UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Desarrollo Back-End: Diseño e integración para software Dinodock

Alexander Jonathan Bernal Diaz

Alexander.bernal@sansano.usm.cl

Catherine Gómez

Profesora Guía

Resumen

La empresa Sitrans Depósito & Logística enfrenta problemas de eficiencia en la gestión de contenedores vacíos debido al aumento del comercio internacional, lo que genera altos tiempos de espera y costos operativos. La solución propuesta consiste en desarrollador un software que optimice el depósito de contenedores, utilizando funciones algorítmicas y tecnologías como AWS y Python para mejorar la gestión y minimizar movimientos innecesarios.

Los objetivos del trabajo abarcan la implementación de un sistema de autenticación, diseño de una API y la optimización de consulta a la Base de Datos. Se aplicará la metodología Scrum para facilitar la comunicación y el desarrollo ágil del proyecto.

Los resultados esperados consisten en la agilización de las actividades de entrada y salida de contenedores y una alta seguridad en la transferencia y procesamiento de datos. Se espera que la solución impacte positivamente en la eficiencia operativa de Sitrans, reduciendo costos y tiempos de espera, lo que beneficiará tanto a la empresa como sus clientes.

Palabras clave: Eficiencia, Optimización, Contenedores, AWS.

1 Introducción

1.1 Contexto y Antecedentes

La empresa Sitrans Depósito & Logística, es una organización encargada de la recepción, despacho y almacenaje de contenedores vacíos a lo largo de todo Chile. Negocio que, con la creciente demanda de transporte marítimo, impulsada por el aumento del comercio internacional, ha llevado a la necesidad de agilizar el movimiento de contenedores para reducir costos en combustibles, minimizar tiempos de espera y mejorar la sostenibilidad.

1.2 Definición del Problema

La problemática se origina en una falta significativa de un proceso altamente cualificado y orientado al depósito y almacenaje eficiente de contenedores vacíos. Ocasionando altos tiempos de espera en los clientes, aumentos considerables en el uso de combustibles y alta demanda de mano de obra capacitada para realizar estas acciones en conjunto.

Bajo esta mirada, un sistema de información con un íntegro y correcto desarrollo back-end se vuelve fundamental. Utilizando tecnologías de alta gama, fiabilidad, seguridad y escalabilidad.

De las cuales se destacan:

- Base de Datos, encargadas de guardar, recuperar y manipular grandes volúmenes de datos.
- API (Interfaz de Programación de Aplicaciones) que permite la comunicación entre servidores y clientes, facilitando el intercambio de datos.
- Lenguajes de programación como Python, de gran versatilidad e integración, portable para muchos sistemas operativos e ideal para el análisis de datos.
- Servicios en la nube como AWS, Azure y más, capaces de ofrecer todos los servicios anteriormente mencionados en un solo entorno capaz de comunicarse sin inconvenientes, sin menospreciar la seguridad, calidad y escalabilidad de los sistemas de información que utilicen esta tecnología.

1.3 Descripción de la Propuesta de Solución

La solución planteada se basa en el desarrollo de un software especializado en la optimización del almacenaje y manejo de contenedores vacíos dentro de un espacio geográfico delimitado, un desafío logístico crucial para empresas del sector portuario. Este enfoque se orienta hacia la creación de un sistema eficiente, inteligente y automatizado que permita a los operarios maximizar el uso del espacio disponible y agilizar los procesos de entrada y salida, resolviendo así uno de los principales dolores operativos en el ámbito logístico.

El núcleo de la solución radica en la definición de la mejor posición para contenedores entrantes utilizando funciones algorítmicas de optimización. Estas funciones se diseñarán para analizar múltiples variables relevantes, como el tamaño, el tipo, la disponibilidad del espacio y la disposición actual del patio. Este análisis permitirá identificar el lugar más eficiente para almacenar cada contenedor, minimizando los movimiento innecesarios y garantizando un uso óptimo del espacio. Además, el sistema será capaz de adaptarse dinámicamente a cambios en las condiciones operativas, como fluctuaciones en la demanda o restricciones en el espacio, asegurando siempre un desempeño óptimo.

Por otro lado, el software también abordará el proceso de salida de contenedores, seleccionando el contenedor más adecuado según las demandas específicas de los clientes. Estas demandas pueden incluir variables como la naviera de origen, el tamaño del contenedor, su tipo, el tiempo de estadía en el patio y otros requerimientos particulares. Utilizando una función algorítmica avanzada, el sistema analizará todas estas variables para encontrar el contenedor que mejor se ajuste a las necesidades del cliente, a la vez que minimiza la cantidad de movimientos requeridos. Este enfoque no solo agilizará el proceso de entrega, sino que también reducirá los tiempos de espera, optimizará el uso de combustibles y disminuirá el desgaste de los equipos involucrados, como grúas y vehículos de transporte.

Para garantizar un desempeño robusto y escalable, la solución será implementada en un entorno en la nube utilizando Amazon Web Services (AWS). Este servicio en la nube ofrece una plataforma confiable y flexible que permitirá gestionar de manera eficiente todos los aspectos del sistema, desde la infraestructura básica hasta las operaciones más complejas.

Además, todas las operaciones y aplicaciones estarán protegidas por los altos estándares de seguridad que ofrece AWS, incluyendo encriptación de datos, autenticación robusta de usuarios y monitoreo constante de posibles amenazas. Estas medidas garantizan no solo la integridad y confidencialidad de los datos, sino que también la continuidad operativa del sistema, incluso en escenarios de alta demanda o contingencias técnicas.

En el desarrollo del proyecto, se identificaron ciertas limitaciones que influenciaron las decisiones tomadas durante la implementación. Una de las más relevantes fue la necesidad de descartar el uso del servicio en la nube Microsoft Azure, a pesar de ser una alternativa variable y competitiva. Esta decisión se tomó debido a la complejidad técnica asociada con su implementación y al poco conocimiento del equipo respecto a esta tecnología específica. Aunque Azure ofrece capacidades similares a AWS, los recursos necesarios para capacitar al equipo y adaptar la solución a esta plataforma hubieran requerido un tiempo y esfuerzo significativo, comprometiendo los plazos establecidos para el desarrollo del software.

A pesar de esta limitación, la elección de AWS como plataforma en la nube no solo permitió superar estas barreras, sino que también aportó una solución más alineada con los conocimientos y habilidades del equipo, asegurando un desarrollo eficiente y una implementación exitosa. Esta experiencia subraya la importancia de evaluar cuidadosamente las tecnologías disponibles y seleccionar aquellas que maximicen el potencial del equipo y del proyecto en su conjunto.

En resumen, el software Dinodock se presenta como una solución integral diseñada no solo para resolver problemas logísticos específicos, sino también para sentar las bases de una operación más ágil, eficiente y escalable en el manejo de contenedores. Su implementación en AWS, junto con su enfoque en optimización matemática y seguridad robusta, asegura que la solución estará preparada para enfrentar los retos actuales y futuros del sector logístico.

1.4 Objetivos Generales y Específicos

Objetivo del Proyecto

Desarrollar un software dirigido a empresas del sector logístico, enfocado en la optimización de los movimientos de (principalmente) contenedores. El objetivo es priorizar la rotación continua y eficiente de los contenedores, minimizando tanto los movimientos innecesarios como los tiempos de espera.

Objetivo de la Tesina

Desarrollar un software robusto y eficiente capaz de analizar y procesar la estructura lógica del patio, la demarcación de los bloques y la posición geográfica de los contenedores, computar complejas funciones algorítmicas de optimización que permitan calcular eficientemente el movimiento óptimo para el depósito y extracción de contenedores vacíos de la empresa.

Objetivos Específicos

- Diseñar una API para la obtención de datos del cliente (Sitrans).
- Codificar función que simula el ingreso de contenedores.
- Codificar función que simula la salida de contenedores.
- Codificar función algorítmica que permita posicionar un contenedor en la mejor ubicación, considerando, cantidad de movimientos para su posterior obtención y minimizar su impacto en la obtención de otros contenedores.
- Codificar función algorítmica que permita obtener un contenedor, que cumpla con los requerimientos del cliente, aumentando la rotación de los contenedores y minimizando su tiempo de estadía en el patio de contenedores.

Listado de Requerimientos Funcionales

Tabla 1. Requerimientos funcionales

Código requerimiento	Nombre	Descripción
RF-1	Diseñar una API	Consiste en la creación de una API que permita la comunicación entre el servidor del cliente y el sistema Dinodock.
RF-2	Codificar función ingreso	Implica el desarrollo de una función específica para gestionar el ingreso de contenedores al sistema Dinodock.
RF-3	Codificar función salida	Orientada a gestionar la extracción de contenedores del sistema según los requerimientos del cliente.
RF-4	Codificar función posicionar	Enfocada en la creación de una función que determine la posición específica de un contenedor dentro del sistema.
RF-5	Codificar función seleccionar	Identifica y propone el contenedor más adecuado según los parámetros establecidos por el cliente.

1.5 Justificación del Proyecto

Relevancia Técnica:

El desarrollo del back-end del software constituye uno de los pilares fundamentales de este proyecto, dado que aborda de manera directa el principal dolor planteado: la optimización del movimiento de contenedores en un patio delimitado. Este desafío va mucha más allá de los aspectos visuales de una Interfaz de Usuario (UI) o la Experiencia de Usuario (UX); se trata de un problema operativo que impacta significativamente en uno de los procesos más críticos para la empresa Sitrans: el correcto posicionamiento y la obtención eficiente de un contenedor que cumpla con los requisitos de cada cliente. Resolver este problema es esencial para garantizar la continuidad y la eficacia en la cadena logística de la compañía.

