МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

Новгородский государственный университет имени Ярослава Мудрого (НовГУ) Кафедра «Электронных и информационных систем»

Лабораторная работа по дисциплине «Основы языка Python»

Отчёт.

Студент гр. 1095

Золотов Э. В.

(подпись)

«\_\_\_»\_ 2024 г.

Преподаватель

Фуфин М. Н.

(подпись)

« » 2024 г.

г. Великий Новгород

-2024-

# ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать приложение, которое позволяет добавлять, читать, модифицировать и удалять пары ключ-значение. Добавление и модификация имеют по два аргумента, удаление и чтение - один. Если запрашиваемого ключа не существует (при чтении и модификации), сервер должен вернуть ошибку. Хранилище должно работать между запусками программы.

# ПРОГРАММНЫЙ КОД

**from** **fastapi** **import** FastAPI, status, HTTPException

**from** **fastapi.responses** **import** JSONResponse

**from** **sqlalchemy** **import** create\_engine

**from** **sqlalchemy.orm** **import** DeclarativeBase

**from** **sqlalchemy.orm** **import** sessionmaker, Session

**from** **sqlalchemy** **import** Column, Integer, String

# БД

SQLALCHEMY\_DATABASE\_URL = "sqlite:///./cars.db"

engine = create\_engine(

SQLALCHEMY\_DATABASE\_URL, connect\_args={"check\_same\_thread": **False**}

)

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

**class** **Base**(DeclarativeBase): **pass**

**class** **Car**(Base):

\_\_tablename\_\_ = "cars"

id = Column(Integer, primary\_key=**True**)

name = Column(String)

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

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

db = SessionLocal()

# API

app = FastAPI()

**@app**.get('/cars')

**def** **get\_all\_cars**():

**return** db.query(Car).all()

**@app**.post('/cars')

**def** **create\_car**(id, name):

car = Car(id=int(id), name=name)

db.add(car)

db.commit()

db.refresh(car)

**return** JSONResponse(status\_code=**201**, content={

"msg": f"Машина добавлена!"

})

**@app**.get('/cars/{id}')

**def** **get\_car\_by\_id**(id):

id = int(id)

car = db.query(Car).filter(Car.id == id).first()

**if** car **is** **None**:

**return** JSONResponse(status\_code=**404**, content={

"msg": f"Машина с id: {id} не найдена"

})

**return** car

**@app**.put('/cars')

**def** **update\_car**(id, name):

id = int(id)

car = db.query(Car).filter(Car.id == id).first()

**if** car **is** **None**:

**return** JSONResponse(status\_code=**404**, content={

"msg": f"Машина с id: {id} не найдена"

})

car.id = id

car.name = name

db.commit()

db.refresh(car)

**return** car

**@app**.delete('/cars/{id}')

**def** **delete\_car**(id):

id = int(id)

car = db.query(Car).filter(Car.id == id).first()

**if** car **is** **None**:

**return** JSONResponse(status\_code=**404**, content={

"msg": f"Машина с id: {id} не найдена"

})

db.delete(car)

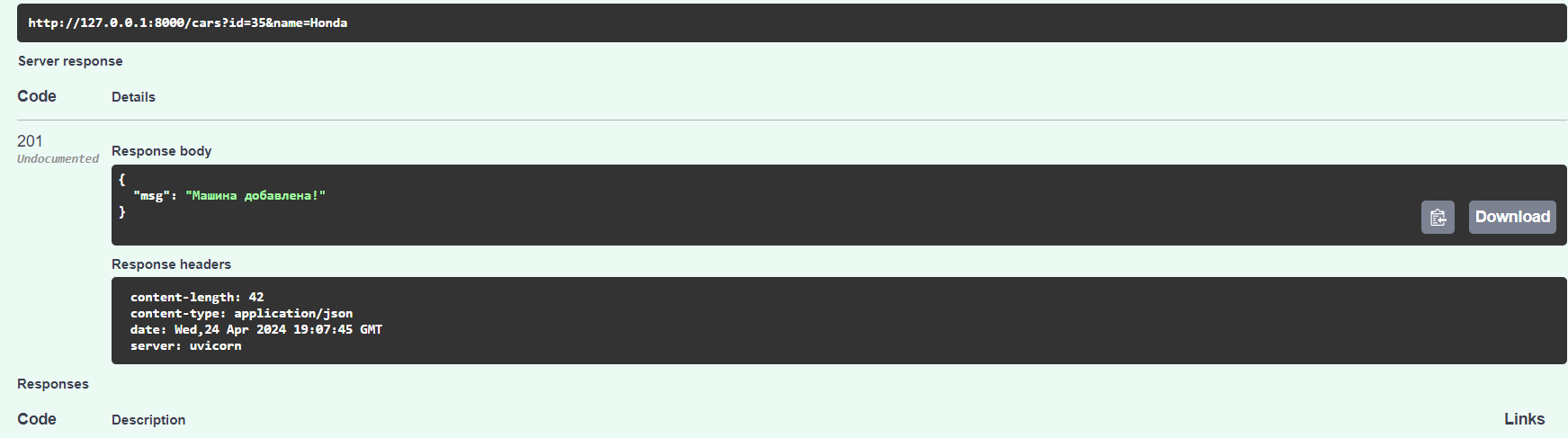
db.commit()

