Universidad ORT Uruguay
Facultad de Ingeniería

Obligatorio de Arquitectura en Sistemas

Romina Arour - 242493
Daiana Aysa - 281651
Gabriel Guerra - 276712

Link al repositorio:
https://github.com/IngSoft-AR-2023-2/242493-276712-281651/tree/main

2024

Introduccion **3**
Antecedentes **3**
Propósito del Sistema 3
Requerimientos Significativos de Arquitectura 3
Resumen de Requerimientos Funcionales 3
Resumen de Requerimientos de Atributos de Calidad 6
Documentación de la Arquitectura **8**
Vistas de modulos 8
Vista de descomposición 8
Vistas de uso 9
Vistas de layers 10
Catálogo de elementos 10
Vistas de Componentes y Conectores 11
Externa 11
Diagrama de Comportamiento 12
Pipeline de creación de un booking 12
Pipeline de inicio de pago de un booking 12
Pipeline de callback por parte de Payment de un booking 13
Decisiones de diseño 13
MySQL 13
MongoDb 13
Manejo de errores 14
Queue Publish Subscribe con prioridad 14
Autenticacion - JWT 14
PM2 y K6 15
ADRs para las Decisiones de Diseño 15
ADR para: Elección del Sistema de Gestión de Bases de Datos 15
ADR para la Selección de MongoDB como Solución de Almacenamiento para Datos de Sensores en Sensor Server 16
ADR para la Gestión de Cambios en los Valores de Sensores Utilizando RabbitMQ 17
ADR para: Uso de JSON Web Tokens (JWT) y delegación de gestión de sesión al Login Server 18
Errores Conocidos 20
Pruebas de carga 20

Introduccion

En este documento, se va a detallar la arquitectura del sistema Inmo2.0

Antecedentes

Propósito del Sistema

El proyecto Inmo2.0 es una nueva forma de gestionar propiedades que se alquilan por poco tiempo, como casas o departamentos de vacaciones. Utilizan tecnología avanzada, incluyendo sensores inteligentes, para vigilar continuamente el estado y la seguridad de las propiedades. Además, esta solución no solo ayuda con las reservas y los pagos, sino que también mejora la administración en general. Todo esto hace que tanto los dueños de las propiedades como los huéspedes tengan una mejor experiencia.

Requerimientos Significativos de Arquitectura

Resumen de Requerimientos Funcionales

Requerimiento	Descripción	Actor
REQ 1 - Login	Cualquier usuario ya registrado podrá autenticarse a través de una plataforma externa, pero la autorización basada en los roles se gestionó dentro del mismo sistema de Inmo2.0.	Administrador, Operario, Inquilino, Propietario
REQ 2 - Usuarios del Sistema	Se tiene una base de datos de usuarios ya precargada, no se tiene que hacer registro.	Administrador
REQ 3 - Búsqueda de inmuebles	Se puede buscar inmuebles. Si se filtra por rango de fechas aparecen los disponibles en estas fechas. Si no se ingresa ese rango, se muestran todos los inmuebles (ocupados y libres). También se puede filtrar por características que tiene cada inmueble (ej: camas, garage, wifi, etc)	Inquilino
REQ 4 - Alta de reservas	Se puede dar de alta una reserva. Se le solicitan datos al Inquilino para poder completar la misma, como por ejemplo: documento, nombre, apellido, teléfono, dirección, país, etc. La reserva implica saber algunos datos del inmueble (código, periodo de reserva). Una vez que se realice la reserva, se le	Inquilino

	notifica al Administrador y al Propietario de lo ocurrido.	
REQ 5 - Realización de pago Online	Se permiten pagos de reservas online integrándose con un sistema de pagos externo. Tras el pago, se notificará si fue aprobado o rechazado. Si es aprobado, se enviará un correo al Propietario y a los Administradores, y el calendario de reservas se actualizará para reflejar la disponibilidad. Si no se realiza el pago, la reserva se cancelará por falta de pago y el inmueble volverá a estar disponible. En este caso, se notificará al Propietario y al inquilino sobre la cancelación.	Inquilino
REQ 6 - Cancelación de reservas	El inquilino podrá cancelar una reserva con unos determinados días de anticipación, recibiendo solo un porcentaje de lo pagado si se excede ese plazo. Para cancelar, deberá identificarse con su email y código de reserva. Una vez cancelada, la reserva pasará a estado “cancelada por el inquilino” y el período del inmueble será liberado.	Inquilino
REQ 7 - Consulta de reserva propia	Para consultar una reserva, el inquilino deberá identificarse con su email y código de reserva. Una vez verificados, el sistema mostrará toda la información detallada de la reserva.	Inquilino
REQ 8 - Consulta de reservas generales	Los administradores y operarios del sistema pueden consultar las reservas en cualquier momento utilizando los siguientes filtros: fecha de reserva desde y hasta, código de inmueble, email del inquilino, ombre y/o apellido del inquilino estado de la reserva	Administrador, Operario
REQ 9 - Alta de Inmueble	Se puede registrar un inmueble proporcionando datos generales como el nombre, capacidad para adultos y niños, cantidad de camas, disponibilidad de aire acondicionado, Wi-Fi, garaje, tipo de propiedad, distancia a la playa, estado, balneario y barrio. El inmueble debe incluir al menos cuatro fotos, cada una con un tamaño máximo de 500kb. Tras las validaciones, el inmueble se quedará en estado "pendiente de pago" y se notificará al	Propietario

	Administrador. Una vez realizado el pago a través de un sistema de pagos externo, el inmueble pasará a estado "activo" y estará disponible para reservas, notificándose nuevamente al Administrador.	
REQ 10 - Indicar agenda de disponibilidad para el año	El propietario puede definir una agenda para fechas futuras que no están ya reservadas, indicando cuándo el inmueble no está disponible por decisión propia y cuándo está habilitado para su alquiler.	Propietario
REQ 11 - Agregar sensor a inmueble	Se pueden asignar sensores a los inmuebles usando un catálogo de sensores. La asignación se hace indicando el código del inmueble y el del sensor. No todas las señales recibidas generarán alertas.	Administrador
REQ 12 - Alta de sensor	El administrador puede registrar un sensor con los siguientes datos: ID del sensor, descripción, número de serie, marca, dirección del servicio, última fecha de chequeo, tipo de servicio, y una o varias propiedades observables.	Administrador
REQ 13 - Monitorizacion	Se permite visualizar el estado actual de los sensores asociados a los inmuebles mediante la ubicación del archivo .json	Administrador
REQ 14 - Analizar estado de los sensores	Hay un sistema que gestiona la recepción y validación de las señales emitidas por los sensores. Según el tipo de medición del sensor se define la propiedad del mismo para saber cómo y a quién notificar. Las notificaciones se mandan por mail.	Administrador
REQ 15 - Consultas de ingresos por alquiler	Dado un código y un rango de fechas, se listarán todas las reservas realizadas dentro de ese período. Cada reserva incluirá los datos básicos como la fecha de inicio y fin, además de la información del inquilino como nombre, email y cualquier otro detalle relevante. Además, se mostrará el importe total pagado por cada período de reserva. Finalmente, se calculará y mostrará la suma total de ingresos percibidos durante el período especificado.	Administrador, Operario
REQ 16 - Consulta de	Se puede ver un porcentaje de	Administrador,

Ocupaciones	ocupación en un rango de fechas, se agrupa por barrio.	Operario
REQ 17 - Ranking de Inmuebles con Problemas	Se listarán los 15 inmuebles con más problemas registrados dentro de un rango de fechas específico. Para cada inmueble, se indicará la cantidad de ocurrencias de los dos problemas más frecuentes. Esto proporcionará una visión clara de los problemas más recurrentes y permitirá priorizar las acciones de mantenimiento necesarias en los inmuebles identificados.	Administrador, Operario
REQ 18 - Emulación de sensores	Se simula la emisión de datos de sensores, proporciona tanto comportamientos simulados como datos de prueba. Este componente es configurable para ajustar la tasa de emisión de solicitudes por minuto y los tipos de sensores que emiten datos. Además, permite una configuración sencilla de los parámetros para facilitar la simulación de diversas situaciones y escenarios de uso.	-

