**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра Информационных Систем**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Управление данными»**

**Тема: База данных для деканата**

**Вариант 7**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Полуянов В. Н. |
| Преподаватель |  | Татарникова Т. М. |

Санкт-Петербург

2024

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

**Студент** Полуянов В.Н.

**Группа** 2372

**Тема работы:** База данных для деканата

**Исходные данные:**

Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога для деканата.

База данных должна содержать данные о контингенте студентов (фамилия,

имя, отчество, год поступления, форма обучения (дневная/вечерняя/заочная), номер или название группы); об учебном плане (название специальности, дисциплина, семестр, количество отводимых на дисциплину часов, форма отчетности (экзамен/зачет)); о журнале успеваемости студентов (год/семестр, студент, дисциплина, оценка). Разработать пакет, состоящий из процедур и функций, позволяющий:

* для указанной формы обучения посчитать количество студентов этой формы обучения;
* для указанной дисциплины получить количество часов и формы отчетности по этой дисциплине; предоставить возможность добавления и изменения информации о студентах, об учебных планах, о журнале успеваемости

при этом предусмотреть курсоры, срабатывающие на некоторые пользовательские исключительные ситуации;

* предоставить возможность добавления и изменения информации о журнале успеваемости.

Необходимо предусмотреть возможность выдачи справки об успеваемости каждому студенту.

**Содержание пояснительной записки:**

Содержание;

Введение;

1. Анализ предметной области;
2. Обоснование модели данных;
3. Обоснование выбора СУБД;
4. Описание функций групп пользователей;
5. Описание функций управления данными;
6. Организация защиты БД;

Заключение;

Список использованных источников;

Приложение А. Руководство пользователя БД;

Приложение Б. Листинг кода.

**Предполагаемый объем пояснительной записки:** не менее 20 страниц

**Дата выдачи задания:** 13.09.2024

**Дата сдачи реферата:** 23.12.2024

**Дата проверки реферата:**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Полуянов В. Н. |
| Преподаватель |  | Татарникова Т. М. |

**АННОТАЦИЯ**

Содержание курсовой работы заключается в проектировании базы данных (БД) для деканата учебного заведения. Полезность такой БД обусловлена необходимостью автоматизации и оптимизации процессов хранения и обработки данных для качественного учета успеваемости студентов.

В работе обоснован выбор системы управления базами данных (СУБД), описаны функции пользователей, операции управления данными, методы защиты данных.

Созданная БД способна повысить эффективность работы деканата. Руководство пользователя и исходный код представлены в приложениях.

**SUMMARY**

The coursework consists in projecting a database (DB) for the dean's office of an academic establishment. The usefulness of such a database is due to the need to automate and optimize the processes of data storage and processing for the qualitative accounting of student performance.

In the work the choice of database management system (DBMS) is justified, user functions, data management operations, data protection methods are described.

The created database is able to increase the efficiency of the dean's office. The user manual and source code are presented in the appendices.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc185810300)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc185810301)

[2. ОБОСНОВАНИЕ МОДЕЛИ ДАННЫХ 9](#_Toc185810302)

[3. ОБОСНОВАНИЕ ВЫБОРА СУБД 11](#_Toc185810303)

[4. ОПИСАНИЕ ФУНКЦИЙ ГРУПП ПОЛЬЗОВАТЕЛЕЙ 12](#_Toc185810304)

[5. ОПИСАНИЕ ФУНКЦИЙ УПРАВЛЕНИЯ ДАННЫМИ 13](#_Toc185810305)

[6. ОРГАНИЗАЦИЯ ЗАЩИТЫ БД 14](#_Toc185810306)

[ЗАКЛЮЧЕНИЕ 15](#_Toc185810307)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc185810308)

[ПРИЛОЖЕНИЕ А РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ БД 17](#_Toc185810309)

[ПРИЛОЖЕНИЕ Б ЛИСТИНГ ПРОГРАММНОГО КОДА 18](#_Toc185810310)

ВВЕДЕНИЕ

В современных условиях объемы данных неизменно растут, и автоматизация деятельности образовательных учреждений является ключевым фактором эффективной работы. Одной из важнейших областей автоматизации являются данные, а именно их хранение и управление ими. Учебные заведения насчитывают огромное количество изучаемых дисциплин, студентов, выставляемых оценок. Все эти данные требуют тщательного и безопасного хранения, ведь потеря дел студентов – большая проблема, с которой не хочется столкнуться. Часто возникает необходимость составления различных однотипных документов и делать это вручную каждый раз – долго, тяжело и чревато ошибками.

Настоящая работа нацелена на проектирование базы данных, которую можно будет применять сотрудникам учебных заведений. Проект должен позволить деканату:

1. Организовать централизованное и упорядоченное хранение данных;
2. Автоматизировать процессы управления данными;
3. Обеспечить удобный доступ к необходимой информации.

Проектирование системы проводится поэтапно. Для достижения поставленной цели следует выполнить следующие задачи:

1. На основе объектов предметной области, спроектировать модель данных;
2. Выбрать СУБД, подходящую для решения поставленных задач;
3. Детально изучить пользователей системы, описать их функции;
4. Описать функции управления данными;
5. Организовать защиту БД: ввести необходимые ограничения целостности для объектов, выдвинуть рекомендации по физической защите (методы и периодичность резервного копирования);
6. Сформировать руководство пользователя системы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В рассматриваемой области следует выделить несколько ключевых объектов: студентов, учебный план и журнал успеваемости. Это объекты, которые должны быть отражены в базе данных. Такая база данных должна решать следующие задачи:

1. Хранить информацию о студентах, учебных планах и успеваемости;
2. Генерировать справки об успеваемости студентов;
3. Подсчитывать количество студентов для заданной формы обучения;
4. Возвращать информацию о заданной дисциплине: количество часов, форму отчетности;
5. Предоставлять функционал добавления, изменения и удаления информации о студентах, учебных планах, успеваемости;
6. Обеспечивать сохранность и безопасность данных, проверять вводимые пользователем данные.

База данных имеет большой потенциал масштабирования, но на данном этапе можно выделить три ключевые группы:

1. Администратор – управляет и настраивает базу данных;
2. Секретарь деканата – имеет доступ ко всем таблицам БД, управляет общей информацией;
3. Преподаватель – имеет доступ к данным студентов и успеваемости, управляет информацией об успеваемости.

На вход могут поступать данные из различных источников:

1. Таблицы-ведомости об успеваемости студентов;
2. Учебные планы;
3. Прочие источники, например, веб-сайт учебного заведения, в случае масштабирования БД.

Система должна предоставлять возможность генерировать справки об успеваемости студентов как выходной документ.

Задачи работы, выдвинутые во введении, и упомянутые задачи БД, с учетом данных и информационных объектов модели, можно достигнуть следующими способами:

1. Для хранения создать таблицы с ключевыми связями;
2. Для запросов на вывод информации подготовить наборы SQL-команд;
3. Для управления данными и безопасностью подготовить процедуры и триггеры контроля корректности изменений, назначить пользователей и права доступа;
4. Для генерации документов подготовить наборы функций, возвращающих нужную информацию.

# 2. ОБОСНОВАНИЕ МОДЕЛИ ДАННЫХ

Рассмотрим детально информационные объекты. Как было сказано ранее, среди них три ключевых: студент, дисциплина, оценка.

Объект студента *student* содержит в себе фамилию, имя, отчество, форму обучения, год поступления и номер группы (*last\_name, first\_name, middle\_name, study\_form, study\_from\_id\_\_form\_name, group\_id\_\_group\_enroll\_year, group\_id*, соответственно). Номер группы является общим для большого числа студентов (в среднем, 20-30 человек на группу) и может содержать в себе год формирования группы. Так, студенты одной группы обязательно будут находиться на одном курсе, следовательно, для избегания дубликатов, опечаток и для снижения объема данных, принято решение вынести группу и год начала обучения в отдельную таблицу (студент может поступить не на первый курс, но это избыточная для данной БД информация). Также, многочисленный ввод однотипных строк о форме обучения, место для частых ошибок, создана таблица с формами обучения для локализации возможных форм и оптимизации памяти.

Дисциплина *subject* содержит в себе информацию о специальности, названии, семестре, числе отводимых часов, форме отчетности (*specialty\_id\_\_specialty\_name, subject\_name, semester, hours, report\_type\_id\_\_report\_type\_name, соответственно*). Специальности стандартизированы министерством образования РФ и доступны в соответствующих таблицах, поэтому для них следует выделить отдельную таблицу, содержащую информацию о коде и названии специальности (при масштабировании можно добавить более точную систему классификации с учетом степени образования), *specialty\_id* введён из соображений оптимизации поиска и доступа. Формы отчетности вынесены в отдельную таблицу по тем же соображениям, что и формы обучения у студентов. При условиях повышенной конфиденциальности, следует добавить атрибут преподавателя для ограничения области видимости последних.

Таблица успеваемости студентов содержит дату выставления *mark\_date* и значение оценки *mark*, а также внешние ключи студента и дисциплины. Дата выставления оценки необходима для допущения пересдач, ID оценки является инкрементным ключом и, в случае пересдачи день-в-день, последняя оценка будет иметь большее значение первичного ключа.

Данные структурированы и имеют выраженные связи, поэтому оптимальным решением будет использование реляционной базы данных.

Опираясь на вышеописанные рассуждения, была построена модель данных, представленная на рисунке 1.

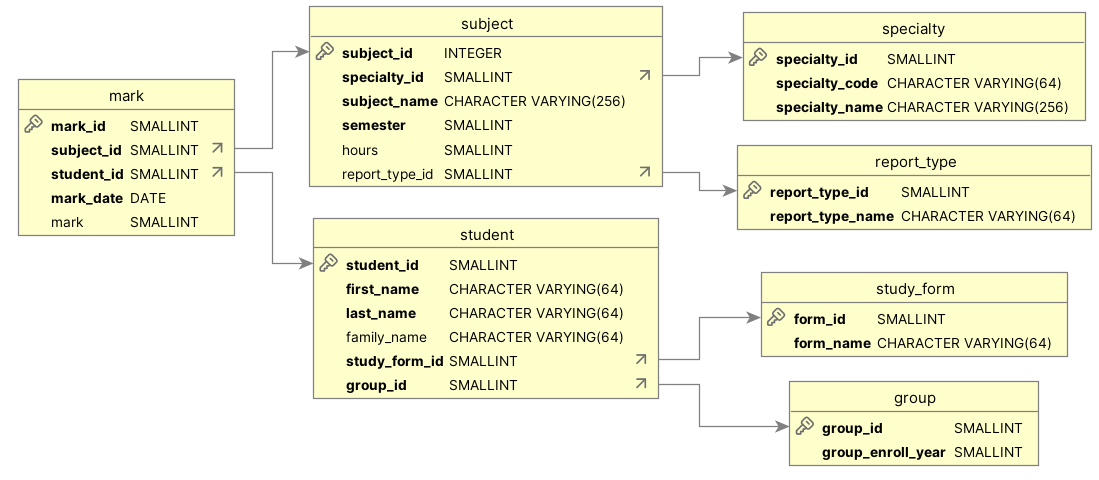


Рисунок 1. Модель данных

# 3. ОБОСНОВАНИЕ ВЫБОРА СУБД

Имея готовую модель данных, можно перейти к выбору системы управления этими данными. Для этого определим требования к СУБД.

Данные обладают сложной структурой и связями, во второй главе было принято решение использовать реляционную базу данных – СУБД должна соответствовать этим принципам.

Образовательные учреждения, в частности, высшие учебные заведения, имеют большое число студентов и преподавателей, документооборот таких учреждений крайне велик. СУБД должна справиться с нагрузкой, особенно в периоды сессий, когда обновления журнала успеваемости часты и объемны. Для этого потребуются механизмы оптимизированного выполнения запросов, методы индексации для быстрого поиска по большим объемам данных, параллельность обработки запросов.

Данные студентов крайне важны и не должны быть утеряны, поэтому среди требований к СУБД находятся отказоустойчивость и общая надежность.

На начальном этапе, расходы на хранение и управление данными должны быть минимальными. Возможно достигнуть полностью бесплатной реализации всех программных средств, за исключением работы администратора и расходов на серверные компоненты.

