PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC

DESARROLLO DE UN SISTEMA AUTOMATIZADO PARA LA GENERACIÓN DE DOCUMENTACIÓN EN EL LABORATORIO ENVIRONALAB

Proyecto de titulación previo a la obtención del título de: Tecnólogo Superior en Desarrollo de Software

Autor:

Michael Anthony Alejandro Muquis

Alejandro Paul Barrionuevo Caiza

Tutor:

Patricio Omar Alvear Granda

Quito, Ecuador

Dedicatoria

Alejandro Barrionuevo:

Este trabajo está dedicado con sincero agradecimiento a mi familia, pilar fundamental en mi vida y formación. A mis padres, por inculcarme desde temprana edad los valores del compromiso, la honestidad y el esfuerzo; su apoyo incondicional ha sido clave para alcanzar esta meta. A mi padre, por enseñarme con el ejemplo el significado del trabajo constante y la responsabilidad. A mi madre, por brindarme siempre su amor, fortaleza y comprensión en los momentos en que más lo necesité. A mis hermanos, quienes con su cercanía, alegría y compañía me dieron la motivación necesaria para no rendirme y seguir adelante. Agradezco también a mis amigos Alexander, Alison y Michael por acompañarme en este proceso. Finalmente, dedico este logro a mi querida mascota Felicia, cuya compañía incondicional y alegría diaria fueron un consuelo invaluable durante este camino.

Michael Alejandro:

Dedico este trabajo a todas las personas que me han acompañado y apoyado a lo largo de esta etapa tan importante de mi vida. A mis padres, quienes no solo me brindaron un apoyo emocional incondicional y me acompañaron con paciencia en cada paso de este viaje, sino que también me han servido de ejemplo constante de que uno debe luchar con determinación por las metas que quiere alcanzar. Les agradezco especialmente por haberme inculcado valores sólidos que han sido fundamentales en la formación de la persona que soy hoy. Sin su guía y apoyo constante, no estoy seguro de que hubiera podido lograr lo que he alcanzado hasta ahora.

A mi hermano, que ha sido una fuente adicional de apoyo emocional, ayudándome a encontrar momentos de distracción y calma en medio del estrés y las exigencias que han surgido a lo largo del camino. A mis abuelos, por sus palabras de aliento, por su compañía constante y por recordarme siempre la importancia de la familia y la unión.

A mis amigos Alexander, Alison y Alejandro, quienes han compartido conmigo todo este recorrido universitario. Juntos hemos vivido situaciones que van desde momentos de alegría y satisfacción, hasta etapas de presión y desesperación, en las que supimos apoyarnos mutuamente para seguir adelante. Creo firmemente que estas experiencias nos han ayudado a crecer no solo como profesionales, sino también como mejores personas.

Tabla de contenidos

Lista de tablas

Lista de figuras

DECLARACIÓN y AUTORIZACIÓN

Yo, **ALEJANDRO MUQUIS MICAHEL ANTHONY** con C.I. 1555238290 autor(a) del trabajo de TITULACIÓN intitulado: "**DESARROLLO DE UN SISTEMA AUTOMATIZADO PARA LA GENERACIÓN DE DOCUMENTACIÓN EN EL LABORATORIO ENVIRONOVALAB"**, previa a la obtención del título de **TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 11 DE MAYO DE 2025

MICHAEL ANTHONY ALEJANDRO MUQUIS

C.I. 1755238290

DECLARACIÓN y AUTORIZACIÓN

Yo, **BARRIONUEVO CAIZA ALEJANDRO PAUL** con C.I. 1754204582 autor(a) del trabajo de TITULACIÓN intitulado: **"DESARROLLO DE UN SISTEMA AUTOMATIZADO PARA LA GENERACIÓN DE DOCUMENTACIÓN EN EL LABORATORIO ENVIRONOVALAB"**, previa a la obtención del título de **TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE** en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, 11 DE MAYO DE 2025

ALEJANDRO PAUL BARRIONUEVO CAIZA

C.I. 1754204582

Agradecimientos

Michael Alejandro:

Este trabajo ha sido posible gracias al aporte, apoyo y compromiso de muchas personas que, de una u otra manera, contribuyeron a que este proyecto se haga realidad. Expreso mi más profundo agradecimiento a nuestro tutor, Patricio Alvear, por sus acertadas observaciones, su guía constante y su orientación brindada con profesionalismo y compromiso en cada etapa del proceso. Su experiencia y consejos nos dieron la seguridad de que avanzábamos por el camino correcto, lo que fue fundamental para mantener la confianza durante todo el desarrollo del trabajo.

Agradezco también a todos los docentes que, a lo largo de mi formación, me transmitieron no solo conocimientos técnicos, sino también valores y principios que hoy forman parte esencial de mi manera de trabajar y de enfrentar los retos profesionales. A mis amigos y compañeros de carrera, Alexander Pavón y Alison Carrion, gracias por compartir ideas, experiencias y desafíos que hicieron este recorrido más enriquecedor y llevadero. Mi reconocimiento se extiende a nuestra coordinadora de carrera, Diana Molina, por su valiosa orientación y dedicación durante el proceso de titulación. Su esfuerzo por asegurarse de que cumplamos con todos los requisitos, así como la información y apoyo brindados en cada momento, fueron de gran ayuda para culminar exitosamente esta etapa.

Finalmente, quiero dar un reconocimiento muy especial a mi compañero de tesis, Alejandro Barrionuevo, por su apoyo incondicional y su disposición para trabajar hombro a hombro en cada fase del proyecto. Juntos enfrentamos desafíos, largas

jornadas de investigación y momentos de incertidumbre, siempre con la convicción y el compromiso de sacar adelante nuestro trabajo. Su esfuerzo, compromiso y compañerismo fueron clave para que hoy podamos ver materializado este logro.

Alejandro Barrionuevo:

Agradezco a mi familia por sostenerme con su cariño, paciencia y confianza en cada etapa de mi proyecto de tesis. Su apoyo silencioso en los días difíciles y su entusiasmo en los logros marcaron la diferencia y me recordaron por qué valía la pena seguir adelante sin rendirse.

Agradezco a mi tutor, Patricio Alvear, por su guía constante en cada revisión y por sus observaciones precisas que encauzaron el proyecto; y a Diana Molina, quien, primero como profesora y coordinadora y luego únicamente como coordinadora, creyó en nosotros, nos animó en los momentos complejos y mantuvo siempre claridad y cercanía en los procesos. Extiendo mi gratitud al cuerpo docente por las herramientas técnicas y los valores compartidos, y a la institución por brindar los recursos y espacios que hicieron posible este trabajo.

A mi perrita Felicia, mi compañera de madrugadas y descansos cortos, gracias por esperar junto a mi escritorio, por celebrar conmigo cada avance y por enseñarme, sin palabras, que la constancia también se construye con ternura.

A mis amigos Alexander, Michael y Alison, gracias por la compañía, las ideas y el ánimo en las jornadas largas y también cuando decíamos que no lo íbamos a lograr, pero siempre salíamos de los problemas y lo solucionábamos y, de manera especial, a Michael Alejandro por su apoyo incondicional, por escuchar, cuestionar y celebrar cada

avance. Cada aporte, grande o pequeño, dejó huella en el resultado del proyecto y en mi crecimiento como programador.

Introducción

El presente proyecto tiene como finalidad el desarrollo de un sistema automatizado para la generación de documentación en el laboratorio ENVIRONOVALAB, institución especializada en análisis ambientales. Actualmente, procesos como la elaboración de proformas, el registro de datos y la creación de informes se realizan de forma manual, principalmente utilizando hojas de cálculo en Excel, lo cual ocasiona demoras operativas, duplicidad de esfuerzos y una alta susceptibilidad a errores humanos.

Con la implementación de una solución tecnológica basada en aplicaciones web, se busca optimizar dichos procesos mediante herramientas modernas que permitan agilizar la creación y gestión de documentos, reducir costos operativos y mejorar significativamente la precisión y la trazabilidad de la información. Esta transformación digital beneficiará tanto al personal interno del laboratorio como a los clientes que reciben los resultados, al ofrecerles un servicio más eficiente, organizado y confiable.

El proyecto contempla el uso de tecnologías actuales como React, JavaScript y Python, integradas con el framework Django para la construcción del backend, así como la utilización de bases de datos no SQL (MongoDB Atlas) para una adecuada organización de la información. La generación de documentos en formato PDF se automatizará mediante herramientas especializadas, garantizando que los reportes sean precisos y estén disponibles en formatos estandarizados.

Además de ofrecer una solución concreta a una problemática real, este trabajo de titulación permite a sus autores aplicar y consolidar conocimientos en áreas

fundamentales del desarrollo de software, como la programación, el diseño de interfaces, la gestión de datos y la implementación de metodologías ágiles, aportando así a su formación profesional y al fortalecimiento de competencias técnicas y analíticas.

1. Capítulo I

1.1. Levantamiento de Requisitos y Diseño del Sistema

1.1.1. Requerimientos Funcionales

1.1.1.1. Funcionalidad N°1: Registro de Usuarios

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F1	Registro	18/05/2025	18/05/2025	- En esta pantalla los nuevos usuarios podrán registrarse. Se validará la información de las credenciales.	Alta
	Entrada	Proceso		Salida	
	Formulario de registro	El usuario ingresa sus credenciales. Se valida formato y unicidad del correo.		El usuario es creado en la base de datos si los datos son válidos	
	Botón de “Registrarse”	Se envía la información al backend para su procesamiento		Muestra mensaje de éxito o errores en el formulario	

Tabla 1: Registro de Usuarios

1.1.1.2. Funcionalidad N°2: Login de Usuarios

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F2	Login	18/05/2025	18/05/2025	- En esta pantalla, el usuario podrá iniciar sesión con su correo y contraseña. Si las credenciales son válidas, accederá al sistema. Si no, se mostrará un mensaje de error.	Alta
	Entrada	Proceso		Salida	

	Formulario del Login	El usuario ingresa sus credenciales. Se valida que existan y sean correctos.	El usuario accede al sistema o recibe un mensaje de error
	Botón de "Iniciar Sesión"	Se envía la información al backend para autenticación	Acceso a la pantalla principal del sistema o mensaje de error

Tabla 2: Login de Usuarios

1.1.1.3. Funcionalidad N°3: Cerrar Sesión

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F3	Cerrar sesión	18/05/2025	18/05/2025	Permite al usuario cerrar la sesión de forma segura y volver al login..	Media
	Entrada	Proceso		Salida	
	Botón "Cerrar sesión"	Se limpia la sesión y se redirecciona		El usuario vuelve a la pantalla login	

Tabla 3: Cerrar Sesión

1.1.1.4. Funcionalidad N°4: Historial

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F4	Historial	18/05/2025	18/05/2025	En esta pantalla se mostrará un historial de acciones recientes realizadas por los usuarios. Se visualizarán columnas con: Nombre del documento, Fecha de acción, Tipo de acción (creación o modificación), y Nombre del usuario.	Media

	Entrada	Proceso	Salida
	Acceso a la opción “Historial””	Se consulta la base de datos por las acciones recientes (últimos 30 días, por ejemplo)	Se muestra una tabla con: Documento, Fecha, Acción, Usuario

Tabla 4: Historial

1.1.1.5. Funcionalidad N°5: Generación de Proformas

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F5	Generación de Proformas	18/05/2025	18/05/2025	En esta pantalla se podrán llenar formularios con datos como cliente, productos, cantidades, totales, y generar proformas. Habrá botones para guardar, previsualizar y exportar a PDF.	Alta
	Entrada	Proceso		Salida	
	Botón “Nueva Proforma”	Se abre un formulario para llenar con los datos del cliente y productos		Se muestra los campos para llenar la proforma	
	Botón “Guardar Proforma”	Se guarda la proforma en la base de datos		Proforma almacenada y visible en la lista	
	Botón “Previsualizar Proforma”	Se abre una previsualización de la proforma		Se muestra como es la proforma antes de descargarla.	
	Botón “Generar PDF”	Se exporta la proforma a un archivo PDF		El usuario descarga el PDF generado	

Tabla 5: Generar Proformas

1.1.1.6. Funcionalidad N°6: Generación de Informes

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F6	Generación de Informes	18/05/2025	18/05/2025	En esta pantalla se podrán llenar formularios con datos como datos de analisis, y generar informes. Habrá botones para guardar, previsualizar y exportar a PDF.	Alta
	Entrada	Proceso		Salida	
	Botón “Nuevo Informe”	Se abre un formulario para llenar con los datos del análisis		Se muestra los campos para llenar el informe	
	Botón “Guardar Informe””	Se guarda la proforma en la base de datos		Informe almacenada y visible en la lista	
	Botón “Previsualizar Informe””	Se abre una previsualización de la proforma		Se muestra el informe en una vista previa	
	Botón “Generar PDF”	Se exporta la proforma a un archivo PDF		El usuario descarga el PDF generado	

Tabla 6: Generar Informes

1.1.1.7. Funcionalidad N°7: Administración de Archivos

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
	Administración de Archivos	18/05/2025	18/05/2025	En esta pantalla se mostrará una tabla con todos los formularios o documentos	Alta

F7				existentes en el sistema. Los usuarios podrán **visualizar**, **editar** o **eliminar** documentos desde esta vista. No se podrá crear nuevos documentos desde aquí.	
	Entrada	Proceso		Salida	
	Botón de “Ver documento”	Se abre una vista previa o modal para visualizar el contenido del documento		Se muestra el contenido completo del documento	
	Botón de “Editar documento”	Se abre el formulario con los campos editables del documento		Se actualiza la información del documento	
	Botón de “Eliminar documento”	Se muestra una alerta de confirmación para eliminar el documento		El documento se elimina de la lista y base de datos	

Tabla 7: Administración de Archivos

1.1.1.8. Funcionalidad N°8: Gestión de Tipos de Muestra

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F8	Gestión de Tipos de Muestra	18/05/2025	18/05/2025	Permite al administrador crear, editar o eliminar tipos de muestra predefinidos. Estos se utilizan para llenar automáticamente análisis en las proformas.	Alta
	Entrada	Proceso		Salida	
	Formulario de tipo de muestra	El administrador ingresa tipo, parámetro, método, unidad, técnica y precio.		El tipo de muestra se guarda en la base de datos.	

	Botones de “Modificar” y “Eliminar”	Se actualiza o elimina un registro existente.	Mensaje de éxito y actualización visual.

Tabla 8: Gestión de Tipos de Muestra

1.1.1.9. Funcionalidad N°9: Recuperación de Contraseña

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F9	Recuperación de Contraseña	18/05/2025	18/05/2025	Permite a los usuarios restablecer su contraseña mediante un enlace enviado a su correo electrónico. Incluye validaciones de formato y seguridad.	Media
	Entrada	Proceso		Salida	
	Correo electrónico	Se valida el correo y se genera un token de recuperación.		El sistema envía un correo con el enlace.	
	Nueva contraseña	El usuario accede al enlace y registra una nueva contraseña.		El sistema confirma el cambio de contraseña.	

Tabla 9: Recuperación de Contraseña

1.1.1.10. Funcionalidad N°10: Cambio de Estado de Proformas e Informes

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F10	Cambio de Estado	18/05/2025	18/05/2025	El administrador puede actualizar el estado de proformas e informes (por hacer, en progreso, terminado).	Baja
	Entrada	Proceso		Salida	
	Selector de estado	El administrador selecciona un nuevo estado.		El estado se actualiza en la base de datos.	

	Evento onchange	Se envía la actualización al backend.	Se refleja visualmente el nuevo estado.

Tabla 10: Descarga de Documentos en PDF

1.1.1.11. Funcionalidad N°11: Descarga de Documentos en PDF

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F11	Descarga de Documentos PDF	18/05/2025	18/05/2025	Permite descargar las proformas e informes generados en formato PDF directamente desde la interfaz.	Alta
	Entrada	Proceso		Salida	
	Botón “Descargar PDF”	El administrador selecciona un nuevo estado.		El estado se actualiza en la base de datos.	
	Identificador del documento	Se verifica que el documento exista.		Si existe, se muestra una descarga exitosa.	

Tabla 11: Descarga de Documentos en PDF

1.1.1.12. Funcionalidad N°12: Filtros y Búsquedas

N°	Nombre de Actividad	Fecha Inicio	Fecha Fin	Descripción	Prioridad
F12	Filtro y Búsqueda	18/05/2025	18/05/2025	Permite filtrar y buscar proformas e informes por código, cliente o fecha, en todas las vistas del sistema.	Alta
	Entrada	Proceso		Salida	
	Campo de búsqueda	Se filtran dinámicamente los datos del backend.		Se actualiza la tabla mostrando coincidencias.	
	Texto ingresado	Se consulta el servidor según el término.		Se muestran resultados o mensaje “no encontrado”.	

Tabla 12: Filtros y Búsquedas

1.1.2. Mockups:

Para el diseño del sistema están creados con la herramienta figma.

1.1.2.1. Pantalla N°1: Login

La pantalla de Login permite a los usuarios acceder al sistema ingresando su nombre de usuario y contraseña. Incluye un botón para iniciar sesión y un enlace para recuperar la contraseña en caso de olvido. Esta pantalla garantiza el acceso seguro al sistema.

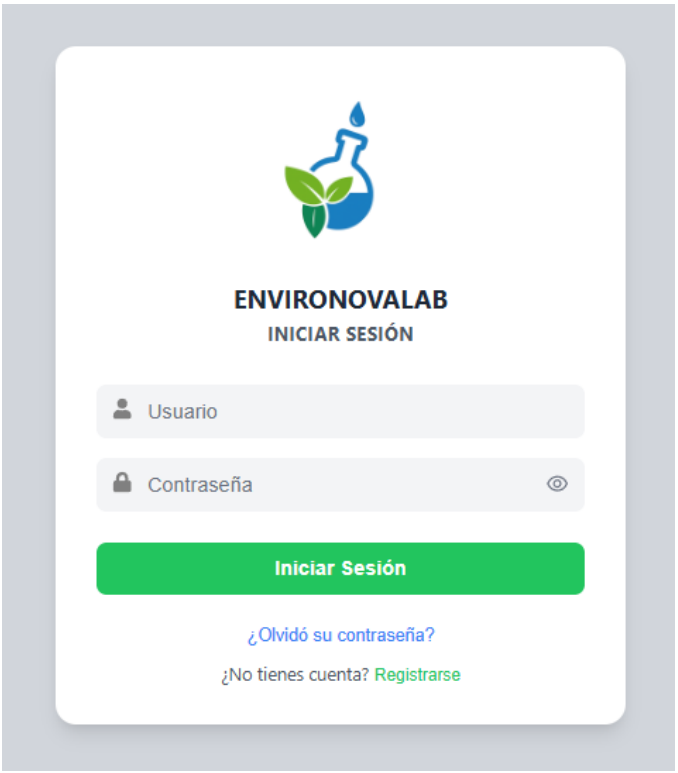

Ilustración 1: Pantalla de Login

1.1.2.2. Pantalla N°2: Registro

La pantalla de Registro permite a nuevos usuarios crear una cuenta ingresando su nombre de usuario, correo electrónico y contraseña. Incluye un botón para completar el registro y un enlace para redirigirse al inicio de sesión si ya tiene una cuenta. Esta pantalla facilita el acceso inicial al sistema de forma clara y segura.

