ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет инфокоммуникационных технологий

Дисциплина:

«Средства веб-программирования»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

«**Реализация серверного приложения FastAPI.**»

Выполнил:

студент группы K33391
Микитчак Иван Михайлович

_____
(подпись)

Проверил:

Говоров Антон Игоревич

_____
(отметка о выполнении)

_____
(подпись)

Санкт-Петербург

2023 г.

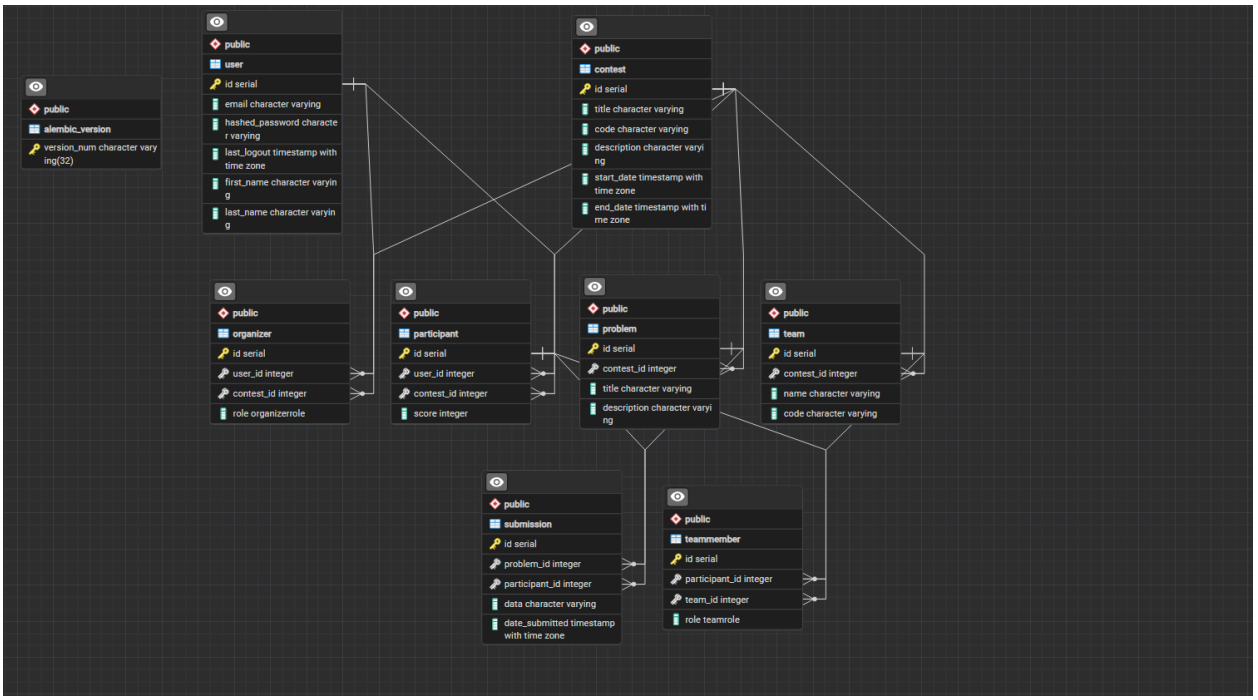
Цель работы:

Научится реализовывать полноценное серверное приложение с помощью фреймворка FastAPI с применением дополнительных средств и библиотек.

Ход работы:

В этой работе я выбрал вариант разработки “системы проведения хакатонов”.