Выдвинутым требованиям подходят несколько СУБД, но было принято решение использовать средства открытой (open-source) PostgreSQL. Для написания дополнительного ПО выбран язык программирования Python за его простоту, лаконичность и обширную поддержку PSQL. Система подразумевает интеграцию в веб-сервис личного кабинета ВУЗа, поэтому для написания пользовательского интерфейса выбран фреймворк Python Django 5.

4. ОПИСАНИЕ ФУНКЦИЙ ГРУПП ПОЛЬЗОВАТЕЛЕЙ

Было выделено три группы пользователей, распределение прав доступа каждой категории к каждому объекту данных приведено в таблице 1. В таблице используются сокращения: S – select, просмотр данных; I – insert, добавление записей; U – update, редактирование записей; D – delete, удаление записей.

Таблица 1. Права доступа

|  |  |  |  |
| --- | --- | --- | --- |
|  | Администратор | Секретарь | Преподаватель |
| Студент | SIUD | SIUD | S |
| Группа | S |
| Форма обучения | S |
| Дисциплина | S |
| Специальность | S |
| Форма отчетности | S |
| Успеваемость | SU |
| Система | S | - |

В соответствие с построенной таблицей прав, была настроена панель администрирования Django. Администратор (superuser) имеет все права, остальные группы в правах ограничены (рис. 2-3). Среди прав преподавателя стоит прокомментировать право на изменение таблицы оценок: оно ограничено изменением только даты и значения оценки.

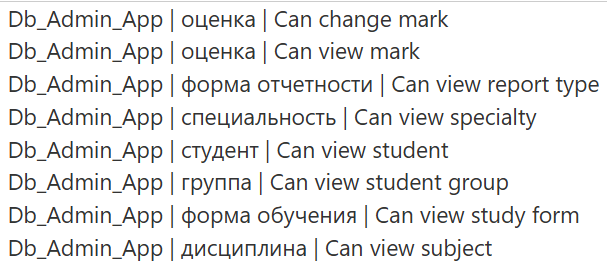


Рисунок 2. Права преподавателя

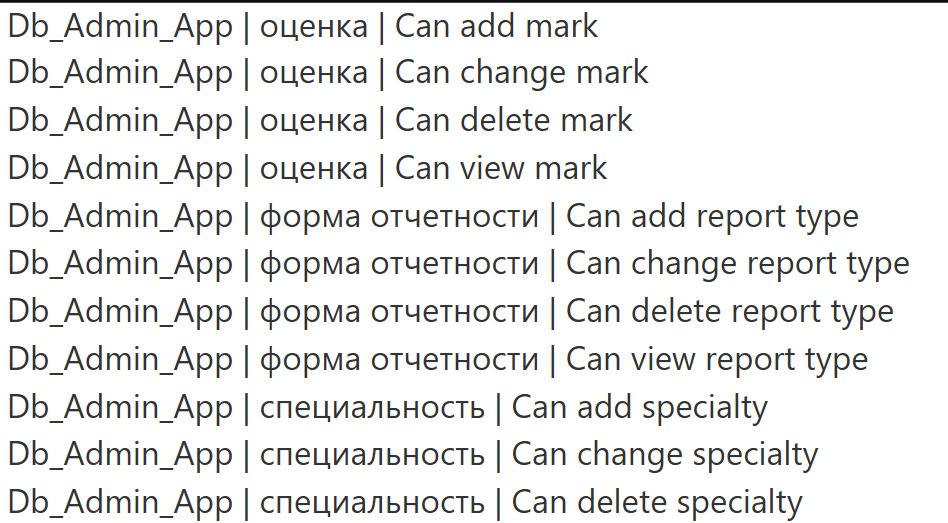


Рисунок 3. Фрагмент прав Секретаря

Опишем подробнее взаимодействие пользователей с системой.

4.1. Администратор

Имеет полный доступ ко всем объектам и отвечает за работоспособность базы данных:

* Настраивает БД;
* Управляет пользователями: создает, изменяет и удаляет учетные записи, назначает права доступа;
* Отслеживает состояние БД: оптимизирует запросы, резервирует данные, восстанавливает копии при необходимости.

4.2. Секретарь деканата

Контролирует основное содержимое базы данных, за исключением системных параметров:

* Заполняет основные таблицы;
* В соответствии с планом, создает пустые записи в журнале успеваемости, включая пересдачи;
* Получает сводные данные, генерирует справки для учащихся.

4.3. Преподаватель

Может просматривать основную информацию без изменений данных. Заполняет подготовленные записи оценок студентов в журнале.

# 5. ОПИСАНИЕ ФУНКЦИЙ УПРАВЛЕНИЯ ДАННЫМИ

Для реализации функционала, выполняющего задачи, выдвинутые к базе данных, были подготовлены шаблоны SQL-запросов и веб-приложение.

5.1. Получение количества студентов по форме обучения

Чтобы получить количество обучающихся по заданной пользователем форме обучения, необходимо предварительно сгруппировать записи о студентах по форме обучения, применить агрегирующую функцию COUNT и оставить счетчик по нужной форме обучения.

В форме SQL запрос для очной формы обучения будет выглядеть следующим образом (рис. 4).

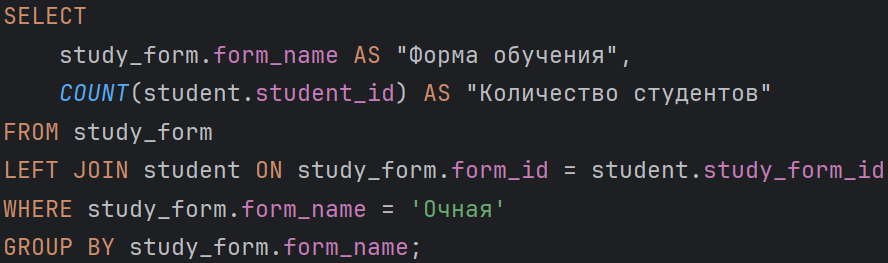


Рисунок 4. SQL-запрос получения количества часов по форме обучения

Страница веб-приложения, реализующего данный функционал, может выглядеть так, как на рисунке 5.

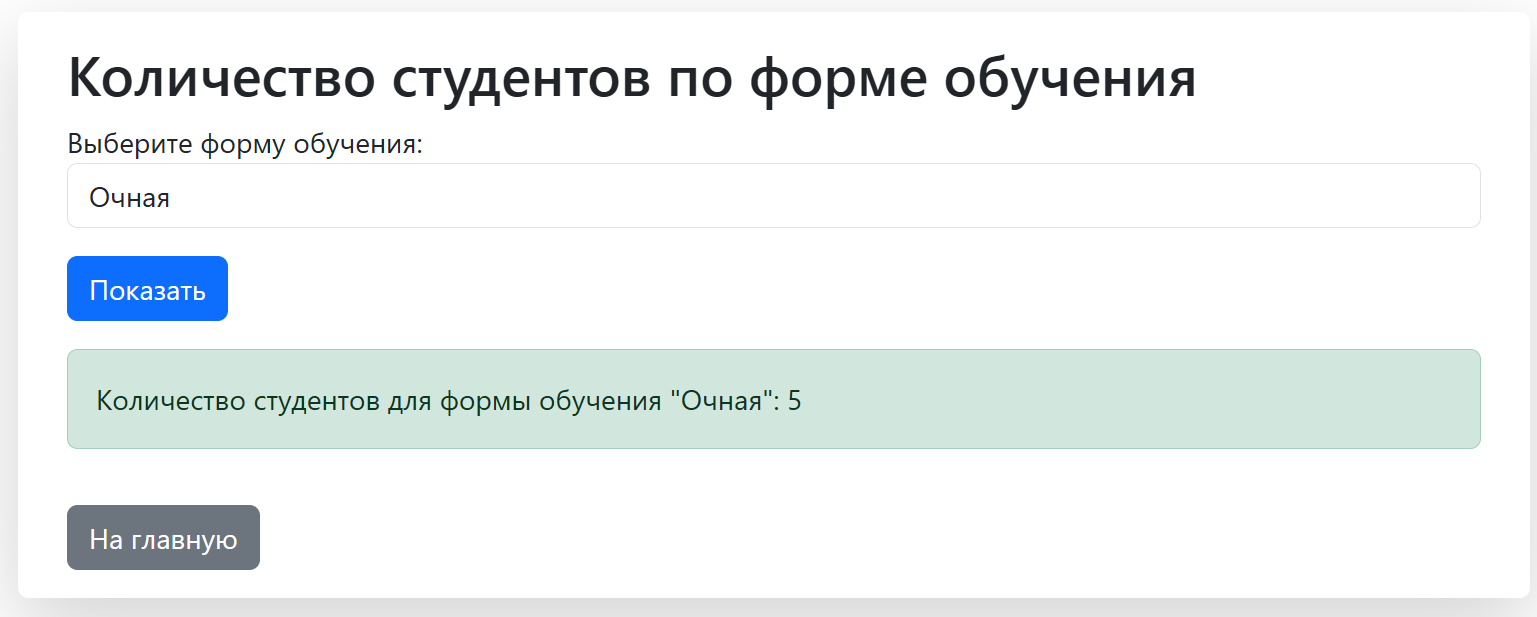


Рисунок 5. Реализация формы получения количества студентов

5.2. Получение количества часов и формы отчетности по дисциплине.

Получить отчет по дисциплине можно, обратившись к соответствующей таблице (рис. 6).

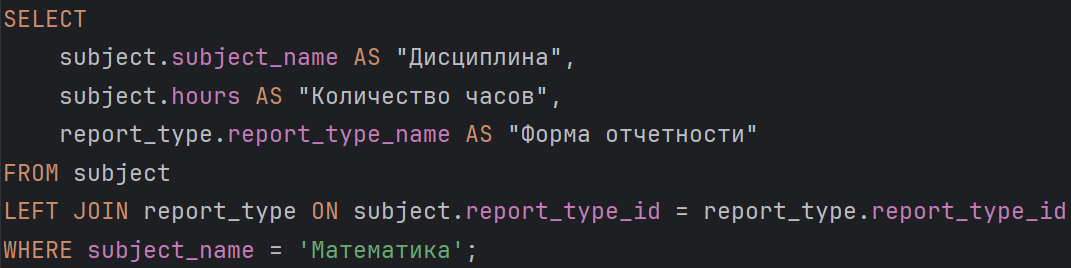


Рисунок 6. SQL-запрос получения отчета по дисциплине

Интерфейс формы представлен на рисунке 7.

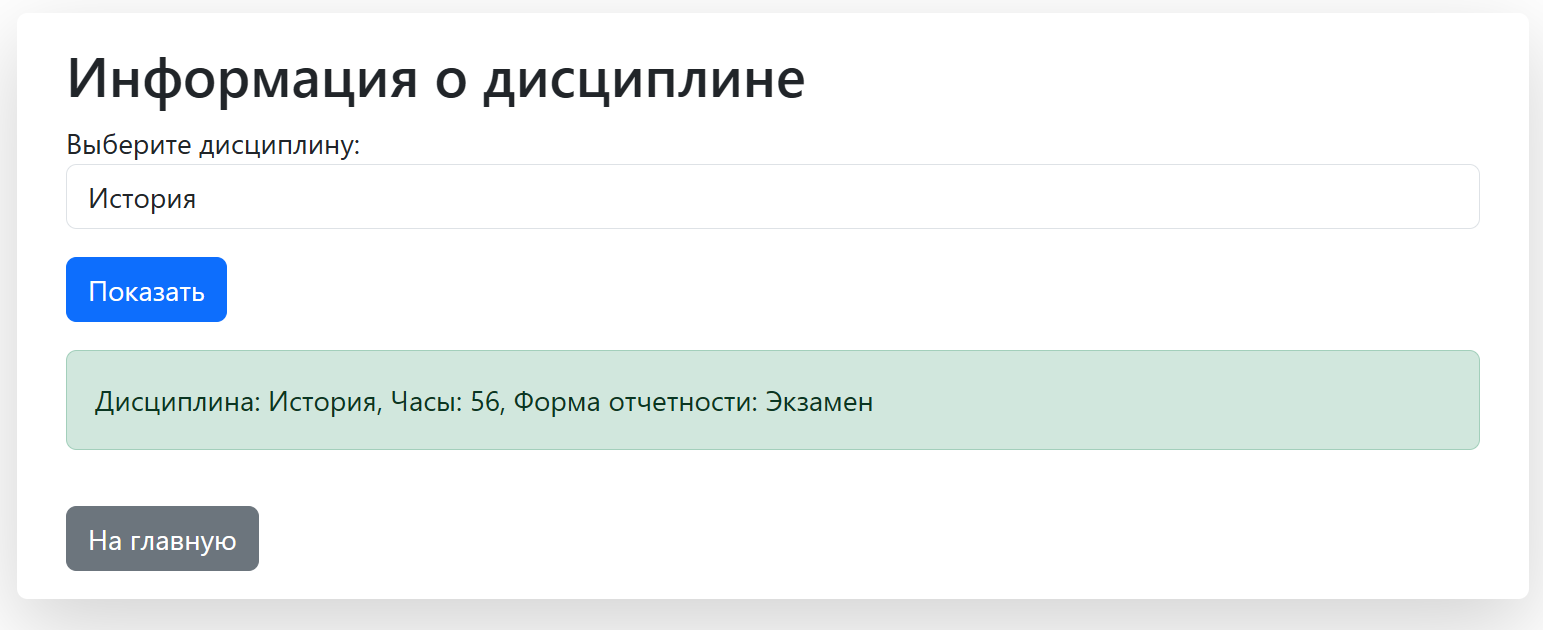


Рисунок 7. Интерфейс формы отчета по дисциплине

5.3. Получение справки об успеваемости студента.

