МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

**Направление подготовки: 10.03.01 Информационная безопасность**

**Образовательная программа: "Информационная безопасность / Information security"**

**Дисциплина:**

**«Web программирование»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1**

**«Разработка Веб сервиса»**

**Выполнил студент(ы):**

Ахаров Али Рустамович

**Проверил:**

\_\_Менщиков Александр Алексеевич\_\_/\_\_\_\_\_\_\_\_\_\_

*ФИО Подпись*

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

*Отметка о выполнении (один из вариантов:*

*отлично, хорошо, удовлетворительно, зачтено)*

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

*Дата*

Санкт-Петербург

2024г.

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc182235189)

[ВВЕДЕНИЕ 4](#_Toc182235190)

[1. Описание архитектуры ПО 5](#_Toc182235191)

[2. Описание структуры базы данных 6](#_Toc182235192)

[3. Описание протокола и форматов передачи данных 7](#_Toc182235193)

[4. Описание API 9](#_Toc182235194)

[5. Тестирование функционала ПО 10](#_Toc182235195)

[ЗАКЛЮЧЕНИЕ 15](#_Toc182235196)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc182235197)

[ПРИЛОЖЕНИЕ 17](#_Toc182235198)

**1**. Назначение и общая архитектура проекта  
Данный проект представляет собой полноценное веб-приложение для просмотра, добавления, редактирования и удаления данных о студентах. Основные задачи, решаемые системой:

* Хранение информации о студентах в базе данных (ФИО, курс, группа, факультет).
* Возможность фильтровать студентов по фамилии.
* Управление записями: добавление новых студентов, редактирование существующих и удаление из списка.

Архитектура решения разделена на три основные компоненты:

* **База данных (PostgreSQL)**: Хранит информацию о студентах в реляционной таблице. Инициализация базы осуществляется с помощью скрипта init.sql.
* **Серверная часть (API на FastAPI)**: Предоставляет REST API для работы с данными. API отвечает за получение списка студентов с фильтрацией, создание новых записей, обновление существующих и удаление. Взаимодействие с БД осуществляет через ORM SQLAlchemy.
* **Клиентская часть (Frontend на React + Material UI)**: Отображает таблицу со студентами, форму для добавления/редактирования, функционал поиска и пагинацию. Интерфейс выполнен в современном стиле, применяются стили Material UI и кастомный дизайн (темные фоны, яркие акценты).

**2.** Технологический стек

* **База данных**: PostgreSQL (поднимается в контейнере Docker).
* **Backend**: Python 3.10, FastAPI, SQLAlchemy, psycopg2, Uvicorn для запуска сервера.
* **Frontend**: React, Material UI (MUI), Axios для запросов к API.
* **Docker/Docker Compose**: Все сервисы (frontend, backend, db) запускаются в изолированных контейнерах.

**3**. Деплой и запуск  
Проект запускается с помощью docker-compose.yml. Основные шаги:

1. Установить Docker и Docker Compose.
2. В корневой директории, где расположен docker-compose.yml, выполнить команду docker-compose up --build.
3. После сборки и запуска контейнеров:
   * База данных поднимется на порту 5433 (перенаправляется на внутренний порт 5432).
   * Backend (API) будет доступен по адресу http://localhost:3000.
   * Frontend будет доступен по адресу http://localhost:80.

**4.** Структура приложения

* **db/**:
  + Dockerfile для сборки контейнера с PostgreSQL.
  + Файл init.sql для инициализации таблицы и наполнения тестовыми данными о студентах.
* **api/**:
  + app/main.py: Основная точка входа в backend. Определение роутов для получения, создания, обновления и удаления студентов.
  + app/models.py: Определение модели таблицы students.
  + app/schemas.py: Pydantic-схемы для валидации входных и выходных данных.
  + app/crud.py: Функции для работы с базой данных (создание, чтение, обновление, удаление).
  + app/database.py: Настройка подключения к базе, создание сессии и базы данных.
  + Dockerfile и requirements.txt для сборки и установки зависимостей backend.
* **frontend/**:
  + Исходный код React-приложения в папке src/.
  + App.jsx: Главный компонент, в котором отображаются таблица со студентами, форма добавления/редактирования, панель поиска и пагинации.
  + Компоненты для таблицы студентов, формы, элементов управления страницами, а также настройки темы (MUI) и стилей.
  + Dockerfile для сборки frontend.
* **docker-compose.yml**: Файл, описывающий сборку и запуск всех трех сервисов (db, api, frontend) в контейнерах.

