**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Факультет компьютерных технологий и информатики**

**Кафедра систем автоматизированного проектирования**

отчет

**по производственной практике (преддипломной практике)**

Тема: **Разработка веб-приложения для ведения ведомостей текущего контроля**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9302 |  | Давтян С. Д. |
| Руководитель практики  к.т.н., доцент |  | Михалков В. А. |
| Согласовано:  Руководитель ВКР  д.т.н., профессор |  | Герасимов И. В. |
| Консультант по ВКР  к.т.н. |  | Кузьмин С. А. |

Санкт-Петербург

2023

**ЗАДАНИЕ   
на ПРЕДДИПЛОМНУЮ практику**

|  |
| --- |
| Студент: Давтян С. Д. |
| Группа: 9302 |
| Тема практики: Разработка веб-приложения для ведения ведомостей текущего контроля |
| Цели преддипломной практики:   1. Получение практических навыков, необходимых для выполнения выпускной квалификационной работы. 2. Сбор материала, необходимого для написания выпускной квалификационной работы.   Задание на преддипломную практику:   * + 1. Согласование технического задания на разработку приложения;     2. Обзор литературы по теме работы;     3. Проектирование архитектуры приложения;     4. Разработка информационной части приложения;     5. Разработка интерфейса клиентской части приложения;     6. Разработка кода клиентской части приложения;     7. Разработка кода серверной части приложения;     8. Тестирование приложения. |

|  |  |  |
| --- | --- | --- |
| Сроки прохождения практики: 04.04.2023 – 17.05.2023 | | |
| Дата сдачи отчета: 16.05.2023 | | |
| Дата защиты отчета: 17.05.2023 | | |
| Студент |  | Давтян С. Д. |
| Руководитель практики  к.т.н., доцент |  | Михалков В. А. |
| Согласовано:  Руководитель ВКР  д.т.н., профессор |  | Герасимов И. В. |
| Консультант по ВКР  к.т.н. |  | Кузьмин С. А. |

**ГРАФИК выполнения заданий   
ПРЕДДИПЛОМНОЙ ПРАКТИКИ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент: Давтян С. Д. | | | | | |
| Группа: 9302 | | | | | |
| Тема практики: Разработка веб-приложения для ведения ведомостей текущего контроля | | | | | |
| **№ п/п** | **Наименование работ** | | | **Срок выполнения** |
| 1 | Согласование технического задания на разработку приложения | | | 04.04 – 05.04 |
| 2 | Обзор литературы по теме работы | | | 06.04 – 13.04 |
| 3 | Проектирование архитектуры приложения | | | 14.04 – 17.04 |
| 4 | Разработка информационной части приложения | | | 18.04 – 22.04 |
| 5 | Разработка интерфейса клиентской части приложения | | | 23.04 – 27.04 |
| 6 | Разработка кода клиентской части приложения | | | 28.04 – 01.05 |
| 7 | Разработка кода серверной части приложения | | | 02.05 – 09.05 |
| 8 | Тестирование приложения | | | 10.05 – 11.05 |
| 9 | Оформление отчета по преддипломной практике | | | 12.05 – 16.05 |
| 10 | Сдача отчета и защита итогов практики | | | 17.05 |
| Студент | |  | Давтян С. Д. | | |
| Руководитель практики  к.т.н., доцент | |  | Михалков В. А. | | |

АННОТАЦИЯ

Целью данной выпускной квалификационной работы является создание веб-приложения для ведения ведомостей текущего контроля, которое может быть использовано для организации дистанционного взаимодействия преподавателя со студентами по выполняемым ими работам в течение семестра.

В данном приложении пользователям сначала нужно зарегистрироваться или авторизоваться. В зависимости от роли пользователя будут доступен дальнейший функционал. Если пользователь является студентом, то он сможет просматривать задания, высылать свои работы, просматривать оценки и комментарии к ним. Если пользователь является преподавателем, то он сможет создавать ведомости, вносить в нее студентов, количество работ, скачивать файлы, отправленные студентами, а также проставлять оценки и писать свои замечания.

Основное отличие разрабатываемого приложения от его аналогов состоит в возможности отобразить в одной ячейке таблицы (ведомости) как отметку (или оценку) за выполненную работу, так и историю замечаний и прикреплённых файлов по ней.

Для реализации задуманного были использованы следующие технологии:

* Клиентская часть приложения разрабатывалась с помощью фреймворка Bootstrap5.
* Серверная часть приложения разрабатывалась с помощью языка программирования Python при активном использовании библиотеки Flask.
* Для работы с файлами на стороне сервера использовалась модуль os.
* Для взаимодействия с СУБД на стороне сервера используется библиотека SQLAlchemy.

