Informe Proyecto API-REST

Asistencia de estudiantes

Integrantes

- Rodrigo Mora Palta
- Andrés Segarra Pávez
- Luis Rivas Sánchez

Profesor

Sebastián Salazar Molina

Computación Paralela y Distribuida

INFB-8090

13 de julio del 2022

Índice

ÍNDICE 2
TABLA DE ILUSTRACIONES 2
INTRODUCCIÓN 3
ENUNCIADO 4
RESOLUCIÓN DEL PROBLEMA 5
CONCLUSIÓN 11

Tabla de ilustraciones

ILUSTRACIÓN 1: DOCKER COMPOSE BUILD 6
ILUSTRACIÓN 2: DOCKER COMPOSE UP 6
ILUSTRACIÓN 3: LOGIN 7
ILUSTRACIÓN 4: GETIN 8
ILUSTRACIÓN 5: GETOUT 9
ILUSTRACIÓN 6: ATTENDANCES 10

Introducción

Una API es un conjunto de definiciones y protocolos para desarrollar e integrar software de aplicación. API significa Interfaz de programación de aplicaciones, estas aparecieron en los primeros días de la informática, mucho antes que las computadoras personales, en esos días se usaban a menudo como bibliotecas de sistemas operativos. Aunque a veces los mensajes se pasan entre hosts, casi siempre se habilitan localmente en el sistema en el que se ejecutan.

Casi 30 años después, las API se han expandido más allá del entorno local. A principios de la década de 2000, se convirtió en una tecnología importante para la integración remota de datos. Las API permiten comunicar sus productos y servicios con otros de una manera que ahorra tiempo y dinero, y ofrecen la posibilidad de simplificar el diseño, la gestión y el uso de las aplicaciones. Las API son una forma simplificada de conectar su propia infraestructura a través del desarrollo de aplicaciones nativas de la nube, pero también le permiten compartir sus datos con clientes y otros usuarios externos.

El proyecto busca la creación de un servicio API REST que pueda servir de método de toma de asistencia de los estudiantes de la UTEM, para esto usaremos el servidor de express js, junto con la librería de AXIOS con la cual poder ejecutar peticiones HTTP mediante lenguaje de programación JavaScript.

Enunciado

Una institución de educación superior, está evaluando una forma de controlar la asistencia a las dependencias de la institución, se han dado cuenta de que casi todos los estudiantes tienen un celular con acceso a internet. Por lo que están desarrollando un prototipo que permita a través de mensajería JSON, se colocarán pantallas que presentarán un QR (que tiene un JSON).

Su servicio rest debe contar:

- **Mecanismo de autenticación:** Para consumir el servicio API, esta debe al menos usar un JWT que se debe obtener desde el siguiente servicio expuesto: https://api.sebastian.cl/UtemAuth/swagger-ui/index.html.
- **Modelo de datos:** Será necesario definir el uso de un modelo de datos, para almacenar la información histórica. La restricción es que se debe utilizar PostgreSQL.
- **Operación REST:** Este servicio rest debe disponibilizar las operaciones presentadas en el swagger adjunto (Expuesto en https://api.sebastian.cl/classroom/swagger-ui/index.html).
    - {protocolo}://{ip}:{puerto}/{contexto}/{operaciones}
        - {protocolo} Indica si es http o https.
        - {ip} Indica la ip (pública) para realizar la conexión.
        - {puerto} Indica el puerto en el cual el servicio está escuchando.
        - {contexto} Es una ruta base específica por grupo.
- **Soporte Cors:** El servicio debe soportar CORS.
- **Docker:** La aplicación debe estar disponible para ser dockerizada.

En cuanto a la implementación, este proyecto debe implementar la aplicación en el lenguaje que prefiera, siempre y cuando sea un lenguaje open source, y que se pueda instalar nativamente (desde el repositorio base) en Ubuntu 22.04 LTS de 64 bits.

