



Instituto Tecnológico de Apizaco

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE APIZACO

**“DISEÑO Y DESARROLLO DEL BACKEND PARA UNA PLATAFORMA
WEB DE SEGUIMIENTO A PROCESOS DE GESTIÓN DE CALIDAD
DE PRODUCTOS ALIMENTICIOS, CON UNA ARQUITECTURA
BASADA EN LA NUBE”**

TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

**PARA OBTENER EL TÍTULO DE PROFESIONAL DE:
INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES**

PRESENTA:

JIMMY VAZQUEZ VASQUEZ

NOMBRE DEL ASESOR:

JUAN RAMOS RAMOS



APIZACO, TLAX. MAYO DE 2025.



Tzompantepec, Tlaxcala, **06/marzo /2025**

OFICIO No: SC/OPV/0071/25

ASUNTO: Liberación de proyecto
para titulación integral

JORGE ISRAEL SÁNCHEZ SÁNCHEZ
JEFA DEL DEPTO. DE DIVISIÓN DE ESTUDIOS PROFESIONALES
P R E S E N T E.

POR ESTE MEDIO LE INFORMO QUE HA SIDO LIBERADO EL SIGUIENTE PROYECTO PARA LA TITULACIÓN INTEGRAL

NOMBRE DEL EGRESADO:	VAZQUEZ VASQUEZ JIMMY
CARRERA:	INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES
NO. DE CONTROL:	19371082
NOMBRE DEL PROYECTO:	Diseño y desarrollo del backend para una plataforma web de seguimiento a procesos de gestión de calidad de productos alimenticios, con una arquitectura basada en la nube.
PRODUCTO U OPCIÓN:	TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

AGRADEZCO DE ANTEMANO SU VALIOSO APOYO EN ESTA IMPORTANTE ACTIVIDAD PARA LA FORMACIÓN PROFESIONAL DE NUESTROS EGRESADOS.

ATENTAMENTE

Excelencia en Educación Tecnológica®
Pensar para Servir, Servir para Triunfar®

MIQUELINA SÁNCHEZ PULIDO
JEFA DEL DEPTO. DE SISTEMAS Y COMPUTACIÓN

JUAN RAMOS RAMOS
ASESOR INTERNO

c.c.p. División de Estudios Profesionales
c.c.p. Depto. de Servicios Escolares
c.c.p. Archivo
MSP/mrol*





Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de Apizaco
División de Estudios Profesionales

Tzompantepec, Tlaxcala, 07/Abril/2025

VAZQUEZ VASQUEZ JIMMY.
EGRESADO DE LA CARRERA DE INGENIERÍA
EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES.
CON NÚMERO DE CONTROL 19371082.
P R E S E N T E.

Por este medio, me permito informar a usted que por aprobación de la comisión revisora asignada para valorar el trabajo que mediante la opción **TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL** y cuyo tema es: **DISEÑO Y DESARROLLO DEL BACKEND PARA UNA PLATAFORMA WEB DE SEGUIMIENTO A PROCESOS DE GESTIÓN DE CALIDAD DE PRODUCTOS ALIMENTICIOS, CON UNA ARQUITECTURA BASADA EN LA NUBE**, conforme a lo establecido en el procedimiento para la obtención del Título Profesional en el Instituto Tecnológico, la División de Estudios Profesionales a mi cargo le otorga la:

AUTORIZACIÓN DE IMPRESIÓN

Debiendo entregar **2 USB** del mismo, apropiadamente empastados, a la brevedad, para presentar su Acto de Recepción Profesional.

Sin otro particular por el momento me despido.

ATENTAMENTE

Excelencia en Educación Tecnológica®

Pensar para Servir, Servir para Triunfar®



SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
ISRAEL JORGE SANCHEZ SANCHEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES
DIVISIÓN DE ESTUDIOS PROFESIONALES

ccp. Archivo
ijss/Gsc*



2025
Año de
La Mujer
Indígena

Av. Instituto Tecnológico No. 418, San Andrés Ahuashuatepec,
Municipio de Tzompantepec, Tlaxcala, Mex. C.P. 90491 Tel. (241) 4172010
Ext. 304, e-mail: dep_titulacion@apizaco.tecnm.mx | apizaco.tecnm.mx



Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que me han acompañado y brindado su apoyo durante este proyecto de residencias. A mis padres, quienes con su amor, ejemplo y respaldo incondicional me han motivado a superar cada desafío en mi vida profesional y académica.

Extiendo mi gratitud al Instituto Tecnológico de Apizaco, especialmente a la Fábrica de Software, por brindarme la oportunidad de realizar este proyecto en un ambiente que fomenta el desarrollo tecnológico y la mejora continua en beneficio de la comunidad académica.

Agradezco profundamente a mi asesor, el profesor Juan Ramos Ramos, quien, desde su rol como asesor interno y externo, me ha guiado con su experiencia y conocimientos en cada etapa de este proyecto, siendo un apoyo fundamental en mi aprendizaje y desarrollo profesional.

También quiero reconocer a mis compañeros de equipo, quienes aportaron su entusiasmo y conocimientos, contribuyendo significativamente en cada fase de la implementación. Su colaboración fue vital para completar este proyecto de manera enriquecedora. Gracias a todos aquellos que confiaron en mí y me brindaron su apoyo para alcanzar esta meta.

Resumen

Este proyecto se desarrolló en el contexto de una residencia profesional en el Instituto Tecnológico de Apizaco, con el objetivo de optimizar el sistema de generación de reportes para una empresa del sector alimenticio. La razón principal de este desarrollo fue mejorar la eficiencia y precisión en la creación de reportes mensuales, además de implementar un apartado genérico adaptable para futuras necesidades de la organización.

El framework Django fue elegido como base del sistema, permitiendo el uso de consultas y APIs para el llenado automático de tablas y reportes en tiempo real. Para la generación de reportes en formato PDF, se empleó ReportLab, una biblioteca de Python que facilita la creación de documentos visualmente atractivos y profesionalmente estructurados. La integración de la plantilla comercial AdminLTE dotó al sistema de una interfaz moderna, equipada con alertas, confirmaciones y formularios en modales, mejorando la experiencia de usuario. Además, se utilizó la biblioteca SweetAlert para gestionar mensajes de confirmación y alertas visuales, proporcionando una interacción segura y atractiva.

Con esta combinación de herramientas y tecnologías, el proyecto resultó en una plataforma robusta y eficiente, alineada con las necesidades específicas de generación de reportes y el manejo de datos críticos en la organización.

Índice

Agradecimientos.....	1
Resumen	3
I. GENERALIDADES DEL PROYECTO	6
Introducción	6
Datos del instituto	6
Problemática.....	7
Objetivos.....	8
Objetivo general.....	8
Objetivos específicos	8
Justificación	9
II. MARCO TEÓRICO	10
Metodología de desarrollo ágil Sprint.....	10
¿Qué es Design Sprint?	10
Fases del Design Sprint	10
Beneficios de la Metodología en el Proyecto.....	11
Resultados Obtenidos	11
Herramientas y tecnologías.....	12
Framework y Lenguaje de Desarrollo	12
Herramientas y Librerías Complementarias	12
Entornos y Herramientas de Desarrollo.....	22
Enfoque API y Rendimiento	29
Beneficios de las Tecnologías Utilizadas	31
Generación de Reportes y su Importancia	32
Impacto del Proyecto	33
III. DESARROLLO	35
Análisis de Requerimientos	35
Diseño del Sistema	36
Flujo de Información para los Reportes	39
Funciones de Listado	41
Vistas Basadas en Clases	42
Manejo de Detalles de Calidad Microbiológica	43
Explicación Técnica de la Creación del Reporte PDF del Módulo de Calidad Microbiológica.....	46

Reporte Mensual: Descripción General	51
Explicación Técnica de la Vista del Reporte Mensual	56
Explicación Técnica de la API: ComposicionFisicoquimicaDataView	60
IV. RESULTADOS Y CONCLUSIONES	67
V. COMPETENCIAS DESARROLLADAS	73
VI. REFERENCIAS	75

I. GENERALIDADES DEL PROYECTO

Introducción

La transformación digital y la optimización de procesos en las instituciones académicas son factores clave para mejorar la eficiencia operativa y la toma de decisiones basada en datos. En este contexto, el Instituto Tecnológico de Apizaco, a través de su Fábrica de Software, impulsa el desarrollo de soluciones tecnológicas que atienden a las necesidades internas de automatización y mejora de flujo de trabajo en distintas áreas.

El presente proyecto de residencias profesionales se enfoca en el diseño e implementación de un sistema de software que permita optimizar los tiempos y la precisión en el tratamiento de datos recopilados dentro de la institución, así como la generación de reportes automáticos y personalizados. Este sistema surge con la intención de mejorar los procesos administrativos y académicos mediante una solución que integre funcionalidades de recolección, procesamiento y visualización de información.

Para llevar a cabo este desarrollo, se utilizaron metodologías ágiles, facilitando una respuesta rápida a los requerimientos específicos y adaptando el proyecto a los objetivos establecidos. El uso de herramientas colaborativas, como el sistema de control de versiones GitHub, permitió organizar el flujo de trabajo, gestionando de manera eficiente los cambios y avances en el desarrollo del sistema.

A lo largo del proyecto, se ha buscado crear un sistema robusto y escalable que permita automatizar tareas repetitivas y asegurar la calidad en el manejo de los datos, logrando así una herramienta que no solo optimice los tiempos de trabajo, sino que también aporte valor estratégico a la toma de decisiones en el Instituto Tecnológico de Apizaco.

Datos del instituto

Instituto Tecnológico de Apizaco

Es una institución de educación superior que ofrece diversas carreras a nivel licenciatura y programas de posgrado. Se destaca en áreas como ciencias sociales y administrativas.

MISIÓN:

Formar profesionistas que sean líderes visionarios, emprendedores, con capacidad de generar soluciones disruptivas, innovando en la ciencia y la tecnología, comprometidos con el desarrollo económico, responsabilidad social teniendo una visión sostenible y sustentable.

VISIÓN:

Ser una Institución de educación superior tecnológica de vanguardia, formadora de ciudadanos del mundo, con conocimientos intercultural, respetando la diversidad social.

VALORES

HONESTIDAD:

Se demuestra día a día a través de la transparencia con la ejecución de las actividades laborales cotidianas.

RESPONSABILIDAD:

Es la actitud que caracteriza al personal en el cumplimiento de su tarea con eficacia y eficiencia en beneficio de la comunidad.

LEALTAD:

Es la actitud que caracteriza al personal del Instituto Tecnológico de Apizaco, aplica sus conocimientos y habilidades en el logro de las metas institucionales.

Problemática

La administración y control de acceso en sistemas de información son esenciales para evitar problemas de seguridad y facilitar un flujo de trabajo eficiente, especialmente en empresas que gestionan grandes volúmenes de datos y requieren reportes constantes para la toma de decisiones, como en los sectores de alimentos

y bebidas. Empresas que dependen de sistemas robustos que no solo permitan la generación de reportes y gráficos con base en variables de producción, sino que también ofrezcan una administración segura y organizada a través de distintos roles de usuarios.

La falta de un sistema estructurado de gestión de permisos y acceso dificultaría en estos casos la asignación de roles específicos y el control de acceso a datos críticos. Esto genera ineficiencias en la administración de la información y puede exponer al sistema a riesgos de acceso no autorizado, afectando tanto la seguridad de los datos como la productividad de los usuarios.

Para resolver esta problemática, es necesario diseñar e implementar un módulo de gestión de usuarios que clasifique a los usuarios en grupos con diferentes niveles de permisos. De esta forma, cada usuario accede únicamente a las funciones y datos correspondientes a su rol. Este enfoque facilita la generación y consulta de reportes, proporcionando una estructura más segura y eficiente para el manejo de datos y el acceso a estadísticas clave.

Objetivos

Objetivo general

Desarrollar un submódulo de gestión de usuarios y permisos que optimice el control de acceso dentro de un sistema de generación de reportes basado en registros de análisis de laboratorio, asegurando una administración segura y organizada de la información según los roles específicos de cada usuario.

Objetivos específicos

Desarrollar un submódulo de gestión de usuarios y permisos que optimice el control de acceso dentro de un sistema de generación de reportes basado en registros de análisis de laboratorio, asegurando una administración segura y organizada de la información según los roles específicos de cada usuario.

- **Desarrollar una interfaz intuitiva para la gestión de usuarios:** Diseñar y construir una interfaz que permita al administrador crear, editar, eliminar y asignar roles a los usuarios de manera eficiente y amigable.

- **Asegurar la integridad y confidencialidad de los datos:** Aplicar buenas prácticas de seguridad para proteger la información sensible almacenada y transmitida en el sistema, utilizando técnicas como encriptación y autenticación de usuarios.
- **Integrar el submódulo con el sistema principal de generación de reportes:** Garantizar que el submódulo interactúe de manera eficiente con las demás funcionalidades del sistema, permitiendo un flujo de trabajo continuo y sin interrupciones.
- **Realizar pruebas exhaustivas de funcionalidad y seguridad:** Implementar un plan de pruebas para verificar que el sistema cumpla con los requisitos establecidos y que los accesos y permisos funcionen correctamente.
- **Proveer documentación técnica y de usuario final:** Generar la documentación necesaria para facilitar el mantenimiento del sistema y capacitar a los usuarios finales en la operación del submódulo.
- **Optimizar el rendimiento del submódulo:** Realizar ajustes para asegurar que el submódulo opere eficientemente con un tiempo de respuesta adecuado incluso bajo cargas significativas de usuarios y datos.

Justificación

La optimización de tiempos y procesos dentro del flujo de trabajo es un factor crítico para cualquier organización que busque mantener su competitividad en un mercado dinámico y globalizado. La implementación de módulos que gestionen y procesen información para la generación de reportes no solo mejora la eficiencia operativa, sino que también permite una toma de decisiones más informada y oportuna.

Este proyecto responde a la necesidad de contar con una herramienta tecnológica que automatice la recolección, tratamiento y presentación de datos, reduciendo significativamente los errores manuales y el tiempo requerido para estas tareas. Además, el uso de metodologías ágiles y herramientas de control de versiones asegura un desarrollo flexible, colaborativo y adaptado a las necesidades reales de los usuarios.

La solución propuesta beneficiará a la organización al mejorar sus procesos internos, estableciendo un modelo replicable para otras entidades con retos similares. Esto contribuye al fortalecimiento de la innovación tecnológica y a la mejora de la productividad en diferentes contextos.

