**Министерство науки и высшего образования Российской Федерации**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Информационная безопасность баз данных»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

«Реализация БД в рамках СУБД»

**Выполнили:**

Ахраров Али, студент группы N3350

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**Проверил:**

Салихов Максим Русланович

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(отметка о выполнении)

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

1. **Цель работы:**

Получение навыков использования серверных языков программирования, фреймворков по работе с БД, ORM-систем

Задание

1.      На основе БД, созданной в предыдущих лабораторных работах, необходимо создать сервис, который взаимодействует с разработанной БД. В качестве сервиса среди прочего могут быть разработаны:

- веб-интерфейс;

- desktop приложение;

- API;

- сервисы предобработки данных, которые заполняют базу данных (обработка websocket соединений, парсинг страниц, json и др)

- сервисы для защиты данных при их обработке, записи, считывании из/в БД (шифрование, обезличивание, сетевая защита и др.)

 (*список не исчерпывает все возможные сервисы, см. п.4 в примечании*)

Дайте краткое обоснование выбранной среде разработки, языку и фреймворку, который вы используете для взаимодействия с базой данных.

2.  В рамках ЛР опишите каким образом вы взаимодействуете с базой данных. С помощью каких фреймворков или языков. Продемонстрируйте несколько примеров по выборке, вставке, удалении данных из вашей БД с помощью выбранного вами фреймворка или языка программирования. Проконтролируйте корректность работы систем логирования и разграничения доступа, которые реализованы в 1-3 ЛР.

3. Для реализованного сервиса укажите кратко его основные функции, назначение, кратко опишите структуру. Дополнительно опишите и оцените методы защиты данных, реализованные вами и имеющиеся в наличии в разработанном сервисе. Выделите аналоги, недостатки и преимущества решений, которые вы реализовали.

# Взаимодействие с базой данных: используемые технологии и методы

Для разработки веб-сервиса, взаимодействующего с базой данных, был выбран язык программирования \*\*Python\*\* и фреймворк \*\*Flask\*\*. Основными причинами выбора Python являются его простота, богатая экосистема библиотек и хорошая интеграция с базами данных. Flask был выбран как легкий фреймворк, подходящий для создания API и веб-интерфейсов. Библиотека \*\*SQLAlchemy\*\* использовалась для удобного взаимодействия с базой данных PostgreSQL с помощью ORM (Object-Relational Mapping), что позволило писать запросы на уровне Python-кода без необходимости писать SQL напрямую.

**Установка и настройка окружения**

1. Установка необходимых библиотек:

pip install sqlalchemy psycopg2 flask

* SQLAlchemy: для работы с ORM и выполнения операций с базой данных.
* psycopg2: драйвер для подключения к PostgreSQL.
* Flask: фреймворк для создания веб-приложения.

2. Конфигурация подключения к базе данных осуществляется через SQLAlchemy. Пример кода для подключения:

from sqlalchemy import create\_engine

from sqlalchemy.orm import sessionmaker

DATABASE\_URL = 'postgresql+psycopg2://username:password@localhost/db\_name'

engine = create\_engine(DATABASE\_URL)

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

3. Для работы с таблицами и их отображения в Python-классы использовались ORM-классы. Пример модели `Teacher`:

from sqlalchemy import Column, Integer, String

from sqlalchemy.ext.declarative import declarative\_base

Base = declarative\_base()

class Teacher(Base):

\_\_tablename\_\_ = 'teachers'

teacher\_id = Column(Integer, primary\_key=True, index=True)

teacher\_firstname = Column(String, nullable=False)

teacher\_lastname = Column(String, nullable=False)

teacher\_email = Column(String, unique=True, nullable=False)

**Примеры выполнения операций с базой данных**

1. Выборка данных (SELECT):

Для получения списка преподавателей из базы данных используется следующий код:

