Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Факультет электронно-информационных систем

Кафедра ИИТ

Лабораторная работа №5

По дисциплине «Современные платформы программирования»

Выполнила:

Студентка 3 курса

Группы ПО-11

Лозицкая М.А.

Проверил:

Козик И.Д.

Брест 2025

**Цель работы:** приобрести практические навыки разработки API и баз данных

**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 13  
База данных Формула-1

Код программы:  
main.py

from fastapi import FastAPI, Depends, HTTPException

from sqlalchemy.orm import Session

from database import get\_db

from crud import \*

from pydantic import BaseModel

from datetime import date

from typing import List

app = FastAPI()

# Pydantic модели

class TeamCreate(BaseModel):

name: str

headquarters: str

principal: str

class DriverCreate(BaseModel):

name: str

nationality: str

date\_of\_birth: date

team\_id: int

class RaceResultCreate(BaseModel):

driver\_id: int

gp\_id: int

position: int

points: int

class DriverResponse(BaseModel):

id: int

name: str

nationality: str

date\_of\_birth: date

team\_id: int

class Config:

orm\_mode = True

class GrandPrixResponse(BaseModel):

id: int

name: str

date: date

track\_id: int

class Config:

orm\_mode = True

class RaceResultResponse(BaseModel):

id: int

position: int

points: int

driver: DriverResponse

grand\_prix: GrandPrixResponse

class Config:

orm\_mode = True

class TrackCreate(BaseModel):

name: str

country: str

length: int

lap\_record: str

class GrandPrixCreate(BaseModel):

name: str

date: date

track\_id: int

# Эндпоинты для команд

@app.post("/teams/")

def create\_new\_team(team: TeamCreate, db: Session = Depends(get\_db)):

return create\_team(db, \*\*team.dict())

@app.get("/teams/")

def read\_teams(db: Session = Depends(get\_db)):

return get\_teams(db)

# Эндпоинты для гонщиков

@app.post("/drivers/")

def create\_new\_driver(driver: DriverCreate, db: Session = Depends(get\_db)):

return create\_driver(db, \*\*driver.dict())

@app.get("/drivers/")

def read\_drivers(db: Session = Depends(get\_db)):

return get\_drivers(db)

# Эндпоинты для результатов

@app.get("/results/", response\_model=List[RaceResultResponse])

def read\_results(db: Session = Depends(get\_db)):

results = get\_race\_results(db)

return results

@app.get("/results/")

def read\_results(db: Session = Depends(get\_db)):

return get\_race\_results(db)

@app.post("/tracks/")

def create\_new\_track(track: TrackCreate, db: Session = Depends(get\_db)):

return create\_track(db, \*\*track.dict())

@app.post("/grands\_prix/")

def create\_new\_grand\_prix(gp: GrandPrixCreate, db: Session = Depends(get\_db)):

return create\_grand\_prix(db, \*\*gp.dict())

# Запуск: uvicorn main:app –reload

crud.py

from datetime import date  
from sqlalchemy.orm import Session  
from database import Team, Driver, Track, GrandPrix, RaceResult  
from sqlalchemy.orm import joinedload  
# CRUD для команд  
def create\_team(db: Session, name: str, headquarters: str, principal: str):  
 team = Team(name=name, headquarters=headquarters, principal=principal)  
 db.add(team)  
 db.commit()  
 db.refresh(team)  
 return team  
def get\_teams(db: Session):  
 return db.query(Team).all()  
# CRUD для гонщиков  
def create\_driver(db: Session, name: str, nationality: str, date\_of\_birth: date, team\_id: int):  
 driver = Driver(name=name, nationality=nationality,  
 date\_of\_birth=date\_of\_birth, team\_id=team\_id)  
 db.add(driver)  
 db.commit()  
 db.refresh(driver)  
 return driver  
def get\_drivers(db: Session):  
 return db.query(Driver).all()  
# CRUD для результатов гонок  
def create\_race\_result(db: Session, driver\_id: int, gp\_id: int, position: int, points: int):  
 result = RaceResult(driver\_id=driver\_id, gp\_id=gp\_id,  
 position=position, points=points)  
 db.add(result)  
 db.commit()  
 db.refresh(result)  
 return result  
def get\_race\_results(db: Session):  
 return db.query(RaceResult)\  
 .options(  
 joinedload(RaceResult.driver),  
 joinedload(RaceResult.grand\_prix)  
 )\  
 .all()  
# CRUD для трасс  
def create\_track(db: Session, name: str, country: str, length: int, lap\_record: str):  
 track = Track(name=name, country=country, length=length, lap\_record=lap\_record)  
 db.add(track)  
 db.commit()  
 db.refresh(track)  
 return track  
# CRUD для гран-при  
def create\_grand\_prix(db: Session, name: str, date: date, track\_id: int):  
 gp = GrandPrix(name=name, date=date, track\_id=track\_id)  
 db.add(gp)  
 db.commit()  
 db.refresh(gp)  
 return gp

database.py

from sqlalchemy import create\_engine, Column, Integer, String, Date, ForeignKey  
from sqlalchemy.orm import declarative\_base, sessionmaker, relationship  
Base = declarative\_base()  
class Team(Base):  
 \_\_tablename\_\_ = 'teams'  
 id = Column(Integer, primary\_key=True)  
 name = Column(String, unique=True)  
 headquarters = Column(String)  
 principal = Column(String)  
 drivers = relationship("Driver", back\_populates="team")  
class Driver(Base):  
 \_\_tablename\_\_ = 'drivers'  
 id = Column(Integer, primary\_key=True)  
 name = Column(String)  
 nationality = Column(String)  
 date\_of\_birth = Column(Date)  
 team\_id = Column(Integer, ForeignKey('teams.id'))  
 team = relationship("Team", back\_populates="drivers")  
 results = relationship("RaceResult", back\_populates="driver")  
class Track(Base):  
 \_\_tablename\_\_ = 'tracks'  
 id = Column(Integer, primary\_key=True)  
 name = Column(String)  
 country = Column(String)  
 length = Column(Integer) # в метрах  
 lap\_record = Column(String)  
 grands\_prix = relationship("GrandPrix", back\_populates="track")  
class GrandPrix(Base):  
 \_\_tablename\_\_ = 'grands\_prix'  
 id = Column(Integer, primary\_key=True)  
 name = Column(String)  
 date = Column(Date)  
 track\_id = Column(Integer, ForeignKey('tracks.id'))  
 track = relationship("Track", back\_populates="grands\_prix")  
 results = relationship("RaceResult", back\_populates="grand\_prix")  
class RaceResult(Base):  
 \_\_tablename\_\_ = 'race\_results'  
 id = Column(Integer, primary\_key=True)  
 driver\_id = Column(Integer, ForeignKey('drivers.id'))  
 gp\_id = Column(Integer, ForeignKey('grands\_prix.id'))  
 position = Column(Integer)  
 points = Column(Integer)  
 driver = relationship("Driver", back\_populates="results")  
 grand\_prix = relationship("GrandPrix", back\_populates="results")  
engine = create\_engine('sqlite:///f1\_database.db')  
Base.metadata.create\_all(engine)  
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)  
def get\_db():  
 db = SessionLocal()  
 try:  
 yield db  
 finally:  
 db.close()

**Вывод:** приобресла практические навыки разработки API и баз данных