PRUEBA TÉCNICA PARA PLAYVOX Agosto de 2019 Mario Fernando Espinosa - mario.f.espinosa@gmail.com

1. DESCRIPCIÓN

Se desarrolló un proyecto que permite efectuar operaciones CRUD de usuarios a través de Servicios Web RestFul, para esto se implementaron cuatro (4) contenedores, los cuales se explican a continuación:

1 **Contenedor:** Ejecuta el microservicio *UserServices* que permite efectuar CRUD de usuarios. Fue desarrollado en el framework Django, este se conecta a la Base de Datos *Users* en el motor MongoDB, esta base de datos se encuentra en un contenedor diferente.

2 **Contenedor:** Ejecuta el microservicio *NoteServices* que permite listar o crear notas de usuarios. Fue desarrollado en el framework Django, este se conecta a la Base de Datos *Notes* en el motor MongoDB, esta base de datos se encuentra en un contenedor diferente.

3 **Contenedor:** MongoDB, se aloja la base de datos *Users*.

4 **Contenedor:** MongoDB. se aloja la base de datos *Notes*.

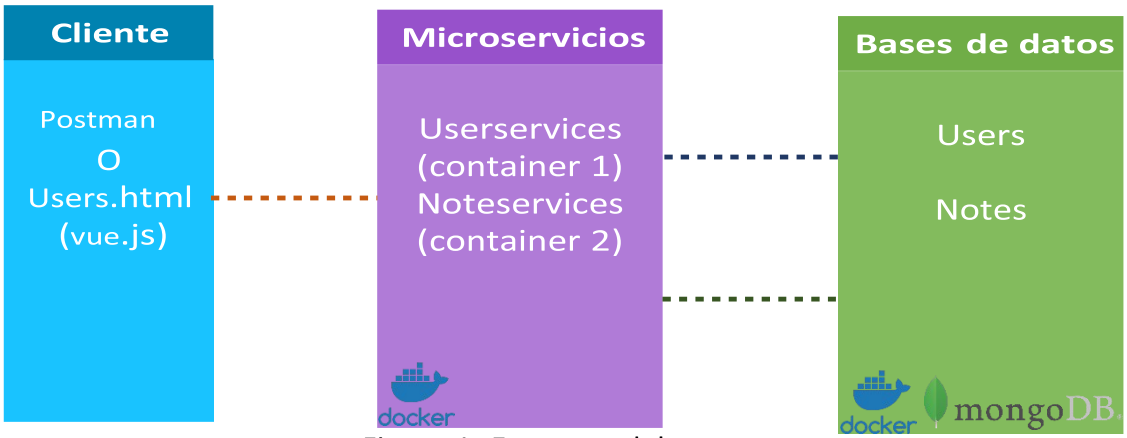

Figura 1: Esquema del proyecto

2. CONFIGURACIÓN

Cada microservicio fue desplegado en un contenedor (en modo Debug). El 1er contenedor corre en el puerto 8000, el 2do contenedor corre en el puerto 8001. Ambos se configuraron para correr en *localhost.* Las bases de datos corren en contenedores diferentes.

Se desarrolló una página web muy básica en el framework Vue.js, para consumir los servicios de los dos primeros contenedores. Es posible que se requiera habilitar el *access-control-allow-origin* en el navegador para ejecutarla. Para esta ejecución no se requiere ningún servidor Web, únicamente se abre el archivo con el navegador, ya que la página utiliza librerías CDN.

Los puertos 8000 y 8001 no estén pueden estar ocupados en el equipo.

3. VERSIONAMIENTO

docker Server v. 19.03.1 docker Client v. 19.03.1

docker compose version 1.21.0-3

mongoDB v. 4.0.11

django v. 2.2.3

django rest framework v. 3.10.2

djongo v. 1.2.33

4. INSTALACIÓN

Descargar y descomprimir los archivos y proyectos del repositorio GitHub: https://github.com/maresp85/testplayvox.git

Luego, ingresar a la carpeta *userservices* y ejecutar los siguientes comandos:

```
docker build -t userservices .

docker-compose up -d
```

Luego, ingresar a la carpeta *noteservicces* y ejecutar los siguientes comandos:

```
docker build -t noteservices .

docker-compose up -d
```

Verificar que los cuatro contenedores estén instalados y corriendo, para esto se ejecuta el comando: docker ps

Abrir la página **users.html** o a través de postman se puede verificar el funcionamiento de los microservicios.

También tengo subidas las imágenes al repositorio Docker Hub, por lo tanto, es posible descargarlas y luego ejecutar el docker-compose.yml de cada proyecto.

docker pull maresp85/testplayvox:userservices docker

pull maresp85/testplayvox:noteservices

5. **URIs**

GET (listar usuarios), **POST** (crear usuarios), **PUT** (actualizar usuarios), **DELETE** (borrar usuarios)	http://localhost:8000/v1/users/
GET (paginación, todos los usuarios)	http://localhost:8000/v1/userslist/
GET (con parámetro query, para búsqueda de usuarios por nombre)	http://localhost:8000/v1/usersearch/
GET (con parámetro id, para búsqueda de notas de un sólo usuario)	http://localhost:8001/v1/notes/1
POST (para crear notas)	http://localhost:8001/v1/notes

6. TESTING

Se efectuaron pruebas con el aplicativo POSTMAN de cada una de las URI con sus correspondientes métodos.