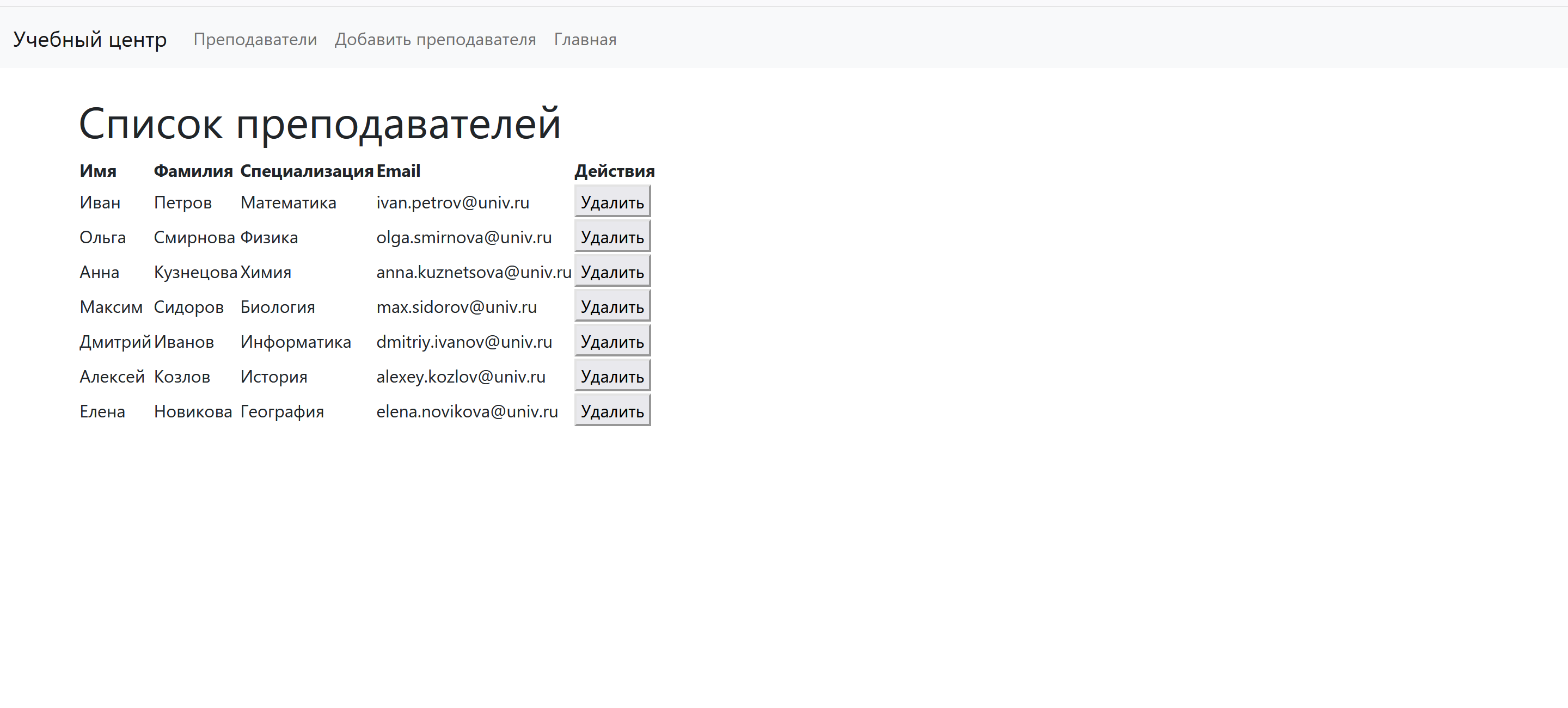
def get\_teachers():

db = SessionLocal()

teachers = db.query(Teacher).all()

db.close()

return teachers



*Рисунок 1* - Пример вывода списка преподавателей

2. Вставка данных (INSERT):

Для добавления нового преподавателя в базу данных используется следующий код:

def add\_teacher(firstname, lastname, email):

