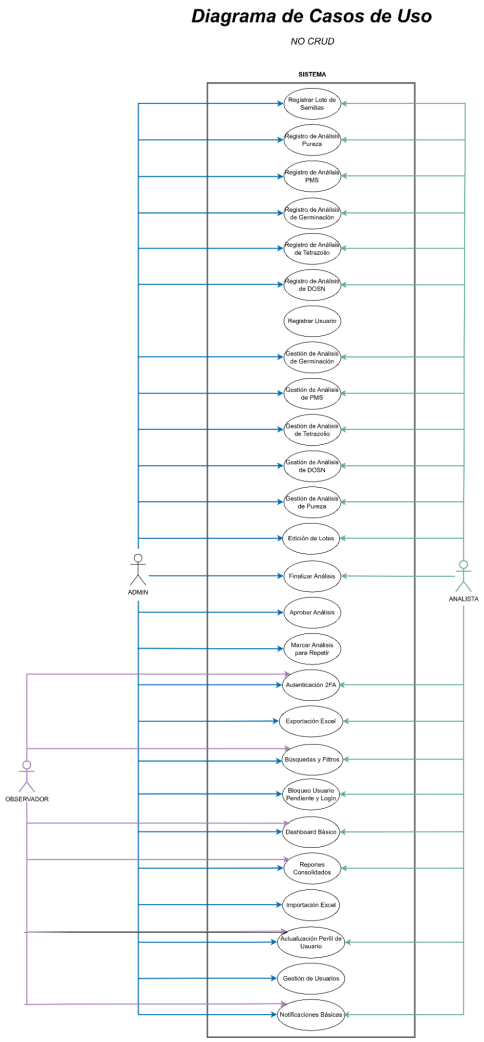


Figura 4: Diagrama de casos de uso: principales interacciones entre actores y el sistema.

La organización del sistema se basa en una arquitectura por capas que incluye la capa de Cliente, donde se ubica la aplicación web, y la capa de Presentación, encargada de gestionar la interfaz y la comunicación inicial con el backend. En esta capa se incorporan además las capacidades PWA (Progressive Web App), lo que permite que la aplicación web se comporte como una aplicación nativa sin necesidad de instalarse desde una tienda. 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.

4.6. 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.

4.6.1. Capa de Cliente (SPA/PWA)

El frontend funciona como una aplicación web moderna basada en Next.js y diseñada como SPA (Single Page Application) 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. [35]

4.6.2. 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

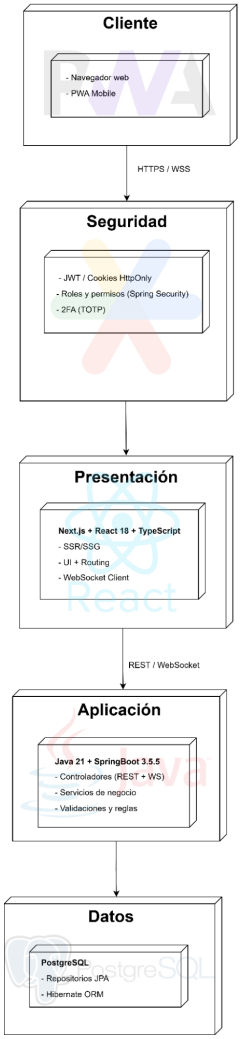


Figura 5: Diagrama de arquitectura: capas y componentes principales del sistema.

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. 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.

4.6.3. 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[32, 26, 3, 25, 27, 31]. 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. También ofrece integración con WebSockets para recibir actualizaciones y notificaciones automáticamente e incluye funcionalidades como *dashboards* interactivos, listados, formularios complejos e instalación como PWA.

4.6.4. 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. [29, 11]

4.6.5. 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.

4.7. 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.

4.7.1. 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.
- 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.