Por ello, el desarrollo eficiente y óptimo de los componentes back-end no solo es una prioridad, sino también el núcleo del éxito de este proyecto. Es en esta capa del sistema donde realmente se ataca el problema de raíz, utilizando herramientas avanzadas y técnicas de alto nivel. Mediante la implementación de funciones algorítmicas de optimización desarrolladas en Python, el back-end será capaz de analizar y procesar grandes volúmenes de datos en tiempo real, garantizando que las decisiones sobre el posicionamiento y selección de contenedores se realicen con la máxima eficiencia posible. Este enfoque permite modelar problemas complejos como la minimización del tiempo de espera, el uso óptimo del espacio y la asignación de recursos de manera dinámica y precisa.

Además, se establecerá una conexión segura, confiable y estable a través de una API, que servirá como el punto de enlace entre los distintos componentes del sistema y permitirá a los usuarios interactuar de manera fluida con la solución. Las respuestas, estructuradas en formatos JSON, facilitarán la integración y garantizarán que los datos procesados sean accesibles de manera clara y eficiente para los usuarios finales. Este diseño asegura que el sistema sea altamente adaptable y que las funciones críticas puedan ejecutarse sin interrupciones ni pérdida de información.

El dolor identificado también impacta directamente a los grueros, operarios especializados en el manejo de grúas, equipos de elevación y maquinaria pesada utilizada para mover, cargar, y descargar contenedores en el patio de almacenamiento. Los cuales, con la solución propuesta tendrán acceso a herramientas que les permitan realizar su trabajo de manera más eficaz y eficiente. Ese beneficio no solo mejora su productividad, sino que también contribuye a crear un entorno laboral más seguro y menos propenso a errores humanos.

En definitiva, el desarrollo back-end no es simplemente una parte técnica del proyecto, sino el motor central que permitirá resolver el problema de manera integral. Desde la implementación de algoritmos avanzados hasta la creación de infraestructuras seguras y escalables, cada componentes está diseñado para garantizar que el software no solo cumpla con las expectativas del cliente, sino que también transforme la manera en que se gestionan los contenedores en patio, optimizando procesos clave y aportando un valor único a la operación de Sitrans.

Beneficios:

Dentro de los beneficios esperados de la implementación del software Dinodock, se destacan una serie de ventajas que abarcan tanto la optimización operativa como la seguridad de los datos. Estos beneficios no solo buscan atender las necesidades específicas del cliente, sino que también establecer una base sólida para la sostenibilidad y crecimiento futuro del sistema. Los principales beneficios definidos son los siguientes:

1. **Agilizar las actividades de entrada y salida de contenedores dentro del patio**
   El software permitirá optimizar significativamente los procesos de ingreso y egreso de contenedores en el patio, que representan uno de los mayores retos operativos en la gestión. Al utilizar funciones avanzadas de optimización, el sistema será capaz de analizar, en tiempo real, el espacio disponible y las características específicas de cada contenedor para determinar la mejor posición en el patio. Esto reducirá el tiempo necesario para ubicar y retirar contenedores, disminuyendo los cuellos de botella y maximizando la productividad.

2. **Garantizar una alta seguridad en la transferencia y manipulación de los datos**
   La seguridad de los datos es un pilar fundamental en el diseño e implementación del sistema. Gracias a la infraestructura basada en Amazon Web Services (AWS), el software contará con mecanismos avanzados de protección, como la encriptación de datos en tránsito y en reposo, y la autenticación de usuarios mediante protocolos seguros. Esto asegura que los datos transferidos entre las APIs, sean inalterables y confidenciales, protegiendo la información crítica de posibles brechas de seguridad.

3. **Asegurar una alta capacidad de cómputo sin menoscabar la eficacia del software**
   El sistema estará diseñado para aprovechar al máximo la escalabilidad y flexibilidad de la arquitectura Serverless de AWS. Esto significa que el software será capaz de manejar grandes volúmenes de solicitudes simultaneas y procesar cálculos complejos, incluso durante picos de alta demanda, sin afectar su rendimiento. Al escalar automáticamente según las necesidades, el sistema garantiza que siempre habrá suficiente capacidad de cómputo para cumplir con los requisitos operativos, manteniendo tiempos de respuesta rápidos y una experiencia fluida para el usuario.

En base a lo anterior, contamos con los siguientes beneficios adicionales:

- **Reducción de costos operativos**: Al minimizar los movimientos necesarios y optimizar el uso del espacio y los recursos, el software reducirá significativamente los costos asociados con el manejo de contenedores.

- **Mejora en la toma de decisiones**: La disponibilidad de datos confiables y en tiempo real permitirá a los operadores tomar decisiones más informadas y estratégicas, mejorando la planificación y ejecución de las operaciones logísticas.

- **Incremento en la productividad personal**: Al proporcionar herramientas intuitivas y funcionales, el sistema reducirá la carga operativa y mejorará la eficiencia de los trabajadores, especialmente de los grueros, quienes tendrán instrucciones claras y optimizadas para realizar sus tareas.

2 Marco Teórico

2.1 Fundamentos del Desarrollo Back-End

El back-end es una de las áreas que componen el desarrollo de un software, aplicación móvil y/o aplicación web la cual se encarga del funcionamiento lógico del sistema a desarrollar, utilizando componentes tales como, Servidor, Base de Datos, APIs, Frameworks, etcétera.

Este enfoque de separar en dos áreas específicas el desarrollo de sistemas front-end y back-end permite a los programadores una mejor organización y escalabilidad. Esta medida, logra facilitar el desarrollo y mantenimiento de aplicaciones complejas o de gran envergadura, permitiendo una mayor distribución de trabajo en los equipos.

Según Tiwari (2023), la separación como tal del front-end y back-end surge a finales de la década de los 90s, cuando en 1996 la World Wide Web Consortium (W3C) presentó Cascading Style Sheets (CSS). Una plantilla de estilos para el desarrollo de aplicaciones web.

Según Tyson (2024), Como tendencias de la industria se encuentra principalmente:

- Integración de la Inteligencia Artificial (IA).
- Arquitectura sin Servidor (FaaS).
- Contenedores.

Cabe mencionar que los sistemas que en mayor medida utilizan back-end son:

- Aplicaciones web.
- Aplicaciones móviles.
- Internet de las Cosas (IoT).

Dentro de las ventajas se definen:

- Escalabilidad, que permite ajustarse a la demanda debido a su enfoque único.
- Mantenibilidad, debido a su encapsulación de la lógica.
- Reusabilidad, con APIs que permiten su utilización desde diversos orígenes.

Y como tal, hay desventajas que son importantes definir:

- Complejidad, grandes arquitecturas llegan a generar dificultad en su uso y mantención.
- Costos, implementar ciertas tecnologías puede resultar costoso en recursos y tiempo.
- Latencia, si no existe una correcta correlación entre front-end y back-end esto puede producir altos tiempos de espera en los usuarios.

Por esta razón, el rol del back-end en este software será de conexión entre cliente servidor mediante una API, almacenar y manipular datos y generar un entorno capaz de compilar funciones y algoritmos del lenguaje de programación Python.

2.2 Arquitectura de Software

Para el desarrollo de este proyecto, se optó por implementar una arquitectura **Serverless**, también conocida como **Function as a Service** (**FaaS**). Este enfoque se caracteriza por su dependencia de una infraestructura en la nube gestionada por un proveedor, lo que elimina la necesidad de configurar, administrar o escalar servidores por parte del equipo de desarrollo. Gracias a esta arquitectura, los desarrolladores pueden enfocarse exclusivamente en la programación y codificación de las funcionalidades especificas del sistema.

Una de las principales ventajas de la arquitectura Serverless es su capacidad para escalar automáticamente en función de la demanda. Esto significa que las funciones del sistema pueden manejar cargas variables de trabajo, desde volúmenes bajos hasta picos de tráfico elevados, sin necesidad de intervención manual. Además, este modelo sigue un esquema de pago por uso, lo que garantiza una mayor eficiencia en los costos, ya que se factura únicamente el tiempo de ejecución y los recursos consumidos por las funciones, en lugar de pagar por servidores inactivos o subutilizados.

Este enfoque también simplificó el proceso de desarrollo, ya que permitió que las funcionalidades del sistema se implementarán como microservicios. Cada función podía ser diseñada, implementada y probada de manera autónoma, lo que aceleró el desarrollo y facilitó la corrección de errores. Además, la modularidad inherente de este enfoque permitió una mayor flexibilidad para introducir nuevas características o realizar actualizaciones sin interrumpir el funcionamiento del sistema en su totalidad.

Desde un punto de vista operativo, la arquitectura mejoró la seguridad y confiabilidad del sistema, ya que el proveedor de la nube (AWS) garantiza medidas de seguridad avanzadas tales como; Cifrado de datos, Autentificación multi factor y Red privada virtual (VPC), redundancia y disponibilidad continua. Lo que permitió al equipo de desarrollo enfocarse en la funcionalidad y en la entrega de valor al cliente.

2.3 Tecnologías y Frameworks Utilizados

Las herramientas empleadas en este proyecto están basadas en una arquitectura robusta y moderna en la nube, específicamente mediante el uso de **Amazon Web Services** (**AWS**). Esta plataforma no solo proporciona una infraestructura altamente escalable y segura, sino que también ofrece un conjunto de servicios integrados que facilitan el desarrollo, la implementación y gestión de aplicaciones de software de manera eficiente. La elección de AWS responde a la necesidad de garantizar un rendimiento óptimo y una alta disponibilidad, aspectos críticos para el éxito de nuestro proyecto.

Uno de los componentes clave implementados en AWS son las APIs, que actúan como un puente entre el cliente y el servidor. Estas APIs permiten establecer una conexión estable, segura y de baja latencia, asegurando que los datos puedan ser transmitidos de manera eficiente entre ambas partes. Mediante el uso de servicios como AWS API Gateway, se facilita la gestión de las solicitudes entrantes y se mejora la experiencia del usuario al proporcionar tiempos de respuesta rápidos y confiables.