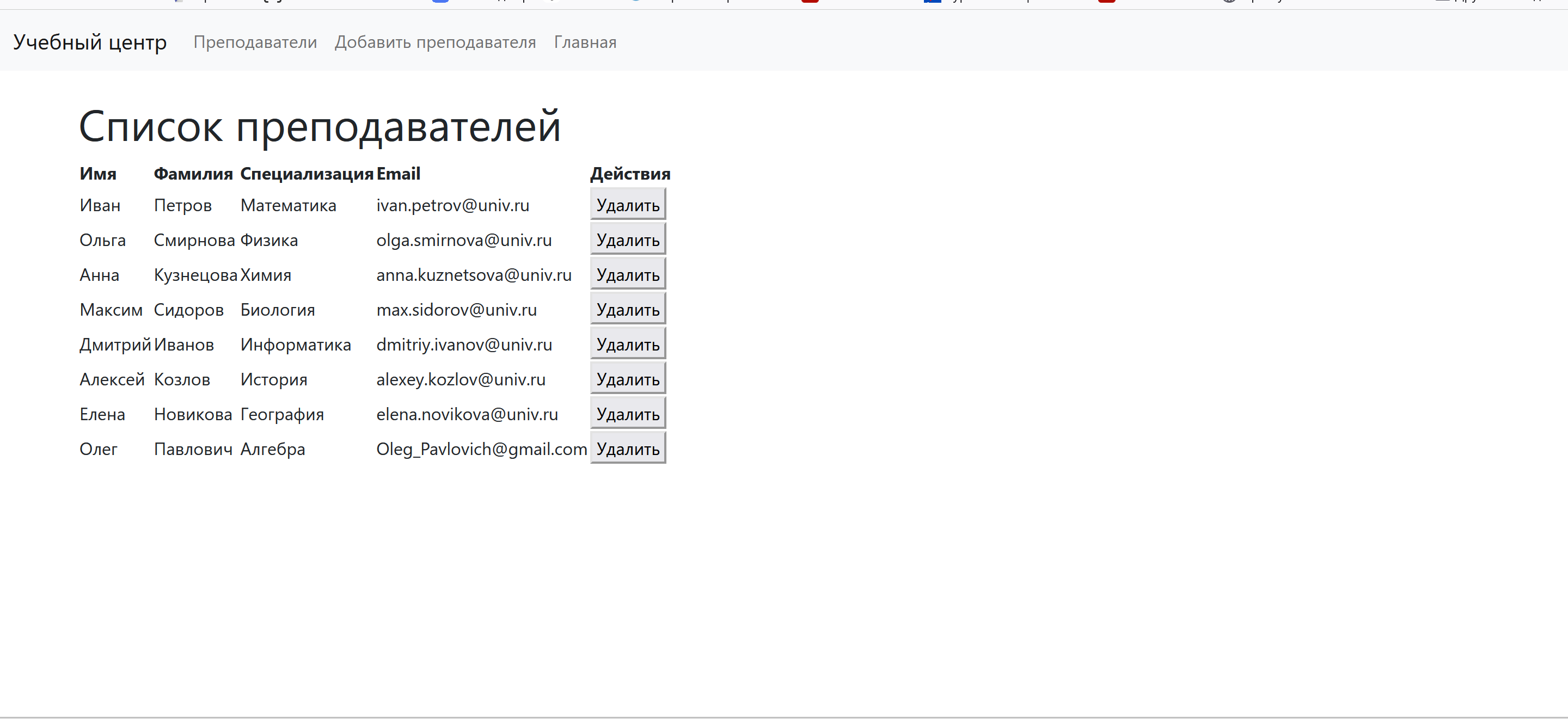
db = SessionLocal()

new\_teacher = Teacher(teacher\_firstname=firstname, teacher\_lastname=lastname, teacher\_email=email)

db.add(new\_teacher)

db.commit()

db.close()



*Рисунок 2* - Пример добавления нового преподавателя

3. Удаление данных (DELETE):

Для удаления преподавателя по ID используется следующий код:

def delete\_teacher(teacher\_id):

db = SessionLocal()