4.7.2. Subsistema de Seguridad

- Gestiona la autenticación con JWT y Cookies HttpOnly.
- 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.
- Incluye middleware que protege rutas y verifica sesiones en el frontend.
- Genera alertas por eventos críticos de seguridad (cambios de clave).

4.7.3. Subsistema de Presentación (Frontend)

- SPA/PWA construida con componentes reutilizables.
- 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.
- 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.

4.7.4. 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.

4.7.5. 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.

5. Implementación

5.1. 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. [14, 2]

5.2. Tecnologías utilizadas

5.2.1. 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. [35]

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

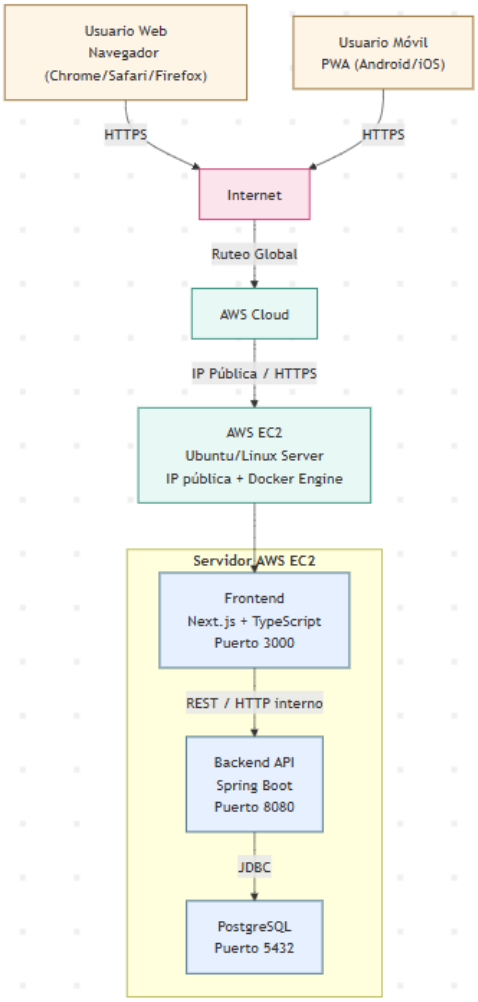


Figura 6: Diagrama de *deployment* (UML) que muestra la distribución de nodos y contenedores en la infraestructura de despliegue.

reutilizables y mantenibles, alineados con estándares modernos de diseño. [31, 25, 27]

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. [16, 33]

5.2.2. 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. [29]

5.2.3. 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. [23]

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.

5.2.4. 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. [24]

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.

5.2.5. 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.

[30, 13, 12]

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

5.2.6. 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. [18, 5]

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. [4]

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.[10] 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.

5.3. Horas Dedicadas

A continuación se detallarán los registros de horas por etapa del proyecto. Estos fueron gestionados utilizando un proyecto en Toggl. [34]

Etapa	**Horas**
Etapa 1	60
Etapa 2	553
Etapa 3	??

5.3.1. 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.

5.3.2. 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.

5.3.3. 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.

6. 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. [14]

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. [22]

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. [9, 19]

6.1. 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. [4]

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. [20]
- IntelliJ IDEA, utilizado para el backend en Java, aprovechando su soporte avanzado para Spring Boot, gestión de dependencias con Maven y depuración integrada. [17]

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. [7, 8]

Esta combinación permitió a todos los integrantes del equipo trabajar de forma eficiente y con herramientas adecuadas para cada tecnología.

7. Problemas Encontrados

7.1. 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.

[14, 15]

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.

[6, 15]

Estas suposiciones llevaron a diseñar el módulo bajo una estructura que no reflejaba la complejidad real del proceso.

7.1.1. 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. [14]

8. Testing de la Aplicación Desarrollada

8.1. Testing General del Proyecto

El testing de la aplicación se llevó adelante directamente por quienes trabajaron en el proyecto, ya que no se contó con un equipo de pruebas dedicado. Las verificaciones se fueron realizando a medida que se sumaban nuevas funcionalidades, lo que permitió detectar errores con rapidez y corregirlos antes de avanzar con el resto del desarrollo.