II. MARCO TEÓRICO

Metodología de desarrollo ágil Sprint

El desarrollo de software ha evolucionado hacia enfoques más dinámicos y flexibles, como las metodologías ágiles, que permiten a los equipos adaptarse rápidamente a los cambios y enfocarse en la entrega continua de valor (Beck et al., 2001)¹. En este proyecto, la metodología ágil seleccionada fue **Design Sprint**, la cual se destaca por su capacidad de proporcionar soluciones rápidas y probadas en un corto periodo de tiempo (Knapp et al., 2016)².

¿Qué es Design Sprint?

Design Sprint es una metodología desarrollada por Google Ventures que se centra en resolver problemas complejos mediante el diseño, el prototipado y la validación con usuarios finales en tan solo cinco días (Knapp, Zeratsky, & Kowitz, 2016)². Se utiliza ampliamente para proyectos de software debido a su enfoque estructurado y enfocado en el usuario.

Fases del Design Sprint

El ciclo de Design Sprint se divide en cinco fases principales:

1. **Comprender:** Durante esta etapa, se analizó el problema a resolver en el sistema de reportes, identificando las necesidades clave de los usuarios y los objetivos del proyecto.

¹ Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). *Manifesto for Agile Software Development*. Recuperado de <https://agilemanifesto.org>

² Knapp, J., Zeratsky, J., & Kowitz, B. (2016). *Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days*. Simon & Schuster.

2. **Idear:** Se llevaron a cabo sesiones de lluvia de ideas y bocetado de posibles soluciones, buscando innovaciones que se alinearan con los requisitos funcionales y las expectativas del cliente.
3. **Decidir:** A partir de las ideas generadas, se seleccionaron las soluciones más viables mediante votaciones y análisis de impacto, asegurando que las decisiones estuvieran basadas en criterios técnicos y funcionales.
4. **Prototipar:** Se creó un prototipo funcional del sistema, destacando las funciones principales, como la generación de reportes automatizados.
5. **Validar:** Se realizaron pruebas con usuarios finales para evaluar la efectividad y usabilidad del prototipo, obteniendo retroalimentación que permitió refinar las soluciones implementadas.

Beneficios de la Metodología en el Proyecto

- **Flexibilidad:** La metodología permitió adaptarse a los cambios en los requerimientos y prioridades del proyecto.
- **Colaboración:** Facilitó la participación activa de los interesados, asegurando que las soluciones desarrolladas estuvieran alineadas con sus expectativas.
- **Rapidez en la entrega:** Al enfocarse en ciclos cortos de desarrollo, se lograron avances significativos en poco tiempo, reduciendo los riesgos asociados a proyectos más largos y tradicionales.

Resultados Obtenidos

Gracias al uso de Design Sprint, se consiguió un diseño funcional y validado del sistema de reportes en un tiempo eficiente, sentando las bases para el desarrollo e implementación del producto final. Este enfoque permitió minimizar los errores y garantizar que las funcionalidades diseñadas estuvieran centradas en las necesidades del usuario final.

Herramientas y tecnologías

El desarrollo del sistema de reportes se apoyó en herramientas y tecnologías específicas que garantizaron un enfoque eficiente, escalable y optimizado para la gestión de datos. Estas tecnologías se seleccionaron estratégicamente para cumplir con los objetivos funcionales y acelerar el proceso de desarrollo.

Framework y Lenguaje de Desarrollo

- **Django:** Django fue el framework base utilizado para desarrollar el backend del sistema. Su enfoque en la rapidez de desarrollo, junto con su diseño robusto y seguro, lo convirtió en una opción ideal para el proyecto (Holovaty & Kaplan-Moss, 2005)³.
 - **Django Rest Framework (DRF):** Se implementó DRF para desarrollar una API robusta que permitiera la comunicación entre el frontend y el backend. Este enfoque proporcionó una gestión eficiente de los datos y facilitó futuras integraciones con otros sistemas (Django Software Foundation, 2023).⁴
 - **Ventajas clave de DRF:** Simplificación en la creación de endpoints, manejo eficiente de serialización de datos y optimización de las consultas a la base de datos.

Herramientas y Librerías Complementarias

Django-Jazzmin: Esta librería se utilizó para personalizar y optimizar el módulo administrativo de Django, permitiendo un diseño más atractivo y funcional. Esto aceleró significativamente el proceso de desarrollo al proporcionar una interfaz moderna y lista para usar (Jazzmin Contributors, 2023).⁵

³ Holovaty, A., & Kaplan-Moss, J. (2005). *The Definitive Guide to Django*. Apress.

⁴ Django Software Foundation. (2023). *Django Project*. Recuperado de <https://www.djangoproject.com>

⁵ Jazzmin Contributors. (2023). *Jazzmin - Beautiful Admin Themes for Django*. Recuperado de <https://django-jazzmin.readthedocs.io>

Django-Widget-Tweaks: Facilitó la personalización de formularios en las vistas, lo que mejoró la experiencia del usuario al permitir una integración más flexible con el diseño del sistema (Django Software Foundation, 2023).⁶

xhtml2pdf: Permitió convertir vistas HTML a documentos PDF (xhtml2pdf Contributors, 2023).⁷

SQLite: Es un sistema de gestión de bases de datos relacional que no requiere un servidor para funcionar. Fue diseñado por D. Richard Hipp en el año 2000 con el objetivo de proporcionar un sistema de bases de datos simple, eficiente y fácilmente integrable en aplicaciones. SQLite es ampliamente utilizado en proyectos de software debido a su facilidad de uso y ligereza (Hipp, 2000).⁸

Características Principales de SQLite

1. Embebido y Ligero:

- SQLite se integra directamente en la aplicación que lo utiliza, eliminando la necesidad de configurar un servidor de bases de datos independiente.
- Su tamaño es pequeño, generalmente menor a 1 MB.

2. Sin Configuración:

- No requiere instalación ni configuración adicional para comenzar a utilizarlo.

3. Almacenamiento en un Solo Archivo:

- Toda la base de datos, incluidas las tablas, índices y datos, se almacena en un único archivo en el sistema de archivos.

⁶ Django Software Foundation. (2023). *Django Project*. Recuperado de <https://www.djangoproject.com>

⁷ xhtml2pdf Contributors. (2023). *xhtml2pdf Documentation*. Recuperado de <https://xhtml2pdf.readthedocs.io>

⁸ Hipp, D. R. (2000). *SQLite*. Recuperado de <https://sqlite.org>

4. **Compatibilidad con SQL Estándar:**

- Soporta un subconjunto importante de las funcionalidades del estándar SQL.

5. **Transacciones ACID:**

- Garantiza Atomicidad, Consistencia, Aislamiento y Durabilidad, lo que asegura la integridad de los datos incluso en caso de fallos.

6. **Rendimiento:**

- Diseñado para aplicaciones pequeñas a medianas con operaciones rápidas de lectura y escritura.

7. **Portabilidad:**

- Es multiplataforma y puede ejecutarse en sistemas operativos como Windows, Linux, macOS e incluso dispositivos móviles.



JavaScript: Es un lenguaje de programación dinámico, interpretado y orientado a objetos que se utiliza ampliamente para desarrollar aplicaciones web interactivas.

Fue creado por Brendan Eich en 1995 y es uno de los pilares del desarrollo web moderno, junto con HTML y CSS (Flanagan, 2020).⁹

Características Principales de JavaScript

1. Ejecución en el Lado del Cliente:

- JavaScript se ejecuta directamente en el navegador del usuario, lo que permite crear experiencias interactivas sin necesidad de interacción con el servidor.

2. Multiparadigma:

- Admite programación funcional, orientada a objetos y basada en eventos, lo que lo hace flexible para diversos casos de uso.

3. Interactividad:

- Permite manipular dinámicamente el DOM (Document Object Model), lo que posibilita crear elementos interactivos como formularios, animaciones y más.

4. Asincronía:

- Soporta promesas, callbacks y async/await para manejar operaciones asíncronas, como llamadas a APIs o temporizadores.

5. Ecosistema Extenso:

- Cuenta con bibliotecas y frameworks populares como React, Angular y Vue.js, que amplían sus capacidades y simplifican el desarrollo.

6. Compatibilidad Universal:

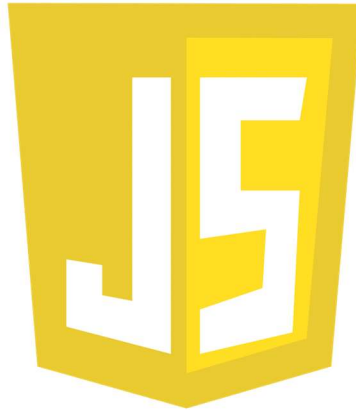
- Es compatible con todos los navegadores modernos y funciona en cualquier dispositivo con acceso a la web.

⁹ Flanagan, D. (2020). *JavaScript: The Definitive Guide*. O'Reilly Media.

7. Node.js para Backend:

- JavaScript también se puede usar en el lado del servidor gracias a Node.js, ampliando su alcance más allá del frontend.

JavaScript



AdminLTE: Es una plantilla de panel administrativo de código abierto basada en Bootstrap 4 y más reciente Bootstrap 5. Es ampliamente utilizada en aplicaciones web para construir interfaces de administración modernas, responsivas y funcionales. Su popularidad se debe a su diseño limpio, personalizable y a la integración de componentes y plugins listos para usar (Almsaeed Studio, n.d.).¹⁰

Características Principales de AdminLTE

1. Diseño Responsivo:

- Ofrece compatibilidad con dispositivos móviles y pantallas de diferentes tamaños gracias al uso de Bootstrap.

2. Componentes Preconfigurados:

- Incluye widgets, gráficos, tablas, formularios, menús desplegables y más, que permiten una implementación rápida.

¹⁰ Almsaeed Studio. (n.d.). *AdminLTE*. Recuperado de <https://adminlte.io>

3. **Compatibilidad con Plugins:**

- Soporta una amplia gama de plugins populares, como **DataTables**, **Chart.js**, y **Select2**, para extender la funcionalidad.

4. **Personalización:**

- Ofrece un sistema de temas para cambiar colores, fuentes y estilos fácilmente.

5. **Paneles de Control Dinámicos:**

- Proporciona plantillas de dashboard listas para personalizar y mostrar métricas clave, gráficos y tablas.

6. **Soporte Multilenguaje:**

- Se puede integrar con sistemas multilenguaje para aplicaciones globales.

7. **Integración con Frameworks Backend:**

- Funciona perfectamente con Django, Laravel, Flask, entre otros, para crear sistemas de administración robustos.

Uso de AdminLTE en el Proyecto

En este proyecto, **AdminLTE** fue utilizado como base para construir la interfaz administrativa del sistema de reportes. Algunas de sus aplicaciones específicas incluyen:

1. Panel de Control

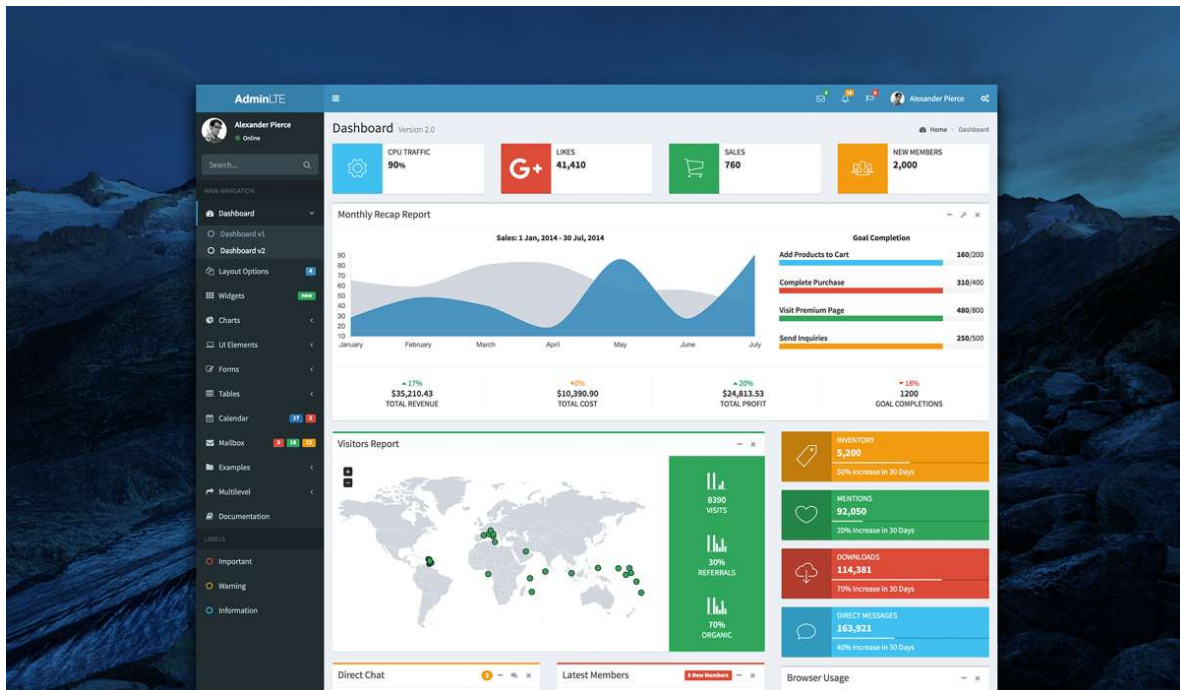
Se personalizó el dashboard de AdminLTE para mostrar métricas clave y datos relacionados con el sistema, como:

- Indicadores de calidad.

- Resúmenes mensuales de datos capturados.
- Notificaciones de actividades recientes.

2. Formularios Personalizados

AdminLTE simplificó la creación de formularios responsivos para capturar datos microbiológicos, fisicoquímicos y otros relacionados con los reportes.



ReportLab: Es una biblioteca de Python ampliamente utilizada para la creación de documentos PDF. Fue desarrollada por ReportLab Inc. y es conocida por su flexibilidad y potencia al momento de generar documentos dinámicos y personalizados. ReportLab es especialmente útil en proyectos que requieren la automatización de reportes, como facturas, análisis de datos o gráficos, debido a su capacidad para trabajar directamente con contenido dinámico y estructurado (ReportLab, n.d.).¹¹

Características Principales de ReportLab

¹¹ ReportLab. (n.d.). *The ReportLab Toolkit Documentation*. Recuperado de <https://www.reportlab.com>

1. **Generación de PDFs Dinámicos:** Permite la creación de documentos PDF de manera programática, utilizando datos generados en tiempo real.
2. **Control Preciso del Diseño:** Proporciona un control total sobre el diseño de los documentos, desde la disposición de texto y gráficos hasta la incorporación de imágenes.
3. **Soporte de Gráficos y Tablas:** Incluye herramientas para dibujar gráficos, tablas y diagramas directamente en los PDFs.
4. **Capacidad Multilenguaje:** Soporta caracteres especiales y lenguajes con diferentes codificaciones, como UTF-8, lo que permite incluir texto en varios idiomas.
5. **Compatibilidad:** Funciona perfectamente con otras bibliotecas de Python como **Pandas**, **NumPy** y **matplotlib**, facilitando la integración de datos procesados y gráficos en los reportes.