Resumen de Requerimientos de Atributos de Calidad

Requerimiento	Atributo de Calidad	Descripción
REQ 1 - Login	Seguridad Integridad	Integridad: Seguridad: Se debe restringir el acceso de los usuarios a las características del sistema que no hayan sido autorizados.
REQ 2 - Usuarios del Sistema	Integridad	La integridad en una base de datos de usuarios significa asegurar que los datos sean precisos, completos y consistentes. Esto incluye evitar duplicados, asegurar que los campos como correos electrónicos y fechas sean válidos, mantener relaciones correctas entre tablas (si las hay), y asegurarse de que los estados y atributos de los usuarios sean siempre exactos y actualizados.
REQ 3 - Búsqueda de inmuebles	Rendimiento, escalabilidad	Escalabilidad: Debe manejar incrementos en el número de usuarios y la cantidad de datos sin degradación del servicio. Esto

		implica diseñar una arquitectura que permita escalar horizontalmente (agregar más recursos o servidores) conforme aumente la demanda. Rendimiento: Es crucial que las consultas de búsqueda sean rápidas y eficientes, especialmente al aplicar filtros complejos como fechas y múltiples características de inmuebles. Esto requiere optimización de consultas, uso de índices adecuados y técnicas de caching para resultados frecuentes.
REQ 4 - Alta de reservas	Integridad, disponibilidad, usabilidad	Integridad: Asegurar que los datos recolectados del inquilino y del inmueble sean precisos y completos. Esto garantiza que la reserva se registre correctamente y que la información enviada al administrador y al propietario sea exacta y confiable. Disponibilidad: Garantizar que el sistema esté disponible y accesible para realizar la reserva en cualquier momento. Esto implica una infraestructura robusta y respaldo adecuado para evitar interrupciones en el proceso de reserva. Usabilidad: Facilitar un proceso intuitivo y fluido para el inquilino al proporcionar sus datos y completar la reserva. Esto incluye interfaces amigables que guíen al usuario paso a paso y notificaciones claras que confirmen la reserva tanto para el administrador como para el propietario.
REQ 5 - Realización de pago Online	Seguridad, disponibilidad, modificabilidad	Seguridad: Garantizar la protección de los datos sensibles del usuario durante el proceso de pago, cumpliendo con estándares de seguridad como PCI-DSS (Payment Card Industry Data Security Standard). Esto incluye cifrado de datos, manejo seguro de información personal y transparencia en las políticas de privacidad. Disponibilidad: Asegurar que el sistema de pagos esté siempre accesible para que los usuarios puedan realizar transacciones sin interrupciones. Esto implica una infraestructura robusta, redundancia de servidores y monitoreo constante para evitar caídas del sistema. Modificabilidad: Facilitar la integración y actualización del sistema de pagos

		externo según sea necesario, para adaptarse a cambios en regulaciones o tecnología. Esto requiere una arquitectura flexible y modular, junto con documentación clara y pruebas rigurosas antes de cualquier implementación.
REQ 6 - Cancelación de reservas	Modificabilidad	Modificabilidad: En el proceso de cancelación de reservas por parte del inquilino, se debe asegurar que el sistema pueda ser fácilmente ajustado para: Modificar los términos de la política de cancelación, como los días de anticipación requeridos y el porcentaje de reembolso aplicable. Actualizar el estado de la reserva a "cancelada por el inquilino" de manera automática. Liberar el período del inmueble en el calendario de disponibilidad una vez cancelada la reserva.
REQ 7 - Consulta de reserva propia	Disponibilidad, performance, seguridad	Disponibilidad: El sistema debe estar siempre accesible para que los inquilinos puedan consultar sus reservas en cualquier momento. Esto implica una infraestructura robusta y redundante que minimice el tiempo de inactividad. Rendimiento: Las consultas de reserva deben ser rápidas y eficientes, especialmente al requerir la identificación del inquilino mediante email y código de reserva. Esto se logra mediante optimización de consultas y manejo eficiente de datos. Seguridad: Es fundamental garantizar que la información sensible del inquilino, cómo su email y código de reserva, esté protegida durante todo el proceso de consulta. Esto implica usar métodos de autenticación seguros y asegurar la integridad de los datos para evitar accesos no autorizados.
REQ 8 - Consulta de reservas generales	Disponibilidad, performance, seguridad	Disponibilidad: El sistema debe estar siempre accesible para los administradores y operarios, asegurando que puedan consultar reservas en cualquier momento sin interrupciones. Rendimiento: Las consultas deben ser rápidas y eficientes, especialmente al utilizar múltiples filtros como fecha de