ABSTRACT

The purpose of this graduate qualification work is to create a web application to keep records of current control, which can be used to organize remote interaction between the teacher and the students on the work they perform during the semester.

In this application, users must first register or log in. Depending on the user's role, further functionality will be available. If the user is a student, they will be able to view assignments, send their work, and view grades. If the user is an instructor, they will be able to create a worksheet, enter students, number of papers, receive files sent by students, and write their notes and assign grades.

The main difference between the application under development and its analogues is the ability to display both the mark (or grade) for the completed work and the history of comments and attached files on it in a single table cell (sheet).

The following technologies were used to implement the idea:

* The client part of the application was developed using the Bootstrap5 framework.
* The server part of the application was developed using the Python programming language with active use of the Flask library.
* The os module was used to handle the files on the server side.
* The SQLAlchemy library is used to interact with the DBMS on the server side.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 8](#_Toc135213955)

[1. ПОСТАНОВКА ЗАДАЧИ 9](#_Toc135213956)

[1.1. Тема проекта 9](#_Toc135213957)

[1.2. Главные задачи 9](#_Toc135213958)

[1.3. Детальное описание приложения 9](#_Toc135213959)

[1.4. Описание предметной области 9](#_Toc135213960)

[1.5. Функционал приложения 11](#_Toc135213961)

[1.6. Используемые технологии 12](#_Toc135213962)

[1.7. Этапы разработки приложения 12](#_Toc135213963)

[2. ВЫБОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ 14](#_Toc135213964)

[2.1. Выбор языка программирования 14](#_Toc135213965)

[2.2. Выбор технологий 17](#_Toc135213966)

[2.3. Выбор базы данных 19](#_Toc135213967)

[2.4. Выбор среды разработки 20](#_Toc135213968)

[3. РАЗРАБОТКА ПРИЛОЖЕНИЯ 23](#_Toc135213969)

[3.1. Архитектурная часть приложения 23](#_Toc135213970)

[3.2. Информационная часть приложения 28](#_Toc135213971)

[3.3. Клиентская часть приложения 31](#_Toc135213972)

[3.4. Серверная часть приложения 34](#_Toc135213973)

[3.5. Тестирование приложения 39](#_Toc135213974)

[ЗАКЛЮЧЕНИЕ 53](#_Toc135213975)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 54](#_Toc135213976)

ВВЕДЕНИЕ

Актуальность разработки приложений для организации учебного процесса непрерывно возрастает. Многие учебные заведения используют дистанционное взаимодействие студентов с преподавателем. Одной из проблем, которая возникает при таком переходе, является необходимость ведения ведомости текущего контроля. Традиционные методы ведения ведомостей на бумажных носителях уже не всегда эффективны и удобны в использовании. Разработка данного веб-приложения позволит существенно облегчить этот процесс.

Существенным отличием данной работы от уже существующих аналогов состоит в возможности отобразить в одной ячейке таблицы (ведомости) как отметку (или оценку) за выполненную работу, так и историю замечаний и прикреплённых файлов по ней.

# ПОСТАНОВКА ЗАДАЧИ

## Тема проекта

Разработка веб-приложения для ведения ведомостей текущего контроля.

## Главные задачи

1. Организация дистанционного взаимодействия преподавателя со студентами по выполняемым ими работам в течение семестра.
2. Разработка приложения с веб-интерфейсом, в котором преподаватель может проставлять отметки о проверенных им работах (а также в комментариях оставлять замечания по ним), а студенты могут читать комментарии, а также добавлять файлы работ.

## Детальное описание приложения

Приложение является многопользовательской системой, в которой пользователи могут быть двух видов: преподаватели и студенты.

Преподаватели решают задачи:

* создание предметных дисциплин;
* создание ведомостей по дисциплинам;
* проверка присланных работ и написание по ним замечаний;
* проставление оценок в ведомость.

Студенты решают задачи:

* добавление файла решения задачи;
* просмотр замечаний от преподавателя.

## Описание предметной области

Таблица «Пользователь»:

* id;
* имя;
* фамилия;
* отчество (если есть);
* e-mail;
* пароль;
* роль;
* id группы;
* список работ;

Таблица «Предмет»:

* id;
* название;
* ведомости;

Таблица «Группа»:

* id;
* номер группы;
* студенты;

Таблица «Ведомость»:

* id;
* учебный год;
* семестр;
* id предмета;
* группы;
* работы;

Таблица «Столбик с работами»:

* id;
* название;
* список работ;
* id ведомости;

Таблица «Работа»:

* id;
* название файла программы;
* оценка за программу;
* название файла отчета;
* оценка за отчет;
* id столбика с работами;
* id пользователя;
* комментарии;

Таблица «Комментарий»:

* id;
* текст комментария;
* время комментария;
* id работы.