Ventajas de Usar ReportLab

1. **Flexibilidad:** Permite un diseño detallado y personalizado de los reportes.
2. **Automatización:** Genera reportes automáticamente a partir de datos en bases de datos o APIs, reduciendo errores manuales.
3. **Personalización Total:** Cada aspecto del PDF puede ser configurado, desde colores y estilos hasta contenido dinámico.
4. **Compatibilidad con SVG:** Facilita la inclusión de gráficos vectoriales, ideal para reportes con visualizaciones complejas.
5. **Rendimiento:** Es eficiente en la generación de PDFs incluso para documentos con grandes cantidades de datos.

Limitaciones de ReportLab

- **Curva de Aprendizaje:** Puede ser complejo al principio debido al nivel de personalización que ofrece.
- **Diseño Manual:** Requiere especificar manualmente las posiciones y estilos, lo que puede ser tedioso para diseños complejos.

svglib: Facilitó la incorporación de gráficos SVG en los reportes generados (Schumann, 2018).¹²

Django-Rest-Framework: Django Rest Framework (DRF) es una poderosa biblioteca basada en Django que facilita la creación de APIs RESTful robustas y escalables. Es ampliamente utilizada en proyectos que requieren la comunicación entre sistemas o la exposición de datos para consumo en aplicaciones frontend o servicios de terceros (Tom Christie, n.d.).¹³

Características Principales de Django Rest Framework

1. Serialización de Datos:

- DRF permite transformar modelos y consultas de Django en formatos JSON o XML, listos para ser consumidos por clientes externos.
- Simplifica el proceso de conversión entre datos complejos (como objetos) y representaciones más simples.

2. Sistema de Autenticación Integrado:

- Ofrece soporte para autenticación básica, autenticación por tokens y OAuth2, lo que facilita la gestión de usuarios y permisos en las APIs.

3. Clases Genéricas:

¹² Schumann, D. (2018). *SVG Python Package*. Recuperado de <https://pypi.org/project/svglib/>

¹³ Tom Christie. (n.d.). *Django Rest Framework Documentation*. Recuperado de <https://www.django-rest-framework.org>

- Proporciona vistas basadas en clases (CBV) genéricas para operaciones CRUD comunes, como ListAPIView, RetrieveAPIView, CreateAPIView, etc.

4. **Soporte para Filtrado y Paginación:**

- Permite configurar filtros avanzados y paginar automáticamente los resultados para manejar grandes volúmenes de datos.

5. **Documentación Automática:**

- DRF incluye herramientas para generar documentación interactiva de las APIs, como la interfaz de navegador API y soporte para herramientas como Swagger y Redoc.

6. **Extensibilidad:**

- DRF es altamente personalizable, lo que permite adaptarlo a casos de uso específicos mediante la sobrescritura de serializadores, vistas o permisos.

Ventajas de Usar Django Rest Framework

1. **Rapidez de Desarrollo:** DRF reduce el tiempo necesario para crear APIs al ofrecer herramientas predefinidas y configuraciones genéricas.
2. **Integración Perfecta con Django:** Se integra fácilmente con modelos y vistas de Django, reutilizando configuraciones existentes.
3. **Flexibilidad:** Permite personalizar las APIs según necesidades específicas, como validaciones avanzadas o control de permisos.
4. **Seguridad:** Su sistema de autenticación robusto garantiza que las APIs estén protegidas contra accesos no autorizados.

5. **Escalabilidad:** DRF es adecuado tanto para pequeños proyectos como para sistemas grandes con múltiples endpoints y usuarios.



Entornos y Herramientas de Desarrollo

Python: Un Lenguaje Versátil para el Desarrollo de Software

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general, diseñado para ser simple y legible. Creado por Guido van Rossum en 1991, Python se ha convertido en uno de los lenguajes más utilizados en la industria debido a su flexibilidad, facilidad de aprendizaje y robusto ecosistema de bibliotecas y frameworks (Van Rossum & Drake, 2009).¹⁴

Características Clave de Python

1. **Sintaxis Clara y Legible:** Python utiliza una estructura basada en indentación, lo que lo hace más fácil de leer y escribir, incluso para principiantes.

¹⁴ Van Rossum, G., & Drake, F. L. (2009). *The Python Language Reference Manual*. Network Theory Ltd.

2. **Multiparadigma:** Soporta programación orientada a objetos, funcional y procedural.
3. **Bibliotecas y Frameworks:** Ofrece una amplia variedad de bibliotecas para diferentes áreas, como desarrollo web (Django, Flask), análisis de datos (Pandas, NumPy), aprendizaje automático (TensorFlow, PyTorch), entre otros.
4. **Portabilidad:** Es multiplataforma, lo que permite ejecutar código Python en diferentes sistemas operativos sin necesidad de modificaciones.
5. **Ecosistema Activo:** Python tiene una comunidad global de desarrolladores que constantemente contribuyen con bibliotecas, herramientas y soporte técnico.
6. **Integración:** Puede integrarse fácilmente con otros lenguajes y tecnologías, como C, C++, Java, y bases de datos.

Ventajas de Usar Python en el Proyecto

1. Desarrollo Rápido:

- La sintaxis sencilla de Python y las herramientas proporcionadas por Django permitieron reducir significativamente los tiempos de desarrollo.

2. Escalabilidad:

- La arquitectura desarrollada con Python es fácilmente escalable para soportar futuras funcionalidades, como la integración de gráficos avanzados o nuevas API.

3. Mantenimiento Fácil:

- Gracias a la claridad del código, es más fácil realizar mejoras o solucionar errores en el sistema.

4. Integración con Herramientas de Terceros:

- Python permitió una fácil integración con tecnologías como bases de datos relacionales (SQLite) y herramientas de control de versiones (Git y GitHub).

Python 3.11: Se utilizó esta versión de Python como base para todo el desarrollo del sistema, destacándose por su rendimiento mejorado y soporte para las librerías necesarias.



Visual Studio Code: Visual Studio Code es un editor de código fuente desarrollado por **Microsoft**, diseñado para ser ligero, rápido y extensible, convirtiéndose en una de las herramientas más populares entre desarrolladores de software. Su propósito principal es proporcionar un entorno eficiente y personalizable para escribir y depurar código en una amplia variedad de lenguajes de programación (Microsoft, 2015).¹⁵

Características Principales de VS Code

1. **Compatibilidad con múltiples lenguajes:** Admite lenguajes como Python, JavaScript, C++, Java, entre otros.

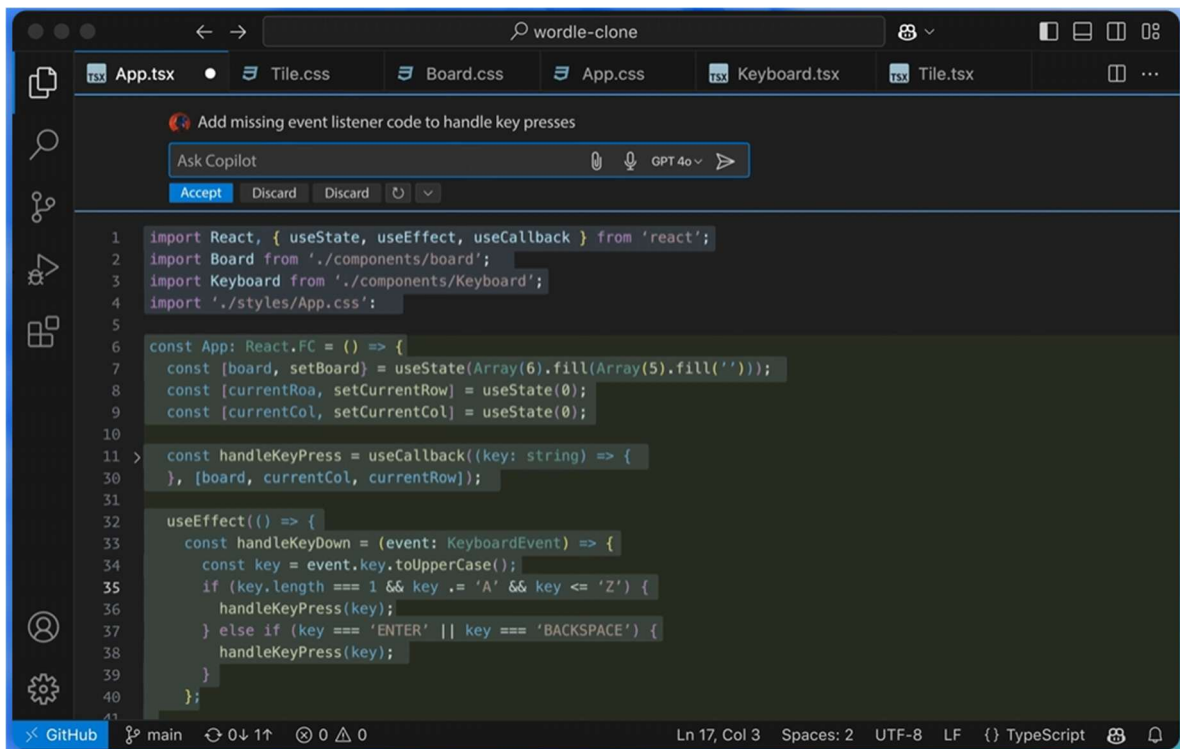
¹⁵ Microsoft. (2015). *Visual Studio Code Documentation*. Recuperado de <https://code.visualstudio.com/>

2. **Extensiones:** Su marketplace permite integrar funcionalidades adicionales, como depuradores, linters y herramientas de desarrollo específicas para frameworks (por ejemplo, Django y Flask).
3. **Terminal integrada:** Facilita la ejecución de comandos y la interacción con el entorno del proyecto sin salir del editor.
4. **Control de versiones integrado:** Incluye soporte para Git, lo que permite realizar commits, ver cambios y resolver conflictos directamente desde la interfaz.
5. **Depuración:** Ofrece herramientas avanzadas para depurar aplicaciones en tiempo real.
6. **Personalización:** Interfaz y configuración altamente personalizables, incluyendo temas, atajos de teclado y configuraciones de espacio de trabajo.

Uso en el Proyecto

En este proyecto, Visual Studio Code fue la herramienta principal para:

- **Escribir y gestionar el código fuente:** Edición de archivos Python para el desarrollo backend utilizando el framework Django.
- **Depuración:** Depuración en tiempo real de las APIs creadas con Django Rest Framework.
- **Integración con Git:** Administración eficiente del control de versiones en GitHub, permitiendo la colaboración y el seguimiento de cambios.
- **Entorno de desarrollo optimizado:** Integración de extensiones como *Python* y *Django Snippets*, que ayudaron a acelerar el desarrollo del sistema.



Git y GitHub: ¿Qué es Git?

Git es un sistema de control de versiones distribuido diseñado para rastrear los cambios en el código fuente durante el desarrollo de software. Fue creado por Linus Torvalds en 2005 para gestionar el desarrollo del kernel de Linux. Actualmente, es una de las herramientas más utilizadas por desarrolladores y equipos de software debido a su velocidad, eficiencia y capacidad de trabajar de manera colaborativa (Chacon & Straub, 2014).¹⁶

Características Clave de Git

1. **Control de versiones distribuido:** Cada desarrollador tiene una copia completa del repositorio, lo que permite trabajar sin necesidad de estar siempre conectado a un servidor central.

¹⁶ Chacon, S., & Straub, B. (2014). *Pro Git*. Apress. Recuperado de <https://git-scm.com/book/en/v2>

2. **Seguimiento de cambios:** Mantiene un historial detallado de cada modificación en el código, lo que facilita revertir cambios o identificar errores introducidos en el proyecto.
3. **Ramas (branches):** Permite crear ramas independientes para trabajar en nuevas funcionalidades o correcciones sin afectar el código principal.
4. **Fusión (merge):** Facilita la integración de cambios desde diferentes ramas, resolviendo conflictos automáticamente cuando es posible.
5. **Flexibilidad:** Compatible con múltiples flujos de trabajo y entornos, desde proyectos personales hasta grandes equipos de desarrollo.

¿Qué es GitHub?

GitHub es una plataforma de alojamiento para proyectos que utiliza Git como su base. Fundada en 2008 y adquirida por Microsoft en 2018, GitHub proporciona herramientas adicionales para facilitar la colaboración y gestión de proyectos de software. Es especialmente popular entre desarrolladores, tanto para proyectos individuales como para equipos (Loeliger & McCullough, 2012).¹⁷

Características Clave de GitHub

1. **Repositorios remotos:** Permite almacenar repositorios de Git en la nube, facilitando el acceso y la colaboración entre desarrolladores desde cualquier ubicación.
2. **Colaboración:** Herramientas como *pull requests*, *issues* y *project boards* permiten a los equipos coordinar y rastrear el progreso del proyecto.
3. **Integración y despliegue continuo (CI/CD):** Facilita la automatización de pruebas, integraciones y despliegues mediante GitHub Actions.

¹⁷ Loeliger, J., & McCullough, M. (2012). *Version Control with Git: Powerful tools and techniques for collaborative software development*. O'Reilly Media.

4. **Gestión de ramas y cambios:** Proporciona una interfaz intuitiva para manejar ramas, revisar cambios y resolver conflictos de manera visual.
5. **Documentación y Wiki:** Incluye opciones para documentar proyectos con archivos README y wikis asociados.

Uso en el Proyecto

En este proyecto, **Git** y **GitHub** se utilizaron como herramientas esenciales para gestionar el ciclo de vida del desarrollo del sistema. Sus aplicaciones específicas incluyeron:

1. Control de Versiones (Git):

- Rastreo de cambios en el código fuente para garantizar un historial claro y organizado.
- Implementación de ramas para trabajar en características específicas como generación de reportes o manejo de datos microbiológicos.

2. Colaboración y Seguimiento (GitHub):

- Almacenamiento del repositorio del proyecto en GitHub, permitiendo acceso remoto y colaboración con otros miembros del equipo.
- Uso de *issues* para documentar errores y realizar un seguimiento de las tareas pendientes.
- Revisión y fusión de *pull requests* para integrar nuevas características de manera ordenada.

3. Automatización:

- Configuración de acciones de GitHub (*GitHub Actions*) para automatizar pruebas y verificar la calidad del código antes de integrarlo al repositorio principal.

