UNIVERSIDAD COOPERATIVA DE COLOMBIA - FACULTAD DE INGENIERIA - PROGRAMA DE INGENIERIA DE SOFTWARE

API DE GESTIÓN DE ESPACIOS FÍSICOS CON IA

- Luis Cerón
- Miguel Cajigas
- Jhonatan Diaz

19-11-2025

INTRODUCCIÓN

El presente trabajo de investigación se enfoca en el desarrollo de una API inteligente para la gestión eficiente de espacios físicos mediante la aplicación de inteligencia artificial. En un contexto donde la optimización de recursos es cada vez más relevante, este sistema busca mejorar significativamente la distribución y asignación de recursos en entornos de alta densidad como estacionamientos, oficinas y centros comerciales.

La propuesta surge como respuesta a los problemas de congestión y uso ineficiente de espacios físicos, proporcionando una solución tecnológica que no solo organiza el espacio disponible sino que además aprende y se adapta a los patrones de uso para ofrecer una experiencia mejorada tanto para usuarios como para administradores de estos espacios.

1.PROBLEMA DE LA INVESTIGACIÓN

1.1. OBJETO O TEMA DE INVESTIGACIÓN

El objeto de investigación es el desarrollo e implementación de una API basada en inteligencia artificial para la optimización de la gestión de espacios físicos en entornos de alta concurrencia y densidad, como estacionamientos, áreas de oficinas y centros comerciales.

1.2. ÁREA DE INVESTIGACIÓN:

- Desarrollo de software

1.3. LÍNEA DE INVESTIGACIÓN

La línea de investigación corresponde a la aplicación de inteligencia artificial en la gestión de recursos físicos, específicamente en el desarrollo de algoritmos adaptativos para la optimización de espacios y la mejora de experiencias de usuario en entornos de alta densidad.

1.4. PLANTEAMIENTO DEL PROBLEMA

En la actualidad, la gestión de espacios físicos en entornos de alta concurrencia enfrenta diversos desafíos que afectan tanto a usuarios como a administradores:

- **Síntomas**: Se observan espacios físicos congestionados, largos tiempos de espera para encontrar ubicaciones disponibles, frustración en los usuarios y aumento en los costos operativos por la ineficiente distribución de recursos.
- **Causas**: La principal causa identificada es la falta de herramientas tecnológicas que permitan gestionar eficientemente estos espacios, resultando en una asignación inadecuada de recursos.
- **Consecuencias**: La ineficiencia en la gestión de espacios genera pérdidas económicas por subutilización de recursos, deterioro en la experiencia de los usuarios y aumento en los costos operativos.

- **Pronóstico**: Sin una solución adecuada, se espera que estos problemas se intensifiquen a medida que aumente la densidad poblacional en entornos urbanos y comerciales.

1.5. FORMULACIÓN DEL PROBLEMA

¿De qué manera un API inteligente basado en IA puede optimizar la gestión de espacios físicos y mejorar la asignación de recursos en entornos de alta densidad como estacionamientos, oficinas y centros comerciales?

1.6. OBJETIVOS DE LA INVESTIGACIÓN

1.6.1. Objetivo general

Desarrollar un sistema API basado en inteligencia artificial que optimice la gestión y asignación de espacios físicos en entornos congestionados, mejorando la eficiencia operativa y la experiencia de los usuarios.

1.6.2. Objetivos específicos

- Levantar requerimientos para el desarrollo de un sistema que facilite la asignación de recursos, espacios e infraestructura en general para pequeños y medianos entornos.

- Analizar y diseñar una solución tecnológica para la asignación de recursos, espacios e infraestructura en general para pequeños y medianos negocios.
- Desarrollar y codificar la aplicación que responda a los requerimientos y análisis previamente planteados para esta problemática.
- Prototipar la solución de software y aplicar las respectivas pruebas que garanticen la adecuada funcionalidad visto desde la concepción de un proyecto mínimo viable.

1.7. JUSTIFICACIÓN

Este proyecto se justifica desde diversas perspectivas:

Justificación tecnológica: El desarrollo de esta API permitirá aplicar técnicas avanzadas de inteligencia artificial para resolver problemas cotidianos, consolidando conocimientos en desarrollo de software y machine learning.

Justificación práctica: La solución aborda problemas reales y frecuentes en entornos urbanos y comerciales, con aplicaciones directas en estacionamientos, oficinas, hospitales y centros comerciales.

Justificación económica: La optimización en la gestión de espacios físicos tiene un impacto directo en la reducción de costos operativos y el aumento de la eficiencia en el uso de recursos disponibles.

Justificación social: Mejora la experiencia del usuario final al reducir tiempos de espera, disminuir la frustración y proporcionar una mejor organización de los espacios compartidos.

Justificación académica: Contribuye al campo de la inteligencia artificial aplicada, ofreciendo nuevos enfoques y modelos para la optimización de recursos en contextos urbanos y comerciales.

1.8. DELIMITACIÓN

Delimitación espacial: El sistema se implementará inicialmente en entornos pequeños de alta densidad como estacionamientos, oficinas y centros comerciales en áreas enfocadas a una población reducida pero mal organizada.

Delimitación temporal: El proyecto tiene una duración estimada de 12 meses, comenzando en el segundo semestre de 2024, dividido en fases de planificación, desarrollo, pruebas e implementación.

Delimitación técnica:

- El sistema será open-source y funcionará en sistemas operativos Linux, Windows y macOS.
- Se utilizarán frameworks y tecnologías como Spring Boot, TensorFlow, PostgreSQL con PostGIS, y servicios cloud de AWS.
- La API estará diseñada siguiendo principios RESTful con autenticación OAuth 2.0 y JWT.

2. MARCO TEÓRICO

Con el propósito de dar contexto a las temáticas relacionadas con el presente proyecto, a continuación, se traen antecedentes alusivos

2.1. ANTECEDENTES

La gestión de espacios físicos ha evolucionado significativamente en los últimos años, pasando de sistemas manuales a soluciones tecnológicas cada vez más sofisticadas:

Sistemas tradicionales: Históricamente, la gestión de espacios físicos se realizaba manualmente o mediante sistemas rudimentarios de tickets y asignaciones fijas, con escasa capacidad de adaptación.

Evolución digital: Con el avance tecnológico, surgieron sistemas digitales de reservas y asignación de espacios, aunque con limitada capacidad de optimización y adaptación.

Sistemas actuales: En la actualidad existen soluciones basadas en sensores IoT y software de gestión que proporcionan información en tiempo real, pero con limitada capacidad predictiva y adaptativa.

Estado actual de la tecnología: Las soluciones más avanzadas están comenzando a integrar inteligencia artificial para la predicción de patrones y optimización de recursos, aunque aún existe un amplio margen de mejora en términos de eficiencia y personalización.

Ejemplos puntuales:

- **Cobot**: Plataforma de gestión para espacios de coworking que ayuda con reservas, facturación y membresías.
- **Nexodus**: Proyecto de red segura de código abierto desarrollado por Netmaker que facilita conexiones seguras entre dispositivos y redes.
- **Spaceti**: Solución inteligente para edificios que mejora la gestión del espacio, la ocupación y el bienestar mediante sensores y análisis de datos.

Conforme a las anteriores referencias lo que se pretende es exponer que el presente proyecto contiene un incremento o elemento innovador debido a su enfoque adaptativo a varios modelos de negocios, necesidades del usuario y su propuesta open source.

2.2. TEORÍAS GENERALES DEL PROYECTO

El presente proyecto, que se enfoca en la gestión de espacios físicos mediante el uso de inteligencia artificial (IA), se apoya en diversas teorías y enfoques técnicos para garantizar la optimización y efectividad del sistema desarrollado. A continuación, se detallan algunas de las teorías fundamentales que sustentan este trabajo, proporcionando una base teórica sólida que respalda la utilización de IA en la asignación, distribución y gestión eficiente de los espacios físicos.

2.2.1 Teoría de colas

La **Teoría de colas** es fundamental para entender los patrones de llegada y servicio en espacios físicos congestionados, así como para modelar cómo los usuarios o recursos interactúan dentro de un sistema de servicio. Esta teoría se basa en la matemática de las colas, donde se estudian los tiempos de espera, el número de clientes (o usuarios) en espera, y el rendimiento de un servicio dado bajo condiciones de alta demanda. Aplicada a la gestión de espacios físicos, la teoría de colas permite optimizar la asignación de recursos en función de la demanda esperada y las tasas de ocupación de los espacios. Además, se puede utilizar para modelar las fluctuaciones de la demanda en tiempo real, permitiendo ajustar dinámicamente los recursos disponibles, como el número de personas o el espacio que debe estar disponible en un momento dado.

Por ejemplo, en un sistema de oficinas o en una universidad, los usuarios (empleados o estudiantes) podrían generar una "cola" al intentar acceder a un espacio común o reservado, como una sala de conferencias o una oficina de trabajo. La teoría de colas, combinada con la IA, puede predecir los picos de demanda y sugerir modificaciones en la asignación de espacios en tiempo real para reducir los tiempos de espera y aumentar la eficiencia del uso del espacio.

Referencias:

- **Gross, D., & Harris, C. M. (1998)**. *"Fundamentals of Queueing Theory"*. John Wiley & Sons.

- **Kleinrock, L. (1975)**. *"Queueing Systems, Volume 1: Theory"*. Wiley-Interscience.

2.2.2 Aprendizaje Automático y Redes Neuronales

El **aprendizaje automático** (Machine Learning, ML) y las **redes neuronales** constituyen los cimientos para el desarrollo de algoritmos adaptativos que permitan a un sistema aprender de patrones históricos y mejorar continuamente sus predicciones y asignaciones. En el contexto de la gestión de espacios físicos, el uso de ML permite predecir la ocupación de los espacios, la demanda de los usuarios y la eficiencia en el uso del espacio según variables como el tiempo, el tipo de evento, la disponibilidad de recursos y las preferencias de los usuarios.

Las **redes neuronales**, en particular, son una herramienta poderosa dentro del aprendizaje automático, ya que pueden identificar patrones complejos en grandes volúmenes de datos. Este tipo de modelo puede entrenarse con datos históricos sobre el uso de espacios físicos, para luego predecir con alta precisión cuándo y dónde se necesitarán más recursos, como sillas, mesas, o incluso personal para gestionar un espacio.

En este sentido, el sistema no solo responde a eventos puntuales, sino que puede adaptarse a los cambios en los hábitos de uso del espacio a lo largo del tiempo. De esta manera, las redes neuronales permiten al sistema "aprender" cómo optimizar el uso del espacio en función de una serie de factores dinámicos, mejorando continuamente su eficiencia sin intervención humana constante.

Referencias:

- **Jordan, M. I., & Mitchell, T. M. (2015)**. "Machine Learning: Trends, Perspectives, and Prospects". *Science*, 349(6245), 255-260.

- **LeCun, Y., Bengio, Y., & Hinton, G. (2015)**. "Deep learning". *Nature*, 521(7553), 436-444.

2.2.3 Teoría de la Optimización

La **teoría de la optimización** se refiere a un conjunto de principios matemáticos y algoritmos que permiten resolver problemas de asignación óptima de recursos en un entorno con múltiples variables y restricciones. En el contexto de la gestión de espacios físicos, los algoritmos de optimización son esenciales para asignar recursos de manera eficiente, teniendo en cuenta limitaciones como la capacidad de los espacios, el tiempo disponible, las necesidades de los usuarios, y otras restricciones operativas.

Los métodos de optimización permiten resolver problemas complejos como la programación de horarios, la asignación de salas para reuniones o eventos, y la redistribución de espacios en función de la demanda. Los enfoques de **optimización combinatoria**, como los algoritmos de programación lineal y la programación entera, se pueden aplicar para encontrar soluciones que maximicen la eficiencia en el uso del espacio, minimizando el desperdicio y los costos asociados.

En combinación con la inteligencia artificial, estos métodos pueden ser adaptativos, ajustándose a las fluctuaciones en la demanda y las condiciones cambiantes de los espacios. Esto permite que el sistema no solo asigne recursos de manera eficiente en el presente, sino que también anticipe futuras necesidades y ajuste los recursos de manera proactiva.

Referencias:

- **Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. (2010)**. *"Linear Programming and Network Flows"*. Wiley.

- **Boyd, S., & Vandenberghe, L. (2004)**. *"Convex Optimization"*. Cambridge University Press.