Сначала проектируем модель базы данных:



Создадим соответствующие модели с использованием SQLModel.

```python
class User(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)
    email: EmailStr = Field(unique=True, nullable=False, index=True)
    hashed_password: str
    last_logout: Optional[datetime] = Field(default=None, sa_column=Column(DateTime(timezone=True)))
    first_name: Optional[str] = None
    last_name: Optional[str] = None

    organizers: List["Organizer"] = Relationship(back_populates="user", cascade_delete=True)
    participants: List["Participant"] = Relationship(back_populates="user", cascade_delete=True)

    def __str__(self):
        return (f"User ("
                f"\"id\": {id}, "
                f"\"email\": {self.email}, "
                f"\"first_name\": {self.first_name}, "
                f"\"last_name\": {self.last_name})")

```

```python
class Contest(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)
    title: str = Field(nullable=False)
    code: str = Field(nullable=False)
    description: str = Field(nullable=False)
    start_date: datetime = Field(sa_column=Column(DateTime(timezone=True),  nullable=False))
    end_date: datetime = Field(sa_column=Column(DateTime(timezone=True), nullable=False))

    organizers: List["Organizer"] = Relationship(back_populates="contest", cascade_delete=True)
    participants: List["Participant"] = Relationship(back_populates="contest", cascade_delete=True)
    teams: List["Team"] = Relationship(back_populates="contest", cascade_delete=True)
    problems: List["Problem"] = Relationship(back_populates="contest", cascade_delete=True)

    @model_validator(mode="after")
    def validate_dates(self):
        if self.end_date <= self.start_date:
            raise ValueError("The end_date must be after the start_date.")

        return self
```

```python
class Organizer(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)

    user_id: int = Field(foreign_key="user.id", nullable=False, index=True)
    user: User = Relationship(back_populates="organizers")

    contest_id: int = Field(foreign_key="contest.id", nullable=False, index=True)
    contest: Contest = Relationship(back_populates="organizers")

    role: OrganizerRole

    __table_args__ = (
        UniqueConstraint("user_id", "contest_id", name="uq_user_contest_organizer"),
    )

    def __str__(self):
        return (f"Organizer ("
                f"\"id\": {self.id}, "
                f"\"user_id\": {self.user_id}, "
                f"\"contest_id\": {self.contest_id}, "
                f"\"role\": {self.role})")
```

```python
class Participant(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)

    user_id: int = Field(foreign_key="user.id", index=True, nullable=False)
    user: User = Relationship(back_populates="participants")

    contest_id: int = Field(foreign_key="contest.id", index=True, nullable=False)
    contest: Contest = Relationship(back_populates="participants")

    score: int = Field(default=0, nullable=False)

    team_member: Optional["TeamMember"] = Relationship(back_populates="participant", cascade_delete=True)
    submissions: List["Submission"] = Relationship(back_populates="participant", cascade_delete=True)

    __table_args__ = (
        UniqueConstraint("user_id", "contest_id", name="uq_user_contest_participant"),
    )

    @field_validator("score")
    def validate_score(cls, value):
        if value < 0 or value > 100:
            raise ValueError("Score must be in range [0, 100].")
        return value

    def __str__(self):
        return (f"Participant ("
                f"\"id\": {self.id}, "
                f"\"user_id\": {self.user_id}, "
                f"\"contest_id\": {self.contest_id}, "
                f"\"score\": {self.score})")
```

```python
class Team(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)

    contest_id: int = Field(foreign_key="contest.id", nullable=False)
    contest: Contest = Relationship(back_populates="teams")

    name: str = Field(nullable=False)
    code: str = Field(nullable=False)

    team_members: List["TeamMember"] = Relationship(back_populates="team", cascade_delete=True)
```

```python
class TeamMember(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)

    participant_id: int = Field(foreign_key="participant.id", nullable=False)
    participant: Participant = Relationship(back_populates="team_member")

    team_id: int = Field(foreign_key="team.id", nullable=False)
    team: Team = Relationship(back_populates="team_members")

    role: TeamRole = Field(nullable=False)

    __table_args__ = (
        UniqueConstraint("participant_id", "team_id", name="uq_participant_team"),
    )

    @model_validator(mode="after")
    def validate_contest_participation(self):
        if self.team.contest_id != self.participant.contest_id:
            raise ValueError("team and participant must relate to the same contest")

        return self
```

```python
class Problem(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)

    contest_id: int = Field(nullable=False, foreign_key="contest.id")
    contest: Contest = Relationship(back_populates="problems")

    title: str
    description: str

    submissions: List["Submission"] = Relationship(back_populates="problem", cascade_delete=True)
```

```python
class Submission(SQLModel, table=True):
    id: Optional[int] = Field(default=None, primary_key=True)

    problem_id: int = Field(nullable=False, foreign_key="problem.id")
    problem: Problem = Relationship(back_populates="submissions")

    participant_id: int = Field(nullable=False, foreign_key="participant.id")
    participant: Participant = Relationship(back_populates="submissions")

    data: str = Field(nullable=False)
    date_submitted: datetime = Field(sa_column=Column(DateTime(timezone=True), default=datetime.now().astimezone(), nullable=False))
```