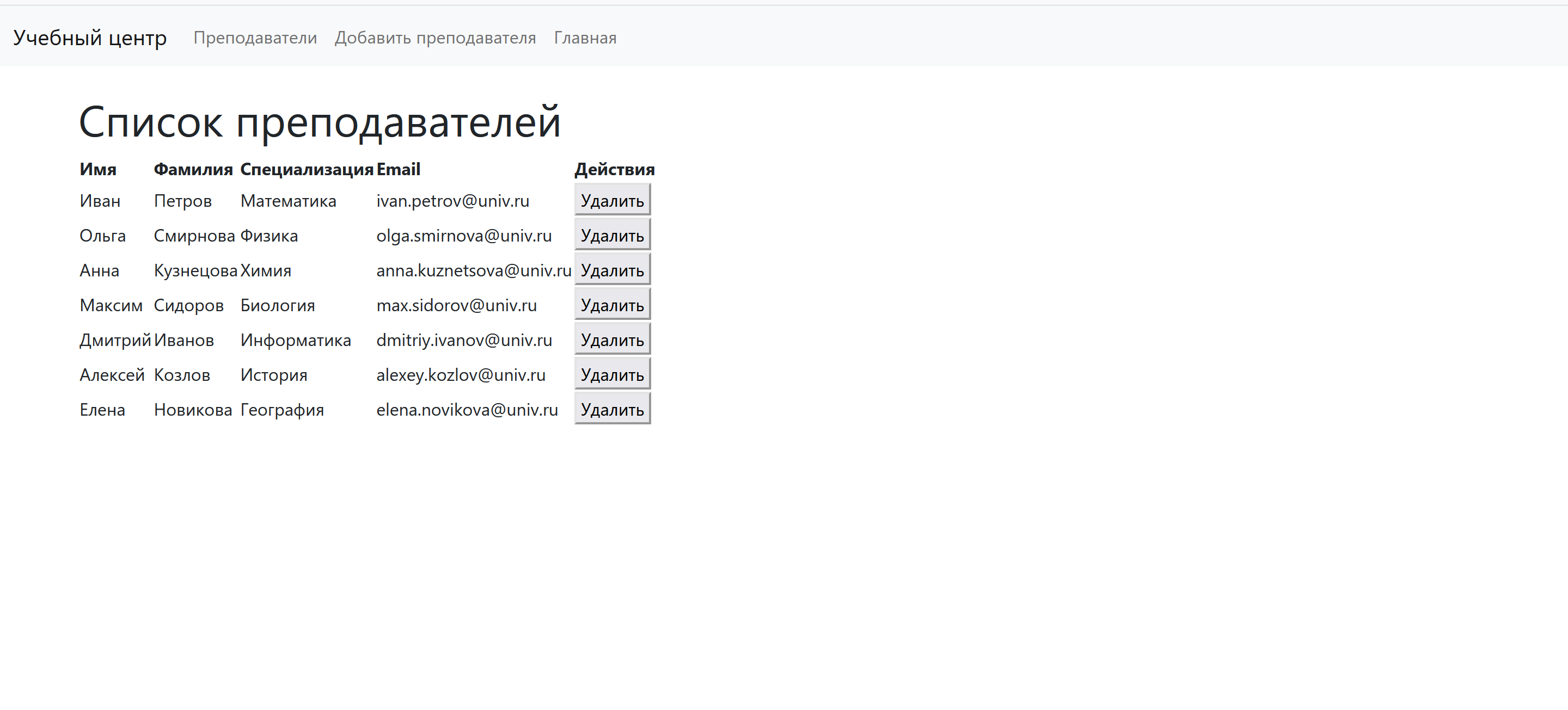
teacher = db.query(Teacher).filter(Teacher.teacher\_id == teacher\_id).first()

if teacher:

db.delete(teacher)

db.commit()

db.close()



*Рисунок 3* - Пример удаления преподавателя из базы данных

**Система логирования и разграничения доступа**

1.Логирование:

- Для контроля за действиями в приложении используется модуль `logging`. Логирование настраивается для фиксации важных действий, таких как добавление и удаление записей, а также для регистрации ошибок.