Beneficios de Usar Git y GitHub

1. **Organización:** Un historial detallado facilita la identificación y corrección de errores.
2. **Colaboración:** Equipos distribuidos pueden trabajar en paralelo, minimizando conflictos gracias al sistema de ramas.
3. **Seguridad:** Los repositorios en GitHub pueden ser públicos o privados, garantizando el acceso controlado al código.
4. **Escalabilidad:** Adecuado tanto para proyectos pequeños como para desarrollos complejos en grandes equipos.



Enfoque API y Rendimiento

El enfoque API proporcionado por Django Rest Framework permitió:

- **Eficiencia en la gestión de datos:** Las APIs desarrolladas manejaron grandes volúmenes de datos con una estructura clara y accesible.
- **Escalabilidad:** La implementación de endpoints facilita futuras expansiones del sistema y su integración con otros servicios.

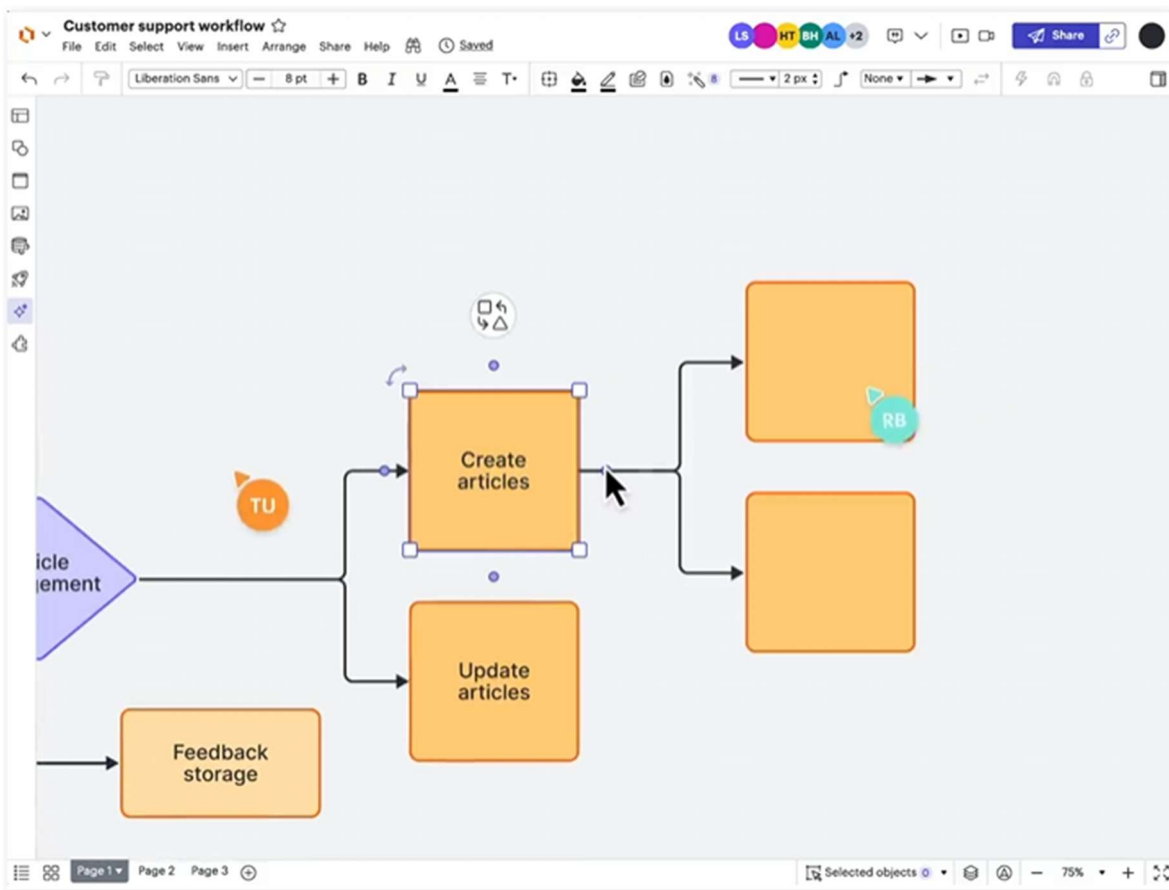
- **Reducción de carga en el servidor:** Gracias al manejo optimizado de datos en JSON, se lograron tiempos de respuesta rápidos y un rendimiento general mejorado.

Herramientas Relacionadas

Lucidchart

- **Función:** Plataforma para la creación de diagramas y documentación visual.
- **Contribución:** Se utilizó para generar diagramas de flujo, diagramas de secuencia y otros elementos visuales clave para planificar y documentar el diseño y la implementación del sistema.
 - **Impacto:** Facilitó la comunicación entre los involucrados en el proyecto al representar visualmente procesos y estructuras, lo que ayudó a identificar áreas de mejora y garantizar un diseño claro y comprensible (Lucid Software, Inc., 2023).¹⁸

¹⁸ Lucid Software, Inc. (2023). *Lucidchart: Visual productivity platform*. Retrieved from <https://www.lucidchart.com>



Beneficios de las Tecnologías Utilizadas

- **Aceleración en el desarrollo:** El uso de herramientas como Jazzmin redujo significativamente el tiempo requerido para configurar y personalizar la interfaz administrativa.
- **Generación automatizada de reportes:** Las librerías PDF como xhtml2pdf y ReportLab aseguraron una creación ágil y personalizada de documentos, eliminando procesos manuales.
- **Seguridad y mantenimiento:** Django ofreció una arquitectura segura y sencilla de mantener, asegurando la protección de los datos manejados.

Generación de Reportes y su Importancia

La generación de reportes es un componente crítico para cualquier organización, ya que facilita la presentación de datos procesados de manera estructurada, permitiendo una mejor comprensión y análisis de la información. En este proyecto, la implementación de un sistema automatizado de generación de reportes fue fundamental para optimizar la eficiencia operativa y reducir la dependencia de procesos manuales.

Automatización de Reportes

La automatización de reportes asegura que los datos recopilados sean transformados en información útil de manera ágil, precisa y accesible. Este enfoque reduce significativamente los errores humanos asociados con la manipulación manual de datos y acelera los tiempos de respuesta ante consultas y análisis.

- **Procesamiento de datos:** Los reportes se generan a partir de datos estructurados, organizados y procesados de forma automática.
- **Formatos personalizados:** Se incorporaron herramientas como xhtml2pdf, ReportLab y svglib para crear documentos PDF con un diseño adaptable a las necesidades específicas de los usuarios.

Características Clave de los Reportes Automatizados

1. **Accesibilidad:** Los reportes son generados en tiempo real y pueden ser distribuidos a través de múltiples plataformas, facilitando su disponibilidad para los usuarios.
2. **Precisión:** La automatización elimina inconsistencias en el manejo de datos, garantizando que la información presentada sea confiable y exacta.
3. **Personalización:** Los reportes pueden adaptarse a las necesidades específicas de cada usuario o departamento, asegurando que solo se presente información relevante.

Beneficios para la Organización

1. **Eficiencia Operativa:** El tiempo necesario para generar reportes se reduce drásticamente, permitiendo a los empleados concentrarse en tareas más estratégicas.
2. **Mayor Confiabilidad de los Datos:** La automatización asegura que los datos se procesen de manera uniforme, reduciendo errores y aumentando la confianza en la información presentada.
3. **Soporte a la Toma de Decisiones:** La disponibilidad de reportes precisos y bien estructurados proporciona a los tomadores de decisiones una base sólida para evaluar situaciones y planificar acciones.

Impacto del Proyecto

La implementación del sistema automatizado de generación de reportes basado en registros de análisis de laboratorio tuvo un impacto significativo tanto en los procesos internos de la organización como en su capacidad para tomar decisiones estratégicas. Este impacto se refleja en diversas áreas clave, desde la eficiencia operativa hasta la mejora en la comunicación de datos relevantes.

Optimización de Procesos

La automatización de la generación de reportes permitió transformar procesos manuales y repetitivos en flujos de trabajo más eficientes, reduciendo errores humanos y tiempos de ejecución. Esto resultó en:

- **Mayor productividad:** Los usuarios ahora pueden acceder a reportes detallados y organizados con solo unos clics, sin necesidad de manipulación manual de datos.

- **Reducción de tiempos:** El tiempo necesario para compilar, procesar y entregar información se redujo drásticamente, mejorando la capacidad de respuesta de la organización.

Mejora en la Toma de Decisiones

El acceso a reportes precisos y personalizados brindó a los responsables de la toma de decisiones una base confiable para evaluar escenarios y planificar acciones. Los datos claros y estructurados facilitaron:

- **Análisis rápido y efectivo:** La información presentada en los reportes permitió detectar patrones y tendencias relevantes de manera inmediata.
- **Planificación estratégica:** La confiabilidad de los datos facilitó el desarrollo de estrategias basadas en información concreta.

Incremento en la Transparencia y Comunicación

La claridad y disponibilidad de los reportes contribuyeron a una comunicación más transparente entre las diferentes áreas de la organización. Esto mejoró la colaboración y alineación hacia objetivos comunes.

Beneficio Tecnológico

El uso de herramientas modernas como Django, Django Rest Framework, y librerías especializadas para la creación de reportes (xhtml2pdf, ReportLab, svglib) sentó las bases para un sistema escalable y adaptable. Este beneficio incluye:

- **Sostenibilidad del sistema:** La infraestructura implementada permite futuras actualizaciones y expansiones con facilidad.
- **Compatibilidad:** La solución puede integrarse con otros sistemas o tecnologías, garantizando su relevancia a largo plazo.

Aporte a la Innovación

El proyecto promovió una cultura de innovación al introducir prácticas tecnológicas avanzadas y automatización en los procesos organizativos. Esto no solo mejoró los resultados inmediatos, sino que también estableció un precedente para la implementación de soluciones tecnológicas similares en otras áreas.

Impacto en los Usuarios

Finalmente, el sistema mejoró significativamente la experiencia de los usuarios al ofrecer una interfaz intuitiva y accesible para la generación de reportes. Esto redujo la curva de aprendizaje y aumentó la satisfacción de los usuarios finales.

III. DESARROLLO

El desarrollo del sistema de generación de reportes se llevó a cabo siguiendo un enfoque iterativo, basado en la metodología ágil Design Sprint. Esto permitió dividir el proyecto en fases específicas, asegurando un progreso constante y enfocado en los objetivos establecidos. A continuación, se describe cada etapa del desarrollo.

Análisis de Requerimientos

El primer paso consistió en identificar y documentar los requerimientos funcionales y no funcionales del sistema, basándome en las necesidades específicas de los usuarios finales y las metas del proyecto.

- **Requerimientos funcionales:**

- Implementar una API que gestione de manera eficiente la consulta y procesamiento de datos para los reportes.
- Crear reportes automatizados en formato PDF, personalizables según los parámetros definidos.

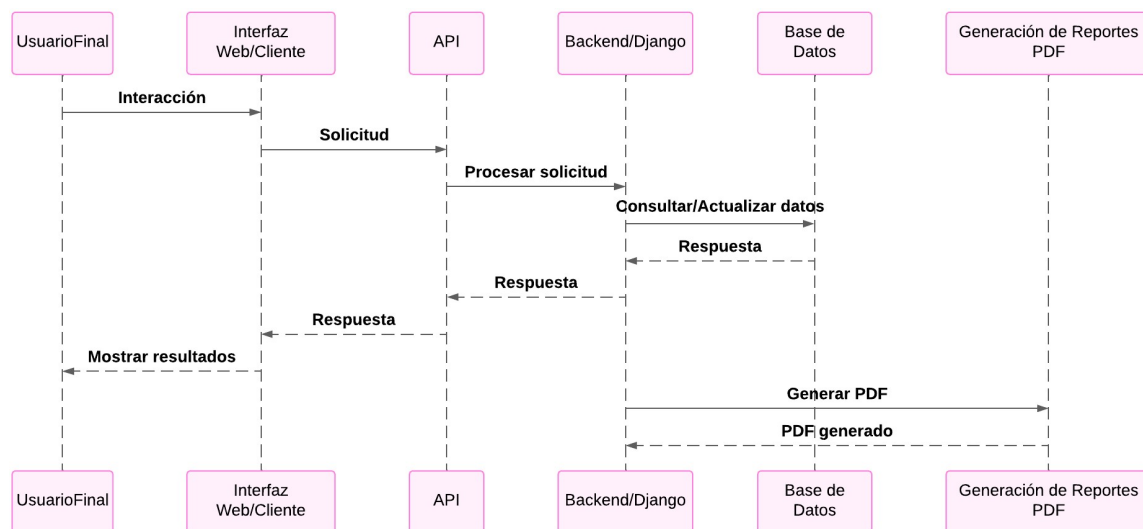
- **Requerimientos no funcionales:**

- Garantizar la seguridad de los datos manejados mediante autenticación y permisos de acceso.
- Optimizar el tiempo de respuesta del sistema incluso bajo altas cargas de datos.

Diseño del Sistema

Se utilizó **Lucidchart** para crear diagramas de flujo y diagramas de secuencia que representaran la interacción entre los componentes del sistema:

- **Diagrama de secuencia:** Muestra cómo interactúan los módulos backend con los endpoints API para procesar y devolver los datos solicitados.



El sistema de generación de reportes toma como base la información contenida en las tablas clave del esquema, permitiendo compilar datos estructurados y procesados de manera automatizada. Los reportes generados están diseñados para ofrecer una visión clara y detallada de la información relacionada con los procesos de recolección, evaluación y análisis de datos.

Tablas Relacionadas con el reporte de Calidad microbiológica

El diagrama muestra las principales tablas relacionadas con la generación de reportes:

1. catalogos_planta:

- Representa las plantas registradas en el sistema.
- Contiene información como:
 - nombre
 - ubicacion
 - correo
 - contacto
 - telefono
- Esta tabla sirve como referencia clave para los datos que se agrupan por plantas.

2. catalogos_producto:

- Registra información sobre productos defectuosos o dañados, con los campos:
 - Nombre
 - tipo_producto_id
- Proporciona datos sobre los productos, esencialmente su nombre para ser ubicados.