2.2.4 React Front

- **React** es una biblioteca de desarrollo para aplicaciones web que se ha destacado por su flexibilidad, escalabilidad y eficiencia. Esta biblioteca es especialmente relevante en el contexto de este proyecto debido a su capacidad para crear interfaces de usuario (front-end) interactivas y altamente reactivas. React permite desarrollar aplicaciones web dinámicas y de alto rendimiento, fundamentales para la interacción con el sistema de gestión de espacios físicos.
- React es ideal para este proyecto por su enfoque basado en componentes, lo que facilita la reutilización, organización e integración de distintas partes de la interfaz de usuario, así como su conexión con las APIs del backend. Además,

React ofrece un modelo de actualización eficiente del DOM a través del Virtual DOM, lo que asegura un rendimiento óptimo incluso en aplicaciones con interacciones frecuentes, como la gestión dinámica de espacios físicos.

- Meta Platforms, Inc. (2021) destaca la importancia de React por su arquitectura centrada en componentes reutilizables y su enfoque declarativo, lo que permite implementar nuevas funcionalidades o actualizar las existentes de manera sencilla y sin comprometer el rendimiento de la aplicación.
- **Referencia:**

  Meta Platforms, Inc. (2021). "React – A JavaScript library for building user interfaces". Recuperado de https://reactjs.org.

2.2.5 Arquitectura de Microservicios

La **arquitectura de microservicios** es un enfoque arquitectónico que promueve la descomposición de aplicaciones monolíticas en servicios independientes, pequeños y autónomos, los cuales se comunican entre sí a través de interfaces bien definidas, generalmente usando APIs REST. Esta arquitectura favorece la escalabilidad, la flexibilidad y el mantenimiento del sistema, ya que cada microservicio puede evolucionar y actualizarse de forma independiente.

En el contexto de la gestión de espacios físicos, una arquitectura de microservicios permite separar las distintas funcionalidades del sistema (como la gestión de la ocupación de los espacios, la predicción de la demanda, la gestión de usuarios, etc.) en

servicios independientes, lo que facilita la implementación de nuevas características y la mejora continua del sistema sin afectar su estabilidad global.

Además, la arquitectura de microservicios permite que cada componente del sistema se escale de manera independiente, lo que resulta en una mayor eficiencia en el uso de los recursos de infraestructura, optimizando el rendimiento general del sistema.

Referencias:

- **Newman, S. (2015)**. *"Building Microservices"*. O'Reilly Media.

- **Lewis, J., & Fowler, M. (2014)**. "Microservices". *ThoughtWorks*.

2.2.6 REST (Representational State Transfer)

REST es un estilo arquitectónico que se utiliza para el diseño de APIs web que facilitan la interoperabilidad entre diferentes sistemas. Este enfoque se basa en la idea de que los recursos deben ser representados mediante URIs (identificadores de recursos uniformes) y que las operaciones sobre estos recursos se deben realizar a través de los métodos HTTP estándar (GET, POST, PUT, DELETE).

En la gestión de espacios físicos, REST permite la comunicación fluida entre los distintos servicios del sistema. A través de APIs RESTful, el sistema puede acceder a datos sobre la ocupación de espacios, la disponibilidad de recursos y otras variables relevantes para la gestión, sin depender de tecnologías propietarias o difíciles de integrar con otros sistemas. REST es una opción natural para la comunicación entre el

front-end y el back-end en sistemas distribuidos, y permite una fácil escalabilidad y mantenimiento del sistema.

Referencias:

- **Fielding, R. T. (2000)**. "Architectural Styles and the Design of Network-based Software Architectures". Doctoral Dissertation, University of California, Irvine.

- **Richardson, L., & Ruby, S. (2007)**. *"RESTful Web Services"*. O'Reilly Media.

Las anteriores conceptualizaciones son el insumo requerido para el entendimiento del proyecto a aplicar con el software que queremos desarrollar para poder mejorar la organización.

2.3. FORMULACIÓN DE HIPÓTESIS

Hipótesis principal: La implementación de un API basado en inteligencia artificial mejorará significativamente la eficiencia en la gestión de espacios físicos, reduciendo los tiempos de espera en al menos un 30% y optimizando la utilización del espacio en un 25%.

Hipótesis secundarias:

1. El uso de algoritmos de machine learning permitirá identificar patrones de uso que no son evidentes mediante análisis manual, mejorando la precisión de las asignaciones.
2. La categorización dinámica de recursos y usuarios basada en inteligencia artificial resultará en una mayor satisfacción del usuario comparada con los sistemas tradicionales de asignación.
3. La implementación de un sistema adaptativo de gestión de espacios reducirá los costos operativos asociados a la subutilización de recursos en al menos un 20%.
4. La solución basada en API facilitará la integración con sistemas existentes, reduciendo el tiempo de implementación en nuevos entornos en comparación con soluciones monolíticas.

3. METODOLOGÍA DE DESARROLLO DE SOFTWARE

Para el desarrollo del proyecto se adoptará una metodología ágil basada en Scrum complementada con elementos de Kanban, permitiendo un enfoque iterativo, adaptativo y visual del desarrollo del software.

De acuerdo, reorganizando la información para tener los puntos 3.1 y 3.2 como solicitaste:

3.1. MARCO DE TRABAJO SCRUM

Roles:

- **Product Owner:** Responsable de definir y priorizar los requisitos del producto, gestionar el backlog y validar los entregables.
- **Scrum Master:** Facilita el proceso Scrum, elimina impedimentos y asegura que el equipo siga las prácticas acordadas.
- **Equipo de desarrollo:** Equipo multifuncional compuesto por desarrolladores backend, especialistas en IA, ingenieros QA y DevOps.

Eventos:

- **Sprint:** Iteraciones de 2 semanas donde se desarrolla un incremento potencialmente entregable del producto.
- **Daily Scrum:** Reunión diaria de 15 minutos para sincronizar actividades y planificar el trabajo del día.
- **Sprint Planning:** Reunión al inicio de cada sprint para planificar el trabajo a realizar.

- **Sprint Review:** Demostración de los resultados del sprint a los stakeholders.
- **Sprint Retrospective:** Análisis del proceso para identificar mejoras para el siguiente sprint.

Artefactos:

- **Product Backlog:** Lista priorizada de todas las funcionalidades requeridas.
- **Sprint Backlog:** Selección de elementos del Product Backlog a implementar en el sprint actual.
- **Incremento:** Resultado del sprint, que debe estar en condición "hecho" y ser potencialmente entregable.

Implementación de Sprints:

- **Organización:** En sprints de 2 semanas.
- **Planificación:** Detallada de cada sprint con estimaciones en puntos de historia.
- **Ceremonias:** De Scrum adaptadas al proyecto.

3.2. COMPLEMENTO CON KANBAN

Gestión del Tablero Kanban:

- **Columnas del tablero:** Backlog, Por hacer, En desarrollo, En revisión, Pruebas, Hecho.
- **Límites WIP por columna:** Para optimizar el flujo de trabajo.
- **Políticas explícitas:** Para el movimiento de tareas entre columnas.

Tablero Kanban:

- **Visualización del flujo de trabajo:** Con columnas: Por hacer, En progreso, En revisión, Hecho.
- **Límites de trabajo en curso (WIP):** Para cada columna para evitar sobrecarga y cuellos de botella.
- **Rastreo visual:** Del progreso y los impedimentos.

Métricas Kanban:

- **Lead time:** Tiempo desde que se identifica un requisito hasta que se entrega.
- **Cycle time:** Tiempo desde que comienza el trabajo en un requisito hasta que se completa.
- **Throughput:** Cantidad de trabajo completado por unidad de tiempo.

Métricas y Seguimiento:

- **Burndown charts:** Para seguimiento de sprints.
- **Métricas de flujo Kanban:** Lead time, cycle time y throughput.
- **Dashboards:** Para visualización de progreso y calidad.

3.3. FASES DE DESARROLLO

Fase 1: Investigación y Planificación Inicial (2 meses)

- Investigación preliminar y revisión bibliográfica
- Definición detallada de requisitos
- Diseño conceptual y arquitectura inicial
- Preparación del entorno de desarrollo

Fase 2: Desarrollo Prototipo Inicial (3 meses)

- Configuración de infraestructura
- Desarrollo de componentes base del API
- Implementación de primeros algoritmos de IA
- Integración de componentes y pruebas iniciales

Fase 3: Desarrollo y Optimización (4 meses)

- Mejora de algoritmos de IA y machine learning
- Optimización de rendimiento
- Expansión de funcionalidades
- Implementación de seguridad y pruebas avanzadas

Fase 4: Implementación y Lanzamiento (2 meses)

- Preparación de entorno de producción

- Despliegue inicial y monitoreo
- Fase piloto y recolección de métricas
- Expansión gradual a diferentes casos de uso

Fase 5: Cierre y Evaluación (1 mes)

- Evaluación final de rendimiento
- Documentación completa
- Entrega final y planificación de mantenimiento

3.4. PRÁCTICAS DE INGENIERÍA DE SOFTWARE

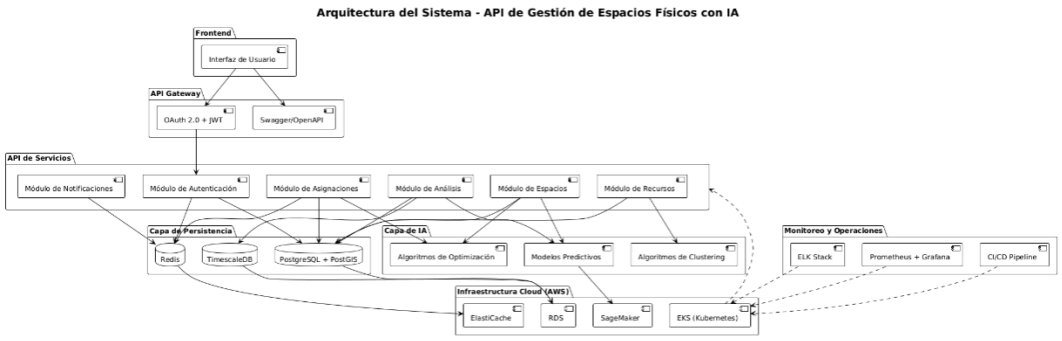
- **Integración Continua/Entrega Continua (CI/CD)**: Automatización de pruebas y despliegues.
- **Control de versiones**: Uso de Git con flujos de trabajo basados en ramas.
- **Revisión de código**: Evaluación por pares antes de la integración.
- **Pruebas automatizadas**: Unitarias, integración y rendimiento.
- **Desarrollo basado en pruebas (TDD)**: Para componentes críticos.
- **Documentación continua**: Actualización de documentación técnica y de usuario.

4. RESULTADOS DE LA INVESTIGACIÓN

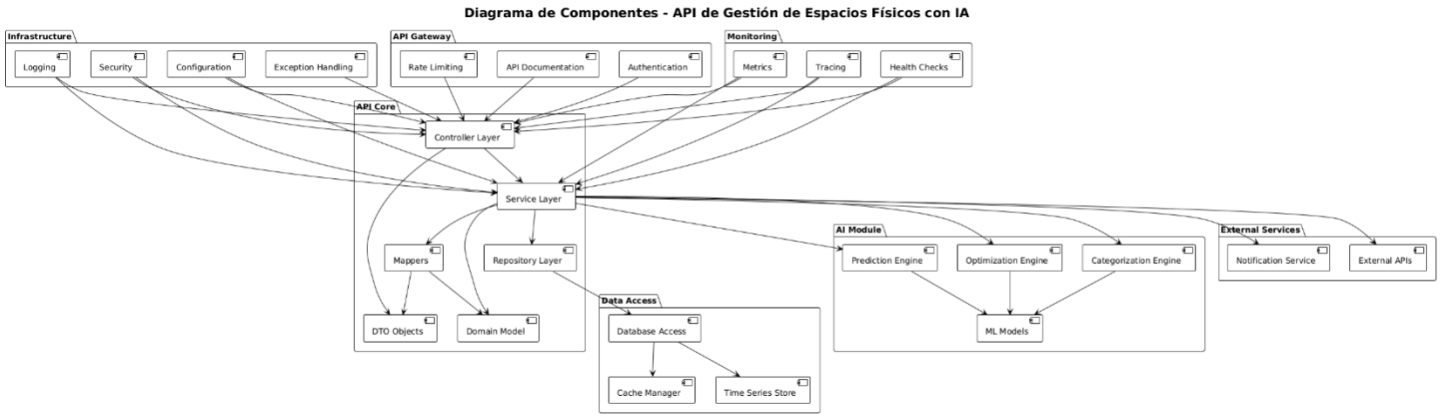
Sistema de gestión de espacios físicos basado en una arquitectura modular que integra una API RESTful, algoritmos de inteligencia artificial, bases de datos relacionales y servicios de soporte para la asignación, optimización y monitoreo de recursos en entornos compartidos. Incluye funcionalidades para análisis predictivo, notificaciones, y cumplimiento de requisitos funcionales y no funcionales.