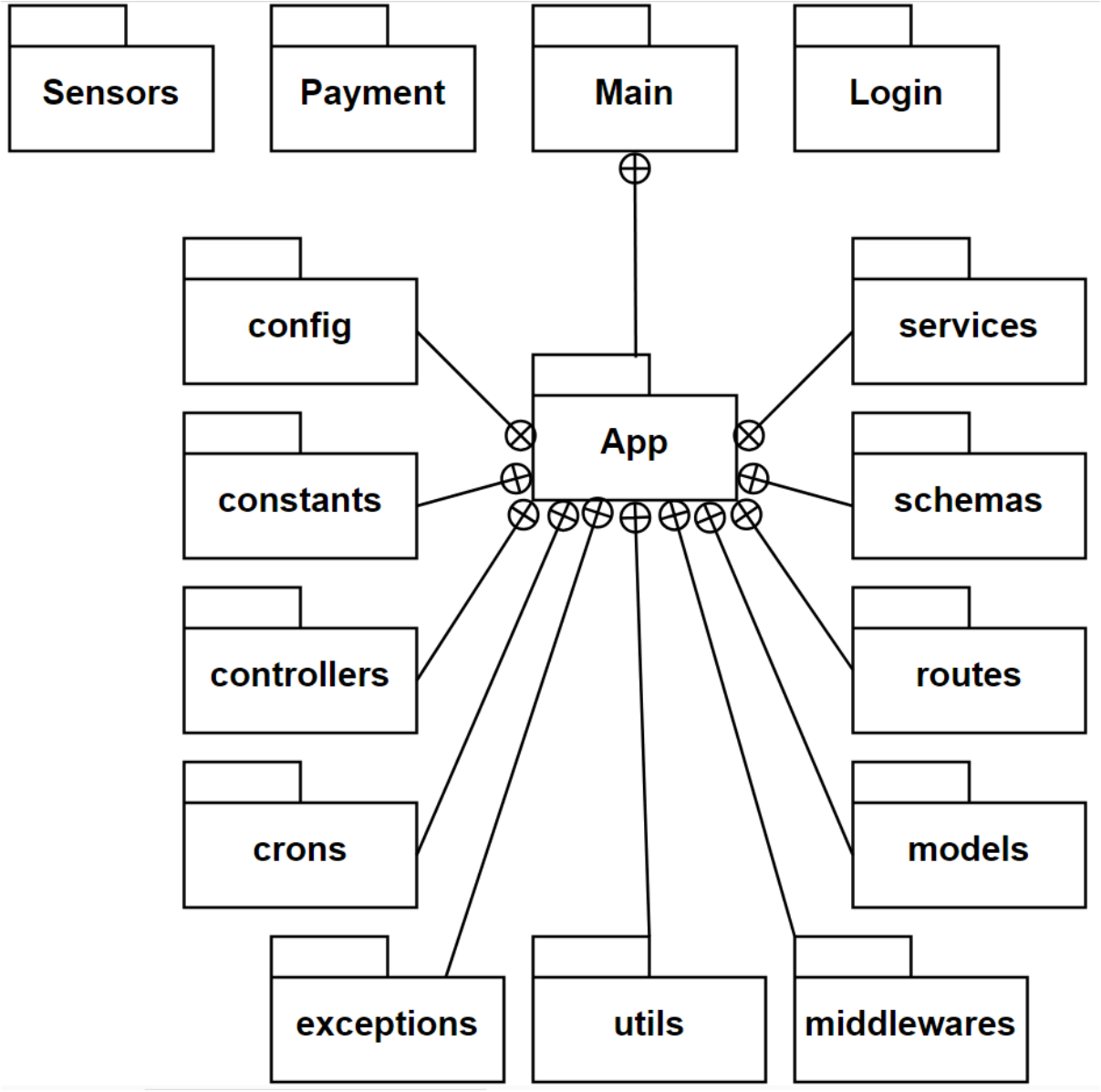
		reserva, código de inmueble, email del inquilino, nombre/apellido del inquilino y estado de la reserva. Seguridad: Se deben implementar medidas robustas para proteger la información confidencial, como autenticación segura y control de acceso, garantizando que solo personal autorizado acceda a los datos de los inquilinos de manera segura y protegida.
REQ 9 - Alta de Inmueble	Integridad, disponibilidad, usabilidad	Integridad: Asegurar que los datos proporcionados del inmueble sean precisos y completos, incluyendo detalles como capacidad, características (aire acondicionado, Wi-Fi, garaje), ubicación y fotos de calidad. Disponibilidad: Mantener el sistema accesible para que los usuarios puedan registrar inmuebles en cualquier momento. Esto implica una infraestructura confiable y redundante que minimice el tiempo de inactividad. Usabilidad: Facilitar un proceso intuitivo y eficiente para que los propietarios puedan registrar sus inmuebles. Esto incluye guiarlos paso a paso, validar la información ingresada y notificar claramente al administrador sobre el estado del inmueble (pendiente de pago o activo).
REQ 10 - Indicar agenda de disponibilidad para el año	Disponibilidad, performance, rendimiento	Disponibilidad: El propietario debe poder configurar fácilmente la disponibilidad del inmueble para fechas futuras. Rendimiento: El sistema debe responder rápidamente al actualizar la agenda de disponibilidad, asegurando eficiencia incluso con muchas fechas involucradas. Performance: El sistema debe procesar rápidamente las actualizaciones de disponibilidad del inmueble, asegurando tiempos de respuesta eficientes sin importar la cantidad de fechas que se estén modificando.
REQ 11 - Agregar sensor a inmueble	Integridad, disponibilidad, seguridad	Integridad: Garantizar que la asignación de sensores sea precisa y completa, evitando errores en los datos. Disponibilidad: Asegurar que el sistema esté siempre disponible para la asignación de sensores. Seguridad: Proteger la información

		sensible durante el proceso de asignación mediante medidas de autenticación y control de acceso.
REQ 12 - Alta de sensor	Integridad, disponibilidad, usabilidad	Integridad: Registrar los datos del sensor de forma precisa y completa para garantizar su fiabilidad. Disponibilidad: Mantener el sistema siempre accesible para registrar sensores en cualquier momento. Usabilidad: Facilitar un proceso intuitivo y eficiente para el administrador al ingresar y gestionar la información del sensor.
REQ 13 - Monitorizacion	Disponibilidad	Disponibilidad: El archivo .json que describe el estado de los sensores debe estar siempre accesible y actualizado, asegurando que los usuarios puedan visualizar la información del estado de los sensores en tiempo real de manera fiable y oportuna.
REQ 14 - Analizar estado de los sensores	Disponibilidad	Idem al anterior
REQ 15 - Consultas de ingresos por alquiler	Rendimiento	Rendimiento: Para poder procesar las consultas de manera eficiente y rápida, especialmente al listar reservas dentro de un rango de fechas específico, calcular importes totales pagados y sumar ingresos percibidos. La idea es optimizar las consultas de base de datos y asegurar tiempos de respuesta adecuados incluso con grandes volúmenes de datos.
REQ 16 - Consulta de Ocupaciones	Eficiencia	Eficiencia: Capacidad del sistema para calcular y mostrar el porcentaje de ocupación en un rango de fechas de manera rápida y sin demoras innecesarias.
REQ 17 - Ranking de Inmuebles con Problemas	Escalabilidad	Escalabilidad: Es importante que el sistema pueda procesar eficientemente y mostrar la información incluso cuando hay una gran cantidad de datos y múltiples usuarios accediendo simultáneamente.
REQ 18 - Emulación de sensores	Disponibilidad, usabilidad	