8.2. Pruebas Manuales Iniciales

Cada funcionalidad fue probada primero de manera manual, revisando su comportamiento desde la interfaz o enviando solicitudes a la API según correspondiera. Una vez que el sistema estuvo completo, se hizo una revisión general de todos los casos de uso para asegurarse de que los flujos funcionaran de forma coherente y sin generar fallos entre distintas partes de la aplicación.

8.3. Pruebas en Backend (Spring Boot)

En el backend, desarrollado con Spring Boot, se implementaron pruebas unitarias e integraciones utilizando JUnit, Mockito y la herramienta de cobertura JaCoCo. Esto permitió verificar la lógica de negocio, los controladores y varios de los servicios internos. La cobertura superó el 80 %, un valor adecuado para tener un buen nivel de confianza en el comportamiento de las áreas más importantes del código. [29, 18, 21, 5]

8.4. Pruebas en Frontend (Next.js + TypeScript)

El frontend, construido con Next.js y TypeScript, también incorporó pruebas automatizadas mediante Jest y React Testing Library. Estas pruebas abarcaron componentes, funciones auxiliares y ciertos elementos de la lógica de presentación. Al igual que en el backend, el nivel de cobertura superó el 80 %, lo que ayudó a detectar errores de visualización, manejo de datos y algunos detalles que surgieron durante la integración con la API. [35, 16, 33]

8.5. Pruebas Manuales en Web y PWA

Además de estas pruebas automatizadas, se realizaron pruebas manuales tanto en la versión web como en la PWA. En esta etapa se revisó la navegación completa, el funcionamiento en dispositivos móviles, la instalación de la aplicación y los flujos principales de interacción desde el punto de vista del usuario final.

8.6. Pruebas *End-to-End* (E2E)

Por último, se realizaron pruebas *end-to-end* (E2E) para evaluar el funcionamiento del sistema completo, desde la interfaz hasta la base de datos. Este tipo de testing se centró en reproducir escenarios reales de uso, verificando que los distintos componentes interactúan de forma correcta y continua. Las pruebas E2E permitieron validar flujos completos como el inicio de sesión, la carga y consulta de datos, la comunicación con la API y la correcta actualización de la información en la base de datos. Este enfoque fue especialmente útil para detectar errores que no aparecen en pruebas aisladas, como problemas de integración, diferencias en el formato de los datos o comportamientos inesperados al combinar varias funcionalidades en un mismo flujo.

8.7. Conclusión

Gracias a esta combinación de pruebas unitarias, integradas, automatizadas, manuales y E2E, se logró obtener un sistema estable, con un buen nivel de calidad y sin fallos críticos al finalizar el desarrollo.

9. La Solución Desarrollada

El Sistema de Gestión de Análisis de Semillas desarrollado para INIA es una plataforma web integral que digitaliza y centraliza todo el proceso de análisis de calidad de semillas, reemplazando el manejo fragmentado que anteriormente se realizaba en múltiples planillas de Excel. La solución permite registrar lotes que ingresan al laboratorio, realizar distintos tipos de análisis especializados y generar reportes estandarizados para INIA e INASE con mayor precisión, trazabilidad y control. [14, 29, 35]

9.1. Objetivos de la Solución

La solución tuvo como objetivos principales:

- Centralizar la información de todos los análisis en un único sistema accesible desde cualquier dispositivo.
- Estandarizar el registro de los lotes de semillas y de los resultados obtenidos en cada análisis.
- Reducir errores eliminando la manipulación manual de múltiples planillas.
- Aumentar la trazabilidad mediante estados de avance y registro estructurado de datos.
- Mejorar la eficiencia del laboratorio, automatizando validaciones, cálculos y generación de reportes.
- Facilitar el cumplimiento normativo, generando reportes compatibles con los requerimientos de INIA e INASE.
- Asegurar el acceso controlado, mediante roles de usuario y autenticación JWT. [13]

9.1.1. Gestión de Lotes de Semillas

- Registro de lotes ingresados al laboratorio con datos como especie, variedad, empresa, kilos y humedad.

