Proyecto Rest API

Documentación

Integrantes:

- Gustavo Oyarce Peralta
- Nicolás Jara Carvajal
- Tomás Lillo Silva

Asignatura: Computación Paralela y Distribuida

Sección: 411

Docente: Sebastian Salazar Molina

Contenidos

Introducción

En la actualidad, la arquitectura REST es un estándar fundamental para el desarrollo de servicios web eficientes, interoperables y escalables. Este proyecto tiene como objetivo utilizar la arquitectura REST (Transferencia de Estado Representacional) con fin de desarrollar servicios web, este servicio utilizará la autenticación de Google (oauth 2.0) lo que entrega diferentes tipos de beneficios de seguridad como el proceso de autorización con consentimiento esto ayuda a proteger la privacidad de los usuarios. Por otro lado, entrega una gran escalabilidad debido a que google es el encargado de administrar una mayor llegada de usuarios.

Objetivos

- Comprender el funcionamiento del protocolo HTTP (sus verbos y estados).
- Comprender el funcionamiento de aplicaciones stateless, mecanismos asíncronos y funcionamiento REST.
- Diseñar e implementar un proyecto de software con requerimientos específicos.

Especificaciones

En el escenario supuesto de que la Universidad Tecnológica Metropolitana, está evaluando una forma de implementar un sistema para evaluar cualitativamente las distintas clases que se realizan en la universidad.

Autenticación

El sistema debe contener los usuarios de la UTEM, se necesita que el mecanismo de autenticación sea el provisto por Google (oauth 2.0), lo que garantiza la validez de la autenticación, es necesario además que exista un manejo de roles, para una personalización futura. Cada llamada a los servicios api, debe identificar al usuario y tener un access token, para validar la llamada contra los servicios de Google, esto se debe realizar un token bearer jwt.
La implementación oauth no es obligatoria (pero su uso sí), el sistema debe validar el token jwt, para asegurar la integridad de la autenticación Google.

Tecnologías Usadas

Node JS

En este proyecto se utiliza Node.js, debido a que por ser el entorno de ejecución de JavaScript es más común en entornos de la web. Una API (Application Programming Interface) es una interfaz que permite la comunicación entre diferentes aplicaciones o servicios, mediante el intercambio de datos en un formato estandarizado. Hacer APIs con Node.js es más sencillo, ya que se puede usar el mismo lenguaje tanto en el lado del servidor como en el del cliente. Node.js permite crear aplicaciones web escalables y eficientes, aprovechando las ventajas de JavaScript como lenguaje dinámico y multiplataforma. Además, Node.js cuenta con una gran comunidad de desarrolladores que aportan librerías y módulos para facilitar el desarrollo de diferentes tipos de proyectos.

Express

En el proyecto de API se utiliza Express porque es un framework de Node.js que facilita el desarrollo de aplicaciones web y servicios RESTful. Esta herramienta fue usada debido a que ofrece una serie de ventajas, como:

- Una arquitectura basada en middleware que permite modularizar el código y reutilizar funcionalidades.

- Una sintaxis sencilla y flexible que permite crear rutas, manejar peticiones y respuestas, y configurar el servidor.

- Una gran variedad de paquetes y extensiones disponibles en el ecosistema de Node.js que amplían las capacidades de Express.

- Una alta compatibilidad con bases de datos, tanto relacionales como no relacionales, mediante el uso de drivers o librerías específicas.

- Una buena documentación y una amplia comunidad de desarrolladores que ofrecen soporte y recursos.

Por estas razones, Express es una opción muy popular y adecuada para crear APIs robustas, escalables y eficientes.

