CONTENIDO ESPECÍFICO 3

Desarrollo Fullstack con React y Django

Objetivo

Construir una aplicación web tipo Netflix con un sistema de autenticación de usuarios, donde un usuario pueda iniciar sesión utilizando un formulario en React (Frontend) y validando su identidad en Django (Backend). La comunicación entre ambas partes se realiza mediante API REST con token de autenticación (JWT).

Práctica 3.1: Autenticación de usuarios – Inicio de sesión con React y Django

- **Asignación de roles y organización del proyecto**
- **Antes de empezar:**
  1. **Trabaja con el mismo compañero que elegiste en la actividad 1.3.**
  2. **En Jira, el responsable de Backend y el responsable de Frontend deben asignarse actividades por separado.**
  3. **El proyecto completo se debe alojar en un repositorio en GitHub.**
- **Actividades en Jira**
- **Tareas para el desarrollador Backend:**
  - **Crear la API de autenticación en Django (registro, login).**
  - **Configurar el uso de JWT (JSON Web Tokens) para la seguridad.**
  - **Habilitar CORS para permitir peticiones desde React.**
- **Tareas para el desarrollador Frontend:**
  - **Crear el formulario de inicio de sesión en React.**
  - **Conectarse a la API creada por el backend.**
  - **Mostrar mensajes si el inicio de sesión fue exitoso o fallido.**

PASOS

Paso 1: Crear repositorio en GitHub y organizar ramas.

- Antes de comenzar a programar, es importante tener bien organizado el proyecto. Usaremos GitHub para trabajar en equipo y crear ramas separadas para frontend y backend.

- Instrucciones:

  - Uno de los dos compañeros debe crear un repositorio en GitHub, con nombre:
    netflix-auth-app
  - Desde la página de github debemos invitar a nuestro compañero como colaborador.
  - Nuestro compañero debe de aceptar la invitación para poder realizar push en el mismo proyecto.
  - Después de creado, ese mismo compañero debe clonar el proyecto a su computadora:

```
git clone https://github.com/usuario/netflix-auth-app.git
cd netflix-auth-app
```

  - Crea un primer commit para poder crear nuestras ramas:

```
git add .
git commit -m "Primer commit"  # Crea el primer commit
```

  - Crear dos ramas para separar responsabilidades:

```
git branch frontend
git branch backend
```

  - Verifica en qué rama estás actualmente:

```
git branch
```

- La rama activa se marca con un asterisco *. Por ejemplo:

```
* main
  frontend
  backend
```

- Cambia de rama si no estás en la que te corresponde:

```
git checkout backend    # Si eres el de backend
git checkout frontend   # Si eres el de frontend
```

- Recuerda: Después de cada avance, haz commit y push:

Paso 2: Preparar el entorno.

1. En el explorador de archivos, crea una carpeta llamada "CECYTEMDespliegue".
2. Abre la carpeta "CECYTEMDespliegue" y crea una carpeta llamada “practica3_1”.
3. Abre Visual Studio Code y selecciona File > Open Folder, abre la carpeta "practica3_1".

Paso 3: Crear proyecto Django con autenticación (Backend).

- Crear entorno virtual y proyecto:

```
python -m venv venv
source venv/bin/activate    # Linux o macOS
venv\Scripts\activate       # Windows

pip install django djangorestframework djangorestframework-simplejwt
django-admin startproject backend .
```

- Crear una app para usuarios:

```
python manage.py startapp users
```

- Registrar la app en settings.py:

```
# backend/settings.py
INSTALLED_APPS = [
    ...
        'rest_framework',
        'users',
'corsheaders',
]

# Agregar configuración JWT
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    )
}
```

- Crear modelo personalizado de usuario (opcional):

```python
# users/models.py
from django.contrib.auth.models import AbstractUser

class CustomUser(AbstractUser):
    pass
```

- Luego, en settings.py, agregar:

```python
AUTH_USER_MODEL = 'users.CustomUser'
```

- En la carpeta users, crea una vista de registro dentro del archivo views.py:

```python
# users/views.py
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from django.contrib.auth import get_user_model

User = get_user_model()

class RegisterView(APIView):
    def post(self, request):
        username = request.data.get('username')
        password = request.data.get('password')

        if User.objects.filter(username=username).exists():
            return Response({'error': 'El usuario ya existe'}, status=status.HTTP_400_BAD_REQUEST)

        user = User.objects.create_user(username=username, password=password)
        return Response({'message': 'Usuario creado con éxito'}, status=status.HTTP_201_CREATED)
```

