МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №5

Специальность ПО11

Выполнил
Гулевич Е.А.
студент группы ПО11

Проверил
А. А. Крощенко
ст. преп. кафедры ИИТ,
02.05.2025 г.

Брест 2025

Цель работы: приобрести практические навыки разработки API и баз данных

Вариант 7

Задание : Реализовать базу данных из не менее 5 таблиц на заданную тематику.
При реализации продумать типизацию полей и внешние ключи в таблицах;
2. Визуализировать разработанную БД с помощью схемы, на которой отображены все таблицы и связи между ними (пример, схема на рис. 1);
3. На языке Python с использованием SQLAlchemy реализовать подключение к БД;
4. Реализовать основные операции с данными (выборку, добавление, удаление, модификацию);
5. Для каждой реализованной операции с использованием FastAPI реализовать отдельный эндпойнт;
Базу данные можно реализовать в любой СУБД (MySQL, PostgreSQL, SQLite и др.)
База данных - Компьютерная лаборатория

Код программы :

main.py

```
from fastapi import FastAPI, Depends, HTTPException
from sqlalchemy.orm import Session
from database import SessionLocal
from crud import (
    create_computer, get_computer,
    create_software, create_employee, create_repair
)
from schemas import (
    ComputerCreate, SoftwareCreate,
    EmployeeCreate, RepairCreate
)

app = FastAPI()

# Зависимость для сессии БД
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# Эндпоинты для Computers
@app.post("/computers/", response_model=ComputerCreate)
def add_computer(computer: ComputerCreate, db: Session = Depends(get_db)):
    return create_computer(db, **computer.dict())

@app.get("/computers/{computer_id}")
def read_computer(computer_id: int, db: Session = Depends(get_db)):
    db_computer = get_computer(db, computer_id)
```

```
    if not db_computer:
        raise HTTPException(status_code=404, detail="Computer not found")
    return db_computer

# Эндпоинты для Software
@app.post("/software/", response_model=SoftwareCreate)
def add_software(software: SoftwareCreate, db: Session = Depends(get_db)):
    return create_software(db, **software.dict())

# Эндпоинты для Employees
@app.post("/employees/", response_model=EmployeeCreate)
def add_employee(employee: EmployeeCreate, db: Session = Depends(get_db)):
    return create_employee(db, **employee.dict())

# Эндпоинты для Repairs
@app.post("/repairs/", response_model=RepairCreate)
def add_repair(repair: RepairCreate, db: Session = Depends(get_db)):
    return create_repair(db, **repair.dict())

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

crud.py

```
from sqlalchemy.orm import Session
from database import Computer, Software, Employee, Repair

# Computers
def create_computer(db: Session, model: str, status: str, employee_id: int):
    db_computer = Computer(model=model, status=status, employee_id=employee_id)
    db.add(db_computer)
    db.commit()
    return db_computer

def get_computer(db: Session, computer_id: int):
    return db.query(Computer).filter(Computer.id == computer_id).first()

# Software
def create_software(db: Session, name: str, version: str):
    db_software = Software(name=name, version=version)
    db.add(db_software)
    db.commit()
    return db_software

# Employees
def create_employee(db: Session, name: str, position: str):
    db_employee = Employee(name=name, position=position)
    db.add(db_employee)
    db.commit()
    return db_employee
```