Справка об успеваемости является сводной таблицей всех оценок, полученных студентом. Для получения такой таблицы, необходимо отфильтровать записи журнала успеваемости по id студента (метод должен предварительно получить id студента по его данным). Таблицу успеваемости студента с id=1 можно получить, выполнив SQL-запрос, представленный на рисунке 8.

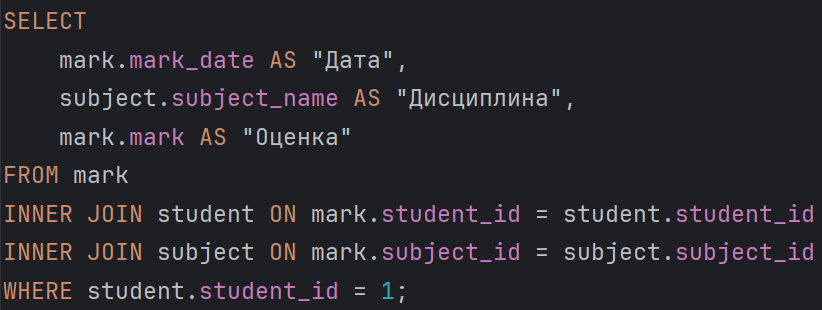


Рисунок 8. SQL-запрос получения справки об успеваемости

Настоящий интерфейс (рис. 9) позволяет с легкостью автоматизировать процесс создания готовых к печати .docx документов – справок об успеваемости. Для реализации такого механизма следует применить библиотеку python-docx.

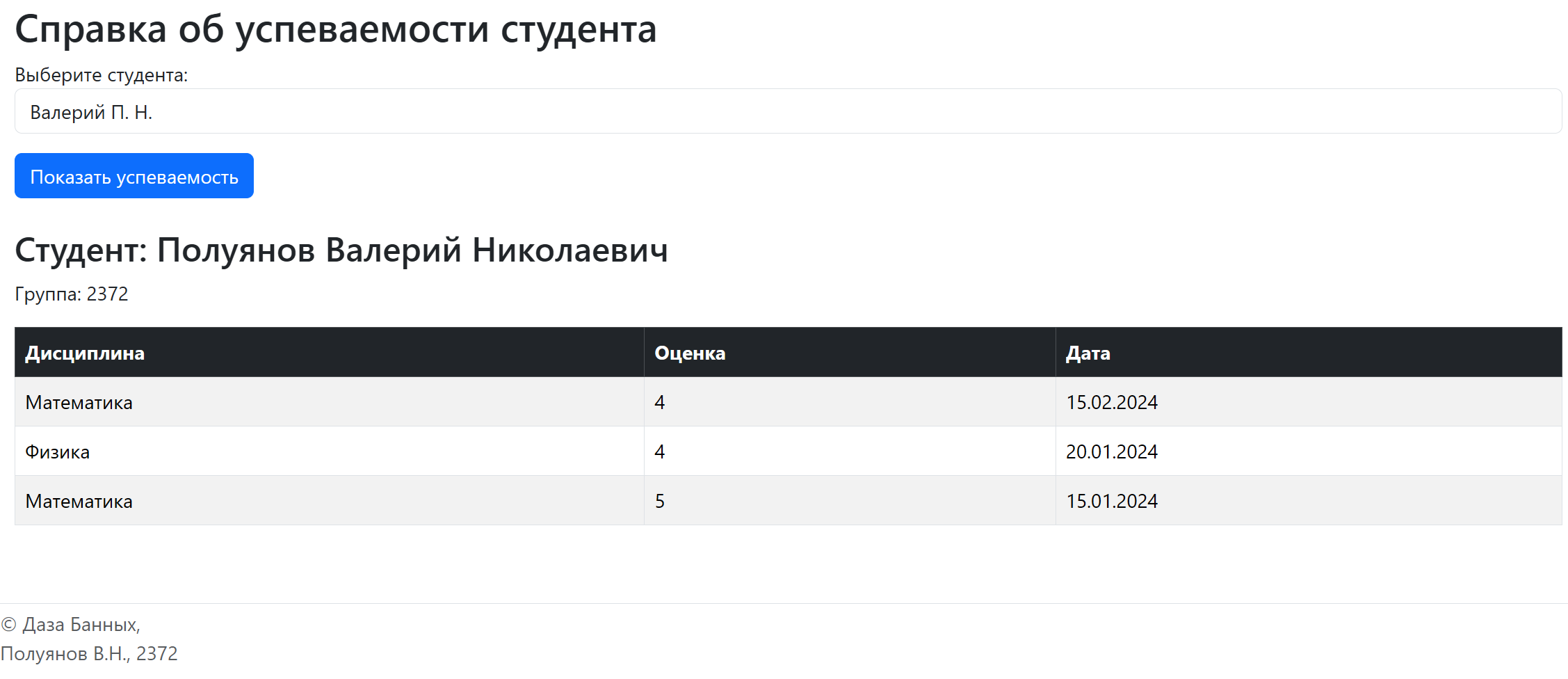


Рисунок 9. Интерфейс получения справки об успеваемости

5.4. Просмотр и изменение данных

Просмотреть необходимые данные можно с помощью команды SELECT, изменить – с помощью UPDATE. Для качественного, удобного и безопасного управления данными, был реализован интерфейс панели администрирования Django с поддержкой групп пользователей, описанных в предыдущей главе.

Попасть в интерфейс просмотра и управления данными можно по кнопке «База данных» на главной странице информационной системы (рис. 10).

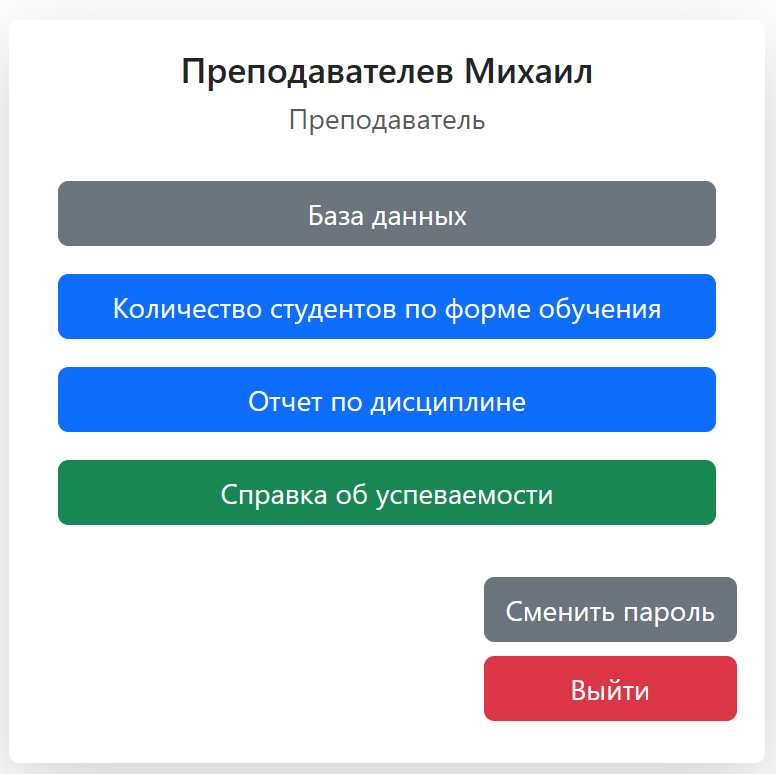


Рисунок 10. Главная страница

После нажатия на кнопку, пользователь попадет в настроенную систему администрирования, где, в зависимости от своих прав, он сможет просматривать и редактировать таблицы (рис. 11).

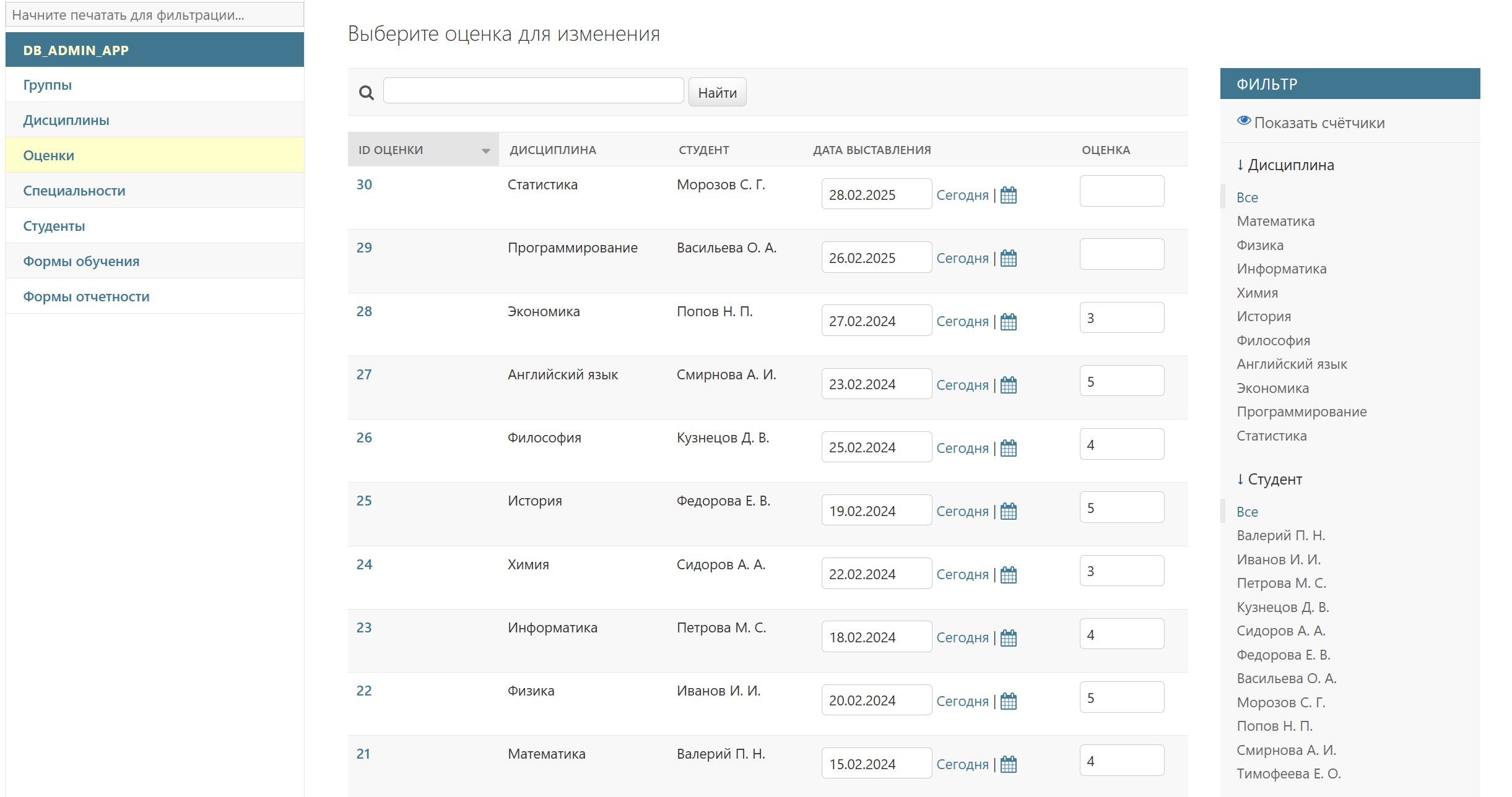


Рисунок 11. Интерфейс панели администрирования

# 6. ОРГАНИЗАЦИЯ ЗАЩИТЫ БД

Функциональность готового приложения должна быть проверена с помощью тестирования следующих компонентов системы:

Защита настоящей базы данных содержит в себе несколько этапов:

6.1. Нормализация БД

Нормализация базы данных направлена на устранение дубликатов, снижение требований к памяти и минимизации рисков аномалий при изменениях данных. В рамках данной работы была проведена нормализация до третьей нормальной формы. Данные разделены на логические таблицы образом, исключающим дублирование информации, процесс разбиения рассмотрен во второй главе.