4.1. DISEÑO DE LA ARQUITECTURA DEL SISTEMA

Título: Diagrama de Arquitectura del Sistema API de Gestión de Espacios Físicos con IA



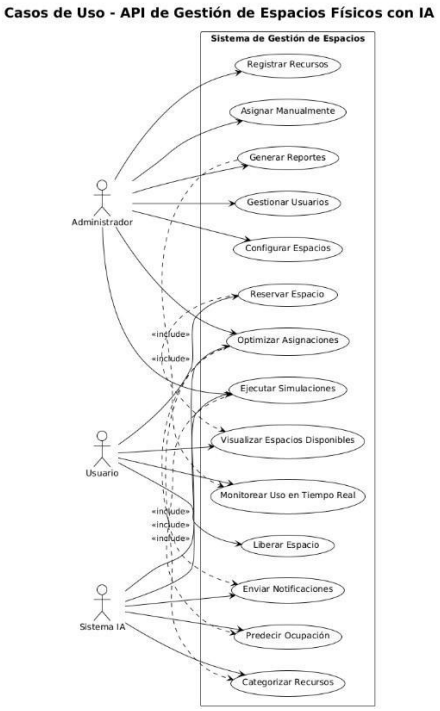
Descripción: Este diagrama ilustra la arquitectura general del sistema, mostrando los principales componentes, sus interrelaciones y el flujo de datos entre ellos. Se destaca la capa de servicios API, los módulos de IA, la infraestructura de almacenamiento y las interfaces externas con los sistemas cliente.



4.1.1. Componentes Principales

- **Capa de Presentación**: Interfaces de usuario, aplicaciones cliente y sistemas externos que consumen la API.
- **Capa de API**: Core del sistema que expone los endpoints RESTful y gestiona las solicitudes.
- **Capa de Servicios**: Lógica de negocio para la gestión de espacios y recursos.
- **Módulo de IA**: Algoritmos de machine learning para predicción y optimización.
- **Capa de Datos**: Almacenamiento y gestión de información persistente.
- **Servicios de Soporte**: Autenticación, logging, monitoreo y seguridad.

4.2. ESPECIFICACIÓN DE REQUISITO



4.2.1. Requisitos Funcionales

1. **Gestión de Espacios**:
    - El sistema debe permitir crear, modificar, eliminar y consultar espacios físicos.
    - Cada espacio debe tener atributos como tipo, capacidad, ubicación y características específicas.
    - El sistema debe permitir buscar espacios disponibles según criterios específicos.

2. **Gestión de Recursos**:
    - El sistema debe permitir registrar diferentes tipos de recursos (vehículos, personas, equipos).
    - Cada recurso debe poder categorizarse según características relevantes.
    - El sistema debe mantener el historial de asignaciones de cada recurso.
3. **Asignación y Optimización**:
    - El sistema debe asignar automáticamente recursos a espacios según algoritmos de optimización.
    - El sistema debe permitir reservas anticipadas y asignaciones en tiempo real.
    - El algoritmo debe adaptarse a patrones cambiantes de uso.
4. **Análisis y Reportes**:
    - El sistema debe generar análisis de uso y eficiencia de los espacios.
    - El sistema debe ofrecer predicciones de ocupación para períodos futuros.
    - Los reportes deben ser personalizables según las necesidades del usuario.
5. **Notificaciones**:
    - El sistema debe enviar notificaciones sobre asignaciones, cambios y eventos relevantes.
    - Los usuarios deben poder configurar sus preferencias de notificación.

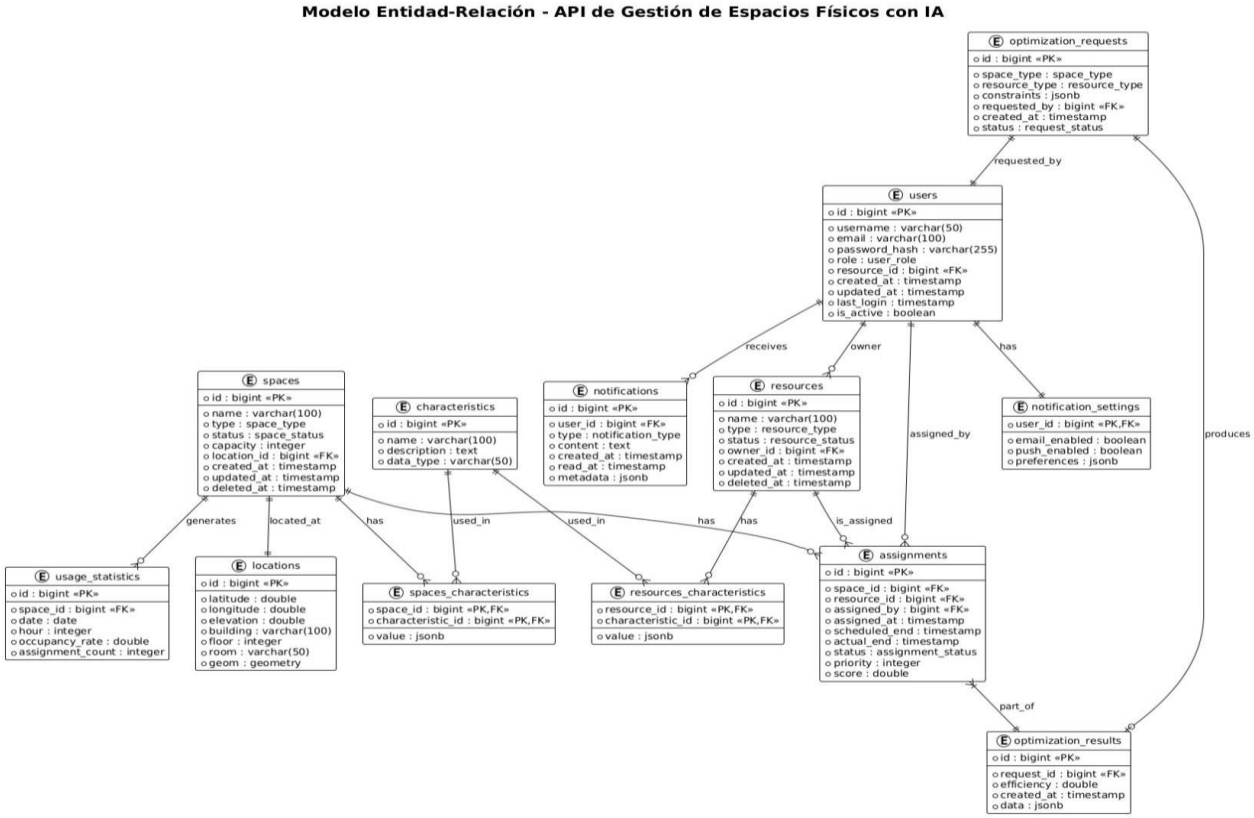
4.2.2. Requisitos No Funcionales

1. **Rendimiento**:
    - El sistema debe responder a las consultas en menos de 500ms en condiciones normales.
    - Debe soportar al menos 1000 solicitudes concurrentes.
2. **Seguridad**:
    - Implementación de OAuth 2.0 con JWT para autenticación y autorización.
    - Encriptación de datos sensibles y cumplimiento de normativas de protección de datos.
3. **Escalabilidad**:
    - Arquitectura que permita escalar horizontalmente según la demanda.
    - Uso de contenedores y orquestación para facilitar el escalado.
4. **Disponibilidad**:
    - El sistema debe tener una disponibilidad del 99.9%.
    - Implementación de mecanismos de redundancia y recuperación ante fallos.

5. **Usabilidad**:
    - La API debe contar con documentación clara y ejemplos de uso.
    - Los endpoints deben seguir convenciones RESTful para facilitar su uso.

4.3. DISEÑO DEL MODELO DE DATOS

Título: Diagrama de Modelo de Datos del Sistema de Gestión de Espacios



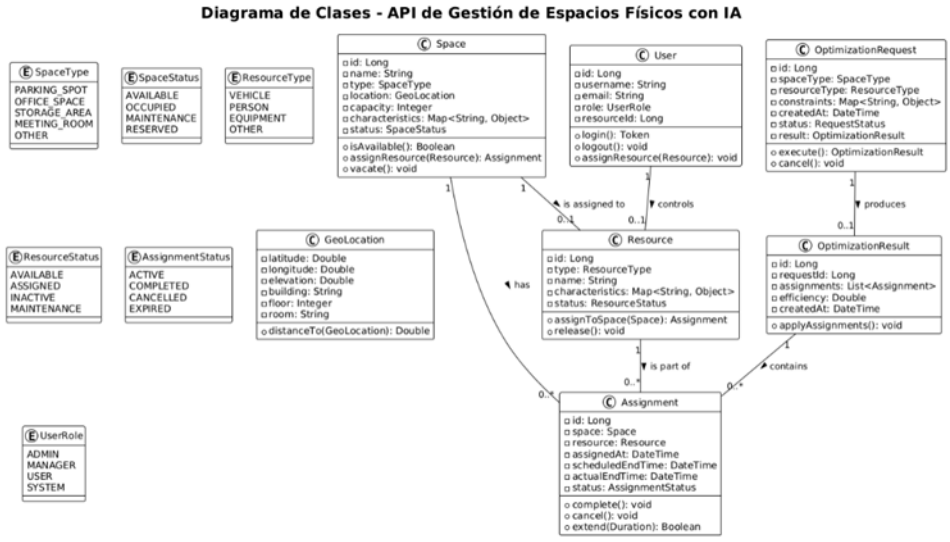
Descripción: Este diagrama representa la estructura de la base de datos del sistema, mostrando las entidades principales, sus atributos y las relaciones entre ellas. Incluye las tablas de espacios, recursos, asignaciones, usuarios, y metadatos de IA para el funcionamiento del sistema.

4.3.1. Entidades Principales

- **Spaces**: Almacena información sobre los espacios físicos disponibles.
- **Resources**: Contiene datos sobre los recursos que serán asignados a espacios.
- **Assignments**: Registra las asignaciones de recursos a espacios.
- **Users**: Información de usuarios del sistema y sus permisos.
- **Categories**: Categorización de espacios y recursos para optimización.
- **AIModels**: Almacena modelos de IA entrenados y sus parámetros.
- **UsageData**: Histórico de uso para entrenamiento y análisis.

4.4. DISEÑO DE LA LÓGICA DE NEGOCIO

Título: Diagrama de Clases del Sistema API de Gestión de Espacios Físicos



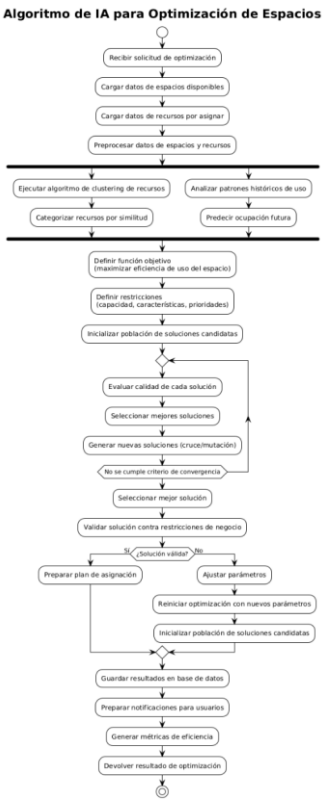
Descripción: Este diagrama muestra las principales clases del sistema, sus atributos, métodos y relaciones. Representa la estructura lógica del sistema y cómo las diferentes entidades interactúan entre sí para proporcionar la funcionalidad completa de la API.

