EXTENSIONES REQUERIDAS - CONOCIMIENTO TÉCNICO

Nombre:

Juan Carlos Berdugo Gómez

FABRICA DE SOFTWARE

Dirigido:

JEFE E INSTRUCTORES

CENTRO DE GESTION DE MERCADOS LOGISTICA Y TECNOLOGIAS DE LA INFORMACION

1. FastAPI Framework

- Características principales:

FastAPI es un framework moderno y eficiente para construir APIs en Python 3.7+ y superiores, aprovechando al máximo los type hints y la programación asíncrona. Permite definir endpoints de manera declarativa, soporta validación automática de datos, serialización y deserialización, y está basado en los estándares OpenAPI y JSON Schema. Su diseño facilita la creación de APIs robustas, seguras y fáciles de mantener, permitiendo además el desarrollo rápido y la integración sencilla con otras herramientas del ecosistema Python.

- Ventajas sobre otros frameworks:

FastAPI destaca frente a frameworks tradicionales como Flask o Django REST Framework por su rendimiento superior gracias al soporte nativo de async/await, lo que permite manejar muchas conexiones concurrentes de manera eficiente. Además, su integración con Pydantic para la validación de datos y la generación automática de documentación reduce la cantidad de código repetitivo y minimiza errores. Su curva de aprendizaje es baja para quienes ya conocen Python moderno, y su comunidad está creciendo rápidamente, lo que garantiza soporte y evolución constante.

- Documentación automática:

Una de las grandes ventajas de FastAPI es la generación automática de documentación interactiva (Swagger UI y Redoc) a partir de los type hints y modelos Pydantic definidos en el código. Esto facilita la colaboración entre equipos, la integración con clientes externos y el testing manual de los endpoints, mejorando la transparencia y la mantenibilidad del proyecto.

2. SQLAlchemy ORM

- Patrón ORM vs SQL directo:

SQLAlchemy implementa el patrón ORM (Object-Relational Mapping), permitiendo mapear clases de Python a tablas de bases de datos relacionales. Esto facilita la manipulación de datos como objetos, evitando la escritura de SQL manual y reduciendo la posibilidad de errores de sintaxis o inyección de SQL. El ORM promueve la portabilidad entre diferentes motores de base de datos y mejora la mantenibilidad del código, ya que los cambios en el modelo de datos se reflejan directamente en las clases Python.

- Async/await en SQLAlchemy:

Las versiones más recientes de SQLAlchemy soportan operaciones asíncronas usando async/await, lo que permite aprovechar al máximo la concurrencia en aplicaciones modernas como las desarrolladas con FastAPI. Esto es fundamental para escalar aplicaciones que manejan múltiples conexiones simultáneas, mejorando el rendimiento y la experiencia del usuario final.

- Migraciones con Alembic:

Alembic es la herramienta oficial de migraciones para SQLAlchemy. Permite versionar y aplicar cambios en el esquema de la base de datos de forma controlada y reproducible. Esto es esencial en entornos colaborativos y de CI/CD, ya que garantiza que todos los entornos de desarrollo, testing y producción tengan el mismo esquema de base de datos, evitando inconsistencias y errores difíciles de depurar.

3. Pydantic

- Validación de datos:

Pydantic es una librería que permite validar y serializar datos usando modelos basados en type hints de Python. Esto asegura que los datos recibidos y enviados por la API cumplen con los tipos y restricciones definidos, reduciendo errores y mejorando la seguridad y robustez de la aplicación.

- Schemas vs Models:

En el contexto de FastAPI, los schemas Pydantic se utilizan para definir la estructura de los datos de entrada y salida de la API, mientras que los models suelen representar entidades internas o de dominio. Esta separación permite mantener una arquitectura limpia y desacoplada, facilitando la evolución del sistema y la integración con otros servicios.

- Type hints en Python:

Pydantic aprovecha los type hints para validar automáticamente los datos y generar documentación precisa. Esto mejora la claridad del código, facilita el mantenimiento y reduce la posibilidad de errores en tiempo de ejecución, ya que los problemas de tipo se detectan de forma temprana.

4. JWT (PyJWT)

- Estructura de tokens JWT:

Un JWT (JSON Web Token) consta de tres partes: header, payload y signature. Permite transmitir información segura y verificable entre partes, como la identidad de un usuario y sus permisos, de forma compacta y eficiente.

- Seguridad y validación:

Los JWT deben firmarse con una clave secreta o un par de claves pública/privada para garantizar su integridad y autenticidad. Es fundamental validar la firma y la expiración de los tokens en cada petición para evitar accesos no autorizados y proteger la aplicación contra ataques comunes como el replay attack.

- Refresh tokens:

Los refresh tokens permiten renovar el access token sin pedir credenciales nuevamente, mejorando la experiencia de usuario y la seguridad. Implementar correctamente el mecanismo de refresh tokens es clave para mantener sesiones seguras y minimizar el riesgo de robo de credenciales.