Resolución del problema

Lo primero que se debe hacer es la declaración de tecnologías a usar, en este caso se utilizó el framework Node JS para la implementación del código base, por la capacidad de usar diferentes librerías a través de NPM (Node Package Manager). Librerías como CORS, jwtDecode y AXIOS fueron empleadas, además de la librería pg con la cual poder conectarse a la base de datos de Postgres.

También, se empleó Express para la conexión de los distintos datos de la aplicación con el sistema de Base de datos PostgreSQL.

Inicialmente, se levantó el servidor donde se realizaron las configuraciones correspondientes para permitir el flujo de datos, rutas iniciales, puerto del servidor, etc.; para posteriormente realizar el modelamiento de la base de datos en la cual se almacenan los respectivos datos obtenidos en los ingresos de estudiantes. Cabe señalar que es una API-REST orientada al registro de estudiantes al momento de asistir a una clase dentro de la institución de estudios, la cual busca registrar su asistencia mediante un código de respuesta rápida del estudiante (QR-Code).

A continuación se mostrará paso a paso su funcionamiento y cómo se debe ejecutar la API-REST desarrollada, para este caso, en la plataforma de Linux:

Antes de iniciar, debemos preparar el sistema con los siguientes requerimientos:

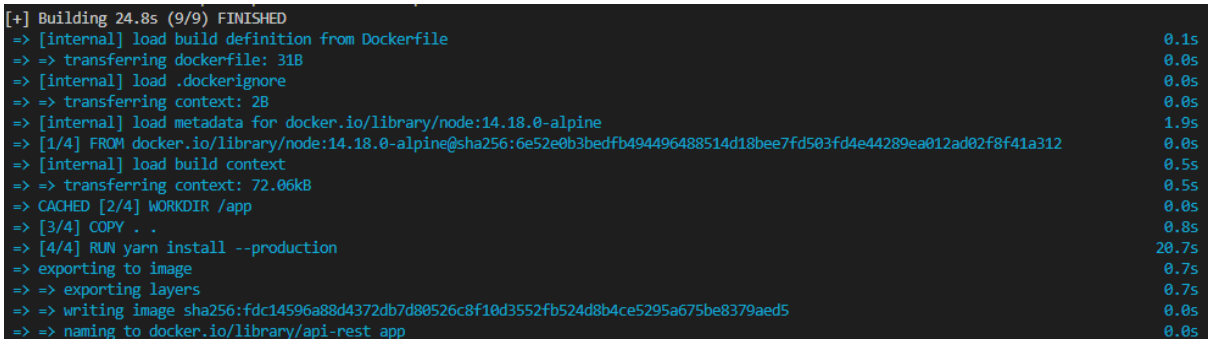
- **NodeJS**: Se debe correr el comando por terminal:
  - sudo apt-get install nodejs
- **NPM**: Este manejador de paquetes se instala con el comando:
  - sudo apt-get install npm
- **PostgreSQL**: El administrador de bases de datos se instala con:
  - sudo apt install postgresql postgresql-contrib
- **Docker**: Para instalar este servicio de levantamiento de contenedores debe ser instalado mediante el siguiente link:
  - https://docs.docker.com/engine/install/ubuntu/

Para comenzar debemos abrir una nueva terminal dentro del proyecto, en la cual se ingresa “**docker compose build**“ y “**docker compose up**“, las cuales se ejecutan

en ese orden. Una vez hecha, para el caso de este segundo comando, se procede a crear la base de datos y que en ciertas ocasiones se puede requerir de un segundo lanzamiento de este mismo comando (docker compose up) para conectarse efectivamente a la base de datos ya creada. Este servicio queda expuesto en el puerto 3000.

Primero se levantará el comando de Docker con **docker compose build**:

Ilustración 1: docker compose build



Fuente: Elaboración propia.

Luego el mensaje generado por **docker compose up**, que dirá que se abre el puerto correspondiente, que postgres, está corriendo, y que los contenedores de Docker están habilitados.

Ilustración 2: docker compose up

Fuente: Elaboración propia.

Siguiendo con la implementación, se realizan las pruebas correspondientes de éxito, en nuestro caso, utilizando Postman.