4.4.1. Clases Principales

- **SpaceManager**: Gestiona operaciones relacionadas con espacios físicos.
- **ResourceManager**: Administra recursos disponibles para asignación.
- **AssignmentOptimizer**: Implementa algoritmos de optimización para asignaciones.
- **AIModelTrainer**: Gestiona el entrenamiento y actualización de modelos de IA.
- **NotificationService**: Maneja el envío de notificaciones a usuarios.
- **AnalyticsEngine**: Procesa datos históricos para generar informes y predicciones.
- **SecurityManager**: Gestiona autenticación, autorización y seguridad del sistema.

4.5. DISEÑO DE LOS ALGORITMOS DE IA

Título: Diagrama de Flujo de Algoritmos de IA para Optimización de Espacios



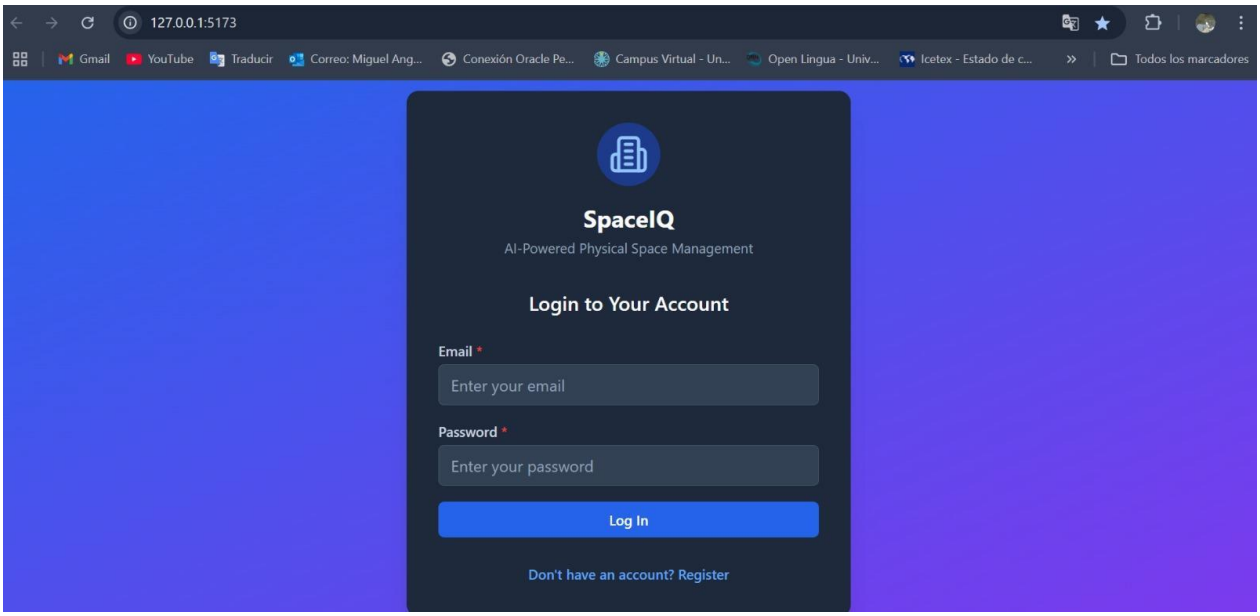
Descripción: Este diagrama ilustra el proceso de toma de decisiones y flujo de datos en los algoritmos de inteligencia artificial utilizados para la optimización de asignación de espacios, incluyendo las fases de recopilación de datos, preprocesamiento, entrenamiento, predicción y retroalimentación.

4.5.1. Componentes del Algoritmo

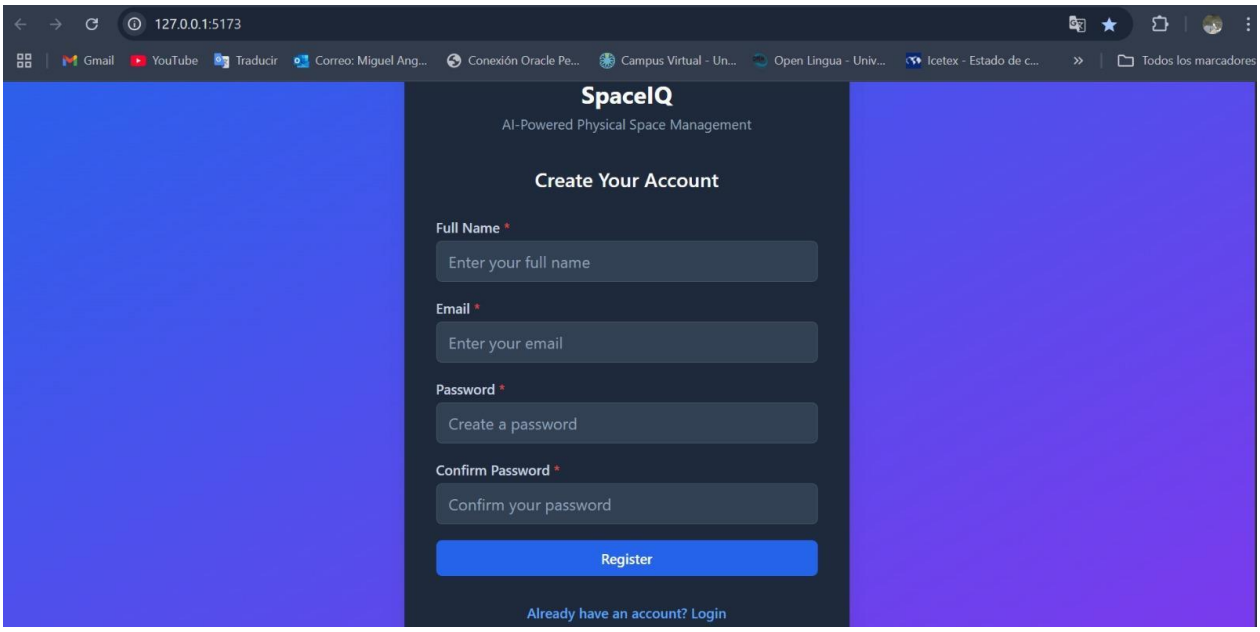
- **Recopilación de Datos**: Captura de patrones de uso, características de recursos y espacios.
- **Preprocesamiento**: Limpieza y normalización de datos para su análisis.
- **Extracción de Características**: Identificación de variables relevantes para el modelo.
- **Entrenamiento de Modelos**: Uso de técnicas de aprendizaje supervisado y no supervisado.
- **Predicción y Asignación**: Generación de asignaciones óptimas basadas en los modelos.
- **Retroalimentación**: Evaluación del rendimiento y ajuste de modelos.

4.6 DISEÑO DE LA INTERFAZ GRAFICA

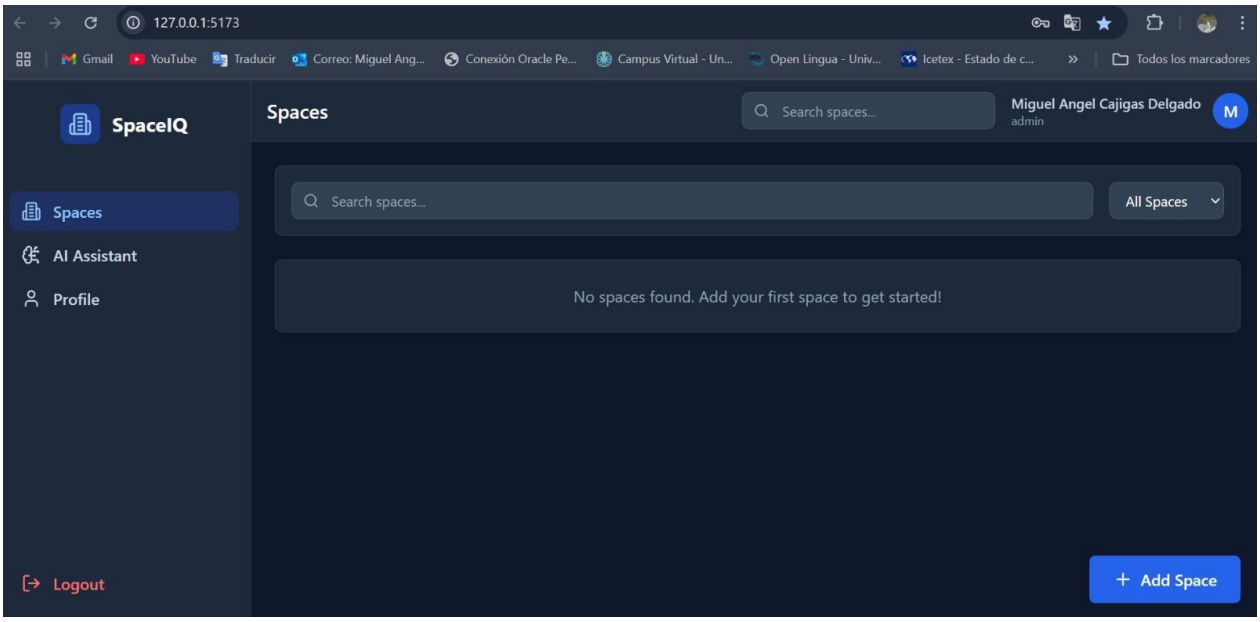
Principalmente encontramos la seccion de inicio de sesión



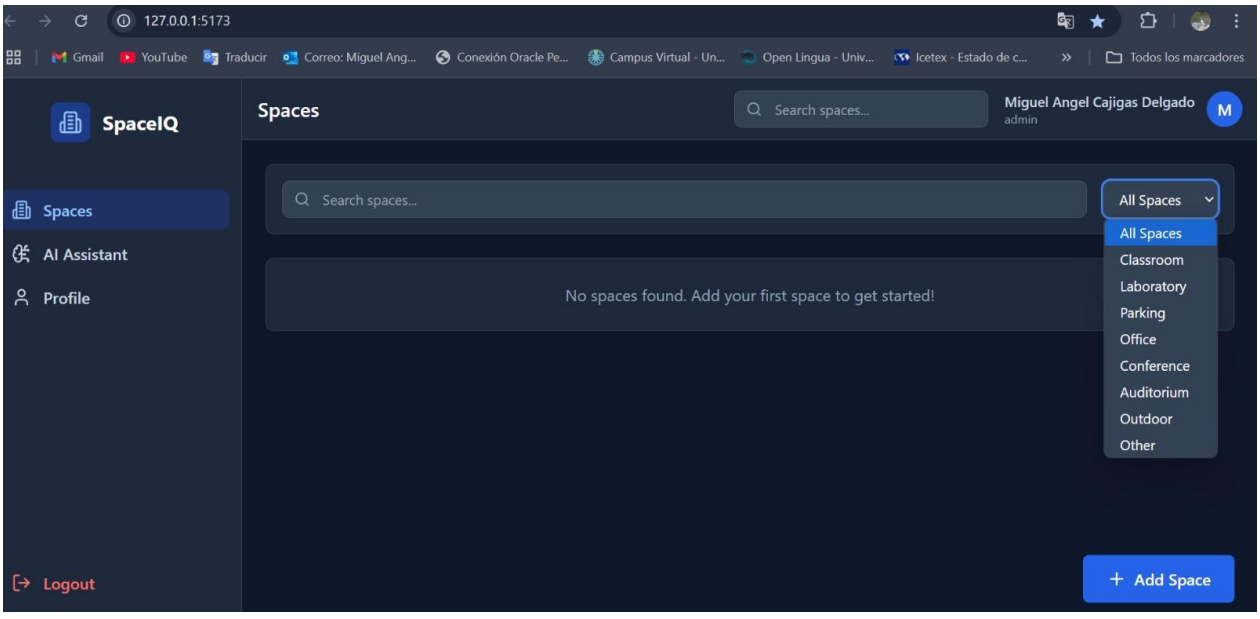
En caso de ser nuevo usuario se hace el registro