Documentación de la Arquitectura

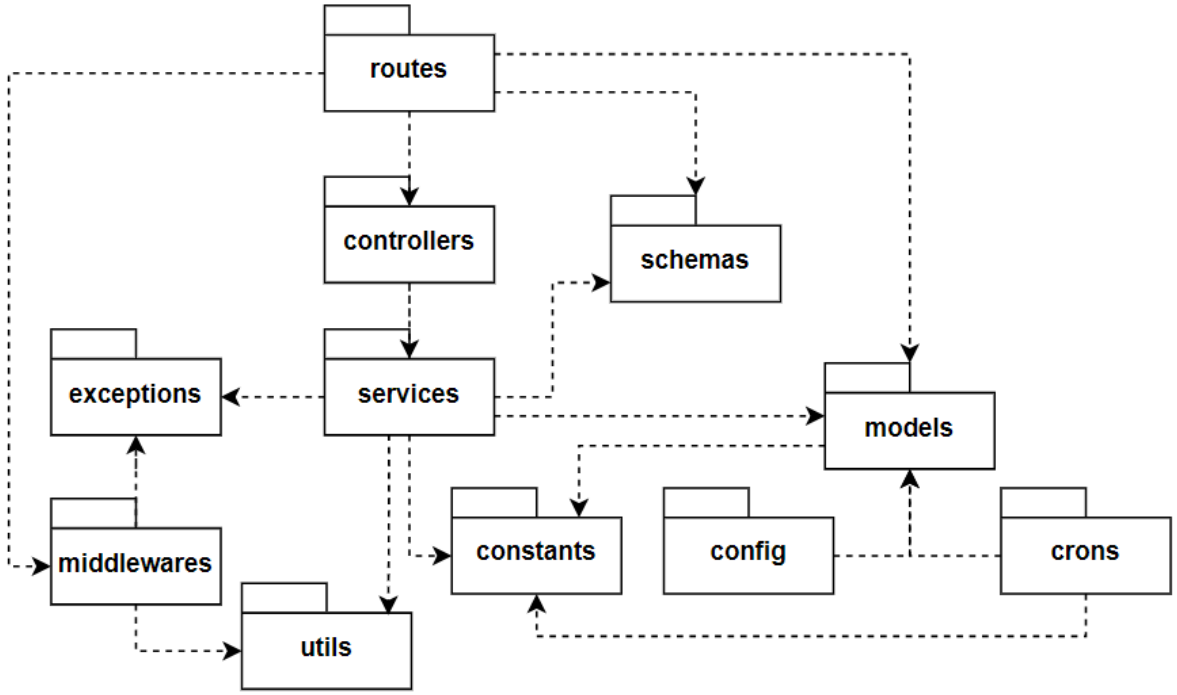
Vistas de modulos

Vista de descomposición



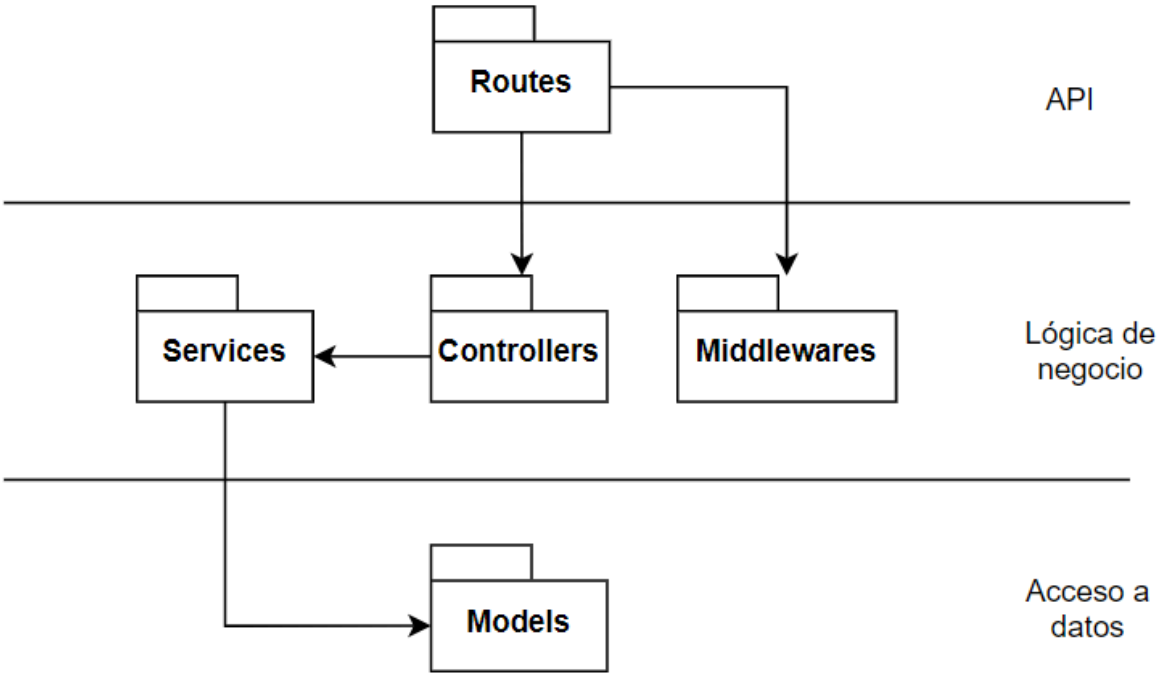
Los servidores Sensors, Payment, Login y Main funcionan de manera similar, pero Main contiene más paquetes, por lo que lo usamos como ejemplo. Luego se va a detallar el rol de cada paquete que forma parte de la aplicación.

Vistas de uso



En esta imagen se observa cómo cada paquete se interconecta dentro de la aplicación del servidor principal.
El paquete Services gestiona todo lo relacionado con la base de datos, incluyendo la gestión de excepciones cuando ocurre algún error, entre otras funciones.
Controllers actúa como intermediario entre Routes y Services, siendo el paso previo a la validación del formato de los datos enviados, permisos de roles, etc., que se realiza en Routes.

Vistas de layers



Esta vista de layers representa el comportamiento de los dos elementos principales del sistema, tanto Main como Sensors. Los elementos Payment y Login son auxiliares, emulan el comportamiento de servicios externos, por lo que no tienen representación en este diagrama.

Catálogo de elementos

Elemento	Responsabilidades
Main	Sistema principal de Inmo2.0
Sensor	Sistema secundario de manejo, guardado y simulación de sensores
Login	Sistema externo de ingreso a la plataforma
Payment	Sistema externo de pagos online.
Config	Maneja configuración de Bases de Datos, manejo de archivos (tanto lectura como escritura) entre otros.
Constants	Elementos constantes en Main Server se encuentran acá
Controllers	Toman el pedido de las request y las conectan con las correspondientes funciones de la lógica de negocio.
Crons	Específicamente para chequear pagos pendientes en Main Server

Elemento	Responsabilidades
Exceptions	Las posibles excepciones que pueden surgir están definidas aca
Middlewares	Validación, autenticación de los datos, manejo de errores
Models	Modelos de los objetos para las DBs
Routes	Puntos de entrada al sistema
Schemas	Validación del lado del cliente de cómo se envían los datos
Services	Contiene la lógica de negocio

Vistas de Componentes y Conectores

Externa

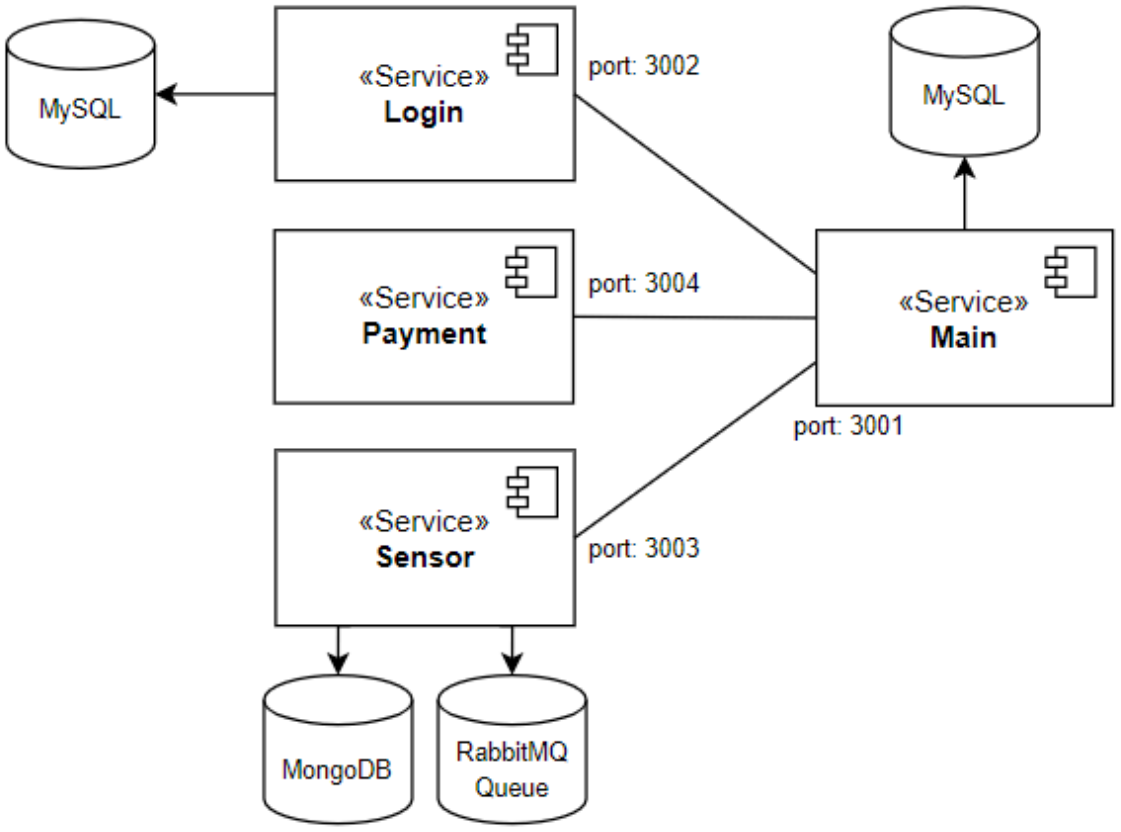
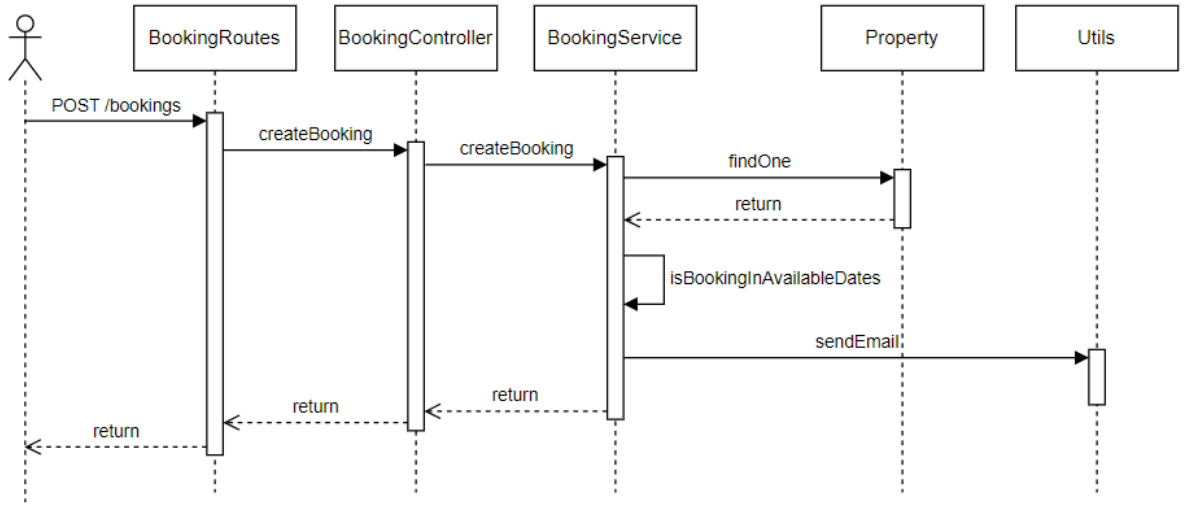