- Crear archivo urls.py en la app users si no existe:

```python
# backend/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('users.urls')),
]
```

- Y enlazarlo en el urls.py del proyecto principal:

```python
# backend/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('users.urls')),
]
```

- **Migraciones y superusuario:**

```
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
```

- **Te pedirá:**
  - **Nombre de usuario (Username).**
  - **Correo electrónico (opcional, puedes dejarlo vacío con Enter).**
  - **Contraseña (no verás nada en la pantalla al escribirla, pero se está registrando).**
  - **Confirmación de contraseña (vuelve a escribirla, tampoco se verá).**
- **Presiona Enter después de escribir cada dato.**

Paso 4: Habilitar CORS (Permitir comunicación segura entre React y Django).

- **Instalar CORS headers:**

```
pip install django-cors-headers
```

- **En settings.py, agregar:**

```python
INSTALLED_APPS = [
    ...
    'corsheaders',
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    ...
]

CORS_ALLOWED_ORIGINS = [
    "http://localhost:3000",
]
```

Paso 5: Crear proyecto React (Frontend).

- Crear formulario de inicio de sesión
- Desde la rama frontend:

```
npx create-react-app@latest frontend
cd frontend
```

- Instalar Axios para hacer peticiones:

```
npm install axios
```

- Crear componente de login src/Login.js
- Conectar el frontend con el backend dentro de la carpeta src:

```jsx
// src/Login.js
import React, { useState } from 'react';
import axios from 'axios';

const Login = () => {
  const [isRegistering, setIsRegistering] = useState(false);
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [message, setMessage] = useState('');

  const handleSubmit = async (e) => {
    e.preventDefault();
    const url = isRegistering
      ? 'http://localhost:8000/api/register/' // endpoint para registro
      : 'http://localhost:8000/api/login/';   // endpoint para login

    try {
      const response = await axios.post(url, {
        username,
        password
      });

      if (isRegistering) {
        setMessage('Registro exitoso. Ahora puedes iniciar sesión.');
      } else {
        setMessage('Inicio de sesión exitoso.');
        console.log('Token recibido:', response.data);
```

```
      }
    } catch (error) {
      setMessage('Hubo un error. Verifica tus datos.');
    }
  };

  return (
    <div>
<h1 className="cecyflix-logo">CECYFLIX</h1>
      <h2>{isRegistering ? 'Crear cuenta' : 'Iniciar sesión'}</h2>
      <form onSubmit={handleSubmit}>
        <input
          type="text"
          placeholder="Usuario"
          value={username}
          onChange={(e) => setUsername(e.target.value)}
        />
        <input
          type="password"
          placeholder="Contraseña"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
        />
        <button type="submit">
          {isRegistering ? 'Registrarse' : 'Ingresar'}
        </button>
      </form>
      <p>{message}</p>
      <p>
        {isRegistering ? '¿Ya tienes cuenta?' : '¿No tienes cuenta?'}{' '}
        <span
          onClick={() => setIsRegistering(!isRegistering)}
          style={{ color: '#e50914', cursor: 'pointer' }}
        >
          {isRegistering ? 'Inicia sesión' : 'Regístrate'}
        </span>
      </p>
    </div>
  );
};

export default Login;
```

- ## Modificar el componente frontend/App.js
- **Elimina todo el código y reemplaza por el siguiente:**

```
// src/App.js
import React from 'react';
import './App.css';
import Login from './Login';

function App() {
  return (
    <div className="App">
      <Login />
    </div>
  );
}

export default App;
```