- Visualización y edición de la información del lote.
- Asociación automática de lotes a sus respectivos análisis.

9.1.2. Análisis de Calidad

El sistema permite realizar distintos análisis obligatorios en los laboratorios de semillas:

9.1.2.1. Pureza Física

- Registro de porcentajes de semilla pura, materia inerte, otros cultivos y malezas.
- Cálculos automáticos de proporciones y verificaciones según normas vigentes.

9.1.2.2. Germinación

- Gestión de múltiples repeticiones por muestra.
- Registro de conteos en diferentes días.
- Cálculo automático de germinación final.

9.1.2.3. DOSN (Determinación de Otras Semillas en Número)

- Identificación y registro de semillas de malezas peligrosas.
- Validación automática de especies críticas.

9.1.2.4. PMS (Peso de Mil Semillas)

- Registro de repeticiones y cálculo del peso promedio.
- Indicadores automáticos de calidad del lote.

9.1.2.5. Tetrazolio

- Gestión de múltiples repeticiones por muestra
- Determinación de viabilidad mediante registro visual y categorización.

9.1.3. Flujo de Trabajo del Laboratorio

- Manejo de estados para cada análisis: EN_PROCESO $\rightarrow$ FINALIZADO $\rightarrow$ APROBADO.
- Control de modificaciones según rol del usuario.
- Historial de avances y validaciones.

9.1.4. Reportes y Exportaciones

- Generación automática de reportes en Excel con las 52 columnas requeridas por INIA e INASE.
- Integración con datos del lote, resultados y observaciones.
- Formatos estandarizados y consistentes para auditorías y trazabilidad. [23, 24, 4]

10. Conclusiones

El desarrollo del Sistema de Gestión de Laboratorio de Semillas (SGLS) para INIA Uruguay representa un avance clave en la modernización de los procesos de control de calidad. El proyecto surge para sustituir flujos de trabajo manuales basados en planillas independientes, que generaban dispersión de datos, dificultades en la trazabilidad y una alta dependencia del registro humano. Estas limitaciones afectaban la eficiencia diaria del laboratorio y podían comprometer la confiabilidad de los resultados, por lo que avanzar hacia un sistema centralizado y digital se volvió una necesidad estratégica. [14, 15]

Con la implementación del SGLS, se logró integrar en un único entorno todas las etapas del análisis de semillas, desde la recepción de muestras hasta la emisión de informes finales. Este enfoque unificado permitió eliminar la duplicación de datos, mejorar la trazabilidad de cada lote y asegurar un seguimiento claro y preciso a lo largo de todo el proceso. La automatización de cálculos complejos y la validación de información redujo considerablemente los errores humanos, aportando mayor consistencia y confianza en los resultados generados. Esto no solo agiliza el trabajo interno, sino que también fortalece la calidad de los servicios ofrecidos por el laboratorio. [29, 35, 23, 24]

Finalmente, el sistema no solo cumple con los requerimientos actuales del proceso de certificación, sino que establece una base sólida para el crecimiento futuro. Su diseño organizado y adaptable permite incorporar nuevos módulos, integrar equipamiento automatizado y ampliar las capacidades analíticas y de reporte que el laboratorio pueda necesitar más adelante. En conjunto, el SGLS posiciona a INIA como una institución alineada con las demandas modernas del sector agropecuario, demostrando cómo la digitalización puede potenciar la productividad, garantizar la calidad y respaldar la toma de decisiones con datos confiables. [4]

