UNIVERSIDAD TECNOLÓGICA METROPOLITANA
UTEM del Estado de Chile

Proyecto API-REST Documentado

Grupo-J
Integrantes: Camilo Navas
Andrés Velásquez
Profesor: Sebastián Salazar

Computación Paralela y Distribuida

UNIVERSIDAD TECNOLÓGICA METROPOLITANA
UTEM del Estado de Chile

Introducción

En el mundo tecnológico en el que vivimos, las aplicaciones móviles están cada vez más disponibles en el mercado, por lo que el desarrollo de servicios adicionales es de vital importancia en el mundo actual, por lo que el proyecto se basa en la utilización de mensajería a través de un json para así poder controlar lo que es la asistencia del curso.

Tenemos que destacar los distintos problemas a los que nos enfrentamos para poder realizar este proyecto ya sea tener una buena base de datos por los cuales ingresaran los datos de cada alumno y poder mostrar por pantalla la asistencia correcta. Aparte debemos establecer funciones específicas para no tener ningún tipo de error al iniciar el rest y consumir sus datos.

Objetivos

1. Comprender el funcionamiento del protocolo HTTP (sus verbos y estados).
2. Comprender el funcionamiento de aplicaciones stateless, mecanismos asíncronos y funcionamiento REST.

UNIVERSIDAD TECNOLÓGICA METROPOLITANA
UTEM del Estado de Chile

Desarrollo

Debemos observar bien todos los puntos pedidos por el profesor, en el que al resolver esa incógnita debemos actuar desde lo necesario hasta refinar los detalles, en esta ocasión deberemos consumir el api entregado con anterioridad, del que debemos obtener el jwt para esto debemos crear ciertas carpetas necesarias para la implementación de forma general del backend en el que se encuentra el index.js; una carpeta de rutas; una carpeta de autenticación; carpeta para la base de datos y por último una carpeta de controles. Además, los archivos docker que están relacionados con la creación de tablas para la base de datos postgres.

En primer lugar, el archivo index.js contendrá una funciones asíncronas para poder iniciar la base de datos, aparte de tener en cuenta el puerto que será utilizado en esta ocasión será el 4000, también se verá reflejado una app use para poder inicializar el elemento CORS.

En la siguiente carpeta para ser más exacto el de las rutas podemos obtener las urls que serán implementadas en el swagger, en el que se llamarán dependiendo de lo que pidamos como por ejemplo, el login que nos mostrará la url donde nos logueamos y obtendremos la famosa jwt y su respectivo token, por lo que estos datos nos servirán para los otros llamados que será la asistencia del alumnado, un getin para poder obtener la hora de entrada y por el contrario un getout para obtener la salida. Sin embargo tenemos una carpeta llamada middleware para autenticar el jwt con ciertas validaciones.

Para poder obtener dichas llamadas tenemos que tener el esqueleto principal de todo el proyecto que es la base de datos, en donde está apartado crearemos las tablas que son importantes y de las cuales nos solicitan que es el estudiante, después la sección, la asistencia que las variables importantes que declara serían la entrada y salida, por último, el curso que obtendra informacion basado a lo que le mande la sección y el estudiante.

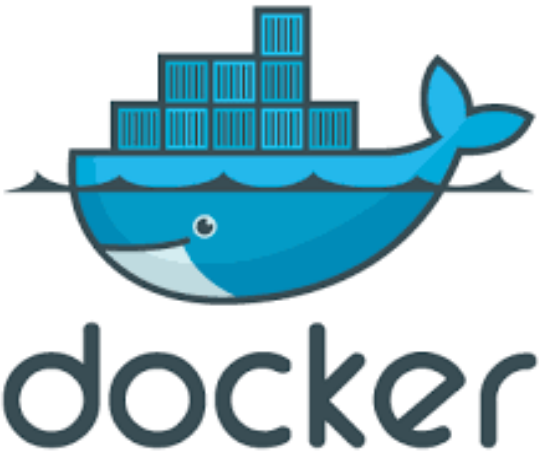
La carpeta controllers está dividido por dos archivos unos de los cuales va sumamente relacionado con la base de datos, y el otro que es el general en donde se resuelve las funciones importantes para realizar el proyecto como es el login del cliente, obtener la asistencia mediante la entrada y salida declarados en la base de datos, tambien la funcion salida de los alumnos en el que requirió de la llamada del estudiante logueado, la sección en la que se encuentra, el curso donde participa y su respectiva asistencia donde se verá reflejada su hora de entrada y salida, por el cual se mostrará en pantalla el classroom; subject; entrance; leaving. Para la función entrada nos referimos a lo mismo pero sin mostrar lo que es el leaving(salida).