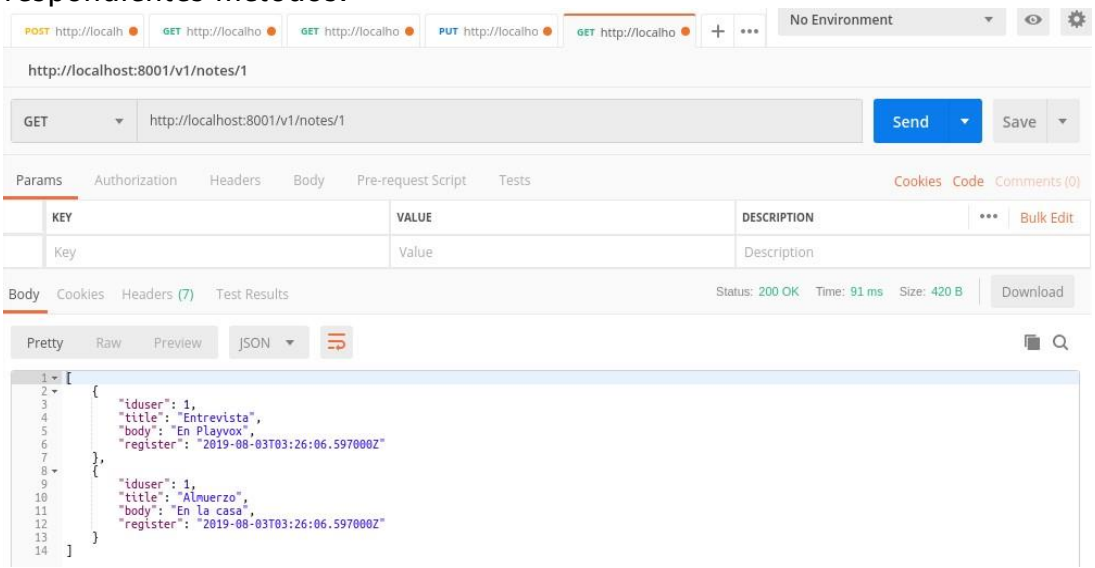

Figura 2: Pruebas en Postman

7. PAGINACIÓN

A modo de demostración, la URI: http://localhost:8000/v1/userslist/, retorna todos los usuarios y la información para realizar la paginación en el frontend indicando cantidad de usuarios en la página, total de páginas, siguiente y previa URI.

Nota: En la página cliente básica users.html (vue.js), no se utilizó esta URI.

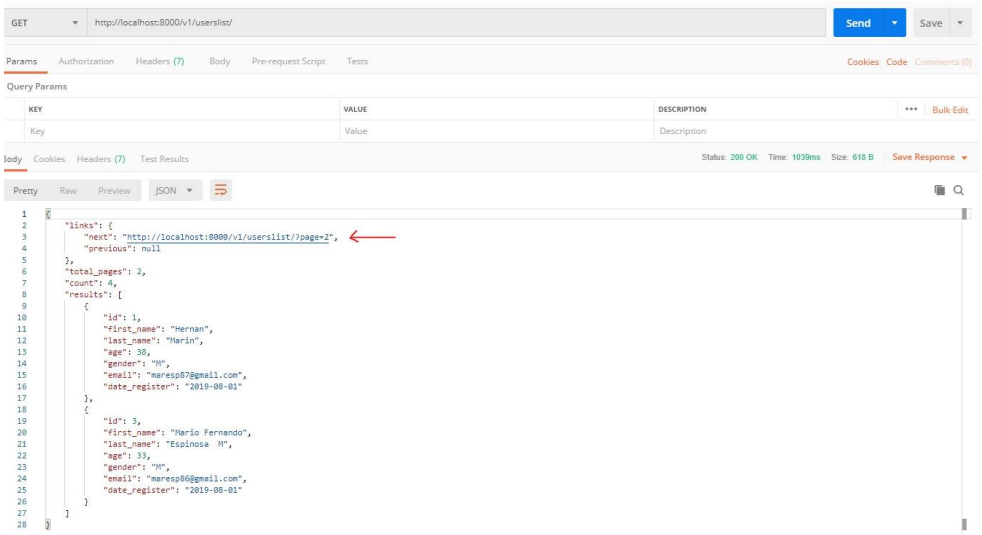

Figura 3: Paginación

8. AUTENTICACIÓN

No se implementó autenticación por token o JWT por cuestión de tiempo, pero ambos métodos los he implementado en otros proyectos.

9. CONSIDERACIONES

Se omitió la instalación de algún microservicio adicional, que sirviera como centralizador o sincronizador de los dos microservicios desarrollados.

No se desplegó el proyecto con el servidor web Gunicorn u otro equivalente. Para efectos de prueba, se utilizó el servidor Web debug integrado de Django.