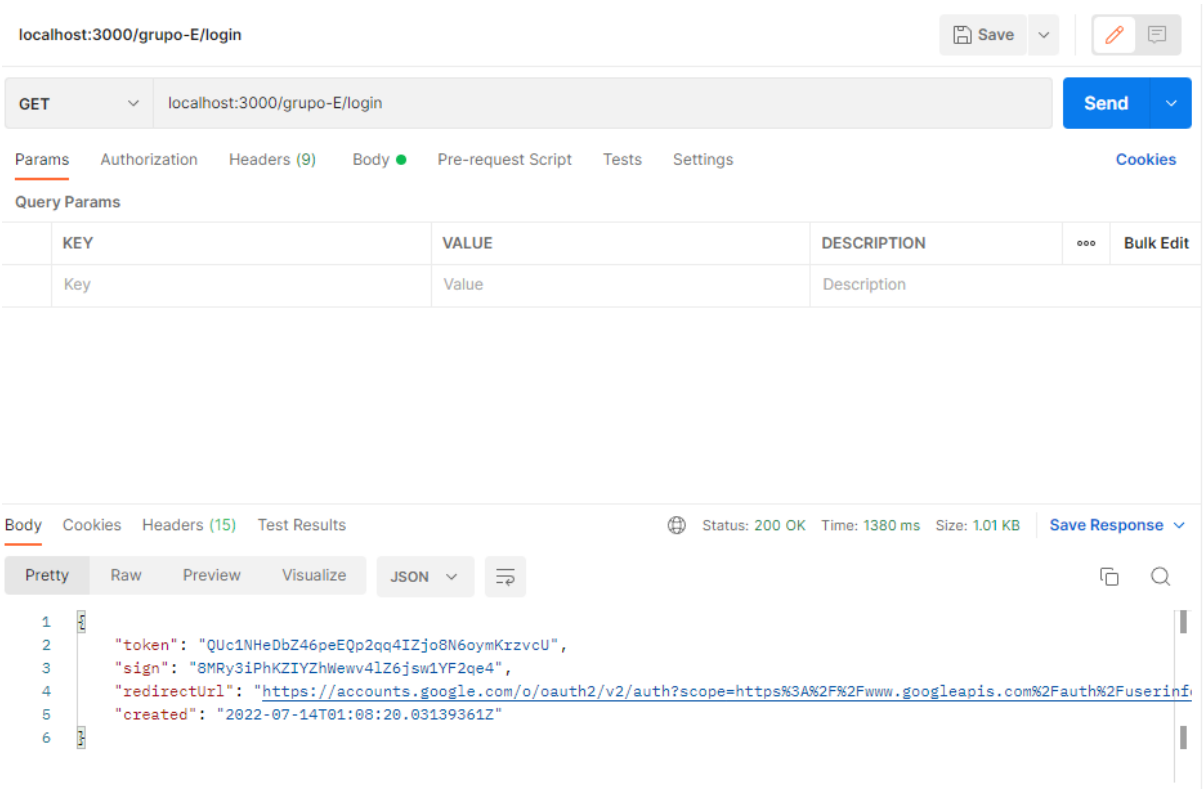
Para las funciones en la carpeta routes, hay que considerar que para obtener la asistencia ("/attendances"), primero se debe de ejecutar peticiones POST tanto para ("/getin") y ("/getout"), que sería lo mismo que obtener el ingreso del estudiante

y su salida para que se valide el chequeo de asistencia ejecutando un GET en ("/attendances").

Para efectuar la utilización de la API-Rest, tenemos que instalar Postman para verificar los resultados de los métodos getin y getout tanto para el login cómo para el Classroom de la API.

Comenzaremos con el login, en el cual tenemos que ingresar con el token ya definido, el cual nos redireccionará al inicio de sesión de cuentas Google, donde se debe ingresar con el correo institucional (@utem.cl). Al ingresar con el correo institucional se efectúa automáticamente el resultado de éxito del login efectuado en el servidor, cómo se muestra a continuación:

Ilustración 3: Login



Fuente: Elaboración propia.