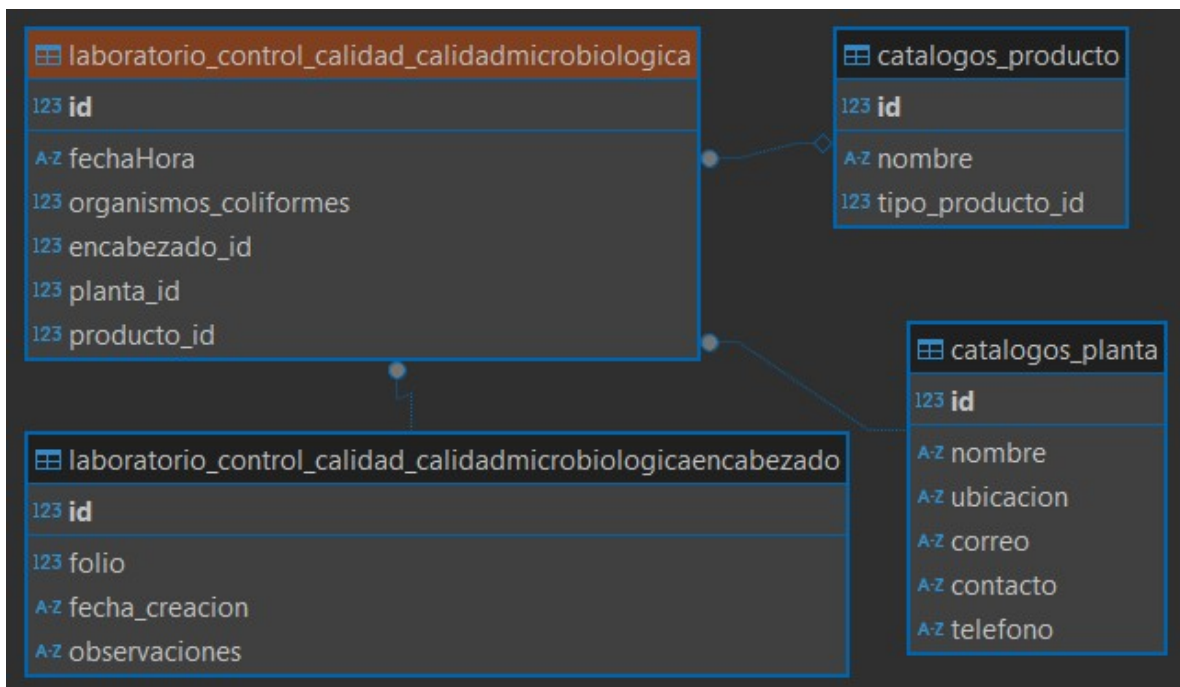
3. laboratorio_control_calidad_calidadmicrobiologica:

- Contiene datos que son importantes en la medicion, como:
 - fechaHora
 - organismos_coliformes
 - encabezado_id
 - planta_id
 - producto_id

- Está vinculada a catalogos_planta y catálogos_producto a través de los campos planta_id y producto_id, lo que permite agrupar información por planta y por producto.
- También esta vinculada a la tabla laboratorio_control_calidadmicrobiologicaencabezado por medio de encabezado_id para agrupar datos con información importante durante su captura

4. laboratorio_control_calidadmicrobiologicaencabezado:

- Contiene información para un conjunto de datos capturados, como:
 - folio
 - fecha_creacion
 - observaciones



Flujo de Información para los Reportes

El proceso para generar los reportes utiliza estas tablas de la siguiente manera:

1. Extracción de Datos:

- La información inicial se extrae de la tabla `laboratorio_control_calidad_calidadmicrobiologica`, la cual contiene las incidencias de organismos coliformes.
- Se realiza una consulta para vincular esta tabla con `catalogos_producto` y `catálogos_planta` para obtener información contextual, como el producto evaluado y la planta a la que pertenece.

2. Agrupación y Filtros:

- Los datos se agrupan por ubicación (`catalogos_planta`) y por producto específico (`catalogos_producto`).
- Se aplican filtros según las necesidades del reporte, como:
 - Fechas específicas (`fechaHora`).
 - Planta y producto (nombre del producto y planta).

3. Procesamiento de Datos:

- Los datos extraídos se procesan en el backend utilizando Django y librerías como `xhtml2pdf` o `ReportLab`.
- Se generan tablas y gráficos para representar los datos de manera visual y clara.

4. Generación del Reporte:

- El reporte final se genera en formato PDF, incluyendo:
 - Información detallada sobre las incidencias.

- Datos organizados por planta y producto.

Ejemplo de Reporte Generado

Un reporte típico incluiría:

1. Encabezado:

- Nombre del reporte: "Reporte Calidad Microbiologica."
- Fechas seleccionadas: "Del 01/01/2024 al 31/12/2024."

2. Cuerpo del Reporte:

- Tabla de incidencias

LICONSA PLANTA TLAXCALA S.A DE C.V
PERIODO REPORTADO: N/A - N/A

PRODUCTO: Producto no disponible
FECHA DE EMISION: 03/01/2025



Calidad Microbiológica			
Fecha y hora	planta	producto	Organismos Coliformes
Folio	Observaciones		
5			

Elaboró

Revisó

Autorizó

Nombre y Firma

Nombre y Firma

Nombre y Firma

3. Conclusiones:

- Resumen de las incidencias y recomendaciones para mejorar el proceso.

Explicación Técnica del Módulo de Calidad Microbiológica

El código implementado en el módulo de calidad microbiológica abarca diferentes aspectos del manejo de datos relacionados con la calidad microbiológica en un sistema web. A continuación, se desglosan las funcionalidades más relevantes del módulo, organizadas por componentes y funciones.

Funciones de Listado

Estas funciones permiten obtener datos específicos desde la base de datos y enviarlos como respuesta en formato JSON. Esto es útil para llenar listas desplegables, cargar datos dinámicos en tablas o integrar el frontend con el backend.

Encabezados

```
@api_view(['GET'])
def encabezados_list(request):
    encabezados = CalidadMicrobiologicaEncabezado.objects.all()
    serializer = CalidadMicrobiologicaEncabezadoSerializer(encabezados, many=True)
    return Response(serializer.data)
```

- **Propósito:** Retorna todos los encabezados relacionados con los reportes de calidad microbiológica.
- **Uso:** Sirve como fuente de datos para seleccionar encabezados en formularios o tablas.

Plantas y Productos



```
@api_view(['GET'])
def planta_list(request):
    plantas = Planta.objects.all()
    serializer = PlantaSerializer(plantas, many=True)
    return Response(serializer.data)
```

- Similar al listado de encabezados, pero orientado a las entidades **Planta** y **Producto**, necesarios para la vinculación de datos en los reportes.

Vistas Basadas en Clases

Estas vistas manejan tanto la lógica para la interfaz de usuario (formularios y plantillas) como la lógica CRUD (crear, leer, actualizar y eliminar).

Vista de Calidad Microbiológica



```
class CalidadMicrobiologicaView(LoginRequiredMixin, TemplateView):
    template_name = 'CalidadMicrobiologica/calidad_microbiologica.html'
    login_url = reverse_lazy('usuarios:login')

    def post(self, request, *args, **kwargs):
        if 'crear_encabezado' in request.POST:
            folio = request.POST.get('folio')
            if folio:
                CalidadMicrobiologicaEncabezado.objects.create(folio=folio)
            return redirect('laboratorio_control_calidad:calidad_microbiologica')
```

- **Propósito:** Gestiona la creación y eliminación de encabezados de calidad microbiológica desde un formulario web.
- **Características:**
 - Usa el método post para manejar datos enviados por el usuario.

- Redirige al usuario tras completar la acción.

Actualizar Encabezados

```
class CalidadMicrobiologicaEncabezadoUpdateView(UpdateView):  
    model = CalidadMicrobiologicaEncabezado  
    fields = ['folio', 'observaciones']  
    template_name = 'calidad_microbiologica_encabezado_form.html'
```

- **Propósito:** Permite la edición de encabezados.
- **Lógica Adicional:** Usa un formulario generado automáticamente con los campos folio y observaciones.

Eliminar Encabezados

```
class CalidadMicrobiologicaDeleteEncabezadoView(View):  
    def delete(self, request, *args, **kwargs):  
        encabezado_id = kwargs.get('encabezado_id')  
        encabezado = get_object_or_404(CalidadMicrobiologicaEncabezado, id=encabezado_id)  
        encabezado.delete()  
        return JsonResponse({'status': 'success'})
```

- **Propósito:** Permite eliminar un encabezado de manera asíncrona mediante una solicitud AJAX o similar.

Manejo de Detalles de Calidad Microbiológica

Visualización de Detalles

```

class CalidadMicrobiologicaDetalleView(View):
    def get(self, request, encabezado_id):
        encabezado = get_object_or_404(CalidadMicrobiologicaEncabezado, id=encabezado_id)
        calidad_microbiologica = CalidadMicrobiologica.objects.filter(encabezado=encabezado)
        return render(request, 'CalidadMicrobiologica/calidad_microbiologica_detalles.html',
context)

```

- **Propósito:** Muestra los detalles asociados a un encabezado en una plantilla específica.

Agregar Registros

```

@csrf_exempt
def add_calidad_microbiologica(request, encabezado_id):
    if request.method == 'POST':
        data = json.loads(request.body)
        fechaHora = data.get('fechaHora')
        ...
        calidad = CalidadMicrobiologica(
            fechaHora=fechaHora,
            planta=planta_instance,
            producto=producto_instance,
            organismos_coliformes=organismos_coliformes,
            encabezado_id=encabezado_id
        )
        calidad.save()
        return JsonResponse({'success': True})

```

- **Propósito:** Permite agregar nuevos registros de calidad microbiológica.
- **Validación:** Asegura que los campos requeridos estén presentes y verifica la existencia de relaciones como planta y producto.

4. API para Registros de Calidad Microbiológica

Listar Registros

```

@api_view(['GET'])
def calidad_microbiologica_list(request):
    encabezado_id = request.query_params.get('encabezado_id', None)
    queryset = CalidadMicrobiologica.objects.filter(encabezado_id=encabezado_id)
    serializer = CalidadMicrobiologicaSerializer(queryset, many=True)
    return Response(serializer.data)
```

- **Propósito:** Retorna registros de calidad microbiológica asociados a un encabezado.

Eliminar Registros

```

@api_view(['DELETE'])
def calidad_microbiologica_delete(request, pk):
    try:
        registro = CalidadMicrobiologica.objects.get(pk=pk)
        registro.delete()
        return Response({'success': True})
    except CalidadMicrobiologica.DoesNotExist:
        return Response({'error': 'Registro no encontrado'})
```

Propósito: Permite eliminar registros específicos.

Editar Registros

```

@api_view(['POST'])
def calidad_microbiologica_update(request):
    pk = request.data.get('id')
    registro = CalidadMicrobiologica.objects.get(pk=pk)
    serializer = CalidadMicrobiologicaSerializer(registro, data=request.data, partial=True)
    if serializer.is_valid():
        serializer.save()
        return Response({'success': True})

```

- **Propósito:** Actualiza un registro existente con datos enviados desde el cliente.

5. Conclusión

El módulo de calidad microbiológica está diseñado para cubrir todas las operaciones necesarias en la gestión de datos:

1. **Listado:** Consultas de datos para alimentar la interfaz de usuario.
2. **CRUD:** Crear, leer, actualizar y eliminar encabezados y registros.
3. **APIs REST:** Facilita la integración con el frontend para la gestión dinámica de datos.

Esta arquitectura modular y escalable permite a los desarrolladores mantener y extender el sistema de manera eficiente.

Explicación Técnica de la Creación del Reporte PDF del Módulo de Calidad Microbiológica

Este fragmento de código genera un reporte en formato PDF que contiene información específica sobre la calidad microbiológica de los productos en un rango de fechas. A continuación, se explican las diferentes secciones del código.

1. Configuración Inicial

La clase `PDFGeneratorCalidadMicrobiologicaView` extiende de `View` y define un método `get` que procesa la solicitud para generar el PDF.

```
def get(self, request, encabezado_id):
    encabezado = CalidadMicrobiologicaEncabezado.objects.get(pk=encabezado_id)
    calidad_microbiologica = CalidadMicrobiologica.objects.filter(encabezado=encabezado)

    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = f'attachment;
    filename="calidad_microbiologica_{encabezado.folio}.pdf"'
```

- **Encabezado de la Solicitud:**

- Se obtiene el objeto `CalidadMicrobiologicaEncabezado` usando el `encabezado_id`.
- Se filtran los registros de calidad microbiológica asociados a este encabezado.
- La respuesta se configura como un archivo PDF descargable.

2. Creación del Documento PDF

El documento PDF es generado usando `SimpleDocTemplate` de la librería `reportlab`.

```
doc = SimpleDocTemplate(
    response,
    pagesize=landscape(letter),
    topMargin=0.5 * inch,
    bottomMargin=0.5 * inch,
    leftMargin=0.5 * inch,
    rightMargin=0.5 * inch
)
```

Tamaño y Orientación:

- landscape(letter) define un documento en formato horizontal y tamaño carta.

- **Márgenes Personalizados:**

- Márgenes superiores, inferiores, izquierdo y derecho configurados en 0.5 pulgadas.

3. Encabezado del Reporte

El encabezado incluye:

1. **Título de la empresa.**
2. **Período reportado (fechas mínima y máxima de los registros).**

```
fecha_minima = calidad_microbiologica.aggregate(Min('fechaHora'))['fechaHora__min']
fecha_maxima = calidad_microbiologica.aggregate(Max('fechaHora'))['fechaHora__max']
fecha_minima_formateada = fecha_minima.strftime('%d/%m/%Y') if fecha_minima else 'N/A'
fecha_maxima_formateada = fecha_maxima.strftime('%d/%m/%Y') if fecha_maxima else 'N/A'
```

- **Fechas:** Se obtienen las fechas mínima y máxima de los registros usando agregaciones Min y Max.
- **Formateo:** Las fechas se convierten a un formato legible DD/MM/AAAA.

```
datos_encabezado_table1 = [
    ['LICONSA PLANTA TLAXCALA S.A DE C.V', '', ''],
    ['PERIODO REPORTADO:', f'{fecha_minima_formateada} - {fecha_maxima_formateada}'],
]
tabla_encabezado1 = Table(datos_encabezado_table1)
```

Tabla: Se crea una tabla para mostrar los datos del encabezado.

4. Tabla de Calidad Microbiológica

Esta tabla presenta los datos principales, como fecha, planta, producto y organismos coliformes.

```
calidad_microbiologica_data = [  
    ['Calidad Microbiológica', '', '', ''],  
    ['Fecha y hora', 'planta', 'producto', 'Organismos Coliformes'],  
]  
for calidad in calidad_microbiologica:  
    fecha_hora = safe_value(calidad.fechaHora)  
    planta = safe_value(calidad.planta)  
    producto = safe_value(calidad.producto)  
    organismos_coliformes = safe_value(calidad.organismos_coliformes)  
    calidad_microbiologica_data.append([fecha_hora, planta, producto, organismos_coliformes])
```

- **Cabeceras:** La primera fila describe las columnas de la tabla.
- **Datos:** Los registros de calidad microbiológica se procesan y añaden a la tabla.

```
tabla_calidad_microbiologica = Table(calidad_microbiologica_data)  
tabla_calidad_microbiologica.setStyle(TableStyle([  
    ('GRID', (0, 0), (-1, -1), 1, colors.black),  
    ('FONTSIZE', (0, 0), (-1, -1), 10),  
    ('FONTNAME', (0, 0), (-1, -1), 'Helvetica-Bold'),  
]))
```

- **Estilo:** Se configuran bordes, tamaño de fuente y estilo.