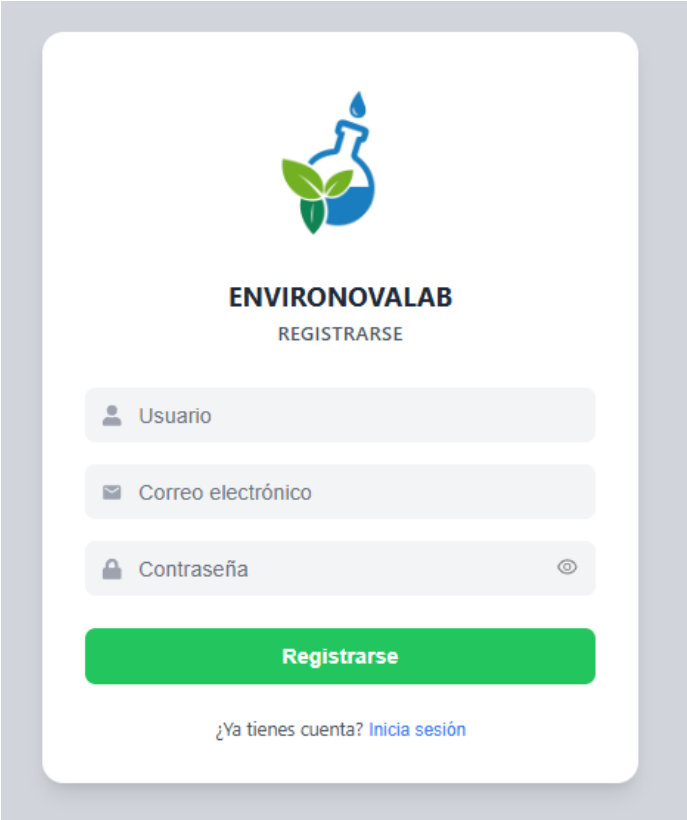

Ilustración 2: Pantalla de Registro

1.1.2.3. Pantalla N°3: Recuperar Contraseña

La pantalla de recuperación permite a los usuarios restablecer su contraseña ingresando su correo electrónico. Incluye un botón para enviar la solicitud y un enlace para volver al login. Esta vista facilita el proceso de recuperación de acceso al sistema.

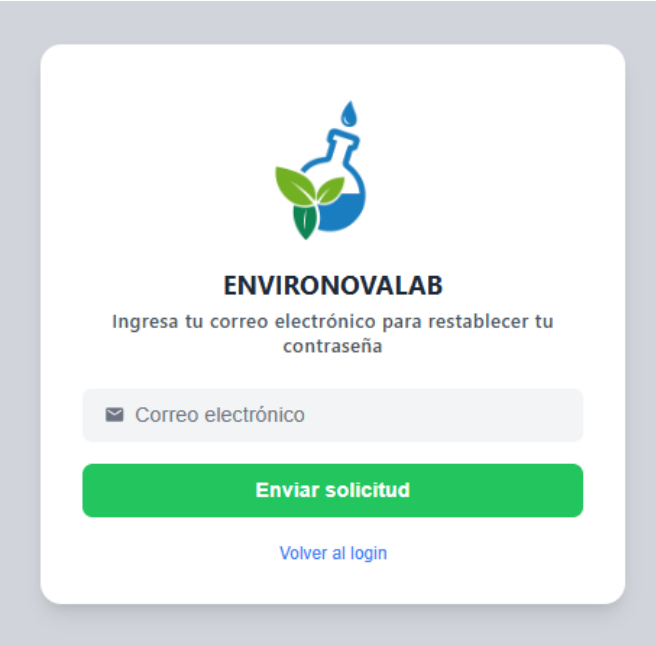

Ilustración 3: Pantalla de Recuperar Contraseña

1.1.2.4. Pantalla N°4: Restablecer Contraseña

La pantalla de restablecimiento permite al usuario ingresar una nueva contraseña para recuperar el acceso al sistema. Incluye un campo de entrada y un botón para confirmar el cambio de forma segura.

Ilustración 4: Pantalla de Restablecer Contraseña

1.1.2.5. Pantalla N°5: Inicio

La pantalla de inicio muestra un saludo personalizado y una tabla con los documentos recientes, incluyendo proformas e informes. Permite filtrar por código y acceder rápidamente a las acciones disponibles para cada documento.

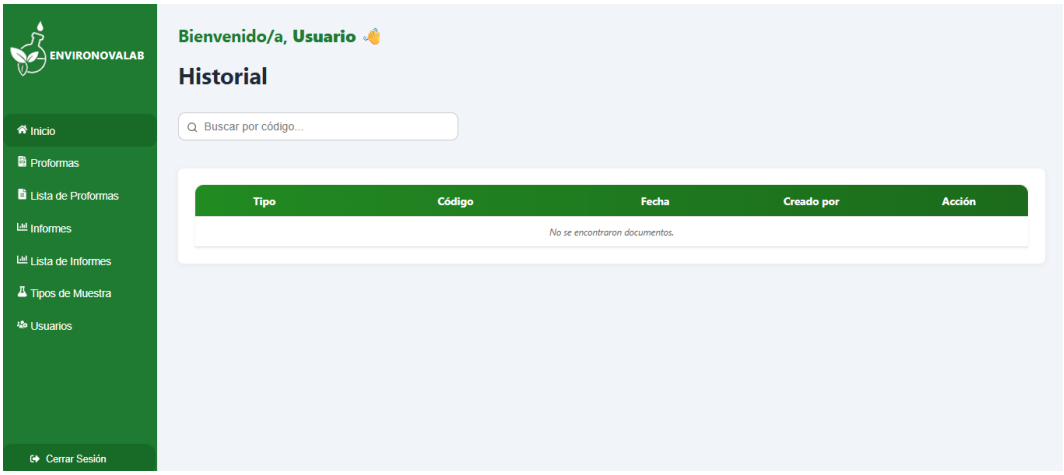

Ilustración 5: Pantalla de Inicio

1.1.2.6. Pantalla N°6: Generar Proforma

La pantalla permite registrar los datos del cliente y agregar ítems de monitoreo para generar una proforma. Incluye botones para limpiar el formulario, cancelar, guardar o agregar nuevos análisis.

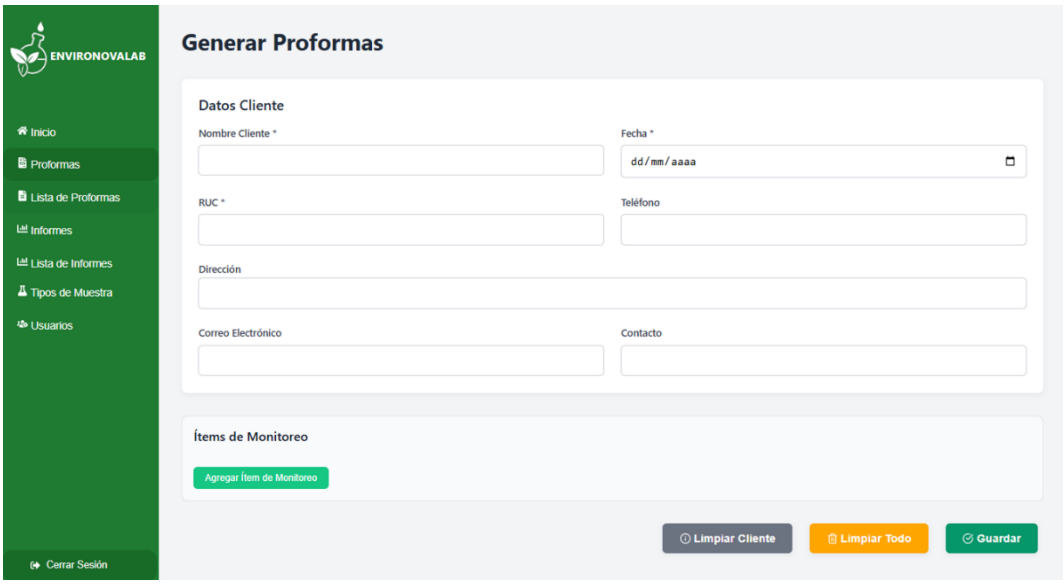

Ilustración 6: Pantalla de Generar Proforma

1.1.2.7. Pantalla N°7: Lista de Proformas

Esta pantalla muestra una tabla con todas las proformas registradas, incluyendo tipo, código, fecha, estado, autor y acciones disponibles. Permite buscar documentos por su código mediante un campo de búsqueda.

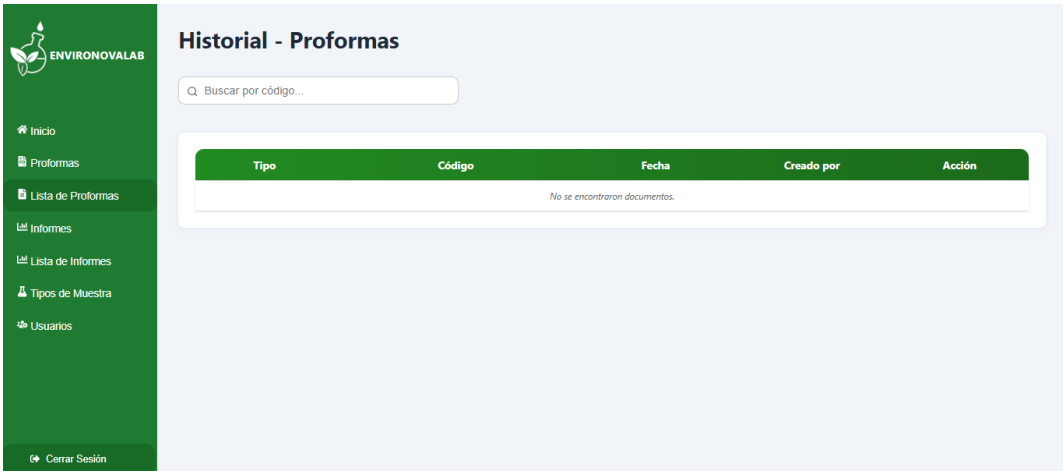

Ilustración 7: Pantalla de Lista de Proformas

1.1.2.8. Pantalla N°8: Generar Informe

En esta pantalla es donde se refleja la pestaña informe que lo primero que hacemos es poner el identificador de la proforma buscamos y salen los datos de la proforma y luego se llena el informe de la muestra que sea.

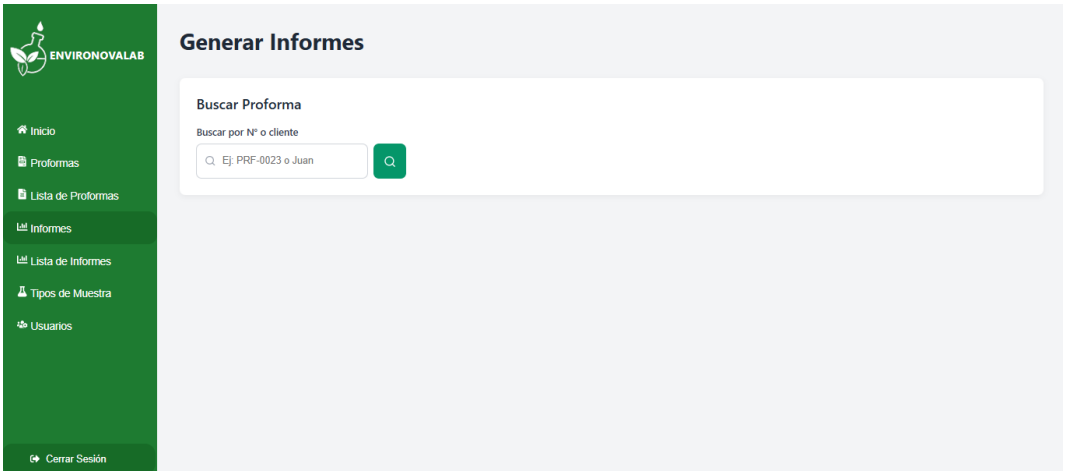

Ilustración 8: Pantalla de Generar Informe

1.1.2.9. Pantalla N°9: Lista de Informes

La pantalla muestra un listado de informes generados, organizados en una tabla con columnas como tipo, código, fecha, estado, autor y acciones. Incluye un buscador para filtrar por código de forma rápida.

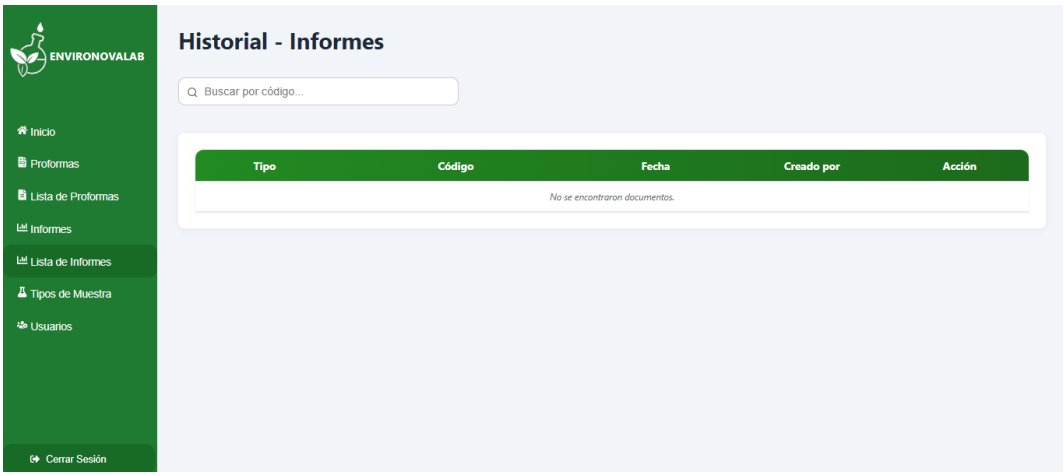

Ilustración 9: Pantalla de Lista de Informes

1.1.2.10. Pantalla N°10: Tipos de Muestra

Esta pantalla permite registrar nuevos tipos de muestra ingresando información como tipo, parámetro, unidad, método, técnica y precio. Incluye un botón para agregar la muestra al sistema de manera rápida.

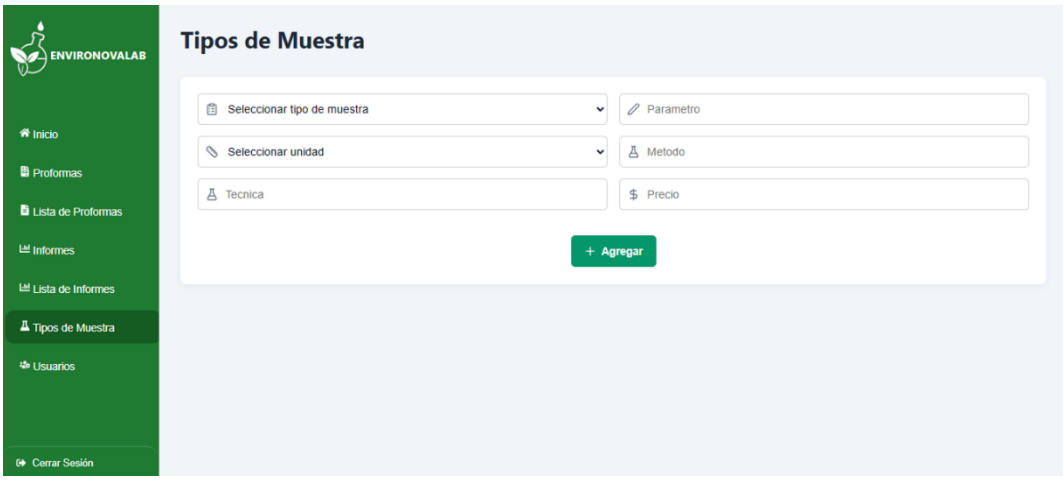

Ilustración 10: Pantalla de Tipos de Muestra

1.1.2.11. Pantalla N°11: Administrar Usuarios

La pantalla permite a los administradores gestionar usuarios del sistema. Muestra una tabla con información de nombre, correo, rol y estado, además de un botón para añadir nuevos usuarios.

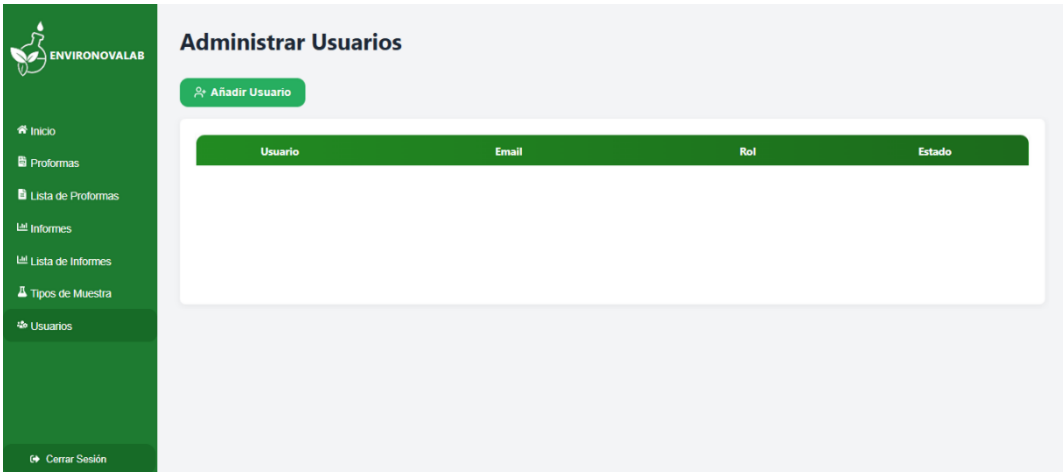

Ilustración 11: Pantalla de Administrar Usuarios

1.1.3. Casos de Usos

1.1.3.1. Caso de Uso 1: Gestión de usuarios

Este caso de uso se refiere a todas las acciones relacionadas con el manejo de los usuarios del sistema. Su propósito es asegurar que solo los usuarios autorizados puedan interactuar con el sistema.

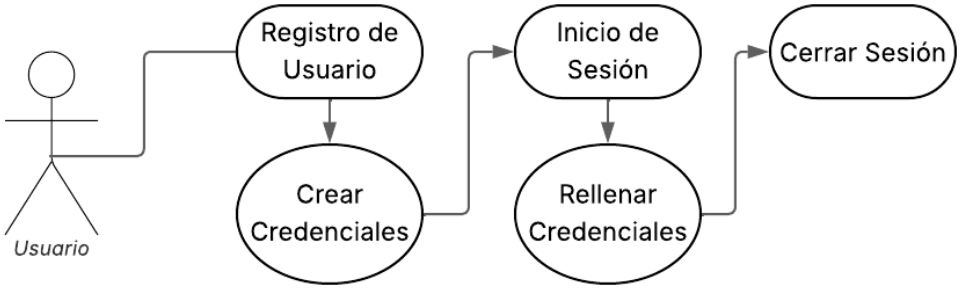

Ilustración 12: Gestión de Usuarios

1.1.3.2. Caso de Uso 2: Creación de Documentos

Este caso de uso cubre el proceso mediante el cual se generan los documentos dentro de la plataforma, tales como proformas e informes.

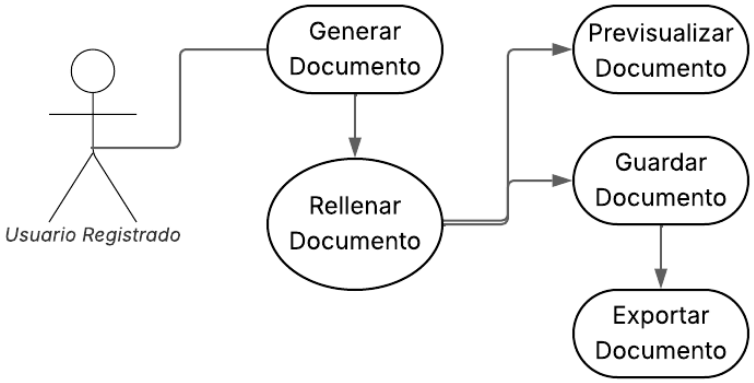

Ilustración 13: Creación de Documentos

1.1.3.3. Caso de Uso 3: Gestión de Documentos

Este caso de uso abarca las funciones de administración de archivos dentro del sistema, incluye el almacenamiento, consulta, modificación y descarga.