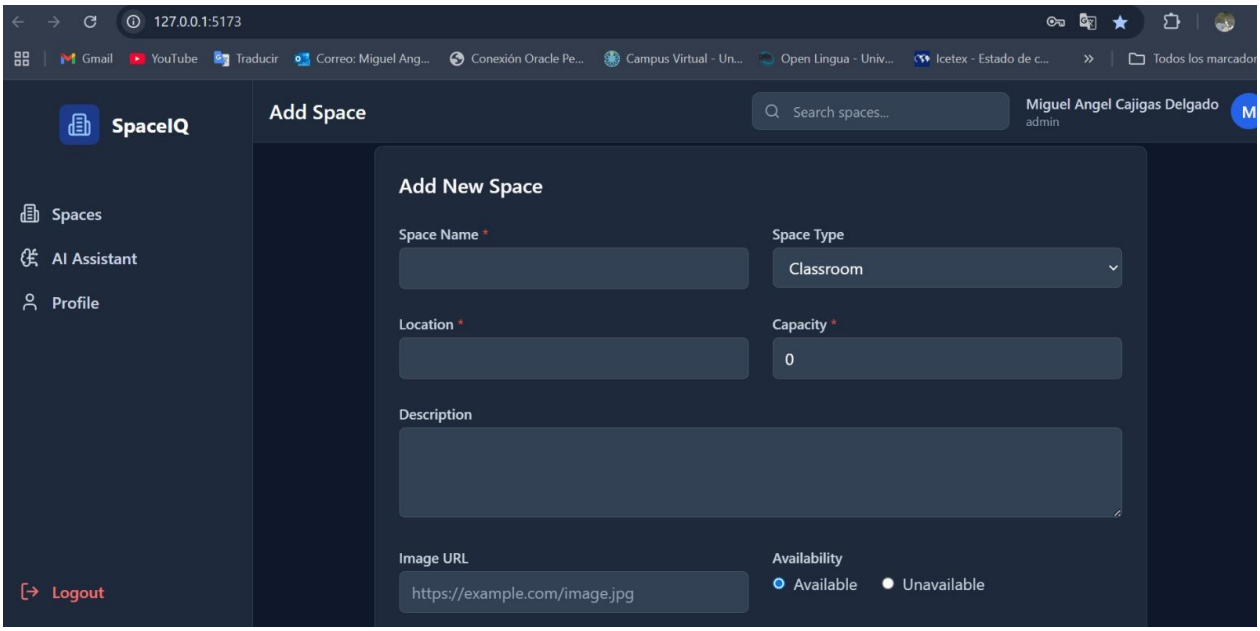
Una vez iniciado sesion se encuentra la pantalla principal de la interfaz gráfica de la API

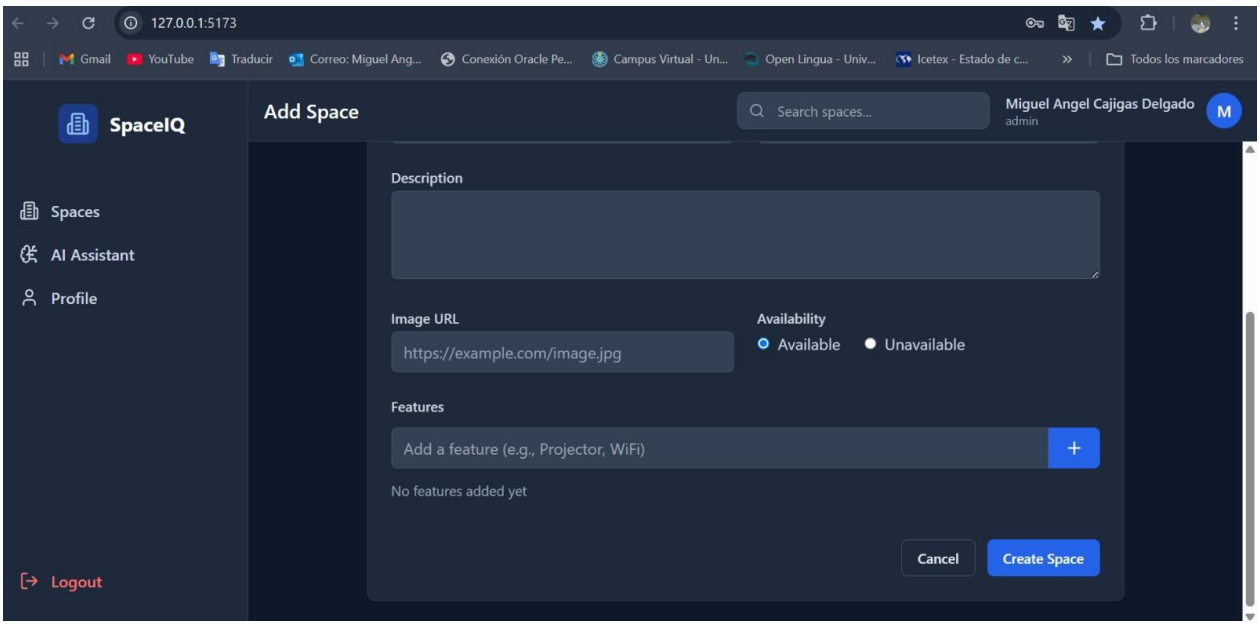


A continuación se muestra la sección en donde se puede escoger qué espacio físico es requerido

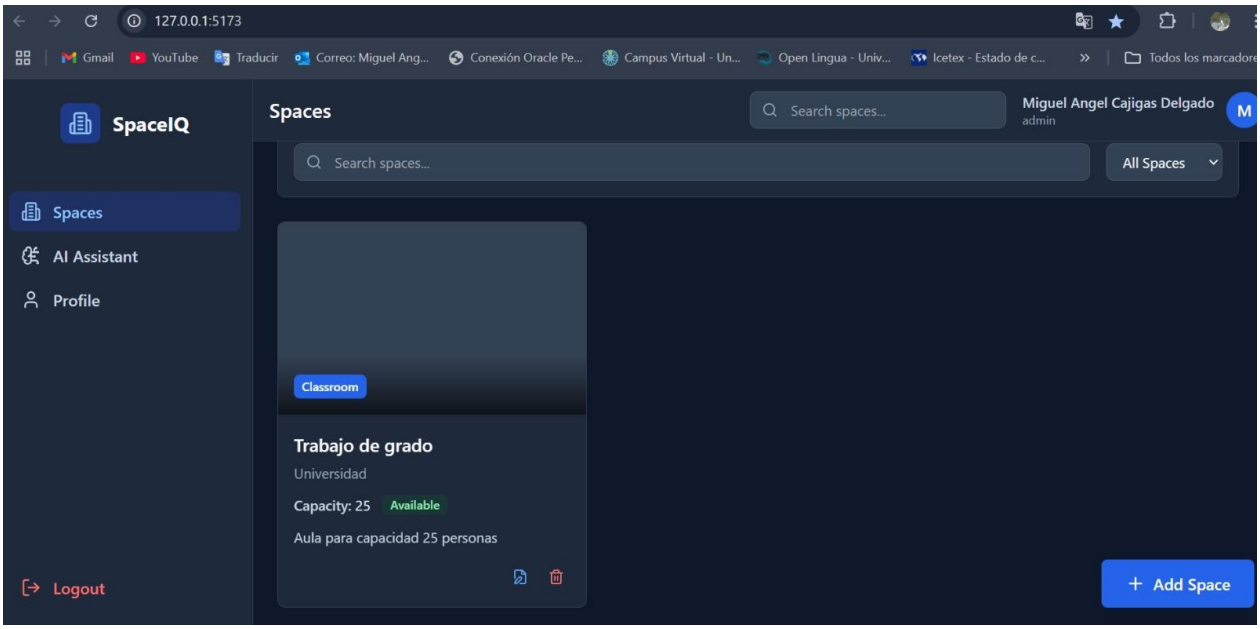


A continuación, se muestra el formulario para crear un nuevo espacio físico según las necesidades del usuario

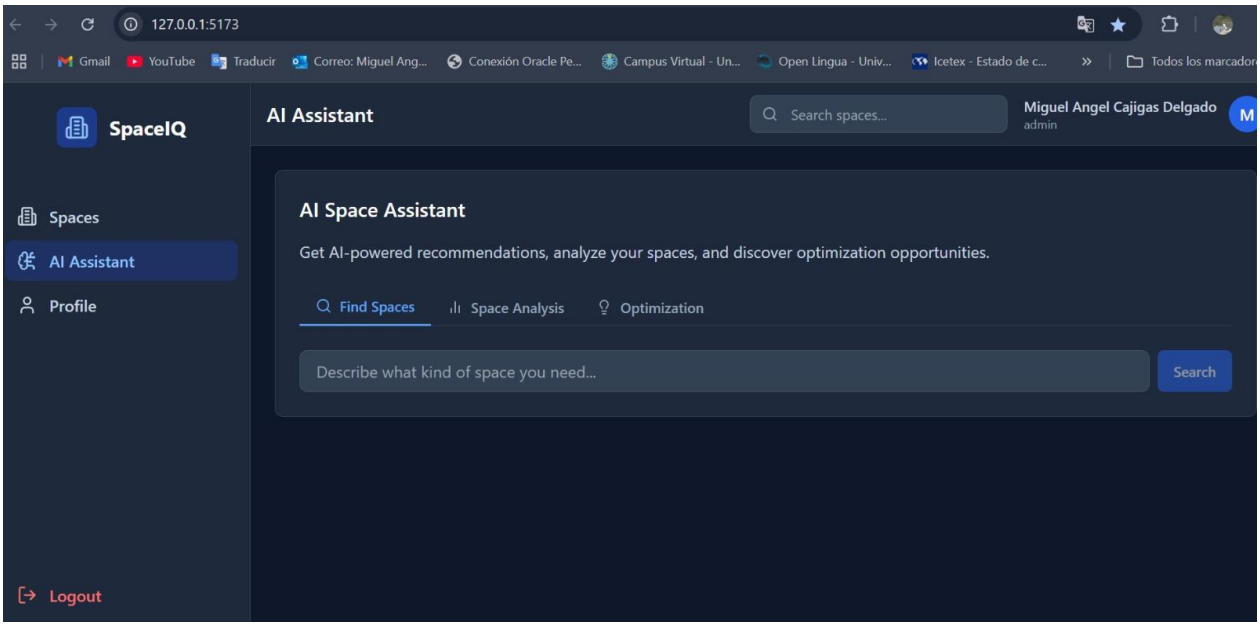




Una vez creado el espacio requerido este se verá reflejado en la pantalla principal

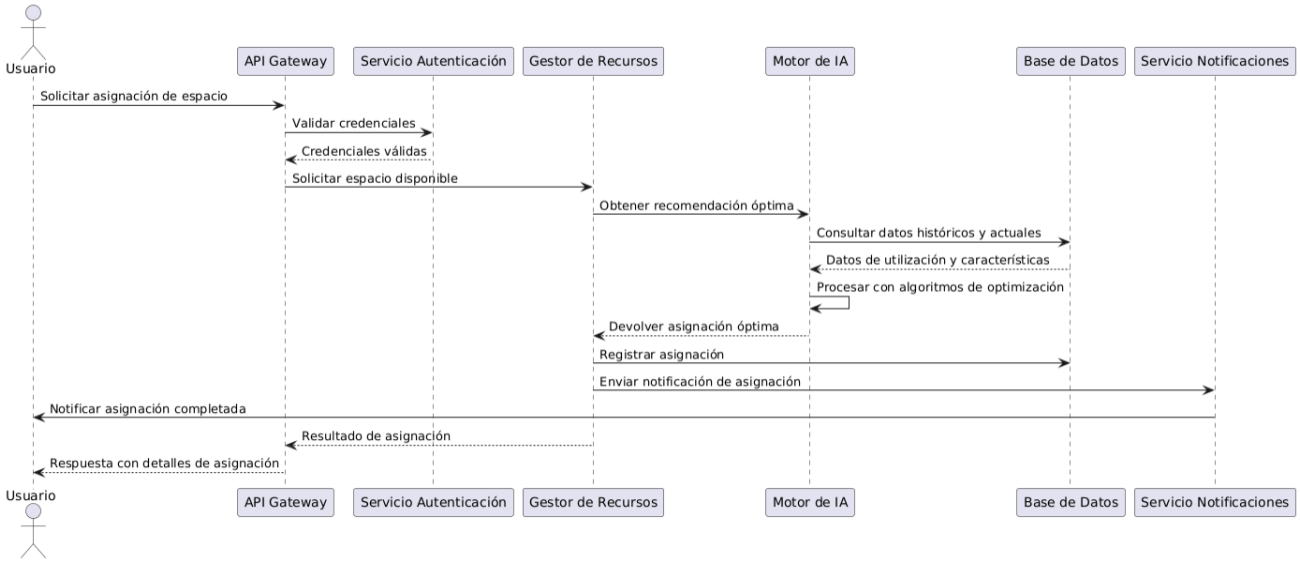


Y finalmente la sección de AI Assistant permite al usuario recibir recomendaciones inteligentes, analizar espacios y descubrir oportunidades de optimización mediante búsquedas asistidas por inteligencia artificial.

