{dev/talles}

FastAPI

SQLAlchemy vs SQLModel

Por: Ricardo Cuéllar

{dev/talles}

FastAPI

SQLAlchemy

- ORM maduro y ampliamente usado.
- Soporta declarative models.
- Flexible se puede usar ORM o SQL puro.
- Amplia comunidad, estable y robusto.
- Compatible con prácticamente todas las bases de datos.

- Requiere más configuración y código repetitivo.
- La curva de aprendizaje puede ser alta para principiantes.

{dev/talles}

FastAPI

SQLModel

- Hecho por el mismo creado de FastAPI (Sebastian Ramírez).
- Combina SQLAlchemy ORM con Pydantic.
- Tipado moderno en Python (type hints).
- Más simple y declarativo que SQLAlchemy.
- Modelo reutilizables: sirve para DB y validación de datos.
- Mejor experiencia para proyectos nuevos.

SQLModel

- No tiene aún el nivel de flexibilidad de SQLAlchemy
- Menor comunidad y documentación comparado con SQLAlchemy.

{dev/talles}

FastAPI

Diferencias clave

Característica	SQLAlchemy	SQLModel
Propósito	Toolkit completo SQL y ORM	Simplificación del ORM para APIs con FastAPI
Base	SQLAlchemy + DeclarativeORM	SQLAlchemy ORM + Pydantic Models
Validación de datos	DTOs a parte con Pydantic externo	Automático porque están en el mismo modelo
Flexibilidad y control	Completo (Core, ORM y SQL nativo)	Menos control a detalle

{dev/talles}

FastAPI

Ejemplo: FastAPI con SQLAlchemy

fastapi_sqllchemy.py

```python
from sqlalchemy.orm import declarative_base, Mapped, mapped_column
from pydantic import BaseModel

Base = declarative_base()

class UserORM(Base):
    __tablename__ = "user"
    id: Mapped[int] = mapped_column(primary_key=True)
    name: Mapped[str]
    email: Mapped[str]

# DTO separado
class UserSchema(BaseModel):
    id: int
    name: str
    email: str
```

Ejemplo: FastAPI con SQLModel

fastapi_sqlmodel.py

```python
from sqlmodel import SQLModel, Field

class User(SQLModel, table=True):
    id: int | None = Field(default=None, primary_key=True)
    name: str
    email: str
```

Observa: no se necesitan más clases

{dev/talles}

FastAPI

¿Cómo elegir el mejor?

Casos de uso	SQLAlchemy	SQLModel
APIs rápidas en FastAPI		
Proyectos sencillos y educativos		
Sistemas grandes, con consultas complejas		
Integración que librerías empresariales		
Control fino de SQL		
Productividad y sencillez		

{dev/talles}

FastAPI

Conclusión

Ambos usan el mismo motor SQLAlchemy entonces decidir cual elegir está más en la sencillez del proyecto y la personalización que necesites. SQLModel no reemplaza al 100% a SQLAlchemy lo extiende, si entiendes SQLModel ya entiendes gran parte del ORM moderno de Python.

En proyectos reales, muchas empresas inician con SQLModel por su sencillez para crear APIs y, si necesitan más personalización o poder, usan directamente SQLAlchemy por debajo.