Nombre: Andy Ezequiel Sanic Tiul Carnet: 202006699

Link: https://github.com/AndyST70/Practicas-SA-A-202006699/tree/main/Tareas%20vacas/tarea%20extra

CASO DE USO VIBE CODING : APLICACIÓN AUTENTICACIÓN TIENDA EN LINEA

Se desarrollo una aplicación de autenticación para un sistema de tienda en línea el sistema permite lo siguiente:

- Se registran nuevos usuarios con nombre, correo y contraseña
- Inician sesión con email y contraseña

Se protegen por ultimo las contraseñas con SHA256, se izo uso de JWT para la autenticación, el frontend usa react, mui, para una interfaz moderna.

BACKEND

Quiero que construyas una aplicación de autenticación básica para una tienda en línea usando las siguientes herramientas y condiciones:

- Backend: Flask en Python (sin usar Blueprints).
- Base de datos: MySQL (usa mysql-connector-python).
- Seguridad: Las contraseñas deben ser hasheadas usando hashlib.sha256.
- Frontend: React con Material-UI (MUI).
- La aplicación debe incluir: Un formulario de registro de usuario (nombre, correo, contraseña). Un formulario de inicio de sesión.
- Validación en ambos formularios (campos vacíos, correos repetidos). Manejo de sesiones simples (usando session de Flask). Interfaz clara y responsiva con Material UI.

Restricciones: No se permite usar Blueprints de Flask.

El código debe estar separado pero en un solo archivo para backend (app.py por ejemplo). Todo debe estar bien documentado con comentarios. crea el script de la base de datos y haz uso de JWT Objetivo final: Que un usuario pueda registrarse y luego iniciar sesión en una tienda. Si el login es exitoso, que se le muestre un mensaje de bienvenida con su nombre. Si falla, un mensaje de error.

FRONTEND

Necesito que construyas el frontend en React con Material UI (MUI) para una aplicación de autenticación que ya tiene su backend funcionando con Flask y MySQL.

La aplicación debe tener:

- Un formulario de registro con los campos: nombre, email, password.
- Un formulario de inicio de sesión con los campos: email, password.
- Un componente de recuperación de contraseña opcional (como Recuperar.jsx).
- Requisitos: Usa useState para manejar el formulario.
- Usa FormData para enviar datos al backend.

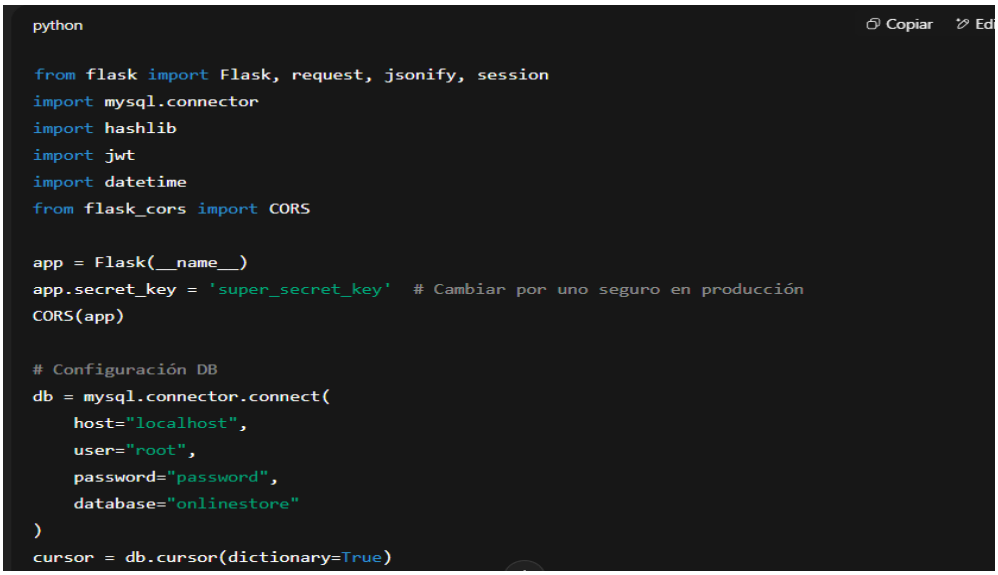
- Usa fetch o funciones importadas desde endpoints/ para conectarse con el backend (por ejemplo: registrar(formData), login(formData)).
- Muestra mensajes de éxito o error con <Typography> usando color verde (success.main) o rojo (error).