**5.** Основные возможности приложения

* **Просмотр списка студентов**: В таблице на фронтенде отображаются студенты, полученные от API. Данные можно пролистывать по страницам.
* **Фильтрация по фамилии**: Вводя часть фамилии в поле поиска, можно фильтровать вывод, отображая только тех студентов, чьи фамилии соответствуют введенному критерию.
* **Добавление студента**: Кнопка "Добавить" открывает форму, где можно ввести все данные о новом студенте. После сохранения происходит запрос к API, обновление базы и отображение нового студента в таблице.
* **Редактирование студента**: Каждая строка таблицы имеет кнопку "Редактировать". Нажатие на нее отобразит форму с данными выбранного студента, которые можно изменить. После сохранения обновленная запись отразится в таблице.
* **Удаление студента**: Кнопка "Удалить" позволяет убрать запись о студенте из базы. После успешного запроса таблица обновится.

**6. Возможные варианты использования и тестирование**

* **Запуск через Docker Compose**: После выполнения docker-compose up открыть в браузере http://localhost:80.

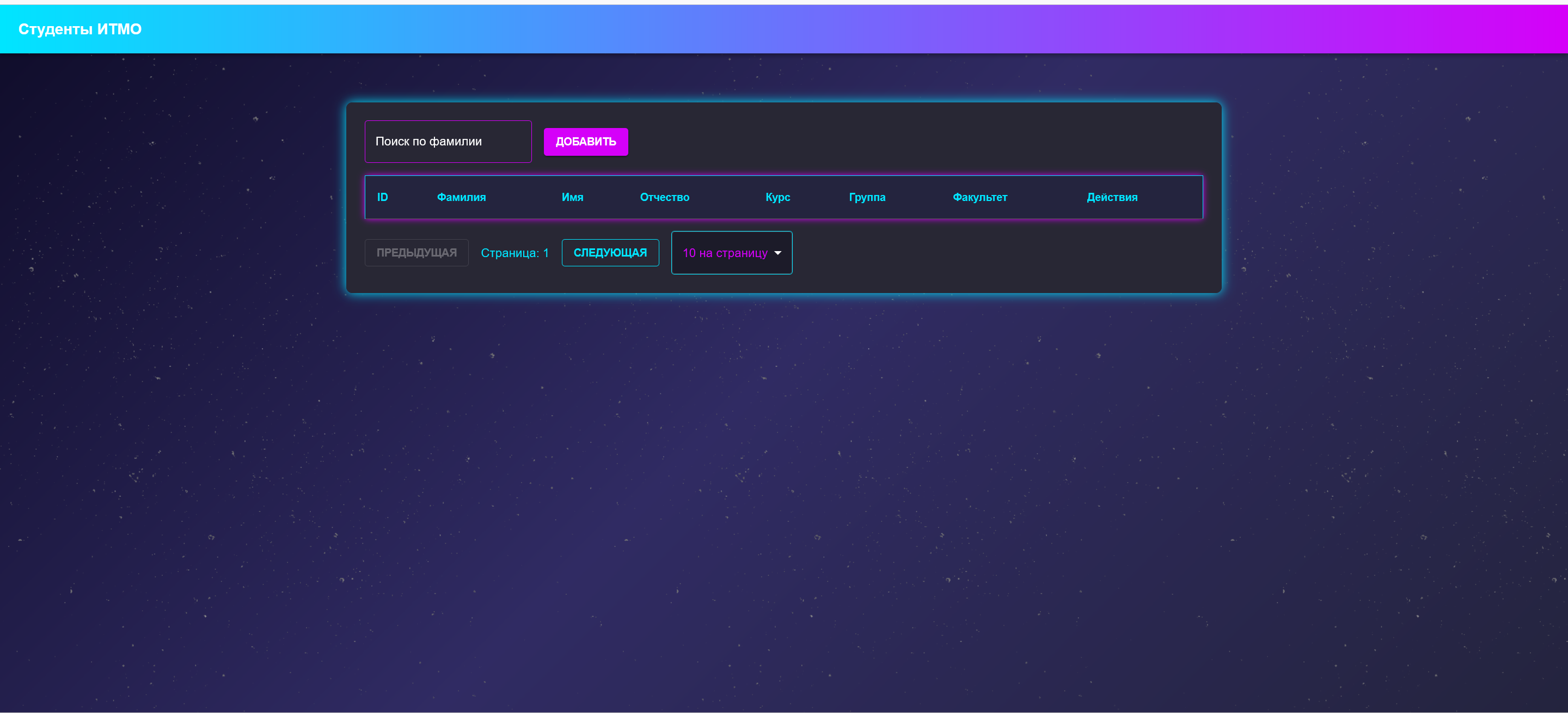


Рис.1 – главная страница сайта

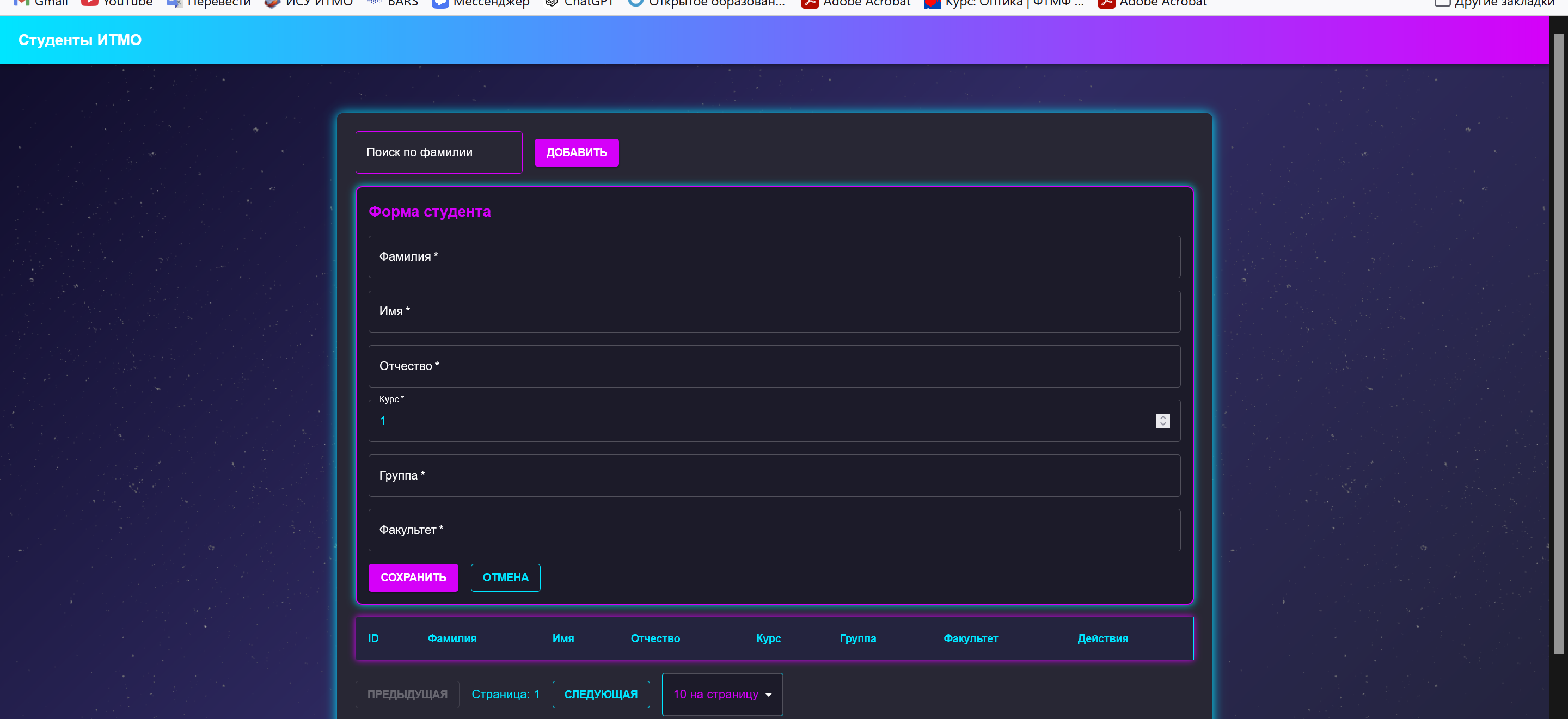


Рис.2 – возможно добавлять новых студентов.