Por último, después de tener todas las funciones implementadas se pasará por el docker ya enlazada, para el muestreo de servidor 4000 en marcha.

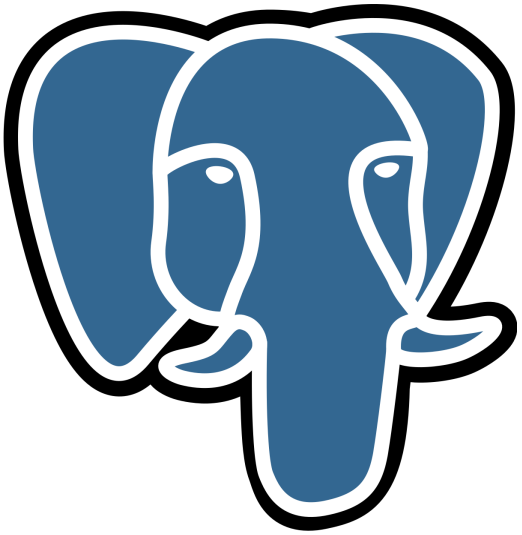
UNIVERSIDAD TECNOLÓGICA METROPOLITANA
UTEM del Estado de Chile

Tecnologías

- **Docker:** Docker es un software que permite crear, probar e implementar ciertas aplicaciones como en este caso nos sirve para realizar que nuestras funciones realicen sus operaciones mediante un sistema linux y aparte pueda guardar en sus contenedores la base de datos deseada que está en completa relación con el postgres.



- **Postgres:** Postgres es un sistema o motor de bases de datos, por el que lo utilizamos para almacenar los tablas de estudiantes, asistencias, cursos y sección para realizar las ciertas operaciones administradas en las funciones para obtener asistencia, obtener entrada y salida del alumnado.



- **Axios:** Es una librería de JavaScript, que funciona como un cliente HTTP basado en promesas para Node.js y el navegador. La ventaja de usar Axios en la API REST es la facilidad de realizar el consumo del mismo, puesto a que viene optimizado para realizar operaciones sencillas de prueba como el request de un servidor y recibir el response de manera ordenada.

AXIOS

- **Cors:** Para poder hacer una solicitud al servidor rest del profesor, hacemos uso de la librería CORS, esta librería regula la colaboración de 2 servidores mediante un usuario y un token para que no haya robo de información de parte de terceros. Por lo tanto CORS es una medida de seguridad que puede ser implementada en las API REST para compartir información importante.

CORS

Access-Control-Allow-Origin

- **Swagger:** Swagger es una herramienta de código abierto que facilita la documentación de las API REST, en donde Swagger muestra una lista de recursos disponibles de la API y a las operaciones que se pueden llamar en este recurso. Swagger especifica también la lista de parámetros de una operación, que incluye el nombre y tipo de los parámetros, si los parámetros son necesarios u opcionales, y la información de nuestros parámetros. Hoy en día, cada vez se crean más servicios para ser consumidos, por lo que un documento Swagger ayuda a que esta documentación se vea de manera clara y sencilla. En nuestro caso Swagger nos facilita las pruebas a realizar gracias a que la API está documentada.

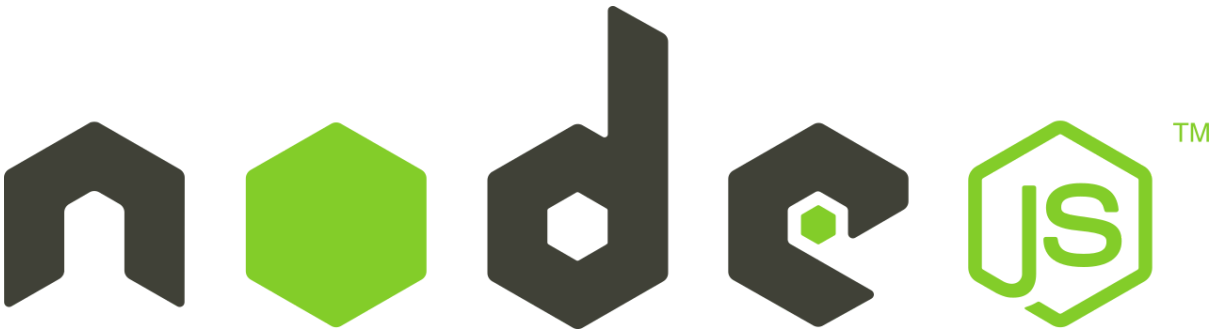
- **Visual Studio Code:** VS Code es un editor de código fuente desarrollado por Microsoft, este es un software de uso libre y funcional en varias plataformas, tales como: Windows, GNU/Linux, macOS. Este software contiene numerosas extensiones para ser trabajadas, la ventaja de usar VS Code en la actualidad es que nos permite utilizar numerosos lenguajes de programación. Para la realización de este proyecto hemos usado VS Code con el lenguaje de programación JavaScript en el entorno de desarrollo Node.js.