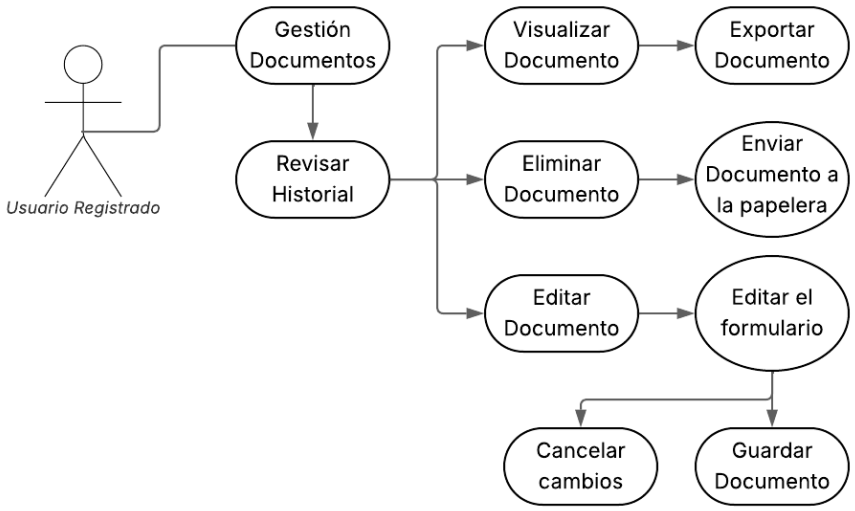

Ilustración 14: Gestión de Documentos

1.1.3.4. Caso de Uso 4: Gestión de Tipos de Muestra

Este caso de uso permite a los administradores del sistema gestionar el catálogo de tipos de muestra. Incluye funcionalidades para crear, modificar y eliminar registros, los cuales se utilizarán posteriormente al generar proformas.

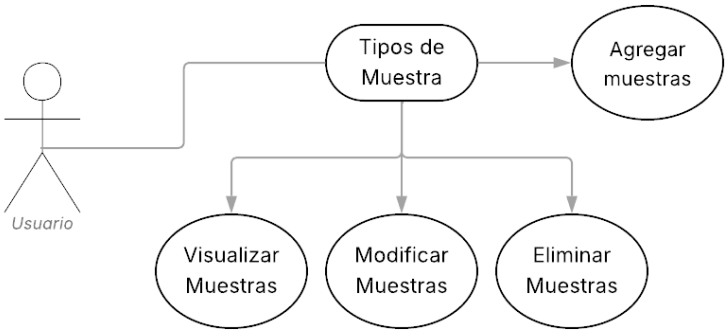

Ilustración 15: Gestión de Tipos de Muestra

1.1.3.5. Caso de Uso 5: Recuperación de Contraseña

Este caso de uso describe el proceso que permite a los usuarios recuperar el acceso a su cuenta en caso de haber olvidado su contraseña.

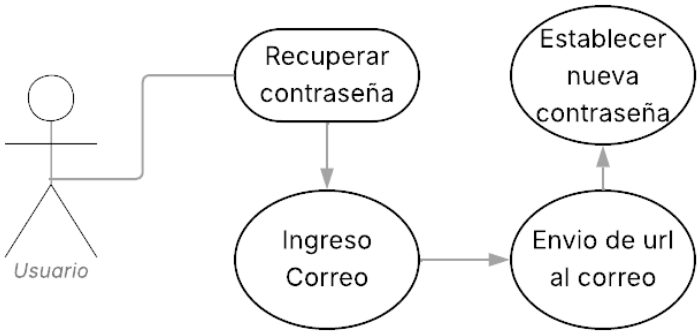

Ilustración 16: Recuperación de Contraseña

1.1.3.6. Caso de Uso 6: Cambio de Estado de Documentos

Este caso de uso describe cómo el administrador puede modificar el estado de proformas o informes técnicos (por hacer, en progreso, terminado).

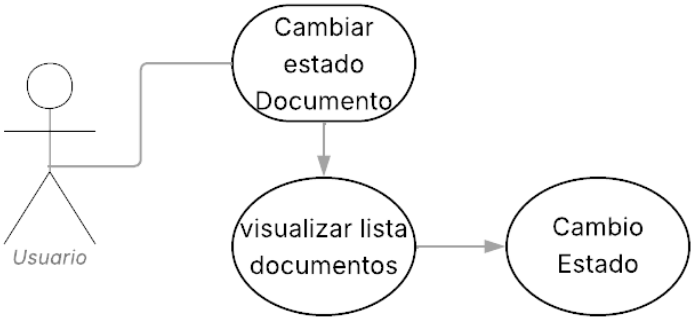

Ilustración 17: Cambio de Estado de Documentos

1.1.3.7. Caso de Uso 7: Filtro y Búsqueda en Listados

Este caso de uso permite a los usuarios filtrar documentos (proformas e informes) por nombre del cliente o código, agilizando su localización.

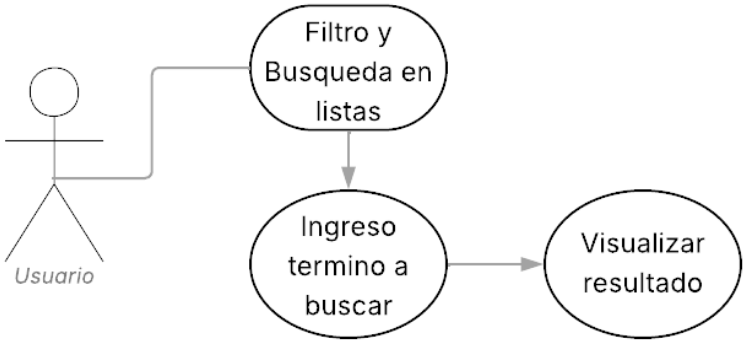

Ilustración 18: Filtro y Búsqueda en Listados

1.1.4. Modelo Entidad-Relación

El modelo entidad-relación representa la estructura lógica de los datos del sistema, permitiendo identificar las entidades clave, sus atributos principales y las relaciones que existen entre ellas. Entre las entidades principales se incluyen los usuarios del sistema, los clientes que solicitan análisis, las proformas generadas como presupuestos previos, los análisis técnicos, los informes resultantes, los datos obtenidos y la configuración de la empresa. Además, se incorpora un catálogo de tipos de muestra para estandarizar y agilizar el registro de análisis.

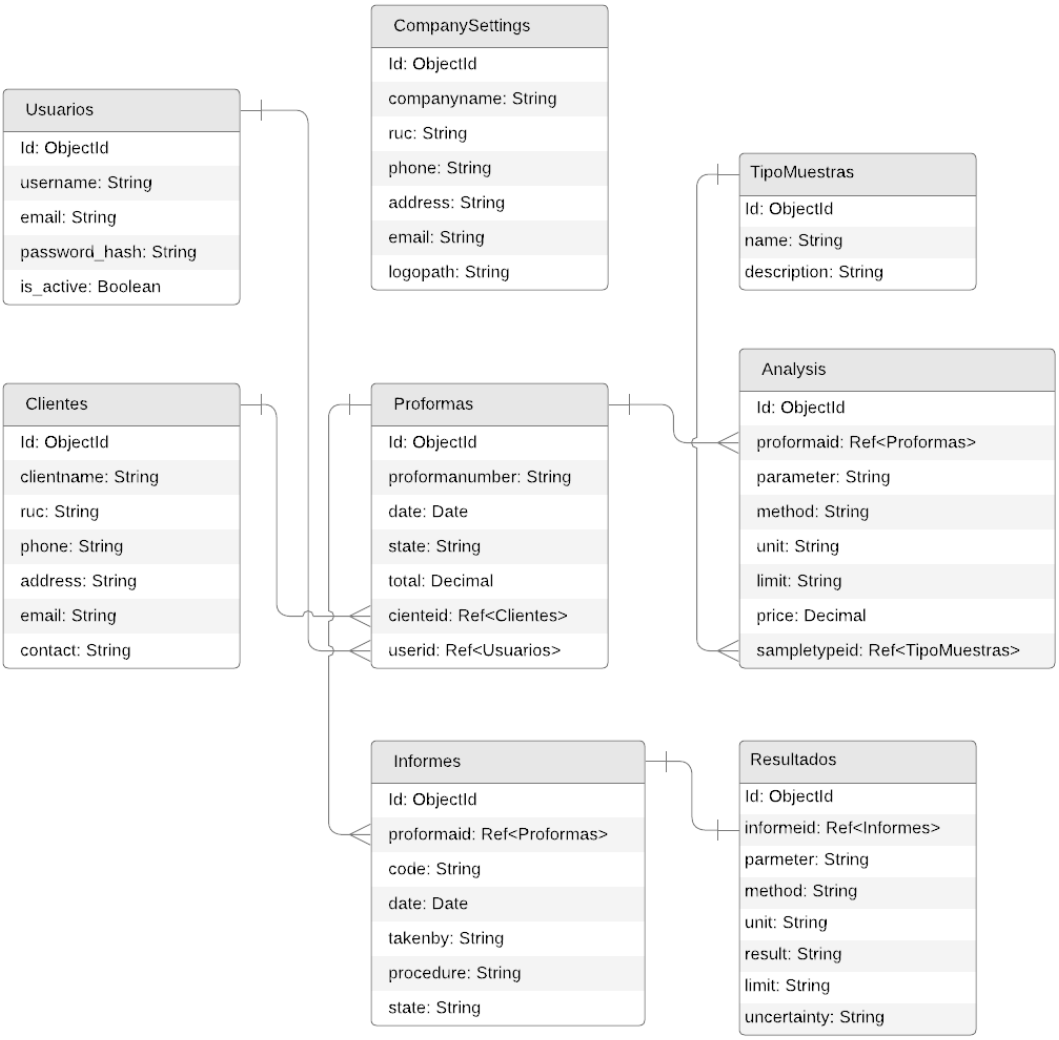

Ilustración 19: Modelo Entidad-Relación

1.1.5. Diagrama de Clases UML

El diagrama de clases UML define la estructura del sistema para gestionar usuarios, clientes, proformas, análisis, informes y resultados. Muestra cómo las clases interactúan para permitir la creación y modificación de proformas, el registro y cálculo de análisis, la generación de informes con sus resultados, y la administración de la configuración institucional, todo vinculado a usuarios autenticados.

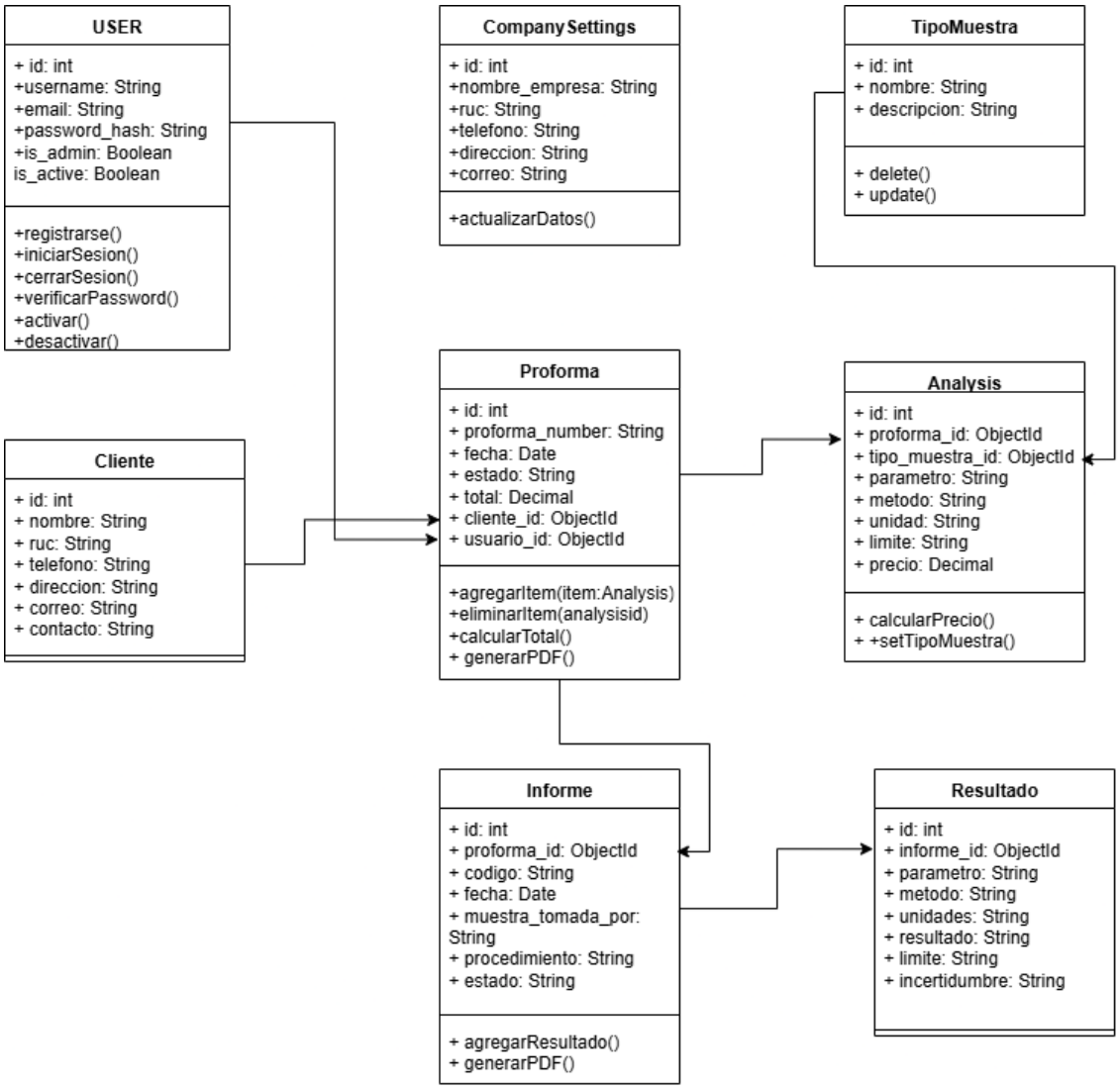

Ilustración 20: Diagrama de Clases

2. Capítulo II

2.1. Construcción del Sistema Frontend

Durante la construcción del sistema frontend se priorizó el uso de tecnologías modernas, una arquitectura modular y una experiencia de usuario clara e intuitiva. El desarrollo fue guiado por principios de organización, reutilización de componentes y coherencia visual, permitiendo una integración efectiva con el backend y una navegación fluida para distintos tipos de usuarios.

2.1.1. Estándar de construcción

Se definieron criterios técnicos y visuales para asegurar la calidad del código, su mantenibilidad y la consistencia entre vistas. Estos estándares abarcan desde la estructura de carpetas hasta el uso de librerías, componentes y validaciones visuales, estableciendo una base sólida para el desarrollo y evolución del sistema.

2.1.1.1. Estructura por carpetas

El frontend está organizado en carpetas components, pages, y styles. Esta separación facilita el mantenimiento del código, permitiendo distinguir entre vistas completas, componentes reutilizables y archivos de estilo CSS.

2.1.1.2. Uso de React funcional

Todos los componentes están desarrollados como funciones utilizando hooks como useState, useEffect y useCallback, lo que garantiza una lógica clara y reactiva en el manejo de estados y efectos secundarios.

2.1.1.3. Formularios controlados

Cada input está vinculado al estado interno del componente. Esto permite validar fácilmente los datos antes de enviarlos al backend, mostrar errores en tiempo real y mejorar la experiencia del usuario.

2.1.1.4. Manejo de peticiones con Axios

Las operaciones con la API se gestionan usando axios, incluyendo el envío de datos, la lectura de información y el manejo de errores. Además, se incluye el token CSRF en cada solicitud para mantener la seguridad.

2.1.1.5. Validaciones y mensajes visuales

Todos los formularios validan que los campos obligatorios estén completos. Los errores o confirmaciones se muestran al usuario mediante mensajes visuales estilizados tipo "toast", acompañados de íconos.

2.1.1.6. Estilo visual consistente

Cada página y componente mantiene una estética uniforme: colores suaves, íconos embebidos dentro de los inputs, botones con bordes redondeados y distribución clara del contenido en tarjetas.

2.1.1.7. Protección de rutas

Las rutas privadas están protegidas por el componente ProtectedRoute, que evita el acceso a páginas restringidas si el usuario no está autenticado o no tiene el rol adecuado. Esto garantiza que solo los usuarios autorizados puedan acceder a ciertas secciones.

2.1.2. Definición de Paginas

El sistema se compone de múltiples páginas diseñadas para cubrir tanto funciones operativas como administrativas. Cada una fue planificada con un propósito específico dentro del flujo de uso, permitiendo al usuario navegar entre módulos de forma lógica y eficiente.

2.1.2.1. Login

Es la página inicial del sistema, donde los usuarios ingresan su nombre de usuario y contraseña para acceder. Incluye validación de campos, opción para mostrar u ocultar la contraseña, y enlaces directos al formulario de registro o recuperación de contraseña en caso de ser necesario.

2.1.2.2. Register

Permite a nuevos usuarios crear una cuenta completando un formulario con su nombre de usuario, correo electrónico y contraseña. La vista proporciona retroalimentación inmediata sobre el éxito o fallo del registro, y redirige automáticamente al login si el registro es exitoso.

2.1.2.3. ForgotPassword

Esta vista permite al usuario iniciar el proceso de recuperación de contraseña, ingresando su correo electrónico. Si el correo es válido, se envía un enlace de restablecimiento que conduce a la siguiente etapa del proceso.

2.1.2.4. ResetPassword

Utilizada luego de recibir el enlace por correo, esta página permite al usuario establecer una nueva contraseña de forma segura. Muestra mensajes claros sobre el estado de la solicitud y redirige al login una vez completado el proceso.

2.1.2.5. Dashboard

Funciona como la página principal del sistema una vez que el usuario ha iniciado sesión. Muestra un saludo personalizado y una tabla con la actividad reciente.

2.1.2.6. ProformaGenerator

Es la vista encargada de generar proformas. Permite ingresar los datos del cliente y seleccionar distintos tipos de análisis desde un catálogo precargado. Al finalizar, se guarda la proforma y se ofrece la descarga del documento en PDF.

2.1.2.7. InformeGenerator

Esta página permite generar un informe técnico a partir de una proforma existente. El usuario puede buscar la proforma por número o nombre del cliente, completar los resultados del análisis y guardar el informe. Una vez guardado, se habilita la descarga del informe en PDF.

2.1.2.8. AdminMuestras

Es una vista administrativa donde se gestionan los tipos de muestra utilizados en el sistema. Permite crear, editar o eliminar registros mediante formularios estilizados y tarjetas informativas, asegurando una administración eficiente del catálogo.

2.1.2.9. UserAdmin

Vista exclusiva para usuarios con rol de administrador. Permite gestionar todos los usuarios registrados en el sistema, crear nuevos, cambiar su rol entre usuario o admin, y activar o desactivar sus cuentas. Toda la gestión se realiza desde una interfaz clara, con notificaciones visuales y formularios emergentes.

2.1.2.10. ProformasList

Es una vista exclusiva para el rol de administrador. Esta lista nos permite ver la cantidad de proformas generadas donde se puede ver la fecha, quien creo la proforma el código y el tipo de documento también nos permite descargar un pdf de ese documento con un botón.

2.1.2.11. InformesList

Es una vista para el rol de administrador y usuario normal. Esta lista nos permite ver la cantidad de informes generadas donde se puede ver la fecha, quien creo el informe, el código y el tipo de documento también nos permite descargar un pdf de ese documento con un botón.