6.2. Распределение прав и создание групп пользователей

Для обеспечения безопасности и конфиденциальности данных, доступ к ним разграничен на основе трех пользовательских групп: администраторы, секретари, преподаватели (см. главу 4) с помощью средств django.contrib.auth в GUI и с помощью функции GRANT на уровне базе данных.

6.3. Ограничения целостности и триггеры проверки данных

Для предотвращения ошибок (опечаток) при работе с данными, внедрены механизмы обеспечения целостности:

1. Обязательность заполнения ключевых полей с использованием NOT NULL;
2. Проверка корректности вводимых значений с помощью ограничений CHECK (например, оценка должна быть в диапазоне от 1 до 100, несмотря на более широкий диапазон smallint);
3. Используются триггеры, обновляющие даты записей (для журнала успеваемости).

6.4. Защита от SQL-инъекций

Применение Django ORM является не только удобным, но и проверенным временем и отвечающим стандартам способом работы с данными. ORM автоматически экранирует входные данные и формирует безопасные SQL-запросы, предварительно проверяя заполняемые формы на наличие CSRF-токена.

6.5. Резервное копирование

Ни один метод не может защитить от непредвиденного искажения и утери данных (например, из-за неисправности серверного оборудования), поэтому регулярное резервное копирование – неотъемлемая часть организации безопасности.

Для данной БД предложен следующий механизм резервного копирования:

1. Ежедневное инкрементное копирование;
2. Еженедельное полное копирование;
3. Выделение отдельного хранилища для копий;
4. Удаление старых копий раз в неделю при обновлении полной копии;

Такой подход к обеспечению безопасности данных является оптимальным и позволит минимизировать риски искажения и утери.

# ЗАКЛЮЧЕНИЕ

Результатом выполнения курсовой работы стало достижение всех поставленных целей и задач. На основе детального анализа предметной области, информационных объектов, групп пользователей и их функций была выбрана система управления реляционными базами данных PostgreSQL, прикладное программное обеспечение Python в сочетании с Django 5.

Спроектированная база данных и прототип пользовательского интерфейса могут позволить эффективно и безопасно организовать хранение и управления большими объемами данных, проходящих через деканат. При разработке были учтены возможности по масштабированию и интеграции информационной системы, что позволяет адаптировать её к требованиям конкретного учебного заведения.

Таким образом, были получены практические навыки по проектированию и разработке баз данных, обеспечению их безопасности и созданию интерфейса для взаимодействия с данными.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Antonio M., «Django 3 in examples». Packt Publishing, 2020, 568 p.;
2. Алан Болье, «Изучаем SQL. Вводный курс для разработчиков и администраторов БД», М., 2007, 310 с.;
3. Новиков Б., Горшкова Е., Графеева Н., «Основы технологий баз данных: учебное пособие», ред. Рогова. Е.В., 2 издание, ДМК-Пресс, 2020 г., 583 с.;
4. PostgreSQL: Документация // Компания Postgres Professional. URL: <https://postgrespro.ru/docs/postgresql> (дата обращения: 23.12.2024);
5. Django documentation // Django project. URL: <https://docs.djangoproject.com/en/5.1/> (дата обращения: 23.12.2024).

# ПРИЛОЖЕНИЕ А РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ БД

1. Руководство по развертыванию приложения

Для развертывания приложения необходимо иметь установленный Python версии не ниже 3.8 с пакетным менеджером pip и СУБД PostgreSQL версии не ниже 14.

**Настройка виртуального окружения.**

Создайте виртуальное окружение, выполнив команду:

python3 -m venv venv

После выполнения команды, запустите виртуальное окружение:

venv/Scripts/activate

Для Windows или

source venv/bin/activate

Для Linux/MaxOS

Виртуальное окружение запущено, установите нужные зависимости:

pip install -r requirements.txt

Эта команда установит Django, Bootstrap и другие зависимости, необходимые для развертывания приложения. Виртуальное окружение подготовлено.

**Развертывание базы данных**.

Создайте базу данных, выполнив SQL-очереди:

database/create\_db\_and\_tables.sql

Для создания БД и таблиц в ней,

database/example\_data/fill\_db.sql

Для заполнения таблиц данными.

**Запуск приложения.**

Проверьте содержимое файла конфигураций db\_admin/example\_config.ini. Он должен содержать правильные параметры подключения к БД.

Примените миграции для создания базы данных в Django:

python3 manage.py makemigrations

Подготовьте аккаунт администратора:

python3 manage.py createsuperuser

Запустите приложение:

python3 manage.py runserver

Сервер будет доступен по адресу http://127.0.0.1:8000/.

1. Руководство по взаимодействию с графическим интерфейсом.

При первом подключении к веб-сайту, будет предложено пройти авторизацию (регистрацию новых пользователей проводит администратор базы данных), нажмите «войти» и укажите свои данные (рис. 12-13).

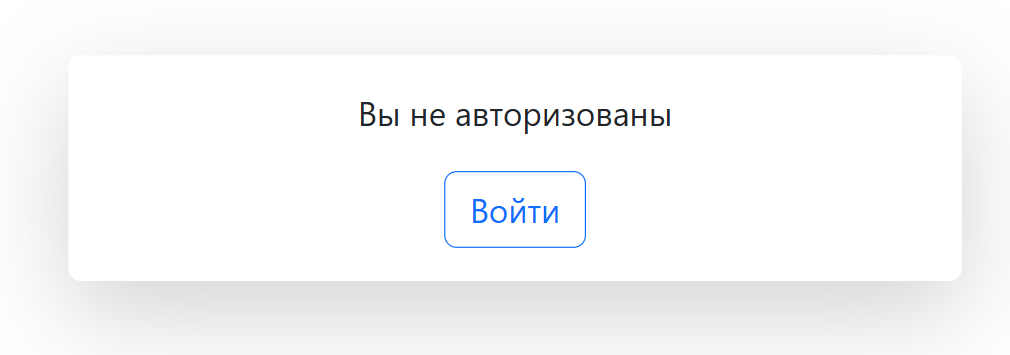


Рисунок 12. Кнопка "войти"

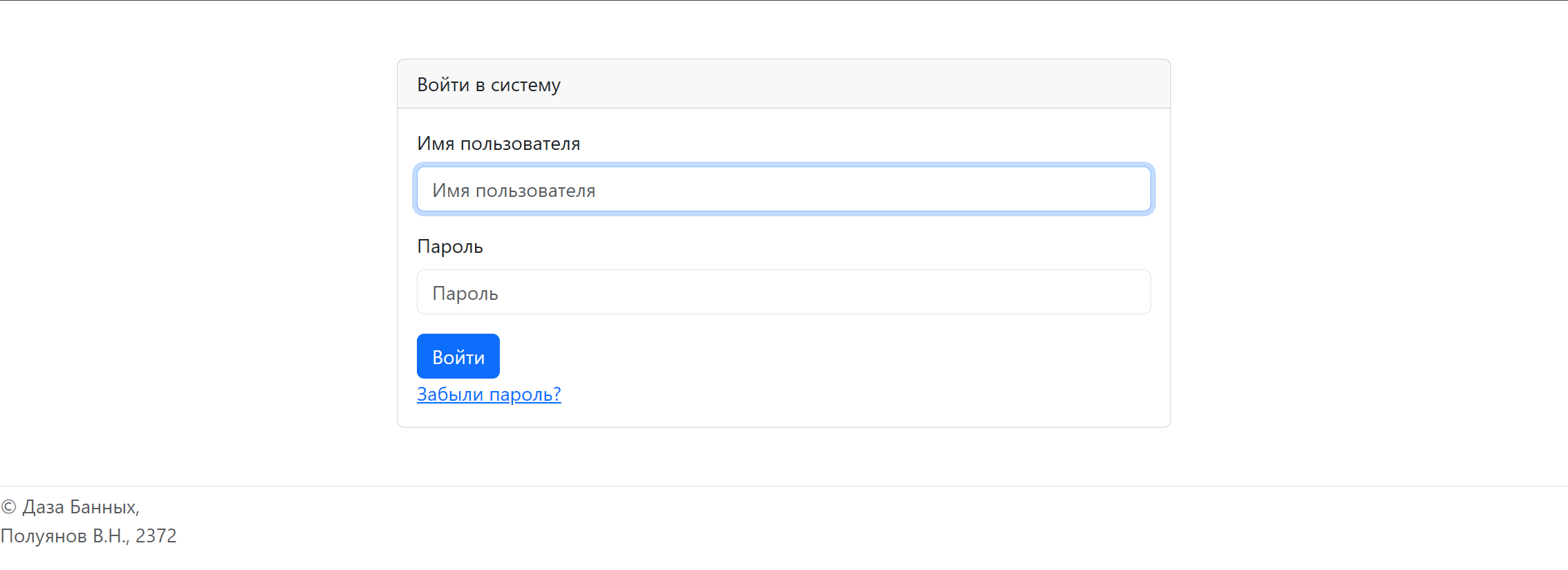


Рисунок 13. Страница авторизации

После успешной авторизации вы увидите основное меню. Оно содержит функции авторизации (выход и смена пароля) и кнопки перехода к страницам просмотра и управления данными (рис. 14):

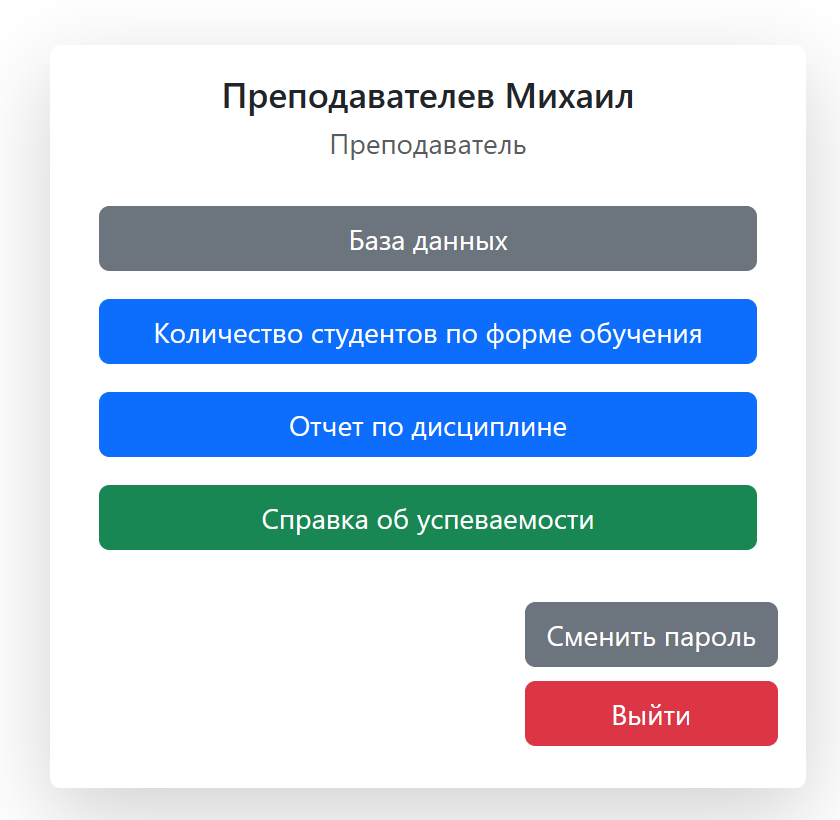


Рисунок 14. Главное меню

1. Кнопка «Выйти»: при нажатии на неё, Вы потеряете доступ к системе и вернётесь на страницу авторизации (рис. 12);
2. Кнопка «Сменить пароль»: при нажатии, Вы перейдете на страницу смены пароля (рис. 15). После успешного изменения пароля, Вы вернетесь на главную страницу (рис. 14);