Diagrama de Comportamiento

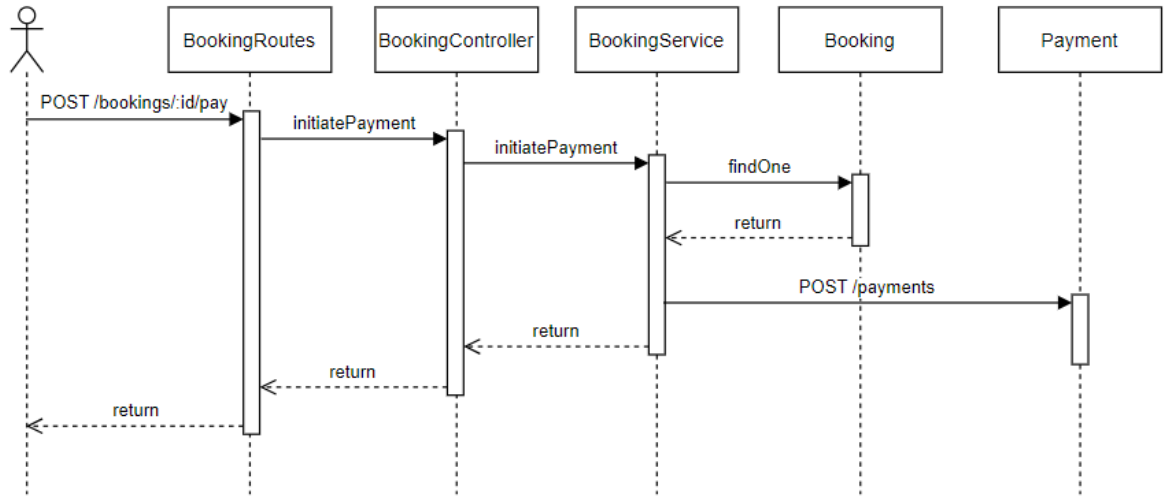
La creación de un booking por parte de un usuario inquilino toma varios pasos, por lo que se separa en distintos diagramas de secuencia. Hay que considerar que tanto para la creación del booking como para el inicio del pago, es necesario que el usuario proporcione un token válido, esto se omitió por legibilidad de los diagramas. Además, en las routes, previo a enviar la información de la solicitud al controller, se hace una validación de los datos que vienen en el body de la request, esto también se omitió por legibilidad.

Pipeline de creación de un booking



Se ilustra el caso de creación de un booking con datos válidos, en una propiedad que está disponible para las fechas indicadas. El booking resultante de esta acción queda con status ‘pending’

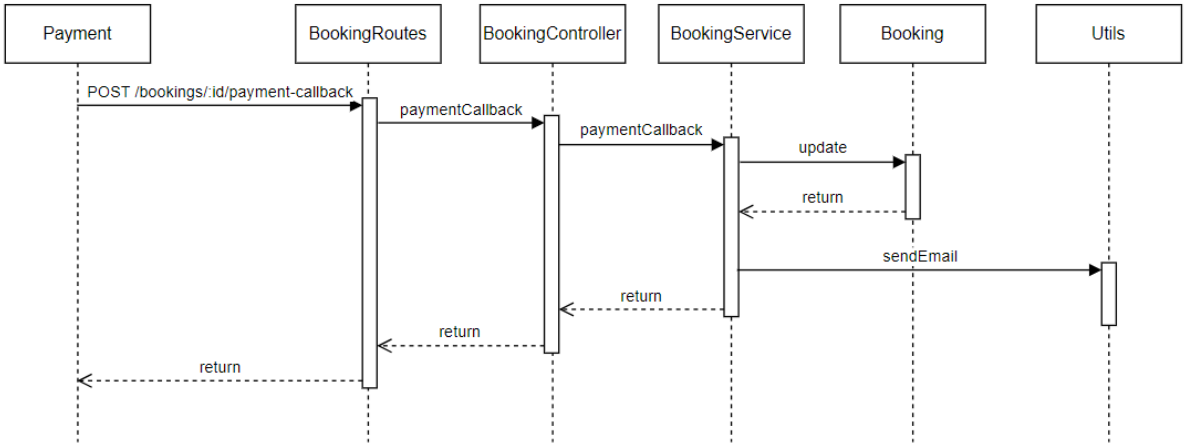
Pipeline de inicio de pago de un booking



Se muestra el caso en que el payment inicia correctamente. El resultado de esta acción es pegarle al servicio externo Payment para indicar un intento de realizar un pago. La acción

del servicio externo no bloquea la ejecución de nuestro servicio Main principal, ya que se le proporciona un endpoint de callback el cual utilizará cuando termine su procesamiento.

Pipeline de callback por parte de Payment de un booking



Finalmente se muestra el caso de exito de payment de un booking, esta acción es lanzada por el servicio Payment mediante el endpoint de callback que se le proporcionó previamente. Luego de esto el booking creado queda con status ‘active’,

Decisiones de diseño

MySQL

Se optó por utilizar MySQL para las bases de datos del Main Server y del Login debido a sus características como sistema de gestión de bases de datos relacional.
A diferencia de MongoDB, MySQL ofrece ventajas significativas en entornos donde la estructura y la coherencia de los datos son fundamentales. Las bases de datos relacionales como MySQL son ideales para aplicaciones que requieren transacciones complejas, integridad referencial y consultas estructuradas.
Esta decisión asegura que nuestras bases de datos puedan manejar eficientemente grandes volúmenes de datos con un alto nivel de confiabilidad y consistencia, siendo una elección sólida para aplicaciones críticas como el Main Server y el sistema de Login.

MongoDb

Se optó por MongoDB como la solución de almacenamiento para los datos de los sensores en Sensor Server debido a que es no relacional. A diferencia de los sistemas de bases de datos relacionales como MySQL, MongoDB está diseñado específicamente para manejar datos no estructurados y semi-estructurados de manera eficiente.
MongoDB ofrece flexibilidad en el esquema de datos y permite una escalabilidad horizontal sencilla, lo cual es crucial para manejar grandes volúmenes de datos generados por múltiples sensores en tiempo real. Esta decisión garantiza que Sensor Server pueda gestionar de manera efectiva la captura, almacenamiento y consulta de datos de sensores.