- **JSON:** JavaScript Object Notation es un formato ligero de intercambio de datos, es fácil de leer y escribir para los programadores, y también es simple de interpretar y generar para las máquinas. Este formato es ideal para trabajar en API REST o AJAX. Y se utiliza muy a menudo en lugar de XML ya que es una estructura ligera y compacta en los códigos. Un JSON puede contener en él 6 tipos de valores, estos son: NULL, INT, CHAR, BOOL, OBJECTS y ARRAYS. En este proyecto el contenido del Swagger está conformado por un JSON.

- **JWT:** JSON Web Token es utilizado para la creación de tokens de acceso que permite la propagación de identidad y privilegios. Los JWT están diseñados para ser compactos y poder ser enviados a las URLs para la autenticación de los usuarios y así poder darles permisos de administrador o algún permiso para trabajar con la documentación. Un JWT contiene información encriptada, y puede ser resuelta por un desencriptador para poder trabajar con los datos en su contenedor. En este proyecto el JWT contiene información del estudiante.

- **Moment.js:** Es una biblioteca de JavaScript que permite manejar fechas en Node.js y también en los navegadores. Esta herramienta es muy útil para trabajar problemas que incluyan entradas y salidas de personas, por lo que en este proyecto es utilizado para ver si el JWT de uso está expirado o no.

- **Node.js:** Es un entorno de tiempo de ejecución de JavaScript, este entorno fue creado por los desarrolladores de JavaScript con la finalidad de crear Sitios Web con la posibilidad de agregar códigos en JavaScript. Node.js aporta características interactivas y dinámicas a páginas web, por lo que node es imprescindible para trabajar API RESTs. Nuestro proyecto trabaja con el entorno de Node para la creación del Swagger en una interfaz web.



- **JsonPayload:** El payload es la parte de esa respuesta que se comunica directamente con el dispositivo que solicitó una petición. En las API REST, generalmente se trata de algunos datos con formato JSON. El uso más común de esto es el JWT, que es utilizado para las credenciales de nuestro proyecto.

JsonPayload

UNIVERSIDAD TECNOLÓGICA METROPOLITANA
UTEM del Estado de Chile

Conclusiones

En el transcurso de este proyecto hemos podido comprender el protocolo HTTP el cual nos permitió acceder a los recursos ubicados en un localhost de la API REST, el protocolo HTTP es útil para entrar en aplicativos creados con el entorno Node.js, solicitando datos para trabajar en un navegador web. Por ende, el protocolo HTTP está basado en el principio Cliente-Servidor, que en el caso del proyecto en el que estamos trabajando el Cliente llega a ser el navegador web que estamos utilizando, y el Servidor es el entorno de ejecución Node.js que es el encargado de enviar la información al cliente.

Se concluye que las aplicaciones Stateless facilitan las creación de instancias y le dan la consistencia al servicio API REST, y en instancias de este proyecto trabajar con los mecanismos asíncronos (async) nos ha permitido mejorar la velocidad de respuesta del servidor hacia el cliente, ya que realiza tareas a la par o de forma paralela sin la necesidad de que la anterior tarea haya terminado, esto provoca la reducción de tiempos de respuesta. Teniendo en consideración lo anterior, REST es útil para trabajar con la especificación HTTP, puesto a que necesita crear instancias con la velocidad óptima de ejecución para enviar de manera rápida una respuesta al cliente. Hoy en día es difícil encontrar un proyecto que no disponga de una API REST, puesto a que los sistemas más conocidos como redes sociales y páginas de entretenimiento disponen de este sistema.