2.1.3. Codificación de Paginas

Las páginas del sistema fueron desarrolladas como componentes funcionales de React, empleando hooks, formularios controlados y estructuras limpias. Cada vista implementa su propia lógica de estado, comunicación con el backend y renderizada visual, manteniendo la coherencia y la separación de responsabilidades.

2.1.3.1. Login.jsx

La página utiliza useState para manejar los campos de usuario y contraseña, y axios para enviar la solicitud de inicio de sesión al backend. Al recibir una respuesta válida, se almacenan datos en localStorage, y se redirige al dashboard. Incluye validaciones básicas, visibilidad de contraseña y mensajes de error mostrados en pantalla.

2.1.3.2. Register.jsx

Esta vista está implementada como un formulario controlado, donde cada cambio de input actualiza el estado del formulario. Al enviarse, se realiza una petición POST al backend con los datos ingresados. Se muestra un mensaje visual con el resultado y se redirige automáticamente al login tras unos segundos si el registro fue exitoso.

2.1.3.3. ForgotPassword.jsx

Utiliza un formulario sencillo que toma el correo electrónico del usuario y lo envía al backend. La respuesta del servidor determina el mensaje que se muestra al usuario, y se gestiona el estado de carga para evitar múltiples envíos. Se incluye validación y navegación hacia el login.

2.1.3.4. ResetPassword.jsx

Obtiene el token desde la URL utilizando useParams. El formulario toma una nueva contraseña y la envía mediante axios al endpoint correspondiente. Al completar con éxito la operación, se muestra un mensaje y se redirige al usuario automáticamente al login.

2.1.3.5. Dashboard.jsx

Esta vista muestra un saludo con el nombre del usuario recuperado desde localStorage y una tabla de actividad reciente. Es una página visualmente sencilla pero funcional, que actúa como punto de inicio para navegar al resto del sistema.

2.1.3.6. ProformaGenerator.jsx

Página compleja estructurada con múltiples secciones: formulario del cliente, selección dinámica de tipos de análisis, validaciones, y notificaciones visuales. Cada análisis se selecciona desde un select, y al guardar, se realiza una cadena de peticiones: primero se guarda el cliente, luego la proforma. Al finalizar, se muestra un botón para descargar el PDF generado.

2.1.3.7. InformeGenerator.jsx

Al ingresar, se carga el listado de proformas y se permite buscarlas por nombre o número. Una vez seleccionada una, se muestran los datos del cliente y una tabla editable para ingresar resultados, límites e incertidumbres. El botón de guardar realiza una validación completa y guarda el informe mediante una petición POST. Si se guarda correctamente, se habilita un botón para descargar el informe en PDF.

2.1.3.8. InformesList.jsx

Este componente carga automáticamente las proformas creadas desde el backend y permite buscarlos por código o nombre del creador mediante una barra interactiva. Muestra una tabla con datos clave como tipo, código, fecha, autor y acciones disponibles. Los usuarios pueden descargar el informe en PDF con un clic, y si no hay

resultados, se muestra un mensaje indicándolo. Todo el contenido se presenta con un diseño claro y responsivo.

2.1.3.9. InformesList.jsx

Este componente muestra los informes más recientes al ingresar, obteniéndolos directamente desde el backend. Permite buscar informes por su código o nombre del creador mediante una barra interactiva. La tabla presenta datos clave como tipo, código, fecha, autor y una opción para descargar el informe en PDF. Asimismo, el diseño es limpio, responsivo y muestra un mensaje visual si no hay resultados coincidentes.

2.1.3.10. AdminMuestras.jsx

Esta vista carga los tipos de muestra existentes y muestra los datos en formato de tarjetas. Tiene un formulario para agregar nuevos registros y un modal emergente que se activa al presionar “Modificar”. La gestión de peticiones usa axios con encabezados CSRF. Toda la vista está acompañada de notificaciones tipo "toast" y validaciones visuales.

2.1.3.11. UserAdmin.jsx

La vista se encarga de cargar todos los usuarios del sistema y permite modificarlos directamente desde una tabla. Los botones permiten alternar el rol y el estado de cada usuario, y se desactivan si el usuario actual intenta modificarse a sí mismo. La creación de usuarios se realiza mediante un modal que incluye campos con íconos embebidos. Toda la interacción se realiza mediante peticiones POST, con manejo de CSRF y actualizaciones automáticas tras cada acción.

2.1.4. Componentes Adicionales

Para reforzar la reutilización, la seguridad y la experiencia de usuario, se desarrollaron componentes auxiliares que apoyan la navegación, el control de acceso, la visualización de notificaciones y la edición de datos. Estos elementos fortalecen la estructura del sistema y complementan las páginas principales.

2.1.4.1. Sidebar.jsx

Este componente representa la barra lateral del sistema, visible en todas las vistas protegidas. Se adapta dinámicamente al rol del usuario, mostrando u ocultando enlaces según corresponda. Incluye íconos, rutas personalizadas y un botón de cierre de sesión que limpia el localStorage y redirige al login.

2.1.4.2. ProtectedRoute.jsx

Se encarga de proteger las rutas del sistema. Verifica si el usuario está autenticado, permitiendo el acceso solo en caso válido. En caso contrario, redirige automáticamente al login o al dashboard, garantizando el control de acceso por rol y sesión.

2.1.4.3. Toasts personalizados

Tanto en ProformaGenerator, InformeGenerator, AdminMuestras como en UserAdmin, se implementa un sistema de notificaciones tipo "toast". Estos toasts se apilan en la parte superior y muestran mensajes de éxito, error o información, acompañados de íconos.

2.1.4.4. Modales (UserAdmin y AdminMuestras)

Se utilizan formularios emergentes para la edición o creación de datos. Estos modales aparecen centrados en pantalla, con fondo semitransparente bloqueante, inputs con íconos, validaciones y botones estilizados para confirmar o cancelar la acción.

2.1.4.5. Formulario con íconos embebidos

En los formularios del sistema, los campos de entrada incluyen íconos, alineados a la izquierda del texto. Esta decisión mejora la experiencia de usuario, facilita la interpretación de cada campo y mantiene una estética visual uniforme.

2.1.4.6. App.js (enrutamiento central)

Aunque no es un componente visual, App.js se encarga de definir todas las rutas del sistema, agrupándolas entre públicas, protegidas y exclusivas para administradores. Utiliza react-router-dom junto con ProtectedRoute para controlar el flujo de navegación del sistema.

2.2. Construcción del Sistema Backend

Durante la construcción del backend se empleó una arquitectura modular basada en el framework Django REST Framework, usando MongoEngine como ORM para trabajar con MongoDB Atlas. El enfoque principal fue garantizar una separación clara de responsabilidades, permitiendo un desarrollo escalable, seguro y mantenible. Cada modelo, vista y serializer fue diseñado para reflejar de forma fiel la lógica del dominio, y se implementaron mecanismos para validación, autenticación, generación de documentos PDF y administración de datos.

2.2.1. Estándar de construcción

Se definieron lineamientos técnicos para asegurar la legibilidad, mantenibilidad y correcta separación entre modelos, serializadores, vistas, utilitarios y rutas:

2.2.1.1. Estructura modular por carpetas

Los archivos están organizados en carpetas como models, serializers, views, utils y urls, con subdivisiones por entidad.

2.2.1.2. Base de Datos NoSQL

Se utilizó mongoengine para definir y gestionar documentos en MongoDB, aprovechando la flexibilidad de los esquemas dinámicos.

2.2.1.3. Controladores con DRF

Se implementaron vistas tipo ViewSet y clases API para manejar operaciones CRUD, acciones personalizadas y generación de PDF.

2.2.1.4. Autenticación y permisos

El sistema incluye autenticación de usuarios con gestión de roles (admin y usuario) y protección por permisos en las rutas.

2.2.1.5. Generación automática de PDFs

Se implementaron funciones reutilizables para generar documentos formateados desde datos del sistema, utilizando ReportLab.

2.2.1.6. Manejo de errores

Se implementó un sistema de manejo de errores con mensajes claros para el usuario, incluyendo validaciones de unicidad, existencia de relaciones y formatos requeridos.

2.2.2. Definición de Modelos

El modelo de datos del sistema está diseñado para representar entidades clave como usuarios, clientes, proformas, informes y análisis de muestras. A continuación, se detallan cada uno de los modelos implementados, indicando para qué sirve cada uno y explicando el propósito de sus campos más relevantes.

2.2.2.1. Modelo User

El modelo User permite gestionar los usuarios del sistema, identificando su rol y estado. Es fundamental para el control de acceso y autenticación.

- **username:** Nombre de usuario. Sirve para iniciar sesión en el login. Es único y obligatorio.
- **password:** Contraseña cifrada del usuario. Es obligatoria para ingresar al sistema.
- **email:** Correo electrónico del usuario. Es único y obligatorio. Se utiliza para recuperación de contraseña.
- **is_admin:** Valor booleano que indica si el usuario tiene permisos de administrador o no.
- **is_active:** Indica si la cuenta está activa o desactivada, lo que permite suspender accesos sin eliminar al usuario.

2.2.2.2. Modelo Client

El modelo Client representa a los clientes del laboratorio, es decir, las personas o empresas que solicitan análisis y reciben proformas o informes.

- **name:** Nombre del cliente. Campo obligatorio que permite identificar al solicitante.
- **ruc:** Número de RUC o identificación tributaria.
- **phone:** Teléfono de contacto del cliente.
- **address:** Dirección física del cliente.
- **email:** Correo electrónico de contacto.
- **contact_person:** Nombre de la persona de contacto dentro de la empresa cliente.
- **created_at / updated_at:** Fechas que indican cuándo se creó o actualizó el registro.

2.2.2.3. Modelo CompanySettings

El modelo CompanySettings define la configuración general de la empresa, como el nombre, dirección y parámetros para la generación de proformas.

- **company_name:** Nombre de la empresa (ej. Environovalab).
- **company_address / company_phone / company_email:** Datos de contacto institucional.
- **company_ruc:** RUC tributario de la empresa.
- **company_logo:** Ruta del archivo con el logo, usado en la generación de PDFs.
- **proforma_prefix:** Prefijo que se antepone al número de cada proforma (ej. PRF-0001).

- **next_proforma_number:** Número secuencial que se incrementa automáticamente al crear nuevas proformas.
- **tax_rate:** Porcentaje de impuesto aplicado a las proformas (ej. 0.12 = 12%).

2.2.2.4. Modelo TipoMuestra

El modelo TipoMuestra representa un catálogo de tipos de análisis predefinidos disponibles en el laboratorio. Sirve para autocompletar la información de análisis al generar una proforma.

- **tipo:** Categoría de la muestra (agua, ruido, etc.).
- **parametro:** Parámetro a analizar (pH, turbidez, etc.).
- **unidad:** Unidad de medida del parámetro.
- **metodo:** Método de análisis usado.
- **tecnica:** Técnica específica empleada en el análisis.
- **precio:** Precio unitario del análisis.

2.2.2.5. Modelo Proforma

El modelo Proforma almacena los presupuestos emitidos a los clientes antes de realizar los análisis. Incluye información financiera y referencia al cliente.

- **client:** Relación con el cliente al que pertenece la proforma.
- **proforma_number:** Código único que identifica a la proforma.
- **date:** Fecha de emisión de la proforma.
- **status:** Estado actual (borrador, enviado, aprobado, rechazado).
- **subtotal / tax_amount / total:** Cálculos automáticos del valor neto, impuestos y total.

- **created_by:** Nombre de quien generó la proforma.
- **pdf_url:** Ruta del PDF generado.
- **created_at / updated_at:** Fechas de creación y última modificación.

2.2.2.6. Modelo Analysis

El modelo Analysis representa cada uno de los ítems de análisis incluidos dentro de una proforma.

- **proforma:** Referencia a la proforma a la que pertenece.
- **parameter / unit / method / technique:** Información técnica del análisis.
- **unit_price:** Precio por unidad de ese análisis.
- **quantity:** Cantidad de veces que se realizará el análisis.
- **subtotal:** Resultado del cálculo unit_price × quantity (se actualiza automáticamente).
- **order:** Número de orden en que se muestra el análisis.
- **created_at:** Fecha de creación del análisis.

2.2.2.7. Modelo Informe

El modelo Informe almacena los informes técnicos finales generados a partir de una proforma ya existente.

- **codigo:** Identificador único del informe (ej. INF-0001).
- **proforma:** Referencia a la proforma de origen (relación uno a uno).
- **created_by:** Nombre del usuario que generó el informe.
- **fecha_emision:** Fecha en que se generó el informe.
- **tomado_por:** Persona responsable de la toma de muestra.

- **procedimiento:** Descripción del procedimiento utilizado.
- **analizado_por:** Nombre del analista responsable.
- **pdf_url:** Ruta del PDF generado para descarga.

2.2.2.8. Modelo Resultado

El modelo Resultado almacena los datos técnicos de cada análisis incluido en un informe.

- **informe:** Referencia al informe al que pertenecen los resultados.
- **parameter / unit / method:** Datos técnicos del análisis realizado.
- **resultados:** Valor obtenido en el análisis.
- **límite:** Límite permisible o normativo para el parámetro.
- **incertidumbre:** Grado de incertidumbre o variación esperada del resultado.

2.2.3. Codificación de Modelos

Los modelos del sistema fueron desarrollados utilizando mongoengine, una herramienta que permite definir documentos en MongoDB con una sintaxis similar a Django ORM. Cada modelo fue ubicado en su propio archivo dentro de la carpeta models/, siguiendo una estructura modular para facilitar su mantenimiento y reutilización. A continuación, se describe cómo se implementaron y cómo se relacionan entre sí:

2.2.3.1. Archivo: user.py

Este archivo contiene la definición del modelo User, encargado de manejar la autenticación, roles y estado de los usuarios. Se utilizaron campos StringField, BooleanField y EmailField con restricciones de unicidad y obligatoriedad. No se

almacena en texto plano la contraseña; esta es cifrada antes de guardarse en el registro mediante vistas de autenticación.

2.2.3.2. Archivo: client.py

Define el modelo Client, utilizado para almacenar la información de cada cliente que solicita análisis. Se implementaron campos de contacto y de seguimiento temporal (created_at, updated_at). Este modelo se relaciona con Proforma como cliente asociado.

2.2.3.3. Archivo: company_settings.py

Contiene la configuración general de la empresa, incluyendo información de contacto, logo y control de numeración de proformas. Se incluye un método interno para generar automáticamente el número de la próxima proforma.

2.2.3.4. Archivo: tipo_muestra.py

Define los tipos de muestras que se pueden analizar. Se usa como base para autocompletar datos en una proforma. Es una tabla de referencia útil para crear análisis estandarizados de forma rápida.

2.2.3.5. Archivo: proforma.py

Modelo central que representa las proformas generadas a los clientes. Incluye lógica para recalcular automáticamente los montos financieros cuando se agregan o modifican análisis.

2.2.3.6. Archivo: analysis.py

Modelo que representa cada análisis individual dentro de una proforma. El método save() actualiza automáticamente el subtotal y llama a recalculate_totals() de la proforma asociada.

2.2.3.7. Archivo: informe.py

Modelo que almacena los informes generados a partir de una proforma. El campo codigo se genera automáticamente con formato incremental (ej. INF-0001).

2.2.3.8. Archivo: resultado.py

Modelo que guarda los resultados detallados del análisis incluido en un informe. Cada campo representa un aspecto técnico del resultado medido, incluyendo su valor e incertidumbre.

2.2.4. Definición de Controladores

Los controladores (vistas) del backend fueron implementados utilizando clases tipo ViewSet de Django REST Framework, lo que permite una estructura clara y reutilizable para manejar operaciones CRUD y acciones personalizadas sobre cada modelo. Las rutas fueron registradas mediante DefaultRouter en el archivo urls.py, permitiendo una integración sencilla entre frontend y backend.

Cada controlador se ubica en un archivo independiente dentro de la carpeta views/, y sigue un patrón uniforme de manejo de datos, validación y respuestas JSON. A continuación, se describen los controladores más relevantes:

2.2.4.1. UserAdminViewSet – Administración de usuarios

Permite listar, crear y modificar usuarios desde una vista exclusiva para administradores.

- **list:** Devuelve todos los usuarios registrados.
- **create:** Registra un nuevo usuario, cifrando la contraseña.
- **update_role:** Cambia el rol (usuario/admin) de un usuario.

- **toggle_active:** Activa o desactiva una cuenta sin eliminarla.

Se accede mediante la ruta: /api/admin/users/

2.2.4.2. ClientViewSet – Gestión de clientes

Maneja el CRUD básico de los clientes y permite buscarlos por nombre o RUC.

- **list:** Devuelve todos los clientes.
- **create:** Registra un nuevo cliente.
- **search:** Permite buscar clientes mediante un parámetro q.

Ruta base: /api/clients/

2.2.4.3. TipoMuestraViewSet – Catálogo de tipos de muestra

Gestiona los tipos de análisis disponibles en el sistema.

- **list, create, retrieve, update, destroy:** Operaciones CRUD completas.
- **search:** ¿Búsqueda por nombre del parámetro (?q=).

Ruta base: /api/tipos-muestra/

2.2.4.4. ProformaViewSet – Gestión de proformas

Permite generar, modificar y visualizar proformas, así como gestionar sus análisis.

- **list:** Muestra todas las proformas registradas.
- **create:** Crea una nueva proforma y genera su PDF.
- **add_analysis:** Agrega un análisis a una proforma específica.
- **remove_analysis:** Elimina un análisis de una proforma.

- **pdf:** Devuelve el PDF de la proforma.
- **informe:** Devuelve los datos necesarios para generar un informe desde una proforma.
- **informe_pdf:** Genera y devuelve el PDF del informe.

Ruta base: /api/proformas/

2.2.4.5. AnalysisViewSet – Reordenamiento de análisis

Permite consultar todos los análisis del sistema y actualizar su orden de visualización.

- **list:** Devuelve todos los análisis.
- **reorder:** Reorganiza el orden de los análisis en una proforma.

Ruta base: /api/analysis/

2.2.4.6. InformeViewSet – Creación y consulta de informes

Gestiona los informes técnicos generados a partir de una proforma.

- **create:** Crea un nuevo informe con datos técnicos.
- **list:** Lista todos los informes generados.
- **resultados:** Devuelve los resultados asociados a un informe específico.

Ruta base: /api/informes/

2.2.4.7. CompanySettingsViewSet – Configuración de empresa

Permite consultar o establecer la configuración institucional.

- **list:** Lista todas las configuraciones.

- **current:** Devuelve (o crea si no existe) la configuración actual.

  Ruta base: /api/settings/

2.2.4.8. Vistas de autenticación y recuperación de contraseña

Estas vistas no usan ViewSet sino clases tipo APIView, registradas manualmente en urls.py.

- **/api/register/:** Registra un nuevo usuario.
- **/api/login/:** Inicia sesión.
- **/api/logout/:** Cierra la sesión.
- **/api/forgot-password/:** Envía un correo con enlace de recuperación.
- **/api/reset-password/<token>/:** Permite restablecer la contraseña mediante un token único.

2.2.5. Codificación de Controladores

La codificación de los controladores se realizó utilizando clases ViewSet de Django REST Framework y APIView para endpoints personalizados. Cada controlador gestiona las operaciones asociadas a una entidad específica del sistema, validando entradas, consultando modelos y devolviendo respuestas en formato JSON. A continuación, se presentan los controladores más representativos:

2.2.5.1. Autenticación: RegisterView, LoginView, LogoutView

Estos controladores gestionan el flujo de autenticación de usuarios mediante clases APIView.

- **RegisterView:** Crea un nuevo usuario con contraseña cifrada.

- **LoginView:** Verifica credenciales y guarda la sesión.
- **LogoutView:** Elimina la sesión activa.