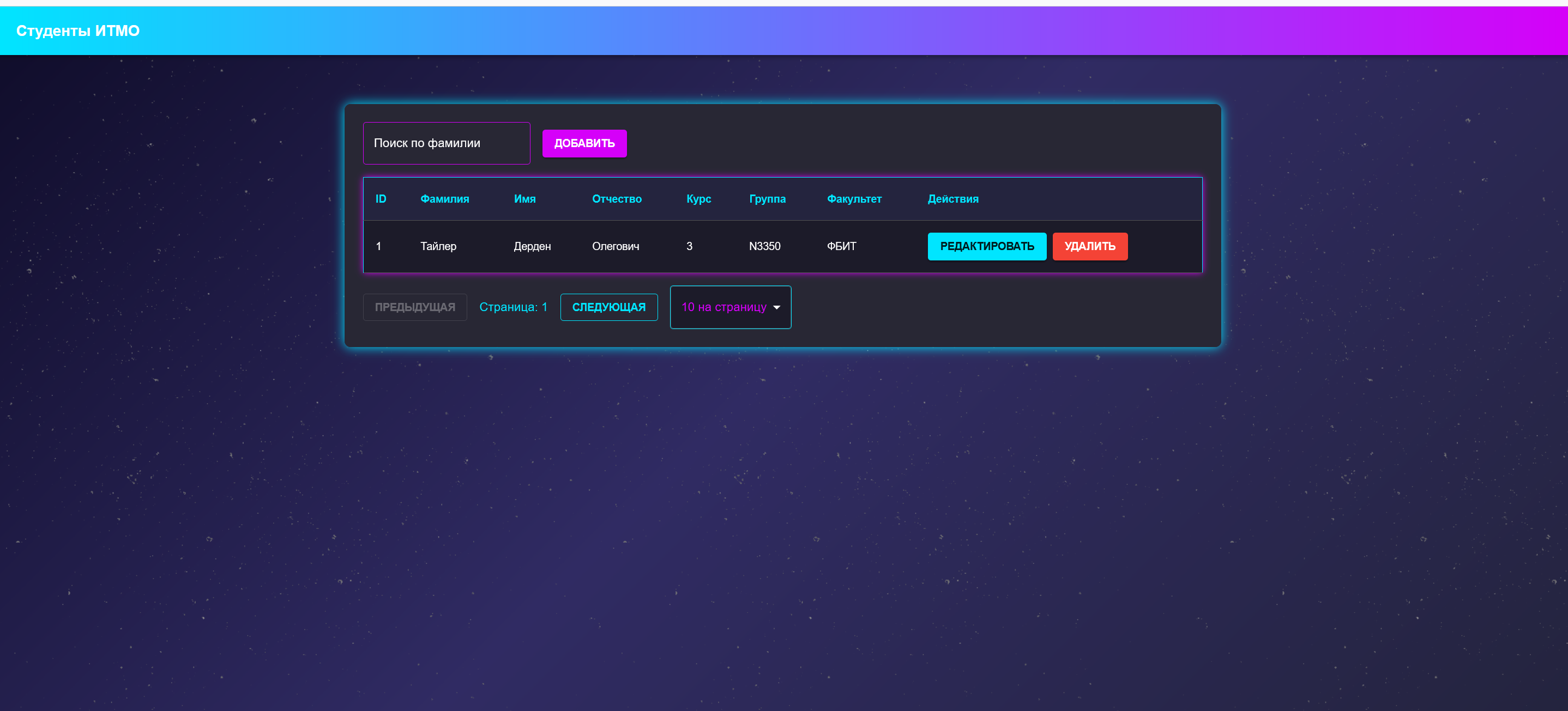


Рис 3 – пример добавления нового студента