FL0

La API está alojada en fl0, una plataforma web que permite desplegar la API de forma fácil y rápida en la nube. Esto significa que el sitio web se comunica con un servidor remoto que procesa las solicitudes y devuelve los datos necesarios para mostrar el contenido. Fue escogida debido a que Fl0 ofrece una solución simple y eficiente para crear y gestionar sitios web dinámicos sin necesidad de configurar servidores ni bases de datos. También debido al poco tiempo de hacer la implementación al servidor entregado y fue una alternativa intuitiva y

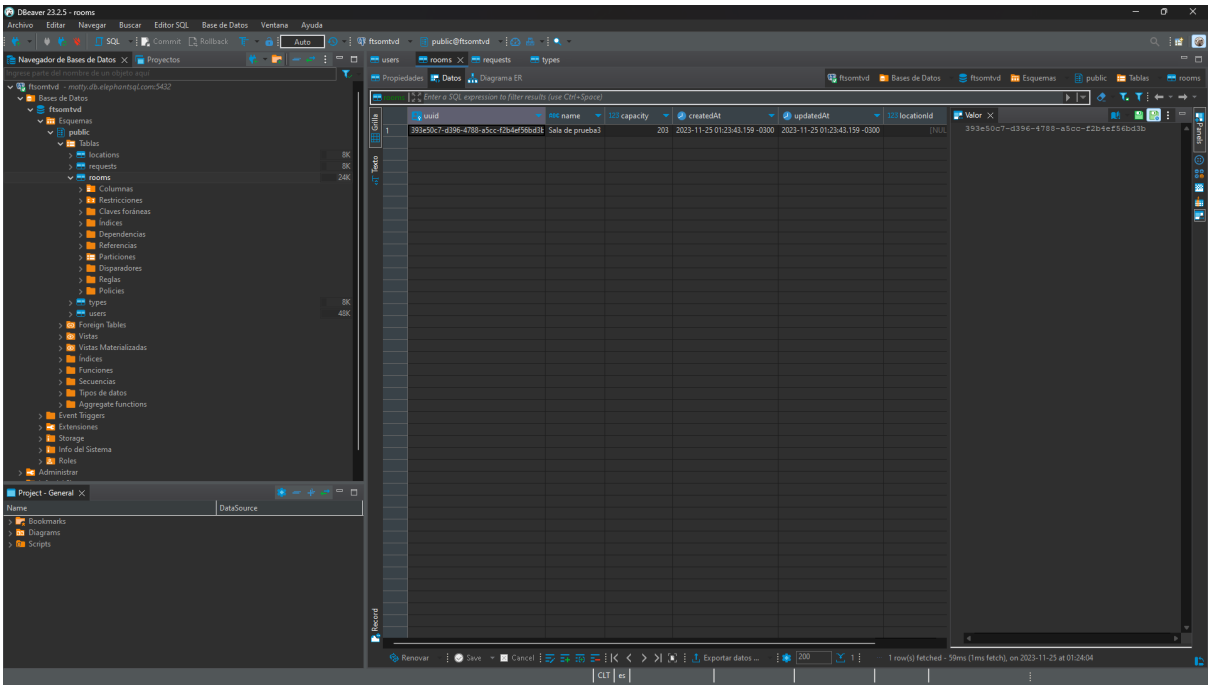
PostgreSQL

Base de datos, se usó principalmente para crear la base de datos, pero que luego fue transferida a ElephantSQL, aunque conservando la estructura y utilizando Postgres en el servidor.

DBeaver

DBeaver es una herramienta de administración de bases de datos que proporciona un entorno gráfico para trabajar con diversas bases de datos. Fue escogida debido a que soporta múltiples bases de datos entre ellas PostgreSQL, además nos proporciona una interfaz gráfica que facilita la administración y consulta de bases de datos. Se pueden explorar las tablas, ejecutar consultas SQL, y gestionar los objetos de la base de datos de una manera visual.

Ilustración 2: "Administrador de Base de Datos"