Además, AWS será responsable de la gestión de nuestro motor de Base de Datos. Para esto, se integrará un servicio como Amazon RDS (Relational Database Service), que ofrece un entorno seguro y optimizado para el almacenamiento, consulta y manipulación de datos. Este servicio se ajusta automáticamente a las necesidades del sistema, garantizando un rendimiento constante incluso en momentos de alta demanda.

Por otro lado, AWS proporcionará el entorno necesario para la compilación y ejecución del lenguaje de programación Python, que será la columna vertebral de los procesos de análisis y manipulación de datos. Python se empleará para analizar, limpiar y procesar los datos recibidos del cliente, asegurando que estén en un formato adecuado para ser utilizados en el software. Este flujo de trabajo será fundamental para garantizar que los datos sean precisos, relevantes y procesables, habilitando funcionalidades clave dentro del sistema.

En este contexto, se ha seleccionado un conjunto de librerías de Python que serán esenciales para llevar a cabo estas tareas. Estas librerías proporcionan herramientas avanzadas para operaciones matemáticas, visualización de datos, manejo de fechas, conexión a bases de datos y optimización matemática. A continuación, se describen brevemente:

- NumPy: Librería fundamental para realizar cálculos matemáticos y operaciones con matrices y arreglos, optimizando el manejo de datos numéricos de grandes volúmenes.

- Matplotlib: Utilizada para crear gráficos y visualizaciones que permitan interpretar de forma clara y visual los resultados que dependen de rangos temporales o análisis cronológicos.

- Datetime: Herramienta indispensable para manejar y manipular fechas y horas, especialmente relevante en funciones que dependen de rangos temporales o análisis cronológicos.

- Pyodbc: Librería para establecer conexiones con bases de datos mediante ODBC, facilitando la integración de nuestro sistema con la base de datos alojada en AWS.

- Pulp: Librería especializada en problemas de optimización, que permitirá modelar y resolver problemas matemáticos complejos relacionados con la funcionalidad del software.

3 Metodología

3.1 Enfoque para el Desarrollo de Back-End

Para la realización de este proyecto, se decidió implementar un enfoque ágil, dada la flexibilidad y adaptabilidad que ofrece para el desarrollo de software en equipo. Más específicamente, se optó por el modelo **Scrum**, una metodología ampliamente reconocida por su capacidad para facilitar el trabajo colaborativo, independiente y eficiente. Este enfoque no solo permitió una mejor gestión del tiempo y los recursos, sino que también fomentó un ambiente de trabajo dinámico y orientado a resultados concretos.

El marco Scrum fue diseñado en torno a **sprints** cortos, cada uno con una duración máxima de dos semanas. Esta selección estratégica permitió un ciclo de trabajo ágil y flexible, en el que cada sprint se enfocó en objetivos claros, específicos y de corto alcance. Estos objetivos no solo aseguraron un avance continuo en el desarrollo del proyecto, sino que también facilitaron la entrega incremental de componentes funcionales del sistema, alineándose con las prioridades establecidas al inicio del desarrollo.

Las reuniones **dailys**, programadas los lunes, miércoles y viernes jugaron un papel fundamental en el éxito del proyecto. Estas reuniones breves y estructuradas ofrecieron un espacio clave para la sincronización del equipo, en donde cada miembro tuvo la oportunidad de comunicar sus avances, problemas y soluciones. Este intercambio constante de información no solo agilizó la toma de decisiones, sino que también ayudó a identificar y resolver inconvenientes de forma temprana, evitando así posibles bloqueos o retrasos significativos.

Gracias a esta metodología y su enfoque colaborativo, el desarrollo del software **Dinodock** se completó de manera fluida y en línea con los plazos iniciales. La estructura de trabajo ágil permitió que el equipo mantuviera una visión clara de los objetivos a lo largo del proyecto, mientras se adaptaba rápidamente a los desafíos que surgieron. Los dos hitos clave definidos previamente fueron fundamentales para la consecución del proyecto, demostrando la efectividad del modelo Scrum y reforzando la importancia de una planificación adecuada y un seguimiento constante.

3.2 Herramientas y Entornos de Desarrollo Utilizados

Durante el desarrollo del proyecto, se emplearon tres plataformas para facilitar la planificación, comunicación y gestión del trabajo: **Trello**, **Discord** y **Github**. Estas herramientas combinadas con la metodología Scrum, formaron una base sólida para garantizar el flujo continuo de trabajo, la colaboración efectiva y el cumplimiento de los objetivos propuestos.

Trello se utilizó como el principal gestor de tareas, desempeñando un rol crucial en la planificación y organización de cada sprint. En esta plataforma, las actividades fueron segmentadas en tres hitos principales: **Pendiente**, **En proceso** y **Hecho**. Esta estructura permitió una visualización clara del estado de cada tarea, facilitando a todos los integrantes del equipo entender en qué fase se encontraba cada actividad en cualquier momento. Gracias a esta segmentación, se fomentó una comunicación fluida y se minimizó la duplicación de esfuerzos, aumentado significativamente la eficiencia.

Discord se convirtió en una herramienta esencial para las reuniones dailys, que fueron programadas tres veces a la semana con una duración máxima de quince minutos. Estas reuniones virtuales sirvieron como un espacio para que cada integrante compartiera sus avances de manera breve y concisa, mencionando las tareas completadas en los días previos, los objetivos a cumplir en los días siguientes, y cualquier obstáculo que pudiera estar interfiriendo con el progreso del sprint. Este enfoque promovió la resolución rápida de problemas, ya que se organizaban reuniones adicionales entre miembros clave cuando era necesario profundizar en la solución de los inconvenientes planteados. Además, durante las dailys se aprovechaba para informar cualquier cambio o ajuste efectuado en el tablero Trello, asegurando que todo el equipo estuviera alineado en cuanto a las actualizaciones más recientes.

Github, por su parte, fue la plataforma central para la gestión y almacenamiento del código del software Dinodock. Se estableció una organización en dos ramas principales: **dev** y **main**, cada una con roles específicos que garantizaban un flujo de trabajo ordenado y eficiente. La rama dev fue utilizada para los desarrollos en curso, caracterizándose por ser la más dinámica y donde se integraban los cambios más frecuentes a medida que los integrantes avanzaban en sus tareas. Esta rama permitió que el equipo colaborara en paralelo, asegurando que las nuevas características y correcciones se consolidaran de forma progresiva. En contraste, la rama main fue tratada como la versión estable del proyecto, sincronizándose únicamente con la rama dev tras completar una serie de pruebas exhaustivas. Estas pruebas garantizaban que las funcionalidades del software fueran completamente operativas antes de realizar cualquier integración, minimizando así el riesgo de introducir errores críticos en el producto final.

4 Diseño de Componentes

4.1 Arquitectura y Estructura de los Componentes

Para este proyecto se optó por una arquitectura en la nube provista por Amazon Web Services (AWS), que constituye una base sólida para garantizar el éxito del software Dinodock. AWS ofrece una infraestructura altamente adaptable, escalable y segura, características que son esenciales para el desarrollo de una solución con el potencial de ser implementada en múltiples sectores industriales y que pueda manejar eficientemente cargas de trabajo variables. Esta elección estratégica asegura que el sistema no solo cumpla con las necesidades actuales del proyecto, sino que también tenga la capacidad de evolucionar y expandirse conforme a las demandas futuras.

En línea con esta visión, la utilización del framework **Flask** es una pieza clave para garantizar la operatividad y eficiencia del código. Flask, conocido por su simplicidad y flexibilidad, permite desarrollar aplicaciones ligeras pero potentes, adaptadas a las necesidades específicas del sistema. Su implementación es fundamental para cumplir con el propósito central del software Dinodock: asignar la mejor posición para un contenedor entrante al depósito y, de manera inversa, seleccionar el contenedor más adecuado para un cliente con la menor cantidad posible de movimientos. Estas operaciones críticas serán gestionadas mediante solicitudes HTTP (**HTTP Requests**) y responderán en formato **JSON**, lo que asegura una integración fluida con otros sistemas y aplicaciones, además de una transferencia de datos eficiente y estructurada.

Dada la magnitud y complejidad del software, los componentes más relevantes para su desarrollo son, por un lado, la arquitectura en la nube proporcionada por AWS, que permite manejar la lógica del negocio y escalar según sea necesario, y por otro, la codificación de la solución misma, diseñada específicamente para resolver los problemas del cliente. Esta solución no solo busca atender las necesidades inmediatas, sino también generar un valor diferencial, ofreciendo una experiencia única y eficiente que posicione al software como un recurso esencial para los usuarios finales.

Un aspecto central de este proyecto es la implementación de funciones matemáticas avanzadas para abordar problemas de optimización, minimización y maximización de recursos. Estas técnicas son esenciales para calcular posiciones ideales dentro del depósito y diseñar rutas optimizadas que reduzcan al mínimo el esfuerzo operativo. Estas funciones permiten modelar escenarios complejos y encontrar soluciones que optimicen variables críticas, como el tiempo de operabilidad, la cantidad de movimientos y el uso del espacio disponible. Al integrar estas capacidades, el software no solo logra cumplir con las expectativas del cliente, sino que también eleva significativamente su eficiencia operativa.

La lógica matemática aplicada no es solo un recurso técnico, sino el corazón de la propuesta de valor del software. Las soluciones proporcionadas deben ser capaces de resolver problemas de forma precisa y dinámica, adaptándose a cambios en tiempo real y garantizando que cada decisión maximice los beneficios para el cliente. Este enfoque meticuloso asegura que el proyecto no solo solvente los problemas actuales, sino que también se convierta en una herramienta estratégica que impulse la productividad y competitividad de los usuarios en sus respectivas industrias.

