# **Reto API REST / Unit Test**

**Node.js / Express.js / PostgreSQL**

El siguiente reto consiste en la construcción de una APIRest para registro y gestión de perfiles de usuarios.

Deberá existir (al menos) un usuario administrador que pueda visualizar y gestionar la información de los usuarios registrados y crear nuevas cuentas.

**Datos necesarios para creación de cuenta:**

* Correo electrónico.
* Contraseña.
* Nombre y apellido.
* Rol
  + Si el registro de la cuenta es público, el usuario por defecto tendrá el rol **‘guest’**.
  + Si la cuenta es creada desde el administrador, deberá indicar si el rol es: **‘admin’** o **‘guest’**.

### **Endpoints requeridos para el módulo público:**

* Registro.
* Inicio de sesión.
* Obtener los datos del perfil.
* Actualizar datos del perfil.
* Cambiar contraseña.
* Cierre de sesión.

### **Endpoints requeridos para el módulo Admin:**

* Listar usuarios.
* Detalles de un usuario.
* Actualizar datos del usuario (incluido el rol).
* Creación de nuevos usuarios.

### **IMPORTANTE:**

* Los endpoints del administrador deben poder diferenciarse de los endpoint de la sección pública. Solo los administradores deberán acceder a ese módulo..
* Los administradores pueden realizar las acciones relacionadas a su perfil como los usuarios ‘guest’.

## **Tecnologías requeridas:**

Para este reto deberás utilizar las siguientes tecnologías:

* Express.js (Javascript / Typescript).
* Base de datos PostgreSQL y un ORM para la interacción con la DB (por ejemplo: Sequelize ORM).
* Documentación con APIDoc o Swagger (Open API).
* Pruebas unitarias (Mocha + Chai, Jest, cualquier otro).

## **¿Qué buscamos?**

* Tu capacidad de análisis
* Tu calidad de codificación
* Tu conocimiento de API REST
* Tus conocimientos de repositorios de datos SQL
* Tu velocidad de desarrollo y despliegue

## **Ten en cuenta:**

* Se evaluará el funcionamiento de cada uno de los requerimientos, además de la claridad del código, la estructura del repositorio, pruebas unitarias, la cantidad de endpoints realizados y la documentación realizada.
* Si pasas del tiempo límite se tendrá en cuenta para la evaluación.
* Si te sobra tiempo, tienes libertad para añadir funcionalidades extra. Lo valoraremos de manera muy positiva.

## **ENTREGA**

Una vez finalices tu proyecto, deberás subirlo a un repositorio público en Github y compartir el enlace. Tu repositorio debe contener los pasos para la ejecución y documentación de los comandos a utilizar (en caso de aplicar).

Necesarias

"bcrypt": "5.1.1",

"bcryptjs": "2.4.3",

"jsonwebtoken": "9.0.2",

Optionals

"cookie-parser": "1.4.6",

"joi": "17.13.3",

"md5": "2.3.0",

"morgan": "1.10.0",

"session": "0.1.0"

/user-management-api

└───src

├───microservices

│ └───users

│ ├───application # Casos de uso (servicios)

│ ├───domain # Entidades y reglas de negocio

│ └───infraestructure

│ ├───api # Controladores y rutas

│ ├───models # Modelos de la base de datos

│ └───repositories # Interacción con la base de datos

└───middleware # Middlewares (autenticación, validaciones)

[este no va docker run --name some-postgres -e POSTGRES\_PASSWORD=mypass -p 5431:5432 -d postgres

docker ps -a]

docker image build -t usersapp:0.0.1 .

docker run -p 3001:3001 usersapp:0.0.1

docker ps -a

docker images

docker start node\_db

docker ps

docker exec -it node\_db psql -U user -d usersapp\_db

docker-compose down

docker-compose up --build

node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"

docker-compose up --build

docker exec -it node\_usersapp /bin/bash

apt update && apt install -y postgresql-client

nslookup node\_db

psql -h node\_db -U user -d usersapp\_db

npx sequelize-cli db:seed:all

docker logs --follow node\_db

PARA VER LA CONSOLA DEL CONTENEDOR  
  
docker exec -it node\_usersapp /bin/bash