Для архитектуры бэкенда были выбраны Python, FastAPI, PostgreSQL, SQLModel и SQLAlchemy

# Стек и структура

1. FastAPI: FastAPI — это современный, быстрый (и высокопроизводительный) веб-фреймворк для создания API на Python 3.7+, использующий стандартные типы Python для валидации, что упрощает его использование с SQLModel и SQLAlchemy. Он прост в использовании и его легко изучить.

2. PostgreSQL: PostgreSQL — это мощная объектно-реляционная система баз данных с открытым исходным кодом. Она обладает высокой расширяемостью и обеспечивает поддержку различных типов данных, индексации и сложных запросов.

3. SQLModel: SQLModel — это библиотека для взаимодействия с базами данных SQL из кода на языке Python с использованием объектов. По сути, она является оберткой поверх Pydantic и SQLAlchemy, что упрощает работу с ними.

4. SQLAlchemy: SQLAlchemy — это ORM для работы с реляционными базами данных в Python. Она предоставляет возможность работать с запросами к базам данных на уровне объектов в Python (валидацию данных в которых берёт на себя SQLModel)

# Примерная структура бэкэнда

project/

│

├── app/

│ ├── main.py

│ ├── db.py

│ ├── models/

│ │ ├── \_\_init\_\_.py

│ │ └── models.py

│ ├── schemas/

│ │ ├── \_\_init\_\_.py

│ │ └── schemas.py

│ └── crud/

│ ├── \_\_init\_\_.py

│ └── crud\_operations.py

│

├── alembic/

│ ├── versions/

│ │ └── ...

│ └── env.py

│

├── tests/

│ └── ...

│

├── requirements.txt

└── main.py

Что за что отвечает:

app/: основной каталог

main.py: инициализирует приложение FastAPI и включает маршрутизаторы

db.py: конфигурация и настройка базы данных с использованием SQLAlchemy

models/: модели баз данных.

models.py: модели SQLAlchemy для таблиц базы данных

schemas/: модели Pydantic для проверки запросов и ответов

schemas.py: модели Pydantic для проверки запросов и ответов

crud/: CRUD-операции (Create, Read, Update, Delete) для взаимодействия с базой данных

crud\_operations.py: CRUD-операции для базы данных

alembic/: скрипты миграции Alembic для управления изменениями схемы базы данных

tests/: тестовые файлы для приложения

requirements.txt: все Python-зависимости, необходимые для проекта

main.py: точка входа в приложение

Пример кода

python

main.py

from fastapi import FastAPI

from .db import engine

from .models import Base

from .routers import router

app = FastAPI()

Create tables on startup

Base.metadata.create\_all(bind=engine)

app.include\_router(router)

python

db.py

from sqlalchemy import create\_engine

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import sessionmaker

SQLALCHEMY\_DATABASE\_URL = "postgresql://user:password@postgresserver/db"

engine = create\_engine(SQLALCHEMY\_DATABASE\_URL)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative\_base()

Плюсы и минусы

Плюсы:

1. FastAPI — это высокопроизводительный веб-фреймворк, обеспечивающий производительность на уровне NodeJS и GO. SQLModel и SQLAlchemy обеспечивают эффективный и высокопроизводительный доступ к базам данных.
2. Удобный дизайн API, автоматическая проверка данных и функции подготовки документации позволяют быстро и эффективно создавать API с помощью FastAPI
3. Модульность: Использование SQLModel и SQLAlchemy обеспечивает модульность (можно изменить объект SQLModel и SQLAlchemy продолжит с ним нормально работать), что облегчает сопровождение и масштабирование приложения
4. “Питоничность”: Использование типов Python и моделей Pydantic для проверки запросов и ответов делает код более “питоничным” и более простым в написании и сопровождении

Минусы:

1) FastAPI требует изучения, особенно для тех, кто не знаком с аннотациями типов Python и асинхронным программированием. Асинхронное программирование сложнее в написании и сопровождении, чем синхронное

2) Недостаточная безопасность: FastAPI не обеспечивает безопасность напрямую, а полагается на пакет fastapi.security

Источники:

https://www.educative.io/answers/what-is-fastapi

https://www.turing.com/kb/fastapi-vs-flask-a-detailed-comparison

https://www.imaginarycloud.com/blog/flask-vs-fastapi/

https://christophergs.com/tutorials/ultimate-fastapi-tutorial-pt-8-project-structure-api-versioning/

https://testdriven.io/blog/fastapi-sqlmodel/

https://fastapi.tiangolo.com/tutorial/sql-databases/

https://sqlmodel.tiangolo.com/tutorial/fastapi/update/

https://github.com/tiangolo/full-stack-fastapi-postgresql