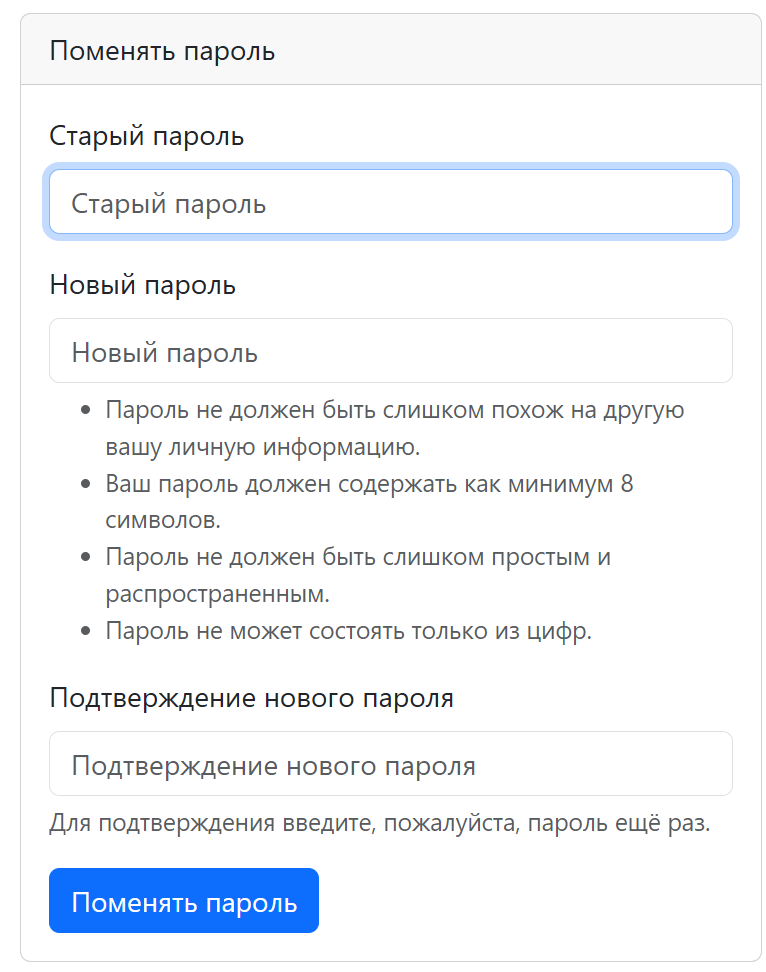


Рисунок 15. Страница смены пароля

1. «Справка об успеваемости»: при нажатии, Вы перейдете на страницу получения справки. Вам будет предложено выбрать интересующего Вас студента (рис. 16). После выбора, при нажатии кнопки «Показать успеваемость», Вы увидите таблицу успеваемости выбранного студента (рис. 17);

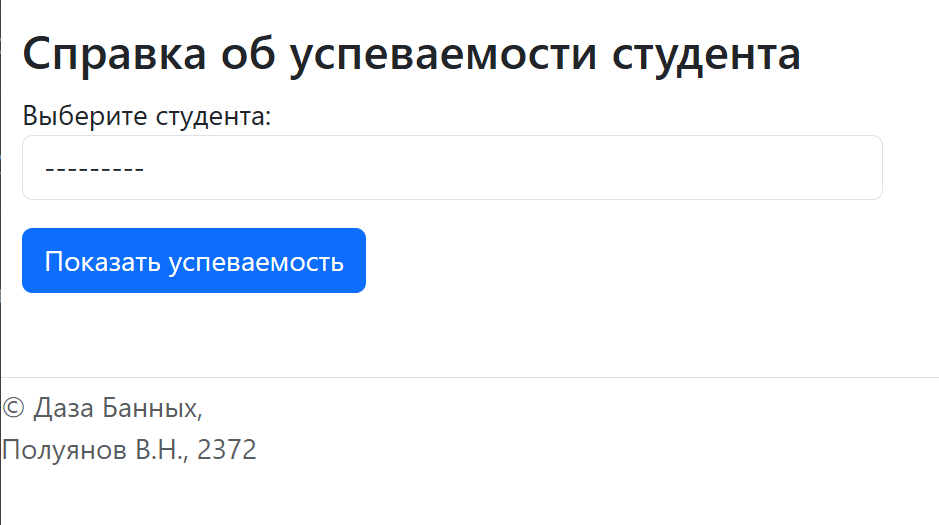


Рисунок 16. Справка об успеваемости студента - до

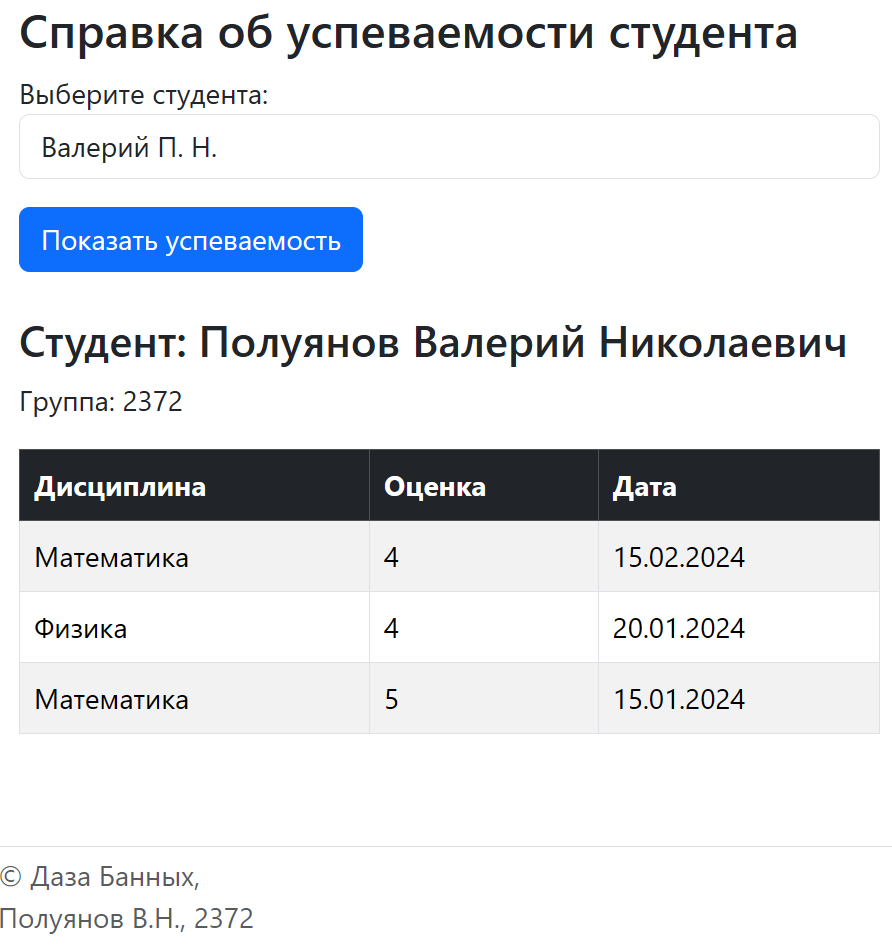


Рисунок 17. Справка об успеваемости студента - после

1. Кнопка «Отчет по дисциплине»: при нажатии, Вы перейдете на страницу получения информации о дисциплине. Вам будет предложено выбрать интересующую дисциплину (рис. 18). После выбора, по нажатии на кнопку «Показать», Вы увидите информацию о выбранной дисциплине (рис. 19);

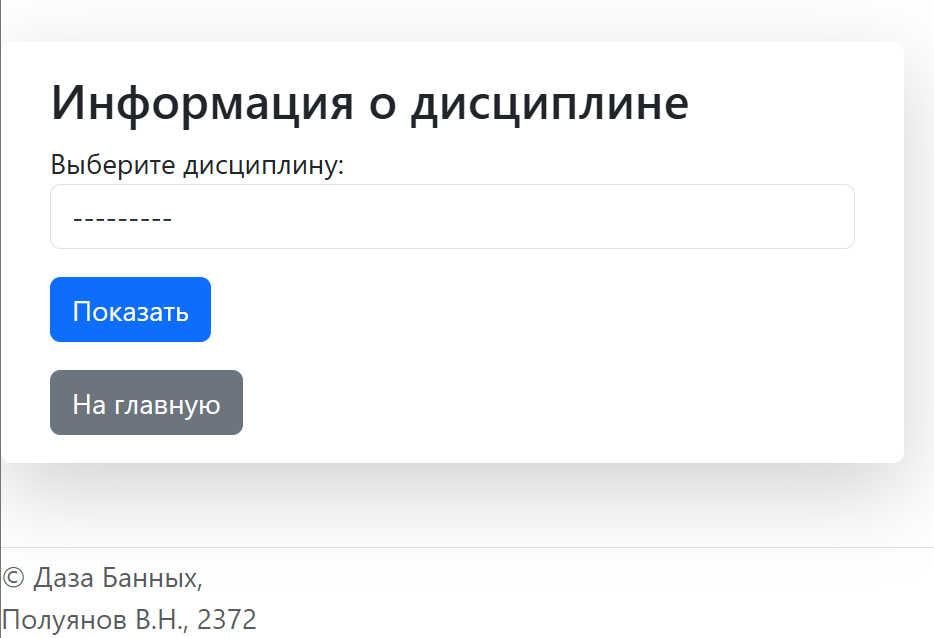


Рисунок 18. Информация о дисциплине - до выбора

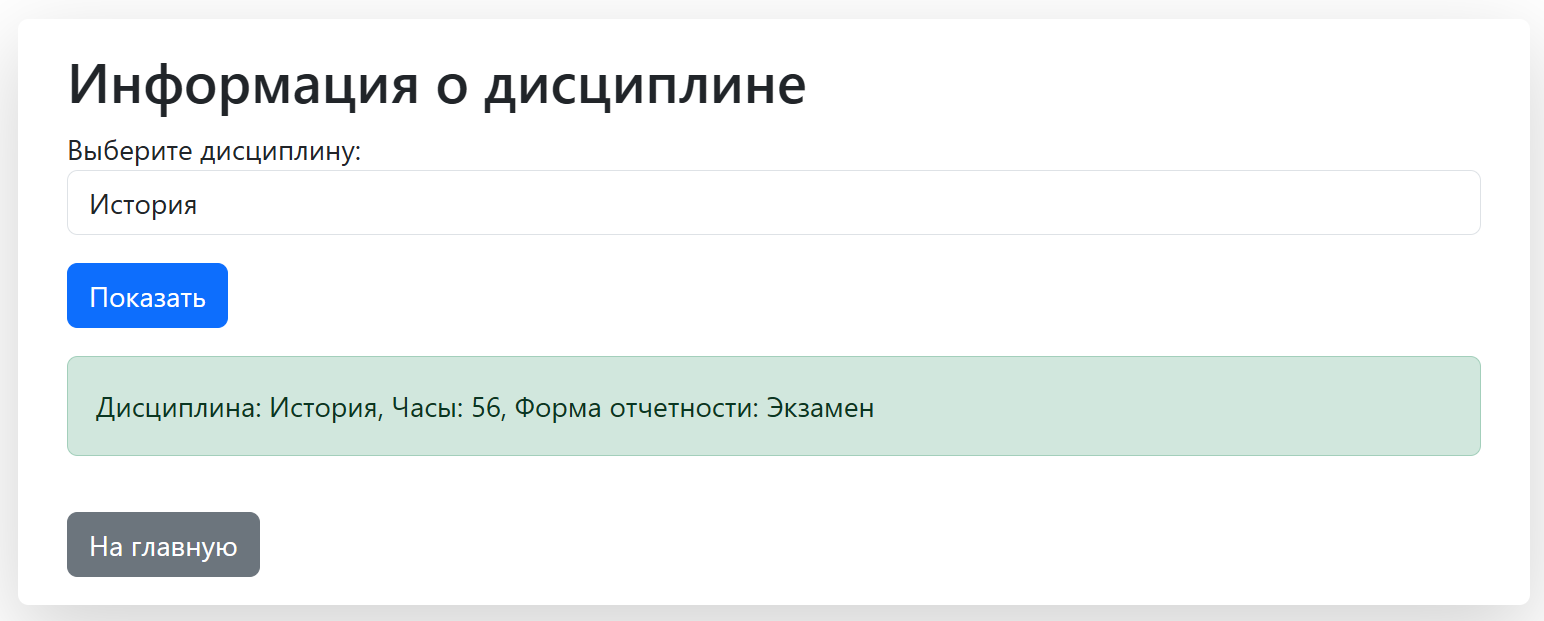


Рисунок 19. Информация о дисциплине - после выбора

1. Кнопка «Количество студентов по форме обучения»: при нажатии, Вы перейдете на страницу получения информации о количестве студентов, обучающихся на задаваемой форме обучения. Вам будет предложено выбрать интересующую Вас форму обучения (рис. 20). После выбора и нажатия на кнопку «Показать», Вы увидите информацию по выбранной форме обучения (рис. 21);

