Booklink

Gestor de Libros digitales

El objetivo principal de este proyecto es desarrollar una aplicación web basada en microservicios que permita a los usuarios buscar libros a través de la API de Google Books, visualizar detalles de cada libro y gestionar una lista de favoritos.

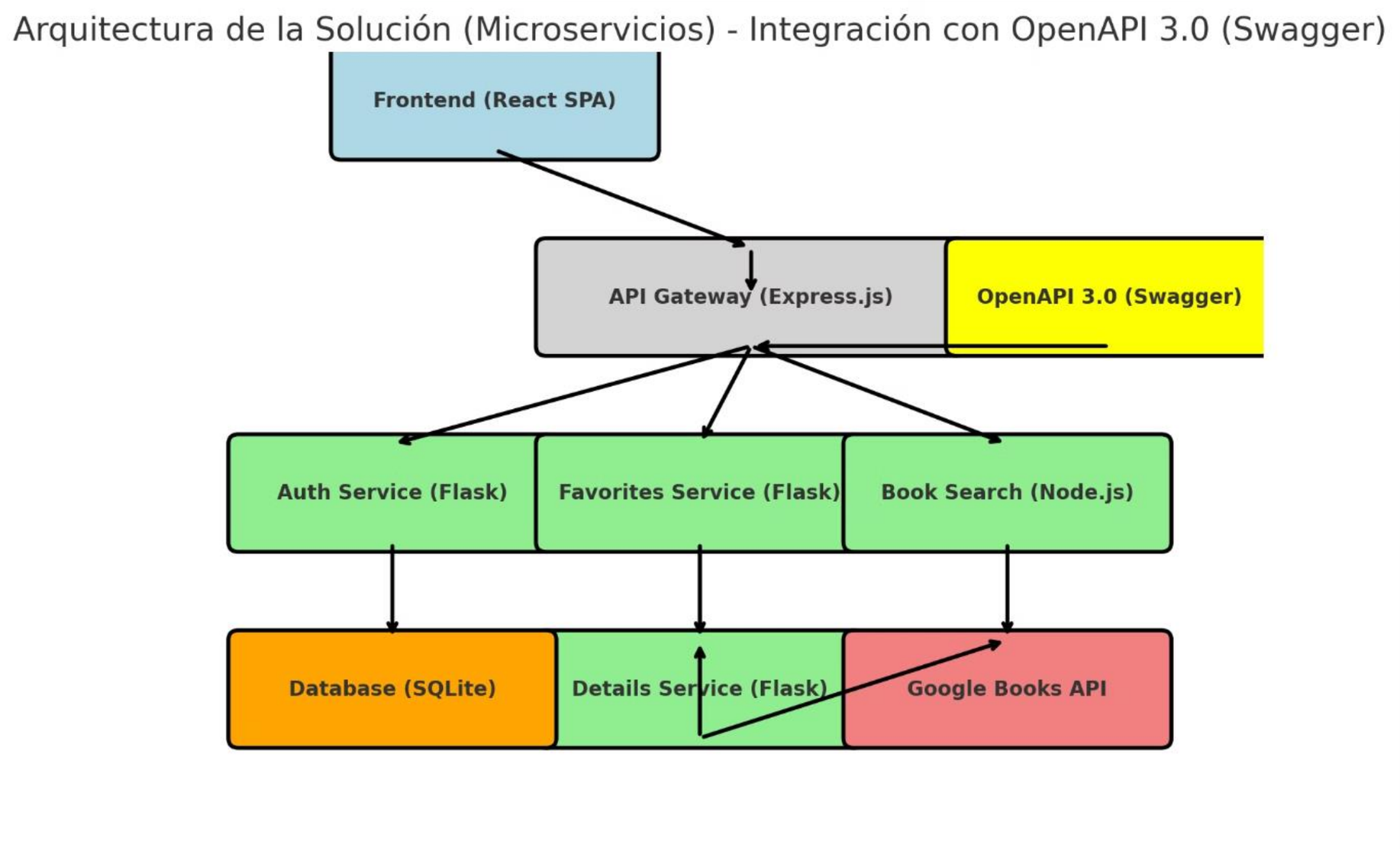
Funcionalidades

- **POST /auth/login** → Inicia sesión y devuelve un token JWT.
- **POST /auth/register** → Registra un nuevo usuario en la base de datos.
- **GET /favorites** → Obtiene la lista de libros favoritos del usuario.
- **POST /favorites** → Agrega un libro a la lista de favoritos.
- **DELETE /favorites/{id}** → Elimina un libro de la lista de favoritos por su ID.
- **GET /details/{bookId}** → Obtiene detalles de un libro usando su ID (buscando en API de Google).
- **GET /search** → Busca libros en la API de Google Books según una consulta.

Modelo de Datos (SQLite)

Tabla Usuarios	Tabla Favoritos
ID de usuario (clave primaria)	ID de favorito (clave primaria)
Nombre de usuario	ID de usuario (clave externa a la tabla de usuarios)
Email	ID de libro
Contraseña (almacenada de manera segura)	Autor
	Fecha
	Foto

Arquitectura de microservicios



Arquitectura de la solución y tecnologías

- **Arquitectura**: Basada en **microservicios** con un **API Gateway** en Node.js para gestionar la comunicación entre servicios.

- **Tecnologías**: Backend con **Node.js (Express.js)** y **Python (Flask)**, base de datos en **SQLite**, autenticación con **JWT**, frontend en **React (TypeScript)**.
- **Gateway**: **API Gateway en Express.js** para centralizar las solicitudes a los microservicios.
- **SPA o MPA**: Es una **Single Page Application (SPA)** en React con rutas dinámicas gestionadas por React Router.
- **Generación de vistas**: Cliente basado en **HTML5 con React**, haciendo **uso extensivo de JavaScript** en el frontend.
- **Documentación API**: Integración con **OpenAPI 3.0 (Swagger)** en el **API Gateway**, permitiendo una documentación interactiva y pruebas de endpoints.

Lecciones aprendidas y aspectos destacados

- **Arquitectura modular**: Implementación de **microservicios** con **API Gateway**, utilizando tecnologías modernas como **Node.js, Flask, Express, React.js, SQLite, TypeScript y JavaScript.**
- **Seguridad y autenticación**: Uso de **JWT** para control de acceso seguro.
- **Integración con API externa**: Uso de **Google Books API**.
- **Documentación profesional**: Implementación de **OpenAPI 3.0 (Swagger)** para pruebas y desarrollo.

¿Por qué una buena nota?

✅ **Aplicación funcional y bien estructurada** con autenticación, búsqueda y favoritos.
✅ **Uso de tecnologías modernas** y estándares como **RESTful y JWT**.
✅ **Uso de Docker** para facilitar la ejecución y despliegue del proyecto.

Conclusiones

Este proyecto ha sido un gran reto al que le he dedicado muchas horas de trabajo. Al principio, la idea de desarrollar una aplicación basada en **microservicios** con múltiples tecnologías como **Node.js, Flask, React y SQLite**, junto con la implementación de **JWT, Docker y OpenAPI 3.0**, me parecía un reto muy complicado. No tenía del todo claro cómo integrar todos los servicios y que la aplicación funcionara de manera fluida.

Sin embargo, a medida que avanzaba en el desarrollo, he entendido mejor la arquitectura, las conexiones entre los microservicios y la importancia de herramientas como el **API Gateway** para centralizar las peticiones.

Ahora, viendo el resultado final, me siento muy satisfecho con el trabajo realizado. He conseguido desarrollar una aplicación funcional, bien estructurada y con buenas prácticas, algo que al principio veía bastante complicado.