Manejo de errores

Implementamos funciones dedicadas para gestionar varios tipos de errores que pueden surgir debido al uso inadecuado del sistema. Entre estos errores se incluyen el error de solicitud incorrecta (Bad Request Error) y el error de recurso no encontrado (Not Found Error), entre otros casos específicos.
Cada función está diseñada para identificar errores y devolver un mensaje explicativo y breve que detalla la causa específica que lo provocó. Esta estrategia no solo mejora la experiencia del usuario al proporcionar retroalimentación clara y contextual sobre los errores encontrados, sino que también facilita el proceso de depuración y resolución de problemas para los desarrolladores del sistema.
Así, aseguramos una gestión efectiva y transparente de errores, promoviendo un uso más seguro y eficiente de nuestra plataforma.

Queue Publish Subscribe con prioridad

Para la notificación y validación de cambios en los valores de los sensores, se decidió utilizar una cola de mensajes con prioridad implementada con RabbitMQ, diseñada bajo el modelo de publisher-subscriber.
Esta elección ofrece numerosas ventajas. RabbitMQ permite una comunicación eficiente y asíncrona entre los diferentes componentes del sistema, asegurando que los mensajes se entreguen de manera oportuna y en el orden de prioridad adecuado.
El modelo publisher-subscriber facilita una arquitectura desacoplada, donde los productores de mensajes (publishers) y los consumidores (subscribers) operan de manera independiente, mejorando la escalabilidad y la flexibilidad del sistema. Además, RabbitMQ proporciona robustas características de fiabilidad y durabilidad, garantizando que los mensajes críticos no se pierdan y se gestionen adecuadamente incluso en caso de fallos.
Esta configuración es ideal para manejar los datos dinámicos y sensibles de los sensores, asegurando que cualquier cambio en los valores se procese y notifique eficientemente, manteniendo la integridad y actualidad de la información en todo momento.

Autenticacion - JWT

Para la validación de usuarios logueados y la determinación de las rutas que cada usuario puede acceder según su rol, se decidió utilizar JSON Web Tokens (JWT) y delegar el almacenamiento de sesión/login al Login Server, manteniéndolo externo al Main Server.
Esta decisión ofrece varias ventajas. JWT permite la autenticación y autorización de manera segura y eficiente, ya que los tokens contienen toda la información necesaria sobre el usuario y sus permisos, lo que elimina la necesidad de almacenar información de sesión en el servidor. Al gestionar las sesiones y logins en el Login Server, se mejora la seguridad y la modularidad del sistema, permitiendo que el Main Server se enfoque exclusivamente en la lógica de negocio y el manejo de datos.
Esto no solo reduce la carga en el Main Server, sino que también facilita la escalabilidad y el mantenimiento del sistema, ya que cualquier cambio en la autenticación o en las políticas de acceso puede ser manejado directamente en el Login Server sin afectar al resto de la aplicación. Además, esta separación refuerza la seguridad al centralizar el control de acceso en un componente dedicado, disminuyendo la superficie de ataque potencial.

Al principio se consideró usar Auth0 para la validación de roles pero luego de varios intentos y de investigación, al ser más enfocado con el uso de frontend, se dificulta el setup del mismo.

PM2 y K6

Se decidió utilizar PM2 como balanceador de carga (load balancer) para mejorar el rendimiento y la disponibilidad del sistema, y K6 para probar el rendimiento, ya que esta combinación es una elección acertada para la simulación y el registro de datos.

PM2 es una herramienta de administración de procesos para aplicaciones Node.js que no solo facilita el monitoreo y la gestión de las aplicaciones en producción, sino que también optimiza el uso de los recursos del servidor mediante el balanceo de carga automático. Esto asegura que las solicitudes se distribuyan equitativamente entre las instancias disponibles, mejorando la capacidad de respuesta y la resiliencia del sistema frente a fallos.

Por otro lado, K6 es una herramienta de pruebas de carga moderna que permite simular condiciones de alta carga y evaluar el rendimiento del sistema bajo diferentes escenarios. Su facilidad de uso y su capacidad para generar informes detallados ayudan a identificar cuellos de botella y optimizar la infraestructura antes de que los problemas afecten a los usuarios finales.

La combinación de PM2 y K6 garantiza que el sistema no solo sea altamente disponible y eficiente, sino que también esté bien preparado para manejar picos de carga y mantener un rendimiento óptimo bajo diversas condiciones operativas. Esta estrategia integral de manejo y prueba del rendimiento es crucial para la simulación precisa y el registro confiable de datos.

ADRs para las Decisiones de Diseño

ADR para: Elección del Sistema de Gestión de Bases de Datos

Título: Selección de MySQL para las Bases de Datos del Main Server y del Login

Status: accepted

Contexto y Declaración del Problema

- Necesidad de un sistema de gestión de bases de datos (SGBD) para manejar las bases de datos del Main Server y del sistema de Login.
- Requerimiento de alta coherencia y estructura de los datos.
- Importancia de la integridad referencial y transacciones complejas para la aplicación.
- Necesidad de manejar grandes volúmenes de datos con fiabilidad y consistencia.

Factores Decisivos

- Coherencia y estructura de datos ofrecida por un SGBD relacional.
- Capacidad para manejar transacciones complejas y asegurar la integridad referencial.

- Fiabilidad y consistencia en el manejo de grandes volúmenes de datos.
- Ventajas de MySQL en comparación con otros SGBD como MongoDB en estos aspectos.

Opciones Consideradas

Opción 1: MySQL

Resultado de la Decisión: Se seleccionó MySQL como el SGBD para las bases de datos del Main Server y del sistema de Login.

Consecuencias Positivas:

- Alta coherencia y estructura de datos, asegurando que los datos se mantengan consistentes y fiables.
- Capacidad de manejar transacciones complejas y mantener la integridad referencial.
- Eficiencia en el manejo de grandes volúmenes de datos.
- Alta fiabilidad y consistencia, adecuada para aplicaciones críticas.

Consecuencias Negativas:

- Potencialmente mayor complejidad en la gestión y administración en comparación con bases de datos NoSQL.
- Puede requerir más recursos en términos de hardware y administración.

Opción 2: MongoDB

Resultado de la Decisión: No se seleccionó MongoDB.

Consecuencias Positivas:

- Mayor flexibilidad en el manejo de datos no estructurados.
- Posiblemente más sencillo de escalar horizontalmente.

Consecuencias Negativas:

- Menor capacidad para manejar transacciones complejas y mantener la integridad referencial.
- Menor consistencia y fiabilidad en comparación con MySQL para las necesidades específicas del Main Server y del sistema de Login.

ADR para la Selección de MongoDB como Solución de Almacenamiento para Datos de Sensores en Sensor Server

Título: Selección de MongoDB como Solución de Almacenamiento

Status: accepted

Contexto y Declaración del Problema: Sensor Server necesita una solución eficiente para almacenar datos de sensores, que son no estructurados y generados en tiempo real por múltiples dispositivos. Es crucial seleccionar una base de datos que soporte escalabilidad horizontal y maneje eficientemente grandes volúmenes de datos no estructurados.

Factores Decisivos:

- Necesidad de manejar datos no estructurados y semiestructurados eficientemente.
- Escalabilidad horizontal para soportar crecimiento futuro.
- Requerimiento de consultas rápidas y eficientes sobre grandes volúmenes de datos en tiempo real.

Opciones Consideradas:

Opción 1: MongoDB

Resultado de la Decisión: Se optó por implementar MongoDB como la solución de almacenamiento para los datos de sensores en Sensor Server.