# ДЕМОНСТРАЦИЯ РАБОТЫ

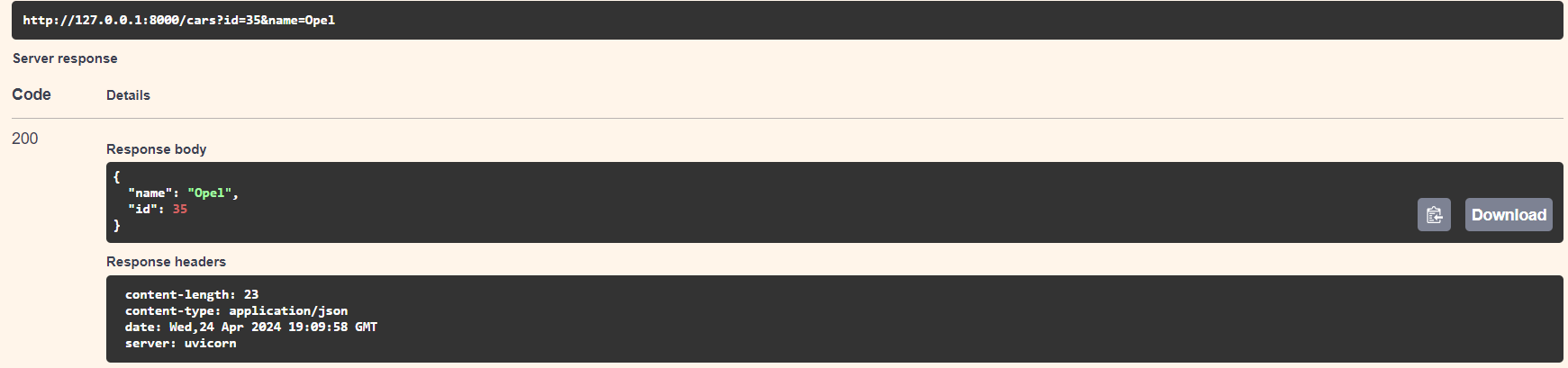
Запрос на вывод всех машин

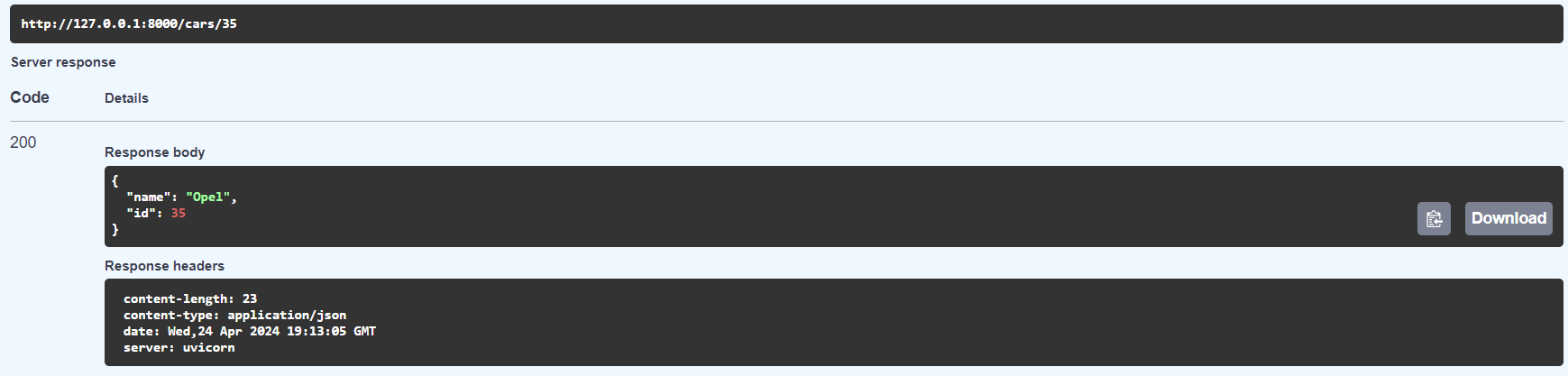


Запрос на добавление машины



Запрос на обновление машины

Запрос на вывод машины по идентификатору



Запрос на удаление авто



Обработка ошибок: не найденная запись



# ВЫВОД

В ходе выполнения лабораторной работы были приобретены навыки работы с СУБД SQLite и ORM библиотекой SQLAlchemy, а также навык разработки API на языке Python.