2.2.5.2. Usuarios: UserAdminViewSet

Permite la gestión completa de usuarios por parte de un administrador. Incluye acciones para cambiar roles y activar/desactivar cuentas.

2.2.5.3. Clientes: ClientViewSet

Controlador para registrar y buscar clientes.

- **search:** Permite búsquedas por nombre o RUC con filtros dinámicos.

2.2.5.4. Proformas: ProformaViewSet

Uno de los controladores más importantes, ya que gestiona la creación y modificación de proformas.

- **add_analysis:** Añade análisis a una proforma específica.
- **remove_analysis:** Elimina análisis y regenera el PDF.
- **informe:** Devuelve los datos técnicos para prellenar el informe.
- **pdf e informe_pdf:** Permiten descargar los documentos generados.

2.2.5.5. Informes: InformeViewSet

Permite registrar informes a partir de proformas y consultar resultados.

- **resultados:** Devuelve una lista de objetos Resultado asociados al informe.

2.2.5.6. Tipos de Muestra: TipoMuestraViewSet

Controlador CRUD completo para administrar tipos de análisis. Incluye métodos retrieve, update, destroy y search.

2.2.5.7. Configuración de Empresa: CompanySettingsViewSet

Retorna o inicializa la configuración de la empresa.

2.2.5.8. Análisis: AnalysisViewSet

Permite listar análisis y modificar su orden en las proformas.

2.2.5.9. Recuperación de contraseña: ForgotPasswordView, ResetPasswordView

Permiten iniciar y completar el proceso de restablecimiento de contraseña. El token se genera y envía por correo. Se valida y aplica el cambio de contraseña en ResetPasswordView.

2.2.6. Definición de Serializadores

Los serializadores son componentes fundamentales del backend que permiten transformar los datos entre los modelos de MongoDB (definidos con mongoengine) y las respuestas o solicitudes en formato JSON que maneja el frontend. También se encargan de validar los datos antes de ser procesados y de construir objetos del sistema a partir de las entradas recibidas.

En este sistema, cada entidad relevante del modelo tiene su correspondiente serializador, lo que permite un manejo claro, seguro y consistente de los datos durante las operaciones de creación, lectura, actualización y eliminación (CRUD). Además, se implementaron serializadores auxiliares para vistas específicas como búsquedas o generación de informes.

2.2.6.1. Funciones principales de los serializadores:

- Validar campos antes de guardar en la base de datos.
- Convertir objetos complejos a representaciones JSON legibles.

- Convertir entradas JSON en instancias válidas de modelos MongoDB.
- Controlar qué campos se exponen y en qué formato (ej. ocultar IDs internos, mostrar nombres de relaciones, etc.).
- Ejecutar lógica adicional como la creación de objetos relacionados (Proforma con Analysis, Informe con Resultados, etc.).

Todos los serializadores del proyecto se ubican en la carpeta serializers/ y siguen una estructura modular, con un archivo por entidad. Esta organización mejora la mantenibilidad del código y permite reutilizar validaciones y formatos en distintos endpoints del sistema.

2.2.7. Codificación de Serializadores

Los serializadores (serializers) definen cómo se transforman los datos entre objetos de MongoDB y las respuestas JSON que utiliza el frontend. También permiten validar, crear o actualizar objetos con seguridad. En este proyecto se utilizaron clases personalizadas serializers.Serializer, lo que brinda mayor control sobre la lógica de serialización y deserialización. A continuación, se detallan los principales serializadores implementados:

2.2.7.1. ProformaSerializer y ProformaCreateSerializer

Gestiona la serialización de proformas y sus análisis relacionados. Devuelve datos del cliente (name, ruc) y una lista de análisis. Calcula el nombre del estado (status_display). Genera automáticamente el número de proforma en create. El serializer auxiliar ProformaCreateSerializer se usa específicamente para recibir datos al crear una nueva proforma junto con sus análisis

2.2.7.2. AnalysisSerializer

Serializa los datos de análisis incluidos en cada proforma. Calcula automáticamente el subtotal como unit_price × quantity en el modelo. Puede ser utilizado tanto en ProformaSerializer como en endpoints independientes.

2.2.7.3. InformeSerializer

Serializa informes técnicos generados a partir de proformas. Usa un mini-serializador para mostrar id y proforma_number. El método create() asocia los resultados al informe automáticamente.

2.2.7.4. ResultadoSerializer

Representa los resultados individuales de un informe. Cada resultado está vinculado a un Informe y contiene información técnica detallada.

2.2.7.5. ClientSerializer y ClientSearchSerializer

Gestiona los datos del cliente. Se utilizan dos versiones: una completa y otra reducida para búsqueda rápida. ClientSearchSerializer permite búsquedas ligeras con solo nombre y RUC.

2.2.7.6. TipoMuestraSerializer

Serializa los tipos de muestra predefinidos. Se usa para autocompletar los análisis al generar proformas.

2.2.7.7. CompanySettingsSerializer

Permite visualizar o editar los datos institucionales. Se usa principalmente para mostrar la cabecera en los PDF generados.

2.2.8. Definición de URLs

Las URLs del sistema definen los puntos de acceso que el frontend puede utilizar para interactuar con el backend. Estas rutas fueron configuradas usando Django REST Framework y están organizadas mediante un DefaultRouter, que permite registrar automáticamente las rutas de cada ViewSet sin necesidad de declararlas manualmente.

Además, las vistas de autenticación y recuperación de contraseña, al no pertenecer a un ViewSet, se definieron de forma explícita usando path(). El archivo urls.py se encuentra en la carpeta principal de la app backend (environovalab_app/) y se compone de dos bloques principales:

2.2.8.1. Registro de rutas automáticas con DefaultRouter

Cada ViewSet queda registrado bajo una ruta base y DRF se encarga de generar automáticamente las rutas correspondientes para list, create, retrieve, update, destroy, y acciones personalizadas (@action).

2.2.8.2. Rutas explícitas para autenticación

Estas rutas fueron definidas manualmente, ya que corresponden a clases basadas en APIView. Estas rutas permiten:

- Registrar nuevos usuarios (/register/)
- Iniciar sesión (/login/)
- Cerrar sesión (/logout/)
- Recuperar contraseña (/forgot-password/)
- Restablecer contraseña con token (/reset-password/<token>/)

2.2.9. Codificación de URLs

La codificación de URLs en el sistema backend se realizó utilizando Django REST Framework junto con el enrutador DefaultRouter. Esta estrategia permite mapear automáticamente las rutas para cada ViewSet, reduciendo la necesidad de escribir manualmente cada endpoint CRUD. Para vistas personalizadas como autenticación y recuperación de contraseña, se utilizaron rutas definidas con path().

El archivo urls.py principal agrupa todas las rutas y actúa como punto de entrada para las solicitudes HTTP que llegan al backend.

2.2.9.1. Archivo: urls.py

Se importan los módulos necesarios y los ViewSets definidos en views/.

2.2.9.2. Registro de rutas REST con DefaultRouter

Cada registro define la ruta base (r'proformas'), el ViewSet que la manejará (ProformaViewSet), un nombre (basename) para uso interno (ej. reverso de URL o pruebas). Esto genera automáticamente rutas como:

- GET /proformas/
- POST /proformas/
- GET /proformas/{id}/informe/
- GET /admin/users/

2.2.9.3. Rutas personalizadas para autenticación y contraseñas

Estas rutas están asociadas a clases APIView, y no forman parte de un ViewSet. Permiten:

- Registrar usuarios (POST /register/)
- Iniciar sesión (POST /login/)
- Cerrar sesión (POST /logout/)
- Recuperar contraseña por correo (POST /forgot-password/)
- Establecer nueva contraseña con token (POST /reset-password/<token>/)

2.2.10. Anexos

2.2.10.1. Repositorio del Proyecto en GitHub

Para facilitar el acceso al código fuente del sistema completo, se ha publicado un repositorio en la plataforma GitHub.

- https://github.com/Alee23432444444444444444444/tesis-integracion.git

3. Capítulo III

3.1. Pruebas y Estabilización Frontend

3.1.1. Inicio de Sesión de Usuario

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso evalúa que el formulario de inicio de sesión funcione correctamente. Se verifica que los campos sean válidos, que el sistema reconozca credenciales existentes y redirija al usuario según su rol (administrador o usuario normal).	
Precondiciones	• El sistema debe estar desplegado y operativo en el navegador. • El servidor backend debe estar activo y responder solicitudes. • El usuario debe estar previamente registrado con correo y contraseña válidos.
Pasos para seguir	1. Abrir el navegador y acceder al sistema. 2. Ingresar el correo electrónico del usuario. 3. Ingresar la contraseña asociada. 4. Presionar el botón “Iniciar Sesión”. 5. Esperar la validación del sistema y observar el resultado.
Datos de Entrada	• username: apbarrio • Contraseña: admin123
Resultados Esperados	• El sistema valida que los campos estén completos. • El correo y la contraseña coinciden con los registrados. • Se muestra un mensaje de inicio de sesión exitoso. • El usuario es redirigido automáticamente a su panel según su rol.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • Todos los campos del formulario están llenos. • El formato del correo es válido. • Las credenciales coinciden con un usuario registrado. • El sistema muestra un mensaje de bienvenida. • Se redirige al panel correspondiente del usuario. RECHAZO: • Falta completar uno o más campos del formulario. • El correo electrónico no tiene el formato correcto. • La contraseña no coincide con la registrada. • El servidor no responde o hay error de conexión. • El usuario no es redirigido y aparece un mensaje de error.
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 13: Prueba Funcional 01

3.1.2. Registro de Nuevo Usuario (desde el panel del Administrador)

DESCRIPCIÓN DEL CASO DE PRUEBA Detalle el caso de prueba que va a realizar: Este caso verifica que el administrador pueda registrar un nuevo usuario desde la sección de gestión. Se valida que se completen correctamente los campos del formulario, que se asigne un rol, y que el sistema responda confirmando el registro exitoso.	
Precondiciones	• El sistema debe estar funcionando correctamente. • El backend debe estar activo y la base de datos accesible. • El usuario que inicia sesión debe tener rol de administrador. • El punto final /api/auth/admin/users/ debe estar disponible.
Pasos para seguir	1. Iniciar sesión en el sistema como administrador. 2. Dirigirse al módulo de "Usuarios" desde el menú lateral. 3. Hacer clic en el botón "Crear Usuario". 4. Completar el formulario del modal con: nombre de usuario, correo, contraseña, rol y estado. 5. Presionar el botón "Guardar". 6. Verificar si aparece un mensaje de éxito y si el nuevo usuario aparece en la tabla.
Datos de Entrada	• Nombre de usuario: informes1 • Correo electrónico: apbarrio@puce.edu.ec • Contraseña: inf123456 • Rol: admin/usuario
Resultados Esperados	• El formulario valida que todos los campos estén llenos y con formatos válidos. • El correo no debe estar repetido. • El servidor recibe los datos correctamente y crea el nuevo usuario. • El sistema muestra un mensaje de confirmación del registro. • El nuevo usuario aparece inmediatamente en la lista de usuarios.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • Todos los campos obligatorios fueron completados correctamente. • El rol asignado es válido y reconocido por el sistema. • No se repite el correo electrónico. • Se muestra mensaje de éxito. • El usuario aparece en la tabla con su rol y estado. RECHAZO: • Falta uno o más campos obligatorios. • El formato del correo es incorrecto.

	• El correo ya está registrado. • El servidor no responde. • No se muestra mensaje de confirmación ni se agrega el usuario.
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 14: Prueba Funcional 02

3.1.3. Recuperar contraseña

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que cualquier usuario registrado pueda recuperar su contraseña utilizando el formulario disponible en la página de "¿Olvidaste tu contraseña?". Se evalúa que el sistema permita ingresar un correo válido, envíe un enlace de recuperación, y permita establecer una nueva contraseña de forma correcta.	
Precondiciones	• El sistema debe estar funcionando correctamente. • El backend debe estar activo y la base de datos accesible. • El usuario debe estar registrado previamente en el sistema. • Debe tener acceso al correo electrónico vinculado a su cuenta.
Pasos para seguir	1. Seleccionar Olvide mi contraseña. 2. Ingresar un correo electrónico válido. 3. Presionar el botón “Enviar”. 4. Revisar la bandeja de entrada del correo electrónico proporcionado. 5. Hacer clic en el enlace de recuperación recibido. 6. Ingresar una nueva contraseña y confirmarla. 7. Presionar “Guardar nueva contraseña”.
Datos de Entrada	• Correo electrónico: apbarrio@environovalab.com • Nueva Contraseña: admin1234
Resultados Esperados	• El sistema muestra un mensaje confirmando que se ha enviado el correo. • El usuario recibe un correo con un enlace válido. • Al hacer clic, accede al formulario de nueva contraseña. • El sistema actualiza correctamente la contraseña. • El usuario puede iniciar sesión con la nueva clave.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El campo de correo del formulario está lleno. • El formato del correo es válido. • El correo coincide con un usuario registrado. • El sistema muestra un mensaje El correo se envía exitosamente. • Se redirige al panel correspondiente del usuario.

	• El enlace de recuperación lleva al formulario de restablecimiento. • La nueva contraseña se guarda correctamente y permite iniciar sesión. RECHAZO: • Falta completar un campo del formulario. • El correo electrónico no tiene el formato correcto. • El correo no está registrado en el sistema. • El servidor no responde o hay error de conexión. • El campo de nueva contraseña está vacío o es inválido.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 15: Prueba Funcional 03

3.1.4. Dashboard visualización reciente de informes y proformas.

DESCRIPCIÓN DEL CASO DE PRUEBA Detalle el caso de prueba que va a realizar: Este caso verifica que el usuario pueda visualizar lo reciente que se ha creado de proformas e informes, filtrarlos por medio de la barra de búsqueda son los últimos siete informes y proformas creados eso puede ver el admin, vuelta en el usuario normal son los últimos siete informes creados.	
Precondiciones	• El usuario debe haber iniciado sesión como administrador en el sistema si quiere ver las proformas e informes. • El sistema debe contener registros de proformas e informes guardados previamente los últimos siete. • El backend debe estar disponible y operativo.
Pasos para seguir	1. Acceder a la pantalla principal 'Dashboard'. 2. Escribir el nombre del cliente o número de documento en la barra de búsqueda. 3. Observar la tabla de resultados filtrados.
Datos de Entrada	• Búsqueda en el search: Número de proforma/informe
Resultados Esperados	• Se muestra correctamente la lista de documentos filtrados. • El usuario puede ver sus documentos asociados.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • La búsqueda arroja resultados correctos. • El sistema muestra la información según el rol del usuario. RECHAZO: • La búsqueda no devuelve ningún resultado válido.
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 16: Prueba Funcional 04

3.1.5. Gestión de Tipos de Muestra

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que un usuario con rol administrador pueda acceder a la sección de Tipos de Muestra desde el panel lateral, visualizar los registros actuales y realizar operaciones de creación, edición y eliminación de muestras. El sistema debe validar que solo los administradores tengan acceso y que todos los campos sean requeridos para registrar una muestra correctamente.	
Precondiciones	• El sistema debe estar funcionando correctamente. • El backend debe estar activo y la base de datos accesible. • El usuario debe haber iniciado sesión como administrador • El usuario debe tener acceso al panel lateral y permisos para visualizar la ruta /admin/tipos-muestra.
Pasos para seguir	1. Iniciar sesión como administrador. 2. En el panel lateral, seleccionar la opción “Tipos de Muestra”. 3. Rellenar el formulario con todos los campos obligatorios. 4. Presionar el botón “Agregar”. 5. Confirmar que la muestra se agregue a la lista. 6. Editar un registro existente haciendo clic en el ícono de lápiz, modificar un campo y guardar. 7. Eliminar una muestra haciendo clic en el ícono de papelera. 8. Confirmar que la notificación muestre el resultado de cada acción.
Datos de Entrada	• Tipo: agua • Parámetro: pH • Unidad: U pH • Método: Electrométrico • Técnica: Medición directa • Precio: 12.50
Resultados Esperados	• El sistema muestra correctamente el formulario con íconos. • Solo los administradores acceden a la vista. • Al agregar, editar o eliminar, se muestra una notificación visual con ícono (CheckCircle para éxito, AlertCircle para errores). • Las muestras se listan automáticamente debajo del formulario. • Al modificar una muestra, el formulario en modal se rellena con los datos anteriores. • El backend responde con los datos actualizados (HTTP 200/201) y actualiza el frontend sin errores.

Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El formulario se completa correctamente con todos los campos. • Se muestran notificaciones positivas para cada operación exitosa. • La lista se actualiza automáticamente después de cada acción. • El botón y campos responden correctamente y el modal se cierra tras guardar. RECHAZO: • Campos incompletos en el formulario. • El usuario no tiene rol admin (la ruta no está visible). • Se pierde conexión con el backend. • El backend devuelve error de validación o status 400/500.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 17: Prueba Funcional 05

3.1.6. Registro y Generación de proformas

DESCRIPCIÓN DEL CASO DE PRUEBA Detalle el caso de prueba que va a realizar: Este caso verifica que el usuario pueda crear una proforma completando todos los campos del formulario, incluyendo datos del cliente, tipo de muestra y análisis. Se evalúa que se calcule correctamente el total, se guarde la información y se habilite la descarga del PDF.	
Precondiciones	• El usuario que inicia sesión debe tener rol de administrador. • El usuario debe haber iniciado sesión correctamente. • El sistema debe tener clientes, tipos de muestra y análisis configurados. • El backend debe estar operativo y la conexión activa. • El formulario debe estar completamente llenado con datos válidos.
Pasos para seguir	1. Acceder a la pantalla 'Generar Proforma'. 2. Completar todos los campos del formulario: cliente, tipo de muestra, análisis. 3. Ingresar cada análisis con su parámetro, método, unidad, precio y cantidad. 4. Presionar el botón 'Guardar'. 5. Verificar si se muestra el mensaje de éxito. 6. Confirmar que se habilita el botón de descarga del PDF.
Datos de Entrada	• Cliente: Empresa XYZ • Tipo de muestra: Agua potable

	• Análisis: pH, método: Potenciométrico, unidad: pH, precio: 10 • Correo: cliente@puce.edu.ec • Dirección: Princesa toa • Cantidad de muestra: 7
Resultados Esperados	• La proforma se guarda correctamente en el sistema. • Se calcula el subtotal, IVA y total. • Aparece una alerta visual confirmando el registro. • Se habilita el botón para descargar el PDF.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • Todos los campos están completos y son válidos. • El cálculo del total es correcto. • Se guarda la proforma sin errores. • Aparece el botón para descargar el PDF. RECHAZO: • Faltan campos por completar. • Error en el cálculo de totales. • Falla en la conexión con el servidor. • No aparece el botón de PDF ni mensaje de éxito.
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 18: Prueba Funcional 06

3.1.7. Generación de Informe desde Proforma

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el usuario pueda generar correctamente un informe técnico a partir de una proforma existente. Se evalúa la búsqueda de proformas por número o cliente, la carga de datos asociados, el llenado de resultados y campos técnicos, el guardado del informe en el sistema y la descarga en formato PDF.	
Precondiciones	• El usuario debe haber iniciado sesión. • Deben existir proformas previas registradas en el sistema. • El endpoint /api/proformas/ debe estar activo y retornar datos válidos. • El backend debe exponer /api/informes/ y /api/proformas/<id>/informe_pdf/.
Pasos para seguir	1. Acceder a la ruta /informes desde el panel lateral. 2. Escribir parte del número o nombre del cliente en el buscador. 3. Hacer clic en “Buscar” y seleccionar una proforma de los resultados. 4. Visualizar los datos cargados automáticamente. 5. Llenar los campos “Muestra tomada por”, “Procedimiento” y resultados (Resultado, Límite, Incertidumbre) para cada parámetro.