5. Pytest

- Testing en Python:

Pytest es el framework de pruebas más popular y flexible en Python. Permite escribir tests simples y potentes para asegurar la calidad del código, detectar errores de forma temprana y facilitar el desarrollo guiado por pruebas (TDD).

- Fixtures y mocks:

Las fixtures en Pytest permiten preparar datos o estados previos a los tests, asegurando que cada prueba se ejecute en un entorno controlado y reproducible. Los mocks simulan dependencias externas, facilitando pruebas unitarias aisladas y reduciendo la dependencia de servicios externos o bases de datos reales.

- Coverage de código:

Pytest puede integrarse con herramientas de coverage para medir qué porcentaje del código está cubierto por tests. Esto ayuda a identificar áreas no testeadas, mejorar la calidad del software y reducir la probabilidad de errores en producción.

6. Redis

- Cache distribuido:

Redis es una base de datos en memoria, ideal para cachear datos y reducir la carga en la base de datos principal. Esto mejora significativamente el rendimiento de la aplicación, especialmente en sistemas distribuidos y de alta concurrencia.

- Sessions management:

Redis se utiliza comúnmente para almacenar sesiones de usuario de forma rápida y escalable, permitiendo compartir el estado entre múltiples instancias de la aplicación y facilitando la escalabilidad horizontal.

- Casos de uso:

Redis es versátil y se utiliza para cache de consultas, almacenamiento de sesiones, colas de tareas, rate limiting, pub/sub, y más. Su velocidad y eficiencia lo convierten en una herramienta fundamental en arquitecturas modernas de microservicios.

7. Uvicorn/ASGI

-Servidor ASGI:

Uvicorn es un servidor ASGI (Asynchronous Server Gateway Interface) ligero y rápido, diseñado para aplicaciones asíncronas como FastAPI. Permite manejar conexiones concurrentes de manera eficiente y soporta WebSockets, HTTP/2 y otras tecnologías modernas.

- Diferencias con WSGI:

ASGI soporta operaciones asíncronas y WebSockets, mientras que WSGI solo permite peticiones síncronas. Esto hace que ASGI sea más adecuado para aplicaciones modernas que requieren alta concurrencia y comunicación en tiempo real.

- Performance:

Uvicorn ofrece excelente rendimiento y bajo consumo de recursos, permitiendo manejar muchas conexiones concurrentes eficientemente. Esto es fundamental para aplicaciones que deben escalar y responder rápidamente a grandes volúmenes de tráfico.

REFERENCIAS:

Features - FastAPI. (s. f.). https://fastapi.tiangolo.com/features/

Dev, D. (2024, 24 enero). Ventajas de usar Fast API el mejor Web Framework de Python. *Medium*. https://dataclouder.medium.com/ventajas-de-usar-fast-api-el-mejor-web-framework-de-python-de7ea0e0aa55

SQLAlchemy ORM — SQLAlchemy 2.0 Documentation. (s. f.). https://docs.sqlalchemy.org/en/20/orm/

SQL vs ORMs vs Query Builders | Compare | Prisma's Data Guide. (s. f.). Prisma's Data Guide. https://www.prisma.io/dataguide/types/relational/comparing-sql-query-builders-and-orms

Auto Generating Migrations — Alembic 1.16.2 documentation. (s. f.). https://alembic.sqlalchemy.org/en/latest/autogenerate.html

Hoffman, H. (2024, 3 abril). *Pydantic: Simplifying Data Validation in Python*. https://realpython.com/python-pydantic/

Admin. (2024, 23 septiembre). Database Schema vs. Model: Exploring Key Differences. *Salvation DATA*. https://www.salvationdata.com/knowledge/database-schema/

Lopez, R. H. Q. (2023, 4 agosto). *¿Qué son los Type hints en Python? Mejorar la calidad de tu código y hazlo más legible*. DEV Community. https://dev.to/rohaquinlop/que-son-los-type-hints-en-python-mejorar-la-calidad-de-tu-codigo-y-hazlo-mas-legible-5e99

Auth. (2024, 30 noviembre). *JSON Web Token Introduction - JWt.io*. JSON Web Tokens - jwt.io. https://jwt.io/introduction

How to Handle JWTs in Python. (s. f.). Auth0 - Blog. https://auth0.com/blog/how-to-handle-jwt-in-python/

Rubio, J. B. P. (2022, 25 noviembre). Herramientas de testing en Python. *OpenWebinars.net*. https://openwebinars.net/blog/herramientas-de-testing-en-python/

Mirakyan, M. (2023, 10 abril). Mocking and Fixtures in Python (87/100 Days of Python). *Medium*. https://martinxpn.medium.com/mocking-and-fixtures-in-python-87-100-days-of-python-b3812e48f491

Redis. (2025, 27 mayo). *Distributed caching*. Redis. https://redis.io/glossary/distributed-caching/

Implementations — ASGI 3.0 documentation. (s. f.). https://asgi.readthedocs.io/en/latest/implementations.html