Ejercicio\_Flask\_Backend\_React\_Typescript\_Frontend

# Introducción

Este es un ejercicio de creación de una aplicación web, con backend en Flask y frontend basado en React usando el lenguaje Typescript. Consta de un servidor backend, el cual usa una base de datos SQLite, y una aplicación cliente, cuyo código está hecho en Typescript. El repositorio se puede clonar en la siguiente dirección: <https://github.com/argoty7/Ejercicio_Flask_Backend_React_Typescript_Frontend.git>

# El servidor de Backend en Flask

El servidor de backend se encuentra en la carpeta "/flask-server". Su código principal está en el archivo "/flask-server/app.py". En dicho archivo:

- Se importa y configura la base de datos

- De configura el gestor de login

- Se define el modelo "User" (Usuario)

- Se definen las rutas que buscará el frontend:

- /register

- /login

- /profile

- /logout

- Se define el puerto por el que el frontend lo puede encontrar

# La base de datos

La base de datos se definió en SQLite por simplicidad y brevedad, ya que el volumen de datos a manejar por la aplicación era muy pequeño. Se encuentra en la ubicación "/flask-server/instance/myapp.db". Consta de una sola tabla llamada "Users". Usando la herramienta de Browser para SQLite se pueden observar los datos ya creados:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Datos en tabla “Users”

# La aplicación de Frontend en React con Typescript

La aplicación de frontend se encuentra en la carpeta "/react-client". Por razones de brevedad, no tiene definidos estilos ni mayores complejidades, sino que es un formulario simple. Su código principal se encuentra en "/react-client/src/App.tsx". En dicho archivo:

- Se define una clase interfaz de usuario

- Se define la función de aplicación la cual establece:

- Los estados iniciales para los parámetros de username, email, password, loggedIn y

profile.

- Una función de efecto para verificar el estado de inicio de la sesión

- Una función de registro de usuario

- Una función de obtención de perfil de usuario

- Una función de cierre de sesión

- Intefaz de usuario con páginas para:

- Registro e ingreso

- Perfil

# Los contenedores Docker

Se crearon 2 imágenes Docker, una para el backend y otra para el frontend, con base en sendos archivos Dockerfile en las carpetas "/flask-server" y "/react-client". Así mismo, el archivo docker-compose sirve para crear dos contenedores con base en dichas imágenes y la ejecución de la aplicación.

# Ejecución

## Creación de imágenes Docker

Una vez descargado el repositorio, se deben crear las imágenes Docker en cada una de las carpetas "/flask-server" y "/react-client".

Para el caso del servidor Flask, se debe iniciar una terminal y ejecutar los siguientes comandos:

- cd .\flask-server\

- docker build -t flask-server .

Para la aplicación React, se debe iniciar una terminal y ejecutar:

- cd .\react-client\

- docker build -t react-client .

## Creación de contenedores

Una vez creadas las imágenes, se debe iniciar una terminal y ejecutar:

docker compose up

## Acceso a la url de la aplicación

Para acceder a la aplicación se debe abrir un browser de web (Chrome, edge, etc.) e introducir la url: "localhost:3000". Allí se encontrará un formulario con los siguientes campos:

- Nombre de usuario

- Correo electrónico

- Contraseña

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Pagina de inicio de la aplicación

Si accede por primera vez, debe llenar los campos como desee y hacer clic en el botón "Registrarse". Si ya hizo registro, puede sólo llenar el campo de "Nombre de Usuario" y "Contraseña".

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Alerta de de inicio de sesión exitoso

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Página de perfil

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Alerta de cierre de sesión exitoso

# Principales retos al desarrollar el ejercicio

Las dos dificultades principales se dieron con el servidor Flask y con los contenedores.

En el primer caso, después de crear ambas aplicaciones, al intentar hacer el primer registro, el servidor Flask rechazaba la conexión con la aplicación frontend. Después de consultar con algunos expertos, se encontró que había un elemento que hacía falta en el archivo "app.py", el cual es un decorador para que login\_manager pueda recibir el usuario por parte de la aplicación. Para corregir dicha situación, se incluyó el siguiente fragmento de código:

'''

@login\_manager.user\_loader

def load\_user(userid):

return User.query.get(userid)

'''

En el segundo caso, el problema era similar, ya que el contenedor del frontend no podía establecer comunicación con el contenedor backend. Igual que en el caso anterior, se tuvo que investigar hasta que se encontró que en el archivo "docker-compose.yml" se definía una red interna llamada "mi-red". Se optó entonces por incluir en el fragmento de inicio de la aplicación, para indicarle el puerto a la aplicación Flask. Para ello, se modificaron las 2 últimas líneas quedando de la siguiente manera:

'''

if \_\_name\_\_ == '\_\_main\_\_':

# Ejecución del servidor, incluyendo el modo de depuración y el puerto

app.run(debug=True, host='0.0.0.0',port=5000)

'''