Напишем код для подключения к базе данных:

students > k33391 > Mikitchak_Ivan > Lr1 > Lab > connection.py > ...

```python
from sqlmodel import create_engine, SQLModel, Session
from dotenv import load_dotenv
import os


load_dotenv()
db_url = os.getenv("DB_URL")
engine = create_engine(db_url, echo=False)


def init_db():
    SQLModel.metadata.create_all(engine)

def get_session():
    with Session(engine) as session:
        yield session

```

Далее создадим следующие эндпоинты:

Авторизация:

- **`POST /auth/login`**

  Эндпоинт для приобретения токена.

- **POST /auth/logout**

  Эндпоинт для инвалидации всех ранее выданных токенов

Пользователи:

- **POST /users**

  Эндпоинт для создания нового пользователя (регистрации)

- **GET /users**

  Эндпоинт для извлечения списка пользователей (поиска)

- **GET /users/me**

  Эндпоинт для извлечения моего пользователя (соответствующего токену)

- **PUT /users/me**

  Эндпоинт для обновления полей моего пользователя

- **DELETE /users/me**

  Эндпоинт для удаления моего пользователя

- **POST /users/update_email**

  Эндпоинт для обновления электронной почты

- **POST /users/update_password**

  Эндпоинт для обновления пароля

- **GET /users/{user_id}**

  Эндпоинт для извлечения пользователя по id

Соревнования:

- **POST /contests**

  Эндпоинт для создания нового соревнования

- **GET /contests**

  Эндпоинт для извлечения списка соревнований (поиска)

- **GET /contests/{contest_id}**

  Эндпоинт для извлечения соревнования по id

- **PUT /contests/{contest_id}**

  Эндпоинт для обновления полей соревнования

- **DELETE /contests/{contest_id}**

  Эндпоинт для удаления соревнования

- **POST /contests/{contest_id}/update_code**

  Эндпоинт для обновления кода соревнования

Организаторы:

- **POST /contests/{contest_id}/organizers**

  Эндпоинт для создания нового организатора соревнования (присоединиться как организатор)

- **GET /contests/{contest_id}/organizers**

  Эндпоинт для извлечения списка организаторов

- **GET /contests/{contest_id}/organizers/me**

  Эндпоинт для извлечения моего организатора

- **DELETE /contests/{contest_id}/organizers/me**

  Эндпоинт для удаления моего организатора (покинуть соревнование как организатор)

- **GET /contests/{contest_id}/organizers/me/{organizer_id}**

  Эндпоинт для извлечения организатора по id

- **DELETE /contests/{contest_id}/organizers/me/{organizer_id}**

  Эндпоинт для удаления организатора по id (бан)

Участники:

- **POST /contests/{contest_id}/participants**

  Эндпоинт создания нового участника соревнования (присоединиться как участник)

- **GET /contests/{contest_id}/participants**

  Эндпоинт для извлечения списка участников соревнования (поиска)