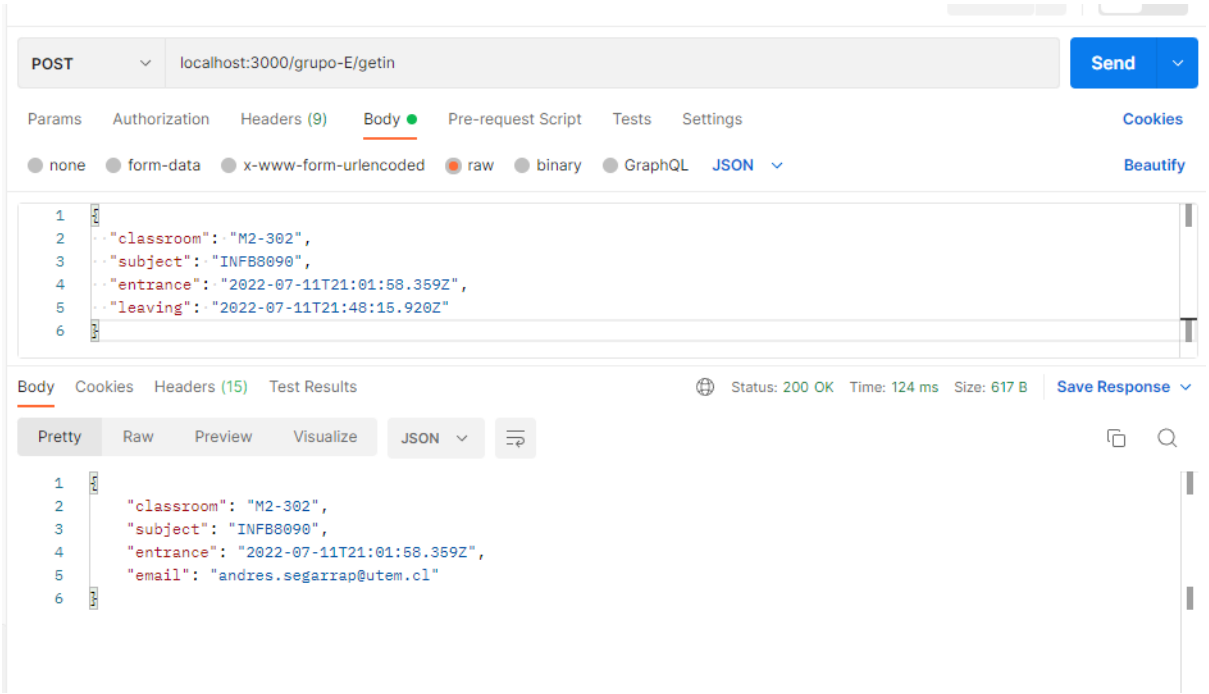
Después de ejecutar el login, se genera el getin del estudiante a la institución mediante el jwt obtenida del login del estudiante/docente, en donde la petición POST al localhost:3000/grupo-E/getin se realiza la entrada del alumno.

En el caso de Postman, se debe efectuar los siguientes parámetros en el body, específicamente seleccionar raw y cambiar TXT a JSON, como se muestra a continuación:

Ilustración 4: Postman-body

Fuente: Elaboración propia.