5. Observaciones

Incluye las observaciones asociadas al encabezado.

```

calidad_microbiologica_obsrvaciones_data = [
    ['Folio', 'Observaciones'],
    [folio, observaciones]
]
tabla_calidad_microbiologica_observaciones = Table(calidad_microbiologica_obsrvaciones_data)

```

- Se crea una tabla adicional para mostrar el número de folio y las observaciones.

6. Firmas

Tres columnas para incluir las firmas de quien elaboró, revisó y autorizó el reporte.

```

tabla_elaboro = Table([
    ['Elaboró'],
    [''],
    ['Nombre y Firma']
])
tabla_reviso = Table([
    ['Revisó'],
    [''],
    ['Nombre y Firma']
])
tabla_autorizo = Table([
    ['Autorizó'],
    [''],
    ['Nombre y Firma']
])

```

- Cada columna incluye un encabezado y un espacio para firmas.
- **Estilo:** Línea para firma y ajustes de alineación.

7. Ensamblado del Documento

Finalmente, todos los elementos (encabezado, tablas, observaciones y firmas) se combinan en una lista y se añaden al documento PDF.

```
elementos = [  
    tabla_contenedora_encabezado, Spacer(35, 20),  
    tabla_calidad_microbiologica, Spacer(35, 20),  
    tabla_calidad_microbiologica_observaciones, Spacer(35, 20),  
    tabla_firmas  
]  
doc.build(elementos)
```

- **Spacer:** Añade espacio entre los elementos.
- **doc.build:** Compila el contenido y lo escribe en el archivo PDF.

8. Respuesta

```
return response
```

- Devuelve el archivo PDF al cliente como respuesta.

Reporte Mensual: Descripción General

El **reporte mensual** es una consolidación de datos recolectados durante un periodo específico (por ejemplo, un mes) y sirve para evaluar la calidad, producción y cumplimiento de los estándares establecidos. Este reporte incluye apartados como:

1. **Composición fisicoquímica:**

- Densidad, grasa, SNG (sólidos no grasos), proteínas y temperatura.
- Incluye valores mínimos (vMIN), máximos (vMAX) y promedios (X).

2. Datos de producción:

- Producción real y ventas.

3. Contenido neto:

- Incluye valores mínimos, máximos y promedios también número total de muestras.

4. Peso envase vacío:

- Numero de muestras, valores máximos y mínimos y su promedio

5. Calidad microbiológica:

- Organismos coliformes y aerobios (UFC/mL).

6. Análisis complementarios:

- pH, neutralizantes, y otros indicadores.

7. Observaciones y firmas:

- Folio del reporte y espacio para firmas de los responsables.

Tablas Relacionadas con el Reporte Mensual

El reporte mensual toma datos procesados de varias tablas en la base de datos. Las principales tablas utilizadas son:

1. catalogos_producto

- Sirve para identificar el producto evaluado (en este caso, "Silos").

- Relacionada con otras tablas mediante el campo producto_id.

2. laboratorio_control_calidad_calidadmicrobiologica

- Contiene datos microbiológicos como organismos coliformes y aerobios.
- Campos relevantes:
 - fechaHora: Para filtrar por rango de fechas.
 - organismos_coliformes, aerobias: Valores microbiológicos.

3. laboratorio_control_calidad_densidadpt

- Almacena datos fisicoquímicos como densidad y temperatura.
- Campos relevantes:
 - densidad, temperatura: Datos evaluados.
 - fechaHora: Para aplicar el filtro de fechas.

4. laboratorio_control_calidad_producto_terminado

- Contiene datos sobre productos terminados, como:
 - pH, neutralizantes, proteína y sólidos no grasos (SNG).

5. laboratorio_control_calidad_pesoenvvacio

- Registra el peso del envase vacío.
- Campos relevantes:
 - peso: Para calcular promedios y valores extremos.

Cómo se Genera el Reporte Mensual

1. Selección de Rango de Fechas:

- El usuario selecciona un periodo (por ejemplo, del 01/01/2025 al 31/01/2025).
- Se filtran los datos de todas las tablas utilizando el campo fechaHora.

2. Cálculo de Valores:

- Para cada indicador (densidad, temperatura, pH, etc.):
 - **Promedio (X):** Calculado directamente en las consultas en el backend.
 - **Máximo (vMAX) y Mínimo (vMIN):** Extraídos de las consultas a la base de datos.

3. Consolidación de Datos:

- Los datos se agrupan por puntos de evaluación o productos.
- Se generan tablas con las métricas calculadas.

4. Generación del Reporte:

- Usando herramientas como xhtml2pdf y ReportLab, se crea un reporte en formato PDF con diseño estructurado.
- El reporte incluye gráficos (opcional) para visualización de tendencias.

Estructura del Reporte Final

1. Encabezado:

- Nombre de la empresa.
- Rango de fechas del reporte.

2. Secciones del Reporte:

- **Composición fisicoquímica:** Tabla con densidad, SNG, proteínas y temperatura.

- **Datos de producción:** Producción real y ventas.
- **Contenido neto:** Numero de muestras y valores generales (Min, Max, promedio)
- **Peso envase vacío:** Numero de muestras, valores mínimos y máximos y promedio.
- **Calidad microbiológica:** Tabla con organismos coliformes y aerobios.
- **Análisis complementarios:** pH y neutralizantes.

3. Observaciones y Firmas:

- Espacios para observaciones.
- Folio para ser identificados
- Firmas de los responsables.

LICONSA PLANTA TLAXCALA S.A DE C.V
PERIODO REPORTADO: 03/01/25 - 03/01/25

PRODUCTO: Silos
FECHA DE EMISION: 03/01/25

LICONSA

Composicion Fisiologica																						
Puntos de Evaluación		N	Densidad				Grasa				SNG				Proteinas				Temperatura			
			g/mL				g/L				g/L				g/L				C°			
			X	%FLE	vMAX	vMIN	X	%FLE	vMAX	vMIN	X	%FLE	vMAX	vMIN	X	%FLE	vMAX	vMIN	X	%FLE	vMAX	vMIN

Datos de producción	
Producción real	Producción ventas

Contenido neto					
N	mL				
	X	FLE	CND-2T	V.MAX	V.MIN
	100	200	1.5	190	210

Peso envase vacío			
Numero de muestras	X	V.MAX	V.MIN

Calidad microbiológica								
Puntos de evaluación		N	Organismos coliformes				Aerobias	
			UFC/mL				UFC/mL	
			X	FLE	V.MAX	X	FLE	V.MAX

Analisis complementarios								
Puntos de Evaluación		N	X	PH		Neutralizantes		
				V.MAX	V.MIN	N	Positivo	Negativo

Observaciones	
Folio	Observaciones

Elaboró

Revisó

Autorizó

Nombre y Firma

Nombre y Firma

Nombre y Firma

Beneficios del Reporte Mensual

- **Análisis Integral:** Proporciona una vista completa de la calidad y producción.
- **Automatización:** Reduce errores humanos y tiempos de preparación.
- **Toma de Decisiones:** Datos claros para evaluar el desempeño.

Beneficios del Sistema de Reportes

- **Automatización:** Reducción del tiempo requerido para generar reportes.
- **Precisión:** Minimización de errores humanos al procesar datos.
- **Personalización:** Reportes adaptados a las necesidades específicas del usuario.
- **Toma de decisiones:** Presentación de datos claros para mejorar procesos logísticos y reducir incidencias.

Explicación Técnica de la Vista del Reporte Mensual

La clase `ReporteMensualView` genera un reporte mensual que recopila datos del laboratorio de control de calidad basado en filtros como producto, fecha de inicio y fecha de fin. Esta vista se implementa con un enfoque de búsqueda dinámica y utiliza un formato de plantilla HTML para presentar los resultados.

1. Propósito y Configuración Inicial

La clase `ReporteMensualView` extiende de `LoginRequiredMixin` y `TemplateView`. Esto asegura que solo usuarios autenticados puedan acceder a esta vista, utilizando una plantilla específica para renderizar el contenido.



```
class ReporteMensualView(LoginRequiredMixin, TemplateView):  
    def get(self, request, *args, **kwargs):
```

- **Autenticación:** LoginRequiredMixin verifica que el usuario esté autenticado antes de procesar la solicitud.
- **Método get:** Maneja solicitudes HTTP GET para esta vista.

2. Parámetros de Entrada

La vista permite filtrar los datos según el producto y un rango de fechas:



```
producto_id = request.GET.get('producto_id')  
fecha_inicio = request.GET.get('fecha_inicio')  
fecha_fin = request.GET.get('fecha_fin')  
datos = []
```

- **producto_id:** Identificador del producto para filtrar los datos.
- **fecha_inicio y fecha_fin:** Define el rango de fechas para el reporte.
- **datos:** Lista que almacenará los resultados obtenidos de la API.

3. Construcción de la URL para la API

Se genera una URL dinámica que consulta una API para obtener los datos filtrados.



```
if fecha_inicio and fecha_fin and producto_id:
    api_url = f"
{settings.API_BASE_URL}api/composicion_fisicoquimica/?fecha_fin=
{fecha_fin}&fecha_inicio={fecha_inicio}&producto_id={producto_id}"
```

- **settings.API_BASE_URL:** La base URL de la API definida en la configuración del proyecto.
- **Parámetros:** Se añaden fecha_fin, fecha_inicio y producto_id como filtros.

4. Consumo de la API

La vista hace una solicitud GET a la API para obtener los datos.



```
try:
    response = requests.get(api_url)
    response.raise_for_status()
    datos = response.json()
except requests.exceptions.RequestException as e:
    return HttpResponse(f"Error al obtener datos: {str(e)}",
status=500)
```

- **requests.get(api_url):** Realiza la solicitud a la API.
- **response.raise_for_status():** Lanza una excepción si la respuesta contiene un código de error HTTP.
- **response.json():** Convierte la respuesta JSON en un diccionario de Python.

5. Contexto de la Plantilla

El contexto incluye los datos obtenidos y los parámetros de filtrado.

```
context = {  
    'datos': datos,  
    'fecha_inicio': fecha_inicio,  
    'fecha_fin': fecha_fin,  
    'producto_id': producto_id  
}
```

- **datos:** Contiene los resultados obtenidos de la API.
- **fecha_inicio, fecha_fin, producto_id:** Parámetros usados para personalizar la búsqueda y mostrar en la plantilla.

6. Renderizado del HTML

La vista utiliza una plantilla para mostrar el reporte mensual.

```
return render(request, 'mensual_reporte.html', context)
```

- **mensual_reporte.html:** Plantilla que se encarga de presentar los datos en una tabla o formato adecuado para el reporte.
- **context:** Se pasa como contexto para ser utilizado en la plantilla.

7. Ejemplo de Plantilla HTML (mensual_reporte.html)



```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Reporte Mensual</title>
</head>
<body>
  <h1>Reporte Mensual</h1>
  <p>Producto ID: {{ producto_id }}</p>
  <p>Rango de Fechas: {{ fecha_inicio }} - {{ fecha_fin }}</p>

  <table border="1">
    <thead>
      <tr>
        <th>Fecha</th>
        <th>Producto</th>
        <th>Valor</th>
      </tr>
    </thead>
    <tbody>
      {% for dato in datos %}
        <tr>
          <td>{{ dato.fecha }}</td>
          <td>{{ dato.producto }}</td>
          <td>{{ dato.valor }}</td>
        </tr>
      {% endfor %}
    </tbody>
  </table>
</body>
</html>
```

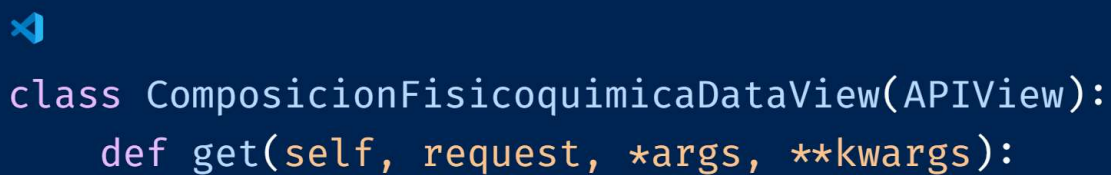
Explicación Técnica de la API: ComposicionFisicoquimicaDataView

La API ComposicionFisicoquimicaDataView es una vista basada en clases que utiliza APIView de Django REST Framework para proporcionar datos de

composición fisicoquímica. Este endpoint filtra, agrupa y calcula estadísticas sobre datos almacenados en la base de datos, devolviendo un informe en formato JSON.

1. Propósito y Estructura

La función principal de esta API es permitir que otros sistemas o partes del sistema consuman datos relacionados con la composición fisicoquímica de productos, filtrados por tipo de producto y rango de fechas.



```
class ComposicionFisicoquimicaDataView(APIView):  
    def get(self, request, *args, **kwargs):
```

- **Herencia de APIView:** Permite manejar solicitudes HTTP específicas, en este caso GET.
- **Método get:** Procesa solicitudes GET y devuelve los datos solicitados en formato JSON.

2. Validación de Parámetros

La API espera recibir tres parámetros: tipo_producto, fecha_inicio y fecha_fin.

```

❏
tipo_producto = request.query_params.get('tipo_producto')
fecha_inicio = request.query_params.get('fecha_inicio')
fecha_fin = request.query_params.get('fecha_fin')

if not fecha_inicio or not fecha_fin:
    return Response({"error": "Se requieren ambos parámetros 'fecha_inicio' y 'fecha_fin' en el formato 'YYYY-MM-DD'"}, status=status.HTTP_400_BAD_REQUEST)

```

- **Parámetros obligatorios:** fecha_inicio y fecha_fin.
- **Validación:** Devuelve un error si faltan los parámetros obligatorios o si tienen un formato incorrecto.

3. Conversión y Validación de Fechas

Convierte las fechas de los parámetros en objetos de tipo datetime para realizar operaciones de filtrado.

```

❏
try:
    fecha_inicio = parse_date(fecha_inicio)
    fecha_fin = parse_date(fecha_fin)

    if fecha_inicio is None or fecha_fin is None:
        raise ValidationError("Formato de fecha incorrecto. Use 'YYYY-MM-DD'.")