- Пример настройки логирования:

python

import logging

logging.basicConfig(level=logging.INFO)

logger = logging.getLogger(\_\_name\_\_)

- Логирование операций:

def add\_teacher(firstname, lastname, email):

db = SessionLocal()

try:

new\_teacher = Teacher(teacher\_firstname=firstname, teacher\_lastname=lastname, teacher\_email=email)

db.add(new\_teacher)

db.commit()

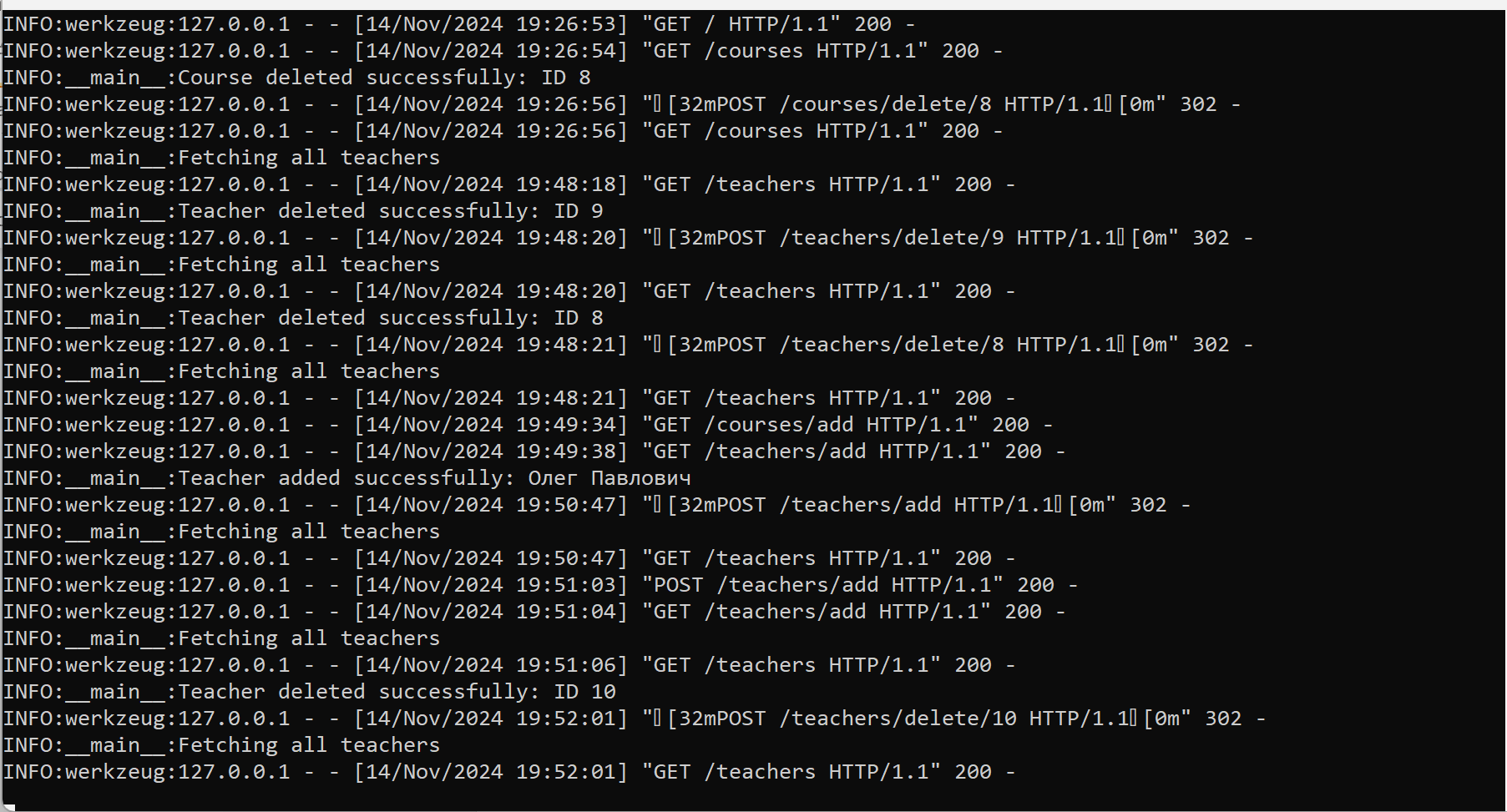
logger.info(f"Added teacher: {firstname} {lastname}")

except Exception as e:

logger.error(f"Error adding teacher: {e}")

finally:

db.close()