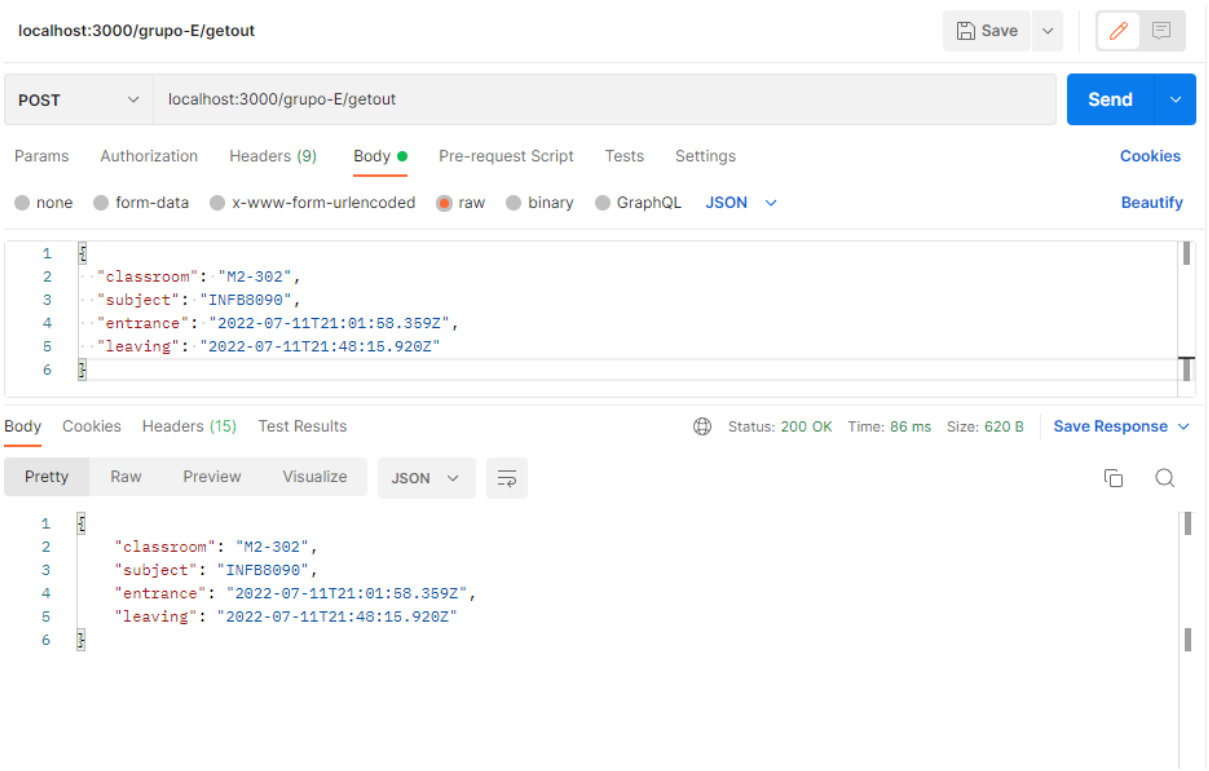
Ilustración 5: Getin



Fuente: Elaboración propia.

Como se muestra en la “ilustración 5”, se ven los datos del estudiante que ingreso a la institución, en donde muestra la sala a la que asistirá, código de la asignatura a la que asiste, su hora de entrada y su correo institucional.