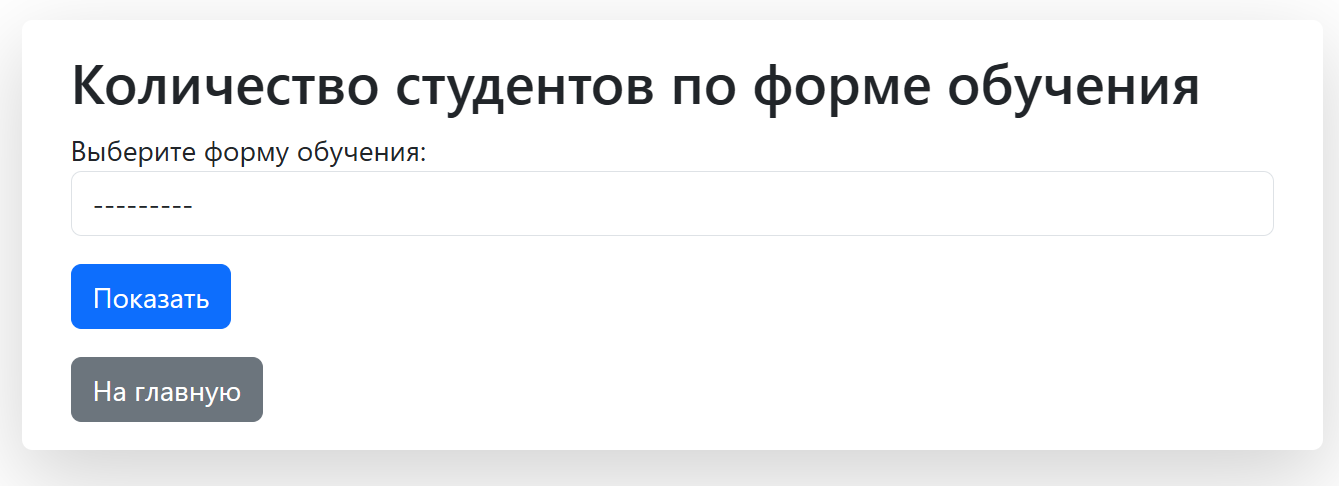


Рисунок 20. Количество студентов - до выбора формы обучения

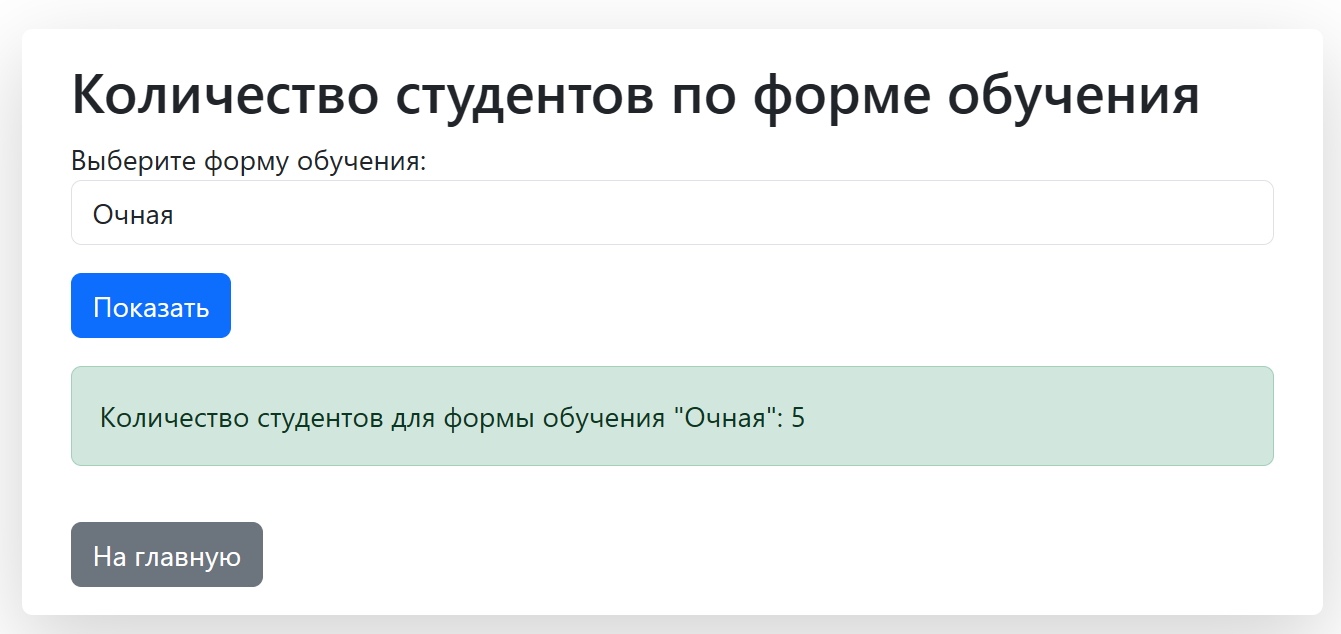


Рисунок 21. Количество студентов - после выбора формы обучения

1. Кнопка «База данных»: при нажатии на эту кнопку, будет открыта панель администрирования Django ORM. Её внешний вид и доступные функции различаются в зависимости от Ваших прав. На рисунках 22-24 представлены возможные варианты внешнего вида главной страницы в соответствие с ролью.



Рисунок 22. Панель администрирования у Преподавателя

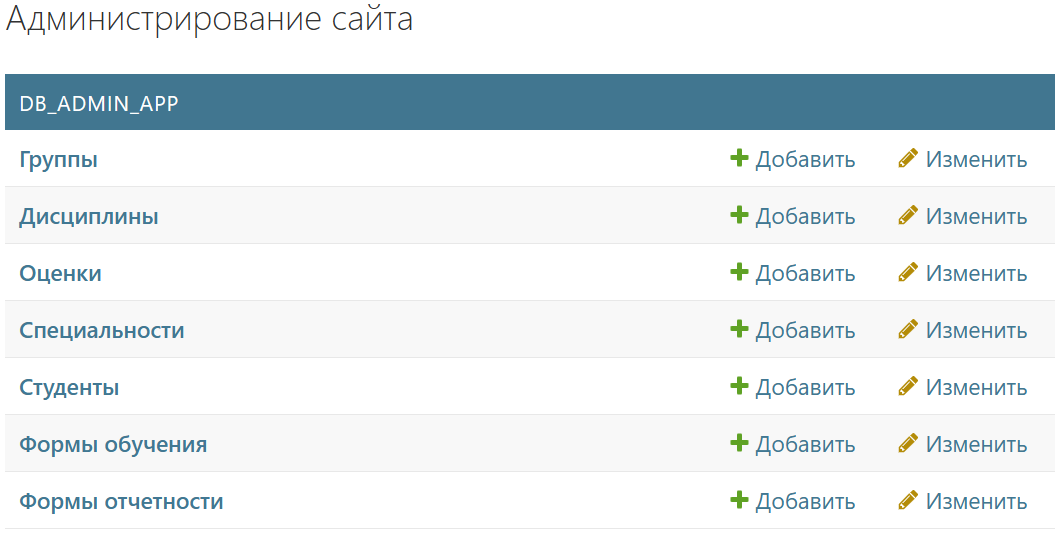


Рисунок 23. Панель администратора у Секретаря



Рисунок 24. Панель администратора у Администратора

Каждая таблица позволяет читать, добавлять, редактировать и удалять записи (если Вы имеете на это право), пример таблицы Студенты для пользователя группы Администратор представлен на рисунке 25.

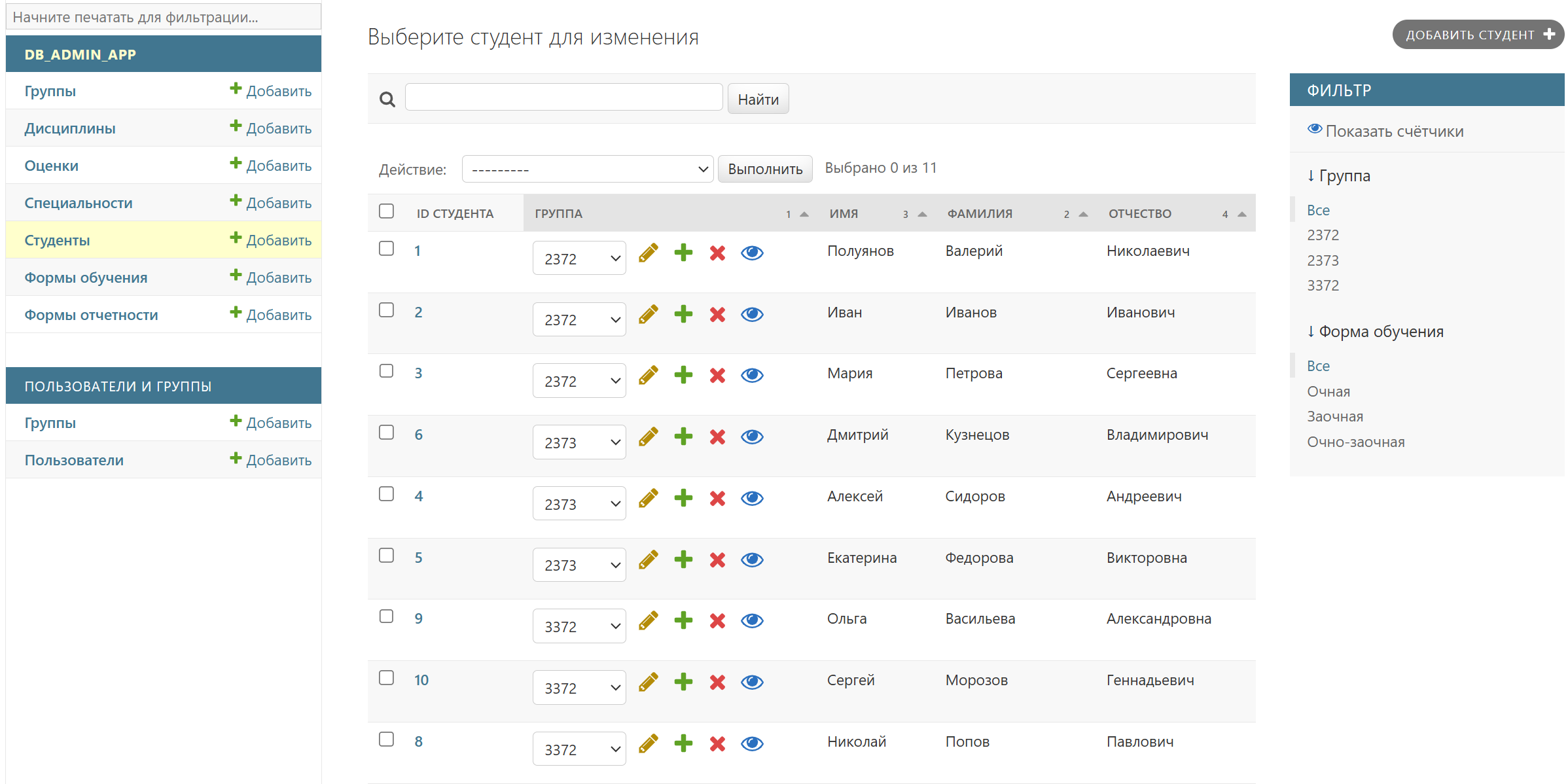


Рисунок 25. Таблица Студенты у Администратора

Та же таблица у Преподавателя, не будет позволять редактировать, добавлять и удалять записи (рис. 26).

