

# Отчет

## Инициализация фреймворка для бэкенда

В качестве основы выбран FastAPI — современный и удобный фреймворк для разработки REST API. Сервер запускается через Uvicorn в режиме разработки с поддержкой автоматической перезагрузки. Результат: приложение доступно по адресу <http://127.0.0.1:8000>, документация открывается на `/docs` и `/redoc`.

## Подключённые библиотеки

- Фреймворк: fastapi, uvicorn
- Работа с базой данных: sqlalchemy (Core), psycopg (драйвер PostgreSQL)
- Модели данных: pydantic
- Конфигурация: python-dotenv (загрузка параметров из .env)

## Подключение базы данных

- Строка подключения хранится в .env.

```
1 DATABASE_URL=postgresql+psycopg://postgres:1111@localhost:5432/FurnitureDB
2
```

- Конфигурация загружается через config.py.

```
1 from dotenv import load_dotenv
2 import os
3
4 load_dotenv(dotenv_path=os.path.join(os.path.dirname(__file__), ".env"))
5
6 DATABASE_URL = os.getenv("DATABASE_URL", "").strip()
7 if not DATABASE_URL:
8     raise RuntimeError("DATABASE_URL is not set")
9
10 print("LOADED:", DATABASE_URL)
11
```

- Работа с запросами реализована в модуле db.py.

```

1
2     from sqlalchemy import create_engine, text
3     from sqlalchemy.engine import Engine
4     from app.config import DATABASE_URL
5
6
7     engine: Engine = create_engine(
8         DATABASE_URL,
9         future=True,
10        pool_pre_ping=True,
11    )
12
13    def execute_query(sql: str, params: dict | None = None):
14        with engine.connect() as conn:
15            result = conn.execute(text(sql), params or {})
16            return result

```

## Архитектура приложения

Принята многоуровневая структура:

- repositories/ — SQL-запросы к базе данных
- services/ — бизнес-логика и расчёты
- schemas/ — Pydantic-модели
- main.py — маршруты FastAPI

## REST-архитектура и эндпоинты

В качестве подхода выбран REST, благодаря простоте и встроенной поддержке в FastAPI. Реализованы следующие эндпоинты:

- GET /products — список продукции с расчётом времени изготовления
- POST /products — добавление новой записи
- PUT /products/{id} — обновление записи
- DELETE /products/{id} — удаление записи

default

GET

/products

Get Products

POST

/products

Create Product

PUT

/products/{product\_id}

Edit Product

DELETE

/products/{product\_id}

Remove Product

Schemas

HTTPValidationError >

Expand all

object

ProductCreate >

Expand all

object

ProductItem >

Expand all

object

ValidationError >

Expand all

object