Consecuencias Positivas:

- MongoDB es una base de datos no relacional diseñada específicamente para manejar datos no estructurados y semi-estructurados de manera eficiente.
- Ofrece flexibilidad en el esquema de datos, lo cual es ideal para los datos variables y no estructurados generados por los sensores.
- Permite una escalabilidad horizontal sencilla mediante la distribución de datos en múltiples servidores, lo que asegura que Sensor Server pueda manejar el crecimiento de datos sin problemas.
- Facilita consultas rápidas sobre grandes volúmenes de datos en tiempo real, crucial para las aplicaciones de monitoreo y control de sensores.

Consecuencias Negativas:

- Requiere habilidades específicas para administrar y optimizar su rendimiento, lo cual podría implicar un costo adicional en términos de recursos humanos especializados.

ADR para la Gestión de Cambios en los Valores de Sensores Utilizando RabbitMQ

Título: Utilización de RabbitMQ para la Notificación y Validación de Cambios en Sensores
Status: accepted

Contexto y Declaración del Problema: Para asegurar la notificación oportuna y la validación eficiente de cambios en los valores de los sensores dentro del sistema, es crucial implementar una solución que garantice la entrega de mensajes en tiempo real y en el orden de prioridad adecuado.

Factores Decisivos:

- Necesidad de comunicación asíncrona y eficiente entre los componentes del sistema.
- Requerimiento de una arquitectura desacoplada que mejore la escalabilidad y flexibilidad.
- Garantía de fiabilidad y durabilidad para la gestión de mensajes críticos, incluso en casos de fallos del sistema.

Opciones Consideradas:

Opción 1: Utilización de RabbitMQ con un modelo publisher-subscriber para la gestión de mensajes.

Resultado de la Decisión: Con la implementación de RabbitMQ utilizando el modelo publisher-subscriber para la gestión de mensajes de cambios en los valores de los sensores, se espera alcanzar los siguientes resultados:

Consecuencias Positivas:

- Comunicación eficiente y asíncrona entre los componentes del sistema.
- Arquitectura desacoplada que mejora la escalabilidad y flexibilidad del sistema.
- Garantía de entrega y orden de mensajes mediante la cola de mensajes con prioridad de RabbitMQ.
- Fiabilidad y durabilidad mejoradas para la gestión de mensajes críticos.

Consecuencias Negativas:

- Requiere configuración inicial y mantenimiento de RabbitMQ, lo cual implica recursos adicionales.

ADR para: Uso de JSON Web Tokens (JWT) y delegación de gestión de sesión al Login Server

Título: Implementación de JSON Web Tokens (JWT) y delegación de gestión de sesión

Status: accepted

Contexto y Declaración del Problema: Para la validación de usuarios logueados y la gestión de rutas basadas en roles dentro de nuestra aplicación, se enfrentó la necesidad de adoptar una solución que garantizara seguridad, eficiencia y escalabilidad. Además, se buscaba mantener separada la lógica de autenticación y sesión del servidor principal (Main Server) para mejorar la modularidad y reducir la carga operativa.

Factores decisivos:

- Seguridad de autenticación y autorización.
- Eficiencia en la gestión de sesiones.
- Modularidad y mantenimiento del sistema.
- Escalabilidad y manejo de cambios en políticas de acceso.

Opciones Consideradas:

Opción 1: Implementación de JSON Web Tokens (JWT) y delegación de gestión de sesión al Login Server.

Resultado de la Decisión: Se decidió implementar JSON Web Tokens (JWT) para la autenticación y autorización, y delegar la gestión de sesión al Login Server.

Consecuencias Positivas:

- Mejora significativa en la seguridad del sistema al eliminar el almacenamiento de información de sesión en el Main Server.
- Incremento en la eficiencia operativa al reducir la carga de gestión de sesiones en el Main Server.
- Aumento de la modularidad al separar las responsabilidades entre autenticación y lógica de negocio.
- Facilitación del mantenimiento y escalabilidad al permitir cambios en la autenticación y políticas de acceso de manera centralizada en el Login Server.

Consecuencias Negativas:

- Requiere una integración robusta y precisa entre el Main Server y el Login Server para asegurar la consistencia y disponibilidad del sistema.
- Dependencia crítica del Login Server, lo cual puede introducir puntos únicos de falla si no se implementan medidas adecuadas de redundancia y seguridad en dicho componente.

Opción 2: Implementación de Auth0 para el manejo de autenticación de roles.

Resultado de la Decisión: Se consideró inicialmente la opción de implementar Auth0 para la validación de roles y gestión de autenticación.

Consecuencias Positivas:

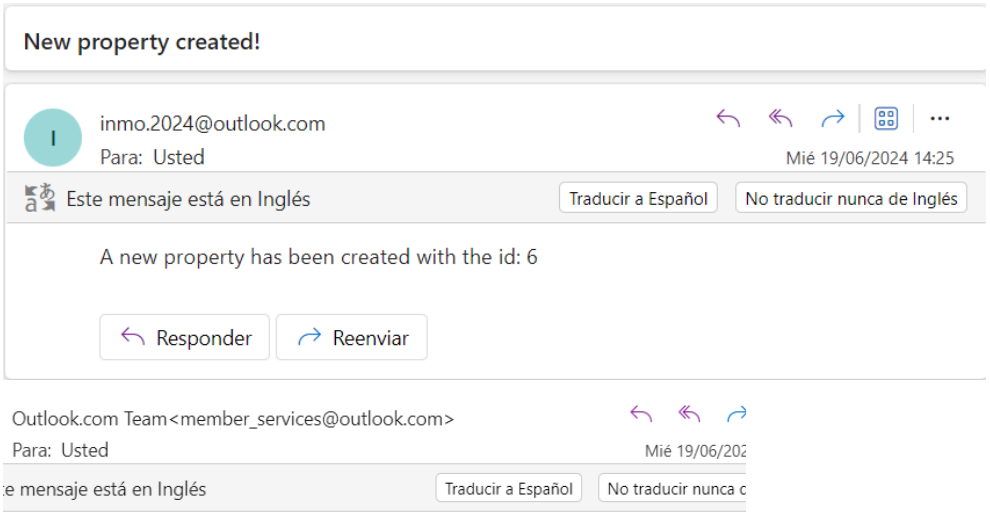
- Integración simplificada y rápida debido a las capacidades preconfiguradas y documentación robusta de Auth0.
- Amplia gama de funcionalidades adicionales como gestión de usuarios, autenticación multifactor y cumplimiento de estándares de seguridad.

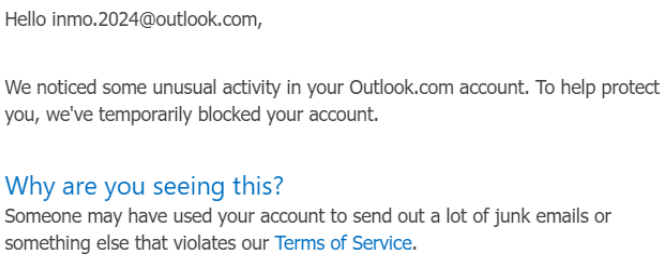
Consecuencias Negativas:

- Limitaciones en la personalización y adaptabilidad a requerimientos específicos de la aplicación. En este proyecto no se tiene Frontend, y Auth0 tiene un enfoque hacia el uso de Frontend.
- Potencial dependencia externa significativa para aspectos críticos de autenticación y autorización.
- Costos asociados con el uso de servicios externos como Auth0 podrían ser más altos a largo plazo en comparación con una solución interna.

Errores y Problemas Conocidos

- El envío de mail para notificaciones se pudo implementar correctamente, pero para esto se creó una cuenta de Outlook para usar como remitente y como destinatario, la cual fue bloqueada por Microsoft por comportamiento extraño. De todas formas si se crea una cuenta nueva y se dan los parámetros de la nueva cuenta en el .env, se podrá comprobar que funciona sin problemas.