HERRAMIENTA DE VIBE CODING : APLICACIÓN DE AUTENTICACIÓN

Se hizo uso de Chat-Gpt para este trabajo

Salida que emitió para el backend

BACKEND



```python
# Ruta de login
@app.route('/login', methods=['POST'])
def login():
    data = request.json
    email = data.get('email', '').strip()
    password = data.get('password', '').strip()

    if not email or not password:
        return jsonify({'error': 'Todos los campos son obligatorios'}), 400

    hashed_password = hash_password(password)
    cursor.execute("SELECT * FROM users WHERE email=%s AND password=%s", (email, hashed_password))
    user = cursor.fetchone()

    if not user:
        return jsonify({'error': 'Credenciales incorrectas'}), 401

    token = jwt.encode({
        'user_id': user['id'],
        'exp': datetime.datetime.utcnow() + datetime.timedelta(hours=2)
    }, app.secret_key, algorithm='HS256')

    return jsonify({'token': token, 'name': user['name']}), 200
```

```python
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

# Ruta de registro
@app.route('/register', methods=['POST'])
def register():
    data = request.json
    name = data.get('name', '').strip()
    email = data.get('email', '').strip()
    password = data.get('password', '').strip()

    if not name or not email or not password:
        return jsonify({'error': 'Todos los campos son obligatorios'}), 400

    cursor.execute("SELECT id FROM users WHERE email=%s", (email,))
    if cursor.fetchone():
        return jsonify({'error': 'El correo ya está registrado'}), 409

    hashed_password = hash_password(password)
    cursor.execute("INSERT INTO users (name, email, password) VALUES (%s, %s, %s)",
                   (name, email, hashed_password))
    db.commit()
    return jsonify({'message': 'Usuario registrado exitosamente'}), 201
```

```python
# Ruta protegida (ejemplo)
@app.route('/profile', methods=['GET'])
def profile():
    auth_header = request.headers.get('Authorization')
    if not auth_header:
        return jsonify({'error': 'Token requerido'}), 401

    try:
        token = auth_header.split(" ")[1]
        decoded = jwt.decode(token, app.secret_key, algorithms=['HS256'])
        cursor.execute("SELECT name FROM users WHERE id=%s", (decoded['user_id'],))
        user = cursor.fetchone()
        return jsonify({'message': f'Bienvenido, {user["name"]}'}), 200
    except Exception:
        return jsonify({'error': 'Token inválido o expirado'}), 401

if __name__ == '__main__':
    app.run(debug=True)
```

FRONTEND

src/Register.jsx

jsx — Copiar — Editar

```jsx
import React, { useState } from 'react';
import { TextField, Button, Typography, Box } from '@mui/material';
import { registrar } from './endpoints';

export default function Register() {
  const [form, setForm] = useState({ name: '', email: '', password: '' });
  const [msg, setMsg] = useState({ text: '', error: false });

  const handleChange = (e) => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };

  const handleSubmit = async () => {
    const { name, email, password } = form;
    if (!name || !email || !password) {
      setMsg({ text: 'Todos los campos son obligatorios', error: true });
      return;
    }

    const fd = new FormData();
    fd.append('name', name);
```

```
    const fd = new FormData();
    fd.append('name', name);
    fd.append('email', email);
    fd.append('password', password);

    const res = await registrar(fd);
    setMsg({
      text: res.message || res.error || 'Error desconocido',
      error: !res.message,
    });
  };

  return (
    <Box mt={4}>
      <Typography variant="h6">Registro</Typography>
      <TextField label="Nombre" name="name" fullWidth margin="normal" onChange={handleChange} />
      <TextField label="Correo" name="email" fullWidth margin="normal" onChange={handleChange} />
      <TextField label="Contraseña" name="password" type="password" fullWidth margin="normal" onCh
      <Button variant="contained" onClick={handleSubmit}>Registrarse</Button>
      {msg.text && (
        <Typography mt={2} color={msg.error ? 'error.main' : 'success.main'}>
          {msg.text}
        </Typography>
      )}
    </Box>
```

Copiar Editar

src/endpoints/index.js

```
export const url = "http://localhost:5000";

export const registrar = async (formData) => {
  return fetch(url + "/register", {
    method: "POST",
    body: formData,
    credentials: "include",
  }).then(res => res.json());
};

export const login = async (formData) => {
  return fetch(url + "/login", {
    method: "POST",
    body: formData,
    credentials: "include",
  }).then(res => res.json());
};
```

Copiar Editar

```
    const fd = new FormData();
    fd.append('email', email);
    fd.append('password', password);

    const res = await login(fd);
    if (res.token) {
      localStorage.setItem('token', res.token);
      onLogin(res.name);
      setMsg({ text: `Bienvenido ${res.name}`, error: false });
    } else {
      setMsg({ text: res.error || 'Error en login', error: true });
    }
  };

  return (
    <Box mt={4}>
      <Typography variant="h6">Iniciar Sesión</Typography>
      <TextField label="Correo" name="email" fullWidth margin="normal" onChange={handleChange} />
      <TextField label="Contraseña" name="password" type="password" fullWidth margin="normal" onCh
      <Button variant="contained" onClick={handleSubmit}>Entrar</Button>
      {msg.text && (
        <Typography mt={2} color={msg.error ? 'error.main' : 'success.main'}>
          {msg.text}
        </Typography>
      )}
```