Figura 1. Componentes principales de la solución

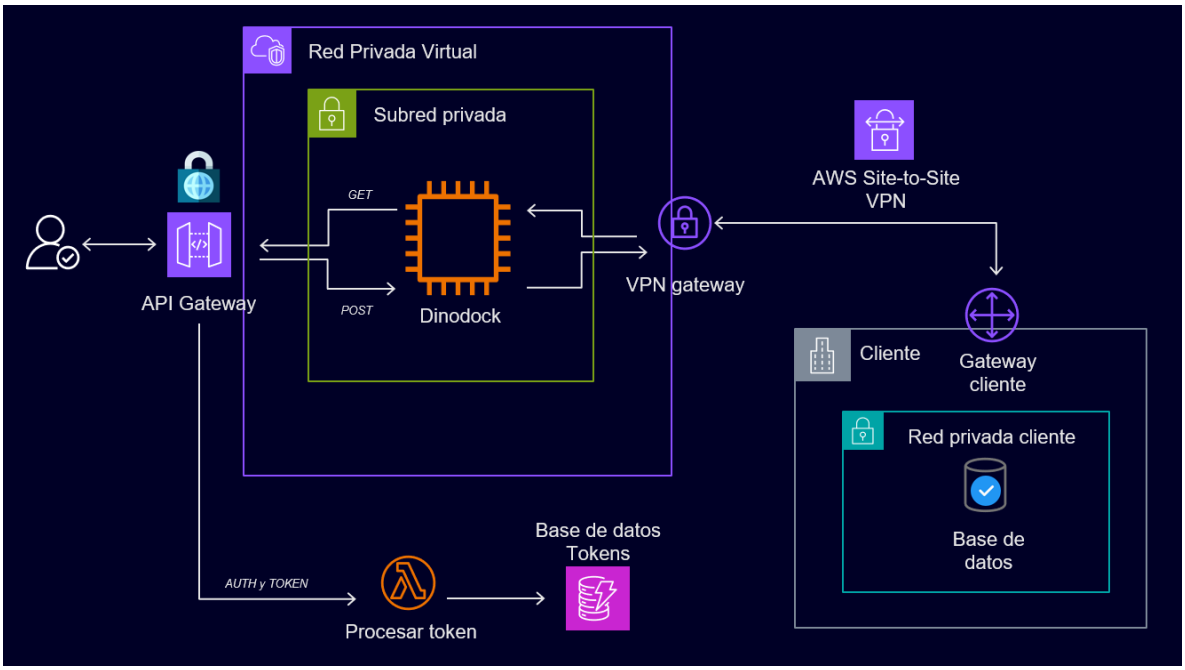

Figura 2. Arquitectura en Amazon Web Services (AWS)

4.2 Integración con el Sistema General

El desarrollo del software Dinodock se enfoca principalmente en la lógica de negocio y la optimización de procesos operativos, estableciendo una clara prioridad en la eficiencia y funcionalidad del sistema. Este enfoque lo posiciona en un plano más estrechamente relacionado con la infraestructura en la nube, dejando en un segundo plano el diseño o implementación de arquetipos visuales, como interfaces de usuario (UI). Esta decisión responde a la naturaleza misma del problema que busca resolver: la optimización de la logística y el manejo de contenedores, una problemática que demanda soluciones técnicas robustas más que una representación gráfica compleja.

La correcta comunicación ente la solución y los artefactos en la nube constituyen un aspecto crítico del proyecto, ya que la funcionalidad del software depende en gran medida de esta relación. Sin una integración adecuada entre las distintas partes del sistema, los usuarios finales no serían capaces de recibir respuestas claras y oportunas, lo que imposibilitaría el uso práctico de la solución. Esto hace que la integración entre los servicios back-end y la infraestructura en la nube no solo sea un componente técnico, sino también un requisito esencial para garantizar la experiencia del usuario y la funcionalidad del sistema.

Para abordar esta necesidad, se ha definido el servicio Flask, que será ejecutado dentro la infraestructura AWS. Flask es un framework ligero, eficiente y versátil que permite manejar solicitudes HTTP y generar respuestas precisas según las necesidades específicas del sistema. Con Flask, las peticiones enviadas al back-end serán procesadas de manera rápida y eficiente, generando las respuestas adecuadas para cada solicitud, ya sea el posicionamiento de un nuevo contenedor en el patio o la selección del contenedor óptimo para un cliente.

En línea con esta solución, se ha definido que todas las respuestas entregadas a los usuarios serán en formato JSON. Este formato es ampliamente reconocido por una estructura clara y concisa, lo que facilita la lectura y el procesamiento de los datos tanto por parte de los sistemas como por los usuarios humanos. El uso del formato JSON garantiza que la información proporcionada sea consistente y fácil de interpretar, independientemente del contexto o la plataforma que la reciba.

La comunicación entre nuestro servidor y el del cliente estará asegurada mediante una AWS VPN Sito-to-Site, un servicio que establece una conexión segura, fluida y estable entre la infraestructura del cliente y el sistema Dinodock. Esta VPN asegura que los datos transferidos entre ambas partes estén completamente encriptados, evitando accesos no autorizados y manteniendo la integridad y confidencialidad de la información. Este nivel de seguridad es especialmente crucial en un entorno donde se maneja datos sensibles y decisiones críticas, ya que garantiza que la información siempre llegue al destino correcto sin riesgos de pérdida o alteración.

Es importante destacar que, en este diseño, la integración del Back-End con el sistema solo puede realizarse de manera eficiente gracias a la arquitectura en la nube proporcionada por AWS. Este enfoque permite gestionar la lógica del sistema, las conexiones seguras y las respuestas rápidas de manera escalable y sin interrupciones. Dado que la funcionalidad del software no requiere una interfaz visual directa para los usuarios finales, esta arquitectura elimina la necesidad de construir elementos UI complejos y se centra exclusivamente en la optimización de procesos logísticos.

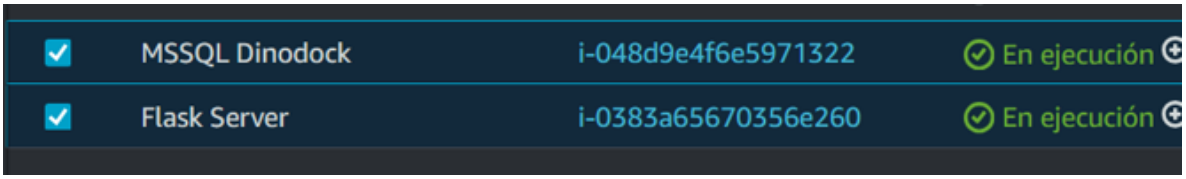

Figura 3. Documentación del correcto funcionamiento de componentes AWS

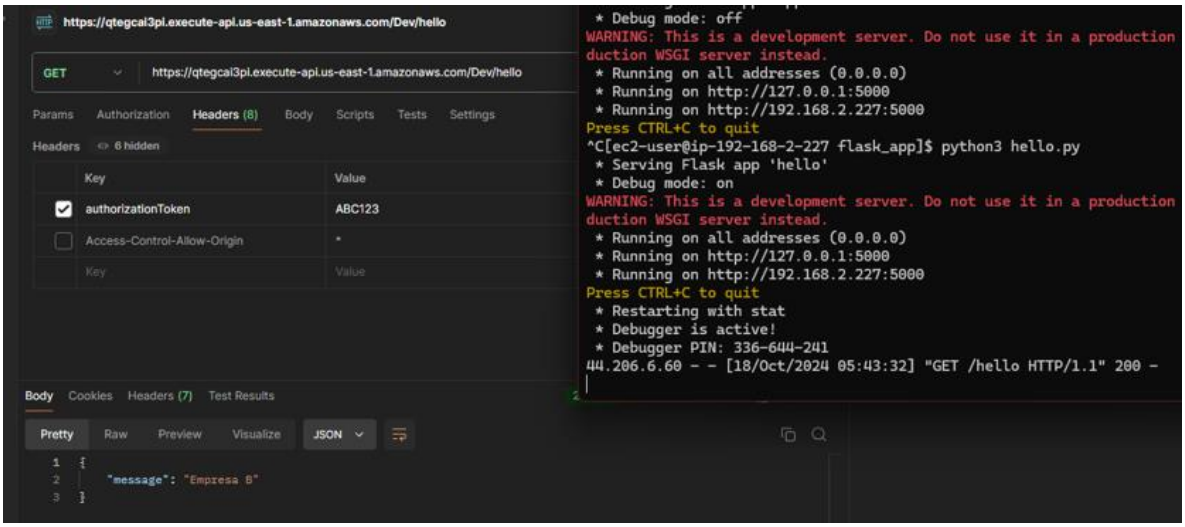

Figura 4. Llamado a la API mediante HTTP Requests

5 Implementación

5.1 Detalles de la Codificación y Desarrollo

Para el desarrollo de este proyecto, y apoyados en la metodología ágil Scrum, establecimos dos hitos claves que nos permitieron estructurar los objetivos tanto del proyecto general como de cada sprint en particular: el **Product Backlog** y el **Sprint Backlog**. Estos elementos fueron fundamentales para garantizar un desarrollo organizado, incremental y enfocado en la entrega de valor continuo.

El Product Backlog representó el eje central de la planificación del proyecto. Este documento recopiló todas las tareas, funcionalidades y requisitos necesarios para alcanzar el objetivo final del software. Aquí se priorizaron las funcionalidades clave en función del valor que aportarían al cliente, la complejidad técnica y la viabilidad dentro del plazo establecido. Cada elemento del backlog se formuló de manera clara y específica, asegurando que los objetivos fueran comprensibles para todos los miembros del equipo y alineados con las expectativas del cliente.

Por otro lado, el Sprint Backlog se derivó directamente del Product Backlog y se utilizó como gruía operativa durante cada sprint. Al inicio de cada iteración, el equipo seleccionó los elementos más prioritarios y alcanzables dentro del tiempo límite del sprint (dos semanas). Estos elementos fueron desglosados en tareas más pequeñas y específicas, asignadas a los miembros del equipo para garantizar su ejecución eficiente. Este enfoque permitió mantener un progreso constante, medir el avance de manera tangible y asegurar que cada sprint entregara un incremento funcional del producto.

