Actividad Evaluada: Integración de Backend con FastAPI

Curso: Desarrollo Web y Móvil

Profesor: Matías Vargas Marín

Duración estimada: 1 semanas

Descripción General

Durante las clases anteriores, los estudiantes desarrollaron una aplicación basada en **frontend** utilizando un framework moderno (Angular, React o Ionic). En esta nueva actividad, deberán extender esa aplicación incorporando un **backend completo** desarrollado con **FastAPI**, respetando la misma lógica de negocio originalmente implementada en el frontend.

El objetivo principal es que el sistema ahora funcione como una **aplicación web completa** con flujo de datos persistente, autenticación y despliegue contenedorizado.

Objetivos de Aprendizaje

- Comprender y aplicar los principios básicos de desarrollo backend con FastAPI.
- Implementar autenticación con registro e inicio de sesión (login y registro) utilizando contraseñas encriptadas.
- Diseñar y documentar un modelo de datos relacional coherente con la lógica de negocio del proyecto.
- Utilizar **Alembic** para realizar y versionar migraciones de la base de datos.
- Desplegar el backend junto a la base de datos utilizando **Docker Compose**.
- Integrar el frontend existente con los endpoints del backend.

Requerimientos del Proyecto

1. Backend con FastAPI

- El backend debe estar desarrollado con **FastAPI**.
- Incluir documentación automática (Swagger en `/docs` y Redoc en `/redoc`).
- Implementar un sistema de **login y registro obligatorio**, usando email y contraseña.
- Las contraseñas deben estar **encriptadas** mediante bcrypt.
- Debe utilizar **tokens JWT** para autenticar peticiones.

2. Modelo de Datos Relacional

- Diseñar un modelo de datos coherente con la aplicación original.
- Implementar el modelo usando **SQLAlchemy**.
- Configurar e implementar **Alembic** para migraciones.
- La base de datos puede ser PostgreSQL, MySQL o SQLite, según preferencia.

3. Dockerización

- El backend y la base de datos deben estar **dockerizados**.
- Se debe incluir un archivo `docker-compose.yml` funcional.
- El servicio debe quedar accesible en `http://localhost:8000`.

4. Integración con Frontend

- El frontend debe consumir los endpoints del backend (autenticación, operaciones CRUD, etc.).
- Debe existir coherencia visual y funcional entre frontend y backend.

Entregables

- Carpeta del proyecto con la estructura:

```
backend/
    app/
    docker-compose.yml
    requirements.txt
frontend/
    src/
    package.json
```

- Capturas o video demostrativo del sistema funcionando.
- Archivo PDF con breve descripción del modelo de datos, endpoints y decisiones técnicas tomadas.

Rúbrica de Evaluación

gray!20 **Criterio**	**Excelente (1.0)**	**Bueno (0.8)**	**Aceptable (0.6)**	**Insuficiente (0.4 o menos)**

1. Funcionamiento general del backend	El sistema funciona completamente, permite login, registro y CRUD del dominio.	Funciona con leves errores menores o limitaciones.	Funciona parcialmente, sin cubrir todo el flujo.	No logra ejecutar correctamente el backend o carece de endpoints funcionales.
2. Diseño del modelo de datos	Modelo bien estructurado, coherente y normalizado, usando claves foráneas correctamente.	Modelo parcialmente correcto con detalles menores.	Modelo incompleto o sin relaciones claras.	No se logra representar el dominio o el modelo presenta errores críticos.
3. Seguridad e implementación de autenticación	JWT y bcrypt correctamente implementados, sin exponer contraseñas.	JWT implementado con pequeños errores.	Login básico sin tokens o sin encriptación real.	No hay autenticación o se guarda contraseña en texto plano.
4. Uso de Alembic y base de datos	Migraciones automáticas funcionando, estructura relacional correcta.	Alembic configurado pero sin uso completo.	Se crea la base manualmente sin migraciones.	No se utiliza Alembic o la base no funciona.
5. Dockerización y despliegue	Docker Compose funcional y estable, levanta API y DB sin errores.	Compose funcional con pequeños detalles.	Solo parte del sistema dockerizado.	Sin dockerización o falla al levantar servicios.

6. Documentación y claridad técnica	Documentación Swagger y PDF con descripción clara y profesional.	Documentación incompleta pero funcional.	Solo Swagger sin explicación adicional.	Sin documentación o descripción técnica.
7. Integración con frontend	Frontend consume correctamente los endpoints del backend.	Integra parcialmente los endpoints.	Conexión incompleta o con errores notables.	Sin integración real o solo mockups.

Puntaje Total

Cada criterio se pondera de forma equitativa. **Nota final** = promedio de todos los criterios.

Entrega

Los proyectos se entregan mediante el aula virtual institucional, adjuntando:

- Carpeta comprimida del proyecto completo.
- Demostración funcional (máx. 10 minutos).
- PDF con descripción técnica.

Observaciones Finales

El énfasis está en la **integración completa del sistema**, por lo que se evaluará tanto el correcto funcionamiento técnico como la calidad del código y documentación. Se recomienda trabajar en parejas, asignando responsabilidades de *frontend* y *backend*.