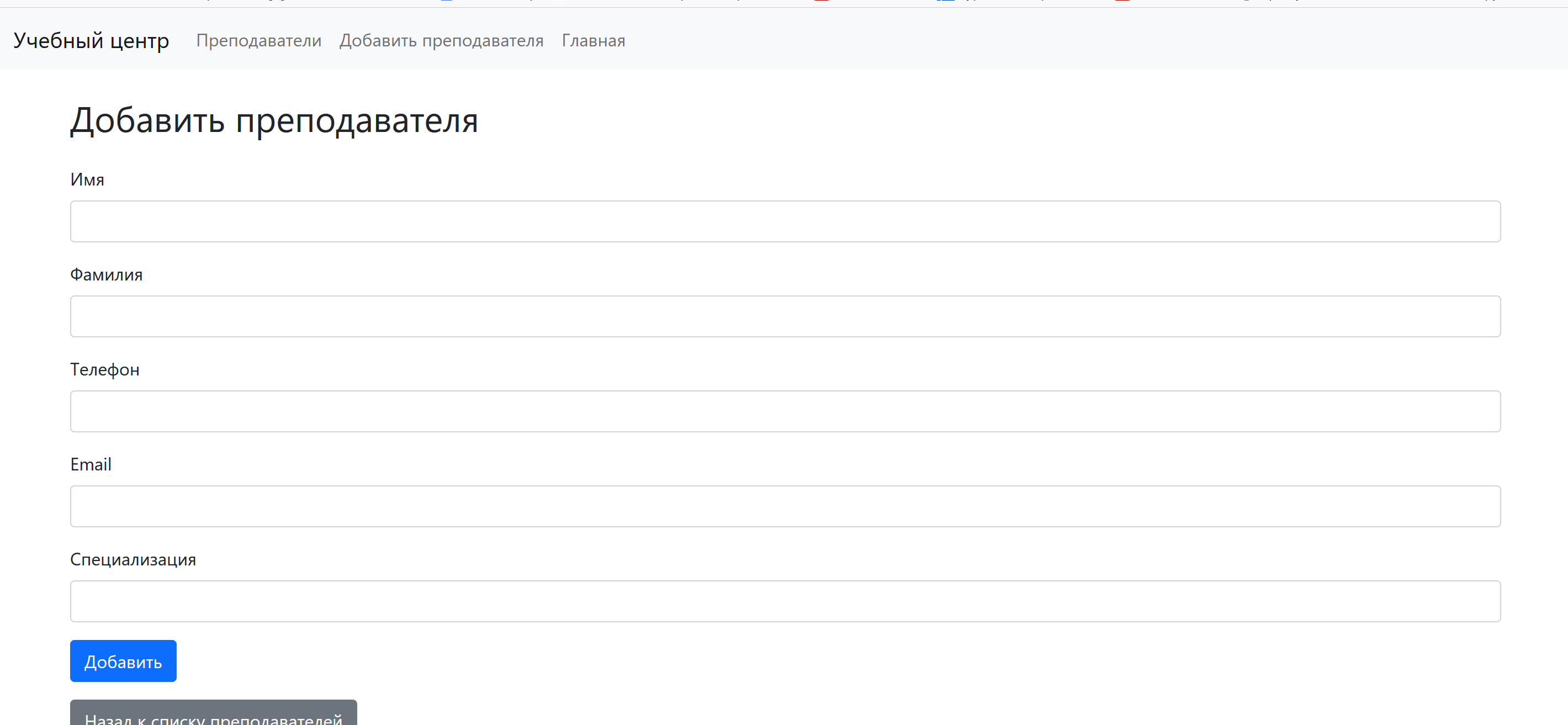


*Рисунок 4* - Пример записи в лог при добавлении преподавателя\*\*

1. Добавление преподавателей и курсов:

- Форма для ввода данных преподавателя и курса.