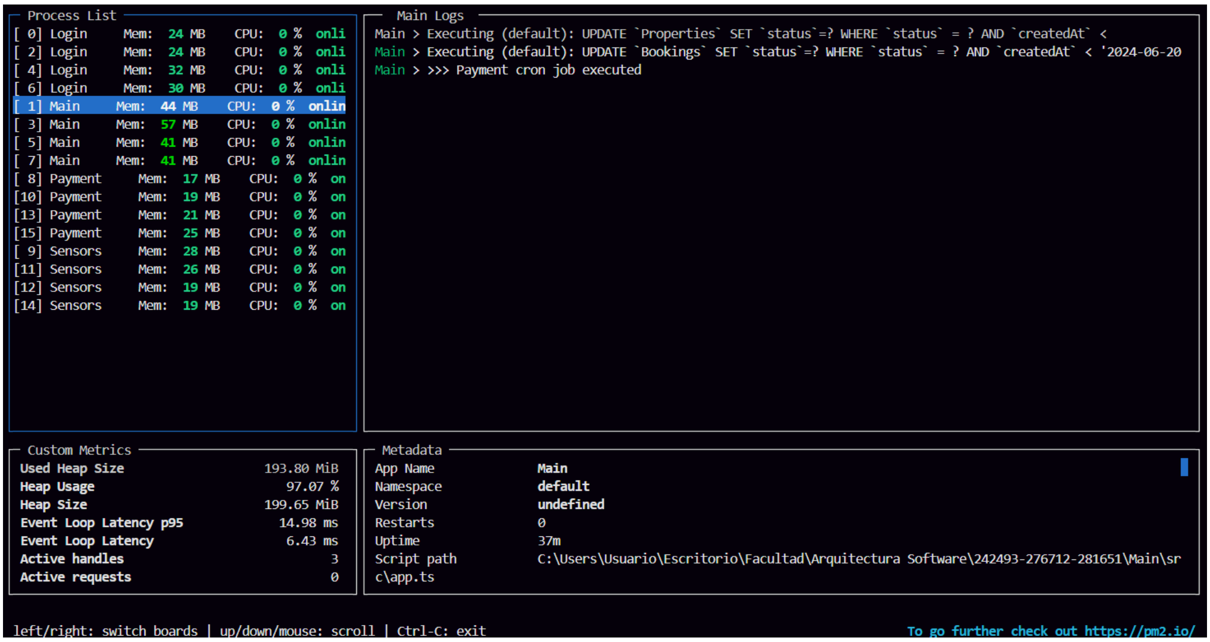




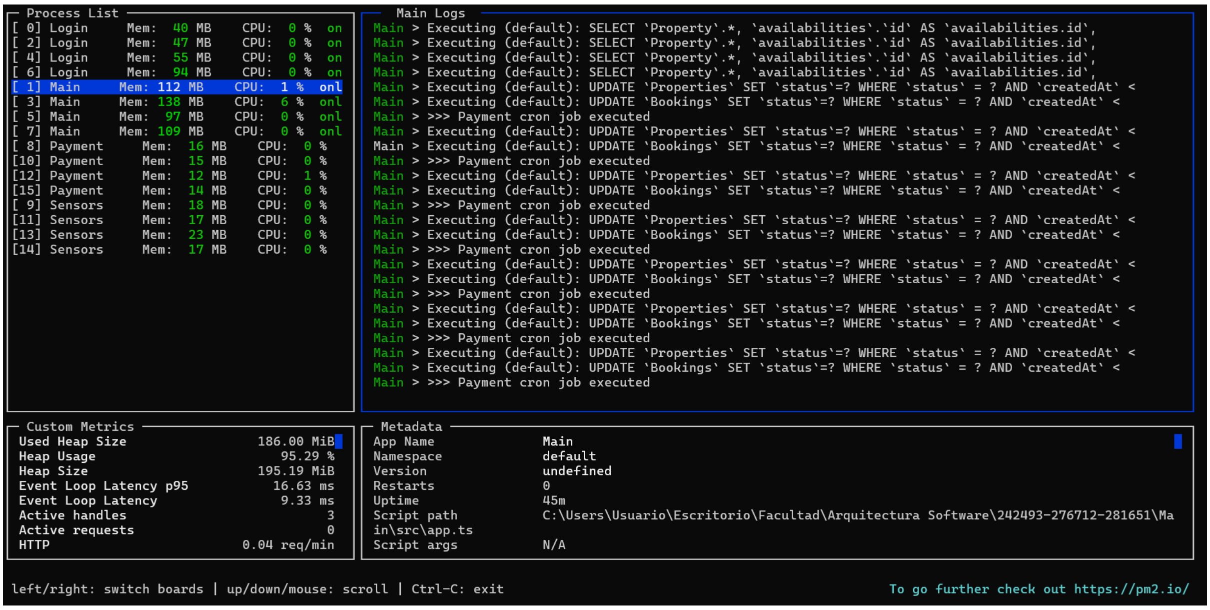
- Los requerimientos N16 y N17 no fueron implementados

Pruebas de carga

Se utilizó K6 para hacer la prueba de carga y evaluar el performance del sistema ante un cantidad de solicitudes elevada en determinado periodo de tiempo.



Se cumplió el requerimiento de realizar 1000 reservas por minuto con una latencia promedio de 500 milisegundos



```
    data_received..................: 34 kB 557 B/s
    data_sent......................: 34 kB 560 B/s
    http_req_blocked...............: avg=157.11µs min=0s      med=0s      max=8.89ms   p(90)=0s       p(95)=0s
    http_req_connecting............: avg=9.45µs   min=0s      med=0s      max=539.1µs  p(90)=0s       p(95)=0s
    http_req_duration..............: avg=47.81ms  min=23.56ms med=33.33ms max=518.43ms p(90)=64.69ms  p(95)=86.34ms
      { expected_response:true }...: avg=47.81ms  min=23.56ms med=33.33ms max=518.43ms p(90)=64.69ms  p(95)=86.34ms
    http_req_failed................: 0.00% ✓ 0       ✗ 57
    http_req_receiving.............: avg=209.15µs min=0s      med=0s      max=1.77ms   p(90)=992.24µs p(95)=1ms
    http_req_sending...............: avg=58.52µs  min=0s      med=0s      max=1.09ms   p(90)=180.56µs p(95)=503.92µs
    http_req_tls_handshaking.......: avg=0s       min=0s      med=0s      max=0s       p(90)=0s       p(95)=0s
    http_req_waiting...............: avg=47.55ms  min=23.56ms med=33.33ms max=516.79ms p(90)=64.69ms  p(95)=86.06ms
    http_reqs......................: 57    0.94483/s
    iteration_duration.............: avg=1.05s    min=1.02s   med=1.03s   max=1.6s     p(90)=1.07s    p(95)=1.09s
    iterations.....................: 57    0.94483/s
    vus............................: 1     min=1     max=1
    vus_max........................: 1     min=1     max=1


running (1m00.3s), 0/1 VUs, 57 complete and 0 interrupted iterations
default ✓ [ 100% ] 0/1 VUs  1m0s
k6 exited with exit code 0
```

Para la ejecución de estas pruebas, se debe seguir los siguientes pasos:

1. Levantar el sistema con `pm2 start ecosystem.config.js`
2. Hacer login como usuario Tenant mediante la coleccion de postman y copiar el token id.
3. Usar este token id en los tests `Main\src\tests\loadTestCreateBooking.ts` y `Main\src\tests\loadTestGetProperties.ts`
4. Correr los tests anteriores usando la extension de k6 para VSCode.