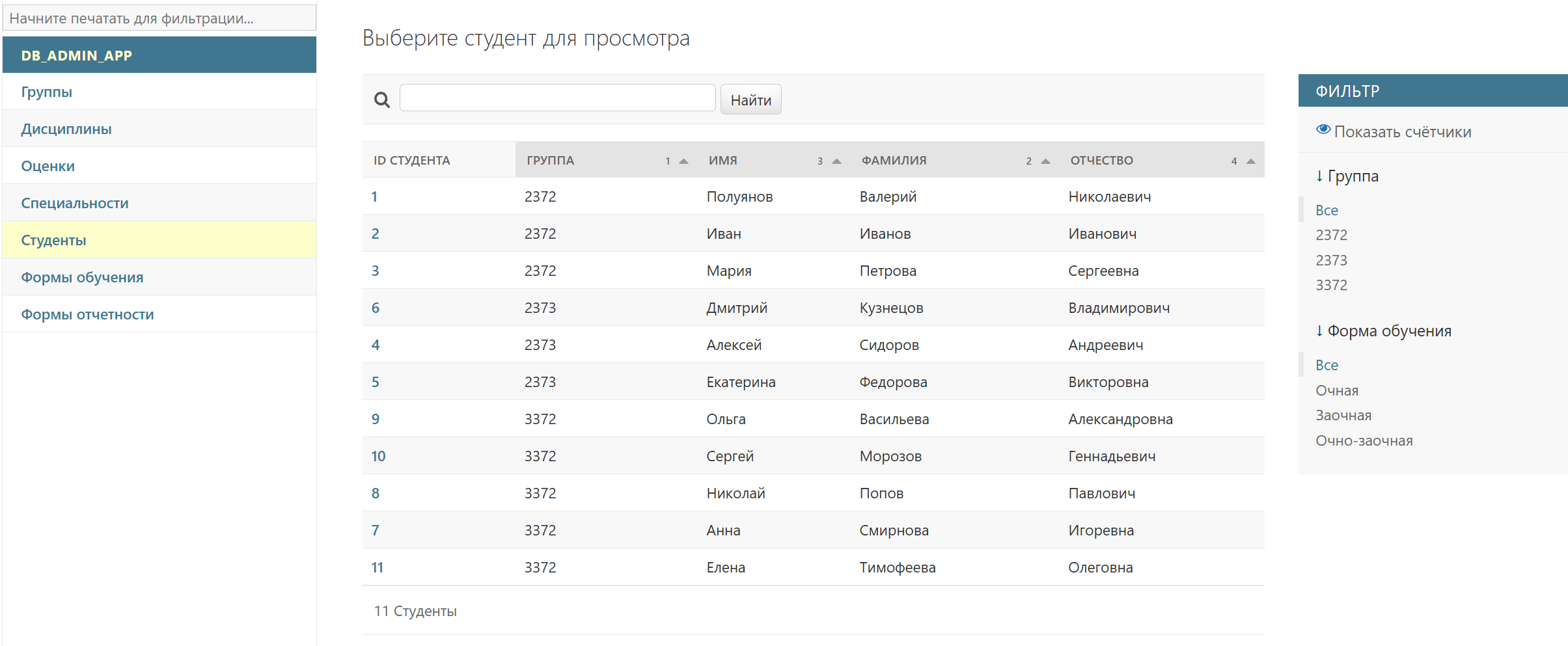


Рисунок 26. Таблица Студенты у Преподавателя

Как было указано в четвертой главе настоящей работы, Преподаватель имеет право изменять только два поля: дату выставления и оценку в таблице Оценка (рис. 27).

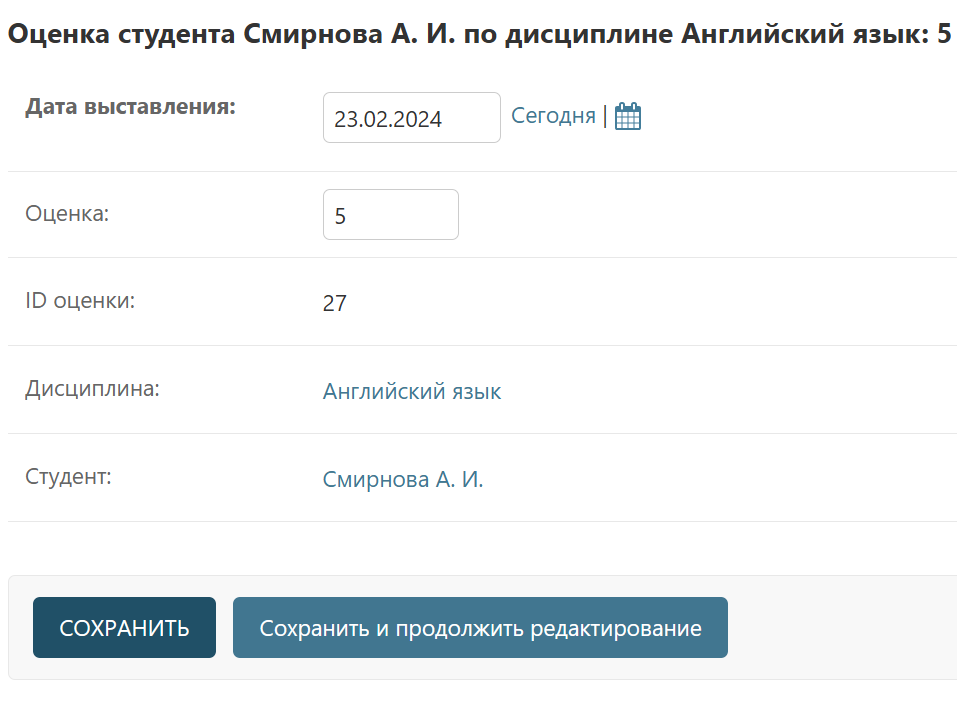


Рисунок 27. Редактирование оценки у Преподавтеля

При этом, Администраторы и Секретари могут изменять все поля, добавлять и удалять записи из этой таблицы (рис. 28).

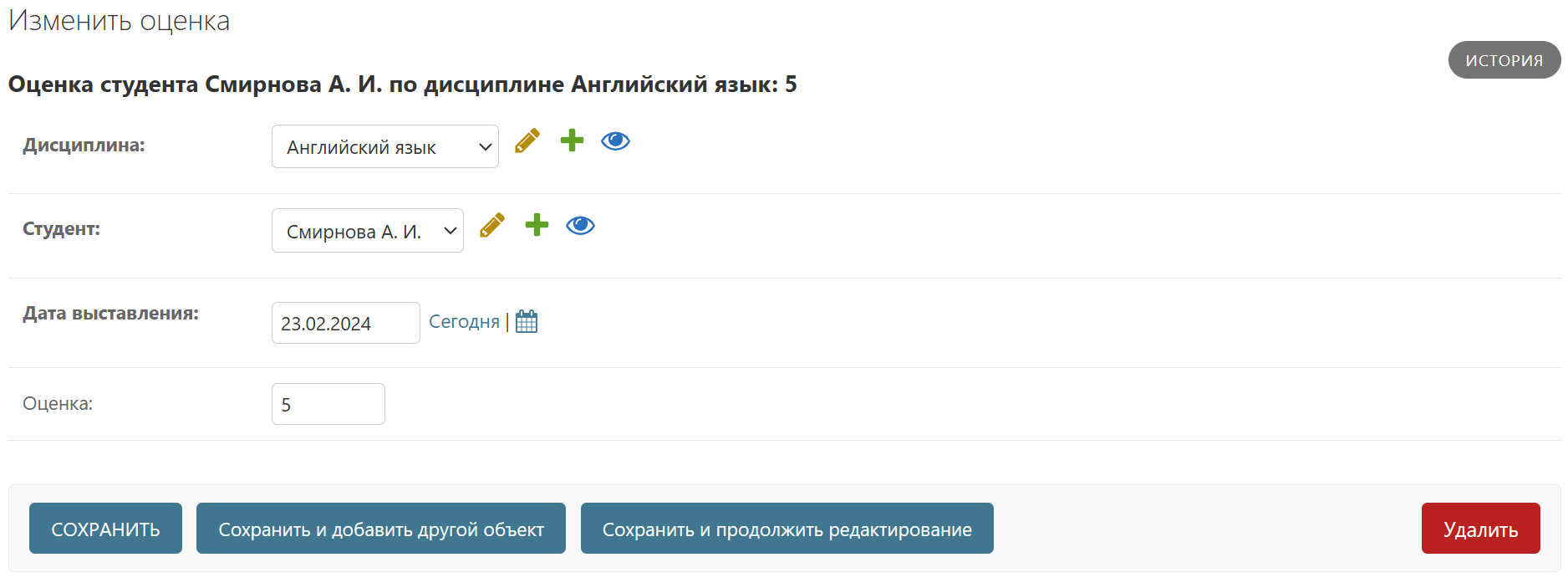


Рисунок 28. Редактирование оценки у Секретаря и Администратора

# ПРИЛОЖЕНИЕ Б ЛИСТИНГ ПРОГРАММНОГО КОДА

SQL-запросы:

*-- Create DB and tables  
  
--  
-- Database: etu-deans-db  
--*CREATE DATABASE "etu-deans-db"  
 WITH  
 OWNER = postgres  
 ENCODING = 'UTF8'  
 LC\_COLLATE = 'Russian\_Russia.1251'  
 LC\_CTYPE = 'Russian\_Russia.1251'  
 LOCALE\_PROVIDER = 'libc'  
 TABLESPACE = pg\_default  
 CONNECTION LIMIT = -1  
 IS\_TEMPLATE = False;  
  
  
  
*--  
-- Table: public.group  
--*CREATE TABLE IF NOT EXISTS public.student\_group  
(  
 group\_id smallint NOT NULL,  
 group\_enroll\_year smallint NOT NULL,  
 CONSTRAINT group\_pkey PRIMARY KEY (group\_id)  
)  
  
TABLESPACE pg\_default;  
  
ALTER TABLE IF EXISTS public."group"  
 OWNER to postgres;  
  
  
  
*--  
-- Table: public.mark  
--*CREATE TABLE IF NOT EXISTS public.mark  
(  
 mark\_id integer NOT NULL DEFAULT nextval('mark\_mark\_id\_seq'::regclass),  
 subject\_id smallint NOT NULL,  
 student\_id smallint NOT NULL,  
 mark\_date date NOT NULL,  
 mark smallint,  
 CONSTRAINT mark\_pkey PRIMARY KEY (mark\_id),  
 CONSTRAINT mark\_\_student\_fkey FOREIGN KEY (student\_id)  
 REFERENCES public.student (student\_id) MATCH FULL  
 ON UPDATE CASCADE  
 ON DELETE CASCADE,  
 CONSTRAINT mark\_\_subject\_fkey FOREIGN KEY (subject\_id)  
 REFERENCES public.subject (subject\_id) MATCH FULL  
 ON UPDATE CASCADE  
 ON DELETE CASCADE  
)  
  
TABLESPACE pg\_default;  
  
ALTER TABLE IF EXISTS public.mark  
 OWNER to postgres;  
  
  
  
*--  
-- Table: public.report\_type  
--*CREATE TABLE IF NOT EXISTS public.report\_type  
(  
 report\_type\_id smallint NOT NULL,  
 report\_type\_name character varying(64) COLLATE pg\_catalog."default" NOT NULL,  
 CONSTRAINT report\_type\_pkey PRIMARY KEY (report\_type\_id)  
)  
  
TABLESPACE pg\_default;  
  
ALTER TABLE IF EXISTS public.report\_type  
 OWNER to postgres;  
  
  
  