La combinación del Product Backlog y el Sprint Backlog permitió al equipo trabajar de manera enfocada y colaborativa, priorizando siempre las tareas que tenían mayor impacto en los objetivos del proyecto. Además, estas herramientas proporcionaron un mecanismo para monitorear el progreso y hacer ajustes rápidos en caso de desviaciones o errores en el camino.

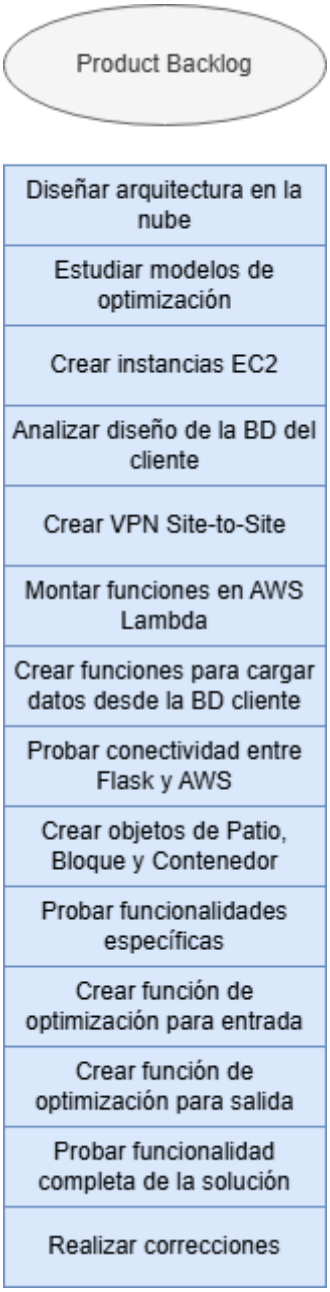

Figura 5. Product Backlog del proyecto

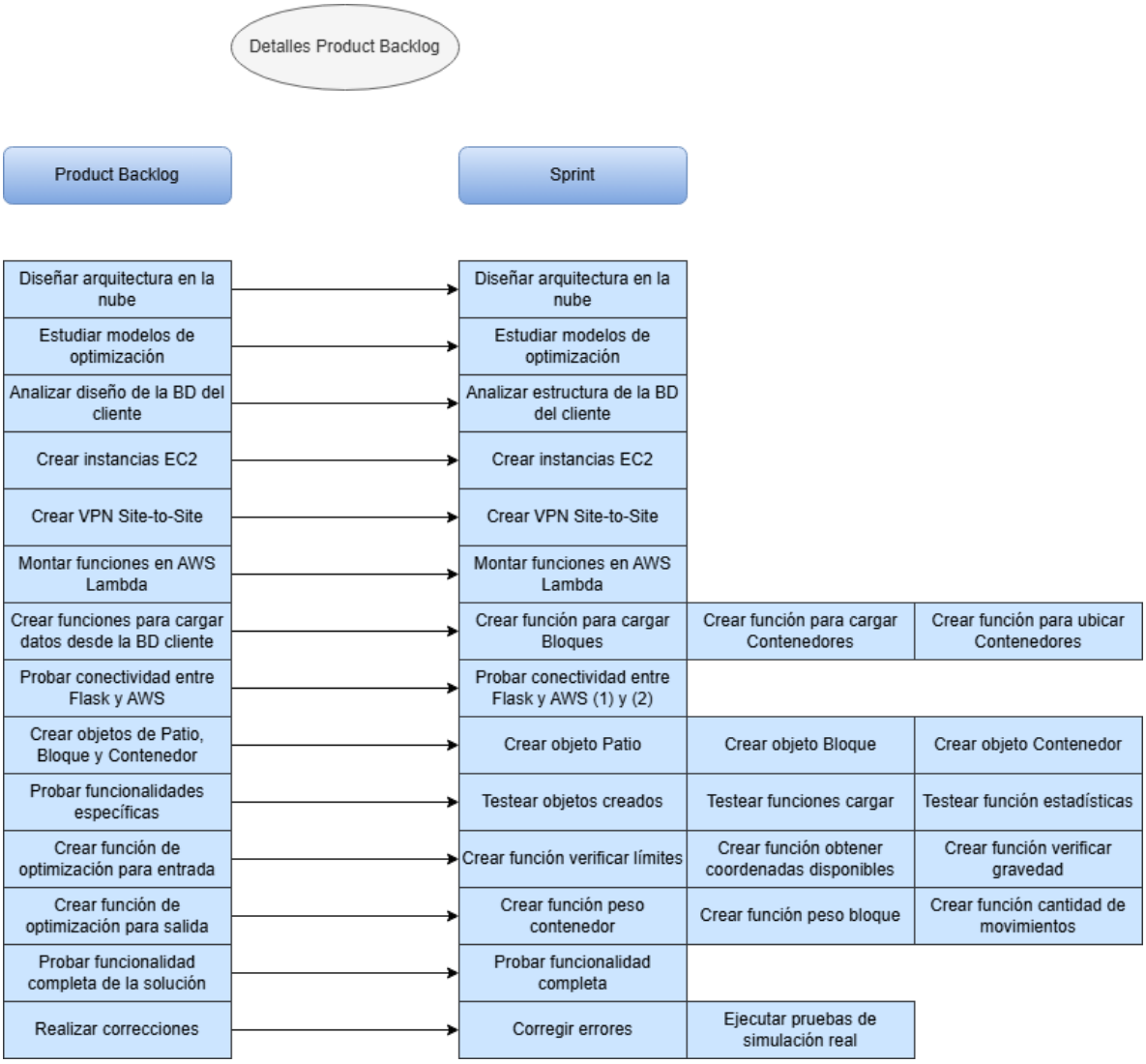

Figura 6. Product Backlog y su descomposición en Sprint Backlog

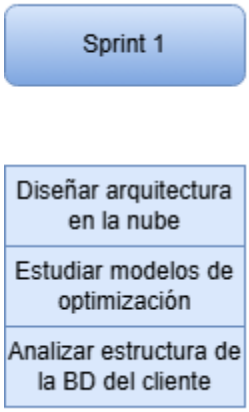

Figura 7. Backlog del Sprint 1

La **Figura 7** representa la planificación del Sprint 1. Este sprint se centra en tareas críticas que sentarán las bases para el desarrollo exitoso del proyecto. A continuación, se detalla cada una de las tareas indicadas en el sprint:

1. Diseñar arquitectura en la nube:
   Esta atarea tiene como objetivo definir la estructura tecnológica en la que se basará el sistema, utilizando servicios en la nube como AWS. Se trata de establecer los componentes principales, como servidores, APIs y seguridad, asegurando que la infraestructura sea escalable, segura y eficiente para soportar las operaciones del software Dinodock.

2. Estudiar modelos de optimización:
   En esta actividad, el equipo se dedicará a analizar y seleccionar los modelos matemáticos más adecuados para resolver problemas de optimización relacionados con el posicionamiento y la selección de contenedores.

3. Analizar estructura de la Base de Datos del cliente:
   Esta tarea consiste en revisar y comprender la estructura de la Base de Datos actual del cliente para identificar cómo integrarla con el sistema propuesto. El equipo evaluará los datos disponibles, su formato, las relaciones entre tablas y la forma de replicar dicha estructura y correlacionarla con el sistema Dinodock y sus funcionalidades.

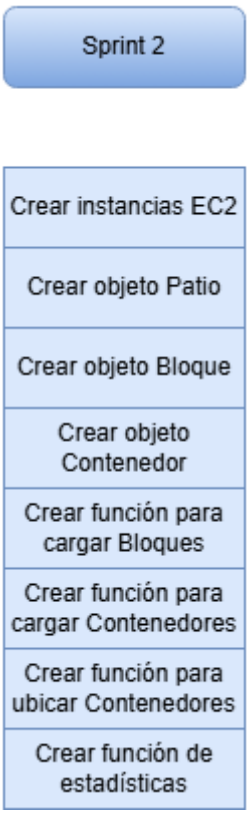

Figura 8. Backlog del Sprint 2

La **Figura 8** representa la planificación del Sprint 2, donde se especifican las tareas necesarias para avanzar en la implementación técnica de Dinodock. A continuación, se describe cada actividad:

1. Crear instancias EC2:
   Se configura y despliega una instancia de servidor virtual en AWS EC2, que será utilizada como el entorno principal para ejecutar el back-end del software. Esto incluye instalar las dependencias necesarias y asegurar la conectividad con la Base de Datos.

2. Crear objeto Patio:
   Se diseña y codifica la estructura del objeto "Patio", que representará el espacio físico donde se encuentran los Bloques. Este objeto solo incluye todos los bloques del espacio geográfico.

3. Crear objeto Bloque:
   Se desarrolla el objeto "Bloque", que representa las unidades organizativas dentro del patio. Este objeto incluye información como coordenadas, capacidad, estado, visibilidad, tamaños máximos permitidos y se relacionará directamente con los contenedores.

4. Crear objeto Contenedor:
   Se codifica el objeto "Contenedor", encargado de almacenar información relevante sobre cada contenedor, como tipo, tamaño, tiempo de estadía y naviera asociada. Este objeto será fundamental para las operaciones de optimización y logística.

5. Crear función para cargar Bloques:
   Desarrollo de una función que permita cargar la información de los bloques dentro del sistema, ya sea como un set de datos o como una consulta a la base de datos del cliente. Esto asegura que el software tenga acceso a la estructura actual del patio.

6. Crear función para cargar Contenedores:
   Se implementa una función para importar y registrar los contenedores existentes en el sistema. Esto incluye validar que la información sea correcta y que no haya datos faltantes.

7. Crear función para ubicar Contenedores:
   Desarrollo de una función algorítmica para asignar cada contendedor al bloque que pertenece y en la coordenada que pertenece.

8. Crear función de estadísticas:
   Se implementa una función para generar estadísticas clave sobre el uso del patio, como navieras más populares por mes y tipos de contenedores más llamados por mes.