```
# Repairs
def create_repair(db: Session, computer_id: int, issue: str, date: str):
    db_repair = Repair(computer_id=computer_id, issue=issue, date=date)
    db.add(db_repair)
    db.commit()
    return db_repair
```

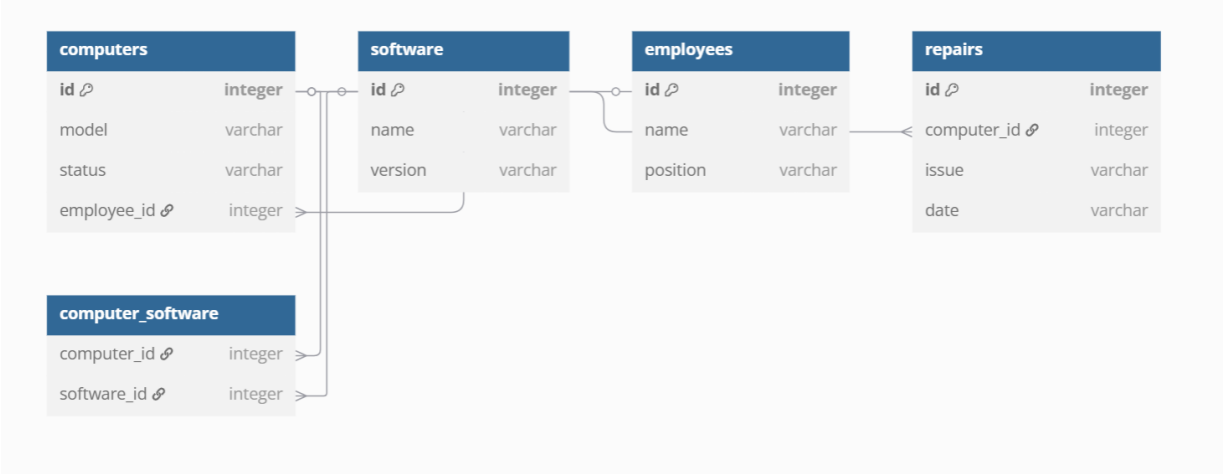
database.py

```
from sqlalchemy import create_engine, Column, Integer, String, ForeignKey, Table
from sqlalchemy.orm import declarative_base, relationship, sessionmaker

Base = declarative_base()

computer_software = Table(
    'computer_software', Base.metadata,
    Column('computer_id', Integer, ForeignKey('computers.id')),
    Column('software_id', Integer, ForeignKey('software.id'))
)

class Computer(Base):
    __tablename__ = 'computers'
    id = Column(Integer, primary_key=True)
    model = Column(String(50), nullable=False)
    status = Column(String(20))  # active, broken, archived
    employee_id = Column(Integer, ForeignKey('employees.id'))
    repairs = relationship("Repair", backref="computer")
    software = relationship("Software", secondary=computer_software)

class Software(Base):
    __tablename__ = 'software'
    id = Column(Integer, primary_key=True)
    name = Column(String(50), unique=True)
    version = Column(String(20))

class Employee(Base):
    __tablename__ = 'employees'
    id = Column(Integer, primary_key=True)
    name = Column(String(50))
    position = Column(String(30))
    computers = relationship("Computer", backref="employee")

class Repair(Base):
    __tablename__ = 'repairs'
    id = Column(Integer, primary_key=True)
    computer_id = Column(Integer, ForeignKey('computers.id'))
    issue = Column(String(200))
    date = Column(String(10))  # Формат: YYYY-MM-DD
engine = create_engine('sqlite:///computer_lab.db', echo=True)
Base.metadata.create_all(engine)
```

SessionLocal = sessionmaker(bind=engine)

Рисунки с результатами работы программы:

```
(.venv) PS D:\PythonProjects\python_beginners_course-main\SPP_labs\spp_lab5> python main.py
2025-04-26 11:03:37,078 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2025-04-26 11:03:37,078 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("computer_software")
2025-04-26 11:03:37,078 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-04-26 11:03:37,079 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("computers")
2025-04-26 11:03:37,080 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-04-26 11:03:37,080 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("software")
2025-04-26 11:03:37,080 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-04-26 11:03:37,081 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("employees")
2025-04-26 11:03:37,081 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-04-26 11:03:37,081 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("repairs")
2025-04-26 11:03:37,081 INFO sqlalchemy.engine.Engine [raw sql] ()
2025-04-26 11:03:37,082 INFO sqlalchemy.engine.Engine COMMIT
INFO:     Started server process [13716]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8002 (Press CTRL+C to quit)
```



Вывод: приобрёл практические навыки разработки API и баз данных