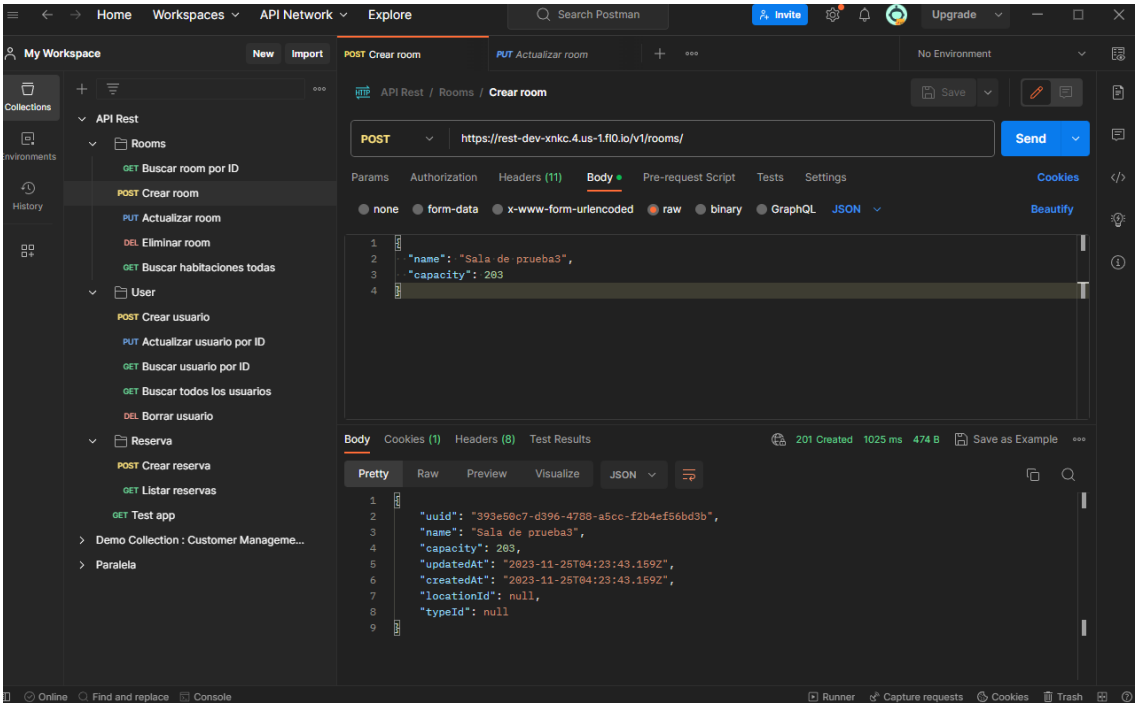


Fuente: Elaboración propia en DBeaver

Postman

Postman se utilizó principalmente para hacer las pruebas automatizadas del proyecto REST, para ello se escribieron scripts de prueba en JavaScript para automatizar la verificación del comportamiento de las respuestas de la API. Esto es útil para garantizar la calidad y la integridad de las API.

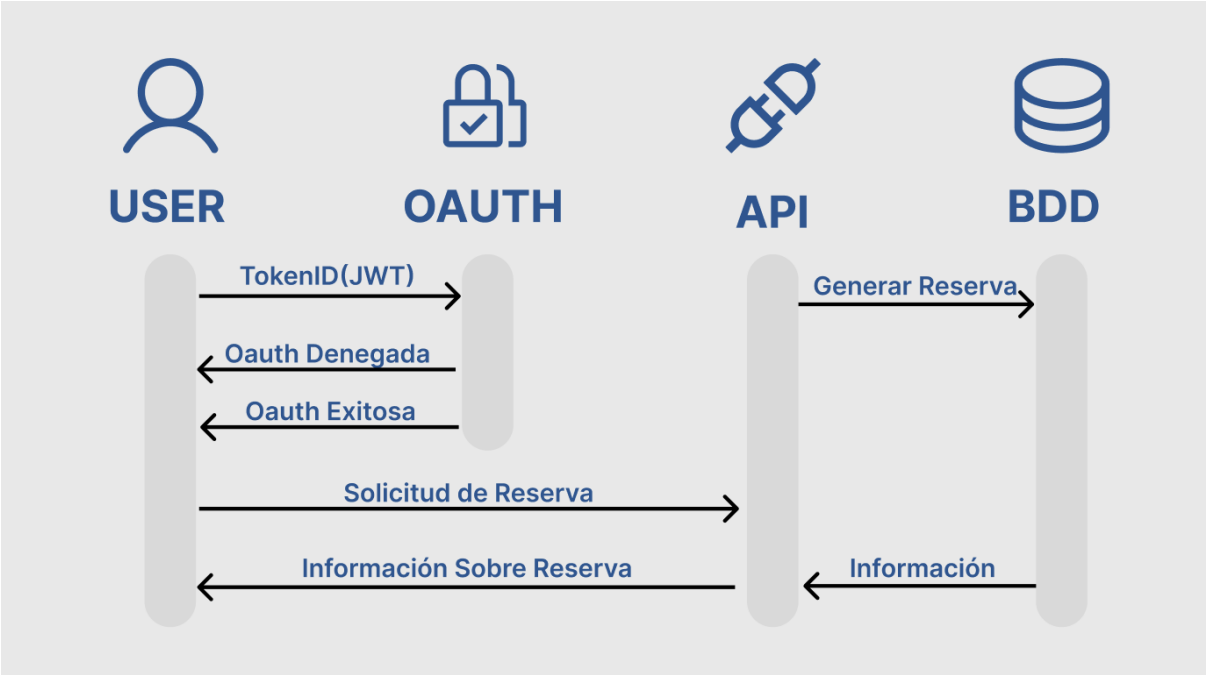
Ilustración 3: “Prueba en Postman”