Este Sprint se centra en la construcción de la lógica interna del software, asegurando que las funciones clave para la gestión de contenedores y bloques estén completamente operativas.

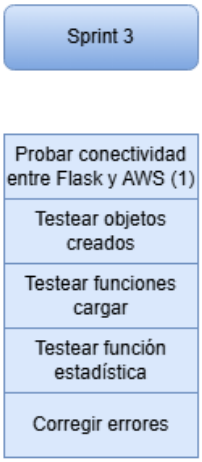

Figura 9. Backlog del Sprint 3

La **Figura 9** representa la planificación del Sprint 3. El cual está enfocado en la validación, pruebas y corrección de los componentes desarrollados en los sprints anteriores. A continuación, se detalla cada una de las actividades:

1. Probar conectividad entre Flask y AWS (1):
   En esta tarea se verifica que el framework Flask, utilizado para gestionar la API del sistema, se comunique correctamente con los servicios en la nube como AWS. Esto incluye comprobar la transmisión de datos desde y hacia las instancias EC2.

2. Testear objetos creados:
   Se prueban los objetos fundamentales del sistema, como Patio, Bloque y Contenedor, para confirmar que sus propiedades y relaciones funcionan según lo esperado. Esto asegura que los objetos estén correctamente definidos y preparados para interactuar con las funciones futuras.

3. Testear funciones cargar:
   Se evalúan las funciones desarrolladas para cargar bloques y contenedores, verificando que procesen correctamente los datos de entrada y que se comporten de manera adecuada en diferentes escenarios.

4. Testear función estadística:
   Se ejecuta la función encargada de generar estadísticas para evaluar su rendimiento y precisión. Se revisa que los datos generados reflejen correctamente el estado del patio, incluyendo métricas coherentes y acorde a los datos.

5. Corregir errores:
   A medida que se identifican problemas o inconsistencias durante las pruebas, esta actividad está orientada a solucionarlos. La corrección de errores garantiza que el sistema sea estable, confiable y funcional antes de avanzar al siguiente sprint.

El Sprint 3 es crucial para consolidar la calidad del sistema desarrollado. Al centrarse en pruebas e iteraciones correctivas, el equipo puede identificar y resolver cualquier problema antes de continuar con el desarrollo de nuevas funcionalidades.

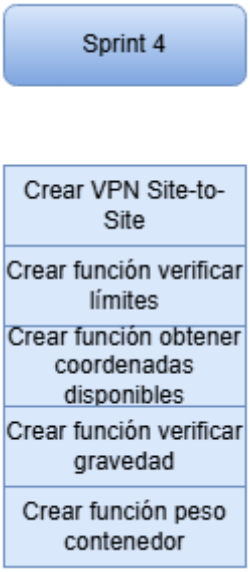

Figura 10. Backlog del Sprint 4

La **Figura 10** representa la planificación del Sprint 4. Este sprint se enfoca en el desarrollo de funcionalidades avanzadas y en la integración de componentes críticos para el correcto funcionamiento de Dinodock. A continuación, se describe cada tarea:

1. Crear VPN Site-to-Site:
   Esta tarea se centra en configurar una conexión VPN Site-to-Site entre el servidor del cliente y la arquitectura en la nube proporcionada por AWS.

2. Crear función verificar límites:
   Se implementa y codifica una función que evalúa si un bloque en el patio ha alcanzado sus límites de capacidad física y, por lo tanto, lógica.

3. Crear función obtener coordenadas disponibles:
   Esta función se encarga de identificar y devolver las coordenadas libres dentro de los bloques, donde se puedan ubicar nuevos contenedores o contenedores en reorganización.

4. Crear función verificar gravedad:
   Codifica una función que evalúa los efectos de que ningún contenedor esté flotando de forma indebida al momento de realizar operaciones de entrada o salida.

5. Crear función peso contenedor:
   Desarrollo de una función que calcule la cantidad de días que ha pasado el contenedor en el bloque, entregando un dato numérico.

Este Sprint 4 busca reforzar la lógica interna del sistema al agregar funcionalidades criticas relacionadas con la logística y seguridad.

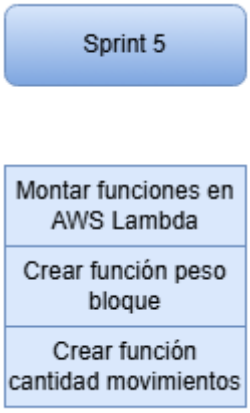

Figura 11. Backlog del Sprint 5

La **Figura 11** representa la planificación del Sprint 5. Este sprint se centra en el desarrollo de nuevas funciones relacionadas con el análisis del peso del bloque y la cantidad de movimientos de los contenedores para efectuar una salida. A continuación, se detalla cada actividad:

1. Montar funciones en AWS Lambda:
   Este paso implica desplegar todas las funciones clave del sistema en AWS Lambda. Esto permitirá que las funciones sean accesibles de forma ágil y escalable, reduciendo costos operativos al ejecutarlas solo cuando son requeridas.

2. Crear función peso bloque:
   Desarrollo de una función que calcule el peso total de cada bloque, basándose en la suma de los pesos individuales de los contenedores asignados a dicho bloque. Esta métrica permite identificar los bloques con contenedores que han estado durante mayor o menor tiempo en el patio.

3. Crear función cantidad movimientos:
   Implementa una función que calcula y registra la cantidad de movimientos necesarios para obtener un contenedor. Esta métrica permite notificar aquellos contenedores que son accesibles en mayor o menor medida dependiendo de la cantidad de operaciones a realizar.

El objetivo principal de este Sprint es consolidar las funcionalidades e integrarlas de forma eficiente con la arquitectura en la nube AWS.

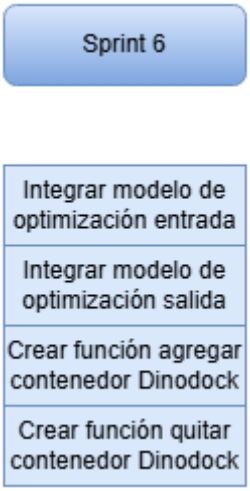

Figura 12. Backlog del Sprint 6

La **Figura 12** representa la planificación del Sprint 6. El cual marca una etapa clave en el desarrollo del software Dinodock, enfocándose en la integración de los modelos de optimización para las operaciones de entrada y salida de contenedores, así como en la creación de funciones específicas para la gestión directa de contenedores en el sistema. Estas tareas son esenciales para garantizar que el software cumpla con su objetivo principal: optimizar el almacenamiento y movimiento de contenedores en el depósito, minimizando los tiempos de espera y recursos utilizados. A continuación, se describen las tareas:

1. Integrar modelo de optimización entrada:
   Este modelo se encargará de determinar la mejor posición para un contenedor entrante al patio, utilizando métricas, tales como, peso bloque, marcas populares que salen por mes.

2. Integrar modelo de optimización salida:
   Este modelo está orientado a identificar el contenedor más adecuado para cumplir con las demandas de los clientes, minimizando la cantidad de movimientos necesarios para su extracción y maximizar el peso del contenedor.

3. Crear función agregar contenedor Dinodock:
   Desarrollo de una función que ejecuta el modelo de optimización para entrada y responde el bloque, junto a su coordenada óptima.

4. Crear función quitar contenedor Dinodock:
   Desarrollo de una función que ejecuta el modelo de optimización para salida y responde el contenedor idóneo, el bloque y la coordenada de ubicación.

El objetivo principal de este Sprint es consolidar las operaciones fundamentales del sistema Dinodock, integrando los modelos de optimización con las funciones de ingreso y egreso de contenedores.

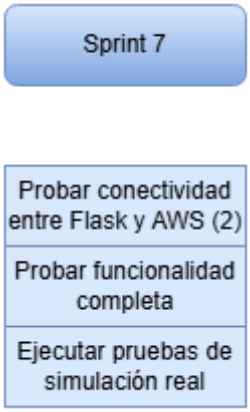

Figura 13. Backlog del Sprint 7

La **Figura 13** representa la planificación del Sprint 7. El cual representa una etapa crítica en el desarrollo del proyecto, donde se enfoca en probar la funcionalidad completa del sistema, asegurando que todos los componentes integrados operen de manera eficiente y sin errores. A continuación, se detallan las tareas:

1. Probar conectividad entre Flask y AWS:
   Esta tarea busca garantizar que el framework Flask, encargado de gestionar las solicitudes y respuestas de la API, se comunique correctamente con los servicios de AWS.

2. Probar funcionalidad completa:
   Tarea en la que se realiza una prueba integral del sistema para confirmar que todas las funciones, modelos y componentes trabajan de manera conjunta y sin problemas.

3. Ejecutar pruebas de simulación real:
   Paso que busca replicar escenarios reales simulados de operación, utilizando datos de entrada similares a los esperados y dictaminados por el cliente.

Este Sprint marca el último hito, donde el objetivo principal es garantizar que el software esté listo para su implementación y uso en un entorno real. Las pruebas realizadas durante este sprint no solo validan la funcionalidad técnica del sistema, sino que también aseguran que el software cumple con los estándares de calidad, escalabilidad y seguridad requeridos.