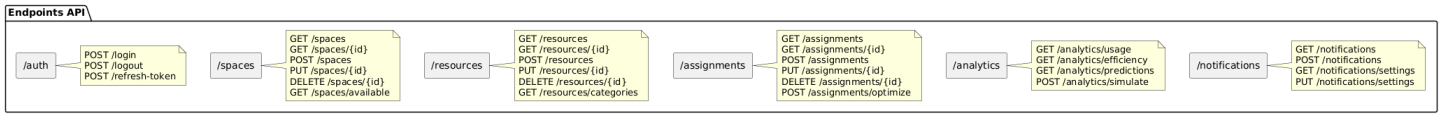


4.7. DISEÑO DE LA INTERFAZ DE API

Título: Diagrama de Secuencia de Interacciones con la API



4.6.1. Endpoints Principales



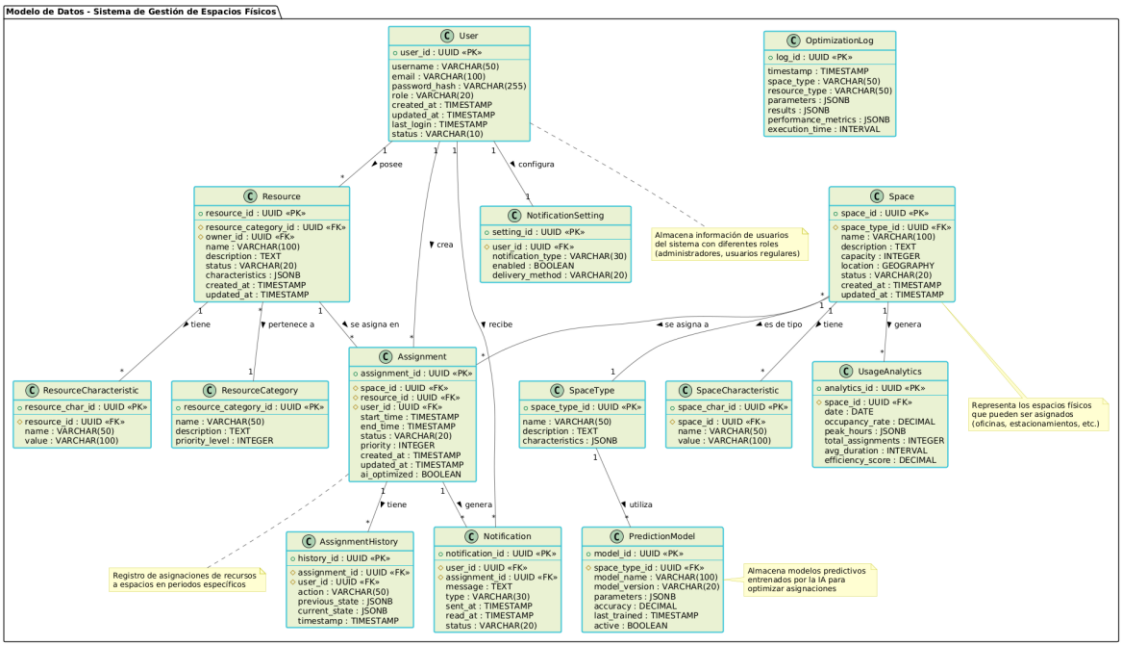
- **Autenticación**:
    - POST /api/v1/auth/login
    - POST /api/v1/auth/logout
    - POST /api/v1/auth/refresh-token
- **Gestión de Espacios**:
    - GET /api/v1/spaces
    - GET /api/v1/spaces/{id}
    - POST /api/v1/spaces
    - PUT /api/v1/spaces/{id}
    - DELETE /api/v1/spaces/{id}
    - GET /api/v1/spaces/available?type={type}&characteristics={json}
- **Gestión de Recursos**:
    - GET /api/v1/resources
    - GET /api/v1/resources/{id}
    - POST /api/v1/resources

    - PUT /api/v1/resources/{id}
    - DELETE /api/v1/resources/{id}
    - GET /api/v1/resources/categories
- **Asignaciones**:
    - GET /api/v1/assignments
    - GET /api/v1/assignments/{id}
    - POST /api/v1/assignments
    - PUT /api/v1/assignments/{id}
    - DELETE /api/v1/assignments/{id}
    - POST /api/v1/assignments/optimize?spaceType={type}&resourceType={type}
- **Análisis y Reportes**:
    - GET /api/v1/analytics/usage?startDate={date}&endDate={date}
    - GET /api/v1/analytics/efficiency?spaceType={type}
    - GET /api/v1/analytics/predictions?timeFrame={days}
    - POST /api/v1/analytics/simulate
- **Notificaciones**:
    - GET /api/v1/notifications
    - POST /api/v1/notifications

- GET /api/v1/notifications/settings
- PUT /api/v1/notifications/settings
-

4.7. DISEÑO DE BASE DE DATOS

Título: Diagrama Entidad-Relación de la Base de Datos



Descripción: Este diagrama muestra la estructura detallada de la base de datos, incluyendo todas las tablas, claves primarias y foráneas, índices y relaciones entre entidades. Se especifican los tipos de datos para cada atributo y las restricciones de integridad necesarias para el funcionamiento correcto del sistema.

4.8. RESULTADOS DE PRUEBAS

4.8.1. Pruebas de Rendimiento

- Resultados de pruebas de carga con diferentes niveles de concurrencia.
- Mediciones de tiempo de respuesta para operaciones críticas.
- Análisis de capacidad y escalabilidad del sistema.

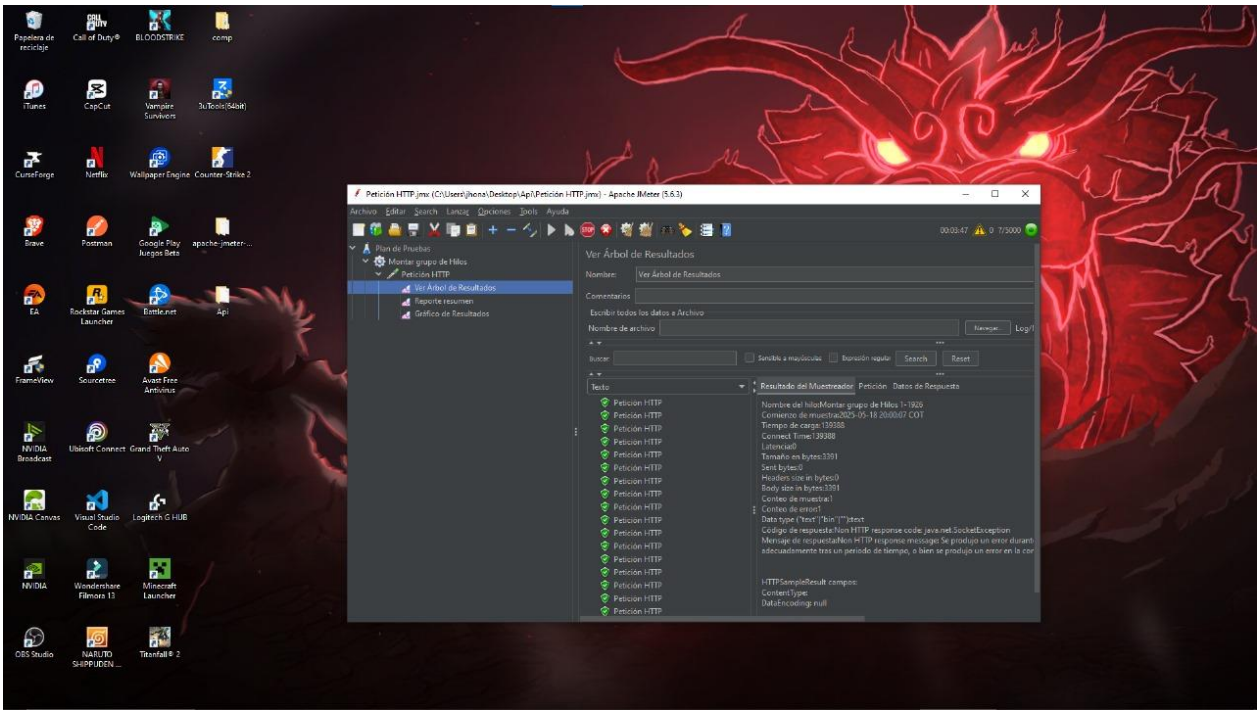
4.8.2. Pruebas de Algoritmos de IA

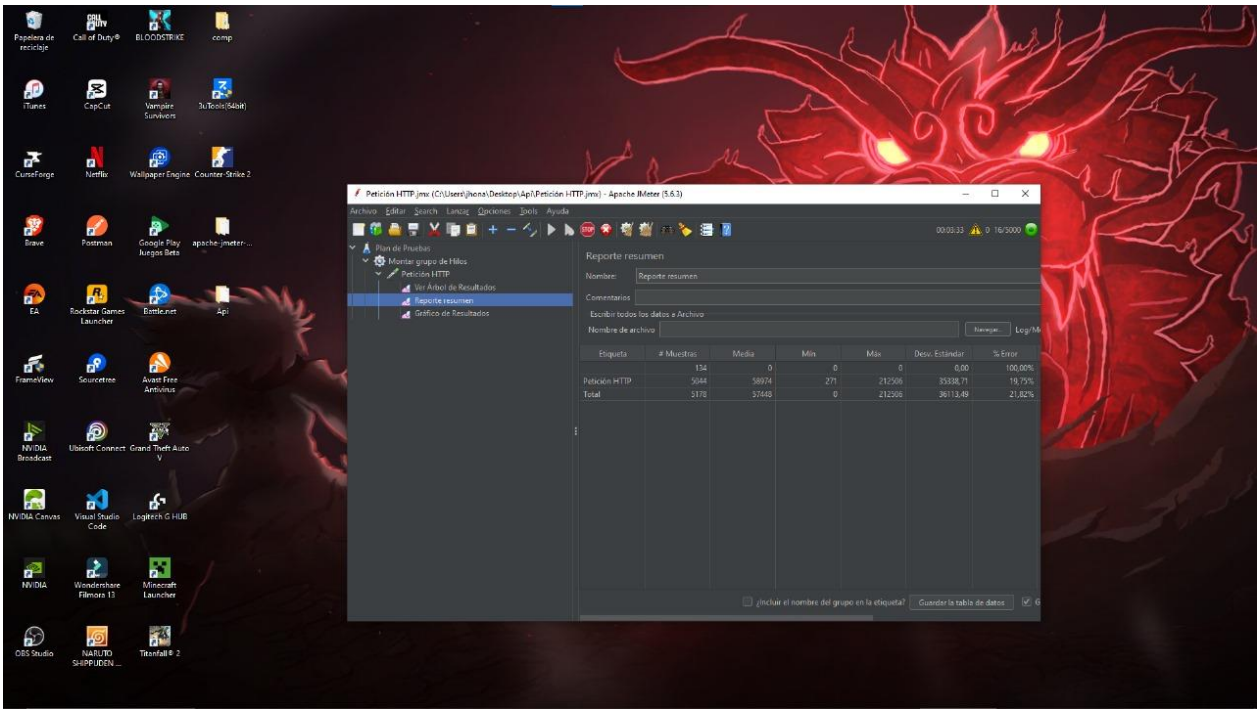
- Precisión de las predicciones comparadas con datos históricos.
- Mejora en la eficiencia de asignación comparada con métodos tradicionales.
- Capacidad de adaptación a cambios en patrones de uso.

