МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №5
Специальность ПО-11

Выполнил:
А. А. Билялова
студент группы ПО11

Проверил:
А. А. Крощенко,
ст. преп. кафедры ИИТ,
26.04.2025

Брест 2025

Цель работы: приобрести практические навыки разработки API и баз данных

Общее задание:

1. 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.)

Вариант 3: База данных Торгово-закупочная деятельность фирмы.

Код программы:

database.py:

```
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

SQLALCHEMY_DATABASE_URL = "sqlite:///./firm.db"

engine = create_engine(
    SQLALCHEMY_DATABASE_URL, connect_args={"check_same_thread": False}
)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()
```

main.py:

```
from fastapi import FastAPI, Depends
from sqlalchemy.orm import Session
import models, schemas, crud
from database import SessionLocal, engine

models.Base.metadata.create_all(bind=engine)

app = FastAPI()

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

@app.post("/suppliers/", response_model=schemas.SupplierOut)
def add_supplier(supplier: schemas.SupplierCreate, db: Session = Depends(get_db)):
    return crud.create_supplier(db, supplier)

@app.post("/products/", response_model=schemas.ProductOut)
def add_product(product: schemas.ProductCreate, db: Session = Depends(get_db)):
    return crud.create_product(db, product)
```

```
@app.post("/customers/", response_model=schemas.CustomerOut)
def add_customer(customer: schemas.CustomerCreate, db: Session = Depends(get_db)):
    return crud.create_customer(db, customer)

@app.post("/orders/", response_model=schemas.OrderOut)
def add_order(order: schemas.OrderCreate, db: Session = Depends(get_db)):
    return crud.create_order(db, order)
```

crud.py:

```
from sqlalchemy.orm import Session
import models, schemas

def create_supplier(db: Session, supplier: schemas.SupplierCreate):
    db_supplier = models.Supplier(**supplier.dict())
    db.add(db_supplier)
    db.commit()
    db.refresh(db_supplier)
    return db_supplier

def create_product(db: Session, product: schemas.ProductCreate):
    db_product = models.Product(**product.dict())
    db.add(db_product)
    db.commit()
    db.refresh(db_product)
    return db_product

def create_customer(db: Session, customer: schemas.CustomerCreate):
    db_customer = models.Customer(**customer.dict())
    db.add(db_customer)
    db.commit()
    db.refresh(db_customer)
    return db_customer

def create_order(db: Session, order: schemas.OrderCreate):
    db_order = models.Order(customer_id=order.customer_id)
    db.add(db_order)
    db.commit()
    db.refresh(db_order)

    for item in order.items:
        db_item = models.OrderItem(
            order_id=db_order.id,
            product_id=item.product_id,
            quantity=item.quantity
        )
        db.add(db_item)
    db.commit()
    return db_order
```

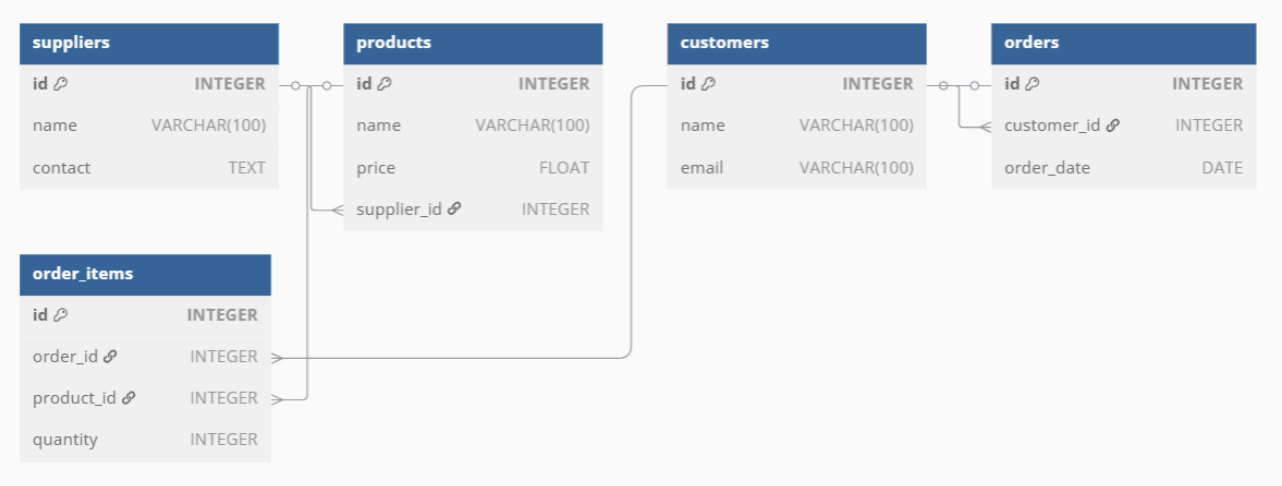
models.py:

```
from sqlalchemy import Column, Integer, String, Float, ForeignKey, DateTime
from sqlalchemy.orm import relationship
from database import Base
from datetime import datetime

class Supplier(Base):
    __tablename__ = "suppliers"
    id = Column(Integer, primary_key=True, index=True)
    name = Column(String, nullable=False)
    contact = Column(String)

    products = relationship("Product", back_populates="supplier")

class Product(Base):
    __tablename__ = "products"
    id = Column(Integer, primary_key=True, index=True)
    name = Column(String, nullable=False)
    price = Column(Float, nullable=False)
```

```
    supplier_id = Column(Integer, ForeignKey("suppliers.id"))

    supplier = relationship("Supplier", back_populates="products")
    order_items = relationship("OrderItem", back_populates="product")

class Customer(Base):
    __tablename__ = "customers"
    id = Column(Integer, primary_key=True, index=True)
    name = Column(String, nullable=False)
    email = Column(String)

    orders = relationship("Order", back_populates="customer")

class Order(Base):
    __tablename__ = "orders"
    id = Column(Integer, primary_key=True, index=True)
    customer_id = Column(Integer, ForeignKey("customers.id"))
    order_date = Column(DateTime, default=datetime.utcnow)

    customer = relationship("Customer", back_populates="orders")
    items = relationship("OrderItem", back_populates="order")

class OrderItem(Base):
    __tablename__ = "order_items"
    id = Column(Integer, primary_key=True, index=True)
    order_id = Column(Integer, ForeignKey("orders.id"))
    product_id = Column(Integer, ForeignKey("products.id"))
    quantity = Column(Integer, nullable=False)

    order = relationship("Order", back_populates="items")
    product = relationship("Product", back_populates="order_items")
```

Рисунки с результатами работы программы

```
INFO:     Started server process [16312]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:62999 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:62999 - "GET /openapi.json HTTP/1.1" 200 OK
```



Вывод: приобрела практические навыки разработки API и баз данных.