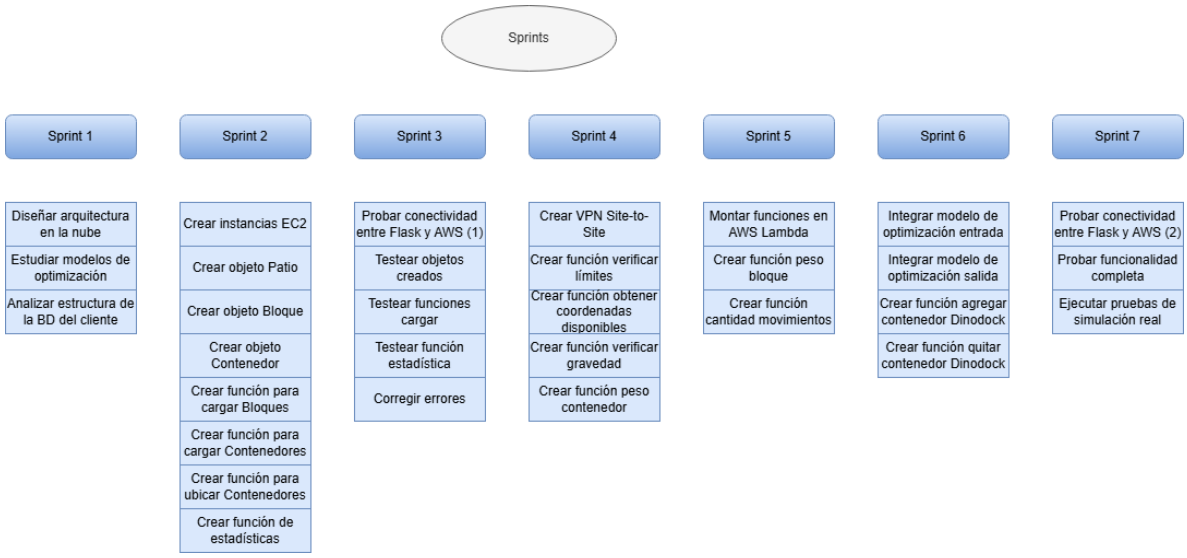

Figura 14. Todos los Sprints ordenados

Figura 15. Diagrama de clase Dinodock

La **Figura 15** representa el diagrama de clase del software Dinodock. Ilustra las relaciones entre las tres clases principales: Patio, Bloque y Contenedor. Estas clases son los pilares fundamentales del modelo de datos del software, cada una con propiedades y funciones específicas para representar el flujo lógico del almacenamiento y manipulación de contenedores en un entorno controlado.

Este diseño planteado promueve la organización jerárquica y modular del sistema, asegurando la escalabilidad y claridad en la gestión de los datos. Además, el uso de identificadores únicos y propiedades específicas permite una trazabilidad eficiente y un control detallado sobre cada nivel del modelo.

5.2 Uso de Buenas Prácticas y Patrones de Diseño

Para el desarrollo del software Dinodock, se establecieron estándares y prácticas que garantizaron la claridad, organización y eficiencia de la programación del sistema. Entre las principales decisiones tomadas, destaca la utilización del estilo de escritura **Snake Case** para el nombramiento de todas las funciones, asegurando un formato uniforme y legible en el código. Este estilo, caracterizado por separar palabras con guiones bajos y escribir en minúsculas, facilitó tanto la colaboración entre desarrolladores como la posterior revisión del código.

Además, se definieron clases únicas y bien estructurada para modelar los objetos fundamentales del sistema: Patio, Bloque y Contenedor. Estas clases encapsulan las propiedades y comportamientos necesarios para representar las operaciones clave del software, asegurando que cada entidad sea independiente y clara en su propósito. Por ejemplo, la clase Patio representa el espacio geográfico de almacenamiento, mientras que Bloque gestiona segmentaciones especificas dentro de ese espacio, y Contenedor almacena las características propias de cada unidad. Este enfoque orientado a objetos no solo favorece la escalabilidad del sistema, sino también su mantenibilidad.

Para fomentar el buen desempeño del equipo y garantizar un desarrollo eficiente, se adoptaron cinco prácticas clave:

1. Modularidad de las funciones:
   Cada función del software fue diseñada para realizar una tarea específica y bien delimitada, lo que evita redundancias y facilita su reutilización en diferentes partes del sistema. Por ejemplo, la función encargada de verificar los límites de un bloque fue independiente de la que calcula el peso total de los contenedores, permitiendo combinarlas según las necesidades del sistema. Esta modularidad también facilita la detección y corrección de errores, ya que las funciones aisladas son más fáciles de probar y depurar.

2. Uso de un IDE común:
   Se optó por un IDE (Entorno de Desarrollo Integrado) común para todos los desarrolladores, lo que garantizó la consistencia en las herramientas utilizadas y mejoró la colaboración. En este caso, el IDE seleccionado fue Visual Studio Code (VSC), debido a su capacidad para trabajar con múltiples lenguajes, y siendo uno de ellos Python, sus funciones avanzadas para el análisis de código y su facilidad en el uso.
   Este enfoque permitió al equipo compartir configuraciones, utilizar plantillas comunes y acceder a funcionalidades como el autocompletado y la detección temprana de errores.

3. Testeo regular por terceros:
   Una parte fundamental del proceso fue la realización de pruebas por parte de integrantes del equipo que no estaban directamente involucrados en la programación. Este testeo externo proporcionó una perspectiva objetiva y ayudó a identificar problemas o inconsistencias en el sistema que podrían pasar desapercibidas para los desarrolladores principales.
   Además, estas pruebas se realizaron de manera periódica, coincidiendo con el cierre de cada sprint, asegurando así que el software cumpliera con los requisitos funcionales antes de avanzar al siguiente ciclo de desarrollo.

4. Uso de una metodología ágil (Scrum):
   El equipo adoptó una metodología ágil, utilizando el marco de trabajo Srum, para abordar el desarrollo del proyecto. Este enfoque permitió dividir el trabajo en sprints, con objetivos específicos y alcanzables a corto plazo.
   La metodología ágil permitió a los desarrolladores adaptarse rápidamente a los cambios y hacer ajustes sobre la marcha, asegurando que las entregas fueran consistentes y alineadas con los objetivos iniciales.

5. Establecer Sprints definidos y abordables:
   Cada sprint fue cuidadosamente planificado para abordar tareas que fueran alcanzables dentro del tiempo estimado, permitiendo que el equipo pudiera mantener un flujo constante de trabajo sin sobrecargas ni tareas inalcanzables.

6 Pruebas y Validación

6.1 Estrategias de Testing Aplicadas a los Componentes

Para validar y testear la efectividad de las funciones desarrolladas en el software, se ha propuesto implementar un enfoque basado en pruebas unitarias y pruebas específicas por función. Las pruebas unitarias serán fundamentales para verificar que cada componentes individual del sistema funcione según lo esperado, comprobando la exactitud de los cálculos, operaciones lógicas y flujos de datos. Estas pruebas se enfocarán en asegurar que todas las entradas produzcan los resultados esperados en cada función, considerando tanto escenarios estándar como casos límite.

Además, la implementación de pruebas específicas por funcionalidad permitirá evaluar el comportamiento de aquellas funciones que desempeñan roles críticos dentro del sistema, como la optimización de entrada y salida de contenedores, la selección de estos y generación de estadísticas. Estas pruebas no solo medirán el desempeño técnico de las funciones, sino también su alineación con los objetivos planteados para el software.

Por otro lado, siguiendo las buenas prácticas del desarrollo de software, se ha incorporado el testeo por terceros, que consiste en que miembros del equipo no involucrados directamente en la codificación revisen y prueben el sistema. Este enfoque tiene múltiples beneficios: permite identificar problemas desde una perspectiva más objetiva, fomenta la colaboración interdisciplinaria y asegura que los errores se detecten de una manera temprana, reduciendo el impacto en las etapas finales del desarrollo.

Finalmente, todas estas pruebas se realizarán de forma periódica y alineadas con los objetivos de cada sprint, asegurando así una revisión continua del progreso. Este método iterativo no solo valida la calidad del código desarrollado, sino que también fortalece la confiabilidad del sistema y garantiza que las soluciones entregadas cumplan con los estándares de funcionalidad y precisión esperados.

6.2 Resolución de Bugs y Optimización

Para la resolución de bugs se optó por emplear un enfoque basado en ensayo y error, el cual resulta particularmente útil en proyectos donde las funciones poseen una alta interdependencia y complejidad lógica. Este método consiste en realizar pequeñas pruebas controladas en segmentos específicos del código, evaluando minuciosamente su comportamiento y los resultados obtenidos en relación con las respuestas esperadas.

El proceso inicia identificando los posibles puntos de falla mediante un análisis detallado del flujo del programa y los datos que maneja. Posteriormente, se realizan modificaciones graduales y controladas en el código afectado, validando la efectividad de los cambios en cada iteración. Esta estrategia permite no solo corregir el error identificado, sino también entender su causa raíz, lo que ayuda a prevenir futuros problemas similares.

Por último, para maximizar la eficacia del proceso, se integró con las pruebas unitarias y el testeo por terceros, permitiendo validar rápidamente si los ajustes realizados impactan de manera positiva en el sistema sin introducir nuevo errores. Este enfoque iterativo asegura que los bugs se aborden de manera eficiente, manteniendo la calidad y estabilidad del software durante todo el ciclo de desarrollo.

7 Integración con Otros Componentes

7.1 APIs y Servicios Utilizados

En el caso de este proyecto, el apartado front-end no resulta necesario ni imperativo debido a las características particulares y el enfoque central de la solución. Este proyecto está principalmente orientado a resolver un problema lógico mediante la optimización de procesos y la integración eficiente de la solución en un entorno basado en servicios en la nube. Al priorizar la funcionalidad del back-end y la conexión con herramientas y servicios alojados en la nube, como AWS, el proyecto no requiere una interfaz gráfica para la interacción directa con los usuarios finales.

La naturaleza del sistema, que se centra en manejar datos, ejecutar algoritmos complejos de optimización y proporcionar respuestas precisas en formato JSON, asegura que las necesidades del cliente y los operadores del sistema sean satisfechas sin la necesidad de una capa visual o una experiencia de usuario sofisticada. En lugar de eso, las interacciones ocurren mediante solicitudes y respuestas HTTP, aprovechando la arquitectura API-first que facilita la integración del sistema con otros servicios o herramientas que el cliente pudiera necesitar.

Este enfoque minimalista en cuanto al front-end no solo reduce los costos de desarrollo y mantenimiento, sino que también permite al equipo concentrar todos los recursos en garantizar la eficacia, estabilidad y escalabilidad del sistema back-end. Además, al estar diseñado para integrarse directamente con los flujos operativos del cliente, el sistema puede ser gestionado mediante herramientas existentes o interfaces externas, en caso de que sea necesario.