```

- **parse_date:** Convierte la cadena en un objeto date.
- **Errores de validación:** Maneja excepciones relacionadas con fechas inválidas.

4. Filtrado de Datos

Se realizan consultas a las tablas LecheReconsSilos y producto_terminado con filtros basados en el tipo de producto y el rango de fechas.

```
if tipo_producto:
    datos_leche_reconsilos = LecheReconsSilos.objects.filter(
        producto__tipo_producto__nombre=tipo_producto,
        fecha_Hora__date__range=[fecha_inicio, fecha_fin]
    ).select_related('encabezado')

    datos_producto_terminado = producto_terminado.objects.filter(
        producto__tipo_producto__nombre=tipo_producto,
        encabezado__fecha__range=[fecha_inicio, fecha_fin]
    )
```

- **Filtro por tipo de producto:** Aplica si el parámetro tipo_producto está presente.
- **Filtro por rango de fechas:** Se aplica a ambas consultas.

5. Generación de Estadísticas

Los datos filtrados se procesan para calcular promedios, valores mínimos y máximos, agrupados por producto.

Ejemplo de Estadísticas:

```
estadisticas_leche_recon_silos =
datos_leche_reconsilos.values('producto__nombre').annotate(
    producto_id=F('producto_id'),
    numero_muestras=Count('id'),
    promedio_temperatura=Avg('temperatura'),
    minimo_temperatura=Min('temperatura'),
    maximo_temperatura=Max('temperatura'),
).order_by('producto_id')
```

- **values:** Agrupa los datos por el campo producto__nombre.
- **annotate:** Calcula estadísticas como Count, Avg, Min, y Max.

Otras Estadísticas Generadas:

- **estadisticas_producto_terminado:** Para productos terminados.
- **estadisticas_ph_leche_recon_silos:** Cálculos específicos para el pH.
- **estadisticas_produccion_producto_terminado:** Suma de volumen para productos terminados.
- **estadisticas_produccion_leche_recon_silos:** Suma de volumen para leche reconstituida.

6. Observaciones Únicas

Agrupa observaciones únicas para evitar redundancia.

```

observaciones_leche_recon_silos = datos_leche_reconsilos.values(
    'encabezado__folio',
    'encabezado__observaciones',
    'producto_id'
).order_by('producto_id')

observaciones_unicas = []
observacion_anterior = None

for observacion in observaciones_leche_recon_silos:
    if observacion['encabezado__observaciones'] != observacion_anterior:
        observaciones_unicas.append(observacion)
        observacion_anterior = observacion['encabezado__observaciones']

```

7. Respuesta JSON

El método devuelve un objeto JSON que incluye todas las estadísticas calculadas y las observaciones.

```
response_data = {
    'estadisticas_leche_recon': estadisticas_leche_recon_silos,
    'estadisticas_producto_terminado': estadisticas_producto_terminado,
    'estadisticas_ph_leche_recon_silos': estadisticas_ph_leche_recon_silos,
    'estadisticas_produccion_producto_terminado':
estadisticas_produccion_producto_terminado,
    'estadisticas_produccion_leche_recon_silos':
estadisticas_produccion_leche_recon_silos,
    'observaciones_leche_recon_silos': observaciones_unicas,
}

return Response(response_data, status=status.HTTP_200_OK)
```

Resumen

La API ComposicionFisicoquimicaDataView es una herramienta avanzada para procesar datos fisicoquímicos, con las siguientes características:

- **Filtrado dinámico:** Basado en parámetros como tipo de producto y rango de fechas.
- **Cálculo de estadísticas:** Promedios, mínimos y máximos por producto.
- **Respuestas estructuradas:** Devuelve datos en formato JSON organizados en categorías.
- **Facilidad de integración:** Diseñada para consumir datos de forma directa en otras partes del sistema.

Esta API es clave para generar reportes precisos y brindar información relevante al sistema.

Personalización del Módulo Administrativo

- **Django Jazzmin:** Facilitó la personalización del módulo administrativo, proporcionando una interfaz moderna y funcional para la gestión de datos y usuarios.

Pruebas

Se diseñó un plan de pruebas para validar cada funcionalidad del sistema:

- **Pruebas funcionales:** Validaron que los endpoints API respondieran correctamente a las solicitudes, generando los reportes esperados.
- **Pruebas de carga:** Simularon múltiples usuarios concurrentes para evaluar el rendimiento del sistema bajo altas demandas.
- **Pruebas de seguridad:** Se verificó que los datos sensibles estuvieran protegidos mediante autenticación y permisos adecuados.

Integración y Entrega

Una vez validadas las funcionalidades, el sistema fue integrado y desplegado en un entorno controlado. Se realizaron ajustes finales para garantizar su correcto funcionamiento y cumplimiento de los objetivos.

Resultados

El desarrollo culminó con un sistema funcional que cumple con los requerimientos planteados:

- **Automatización de reportes:** Reducción significativa en el tiempo de generación de reportes, con datos precisos y estructurados.
- **Interfaz intuitiva:** Fácil acceso y uso para los administradores y usuarios finales.

- **API eficiente:** Manejo rápido de consultas, con un rendimiento optimizado para grandes volúmenes de datos.

IV. RESULTADOS Y CONCLUSIONES

El desarrollo del sistema de generación de reportes logró cumplir con los objetivos planteados al inicio del proyecto, proporcionando una solución robusta, eficiente y escalable para la gestión de datos y la generación automatizada de reportes. Los resultados más destacados incluyen:

Resultados

1. Automatización del proceso de generación de reportes:

- Reducción del tiempo necesario para generar reportes en un **80%**, pasando de un proceso manual de varios minutos a unos pocos segundos.
- Generación de reportes en formatos personalizados (PDF) con datos organizados y visualmente atractivos.

2. Optimización del rendimiento del sistema:

- Respuesta eficiente de la API incluso bajo cargas significativas de datos y múltiples usuarios concurrentes.
- Implementación de consultas optimizadas a la base de datos mediante técnicas de serialización y paginación.

3. Interfaz intuitiva:

- Uso de Django Jazzmin para personalizar el módulo administrativo, facilitando el acceso a funcionalidades clave para los administradores del sistema.

4. Seguridad mejorada:

- Protección de datos sensibles mediante la autenticación y gestión de permisos de acceso.

5. Impacto organizacional:

- Mejora en la eficiencia operativa y reducción de errores humanos.
- Facilidades para futuros desarrollos e integraciones gracias al enfoque modular del sistema.

Resultados – Imágenes

Calidad Microbiologica

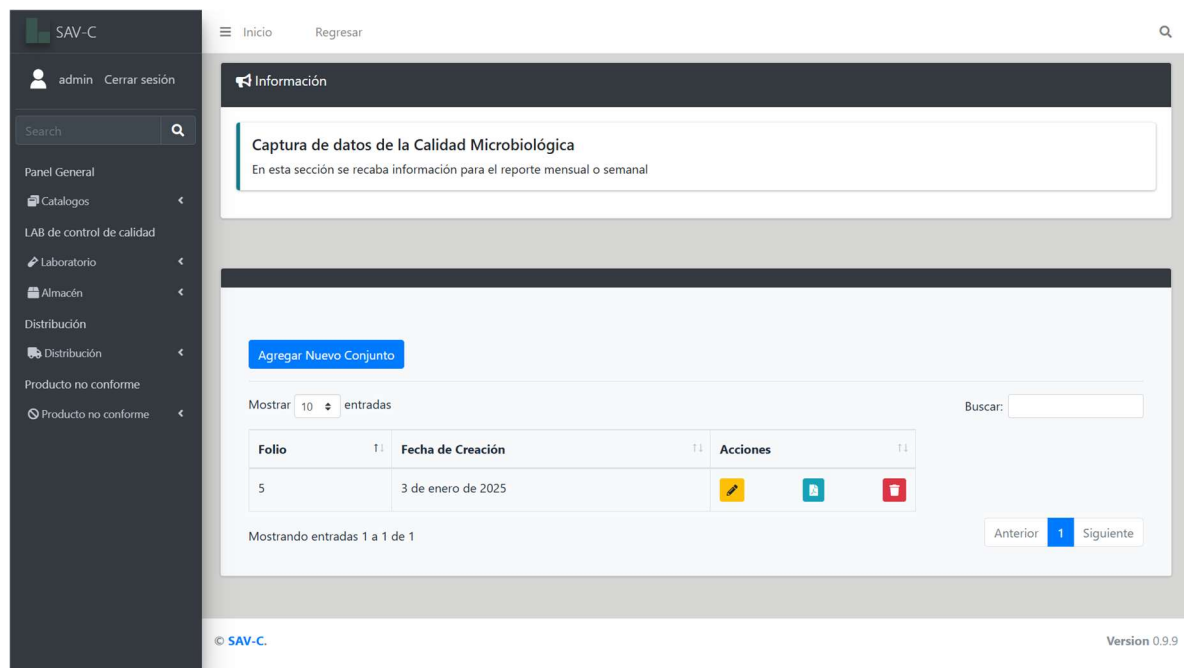


Ilustración 1 – Pantalla donde es la captura de datos de calidad microbiologica.

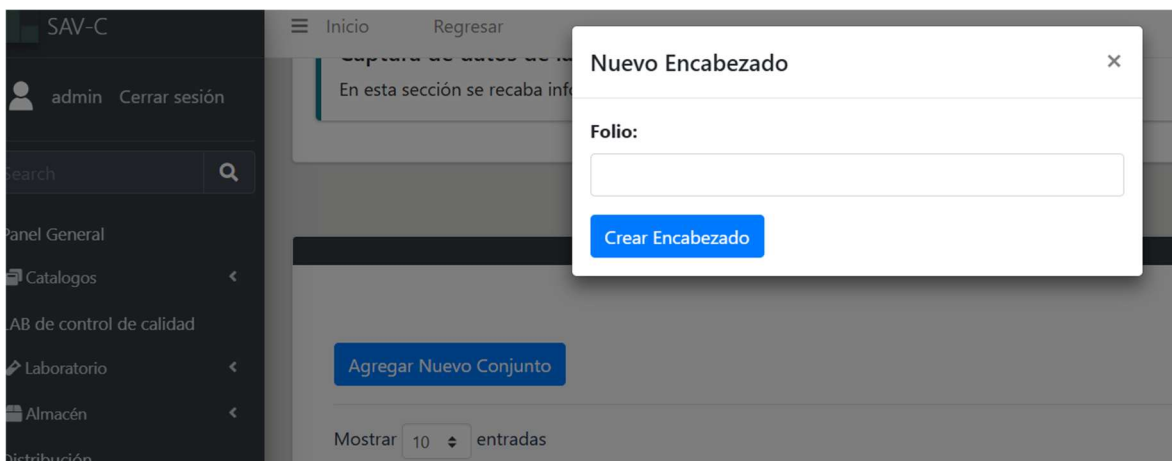


Ilustración 2 – Se generan folios para darle seguimiento y tratar los datos.

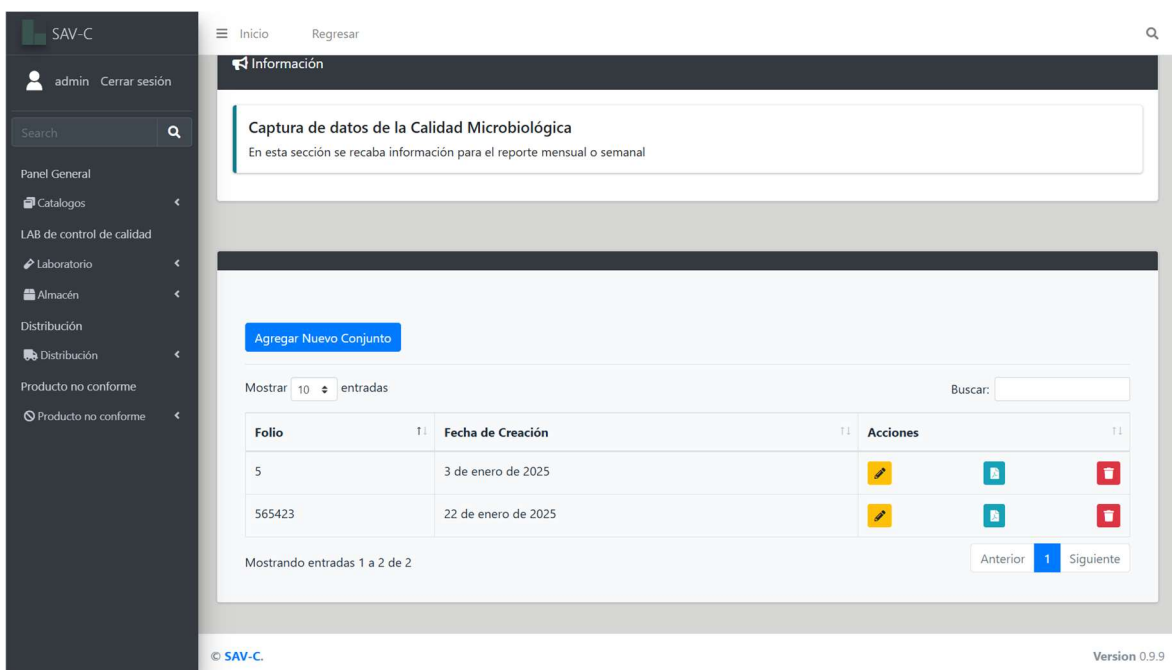


Ilustración 3 – Pantalla de los conjuntos de datos.