	6. Presionar el botón “Guardar Informe”. 7. Si el guardado es exitoso, hacer clic en “Descargar PDF”. 8. (Opcional) Presionar “Limpiar Todo” para reiniciar el formulario.
Datos de Entrada	• Búsqueda: PRF-0012 o Cliente S.A. • Muestra tomada por: Ing. Mario Ruiz • Procedimiento: Método EPA 123 • Resultado: 7.5, Límite: 8.0, Incertidumbre: ±0.1
Resultados Esperados	• El sistema filtra correctamente las proformas por coincidencia de código o cliente. • Los datos asociados se cargan al seleccionar una proforma. • Los campos obligatorios deben completarse para permitir el guardado. • Se muestra una notificación exitosa al guardar el informe. • Al hacer clic en “Descargar PDF”, se abre el informe generado. • El botón “Limpiar Todo” limpia todos los campos y estados visuales.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El sistema permite buscar y seleccionar proformas existentes. • Los campos se completan sin errores y se muestran correctamente. • El botón “Guardar Informe” funciona solo si todos los campos obligatorios están llenos • El PDF generado está disponible tras guardar. RECHAZO: • No se selecciona ninguna proforma. • Faltan campos obligatorios en cliente o análisis. • Error de conexión o respuesta del backend al guardar. • No se genera el PDF tras guardar correctamente.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 19: Prueba Funcional 07

3.1.8. Visualizar y gestionar lista de Proformas

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que los usuarios puedan visualizar correctamente la lista de proformas recientes en una tabla interactiva. Se evalúa la funcionalidad de búsqueda por código, el cambio de estado por parte de administradores y la descarga de cada documento en formato PDF. La vista se adapta según el rol del usuario.	
Precondiciones	• El backend debe estar activo y exponer el endpoint /api/proformas/ y /api/proformas/<id>/pdf/. • El usuario debe haber iniciado sesión.

	• Debe haber proformas registradas previamente. • El rol del usuario debe estar definido en localStorage como "admin" o "user".
Pasos para seguir	1. Iniciar sesión y acceder a la ruta /proformaslist desde el panel lateral. 2. Visualizar el saludo personalizado con el nombre del usuario. 3. Usar el campo de búsqueda para filtrar por código de proforma. 4. Verificar que se muestren los campos: Tipo, Código, Fecha, Estado, Creado por y Acción. 5. Si el usuario es admin, cambiar el estado de una proforma con el selector desplegable. 6. Hacer clic en el botón “Descargar” para obtener el PDF de una proforma. 7. Comprobar que el archivo se descarga correctamente.
Datos de Entrada	• Código de búsqueda: PRO-0001 • Estado a seleccionar: "En progreso" • Proforma de ejemplo: id = 64eb..., status = "Por hacer", proforma_number = "PRO-0001"
Resultados Esperados	• La tabla muestra las proformas más recientes, ordenadas por fecha. • La búsqueda filtra correctamente por coincidencia parcial de código. • El selector de estado se muestra solo si el usuario es admin. • Al cambiar el estado, se actualiza visualmente en la tabla y en el backend. • Al descargar una proforma, se obtiene un archivo PDF nombrado con el código correspondiente.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • La vista se carga sin errores. • El usuario ve su nombre correctamente en el saludo. • La búsqueda es dinámica y efectiva. • Los administradores pueden cambiar el estado de cada proforma. • El botón de descarga funciona correctamente y obtiene el archivo PDF. RECHAZO: • No se listan proformas, aunque existan. • La búsqueda no responde o devuelve mal los datos. • El selector de estado no está disponible para admins. • El botón de descarga no responde o lanza errores. • Se descarga un archivo con nombre erróneo o vacío.

Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 20: Prueba Funcional 08

3.1.9. Visualizar y gestionar lista de Informes

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que los usuarios puedan visualizar la lista de informes técnicos generados, buscar por código, modificar el estado (solo si el usuario tiene rol de administrador) y descargar el archivo PDF del informe. Se prueba la interacción completa con la tabla dinámica.	
Precondiciones	• El usuario debe haber iniciado sesión correctamente. • Deben existir informes previos registrados en el sistema. • El backend debe exponer los endpoints /api/informes/ y /api/proformas/<id>/informe_pdf/. • El campo proforma debe estar vinculado a cada informe para permitir la descarga.
Pasos para seguir	1. Acceder a la ruta /informeslist desde el panel lateral. 2. Visualizar el saludo con el nombre del usuario actual. 3. Usar la barra de búsqueda para filtrar por código del informe. 4. Visualizar los informes en la tabla con columnas: Tipo, Código, Fecha, Estado, Creado por y Acción. 5. Si el usuario tiene rol admin, modificar el estado desde el selector desplegable. 6. Hacer clic en “Descargar” para obtener el informe en PDF. 7. Confirmar que el archivo se descarga correctamente.
Datos de Entrada	• Código de búsqueda: INF-0004 • Nuevo estado seleccionado: Terminado • Informe con proforma.id = "64eb...", codigo = "INF-0004"
Resultados Esperados	• La tabla muestra correctamente los informes más recientes ordenados por fecha. • La búsqueda filtra dinámicamente los resultados por coincidencia de código. • Los administradores pueden modificar el estado del informe y el cambio se refleja al instante. • El botón de descarga obtiene el archivo PDF relacionado con la proforma correspondiente. • Si no hay informes que coincidan con la búsqueda, se muestra un mensaje de “No se encontraron documentos.”
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • La vista se carga correctamente y muestra los informes recientes. • La búsqueda responde al escribir. • La búsqueda es dinámica y efectiva.

	• Solo los administradores ven y usan el selector de estado. • El cambio de estado se guarda correctamente vía API • El informe se descarga como archivo PDF nombrado con su código. RECHAZO: • La tabla está vacía sin motivo (datos no cargados). • El filtro no responde o arroja errores. • El selector de estado no está visible para admin. • La descarga de PDF falla por error de proforma o backend.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 21: Prueba Funcional 09

3.2. Pruebas y Estabilización Backend

3.2.1. Registro de nuevo usuario

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el backend permita registrar un nuevo usuario a través del endpoint de API pública /api/auth/register/. Se evalúa el comportamiento ante datos válidos, campos faltantes y validaciones como existencia previa de correo o nombre de usuario.	
Precondiciones	• El servidor backend debe estar en ejecución. • El endpoint /api/auth/register/ debe estar disponible. • La base de datos debe permitir la creación de nuevos usuarios. • El usuario que se va a registrar no debe existir previamente (correo o nombre de usuario).
Pasos para seguir	1. Enviar una solicitud POST al endpoint /api/auth/register/. 2. Incluir los campos obligatorios en formato JSON: - username - password - email 3. Enviar también opcionalmente: - is_admin (boolean) 4. Verificar la respuesta HTTP del servidor. 5. Comprobar que el usuario se haya guardado correctamente en la base de datos.
Datos de Entrada	• Solicitud válida: json { “username": "nuevo_usuario", "password": "ClaveSegura123",

	"email": "nuevo@correo.com", "is_admin": false} • Solicitud inválida: json { "username": "incompleto", "password": "123456"} • Usuario inactivo: Usuario con is_active = False •
Resultados Esperados	**Registro exitoso:** • Código de estado 201 • Respuesta: {"msg": "Usuario creado correctamente"} **Faltan campos requeridos:** • Código 400 • Respuesta: {"error": "Faltan campos requeridos"} **Usuario o email ya registrado:** • Código 400 • Respuesta: {"error": "Usuario ya existe"} o {"error": "Email ya está en uso"}
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • Se envían todos los campos obligatorios. • El usuario no existe previamente en el sistema. • El backend devuelve 201 y confirma el registro. RECHAZO: • Faltan campos como email, username o password. • El username o email ya existen. • Error interno del servidor (500) o de validación (400).
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 22: Prueba Funcional 10

3.2.2. Autenticación de Usuario

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que la API de autenticación del backend permita a un usuario existente iniciar sesión correctamente usando su nombre de usuario y contraseña. También se evalúa que el sistema maneje adecuadamente errores como credenciales incorrectas, usuarios inactivos o campos faltantes. No se involucra interfaz gráfica en este caso, solo la interacción directa con los endpoints.	
Precondiciones	• El servidor backend debe estar ejecutándose correctamente. • La base de datos debe contener al menos un usuario válido con estado is_active = True.

	• El endpoint /api/auth/login/ debe estar expuesto y funcional.

Pasos para seguir	1. Enviar una solicitud POST al endpoint /api/auth/login/.
	2. En el body de la solicitud, incluir los campos:
	{
	Username: adminuser
	password: admin123
	}
	3. Evaluar la respuesta recibida:
	- Si es exitosa (HTTP 200), debe devolver el nombre de usuario y rol.
	- Si falla, debe indicar el motivo (credenciales inválidas, cuenta inactiva, etc.).

Datos de Entrada	• Solicitud válida:
	json{
	"username": "adminuser",
	"password": "admin123”}
	• Solicitud inválida:
	json {
	“username": "adminuser",
	"password": "claveIncorrecta”}
	• Usuario inactivo: Usuario con is_active = False
	•

Resultados Esperados	**Login correcto:** • Código de estado 200. • Respuesta JSON: json{ "msg": "Login correcto", "username": "adminuser", "is_admin": true } **Credenciales incorrectas:** • Código de estado 400. • Mensaje: "Credenciales incorrectas" **Usuario inactivo:** • Código de estado 403. • Mensaje: "Cuenta inactiva. Contacta con el administrador."
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El usuario existe y está activo. • El nombre de usuario y la contraseña coinciden con los almacenados.

	• El backend responde con un estado 200 y la información del usuario. RECHAZO: • Error 401 datos invalidos • El usuario no existe o la contraseña es incorrecta. • El usuario está inactivo (is_active = False). • Faltan campos obligatorios en la solicitud.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 23: Prueba Funcional 11

3.2.3. Cierre de sesión de usuario

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el backend cierre correctamente la sesión del usuario al recibir una solicitud al endpoint /api/auth/logout/. Se evalúa que los datos de sesión sean eliminados y que el servidor responda con un mensaje de confirmación.	
Precondiciones	• El usuario debe haber iniciado sesión previamente. • La sesión debe estar activa en el servidor (cookie o token válido). • El endpoint /api/auth/logout/ debe estar disponible y funcional.
Pasos para seguir	1. Iniciar sesión en el sistema para generar una sesión activa. 2. Enviar una solicitud POST al endpoint /api/auth/logout/ con las cookies o encabezados necesarios. 3. Verificar que la sesión del usuario se elimine del servidor. 4. Comprobar que la respuesta sea exitosa y contenga un mensaje de confirmación.
Datos de Entrada	• **Ejemplo de solicitud:** Método: POST URL: /api/auth/logout/ Encabezado: Cookie: sessionid=abc123... • **Cuerpo:** *(vacío)* •
Resultados Esperados	• **Cierre exitoso:** Código de estado: 200 Respuesta JSON: json { "msg": "Sesión cerrada"} • La sesión del usuario ya no existe en el servidor.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El usuario tenía sesión activa.

	• Se envió la solicitud con credenciales válidas. • El backend elimina correctamente la sesión. • La respuesta devuelve {"msg": "Sesión cerrada"}. RECHAZO: • No había sesión activa. • Se omiten cookies de sesión. • El servidor lanza error interno.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 24: Prueba Funcional 12

3.2.4. Solicitud de recuperación de contraseña

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita a un usuario registrado iniciar el proceso de recuperación de contraseña a través del endpoint /api/auth/forgot-password/. Se evalúa que, al enviar un correo válido, el sistema genere un token y envíe un enlace de restablecimiento al email proporcionado.	
Precondiciones	• El backend debe estar en ejecución y configurado con un servidor SMTP funcional. • El usuario debe estar registrado previamente en la base de datos con un correo válido. • El endpoint /api/auth/forgot-password/ debe estar habilitado.
Pasos para seguir	1. Enviar una solicitud POST al endpoint /api/auth/forgot-password/. 2. Incluir en el cuerpo JSON el campo email. 3. El servidor valida que el correo exista en el sistema. 4. Si existe, genera un token y envía un correo con el enlace de recuperación. 5. Verificar la respuesta del backend y la llegada del correo.
Datos de Entrada	• **Ejemplo válido:** json {"email": "usuario@correo.com"} • **Ejemplo inválido (campo vacío):** json {"email": ""}
Resultados Esperados	**Correo válido:** • Código de estado: 200 • Respuesta: {"msg": "Correo enviado. Revisa tu correo para recuperar tu cuenta."} • El usuario recibe un correo con un enlace que contiene un token. **Correo inexistente:** • Código: 404 • Respuesta: {"error": "Usuario no encontrado con ese correo"

	Correo vacío: • Código: 400 • Respuesta: {"error": "El campo correo electrónico es requerido"}
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El correo corresponde a un usuario registrado. • Se genera correctamente el token y se almacena temporalmente. • El correo llega al destinatario con el enlace de recuperación. RECHAZO: • El campo email está vacío o mal escrito. • El correo no pertenece a ningún usuario registrado. • Error de conexión al servidor SMTP o fallo en el envío del correo.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 25: Prueba Funcional 13

3.2.5. Restablecer contraseña con token

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el usuario pueda restablecer su contraseña utilizando el enlace recibido por correo electrónico que contiene un token único. El endpoint /api/auth/reset-password/<token>/ permite establecer una nueva contraseña válida y actualizarla en la base de datos.	
Precondiciones	• El usuario debió haber solicitado previamente la recuperación de contraseña. • Debe existir un token válido y activo en el diccionario RESET_TOKENS del backend. • El backend debe estar corriendo y conectado correctamente a la base de datos.
Pasos para seguir	1. Obtener el token de recuperación generado anteriormente (/api/auth/forgot-password/). 2. Enviar una solicitud POST al endpoint /api/auth/reset-password/<token>/. 3. Incluir en el cuerpo de la solicitud el campo password con la nueva contraseña. 4. Verificar la respuesta del backend y que el token sea eliminado tras su uso. 5. Intentar iniciar sesión con la nueva contraseña para confirmar el cambio.
Datos de Entrada	• **Token válido:** 59aeb9e0-c9a3-4e5e-8473-9cf0ad123456 • **Cuerpo de la solicitud:** json {"password": "NuevaClave2024"}

Resultados Esperados	**Token y contraseña válidos:** • Código de estado: 200 • Respuesta: {"msg": "Contraseña actualizada correctamente"} • El token es eliminado del diccionario • RESET_TOKENS. **Token inválido o expirado:** • Código: 400 • Respuesta: {"error": "Token inválido o expirado"}
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El token es válido y corresponde a un usuario. La contraseña es enviada correctamente en el cuerpo de la solicitud. • El sistema actualiza la contraseña del usuario y elimina el token. RECHAZO: • El token no existe, es incorrecto o ya fue usado. • El campo password está vacío o malformado. • Error interno del backend al guardar la nueva contraseña.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 26: Prueba Funcional 14

3.2.6. Registro de tipo de muestra (API - Backend)

DESCRIPCIÓN DEL CASO DE PRUEBA **Endpoint**: POST /api/tipos-muestra/ Este caso valida que el sistema permita agregar nuevos tipos de muestra al catálogo. El registro debe incluir nombre, parámetro, unidad, método, técnica y precio. También se verifican errores por campos faltantes o duplicados.	
Precondiciones	• El usuario debe tener rol de administrador. • El endpoint /api/tipos-muestra/ debe estar operativo. • La base de datos debe estar conectada. • El tipo de muestra no debe estar previamente registrado.
Pasos para seguir	1. Autenticarse como administrador. 2. Enviar una solicitud POST a /api/tipos-muestra/. 3. En el cuerpo de la solicitud incluir todos los datos del análisis. 4. Esperar respuesta del servidor. 5. Verificar que se retorne un código 201 y el nuevo registro.
Datos de Entrada	{ "tipo": "Agua Residual",

	"parametro": "DBO", "unidad": "mg/L", "metodo": "5210B", "tecnica": "Titulación", "precio": 15.50 } •
Resultados Esperados	• Código HTTP `201 Created` • Respuesta JSON con el tipo de muestra registrado • El dato aparece al listar los tipos (GET /api/tipos-muestra/
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • Todos los campos están completos y válidos • No se repite el tipo ya existente • El backend responde con éxito RECHAZO: • Faltan campos obligatorios (ej. método) • Campo duplicado (tipo o parámetro repetido) • Error de validación (precio no numérico) • El usuario no tiene permisos
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 27: Prueba Funcional 15

3.2.7. Listar los tipos de muestra (API - Backend)

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que la API devuelva correctamente la lista de todos los tipos de muestra registrados en el sistema. Se evalúa que el backend responda con código 200, que el formato sea JSON y que se retornen los datos completos de cada tipo de muestra. También se validan posibles errores si el usuario no está autenticado o si hay problemas de conexión.	
Precondiciones	• El servidor backend debe estar en ejecución. • El usuario debe estar autenticado (si el endpoint está protegido). • Deben existir uno o más tipos de muestra previamente registrados.

	• El endpoint /api/tipos-muestra/ debe estar habilitado y funcionando.
Pasos para seguir	1. Iniciar sesión en el sistema o usar un token válido (si se requiere autenticación). 2. Enviar una solicitud GET a la URL:http://localhost:8000/api/tipos-muestra/ 3. Evaluar la respuesta del servidor: • Código de estado HTTP • Estructura del JSON • Contenido del array (al menos un objeto con los campos esperados) .
Datos de Entrada	No aplica •
Resultados Esperados	• Código http 200 ok • Que se refleje las muestras • No hay campos faltantes ni errores de formato.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El servidor responde con código 200 OK. • La estructura del JSON es una lista de objetos. • Los campos de cada objeto están completos (tipo, parámetro, etc.). • El número de registros coincide con lo esperado en la base de datos. RECHAZO: • El servidor responde con error 401 Unauthorized si el usuario no está autenticado. • El servidor devuelve 500 Internal Server Error si hay falla de conexión. • La lista está vacía cuando debería contener elementos. • Alguno de los objetos no tiene todos los campos requeridos. • La respuesta no es un JSON válido.
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 28: Prueba Funcional 16

3.2.8. Actualización de tipo de muestra (API - Backend)

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que la API permita actualizar correctamente los datos de un tipo de muestra existente. Se evalúa que el sistema valide los datos recibidos, actualice el registro en la base de datos, y devuelva una respuesta con los nuevos datos. También se contempla qué sucede si se envían datos inválidos o si el ID no existe.	
Precondiciones	• El servidor backend debe estar activo y conectado a la base de datos. • Debe existir al menos un tipo de muestra previamente registrado. • El usuario debe estar autenticado y autorizado (rol administrador). • El endpoint /api/tipos-muestra/{id}/ debe estar expuesto y aceptando método PUT.
Pasos para seguir	1. Iniciar sesión como administrador. 2. Obtener el id de un tipo de muestra ya registrado. 3. Enviar una solicitud PUT a: http://localhost:8000/api/tipos-muestra/{id}/ 4. Incluir en el cuerpo de la solicitud los nuevos datos válidos. 5. Verificar que el servidor responda con código 200 OK y los datos actualizados. 6. Confirmar que los cambios se reflejen al volver a listar (GET /api/tipos-muestra/).

Datos de Entrada	{
	"tipo": "agua",
	"parametro": "Conductividad",
	"unidad": "uS/cm",
	"metodo": "Método EPA 120.1",
	"tecnica": "Sonda multiparamétrica",
	"precio": 12.50
	}

Resultados Esperados	• Código 200 OK al actualizar. • Devuelve los datos actualizados en JSON. • Los cambios se reflejan al listar los tipos.