8 Resultados

8.1 Entrada

```
Ingreso de contenedor:
 Contenedor(ID: 001, Tipo: Seco, Tamaño: 20, Marca: Evergreen, Fecha Ingreso: 2023-01-07 13:16:00, Estado: 1)
Salida esperada (Bloque, coordenada_x, coordenada_y, coordenada_z)
('A', 0, 0, 2)
PS C:\Users\Alexander\Documents\DinoDock_V0.1>
```

Figura 16. Respuesta al ingreso de un contenedor en Dinodock

Para el ingreso de un contenedor, el sistema lleva a cabo un conjunto complejo de procesos, que incluyen cálculos matemáticos, análisis estadísticos y operaciones lógicas, con el objetivo de determinar la mejor posición posible para dicho contenedor dentro del depósito. Este análisis considera múltiples factores esenciales como la marca, el tipo y el tamaño del contenedor, así como las características específicas del espacio disponible en el patio.

Además, el sistema realiza una verificación exhaustiva de todas las limitaciones y restricciones definidas previamente, tales como la capacidad máxima de los bloques y las condiciones de peso y estabilidad. Este paso es fundamental para garantizar que la solución proporcionada no solo sea eficiente desde el punto de vista del almacenamiento, sino que también cumpla con los requisitos de seguridad.

El sistema utiliza modelos de optimización avanzados, los cuales consideran tanto las características individuales del contenedor como el contexto general del patio, incluyendo los movimientos necesarios para ubicar el contenedor en su posición final. Por ejemplo, prioriza ubicaciones que minimicen los movimientos de otros contenedores y que faciliten su acceso futuro, reduciendo los tiempo y costos operativos.

Finalmente, una vez realizado todo el análisis, el sistema entrega al cliente una solución clara y precisa a través de una respuesta en formato JSON, que detalla la posición óptima y las acciones necesarias para su ubicación. Este flujo automatizado y robusto permite que los operadores en terreno puedan llevar a cabo las tareas de manera más eficiente, asegurando que el ingreso de cada contenedor sea lo más rápido y efectivo posible.

8.2 Salida

```
Egreso de contenedor con los siguientes parámetros:
Marca: Evergreen
Tipo: Seco
Tamaño: 20
Salida esperada (Contenedor elegido, Bloque, coordenada_x, coordenada_y, coordenada_z)
(Contenedor(ID: 21, Tipo: Refrigerado, Tamaño: 20, Marca: Evergreen, Fecha Ingreso: 2024-08-12 09:00:00, Estado: 1), 'A', 2, 4, 0)
PS C:\Users\Alexander\Documents\DinoDock_V0.1>
```

Figura 17. Respuesta a la salida de un contenedor en Dinodock

Para la salida de un contenedor, el sistema ejecuta un proceso altamente optimizado que comienza con la búsqueda de todos los contenedores almacenados que cumplan con las características específicas solicitadas por el cliente, como la marca, el tipo y el tamaño. Esta esta primera etapa garantiza que únicamente se consideren los contenedores relevantes, acotando el universo de opciones disponibles y agilizando los cálculos subsiguientes.

Una vez identificados los contenedores que cumplen con los criterios solicitados, el sistema realiza un análisis exhaustivo de cada uno en términos de dos factores calve: los días que el contenedor ha permanecido en el depósito y la cantidad de movimientos necesarios para acceder a él. Este análisis tiene como objetivo encontrar la mejor solución posible desde el punto de vista tanto operativo como logístico.

En el proceso de optimización, el sistema aplica un enfoque dual. Por un lado, maximiza los días en el depósito, priorizando la salida de aquellos contenedores que han permanecido más tiempo almacenados, lo cual permite mantener un flujo eficiente de rotación y liberar espacio para nuevos ingresos. Por otro lado, minimiza la cantidad de movimientos necesarios para extraer el contenedor solicitado, reduciendo los tiempos de operación, el uso de recursos como combustible y maquinaria, y los costos asociados.

Finalmente, el resultado del análisis es presentado al cliente a través de una respuesta estructurada en formato JSON, que detalla el contenedor seleccionado y su ubicación exacta para su extracción. Este enfoque garantiza que el proceso de salida sea ágil, preciso y eficiente, mejorando la experiencia del cliente y optimizando los recursos del depósito.

9 Conclusiones

Durante el desarrollo de este proyecto, uno de los factores de aprendizaje más significativos fue la utilización de clases en el lenguaje de programación Python para replicar el diseño de las tablas de la Base de Datos del cliente. Esta práctica nos permitió no solo simular con precisión su entorno de trabajo, sino también adaptarnos a sus necesidades específicas y comprender el flujo de sus datos de manera más profunda. Gracias a este enfoque, el equipo logró un entendimiento más claro de las relaciones entre los datos y las operaciones necesarias para optimizar el proceso lógico.

El tiempo total que tomó el desarrollo del proyecto fue de aproximadamente de 15 semanas, distribuidas en sprints claramente definidos y abordables. Durante este período, se investigaron diversas herramientas y tecnologías que pudieran facilitar el desarrollo, destacándose el uso de la librería **Pyodbc**. Esta librería fue clave para establecer la conexión con la Base de Datos, permitiendo ejecutar consultar y obtener los datos necesarios para alimentar el sistema. Esta decisión no solo agilizó el proceso, sino que también aseguró la compatibilidad con el entorno del cliente.

Considerando el desarrollo del proyecto, una de las decisiones más difíciles y que más impacto tuvo fue el retraso en asumir que los datos necesarios para simular y testear Dinodock nunca llegarían por parte del patrocinador. Inicialmente, se había planificado todo el desarrollo contando con la idea de que, una vez finalizado, los datos proporcionados por el cliente permitirían probar la solución en condiciones reales. Sin embargo, la falta de estos datos generó incertidumbre en los resultados obtenidos, que en muchos casos resultaron ambiguos y alejados de las expectativas reales.

Mi formación profesional fue de gran ayuda para enfrentar este desafío. Desde los conocimientos técnicos necesarios para desarrollar y codificar las funciones del sistema hasta las habilidades blandas para trabajar en equipo y gestionar situaciones complejas, la formación adquirida a lo largo de estos años resultó invaluable. Esta base me permitió aplicar de manera efectiva las buenas prácticas de desarrollo, así como gestionar los imprevistos y mantener el enfoque en los objetivos del proyecto.

Entre las lecciones más destacadas que dejó este proyecto, está la comprensión de que la interacción con los stakeholders no siempre es fluida o bidireccional. En este caso, las expectativas iniciales no se alinearon con la realidad de las interacciones, lo que dificultó la recopilación de información clave. Esta experiencia refuerza la necesidad de estar preparados para gestionar situaciones complejas e imprevistas, así como trabajar con diferentes estilos de comunicación y colaboración.

En cuanto a los logros alcanzados, se destaca la creación y configuración de variables de peso que fueron fundamentales para la toma de decisiones dentro del sistema. La correcta codificación y utilización de estas variables permitió estructurar la fórmula de optimización, otorgándole dirección y sentido, asegurando que el sistema aumentara o disminuyera los valores adecuados en las operaciones lógicas. Este hito representa un avance significativo en la funcionalidad del software y su capacidad para adaptarse a diferentes escenarios.

De cara al futuro, uno de los aspectos que sería crucial incorporar en próximas iteraciones es la integración de estadísticas basadas en datos históricos. Esto no solo enriquecería el sistema con más variables para la toma de decisiones, sino que también permitiría identificar patrones y tendencias que optimicen aún más el proceso de ingreso y salida de contenedores. Con esta integración, el sistema podría ser aún más preciso al determinar la ubicación óptima para u contenedor entrante o al seleccionar el mejor contenedor para su salida, teniendo en cuenta factores históricos relevantes.

Una posible proyección para este software sería replicar dicho sistema en otros giros de logística, ayudando a las empresas a lograr una mejor gestión de sus productos y cómo los organizan en sus depósitos. La capacidad de optimizar procesos, minimizar tiempos de espera y maximizar el uso del espacio lo convierte en una herramienta adaptable a múltiples industrias que enfrentan retos similares en la gestión de inventarios y almacenamiento.

En conclusión, este proyecto dejó una combinación de aprendizajes, desafíos y logros que refuerzan la importancia de la adaptabilidad, la toma decisiones temprana y la necesidad de contar con estrategias alternativas para mitigar riesgos e imprevistos. La experiencia adquirida no solo fortaleció nuestras habilidades técnicas, sino también nuestra capacidad para resolver problemas, gestionar expectativas y proyectar soluciones con un impacto significativo en el sector logístico.

Se entrega un especial agradecimiento a toda mi familia por apoyarme durante todos estos años, en los buenos y malos momentos, siempre a mi lado. Sobre todo, quiero dar total crédito de este momento a mi madre y abuela, quienes ininterrumpidamente me dieron ánimo y confort en aquellos malos y peores momentos. Gracias a ustedes, logro esta nueva hazaña. Este es mi regalo para ustedes. Espero poder darles muchos más.

Referencias

[1] Tiwari, A. (2023, 19 de Agosto). *The History and Evolution of Web Development: From HTML to the Modern Web*. Medium. https://medium.com/@Abhishek-Tiwari/the-history-and-evolution-of-web-development-from-html-to-the-modern-web-982e3f90e891

[2] Tyson, L. (2024, 2 de enero). *Exploring Back-End Web Development Trends For 2024*. Medium. https://medium.com/@lenaztyson/exploring-back-end-web-development-trends-for-2024-e06082ed3d74

[3] Amazon Web Services. (n.d.). *The Difference Between Frontend and Backend*. https://aws.amazon.com/es/compare/the-difference-between-frontend-and-backend/#:~:text=El%20back%2Dend%20son%20los,las%20aplicaciones%20para%20los%20usuarios.