- **GET /contests/{contest_id}/participants/me**

  Эндпоинт для извлечения моего участника

- **DELETE /contests/{contest_id}/participants/me**

  Эндпоинт для удаления моего участника (покинуть соревнование как участник)

- **GET /contests/{contest_id}/participants/{participant_id}**

  Эндпоинт для извлечения участника по id

- **DELETE /contests/{contest_id}/participants /{participant_id}**

  Эндпоинт для удаления участника по id (бан)

- **POST /contests/{contest_id}/participants/{participant_id} /score**

  Эндпоинт для обновления счета участника

Команды:

- **POST /contests/{contest_id}/teams**

  Эндпоинт для создания новой команды
- **GET /contests/{contest_id}/teams**

  Эндпоинт для извлечения списка команд (поиска)
- **GET /contests/{contest_id}/teams/my**

  Эндпоинт для извлечения моей команды
- **PUT /contests/{contest_id}/teams/my**

  Эндпоинт для обновления полей моей команды
- **DELETE /contests/{contest_id}/teams/my**

  Эндпоинт для удаления моей команды
- **POST /contests/{contest_id}/teams/my/update_code**

  Эндпоинт для обновления кода моей команды
- **GET /contests/{contest_id}/teams/{team_id}**

  Эндпоинт для извлечения команды по id

Члены команд:

- **GET /contests/{contest_id}/teams/my/members**

  Эндпоинт для извлечения списка участников моей команды
- **GET /contests/{contest_id}/teams/my/members/me**

  Эндпоинт для извлечения меня как участника команды
- **DELETE /contests/{contest_id}/teams/my/members/me**

  Эндпоинт для удаления меня как участника команды (покинуть команду)
- **GET /contests/{contest_id}/teams/my/members/{member_id}**

  Эндпоинт для извлечения участника команды по id
- **DELETE /contests/{contest_id}/teams/my/members/{member_id}**

  Эндпоинт для удаления участника команды по id (бан)
- **POST /contests/{contest_id}/teams/{team_id}/members**

  Эндпоинт для создания нового участника команды (присоединиться к команде)
- **GET /contests/{contest_id}/teams/{team_id}/members**

  Эндпоинт для извлечения списка участников команды

- **GET /contests/{contest_id}/teams/{team_id}/members/{member_id}**

  Эндпоинт для извлечения участника команды

Задания:

- **POST /contests/{contest_id}/problems**

  Эндпоинт для создания задачи

- **GET /contests/{contest_id}/problems**

  Эндпоинт для извлечения списка задач

- **GET /contests/{contest_id}/problems/{problem_id}**

  Эндпоинт для извлечения задачи по id

- **PUT /contests/{contest_id}/problems/{problem_id}**

  Эндпоинт для обновления задачи

- **DELETE /contests/{contest_id}/problems/{problem_id}**

  Эндпоинт для удаления задачи

Решения:

- **POST /contests/{contest_id}/problems/{problem_id}/submissions**

  Эндпоинт для создания решения к задаче (отправить решение)

- **GET /contests/{contest_id}/problems/{problem_id}/submissions**

  Эндпоинт для извлечения списка решений к задаче (поиска)

- **GET /contests/{contest_id}/problems/{problem_id}/submissions/{submission_id}**

  Эндпоинт для извлечения решения по id

Ссылки:

- Документация: https://github.com/k0jumba/ITMO_ICT_WebDevelopment_tools_2023-2024/tree/lab1/students/k33391/Mikitchak_Ivan/Lr1/Lab/docs
- Код лабораторной: https://github.com/k0jumba/ITMO_ICT_WebDevelopment_tools_2023-2024/tree/lab1/students/k33391/Mikitchak_Ivan/Lr1/Lab
- Код практик: https://github.com/k0jumba/ITMO_ICT_WebDevelopment_tools_2023-2024/tree/lab1/students/k33391/Mikitchak_Ivan/Lr1/Practice