Ilustración 6: Getout

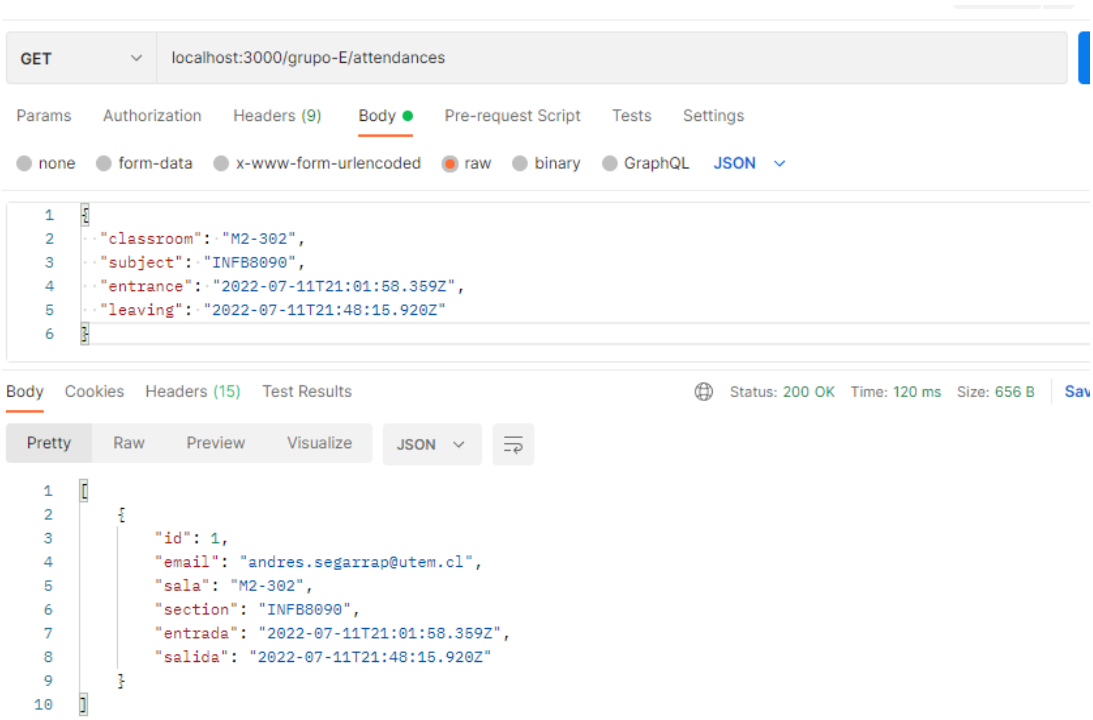


Fuente: Elaboración propia.

Como se muestra en la “ilustración 6” es el paso donde se efectúa el retiro del estudiante de la institución educacional, marcando su hora de retiro. Cabe señalar que esta petición POST se efectúa al localhost:3000/grupo-E/getout, para realizar la salida del alumno.

Continuando, se puede obtener la información al ejecutar la petición de GET/attendances al localhost:3000/grupo-E/attendances, en la cual se obtiene la información de la asistencia del estudiante a la clase en la institución, con sus respectivos datos, id y horas de entrada y salida. Este GET se puede visualizar en la siguiente ilustracion:

Ilustración 7: Attendances

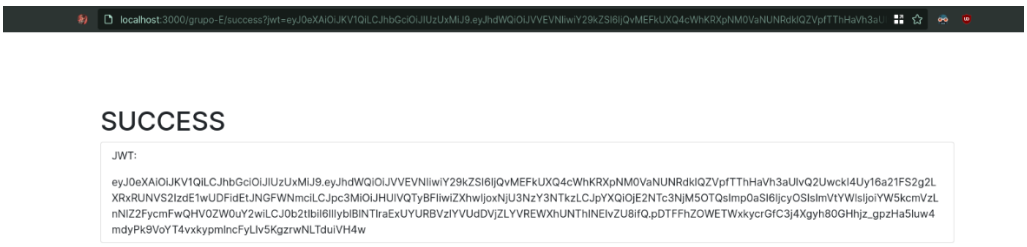


Fuente: Elaboración propia.

Al finalizar los métodos getin y getout, se guarda la información del proceso en la base de datos, no sin antes realizar una validación de usuario que se realiza con el método GET (“/login “).

Para finalizar con la API-Rest, se obtiene el resultado del JWT en un HTML como se muestra en la “ilustración 8”, en donde entrega el proceso finalizado de la toma de asistencias de estudiantes de la institución educacional.

Ilustración 8: Resultado HTML de Success



Fuente: Elaboración propia.

Conclusión

La función de la API fue concluida de manera exitosa. El código de ejecución y las tecnologías elegidas fueron las adecuadas para la ejecución de este proyecto. Cabe señalar que pueden existir mejores opciones, pero el desarrollo busca el éxito muchas veces más por sobre la optimización.

Los métodos de lectura para asistencia usados en este proyecto son de hecho bastante común hoy en día, por lo que implementar este tipo de algoritmos es una excelente manera de comenzar a practicar con los distintos niveles de exigencia que el mundo laboral puede llegar a ofrecer.

Adquirir dichos conocimientos es de suma importancia, ya que permite entender el proceso actual del desarrollo en el área de TI, en conjunto con el siguiente trabajo.