Criterios de Aceptación / Rechazo	ACEPTACIÓN • Se actualizan correctamente los campos del tipo de muestra. • El ID enviado es válido y corresponde a un registro existente. • El sistema responde con código 200 OK y los nuevos datos. • La base de datos refleja los cambios inmediatamente. RECHAZO: • El ID no existe → Error 404 Not Found • Faltan campos requeridos → Error 400 Bad Request • El usuario no tiene permisos → Error 403 Forbidden • La sesión ha expirado → Error 401 Unauthorized • El servidor devuelve 500 por un error interno
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 29: Prueba Funcional 17

3.2.9. Eliminación del tipo de muestra (API - Backend)

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita eliminar correctamente un tipo de muestra registrado, usando su ID. Se valida que el backend procese la solicitud, elimine el registro de la base de datos y retorne una confirmación adecuada. También se evalúan errores como intento de eliminación con ID inexistente o sin permisos.	
Precondiciones	• El servidor backend debe estar ejecutándose. • El usuario debe estar autenticado y tener permisos de administrador. • Debe existir al menos un tipo de muestra registrado en la base de datos. • El endpoint DELETE /api/tipos-muestra/{id}/ debe estar habilitado y operativo.
Pasos para seguir	1.Iniciar sesión como administrador. 2.Obtener el id de un tipo de muestra ya registrado. 3.Enviar una solicitud PUT a: http://localhost:8000/api/tipos-muestra/{id}/ 4. Incluir en el cuerpo de la solicitud los nuevos datos válidos.

	5. Verificar que el servidor responda con código 200 OK y los datos actualizados. 6. Confirmar que los cambios se reflejen al volver a listar (GET /api/tipos-muestra/).
Datos de Entrada	• **Método:** DELETE • **URL:** http://localhost:8000/api/tipos-muestra/{id}/ • **Headers:** Content-Type: application/json X-CSRFToken: <token> Cookie de sesión activa •
Resultados Esperados	• Código HTTP: 204 No Content • El tipo de muestra desaparece de la lista (GET). • No se retorna cuerpo de respuesta, solo éxito. • La base de datos refleja que fue eliminado.
Criterios de Aceptación / Rechazo	ACEPTACIÓN • El ID existe y corresponde a un registro válido. • El usuario está autenticado y tiene permisos. • El backend elimina el tipo correctamente y devuelve código 204. RECHAZO: • El ID no existe → error 404 Not Found. • No se elimino • El usuario no tiene permisos → 403 Forbidden. • El usuario no está autenticado → 401 Unauthorized.
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 30: Prueba Funcional 18

3.2.10. Listar los usuarios (API - Backend)

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema devuelva correctamente la lista de usuarios registrados cuando se hace una solicitud GET al endpoint de administración. Se evalúa que los datos estén completos, el formato sea JSON, y se respete el control de acceso (solo administradores pueden acceder).	
Precondiciones	• El usuario debe estar autenticado como administrador.

	• El servidor backend debe estar funcionando. • Debe haber al menos un usuario registrado. • El endpoint /api/admin/users/ debe estar habilitado.
Pasos para seguir	1. Iniciar sesión como usuario con rol de **administrador**. 2. Realizar una solicitud GET a:http://localhost:8000/api/admin/users/ 3. Verificar el código de estado. 4. Revisar que el cuerpo de la respuesta sea una **lista de usuarios**. 5.Confirmar que cada objeto contiene los campos esperados: • id, username, email, is_admin, activo.
Datos de Entrada	No aplica •
Resultados Esperados	• Codigo htto 200 ok • Que se refleje la lista de usuarios registrados
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El usuario tiene permisos (es administrador). • El servidor responde con código 200. • Los datos están completos y bien formateados. • La cantidad de usuarios coincide con la base de datos. RECHAZO: • El usuario no es administrador → 403 Forbidden • No se reflejan los usuarios • No está autenticado → 401 Unauthorized • El servidor devuelve error → 500 Internal Server Error • La respuesta no es JSON o está incompleta
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 31: Prueba Funcional 19

3.2.11. Registro de nuevo usuario (API - Backend)

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita crear un nuevo usuario desde el panel de administración mediante una solicitud POST a la API. Se evalúa que los campos requeridos sean validados correctamente y que el usuario sea creado exitosamente en la base de datos.	
Precondiciones	• El usuario actual debe estar autenticado como administrador. • El endpoint /api/admin/users/ debe estar disponible y aceptando solicitudes POST. • No debe existir otro usuario con el mismo nombre o correo electrónico. • La base de datos debe estar conectada y funcional.

Pasos para seguir	1. Iniciar sesión como administrador. 2. Enviar una solicitud POST al endpoint: http://localhost:8000/api/admin/users/ 3.Incluir en el cuerpo los siguientes campos JSON: • username, email, password, is_admin 4. Verificar que el servidor responda con código 201 Created. 5. Confirmar que el nuevo usuario aparezca al consultar /api/admin/users/.

Datos de Entrada	{
	"username": "nuevo_user",
	"email": "nuevo@correo.com",
	"password": "12345678",
	"is_admin": false
	}
	•

Resultados Esperados	• Todos los campos requeridos fueron completados • El correo electrónico y el nombre de usuario no están duplicados • El sistema guarda correctamente el usuario • El backend responde con estado 201
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El usuario tiene permisos (es administrador). • El servidor responde con código 200. • Los datos están completos y bien formateados. • La cantidad de usuarios coincide con la base de datos. RECHAZO: • Faltan campos obligatorios → 400 Bad Request • El correo o nombre de usuario ya existen → 409 Conflict o 400 • El usuario no está autenticado → 401 Unauthorized • No tiene permisos → 403 Forbidden • Error del servidor → 500 Internal Server Error
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 32: Prueba Funcional 20

3.2.12. Actualización del rol delusuario (API - Backend)

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que un administrador pueda cambiar el rol de un usuario (de usuario normal a administrador o viceversa) mediante la API. Se evalúa que el sistema realice el cambio correctamente y que el usuario no pueda modificarse a sí mismo.	
Precondiciones	• El usuario actual debe estar autenticado como administrador. • Debe existir al menos un usuario distinto del administrador actual. • El endpoint /api/admin/users/{id}/update_role/ debe estar operativo. • El campo is_admin debe enviarse en el cuerpo de la solicitud.
Pasos para seguir	1. Iniciar sesión como administrador. 2. Obtener el id del usuario al que se desea cambiar el rol. 3. Enviar una solicitud POST a: http://localhost:8000/api/admin/users/{id}/update_role/ 4. En el body, incluir el campo is_admin con el nuevo valor (true o false). 5. Verificar si el rol cambia y el servidor responde con código 200.

Datos de Entrada	{
	"is_admin": true
	}
	•

Resultados Esperados	• Código HTTP: 200 OK • El usuario cambia de rol correctamente (admin ↔ usuario). • Se muestra mensaje “Rol actualizado” en la interfaz. • Al volver a listar los usuarios, el cambio se refleja visualmente.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El usuario objetivo no es el mismo que quien está logueado. • El campo `is_admin` es booleano y válido. • El servidor responde `200` y se guarda el cambio. RECHAZO:

	• Se intenta cambiar el rol del usuario actual (no permitido). • No se modifica el rol del usuario • El usuario no tiene permisos → `403 Forbidden`. • El ID es inválido o no existe → `404 Not Found`. • Faltan datos o hay errores en la solicitud → `400 Bad Request`.
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 33: Prueba Funcional 21

3.2.13. Inactivación del usuario (API - Backend)

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita a un administrador cambiar el estado de un usuario (activo ↔ inactivo). Se valida que el backend actualice correctamente el estado en la base de datos y que el cambio se refleje en la interfaz.	
Precondiciones	• El usuario debe estar autenticado como administrador. • El endpoint /api/admin/users/{id}/toggle_active/ debe estar disponible. • El usuario objetivo debe existir en la base de datos. • El usuario que realiza el cambio no debe ser el mismo que el objetivo.
Pasos para seguir	1. Iniciar sesión como administrador. 2. Identificar el ID de un usuario diferente al actual. 3. Enviar una solicitud POST a: http://localhost:8000/api/admin/users/{id}/toggle_active/ 4. No se requiere cuerpo en la solicitud ({} vacío). 5. Verificar si el estado del usuario cambia de activo a inactivo o viceversa. 6. Confirmar que el servidor responde con código 200 OK.

Datos de Entrada	{
	"toggle_active": false
	}

Resultados Esperados	• Código HTTP: 200 OK • El campo activo del usuario cambia de true a false, o viceversa • Se muestra una notificación en pantalla (“Estado actualizado”)

	• El botón en la tabla cambia su estado visual (Activo/Inactivo)
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El usuario no puede entrar al sistema • El ID existe y no es el del usuario actual • El estado se actualiza correctamente • El backend responde con éxito (`200`) RECHAZO: • Se intenta cambiar el estado del propio usuario → acción bloqueada • No deja entrar a la cuenta • El usuario no está autenticado → 401 Unauthorized • No tiene permisos → 403 Forbidden • El ID es inválido → 404 Not Found • Error interno del servidor → 500 Internal Server Error
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 34: Prueba Funcional 22

3.2.14. Listar todas las proformas

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita obtener correctamente la lista de todas las proformas registradas a través del endpoint /api/proformas/. Se evalúa que los datos se retornen en formato JSON, con todos los campos requeridos por el frontend.	
Precondiciones	• El servidor backend debe estar corriendo correctamente. • El endpoint /api/proformas/ debe estar expuesto en el router. • Deben existir al menos una o más proformas registradas en la base de datos.
Pasos para seguir	1. Enviar una solicitud GET al endpoint /api/proformas/. 2. No se requiere autenticación adicional si es pública. 3. Esperar respuesta del backend. 4. Verificar que la respuesta contenga un arreglo de objetos con los campos correctos.
Datos de Entrada	• Método: GET • URL: /api/proformas/ • Headers: (opcional) Accept: application/json
Resultados Esperados	• Código de estado: 200 OK

	• Cuerpo: lista de objetos de proformas, por ejemplo: json [{ • "id": "...", • "proforma_number": "PRF-001", • "date": "...", ...}]
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • La respuesta devuelve un array JSON. • Cada objeto incluye los campos requeridos por el frontend. • La solicitud no falla si no hay registros (retorna array vacío). RECHAZO: • El servidor devuelve código 500, 404 o errores de conexión. • El formato no es JSON. • Faltan campos clave como id, proforma_number, date, etc.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 35: Prueba Funcional 23

3.2.15. Crear una nueva proforma

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita registrar una nueva proforma en la base de datos mediante el endpoint /api/proformas/. Se evalúa que los campos requeridos sean validados correctamente y que, al crearla, se genere automáticamente el PDF correspondiente.	
Precondiciones	• El backend debe estar ejecutándose correctamente. • El endpoint /api/proformas/ debe estar habilitado. • Debe existir al menos un cliente registrado para asociarlo a la proforma. • El campo created_by debe enviarse o asignarse por defecto.
Pasos para seguir	1. Enviar una solicitud POST al endpoint /api/proformas/. 2. Incluir en el cuerpo JSON todos los datos requeridos para la proforma: client_id, tipo_muestra, observaciones, etc. 3. El backend valida los datos y guarda la proforma. 4. Se genera automáticamente un PDF. 5. El backend responde con los datos de la proforma creada.
Datos de Entrada	• **Ejemplo válido:** json { "client": "64ebc123...", "tipo_muestra": "agua", "observaciones": "Proforma urgente", "created_by": "admin1"}

Resultados Esperados	**Proforma creada exitosamente:** • Código de estado: 201 Created • Respuesta JSON con los datos de la proforma creada. • Se genera un PDF en el backend y se actualiza el campo pdf_url. **Si faltan campos obligatorios:** • Código 400 Bad Request • Respuesta con detalle del error.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El cuerpo contiene todos los campos requeridos. • El PDF se genera correctamente (aunque sea en segundo plano). • El backend responde con 201 y los datos completos. RECHAZO: • Faltan campos como client, tipo_muestra • El cliente no existe. • El PDF no se puede generar y lanza error 500.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 36: Prueba Funcional 24

3.2.16. Agregar análisis a una proforma.

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita agregar un nuevo análisis a una proforma existente mediante el endpoint /api/proformas/<id>/add_analysis/. Se evalúa que los datos del análisis sean válidos, se asocien correctamente y que se actualice el PDF de la proforma.	
Precondiciones	• La proforma destino debe existir en la base de datos. • El backend debe estar corriendo y tener configurado el modelo Analysis y su serializer. • El endpoint /api/proformas/<id>/add_analysis/ debe estar disponible.
Pasos para seguir	1. Enviar una solicitud POST al endpoint /api/proformas/<id>/add_analysis/. 2. Incluir los campos del análisis en el cuerpo JSON: - parameter, unit, method, technique, Price 3. El backend guarda el análisis asociado a la proforma. 4. Se actualiza el PDF automáticamente. 5. Se devuelve la proforma actualizada con el análisis añadido.
Datos de Entrada	• URL: /api/proformas/64ebc1234567/add_analysis/ • Json { "parameter": "pH", "unit": "U pH",

	"method": "Potenciométrico", "technique": "Directa", "price": 25.00}
Resultados Esperados	**Operación exitosa:** • Código: 201 Created • Respuesta: JSON con los datos completos de la proforma actualizada. • El análisis aparece asociado en la proforma y el PDF es regenerado. **Errores posibles:** • Código 404 si la proforma no existe. • Código 400 si faltan campos del análisis.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • Todos los campos del análisis están presentes y son válidos. • La proforma existe. • El análisis se guarda correctamente y se refleja en el PDF. RECHAZO: • El ID de proforma es incorrecto o no existe. • Faltan campos como parameter, price, etc. • Error interno al guardar o generar el PDF.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 37: Prueba Funcional 25

3.2.17. Eliminar un análisis de una proforma

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita eliminar un análisis específico asociado a una proforma mediante el endpoint /api/proformas/<id>/remove_analysis/. Se valida que el análisis exista, esté vinculado a la proforma indicada, y que el PDF se actualice automáticamente tras la eliminación.	
Precondiciones	• La proforma debe existir y tener al menos un análisis registrado. • El análisis para eliminar debe estar relacionado con esa proforma. • El backend debe estar corriendo correctamente. • El endpoint /remove_analysis/ debe estar registrado y funcional.
Pasos para seguir	1. Enviar una solicitud DELETE al endpoint /api/proformas/<id>/remove_analysis/. 2. Incluir en el cuerpo JSON el campo analysis_id con el ID del análisis a eliminar. 3. El backend verifica que el análisis existe y pertenece a esa proforma.

	4. El análisis es eliminado y se regenera el PDF. 5. Se retorna la proforma actualizada.
Datos de Entrada	• URL: /api/proformas/64ebc1234567/remove_analysis/ • Json { "analysis_id": "66aaabbbccddeeff1234"}
Resultados Esperados	**Eliminación exitosa:** • La proforma y el análisis existen y están relacionados. • El análisis se elimina correctamente.El PDF es regenerado sin el análisis eliminado. **Errores esperados:** • Código 400 si falta el campo analysis_id. • Código 404 si el análisis no existe o no está relacionado con la proforma. • Código 404 si la proforma no existe.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • La proforma y el análisis existen y están relacionados. • El análisis se elimina correctamente. • El PDF se actualiza. RECHAZO: • Falta el analysis_id en el cuerpo de la solicitud. • El análisis no pertenece a la proforma. • El PDF no se puede regenerar.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 38: Prueba Funcional 26

3.2.18. Descargar PDF de la proforma

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita al usuario descargar el archivo PDF asociado a una proforma específica mediante el endpoint /api/proformas/<id>/pdf/. Se valida que el archivo exista físicamente y que se entregue como una descarga adjunta.	
Precondiciones	• La proforma debe existir en la base de datos. • El archivo PDF correspondiente (proforma.pdf_url) debe haberse generado y existir en el servidor. • El backend debe estar en funcionamiento.
Pasos para seguir	1. Enviar una solicitud GET al endpoint /api/proformas/<id>/pdf/. 2. El backend busca la proforma por ID. 3. Verifica que exista el archivo PDF en la ruta proforma.pdf_url. 4. Si existe, lo retorna como FileResponse descargable. 5. Si no existe, retorna un error apropiado.
Datos de Entrada	• **Método:** GET • **URL:** /api/proformas/64ebc1234567/pdf/

Resultados Esperados	**Descarga exitosa:** • Código de estado: 200 OK • Tipo de respuesta: application/pdf • Comportamiento: el navegador inicia la descarga del archivo con el nombre de la proforma. **Archivo no encontrado:** • Código 404 Not Found • Respuesta: {“error": "PDF no encontrado”} **Proforma inexistente:** • Código 404 Not Found • Respuesta: {“detail": "Proforma no encontrada”}
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • La proforma existe y el PDF ha sido generado previamente. • El archivo se entrega como descarga adjunta con el nombre correcto. RECHAZO: • La proforma no existe. • No se ha generado aún el PDF o la ruta pdf_url es inválida. • Error interno al abrir o servir el archivo.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 39: Prueba Funcional 27

3.2.19. Obtener datos para generar informe

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita obtener todos los datos necesarios para generar un informe técnico a partir de una proforma, usando el endpoint /api/proformas/<id>/informe/. La respuesta debe incluir los datos del cliente, proforma y análisis asociados.	
Precondiciones	• La proforma debe existir y estar correctamente asociada a un cliente y uno o más análisis. • El backend debe estar operativo. • El endpoint /api/proformas/<id>/informe/ debe estar registrado como acción.
Pasos para seguir	1. Enviar una solicitud GET al endpoint /api/proformas/<id>/informe/. 2. El backend busca la proforma por ID. 3. Obtiene los análisis asociados y los datos del cliente relacionado. 4. Devuelve un objeto JSON con todos los datos requeridos para el informe.
Datos de Entrada	• **Método:** GET • **URL:** /api/proformas/64ebc1234567/informe/

Resultados Esperados	**Respuesta exitosa:** • Código: 200 OK • Respuesta JSON: Json { "proforma_number": "PRF-0012", "date": "2025-07-21", "created_by": "admin1", "client_name": "Cliente S.A.", "client_ruc": "1234567890", "client_address": "...", "client_email": "...", "client_contact": "...", "analysis_data": [{"parameter": "pH", "unit": "U pH", "method": "Potenciométrico"}]} **Proforma no encontrada:** • Código: 404 Not Found • Respuesta: { "detail": "Proforma no encontrada" }
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • La proforma existe y tiene cliente y análisis vinculados. • El objeto JSON incluye todos los campos requeridos para armar el informe. RECHAZO: • La proforma no existe o tiene relaciones faltantes. • Error de consulta o datos incompletos.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 40: Prueba Funcional 28

3.2.20. Descargar informe PDF generado desde una proforma

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita descargar correctamente el archivo PDF del informe técnico generado a partir de una proforma, utilizando el endpoint /api/proformas/<id>/informe_pdf/. Se espera que el documento haya sido creado previamente en el servidor y que se entregue como descarga directa.	
Precondiciones	• La proforma debe existir en la base de datos. • Debe haberse generado previamente el PDF del informe técnico (función generate_informe_pdf). • El backend debe estar operativo y tener permisos para acceder al archivo generado.
Pasos para seguir	1. Enviar una solicitud GET al endpoint /api/proformas/<id>/informe_pdf/. 2. El backend busca la proforma y genera el informe PDF si es necesario.