*--  
-- Table: public.specialty  
--*CREATE TABLE IF NOT EXISTS public.specialty  
(  
 specialty\_id smallint NOT NULL DEFAULT nextval('specialty\_specialty\_id\_seq'::regclass),  
 specialty\_code character varying(16) COLLATE pg\_catalog."default" NOT NULL,  
 specialty\_name character varying(256) COLLATE pg\_catalog."default" NOT NULL,  
 CONSTRAINT specialty\_pkey PRIMARY KEY (specialty\_id)  
)  
  
TABLESPACE pg\_default;  
  
ALTER TABLE IF EXISTS public.specialty  
 OWNER to postgres;  
  
  
  
*--  
-- Table: public.student  
--*CREATE TABLE IF NOT EXISTS public.student  
(  
 student\_id integer NOT NULL DEFAULT nextval('student\_student\_id\_seq'::regclass),  
 first\_name character varying(64) COLLATE pg\_catalog."default" NOT NULL,  
 last\_name character varying(64) COLLATE pg\_catalog."default" NOT NULL,  
 family\_name character varying(64) COLLATE pg\_catalog."default",  
 study\_form\_id smallint NOT NULL,  
 group\_id smallint NOT NULL,  
 CONSTRAINT student\_pkey PRIMARY KEY (student\_id),  
 CONSTRAINT student\_\_group\_fkey FOREIGN KEY (group\_id)  
 REFERENCES public.student\_group (group\_id) MATCH FULL  
 ON UPDATE CASCADE  
 ON DELETE SET NULL,  
 CONSTRAINT student\_\_study\_form\_fkey FOREIGN KEY (study\_form\_id)  
 REFERENCES public.study\_form (form\_id) MATCH FULL  
 ON UPDATE CASCADE  
 ON DELETE SET NULL  
)  
  
TABLESPACE pg\_default;  
  
ALTER TABLE IF EXISTS public.student  
 OWNER to postgres;  
  
  
  
*--  
-- Table: public.study\_form  
--*CREATE TABLE IF NOT EXISTS public.study\_form  
(  
 form\_id smallint NOT NULL,  
 form\_name character varying(64) COLLATE pg\_catalog."default" NOT NULL,  
 CONSTRAINT study\_form\_pkey PRIMARY KEY (form\_id)  
)  
  
TABLESPACE pg\_default;  
  
ALTER TABLE IF EXISTS public.study\_form  
 OWNER to postgres;  
  
  
  
*--  
-- Table: public.subject  
--*CREATE TABLE IF NOT EXISTS public.subject  
(  
 subject\_id integer NOT NULL DEFAULT nextval('subject\_subject\_id\_seq'::regclass),  
 specialty\_id smallint NOT NULL,  
 subject\_name character varying(256) COLLATE pg\_catalog."default" NOT NULL,  
 semester smallint NOT NULL,  
 hours smallint,  
 report\_type\_id smallint,  
 CONSTRAINT subject\_pkey PRIMARY KEY (subject\_id),  
 CONSTRAINT subject\_\_report\_type\_fkey FOREIGN KEY (report\_type\_id)  
 REFERENCES public.report\_type (report\_type\_id) MATCH FULL  
 ON UPDATE CASCADE  
 ON DELETE SET NULL  
 NOT VALID,  
 CONSTRAINT subject\_\_specialty\_fkey FOREIGN KEY (specialty\_id)  
 REFERENCES public.specialty (specialty\_id) MATCH FULL  
 ON UPDATE CASCADE  
 ON DELETE SET NULL  
 NOT VALID  
)  
  
TABLESPACE pg\_default;  
  
ALTER TABLE IF EXISTS public.subject  
 OWNER to postgres;

*-- Queries to get all data from tables.*SELECT *\** FROM study\_form;  
SELECT *\** FROM student\_group;  
SELECT *\** FROM report\_type;  
SELECT *\** FROM specialty;  
SELECT *\** FROM student;  
SELECT *\** FROM subject;  
SELECT *\** FROM mark;

*-- Deletes all data from database.*DELETE FROM mark;  
DELETE FROM subject;  
DELETE FROM student;  
DELETE FROM specialty;  
DELETE FROM report\_type;  
DELETE FROM student\_group;  
DELETE FROM study\_form;  
  
TRUNCATE TABLE subject RESTART IDENTITY CASCADE;  
TRUNCATE TABLE student RESTART IDENTITY CASCADE;  
TRUNCATE TABLE specialty RESTART IDENTITY CASCADE;  
TRUNCATE TABLE report\_type RESTART IDENTITY CASCADE;  
TRUNCATE TABLE student\_group RESTART IDENTITY CASCADE;  
TRUNCATE TABLE study\_form RESTART IDENTITY CASCADE;  
TRUNCATE TABLE mark RESTART IDENTITY CASCADE;

*-- An example of implementing a request to obtain data.  
  
--  
-- Get the number of students by study form.  
--*SELECT  
 study\_form.form\_name AS "Форма обучения",  
 *COUNT*(student.student\_id) AS "Количество студентов"  
FROM study\_form  
LEFT JOIN student ON study\_form.form\_id = student.study\_form\_id  
WHERE study\_form.form\_name = 'Очная'  
GROUP BY study\_form.form\_name;  
  
*--  
-- Get hours and report form by discipline.  
--*SELECT  
 subject.subject\_name AS "Дисциплина",  
 subject.hours AS "Количество часов",  
 report\_type.report\_type\_name AS "Форма отчетности"  
FROM subject  
LEFT JOIN report\_type ON subject.report\_type\_id = report\_type.report\_type\_id  
WHERE subject\_name = 'Математика';  
  
*--  
-- Get student performance.  
--*SELECT  
 mark.mark\_date AS "Дата",  
 subject.subject\_name AS "Дисциплина",  
 mark.mark AS "Оценка"  
FROM mark  
INNER JOIN student ON mark.student\_id = student.student\_id  
INNER JOIN subject ON mark.subject\_id = subject.subject\_id  
WHERE student.student\_id = 1;

*-- Fill database with example data*INSERT INTO specialty  
 (specialty\_code, specialty\_name)  
VALUES  
 ('01.03.01', 'Математика'),  
 ('01.04.01', 'Математика'),  
 ('01.05.01', 'Фундаментальная математика и механика'),  
 ('01.06.01', 'Математика и механика'),

-- TRUNCATE --  
 ('54.05.01', 'Монументально-декоративное искусство'),  
 ('54.03.02', 'Декоративно-прикладное искусство и народные промыслы'),  
 ('54.04.02', 'Декоративно-прикладное искусство и народные промыслы'),  
 ('54.05.02', 'Живопись'),  
 ('54.03.03', 'Искусство костюма и текстиля'),  
 ('54.05.03', 'Графика'),  
 ('54.03.04', 'Реставрация'),  
 ('54.04.04', 'Реставрация'),  
 ('54.05.04', 'Скульптура'),  
 ('55.05.01', 'Режиссура кино и телевидения'),  
 ('55.05.02', 'Звукорежиссура аудиовизуальных искусств'),  
 ('55.05.03', 'Кинооператорство'),  
 ('55.05.04', 'Продюсерство'),  
 ('55.05.05', 'Киноведение');  
  
INSERT INTO report\_type  
 (report\_type\_id, report\_type\_name)  
VALUES  
 (1, 'Зачет'),  
 (2, 'Экзамен'),  
 (3, 'Курсовая работа'),  
 (4, 'Диф. зачет');  
  
INSERT INTO study\_form  
 (form\_id, form\_name)  
VALUES  
 (1, 'Очная'),  
 (2, 'Заочная'),  
 (3, 'Очно-заочная');  
  
INSERT INTO student\_group  
 (group\_id, group\_enroll\_year)  
VALUES  
 (2372, 2022),  
 (2373, 2022),  
 (3372, 2021);  
  
INSERT INTO student  
 (first\_name, last\_name, family\_name, study\_form\_id, group\_id)  
VALUES  
 ('Полуянов', 'Валерий', 'Николаевич', 1, 2372),  
 ('Иван', 'Иванов', 'Иванович', 1, 2372),  
 ('Мария', 'Петрова', 'Сергеевна', 2, 2372),  
 ('Алексей', 'Сидоров', 'Андреевич', 1, 2373),  
 ('Екатерина', 'Федорова', 'Викторовна', 3, 2373),  
 ('Дмитрий', 'Кузнецов', 'Владимирович', 1, 2373),  
 ('Анна', 'Смирнова', 'Игоревна', 2, 3372),  
 ('Николай', 'Попов', 'Павлович', 3, 3372),  
 ('Ольга', 'Васильева', 'Александровна', 1, 3372),  
 ('Сергей', 'Морозов', 'Геннадьевич', 2, 3372),  
 ('Елена', 'Тимофеева', 'Олеговна', 3, 3372);  
  
INSERT INTO subject  
 (specialty\_id, subject\_name, semester, hours, report\_type\_id)  
VALUES  
 (1, 'Математика', 1, 64, 1),  
 (1, 'Физика', 2, 72, 1),  
 (2, 'Информатика', 1, 80, 2),  
 (2, 'Химия', 2, 60, 1),  
 (3, 'История', 1, 56, 2),  
 (3, 'Философия', 2, 48, 2),  
 (4, 'Английский язык', 1, 90, 2),  
 (4, 'Экономика', 2, 64, 1),  
 (5, 'Программирование', 1, 100, 2),  
 (5, 'Статистика', 2, 72, 1);  
  
INSERT INTO mark  
 (subject\_id, student\_id, mark\_date, mark)  
VALUES  
 (1, 1, '2024-01-15', 5),  
 (2, 1, '2024-01-20', 4),  
 (3, 2, '2024-01-18', 3),  
 (4, 2, '2024-01-22', 5),  
 (5, 3, '2024-01-19', 4),  
 (6, 3, '2024-01-25', 5),  
 (7, 4, '2024-01-23', 4),  
 (8, 4, '2024-01-27', 3),  
 (9, 5, '2024-01-26', 5),  
 (10, 5, '2024-01-29', 4),  
 (1, 6, '2024-01-15', 3),  
 (2, 6, '2024-01-20', 4),  
 (3, 7, '2024-01-18', 5),  
 (4, 7, '2024-01-22', 5),  
 (5, 8, '2024-01-19', 3),  
 (6, 8, '2024-01-25', 4),  
 (7, 9, '2024-01-23', 5),  
 (8, 9, '2024-01-27', 3),  
 (9, 10, '2024-01-26', 4),  
 (10, 10, '2024-01-29', 5),  
 (1, 1, '2024-02-15', 4),  
 (2, 2, '2024-02-20', 5),  
 (3, 3, '2024-02-18', 4),  
 (4, 4, '2024-02-22', 3),  
 (5, 5, '2024-02-19', 5),  
 (6, 6, '2024-02-25', 4),  
 (7, 7, '2024-02-23', 5),  
 (8, 8, '2024-02-27', 3),  
 (9, 9, '2025-02-26', null),  
 (10, 10, '2025-02-28', null);

Скрипт генерации SQL-запроса из файла со специальностями:

*"""Generate specialties.csv and INSERT SQL command from specialties.txt"""*def main():  
 txt\_path = r'../database/example\_data/src/specialties.txt'  
 csv\_path = r'../database/example\_data/src/specialties.csv'  
  
 *# Parse source.* out\_csv = 'code,name\n'  
 with open(txt\_path, 'r', encoding='utf-8') as f:  
 for line in f:  
 line\_split = line.strip().split()  
 code = line\_split[0]  
 name = ' '.join(line\_split[1:-1])  
 out\_csv += f'{code};"{name}"\n'  
  
 *# Generate CSV.* with open(csv\_path, 'w+', encoding='utf-8') as f:  
 f.write(out\_csv)  
 print(f'Created: {csv\_path}')  
  
 *# Generate SQL INSERT command.* if not out\_csv:  
 return  
 values = []  
 for line in out\_csv.split('\n')[1:-1]:  
 code, name = line.split(';')  
 values.append(f'\t("{code}", {name})'.replace('"', '\''))  
 sql = (  
 f'INSERT INTO specialty'  
 f'\n\t(specialty\_code, specialty\_name)'  
 f'\nVALUES\n'  
 f'{",\n".join(values)};'  
 )  
 print(sql)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Исходный код пользовательского интерфейса представлен в папке db\_admin, руководство по развертыванию представлено в приложении А.