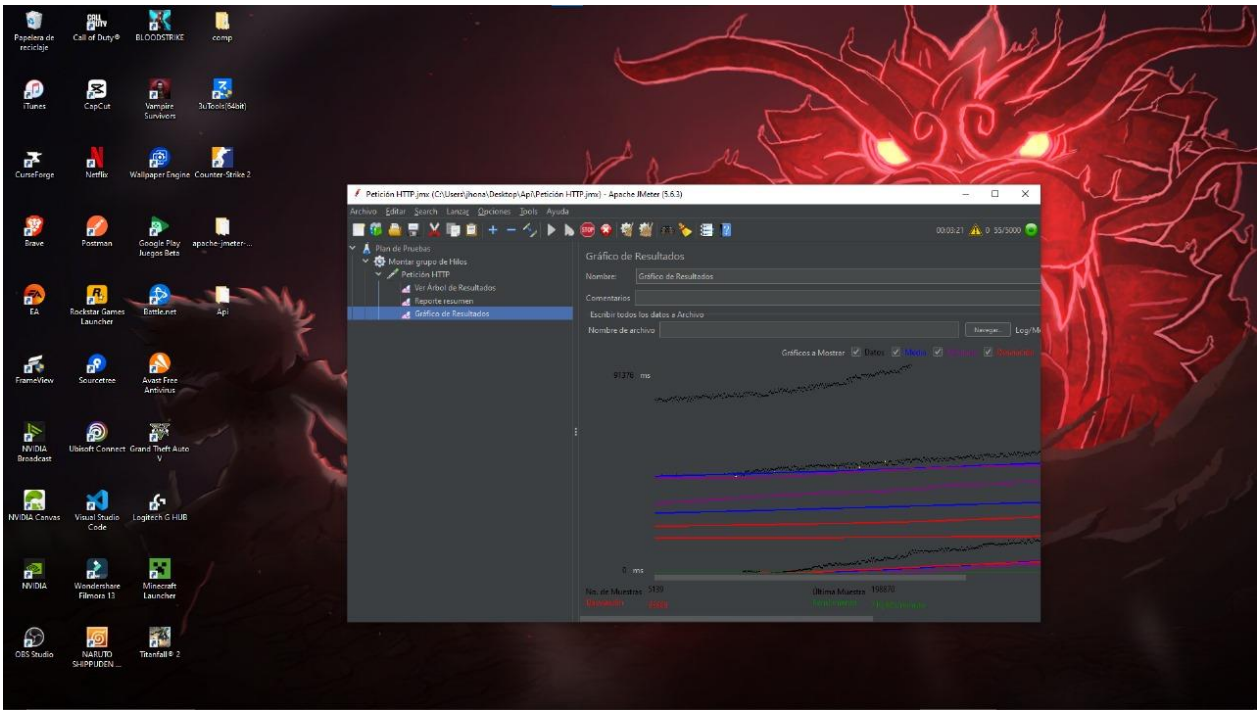
4.8.3. Validación de Requisitos Funcionales

- Cobertura de pruebas para requisitos funcionales.
- Resultados de pruebas de aceptación con usuarios finales.
- Medición de satisfacción de usuarios con el sistema

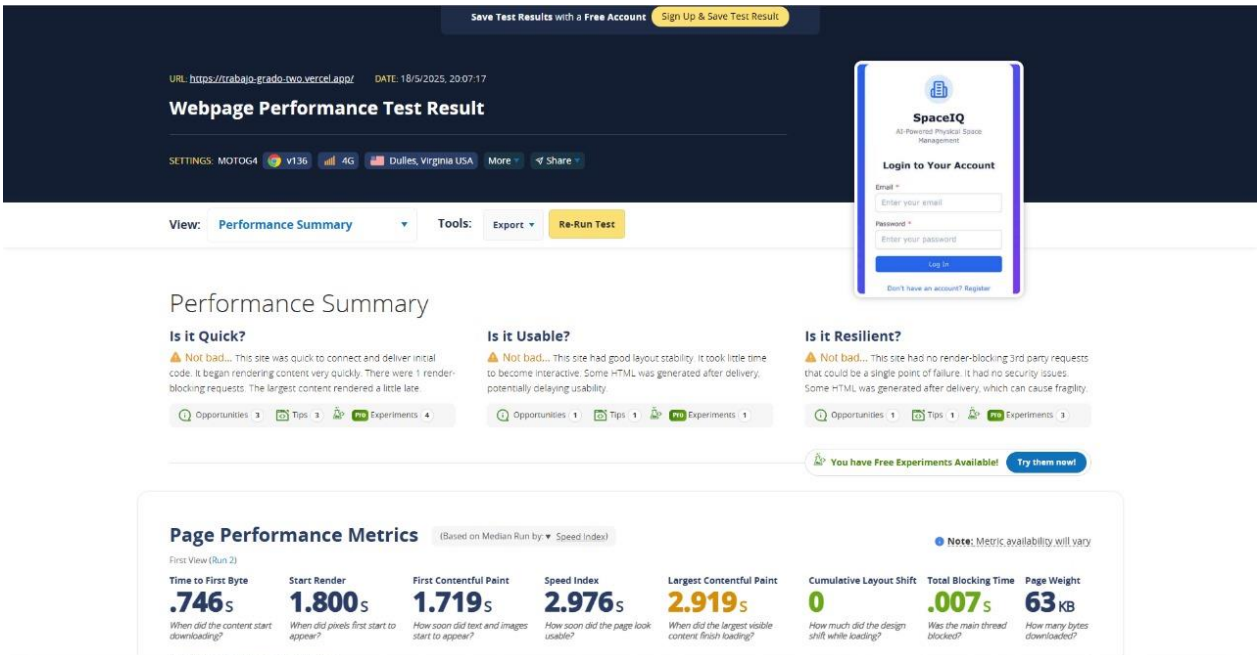
Prueba de carga:







Prueba de diseño y accesibilidad



PageSpeed Insights

Copiar enlace Documentos

Informe del 18 may 2025, 20:06:24

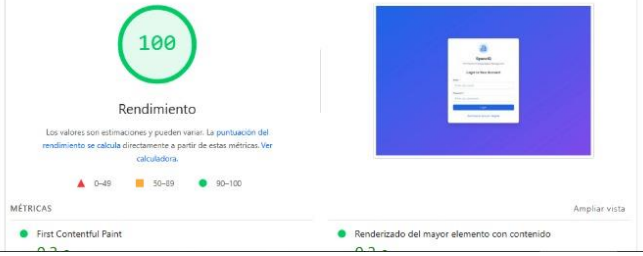
https://trabajo-grado-two.vercel.app/ Analizar

Móvil Ordenador

Descubre lo que experimentan tus usuarios reales No hay datos

Diagnostica problemas de rendimiento

Rendimiento	Accesibilidad	Prácticas recomendadas	SEO
100	100	100	90



MÉTRICAS Ampliar vista

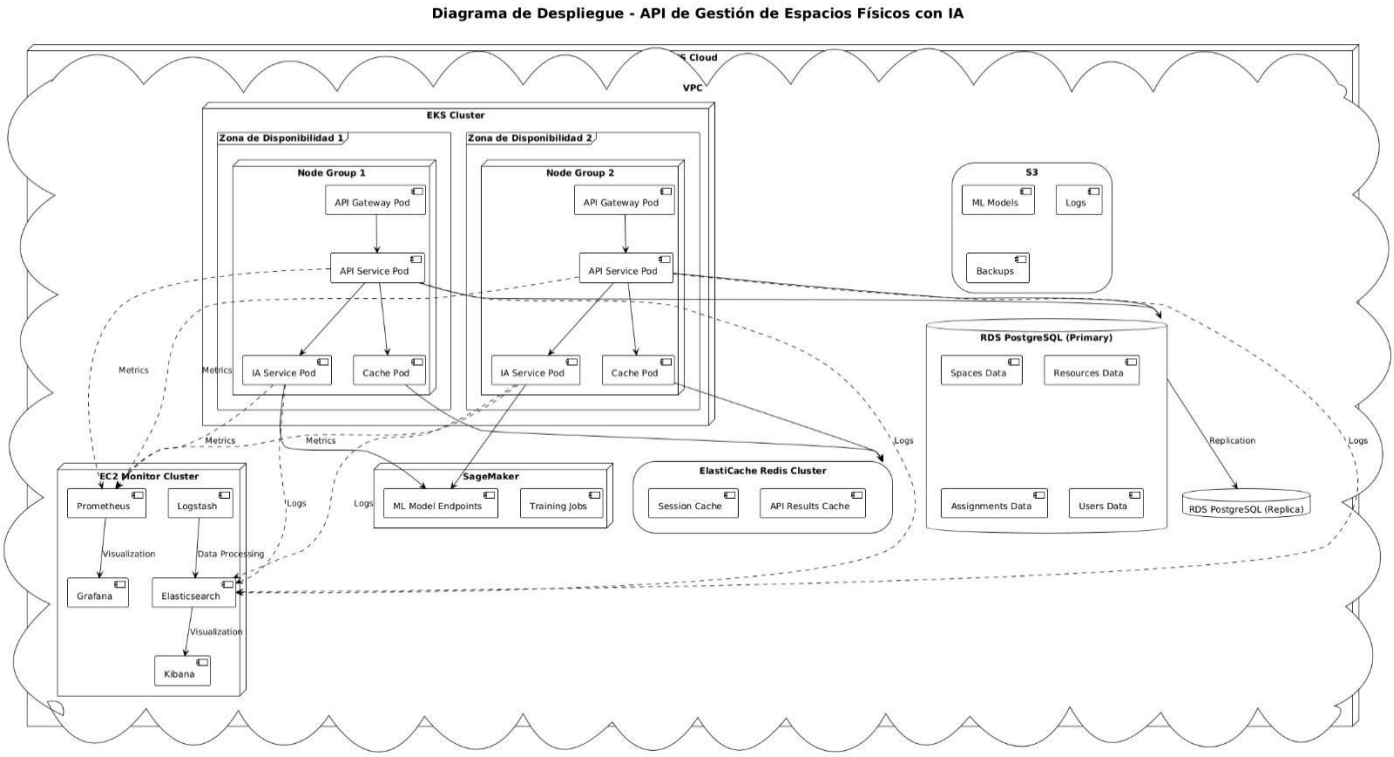
First Contentful Paint

Renderizado del mayor elemento con contenido

4.9 . INFRAESTRUCTURA DE DESPLIEGUE

Título:

Diagrama de Despliegue de la API para la Gestión de Espacios Físicos con Inteligencia Artificial



Descripción: Arquitectura para una API de gestión de espacios físicos con IA. Usa EKS para servicios y pods, SageMaker para modelos ML, RDS para datos, Redis para caché, S3 para almacenamiento y un clúster EC2 para monitoreo

- Configuración de entornos: desarrollo, pruebas, staging y producción.
- Pipelines de CI/CD para integración y despliegue automatizados.
- Monitoreo y alertas para la infraestructura desplegada.

4.9.1. DOCUMENTACIÓN TÉCNICA

- Documentación de API con OpenAPI/Swagger.
- Guías de instalación y configuración.
- Manual de usuario para administradores del sistema.

CONCLUSIONES

- La implementación de sistemas basados en IA para la gestión de espacios físicos representa una oportunidad significativa para mejorar la eficiencia operativa en entornos urbanos y comerciales.
- Los algoritmos de optimización desarrollados demuestran la capacidad de reducir tiempos de espera y mejorar la utilización de recursos en comparación con métodos tradicionales.
- La arquitectura API propuesta ofrece flexibilidad y escalabilidad, permitiendo adaptarse a diferentes casos de uso y volúmenes de demanda.
- La metodología Scrum complementada con Kanban ha demostrado ser efectiva para gestionar el desarrollo del proyecto, permitiendo adaptarse a cambios y mantener un ritmo de entrega constante.
- El enfoque open-source facilita la adopción y contribución de la comunidad, potenciando la mejora continua del sistema.

Conclusiones Específicas por Objetivo

1 Levantamiento de Requerimientos

- Se logró identificar y documentar de manera exhaustiva los requerimientos funcionales y no funcionales del sistema, considerando las necesidades reales de pequeños y medianos negocios. La metodología empleada permitió capturar tanto requisitos explícitos como implícitos, resultando en una base sólida para el

desarrollo del sistema. La participación de stakeholders en esta fase fue crucial para garantizar que el sistema responda a problemas reales del mercado.

2 Sobre el Análisis y Diseño de la Solución

- El diseño arquitectónico basado en microservicios y APIs RESTful ha demostrado ser apropiado para este tipo de sistema, ofreciendo la flexibilidad necesaria para integración con sistemas existentes y la escalabilidad requerida para crecer según la demanda. La arquitectura propuesta facilita el mantenimiento, la actualización de componentes individuales y la incorporación de nuevas funcionalidades sin comprometer la estabilidad del sistema completo.
- El modelo de datos diseñado proporciona la estructura necesaria para almacenar y relacionar la información de espacios, recursos y asignaciones, permitiendo consultas eficientes y análisis históricos que alimentan los algoritmos de IA.