Fuente: Elaboración propia en Postman

Diagramas de secuencia

Ilustración 4: “Diagrama de Secuencia”

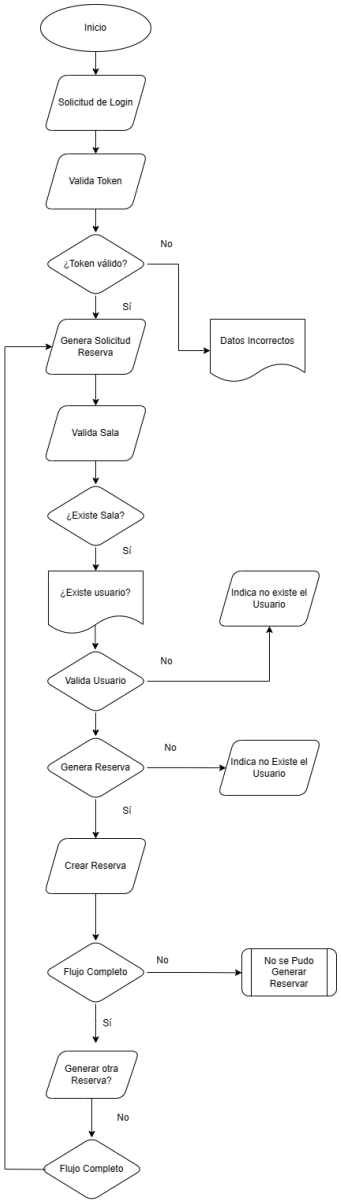


Fuente: Elaboración propia

En este diagrama se puede observar la secuencia del proyecto, en el cual el el usuario hace la entrega del JWT para la autenticación, con lo cual el servicio de google autoriza o deniega la autenticación, si es exitosa puede hacer la solicitud de reserva para lo cual genera la reserva y entrega la información a la API para así dar a conocer al usuario la información de la reserva.

Modelo de Procesos

Ilustración 5: “Modelo de Procesos”