11. Trabajos a Futuro

11.1. Módulo de auditorías y control de calidad

Si bien el sistema registra información detallada de cada análisis, podría incorporarse un módulo específico para auditorías externas e internas más específicas. Este módulo permitiría revisar trazabilidad histórica, validar procedimientos y generar informes automáticos para procesos de certificación, algo especialmente útil cuando se trabaja en conjunto con INASE u otras instituciones reguladoras. [15, 6, 14]

11.2. Gestión documental avanzada

Actualmente se generan reportes en Excel, pero no existe un repositorio interno de documentos. A futuro podría añadirse un módulo que permita almacenar protocolos, fotos de muestras, resultados complementarios, certificados y archivos adjuntos por lote. Esto facilitaría consultas históricas y el intercambio de información entre analistas. [22]

11.3. Automatización de cálculos y validaciones normativas

Aunque el sistema ya automatiza buena parte de los cálculos, todavía existen validaciones que podrían formalizarse según normativas específicas de INASE o reglas internacionales de análisis de semillas. Incluir este módulo permitiría evitar inconsistencias y reducir posibles errores humanos en los análisis más complejos. [15, 6]

11.4. Optimización del flujo de trabajo

Actualmente el sistema maneja estados básicos de los análisis (REGISTRADO (en algunos casos) $\rightarrow$ EN_PROCESO $\rightarrow$ FINALIZADO $\rightarrow$ APROBADO). A futuro podría añadirse un motor más completo de *workflow* que permita definir etapas personalizadas, responsables por análisis, plazos estimados y automatización de pasos repetitivos. Esto ayudaría a estandarizar procesos entre distintos laboratorios. [22, 23]

11.5. Integración con sistemas externos

En una fase posterior podría evaluarse la integración con sistemas institucionales externos utilizados por INIA o INASE, como portales de registro, bases de datos oficiales o plataformas de trazabilidad. Esto reduciría la duplicación de información y agilizaría los procesos administrativos. [23, 24]

Referencias

[1] Airtable. Airtable – spreadsheet-database hybrid for collaboration. https://airtable.com/, 2024. Recuperado: 26 noviembre 2025.

[2] Amazon Web Services (AWS). Amazon ec2 y servicios relacionados – infraestructura en la nube. https://aws.amazon.com/, 2024. Recuperado: 26 noviembre 2025.

[3] Colinhacks. Zod – typescript-first schema declaration and validation library. https://github.com/colinhacks/zod, 2024. Recuperado: 26 noviembre 2025.

[4] Docker, Inc. Docker – contenedores para desarrollo y despliegue. https://www.docker.com/, 2024. Recuperado: 26 noviembre 2025.

[5] EclEmma / JaCoCo. Biblioteca jacoco para cobertura de código java. https://www.jacoco.org/jacoco/, 2023. Recuperado: 26 noviembre 2025.

[6] Food and Agriculture Organization (FAO). Guías para el control de calidad de semillas. http://www.fao.org, 2014. Recuperado: 26 noviembre 2025.

[7] Git Community. Git – sistema de control de versiones distribuido. https://git-scm.com/, 2024. Recuperado: 26 noviembre 2025.

[8] GitHub, Inc. Github – plataforma para alojamiento y revisión de código. https://github.com/, 2024. Recuperado: 26 noviembre 2025.

[9] Google LLC. Google meet – videoconferencias y reuniones en línea. https://meet.google.com/, 2024. Recuperado: 26 noviembre 2025.

[10] Hibernate Community. Hibernate orm – marco de mapeo objeto-relacional para java. https://hibernate.org/orm/, 2024. Recuperado: 26 noviembre 2025.

[11] IETF. Protocolo websocket – rfc 6455. https://datatracker.ietf.org/doc/html/rfc6455, 2011. Recuperado: 26 noviembre 2025.

[12] IETF. Time-based one-time password algorithm (totp) – rfc 6238. https://datatracker.ietf.org/doc/html/rfc6238, 2011. Recuperado: 26 noviembre 2025.