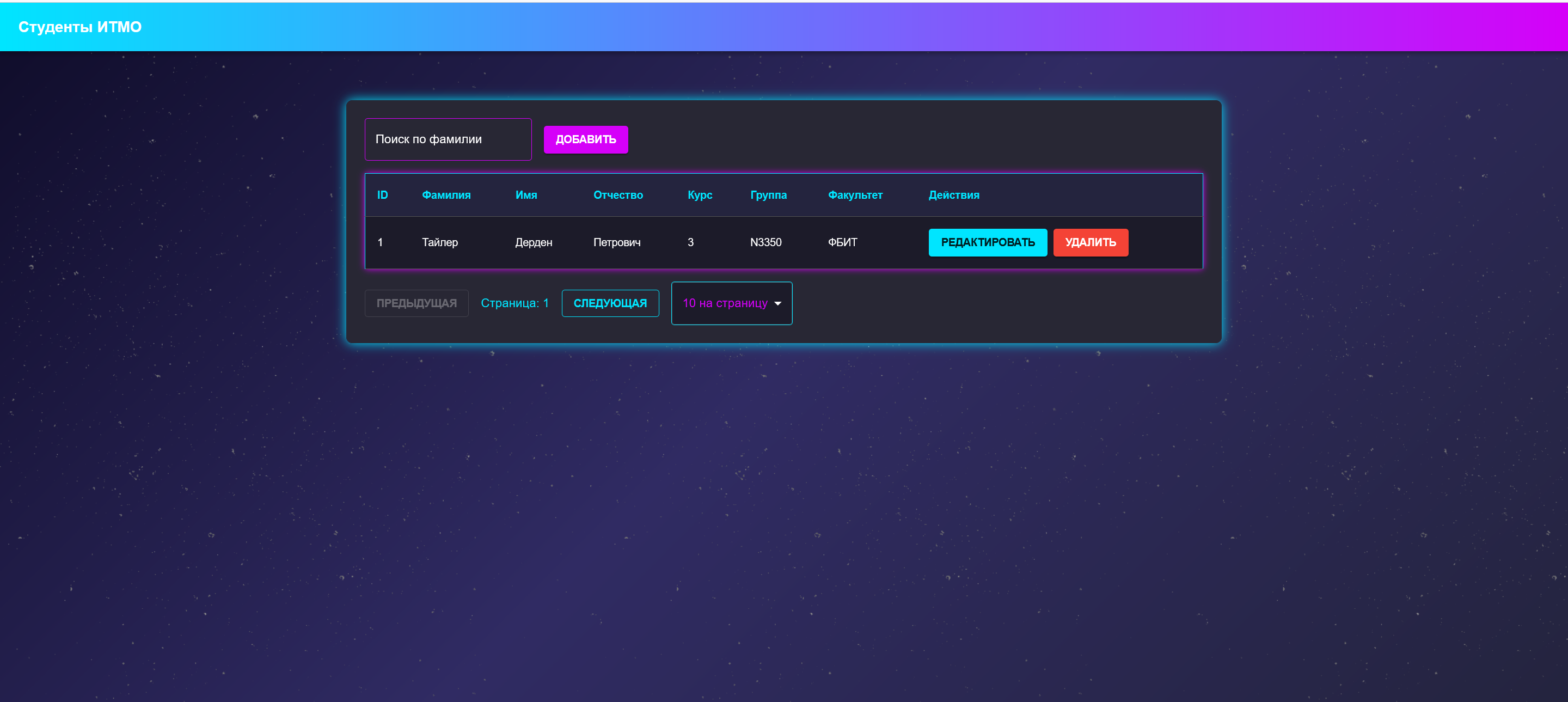


Рис. 4 – пример редактирования записи (отчество)