- ## Modificar el componente frontend/App.css
- **Elimina todo el código y reemplaza por el siguiente:**

```
/* src/App.css */

body {
  margin: 0;
  font-family: 'Segoe UI', sans-serif;
  background-color: #141414; /* negro tipo Netflix */
  color: #fff;
}

.cecyflix-logo {
  font-size: 88px;
  font-weight: bold;
  color: #e50914; /* rojo Netflix */
  text-align: center;
  letter-spacing: 5px;
  font-family: 'Bebas Neue', Gadget, sans-serif;
  margin-bottom: 20px;
  margin-top: -70px;
}

.App {
  display: flex;
  height: 100vh;
  align-items: center;
  justify-content: center;
}
```

```css
form {
  background-color: #1f1f1f;
  padding: 40px;
  border-radius: 10px;
  width: 300px;
  box-shadow: 0px 0px 10px rgba(255, 255, 255, 0.1);
}

input {
  width: 100%;
  padding: 12px;
  margin: 10px 0;
  border: none;
  border-radius: 4px;
  font-size: 16px;
}

button {
  width: 100%;
  padding: 12px;
  margin-top: 10px;
  background-color: #e50914; /* rojo Netflix */
  color: white;
  font-weight: bold;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

button:hover {
  background-color: #f40612;
}

h2 {
  text-align: center;
  margin-bottom: 20px;
}

p {
  text-align: center;
  color: #aaa;
  margin-top: 10px;
}
```

Sincronizar cambios con GitHub y combinar el backend con el frontend

- Sincronización de ramas y ejecución local del proyecto fullstack
- ¿Qué es esto?
- Git te permite guardar tus avances (con `commit`), enviarlos a GitHub (con `push`) y recibir los avances de tu compañero (con `pull`).
  Además, como estamos trabajando en dos ramas separadas (`frontend` y `backend`), tendremos que juntar ambos proyectos en una sola carpeta raíz para poder ejecutarlos de forma local correctamente.

Guardar y subir tus avances al repositorio

- Si eres backend y ya terminaste tu parte:

```
git add .
git commit -m "Terminé la autenticación en Django"
git push origin backend
```

- Si eres frontend y terminaste tu parte:

```
git add .
git commit -m "Formulario de login en React funcionando"
git push origin frontend
```

- Traer los cambios del compañero
- Esto lo debe hacer uno de los dos (de preferencia quien vaya a ejecutar toda la app).

➡ Cambia a la rama principal (`main`) o crea una nueva rama combinada:

```
git checkout main
```

➡ Trae lo más reciente de ambas ramas:

```
git pull origin backend
git pull origin frontend
```

- Esto no sobreescribe, pero es importante que resuelvas conflictos si aparecen (Git te dirá qué archivos están en conflicto y debes corregirlos antes de continuar).
- Después realiza un push sobre la rama main para subir los archivos actualizados a github.

- Organizar el proyecto de forma correcta en carpetas

- Una vez que tengas lo de React y Django descargado, organiza así tu estructura:

```
netflix-auth-app/
├── backend/                # Carpeta con Django
│   ├── manage.py
│   └── ...
├── frontend/               # Carpeta con React
│   ├── package.json
│   └── ...
├── README.md
├── .gitignore
└── ...
```

Paso 6: Ejecutar el backend.

- Abre una terminal en VS Code dentro de la carpeta `backend`:

```
cd backend
```

- Activa el entorno virtual (si no lo hiciste antes):

```
source venv/bin/activate    # En Linux o macOS
venv\Scripts\activate       # En Windows
```

- IMPORTANTE: Debemos estar en la carpeta principal "netflix-auth-app" e instalar django en la rama main y librerías que se instalaron por separado.

- Antes de ejecutar:

1. Antes de ejecutar:

2. Ejecuta este comando en la terminal:

```
pip install djangorestframework-simplejwt
```

- Ejecuta el servidor de Django:

```
python manage.py runserver
```

- Podemos verificar que todo esté en orden ingresando a:

http://localhost:8000/api/login/

Paso 7: Ejecutar el frontend.

- Abre otra terminal nueva, y entra a la carpeta `frontend`:

```
cd frontend
```

- Inicia el servidor React:

```
npm start
```

- Si necesitas hacer cambios y volver a subir

```
git add .
git commit -m "Descripción del cambio"
git push origin nombre-de-tu-rama
```

- Y para actualizar tu código con lo de tu compañero:

```
git pull origin nombre-de-la-rama-del-compañero
```

- Resultado esperado

  - Todo tu proyecto está organizado: React en frontend/, Django en backend/.
  - Puedes ejecutar los servidores por separado en terminales diferentes.
  - Ambas partes están conectadas y funcionando correctamente.
  - Los cambios de ambos están sincronizados en GitHub.