	3. El archivo PDF es retornado como FileResponse con el nombre correcto. 4. El navegador o cliente debe iniciar la descarga.
Datos de Entrada	• **Método:** GET • **URL:** /api/proformas/64ebc1234567/informe_pdf/
Resultados Esperados	**Descarga exitosa:** • Código: 200 OK • Tipo de respuesta: application/pdf • Archivo adjunto con nombre basado en el número de proforma (ej. INF-0012.pdf). **Proforma no encontrada:** • Código: 404 Not Found • Respuesta: { "detail": "Proforma no encontrada" } **Error al generar PDF:** • Código: 500 Internal Server Error (si ocurre alguna excepción).
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • La proforma existe y contiene la información necesaria para generar el informe. • Se genera correctamente el archivo y se entrega como descarga PDF. RECHAZO: • La proforma no existe. • Error al ejecutar generate_informe_pdf. • No se puede abrir el archivo resultante.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 41: Prueba Funcional 29

3.2.21. Crear un nuevo informe técnico

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita crear correctamente un nuevo informe técnico asociado a una proforma mediante el endpoint /api/informes/. Se evalúa que los datos del cliente, análisis y resultados estén completos, y que el informe se almacene correctamente en la base de datos.	
Precondiciones	• El **backend** debe estar ejecutándose correctamente. • El **endpoint** /api/informes/ debe estar habilitado. • Debe existir al menos una proforma válida para asociar el informe. • El campo created_by puede enviarse o asignarse por defecto.
Pasos para seguir	1. Enviar una solicitud POST al endpoint /api/informes/. 2. Incluir en el cuerpo JSON todos los datos requeridos:

	• proforma_id, fecha_emision, analizado_por, tomado_por, procedimiento, resultados. 3. El backend valida los datos. 4. El informe es guardado en la base de datos. 5. Se devuelve una respuesta con los datos del informe creado.

Datos de Entrada	• **Ejemplo válido:**
	json {
	"proforma_id": "64ebc1234567",
	"fecha_emision": "2024-07-21",
	"analizado_por": "admin",
	"tomado_por": "Juan Pérez",
	"procedimiento": "Método EPA",
	"resultados": [
	{
	"parameter": "pH",
	"unit": "U pH",
	"method": "Electrométrico",
	"resultados": "7.5",
	"limite": "8.0",
	"incertidumbre": "±0.1"
	}
	]
	}

Resultados Esperados	**Informe creado exitosamente:** • Código de estado: 201 Created • Respuesta JSON con los datos del informe. • El informe queda almacenado en la base de datos. **Si faltan campos obligatorios:** • Código 400 Bad Request • Respuesta con detalle del error.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El cuerpo contiene todos los campos requeridos correctamente formateados. RECHAZO: • Código de estado: 400 Bad Request • Faltan campos como resultados, proforma_id o procedimiento. • El proforma_id no existe o es inválido
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 42: Prueba Funcional 30

3.2.22. Listar todos los informes técnicos

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita obtener correctamente la lista de todos los informes técnicos registrados en la base de datos mediante el endpoint /api/informes/. Se evalúa que los datos se retornen en formato JSON y que cada informe incluya los campos necesarios para ser visualizado en el frontend	
Precondiciones	• El **backend** debe estar corriendo correctamente. • El **endpoint** /api/informes/ debe estar expuesto y funcional. • Deben existir uno o más informes registrados en la base de datos.
Pasos para seguir	1. Enviar una solicitud GET al endpoint /api/informes/. 2. No es necesario incluir parámetros ni autenticación adicional. 3. Verificar que la respuesta contenga un arreglo JSON. 4. Confirmar que cada informe incluya campos como id, codigo, fecha_emision, status, etc.
Datos de Entrada	**NO APLICA**
Resultados Esperados	**Lista de informes:** • Código de estado: 200 OK • Respuesta JSON con los datos de cada informe de la lista. **Si no existen informes:** • Código de estado: 200 OK • Respuesta: [] (arreglo vacío)
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • El backend devuelve una lista JSON con al menos un informe técnico. • Cada objeto contiene los campos requeridos por el frontend. RECHAZO: • El backend devuelve error 500, 404 o formato incorrecto. • Faltan campos clave como codigo, fecha_emision, etc.
Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 43: Prueba Funcional 31

3.2.23. Obtener resultados asociados a un informe

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita obtener correctamente los resultados analíticos asociados a un informe técnico específico mediante el endpoint

/api/informes/<id>/resultados/. Se evalúa que se retornen solo los resultados relacionados al informe solicitado, con los campos completos.	
Precondiciones	• El **backend** debe estar en ejecución. • El **endpoint** /api/informes/<id>/resultados/ debe estar habilitado. • Debe existir al menos un informe válido con resultados previamente guardados. • El parámetro <id> debe corresponder a un informe existente.
Pasos para seguir	1. Identificar el ID de un informe existente. 2. Enviar una solicitud GET al endpoint /api/informes/<id>/resultados/. 3. Verificar que la respuesta sea un arreglo JSON con los resultados del informe. 4. Confirmar que cada resultado contiene los campos: parameter, unit, method, resultados, limite, incertidumbre.
Datos de Entrada	**NO APLICA**

Resultados Esperados	**Resultadod el informe:**
	• Código de estado: 200 OK
	• Respuesta JSON con los datos del resultado.
	[
	{
	"parameter": "pH",
	"unit": "U pH",
	"method": "Potenciométrico",
	"resultados": "7.5",
	"limite": "8.0",
	"incertidumbre": "±0.1"
	}
	]
	Si no existe el informe:
	• Código de estado: 404 Not Found
	• Mensaje: "Informe no encontrado

Criterios de Aceptación / Rechazo	**ACEPTACIÓN:** • El ID del informe es válido. • Se devuelve una lista con los resultados correctamente formateados del informe. **RECHAZO:** • El ID no existe o fue eliminado. • No hay resultados asociados. • El backend devuelve error 500 o 400.

Desarrollador Asignado	Michael Anthony Alejandro Muquis

Tabla 44: Prueba Funcional 32

3.2.24. Listar todos los análisis técnicos

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita obtener correctamente la lista de todos los análisis registrados mediante el endpoint /api/analysis/. Se evalúa que los datos se retornen en formato JSON y contengan los campos necesarios como parámetro, método, unidad, técnica y orden.	
Precondiciones	• **backend** debe estar en funcionamiento. • El endpoint /api/analysis/ debe estar disponible y accesible. • Deben existir registros de análisis en la base de datos previamente cargados.
Pasos para seguir	1. Enviar una solicitud GET al endpoint /api/analysis/. 2. Verificar que la respuesta sea un arreglo JSON. 3. Validar que cada objeto tenga los campos requeridos (parameter, unit, method, technique, order).
Datos de Entrada	**NO APLICA**
Resultados Esperados	**Resultad exitoso del analisis:** • Código de estado: 200 OK • Respuesta JSON con los datos del resultado. • **Si no existe el analisis:** • Código de estado: 404 Not Found • Mensaje: "Analisis no encontrado • Código: 200 OK • Respuesta: []
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • Se devuelve una lista de análisis correctamente estructurada. • Cada entrada contiene todos los campos esperados. **RECHAZO:** • La respuesta es vacía a pesar de que hay análisis cargados. • El backend responde con código 500 o datos malformados.
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 45: Prueba Funcional 33

3.2.25. Reordenar análisis técnicos en cada informe

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita actualizar el orden de varios análisis dentro de una proforma mediante el endpoint /api/analysis/reorder/. Se evalúa que se reciba un arreglo con los IDs y nuevos valores de orden, y que estos cambios se reflejen correctamente en la base de datos.	
Precondiciones	• **backend** debe estar en funcionamiento. • El endpoint /api/analysis/ debe estar disponible y accesible. • Deben existir registros de análisis en la base de datos previamente cargados.
Pasos para seguir	4. Enviar una solicitud GET al endpoint /api/analysis/. 5. Verificar que la respuesta sea un arreglo JSON. 6. Validar que cada objeto tenga los campos requeridos (parameter, unit, method, technique, order).
Datos de Entrada	**NO APLICA**
Resultados Esperados	**Resultad exitoso del analisis:** • Código de estado: 200 OK • Respuesta JSON con los datos del resultado. • [• { "_id": "685c1f47d50f25f30382f6fc", "proforma": "685c1f47d50f25f30382f6fb", "parameter": "aaa", "unit": "Presencia/Ausencia", "method": "aa", "technique": "aaaaa", "unit_price": 70, "quantity": 6, "subtotal": 420, "order": 0, "created_at": "2025-06-25T16:09:43.580+00:00" }] **Si no existe el analisis:** • Código de estado: 404 Not Found • Mensaje: "Analisis no encontrado • Código: 200 OK • Respuesta: []
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • Se devuelve una lista de análisis correctamente estructurada.

	• Cada análisis contiene los campos esperados y valores válidos **RECHAZO:** • La respuesta es vacía cuando deberían haber datos. • Faltan campos clave como unit_price, quantity, subtotal. • El backend devuelve error 500 o respuesta malformada.
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 46: Prueba Funcional 34

3.2.26. Reordenar análisis técnicos en cada informe

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita obtener correctamente la lista completa de clientes registrados en la base de datos mediante el endpoint /api/clients/. Se espera que los datos se devuelvan en formato JSON con todos los campos requeridos.	
Precondiciones	• El **backend** debe estar en ejecución correctamente. • El endpoint /api/clients/ debe estar expuesto en el router. • Deben existir uno o más clientes registrados en la base de datos.
Pasos para seguir	1. Enviar una solicitud GET al endpoint /api/clients/. 2. No se requiere autenticación adicional. 3. Verificar que la respuesta contenga un arreglo JSON con los datos de los clientes. 4. Confirmar que cada cliente tenga campos como id, name, ruc, address, email, contact_person.
Datos de Entrada	**NO APLICA**

Resultados Esperados	**Resultad exitoso del cleinte:**
	• Código de estado: 200 OK
	• Respuesta JSON con los datos del resultado.
	[
	{
	"id": "64eb12345678",
	"name": "Cliente S.A.",
	"ruc": "0999999999001",
	"address": "Av. Siempre Viva 123",
	"email": "contacto@cliente.com",
	"contact_person": "Juan Pérez"
	}
	]
	Si no existe algún cliente:

	• Código de estado: 404 Not Found • Mensaje: "Analisis no encontrado • Código: 200 OK • Respuesta: []
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • Se devuelve un arreglo JSON con los datos correctos de cada cliente. • El arreglo puede estar vacío, pero nunca debe fallar. **RECHAZO:** • Error de servidor (500). • El backend responde con formato incorrecto. • Faltan campos clave como name, ruc, etc.
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 47: Prueba Funcional 35

3.2.27. Reordenar análisis técnicos en cada informe

DESCRIPCIÓN DEL CASO DE PRUEBA Este caso verifica que el sistema permita registrar correctamente un nuevo cliente mediante el endpoint /api/clients/. Se evalúa que se validen todos los campos requeridos, se almacene el cliente en la base de datos y se devuelva su información en la respuesta.	
Precondiciones	• El **backend** debe estar funcionando correctamente. • El endpoint /api/clients/ debe estar accesible. • El sistema debe estar preparado para aceptar datos de nuevos clientes.
Pasos para seguir	1. Enviar una solicitud POST al endpoint /api/clients/. 2. Incluir en el cuerpo JSON los campos requeridos: • name, ruc, address, email, contact_person. 3. Verificar que el sistema procese correctamente los datos. 4. Confirmar que la respuesta incluya la información registrada.

Datos de Entrada	{
	"name": "Cliente S.A.",
	"ruc": "0999999999001",
	"address": "Av. Siempre Viva 123",
	"email": "contacto@cliente.com",
	"contact_person": "Juan Pérez"
	}

Resultados Esperados	**Cliente creado exitosamente:** • Código de estado: 201 Created • Respuesta JSON con los datos del cliente registrado. **Error de validación:** • Código de estado: 400 Bad Request • Respuesta con detalles del campo faltante o incorrecto.
Criterios de Aceptación / Rechazo	ACEPTACIÓN: • Todos los campos requeridos están presentes y bien formateados. • El sistema guarda los datos correctamente y responde con confirmación. **RECHAZO:** • Faltan uno o más campos obligatorios. • Algún campo tiene formato inválido (ej: email mal escrito). • El servidor devuelve error 400 o 500
Desarrollador Asignado	Alejandro Paul Barrionuevo Caiza

Tabla 48: Prueba Funcional 36

3.3. Despliegue de la aplicación

3.3.1. Explicación de las herramientas y la forma que se realizó el despliegue

El despliegue se realizó de manera local debido a las características operativas de la empresa donde se implementa EnvironabLab. Por esta razón, se optó por utilizar únicamente Docker Compose. La organización cuenta con recursos tecnológicos limitados, tanto a nivel de hardware como de conectividad, ya que no tiene la capacidad para mantener servicios en la nube de forma eficiente. Frente a este escenario, Docker Compose ofreció una solución simple y efectiva para levantar el sistema completo (frontend y backend) con una sola instrucción, sin necesidad de configurar entornos manuales ni instalar múltiples dependencias en cada equipo.

Además, el uso exclusivo de Docker Compose permite encapsular toda la lógica de despliegue dentro de un archivo único (docker-compose.yml), facilitando la instalación y el mantenimiento del sistema por parte del personal técnico local. Esto garantiza que todos los entornos (pruebas, producción o desarrollo) sean reproducibles, estables y consistentes, reduciendo los errores por diferencias de configuración. La decisión de usar solo Docker Compose responde también al principio de portabilidad: cualquier computadora con Docker instalado puede ejecutar el sistema sin modificaciones adicionales, lo que lo hace ideal para una empresa pequeña que busca autonomía tecnológica sin depender de servicios externos costosos o de infraestructura compleja.

Cabe aclarar que no se incluyó la base de datos dentro del `docker-compose.yml`, ya que se utilizó MongoDB Atlas, un servicio de base de datos en la

nube, lo cual permitió aprovechar una solución escalable y accesible sin requerir infraestructura local para la gestión de datos.

3.3.2 Requisitos para ejecutar el sistema EnvironabLab localmente

Para ejecutar correctamente el sistema EnvironabLab de forma local mediante Docker Compose, es necesario que el equipo cumpla con una serie de prerrequisitos básicos. A continuación, se detallan cada uno de ellos y los pasos necesarios para su instalación:

3.3.2.1 Requisitos

- Tener instalado Docker en la computadora que se va a ejecutar el sistema
- Instalar git para poder clonar el proyecto

3.3.2.2 Pasos para la ejecución del sistema

1.- Copiar el enlace del repositorio

https://github.com/Alee23432444444444444444444/tesis-integracion.git

2.- Crear una carpeta en escritorio

3.- Abrir Docker y presionar donde dice terminal y ejecutar el comando:

cd nombre de la carpeta

4.- Luego de llegar a la carpeta por medio del comando ejecutar el siguiente comando para clonar el proyecto:

git clone https://github.com/Alee23432444444444444444444/tesis-integracion.git

5.- Después de ejecutar el comando ejecutar el siguiente comando para entrar al proyecto:

cd tesis-integracion

6.- Luego de esto crear en la carpeta backend de la raíz un archivo y poner de nombre ". env" y copiar y pegar las variables de entorno que está aquí:

EMAIL_HOST=smtp.gmail.com

EMAIL_PORT=587

EMAIL_USE_TLS=True

EMAIL_HOST_USER=environovalab25@gmail.com

EMAIL_HOST_PASSWORD=enkt qrxz wzfk autr

DEFAULT_FROM_EMAIL=environovalab25@gmail.com

7.- Después de colocar las variables de entorno ir a la terminal y asegurar que se está en la ruta **\tesis-integracion\ y** tener abierto Docker**,** copiar y pegar este comando en la terminal:

docker compose up

8.- Por último, esperar unos minutos a que el backend y el fronend se despliegue en el puerto 8000 y en el 3000 se puede demorar unos dos minutos como máximo ya que hay muchas dependencias en el back y front luego que en la terminal salga algo así:

environab_frontend | https://cra.link/deployment

environab_frontend |

environab_frontend | INFO Accepting connections at http://localhost:3000

9.- Presionar la tecla control y hacer click en esta sección “http://localhost:3000”

otra opción es copiar lo que está en dentro de las comillas y pegar en un navegador y presionar la tecla enter y luego de realizar estos pasos podemos ver el sistema ENVIRONALAB

4. Conclusiones

El desarrollo de EnvironabLab permitió implementar un sistema informático completo capaz de gestionar proformas, análisis y clientes dentro de un laboratorio ambiental. A través de la arquitectura cliente-servidor, se logró una solución funcional, robusta y adaptada a las necesidades reales de la organización, mejorando significativamente los procesos manuales que antes se realizaban con herramientas dispersas o ineficientes.

Se utilizaron tecnologías actuales como Django, React, MongoDB Atlas y Docker, lo cual garantizó la creación de un sistema modular, escalable y fácilmente mantenible. En particular, el uso de contenedores y la estrategia de despliegue mediante Docker Compose facilitaron la puesta en marcha del sistema en entornos con recursos limitados, sin comprometer la funcionalidad ni la estabilidad.

Dada la infraestructura limitada de la organización, se tomó la decisión de realizar un despliegue local utilizando exclusivamente Docker Compose. Esta solución garantizó una instalación sencilla, sin requerir servidores en la nube o infraestructura externa, brindando a la empresa autonomía tecnológica y bajo costo de operación.

EnvironabLab representa un avance concreto en la digitalización de procesos dentro del laboratorio, reduciendo errores humanos, acelerando la generación de documentos técnicos (como proformas e informes) y centralizando la información en una única plataforma. Esto sienta las bases para futuras mejoras como el análisis estadístico de resultados o la automatización de notificaciones.

La utilización de MongoDB Atlas como base de datos externa permitió mantener los datos en una infraestructura segura, con respaldo en la nube y mecanismos de autenticación robustos. Esto proporcionó un nivel adicional de integridad y disponibilidad frente a posibles fallos locales, garantizando que la información crítica del laboratorio esté resguardada de manera profesional.

Gracias al uso de una API RESTful bien estructurada en el backend, EnvironabLab queda preparado para integrarse con otros módulos o plataformas externas en el futuro. Esto abre la posibilidad de conectar el sistema con herramientas de facturación, plataformas de monitoreo ambiental o sistemas institucionales, ampliando el alcance y utilidad del software desarrollado.

5. Recomendaciones

Para garantizar la estabilidad del sistema ante futuras actualizaciones, se aconseja la incorporación de pruebas automatizadas (unitarias y de integración) tanto en el backend como en el frontend. Esto facilitaría la detección temprana de errores y contribuiría a mantener la calidad del software.

Se recomienda crear manuales técnicos y de usuario que expliquen el despliegue, mantenimiento y uso del sistema EnvironabLab. Además, sería beneficioso organizar sesiones de capacitación para el personal responsable, asegurando la continuidad operativa aun cuando haya rotación o cambios en el equipo técnico.

Es fundamental realizar estudios previos de factibilidad técnica, económica y operativa antes de iniciar el desarrollo de cualquier sistema informático. Esto garantiza que los recursos disponibles, tanto humanos como materiales, se alineen con los objetivos del proyecto.

Al desarrollar soluciones tecnológicas, es recomendable prestar especial atención a la experiencia del usuario final. Interfaces intuitivas, mensajes de error claros y navegación sencilla ayudan a que el sistema sea adoptado de forma más eficiente, especialmente en contextos donde el personal no cuenta con formación técnica avanzada.

6. Referencias bibliográficas

React. (s.f.). *Documentación oficial*. Meta Platforms, Inc. Recuperado de https://es.react.dev/

MongoDB. (s.f.). *Manual oficial*. MongoDB, Inc. Recuperado de https://www.mongodb.com/es

Docker Inc. (s.f.). *Documentación de Docker Compose*. Recuperado de https://docs.docker.com/compose/

ReportLab. (s.f.). *Guía de usuario*. Recuperado de https://www.reportlab.com/docs/reportlab-userguide.pdf

Figma. (s.f.). *Guía de inicio*. Recuperado de https://help.figma.com/hc/es