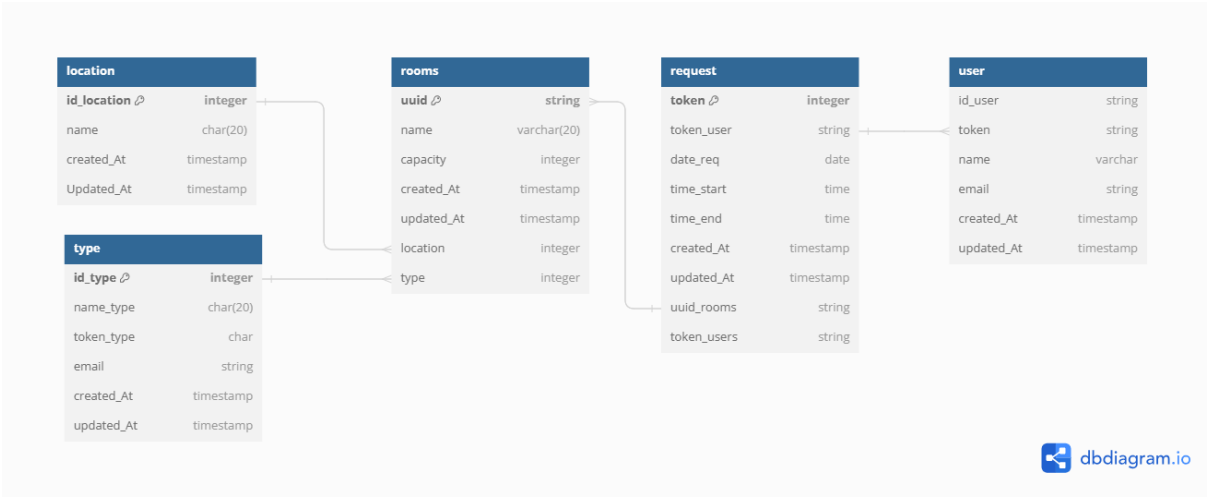


Fuente: Elaboración propia

En esta ilustración se puede observar el proceso del proyecto, desde la solicitud de login hasta cuando se crea y genera la reserva. Para ello previamente se valida el token, la existencia de salas y por último valida el usuario.

Diagramas de Entidad Relación

Ilustración 6: “Diagrama Entidad Relación”



Fuente: Elaboración propia con DbDiagram.io

En este diagrama se puede observar la estructura lógica de la base de datos. Se pueden observar las llaves primarias de id_location en la tabla location, id_type en la tabla type, uuid en la tabla rooms, token en la tabla request. Por otro lado las llaves foráneas type y location de la tabla rooms, por otro lado uuid_rooms de la tabla request, y por último la llave foránea token de la tabla user.

Diccionario de datos

Según lo solicitado, se crearon 5 tablas, las cuales están compuestas de la siguiente manera:

La tabla **room** (sala) contienen los siguientes atributos:

- *uuid*: identificador único de cada sala
- *name:* nombre de la sala
- *capacity:* cantidad de personas que se pueden permitir por recinto
- created_At: tiempo y hora que se crea
- updated_At: tiempo y hora que se actualiza
- *location*: dónde se encuentra la sala
- *type*: tipo de sala (sólo el identificador)

La tabla **request** (reservas) contienen los siguientes atributos:

- *token*: identificador único de reserva
- *token_user:* identificador del usuario
- *date_req*: cuando se solicita la reserva
- *time_start*: hora de inicio de la reserva
- *time_end*: hora de término de la reserva
- created_At: tiempo y hora que se crea
- updated_AT: tiempo y hora que se actualiza
- *uuid_room:* identificador de la sala

Las demás tablas sólo contienen los nombres del tipo de sala y ubicación, o datos del usuario como email (@utem.cl) y nombre.

Conclusiones

Se puede concluir que se lograron los objetivos planteados en el desarrollo de este proyecto API-REST debido a que se logró comprender el funcionamiento del protocolo HTTP y sus verbos, los cuales definen la acción que se realizará en el recurso especificado, por ejemplo los GET se obtienen agenda para un código de sala y fecha dada, con POST se puede consultar las reservas en función de los parámetros dados y por último el DELETE permite anular la reserva utilizando el token.

Anexos

Introduction | dbdiagram Docs. (n.d.). https://dbdiagram.io/docs/

Index | Node.js v20.10.0 Documentation. (n.d.). https://nodejs.org/dist/latest-v20.x/docs/api/

Using Express middleware. (n.d.). https://expressjs.com/en/guide/using-middleware.html

Deploying Express.js | FL0 Documentation. (n.d.). https://docs.fl0.com/docs/getting-started/express

PostgreSQL 16.1 documentation. (2023, November 9). PostgreSQL Documentation. https://www.postgresql.org/docs/16/index.html

Database | Open source SQL Database. (n.d.). Supabase. https://supabase.com/database

A free database designer for developers and analysts. (s. f.).

https://dbdiagram.io/d/Copy-of-Copy-of-Classroom-Reservation-6561697c3be1495787b2f555

XhrdTLS. (s. f.). GitHub - XHRDTLS/INFB8090-REsTAPI: repositorio del proyecto de REST API para computación paralela y distribuida. GitHub. https://github.com/XhrdTLS/INFB8090-RestAPI