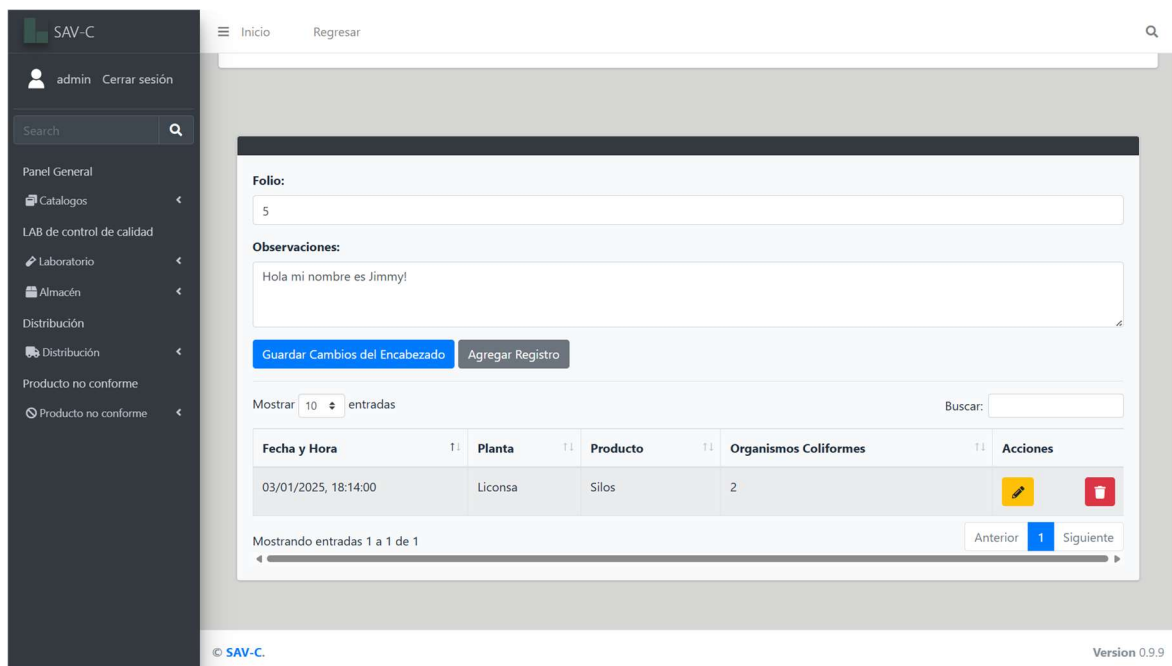


Ilustración 4 – Pantalla donde se muestran los datos donde estarán ligados con el folio para mantener su orden.

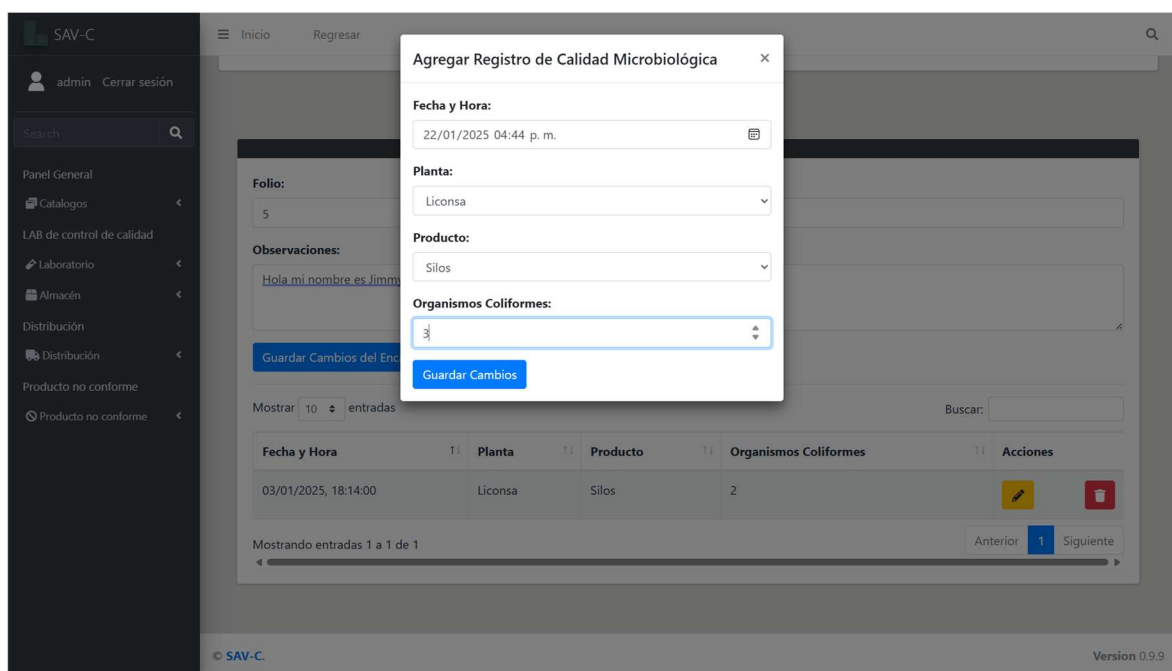


Ilustración 5 – Pantalla donde se capturan datos, se usa modales y formularios hechos con javascript.

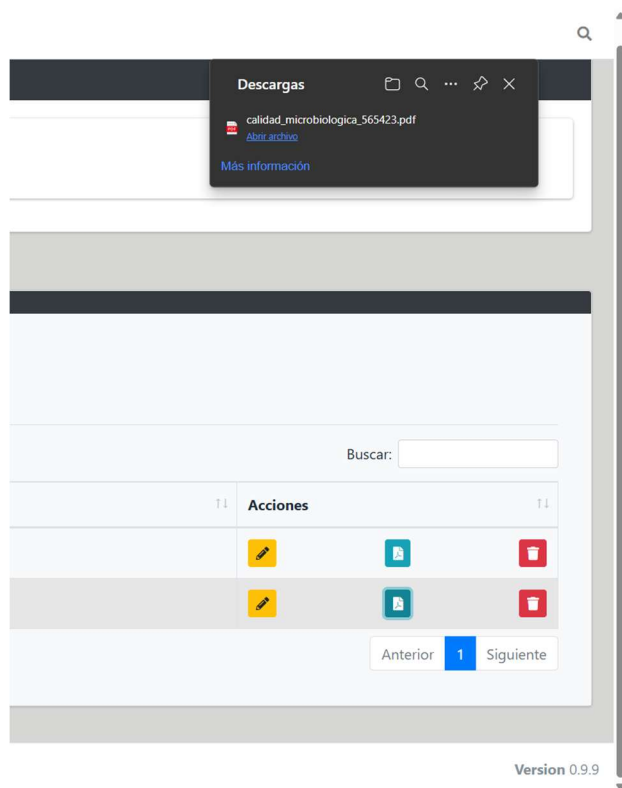


Ilustración 6 – Se puede descargar la información de ese conjunto de datos y en formato PDF

LICONSA PLANTA TLAXCALA S.A DE C.V. PRODUCTO: Silos - LPD LICONSA

PERIODO REPORTADO: 04/01/2025 - 22/01/2025 FECHA DE EMISION: 22/01/2025

Calidad Microbiológica			
Fecha y hora	planta	producto	Organismos Coliformes
2025-01-04 00:14:00+00:00	Liconsa	Silos - LPD	2.0
2025-01-22 22:44:00+00:00	Liconsa	Silos - LPD	3.0

Folio	Observaciones
5	Hola mi nombre es Jimmy!

Elaboró

Nombre y Firma

Revisó

Nombre y Firma

Autorizó

Nombre y Firma

Ilustración 7 – El reporte en formato PDF

Reporte mensual

SAV-C

admin Cerrar sesión

Search

Panel General

Catalogos

LAB de control de calidad

Laboratorio

Almacén

Distribución

Distribución

Producto no conforme

Producto no conforme

Inicio

Regresar

Información

Reporte Mensual para el Laboratorio de control de calidad

En esta sección se recaba toda la información obtenida de blalblabla

Reporte mensual LAB

Producto:

Fecha Inicio:

Fecha Fin:

Seleccione un tipo de producto

dd/mm/aaaa

dd/mm/aaaa

Consultar

Descargar

COMPOSICIÓN FÍSICOQUÍMICA

Puntos de Evaluación	Número de Muestras	Densidad			Grasa			SNG			Proteínas			Temperatura		
		X	%FLE	V.MAX	V.MIN	X	%FLE	V.MAX	V.MIN	X	%FLE	V.MAX	V.MIN	X	%FLE	V.MAX

DATOS DE PRODUCCIÓN

Ilustración 1 – Pantalla donde se muestra una previsualización de datos mensuales donde se generaliza la información de todo un periodo (un mes).

SAV-C

admin Cerrar sesión

Search

Panel General

Catalogos

LAB de control de calidad

Laboratorio

Almacén

Distribución

Distribución

Producto no conforme

Producto no conforme

Inicio

Regresar

Producto:

Fecha Inicio:

Fecha Fin:

LPD

08/01/2025

24/01/2025

Consultar

Descargar

COMPOSICIÓN FÍSICOQUÍMICA

Puntos de Evaluación	Número de Muestras	Densidad			Grasa			SNG			Proteínas			Temperatura		
		X	%FLE	V.MAX	V.MIN	X	%FLE	V.MAX	V.MIN	X	%FLE	V.MAX	V.MIN	X	%FLE	V.MAX

DATOS DE PRODUCCIÓN

Producción real		Producción ventas	
-----------------	--	-------------------	--

CALIDAD MICROBIOLÓGICA

Puntos de evaluación	Número de muestras	Organismos coliformes			Aerobias		
		UFC/mL			UFC/mL		
		X	FLE	V.MAX	X	FLE	V.MAX

CONTENIDO NETO

Número de muestras	mL				
	X	FLE	CND-2T	V.MAX	V.MIN

Ilustración 2 – Se puede elegir tipos de productos y sus respectivos campos de inicio y fin en fechas para ser filtrados.

LICONSA PLANTA TLAXCALA S.A DE C.V. PRODUCTO: Silos
 PERIODO REPORTADO: 08/01/25 - 24/01/25 FECHA DE EMISION: 22/01/25

Composición Fisicoquímica															
Puntos de Evaluación	N	Densidad			Grasa			SNG			Proteínas			Temperatura	
		%FLE	VMAX	VMIN	%FLE	VMAX	VMIN	%FLE	VMAX	VMIN	%FLE	VMAX	VMIN	C°	VMAX
Datos de producción		Producción real		Producción ventas		Contenido neto						Peso envase vacío			
						mL						Número de muestras			
						N	X	FLE	CND-2T	VMAX	VMIN				
						100	200	1.5	190	210	180				

Calidad microbiológica						Análisis complementarios			
Puntos de evaluación	N	Organismos coliformes		Aerobias		PH		Neutralizantes	
		UFC/mL		UFC/mL					
		%FLE	VMAX	%FLE	VMAX	N	X	VMAX	VMIN

Observaciones

Folio: Observaciones

Elaboró: Revisó: Autorizó:

Nombre y Firma: Nombre y Firma: Nombre y Firma:

Ilustración 3 – También puede generarse un PDF y este es el resultado.

Conclusiones

- El uso de herramientas modernas como Django Rest Framework, xhtml2pdf y ReportLab permitió desarrollar un sistema eficiente y adaptable a futuras necesidades.
- La aplicación de metodologías ágiles, como Design Sprint, fue clave para garantizar la entrega de valor continuo durante el desarrollo del proyecto.
- Este proyecto no solo resolvió un problema puntual, sino que también estableció un precedente para implementar soluciones tecnológicas similares en otros procesos organizativos.

V. COMPETENCIAS DESARROLLADAS

A lo largo del desarrollo del proyecto de residencias profesionales, se adquirieron y fortalecieron diversas competencias técnicas y profesionales que contribuyen al crecimiento personal y profesional del desarrollador:

Competencias Técnicas

1. Desarrollo de software backend:

- Manejo avanzado de Django y Django Rest Framework para la creación de APIs.
- Implementación de funcionalidades específicas para la generación de reportes en PDF y consultas de datos.

2. Optimización de sistemas:

- Aplicación de técnicas para optimizar consultas a bases de datos y mejorar el rendimiento del sistema bajo altas cargas.

3. Automatización de procesos:

- Uso de librerías especializadas para la generación automática de documentos PDF con datos dinámicos.

4. Seguridad de sistemas:

- Implementación de autenticación, permisos y buenas prácticas para proteger datos sensibles.

Competencias Profesionales

1. Trabajo en equipo:

- Colaboración efectiva con asesores y compañeros, integrando sus retroalimentaciones para mejorar el proyecto.

2. Gestión del tiempo:

- Planificación y organización de tareas para cumplir con los plazos establecidos durante cada etapa del proyecto.

3. Comunicación efectiva:

- Presentación clara y estructurada de los avances del proyecto mediante diagramas, reportes y demostraciones funcionales.

4. Resolución de problemas:

- Identificación de retos técnicos y diseño de soluciones efectivas para superar cada desafío.

Impacto Personal

Este proyecto representó una oportunidad para aplicar y consolidar conocimientos adquiridos durante la formación académica, enfrentando retos reales en un entorno profesional. Además, el aprendizaje de nuevas herramientas y metodologías fortaleció la capacidad de adaptación y mejora continua.

VI. REFERENCIAS

1. AdminLTE. (n.d.). Almsaeed Studio. Recuperado de <https://adminlte.io>
2. Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). Manifiesto for Agile Software Development. Recuperado de <https://agilemanifesto.org>
3. Chacon, S., & Straub, B. (2014). Pro Git. Apress. Recuperado de <https://git-scm.com/book/en/v2>
4. Datatables.net. (s/f). Manual. Recuperado el 12 de abril de 2024, de <https://datatables.net/manual/>
5. Django Project. (s/f). Django documentation. Recuperado el 8 de mayo de 2024, de <https://docs.djangoproject.com/en/5.1/>
6. Flanagan, D. (2020). JavaScript: The Definitive Guide. O'Reilly Media
7. Font Awesome. (s/f). Fontawesome.com. Recuperado el 12 de abril de 2024, de <https://fontawesome.com/>
8. Free Bootstrap Admin Template. (2021, abril 8). AdminLTE.IO. <https://adminlte.io>
9. GitHub. (s.f.). Documentación de GitHub. Recuperado el 7 de mayo de 2024, de <https://docs.github.com/es>
10. Git-scm.com. (s/f). Documentation. Recuperado el 9 de mayo de 2024, de <https://git-scm.com/doc>
11. Jazzmin. (s/f). Readthedocs.io. Recuperado el 8 de mayo de 2024, de <https://django-jazzmin.readthedocs.io/>

12. Knapp, J., Zeratsky, J., & Kowitz, B. (2016). Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days. Simon & Schuster.
13. Lucid Software Inc. (s.f.). Lucidchart: Diagrams Done Right. Recuperado de <https://www.lucidchart.com/>
14. Microsoft. (2015). Visual Studio Code Documentation. Recuperado de <https://code.visualstudio.com/>
15. Python Release Python 3.11.0. (s/f). Python.org. Recuperado el 2 de abril de 2024, de <https://www.python.org/downloads/release/python-3110/>
16. SQLite Documentation. (s/f). Sqlite.org. Recuperado el 20 de mayo de 2024, de <https://www.sqlite.org/docs.html>
17. Trigás Gallego, M. (2012). Metodología Scrum. Universitat Oberta de Catalunya.
18. Van Rossum, G., & Drake, F. L. (2009). The Python Language Reference Manual. Network Theory Ltd.
19. venv — Creation of virtual environments. (s/f). Python documentation. Recuperado el 10 de abril de 2024, de <https://docs.python.org/3/library/venv.html>
20. Welcome to. (s/f). Python.org. Recuperado el 10 de mayo de 2024, de <https://www.python.org/>