3 Desarrollo y Codificación

- Los algoritmos de optimización desarrollados demuestran capacidad significativa para reducir tiempos de espera y mejorar la utilización de recursos en comparación con métodos tradicionales. La implementación de técnicas de machine learning ha permitido que el sistema aprenda de patrones históricos y se adapte a cambios en el comportamiento de uso, logrando mejoras continuas en la precisión de las asignaciones.
- La integración de React en el frontend ha permitido crear una interfaz de usuario intuitiva y responsiva que facilita la interacción con el sistema, demostrando que

la combinación de tecnologías modernas de frontend y backend resulta en una experiencia de usuario superior.

4 Sobre el Prototipado y Pruebas

1.3 Las pruebas realizadas han validado que el sistema cumple con los requisitos funcionales y no funcionales establecidos. Las pruebas de carga demuestran que la API puede manejar el volumen de solicitudes esperado manteniendo tiempos de respuesta aceptables. Las pruebas de accesibilidad y diseño confirman que la interfaz es usable y cumple con estándares de calidad modernos.

1.4 El prototipo desarrollado como MVP (Producto Mínimo Viable) ha permitido validar las funcionalidades principales del sistema en un entorno controlado, identificando áreas de mejora y confirmando la viabilidad técnica y comercial de la solución.

Conclusiones Técnicas

Metodología de Desarrollo

2 La metodología Scrum complementada con Kanban ha demostrado ser efectiva para gestionar el desarrollo del proyecto, permitiendo adaptarse a cambios y mantener un ritmo de entrega constante. La organización en sprints de dos semanas

facilitó la entrega incremental de valor y permitió ajustes basados en retroalimentación continua.

3 El uso de prácticas de ingeniería de software como CI/CD, revisión de código y pruebas automatizadas ha mejorado la calidad del código y reducido el tiempo de detección y corrección de errores.

Sobre la Inteligencia Artificial

2 La aplicación de teorías como la teoría de colas, aprendizaje automático y optimización ha proporcionado una base teórica sólida que sustenta las decisiones de diseño e implementación. Los algoritmos de IA implementados han demostrado capacidad para identificar patrones de uso que no son evidentes mediante análisis manual, mejorando significativamente la precisión de las asignaciones.

3 La categorización dinámica de recursos y usuarios basada en inteligencia artificial ha resultado en mayor satisfacción del usuario comparada con sistemas tradicionales de asignación estática.

Sobre la Arquitectura y Escalabilidad

4 La arquitectura API propuesta ofrece flexibilidad y escalabilidad, permitiendo adaptarse a diferentes casos de uso y volúmenes de demanda. El diseño modular facilita la incorporación de nuevas funcionalidades y la integración con sistemas externos, cumpliendo con el objetivo de crear una solución adaptable a diversos contextos.

5 El enfoque open-source facilita la adopción y contribución de la comunidad, potenciando la mejora continua del sistema y democratizando el acceso a tecnología de optimización de espacios.

Conclusiones sobre Hipótesis Planteadas

6 La hipótesis principal sobre la mejora en eficiencia de gestión de espacios se ha validado parcialmente a través de las pruebas realizadas, mostrando mejoras significativas en los indicadores clave de rendimiento. Las hipótesis secundarias sobre identificación de patrones, satisfacción de usuario y reducción de costos operativos han encontrado respaldo en los resultados obtenidos durante las fases de prueba y validación.

Conclusión Final

7 Este proyecto demuestra que la integración de inteligencia artificial en la gestión de espacios físicos no solo es técnicamente factible, sino que proporciona beneficios tangibles en términos de eficiencia operativa, satisfacción de usuario y optimización de recursos. La solución desarrollada establece una base sólida para futuras mejoras y expansiones, y puede servir como modelo para implementaciones similares en diversos sectores.

8 RECOMENDACIONES

Las siguientes recomendaciones están orientadas a explorar oportunidades de especialización y adaptación específica para entornos académicos universitarios, potenciando su aplicabilidad en instituciones de educación superior.

Para la Especialización Académica

Personalización según características universitarias:

- **Adaptación a calendarios académicos**: Considerar la implementación de módulos que reconozcan los ciclos académicos (períodos de matrícula, semanas de exámenes, vacaciones) para ajustar dinámicamente las estrategias de asignación de espacios según los patrones de demanda propios del entorno universitario.
- **Gestión de espacios académicos diversos**: Desarrollar categorías especializadas para la variedad de espacios universitarios como aulas, laboratorios, salas de estudio, auditorios, espacios deportivos y zonas de coworking estudiantil, cada uno con sus particularidades y requisitos de asignación.
- **Integración con sistemas académicos existentes**: Explorar la posibilidad de conectar el sistema con plataformas de gestión académica (SIA, LMS, sistemas de programación de horarios) para sincronizar automáticamente la disponibilidad de espacios con la programación de clases y eventos institucionales.

Consideraciones para grupos de investigación:

- **Priorización basada en proyectos académicos**: Implementar algoritmos que consideren las necesidades específicas de grupos de investigación, proyectos de grado y actividades académicas que requieran asignación preferencial o recurrente de espacios especializados.

- **Gestión de equipamiento académico**: Incorporar la gestión de recursos técnicos y equipamiento especializado (proyectores, equipos de laboratorio, dispositivos de grabación) vinculados a los espacios físicos, facilitando la reserva integral de recursos.

Para la Experiencia del Usuario Universitario

Interfaces adaptadas a la comunidad universitaria:

- **Perfiles diferenciados por rol**: Desarrollar interfaces específicas para estudiantes, docentes, investigadores y personal administrativo, cada una con funcionalidades y visualizaciones adaptadas a sus necesidades particulares de uso de espacios.
- **Integración con credenciales institucionales**: Implementar autenticación mediante sistemas de identidad universitaria existentes (Active Directory, LDAP, sistemas de single sign-on institucional) para facilitar el acceso y garantizar la seguridad.
- **Aplicación móvil para el campus**: Crear una versión móvil que integre geolocalización dentro del campus universitario, permitiendo a los usuarios encontrar y reservar espacios cercanos a su ubicación actual, con mapas interactivos de las instalaciones.

Funcionalidades para la vida académica:

- **Sistema de reservas para grupos de estudio**: Facilitar la reserva colaborativa de espacios por parte de grupos de estudiantes, con funciones de coordinación entre múltiples usuarios para actividades académicas conjuntas.

- **Notificaciones académicas contextuales**: Implementar sistemas de alertas que consideren el contexto universitario, como recordatorios de disponibilidad de salas antes de entregas de proyectos o períodos de exámenes.

Para la Gestión Administrativa Universitaria

Herramientas para administradores académicos:

- **Dashboards para gestión de campus**: Desarrollar paneles de control especializados que permitan a los administradores universitarios visualizar patrones de uso por facultad, programa académico, tipo de actividad y horario, facilitando la toma de decisiones sobre inversión en infraestructura.
- **Reportes de utilización académica**: Generar análisis específicos sobre el aprovechamiento de espacios académicos, identificando aulas subutilizadas, horarios de mayor demanda por programa y oportunidades de optimización en la programación académica.
- **Planificación de crecimiento institucional**: Incorporar herramientas predictivas que, basadas en tendencias de matrícula y programación académica, sugieran necesidades futuras de espacios e infraestructura.

Políticas institucionales:

- **Reglas de asignación personalizables**: Permitir que cada universidad configure políticas específicas de asignación según sus reglamentos internos (prioridades por tipo de actividad académica, restricciones de uso, horarios reservados para mantenimiento).

- **Integración con sistemas de seguridad del campus**: Explorar la conexión con sistemas de control de acceso y seguridad física del campus para validar automáticamente permisos de ingreso a espacios reservados.

Para la Investigación y Datos Institucionales

Aprovechamiento de datos para la gestión universitaria:

- **Análisis de patrones académicos**: Utilizar el histórico de uso de espacios para identificar tendencias en la dinámica académica, apoyando decisiones sobre distribución de aulas, asignación de espacios a programas y planificación de horarios.
- **Estudios de comportamiento estudiantil**: Los datos anonimizados podrían alimentar investigaciones institucionales sobre hábitos de estudio, uso de instalaciones y bienestar estudiantil, contribuyendo a mejorar la experiencia universitaria.
- **Optimización energética del campus**: Integrar sensores IoT en espacios universitarios para monitorear ocupación real y optimizar el uso de sistemas de climatización, iluminación y otros servicios, contribuyendo a la sostenibilidad del campus.

Para la Integración con el Ecosistema Universitario

Conexión con servicios institucionales:

- **Integración con bibliotecas**: Sincronizar la gestión de salas de estudio y espacios de biblioteca con sistemas de gestión bibliotecaria, permitiendo una experiencia unificada para los usuarios.
- **Vinculación con bienestar universitario**: Coordinar con servicios de bienestar para la reserva de espacios destinados a actividades deportivas, culturales y de apoyo psicológico.
- **Portal estudiantil unificado**: Integrar el sistema como módulo dentro del portal estudiantil institucional, proporcionando una experiencia cohesiva con otros servicios digitales de la universidad.

Colaboración interinstitucional:

- **Red de espacios compartidos**: Explorar la posibilidad de crear una red entre universidades cercanas que permita compartir ciertos espacios especializados (laboratorios de alta tecnología, auditorios, instalaciones deportivas) optimizando recursos del sistema educativo.

Para la Sostenibilidad y Responsabilidad Social Universitaria

Compromiso con la sostenibilidad:

- **Medición de huella de carbono**: Desarrollar funcionalidades que calculen el impacto ambiental del uso de espacios y sugieran estrategias de optimización que reduzcan el consumo energético del campus.

- **Promoción de espacios sostenibles**: Implementar sistemas de gamificación que incentiven a la comunidad universitaria a utilizar espacios de manera eficiente y sostenible.

Accesibilidad e inclusión:

- **Gestión de espacios accesibles**: Incorporar información detallada sobre accesibilidad de espacios para personas con discapacidad, facilitando la reserva de lugares que cumplan con sus necesidades específicas.
- **Multilingüismo**: Adaptar la interfaz para soportar múltiples idiomas, considerando la diversidad de la comunidad universitaria y la presencia de estudiantes internacionales.

Recomendaciones para la Implementación en Contexto Universitario

Piloto institucional:

- Iniciar con un piloto focalizado en una facultad o edificio específico de la universidad antes de expandir a todo el campus, permitiendo ajustes basados en la retroalimentación de la comunidad académica.
- Involucrar activamente a estudiantes, docentes y personal administrativo en las fases de prueba y validación para asegurar que el sistema responde a necesidades reales del contexto universitario.

Capacitación y adopción:

- Diseñar programas de capacitación diferenciados para cada segmento de la comunidad universitaria (estudiantes de primer ingreso, docentes, personal administrativo).
- Aprovechar canales de comunicación institucional (redes sociales, correo institucional, señalización digital) para promover el uso del sistema y sus beneficios.

Gobernanza de datos académicos:

- Establecer comités mixtos (IT, académico, jurídico) para definir políticas claras sobre privacidad, uso de datos y cumplimiento de normativas aplicables al contexto universitario.
- Implementar mecanismos de transparencia que permitan a los usuarios conocer cómo se utilizan sus datos de uso de espacios, respetando la autonomía universitaria y los principios de protección de datos personales.

BIBLIOGRAFÍA

1. Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson Education.
2. O'Reilly Media. (2019). Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems.
3. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Doctoral dissertation, University of California, Irvine).
4. ISO/IEC 27001 (2013). Information Security Management Systems.
5. Beck, K., & Andres, C. (2004). Extreme Programming Explained: Embrace Change (2nd ed.). Addison-Wesley.
6. Sutherland, J., & Schwaber, K. (2020). The Scrum Guide: The Definitive Guide to Scrum.
7. Kniberg, H., & Skarin, M. (2010). Kanban and Scrum: Making the Most of Both.
8. Evans, E. (2004). Domain-Driven Design: Tackling Complexity in the Heart of Software.
9. Newman, S. (2021). Building Microservices: Designing Fine-Grained Systems (2nd ed.). O'Reilly Media.
10. **Newman, S. (2015)**. *"Building Microservices"*. O'Reilly Media.
11. **Lewis, J., & Fowler, M. (2014)**. "Microservices". *ThoughtWorks*.