[13] IETF. Token web json (jwt) – rfc 7519. https://datatracker.ietf.org/doc/html/rfc7519, 2015. Recuperado: 26 noviembre 2025.

[14] Instituto Nacional de Investigación Agropecuaria (INIA). Inia – sitio oficial. https://www.inia.uy, 2025. Recuperado: 26 noviembre 2025.

[15] International Seed Testing Association (ISTA). *Reglas internacionales para el análisis de semillas*, 2023. Disponible en https://www.seedtest.org. Recuperado: 26 noviembre 2025.

[16] Jest Team. Jest – framework de pruebas para javascript. https://jestjs.io/, 2024. Recuperado: 26 noviembre 2025.

[17] JetBrains. Intellij idea – entorno de desarrollo integrado para java. https://www.jetbrains.com/idea/, 2024. Recuperado: 26 noviembre 2025.

[18] JUnit Team. Guía de usuario de junit 5. https://junit.org/junit5/docs/current/user-guide/, 2023. Recuperado: 26 noviembre 2025.

[19] Meta Platforms. Whatsapp – mensajería de equipo y comunicaciones. https://www.whatsapp.com/, 2024. Recuperado: 26 noviembre 2025.

[20] Microsoft. Visual studio code – editor de código fuente. https://code.visualstudio.com/, 2024. Recuperado: 26 noviembre 2025.

[21] Mockito Team. Mockito – marco para realizar mocks y pruebas unitarias en java. https://site.mockito.org/, 2023. Recuperado: 26 noviembre 2025.

[22] Notion Labs. Notion – gestión de proyectos y documentación colaborativa. https://www.notion.so/, 2024. Recuperado: 26 noviembre 2025.

[23] OpenAPI Initiative. Especificación openapi / swagger – documentación de apis. https://www.openapis.org/, 2023. Recuperado: 26 noviembre 2025.

[24] PostgreSQL Global Development Group. Postgresql – sistema de gestión de bases de datos relacional. https://www.postgresql.org/, 2024. Recuperado: 26 noviembre 2025.

[25] Radix Labs. Radix ui – componentes accesibles para interfaces web. https://www.radix-ui.com/, 2024. Recuperado: 26 noviembre 2025.

[26] React Hook Form Team. React hook form – performant, flexible and extensible forms for react. https://react-hook-form.com/, 2024. Recuperado: 26 noviembre 2025.

[27] shadcn. shadcn/ui – conjunto de componentes y utilidades para react. https://ui.shadcn.com/, 2024. Recuperado: 26 noviembre 2025.

[28] Smartsheet Inc. Smartsheet – work management and collaboration platform. https://www.smartsheet.com/, 2024. Recuperado: 26 noviembre 2025.

[29] Spring Team. Documentación de referencia de spring boot. https://docs.spring.io/spring-boot/docs/current/reference/html/, 2024. Recuperado: 26 noviembre 2025.

[30] Spring Team. Spring security – marco para seguridad en aplicaciones java. https://spring.io/projects/spring-security, 2024. Recuperado: 26 noviembre 2025.

[31] Tailwind Labs. Tailwind css – framework de utilidades para diseño. https://tailwindcss.com/, 2024. Recuperado: 26 noviembre 2025.

[32] TanStack / React Query. React query / tanstack query – data fetching and caching for react. https://tanstack.com/query/latest, 2024. Recuperado: 26 noviembre 2025.

[33] Testing Library. React testing library – pruebas para componentes react. https://testing-library.com/docs/react-testing-library/intro/, 2024. Recuperado: 26 noviembre 2025.

[34] Toggl Track. Toggl track – gestión del tiempo y registro de horas. `https://toggl.com/track`, 2024. Recuperado: 26 noviembre 2025.

[35] Vercel / Next.js Team. Documentación de next.js. `https://nextjs.org/docs/`, 2024. Recuperado: 26 noviembre 2025.