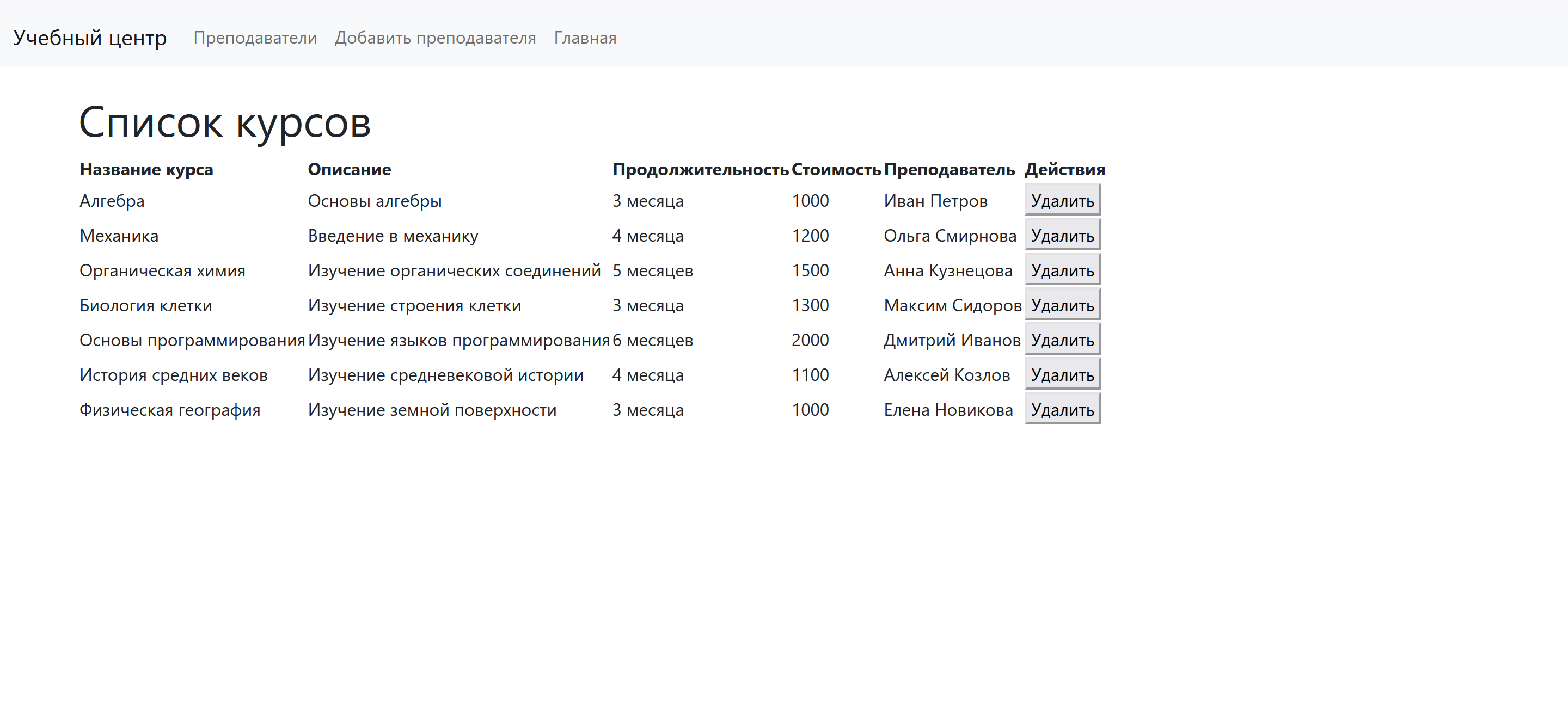
- Обработка данных, включая проверку уникальности полей, таких как email.



*Рисунок 5* - Пример формы добавления преподавателя

2. Просмотр списка преподавателей и курсов:

- Отображение данных из базы данных в виде таблицы.



*Рисунок 6* - Пример вывода списка курсов

4. Логирование и проверка корректности данных:

- Логирование позволяет отслеживать действия пользователей, а также фиксировать ошибки для упрощения отладки.

**Структура сервиса**

1. Главный модуль `app.py`: Управляет маршрутизацией и обработкой запросов.

2. Модуль `database.py`: Настраивает подключение к базе данных и ORM.