Copiar Editar

src/Login.jsx

jsx

Copiar Editar

```jsx
import React, { useState } from 'react';
import { TextField, Button, Typography, Box } from '@mui/material';
import { login } from './endpoints';

export default function Login({ onLogin }) {
  const [form, setForm] = useState({ email: '', password: '' });
  const [msg, setMsg] = useState({ text: '', error: false });

  const handleChange = (e) => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };

  const handleSubmit = async () => {
    const { email, password } = form;
    if (!email || !password) {
      setMsg({ text: 'Todos los campos son obligatorios', error: true });
      return;
    }

    const fd = new FormData();
```

AJUSTES Y MEJORAS PARA BACKEND

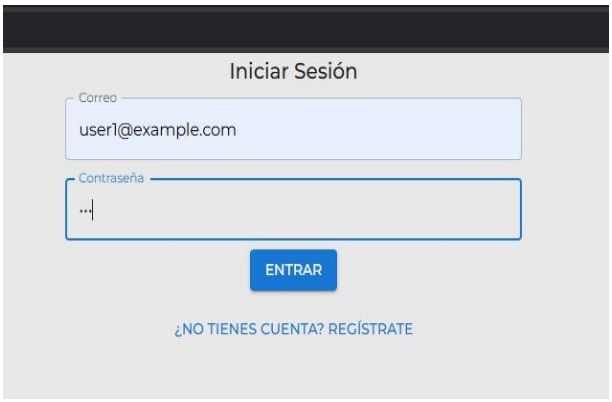
Cabe resaltar que el backend tuvo un par de complicaciones que se arreglaron manualmente, entre ellas la exposición de contraseñas críticas, y un mal manejo al cerrar y abrir las conexiones, pero se arreglo de forma manual, al igual que el manejo de CORS, pero se reestructuro el código en ese sentido y quedó funcional.

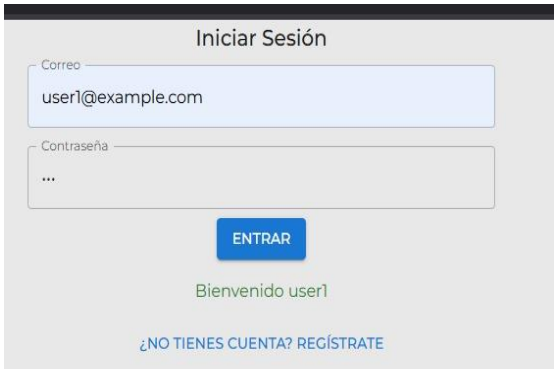
AJUSTES Y MEJORAS PARA FRONT

En el frontend ocurrió algo similar: el código generó algunos problemas relacionados con dimensiones, navegación entre páginas y la lógica de estructura, como hacer la transacción entre páginas, dimensiones de textos, de hecho, toco pedirle la misma documentación una ocasión pero se pudo resolver, genero una página muy bien estructurada, si se tiene un formato inicial y se agrega, se puede optimizar el tiempo de gestión y como se desarrolla nuevas páginas

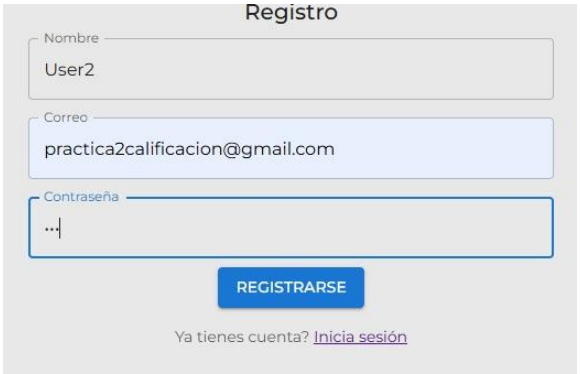
CAPTURAS DE PRODUCTO FINAL

INICIAR SESION

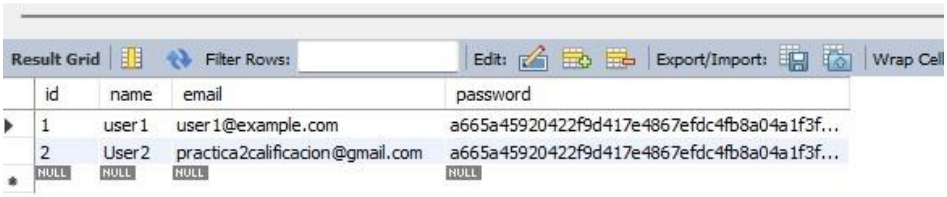




REGISTRAR USUARIO



USUARIO REGISTRADO



id	name	email	password
1	user1	user1@example.com	a665a45920422f9d417e4867efdc4fb8a04a1f3f...
2	User2	practica2calificacion@gmail.com	a665a45920422f9d417e4867efdc4fb8a04a1f3f...
NULL	NULL	NULL	NULL

CONCLUSION

El uso de Vibe Coding permitió acelerar el desarrollo de una aplicación de autenticación básica, resolviendo tareas repetitivas y permitiendo enfocarse en ajustes críticos y experiencia de usuario. Aunque fue necesario intervenir manualmente en varios puntos (como seguridad, CORS y navegación), el flujo base generado fue útil y adaptable.