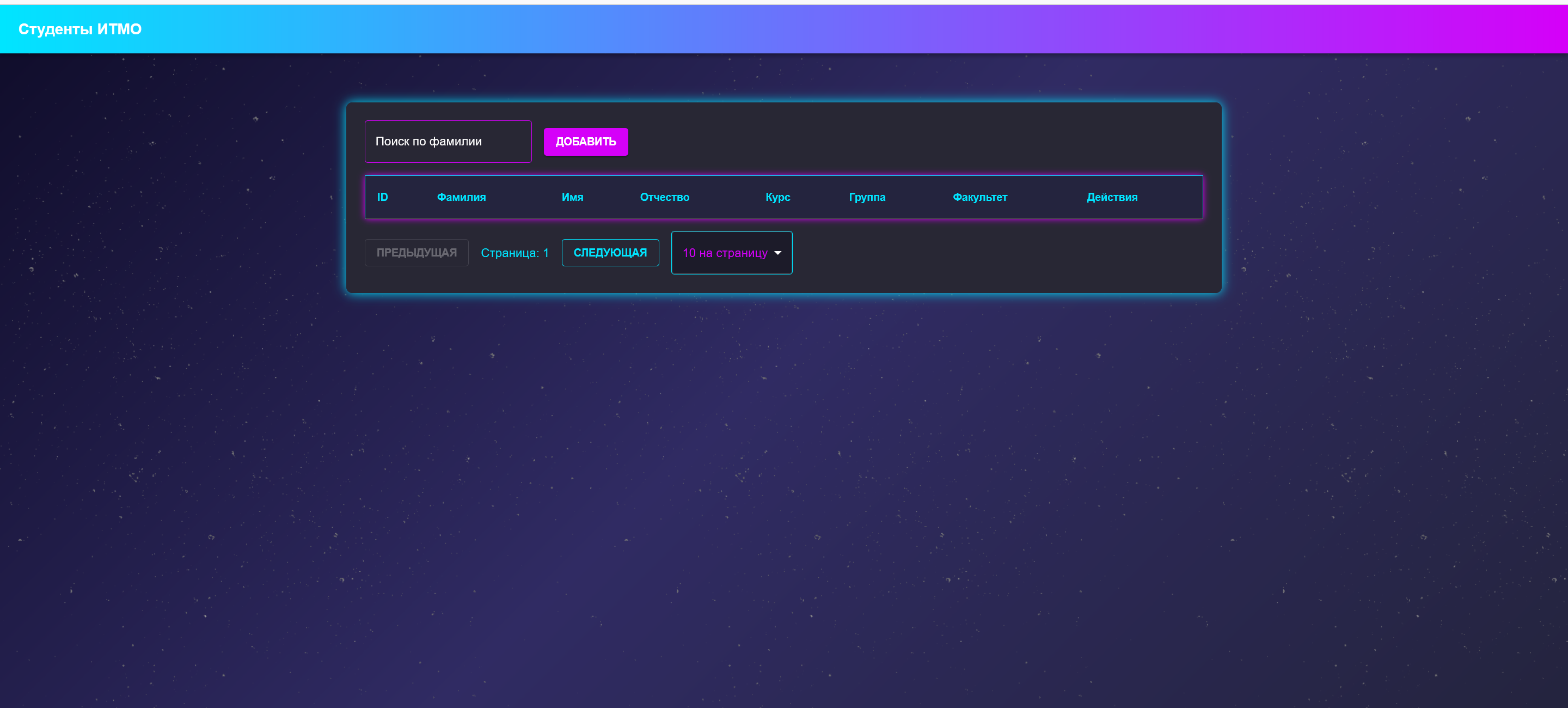


Рис. 5 – пример удаления студена

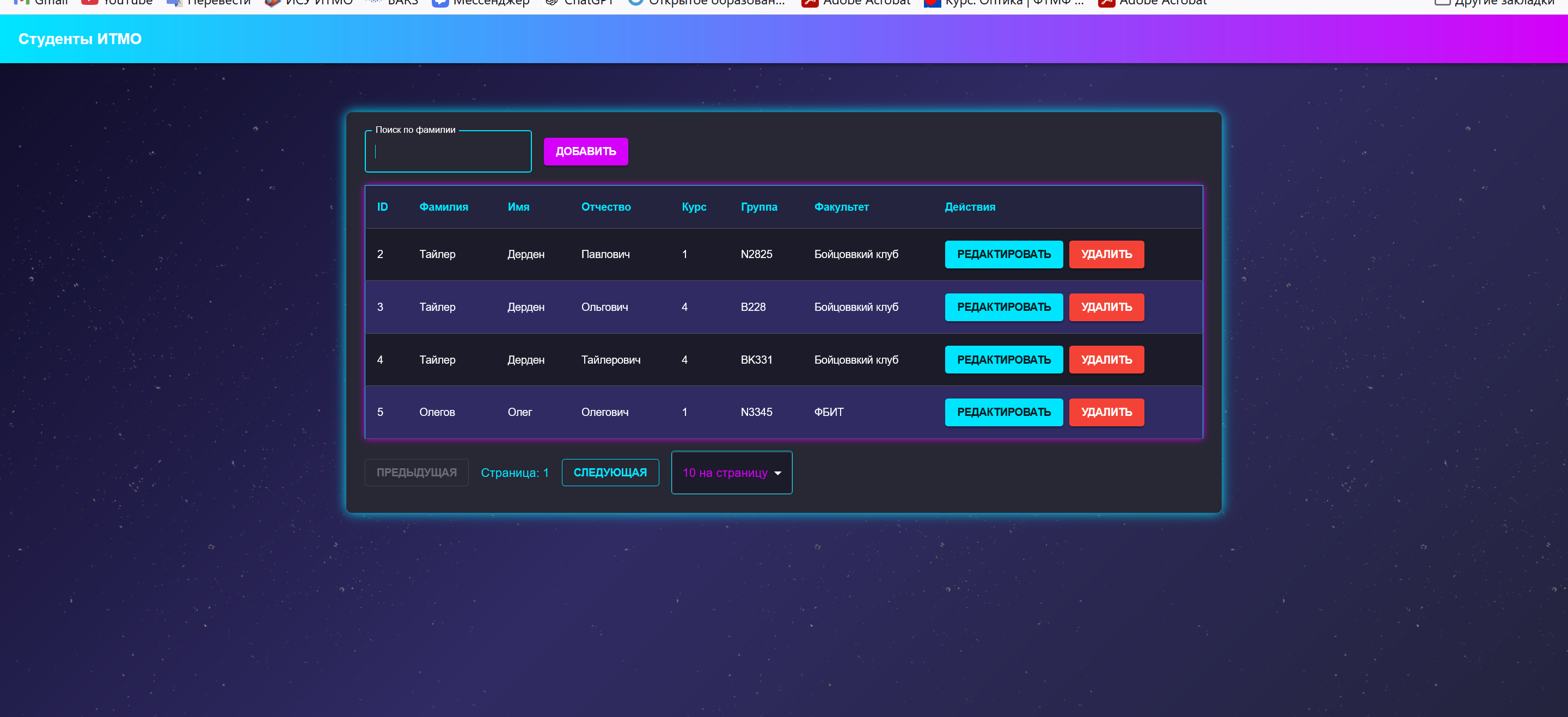


Рис.5 – добавление большего кол-ва студенов

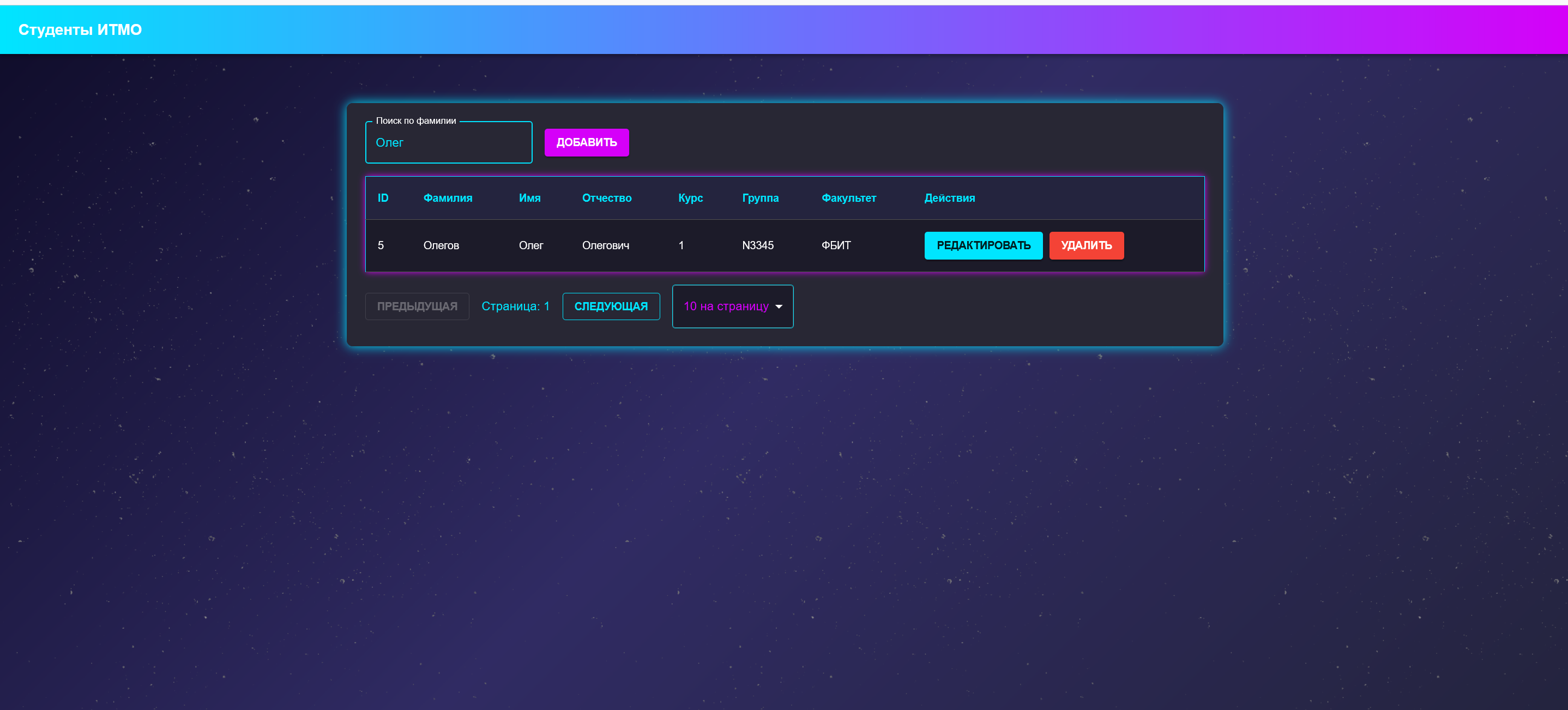


Рис.6 – поиск студентов

**Заключение**

В ходе выполнения лабораторной работы была разработана полнофункциональная веб-система, включающая в себя серверную, клиентскую часть и базу данных. Были достигнуты следующие цели:

1. Реализована структура базы данных, обеспечивающая хранение и управление записями о студентах.
2. Создан REST API на базе FastAPI, позволяющий выполнять основные CRUD-операции с данными.
3. Разработан современный интерфейс на React с использованием Material UI, предоставляющий пользователю удобные инструменты для поиска, фильтрации, добавления, редактирования и удаления данных.
4. Внедрены механизмы деплоя с использованием Docker Compose для обеспечения изоляции и простоты развертывания проекта.

Проект успешно протестирован, все заявленные функциональные возможности были реализованы и протестированы. Результаты демонстрируют соответствие выполненной работы поставленным задачам.

Данный опыт позволил углубить знания в области веб-разработки и научиться эффективно применять современные инструменты и технологии.