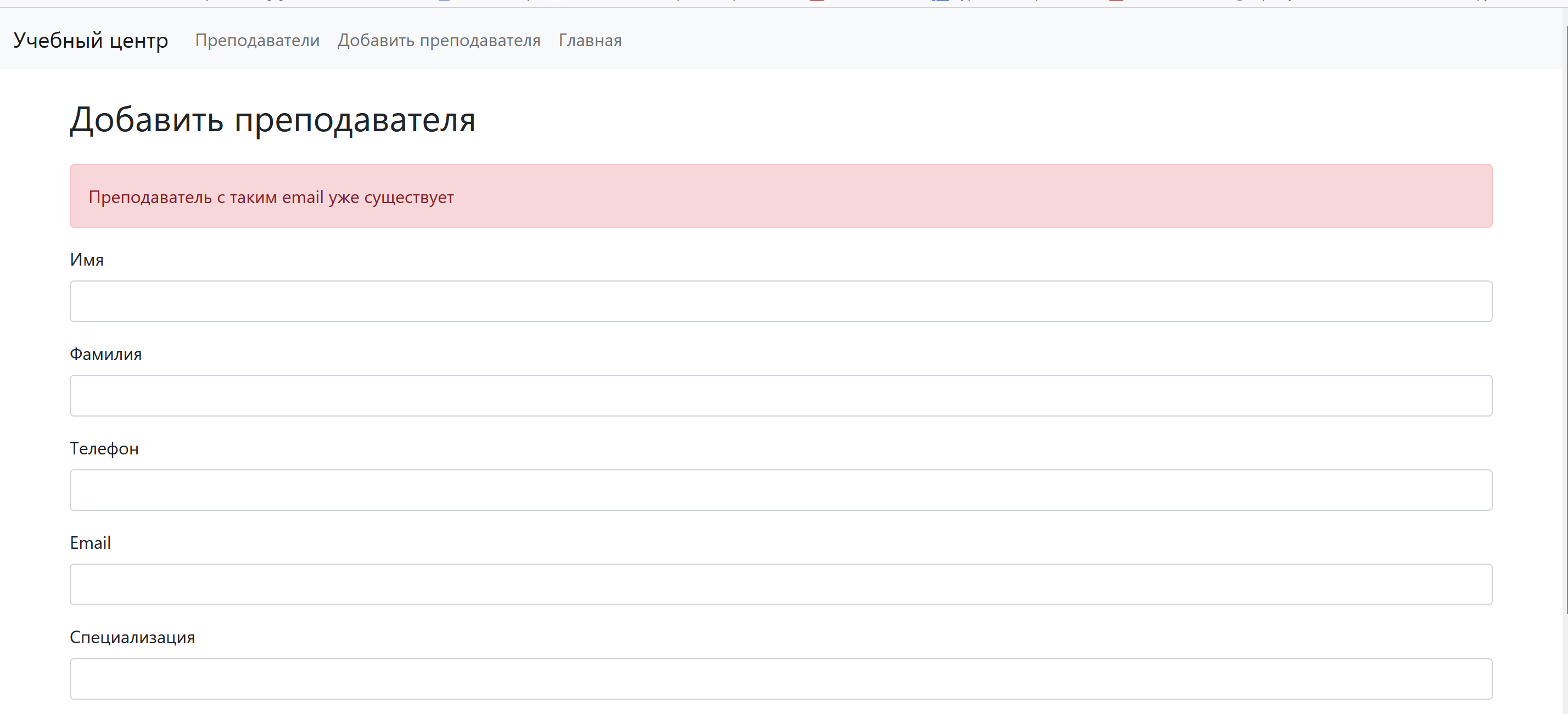
3. Модели данных: ORM-классы, описывающие структуру таблиц базы данных.

4. Шаблоны HTML: Главная страница сайта.

**Методы защиты данных**

1. Валидация данных:

- Весь ввод данных проходит через проверку на сервере. Например, при добавлении преподавателя проверяется уникальность email.



*Рисунок 9* - Пример работы системы валидации данных

**Заключение**

Разработанный сервис обеспечивает базовый функционал для управления данными учебного заведения. Он включает защитные механизмы, логирование и разграничение доступа, что позволяет использовать его в многопользовательской среде с базовыми требованиями к безопасности. Улучшения, предложенные выше, позволят значительно расширить функциональные возможности и повысить безопасность решения, делая его более пригодным для применения в реальной среде.