## Функционал приложения

* Модуль «Преподаватель»:
  + регистрация нового преподавателя;
  + авторизация преподавателя;
  + уточнение дисциплины и состава контрольных точек по ней (с автоматической генерацией ведомости);
  + просмотр файлов, полученных от студентов;
  + проставление отметки о выполнении работы;
  + запись комментария к работе с замечаниями по работе студента;
  + редактирование отметки о выполнении работы.
* Модуль «Студент»:
  + регистрация нового студента;
  + авторизация студента;
  + редактирование данных профиля студента;
  + добавление файла по выполненной работе;
  + просмотр отметки о выполнении работы;
  + просмотр комментария к работе студента.

## Используемые технологии

* На стороне клиента:

фреймворк Bootstrap5 [1] (HTML5 [2], CSS [3], JavaScript [4]);

библиотека jQuery [5];

инструмент Jinja2 [6].

* На стороне сервера:

язык программирования Python [7];

фреймворк Flask [8];

модуль os [9] (для работы с файлами).

* На стороне базы данных:

СУБД SQLite [10];

библиотека SQLAlchemy [11].

* Среда разработки:

Visual Studio Code 11.76.2 [12];

PyCharm Community Edition 2023.1.1 [13].

## Этапы разработки приложения

1. Согласование ТЗ;
2. Разработка архитектуры приложения;
3. Разработка информационной части (базы данных) приложения;
4. Дизайн клиентской части приложения;
5. Разработка кода клиентской части приложения;
6. Разработка кода серверной части приложения;
7. Тестирование приложения.

# ВЫБОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ

## Выбор языка программирования

Язык программирования является основным инструментом при написании любого программного обеспечения. От него зависит сложность написания тех или иных элементов, а также их производительность. Поэтому создание проекта начинается именно с выбора языка программирования.

При создании веб-приложения стоит учитывать, что нужно написать клиентскую и серверную части приложения.

**Клиентская часть.**

Основная задача клиентской части веб-приложения - облегчить работу конечному пользователю и предоставить ему удобный интерфейс для взаимодействия с серверной частью приложения. Для реализации этой задачи были выбраны языки программирования HTML, CSS и JavaScript.

HTML (HyperText Markup Language) - это язык разметки, который определяет структуру и содержание веб-страницы [2]. Он обеспечивает базовую структуру документа, определяет заголовки, параграфы, списки, таблицы и многие другие элементы страницы.

CSS (Cascading Style Sheets) - это язык стилей, который используется для оформления и стилизации элементов HTML [3]. Он позволяет управлять цветом, шрифтами, размерами и расположением элементов страницы.

JavaScript - это язык программирования, который используется для создания интерактивных элементов веб-страницы [4]. Он позволяет создавать динамические элементы, такие как выпадающие меню, всплывающие окна, анимации и многие другие.

Все эти языки программирования взаимодействуют между собой, чтобы создать удобный и функциональный интерфейс для конечного пользователя. HTML формирует структуру страницы, CSS добавляет стили и оформление, а JavaScript обеспечивает интерактивность и динамичность приложения.

Благодаря этому сочетанию языков программирования, разработчики могут создавать профессиональные веб-приложения, которые удобны и функциональны для пользователей.

**Серверная часть.**

Для написания серверной части веб-приложения используются различные языки программирования. Каждый из этих языков имеет свои преимущества и недостатки, поэтому выбор зависит от конкретных требований к проекту и предпочтений разработчика.

Для написания серверной части было принято решение выбрать язык программирования Python [7] по ряду причин:

* Простота и читаемость кода: Python имеет лаконичный и понятный синтаксис, который делает код читабельным и легко поддерживаемым.
* Опыт разработки: имеется опыт разработки приложений с помощью данного языка программирования.
* Ограниченность времени: веб-приложение нужно написать в кратчайшие сроки.
* Большое сообщество и библиотеки: Python является одним из самых популярных языков программирования и имеет большое сообщество разработчиков. Это в свою очередь позволяет использовать готовые библиотеки для решения различных задач.
* Широкие возможности интеграции: Python может легко интегрироваться с другими технологиями, такими как базы данных, фреймворки, API, что позволяет создавать мощные и расширяемые приложения.
* Низкая стоимость разработки: Python имеет низкий порог вхождения для программистов, что позволяет экономить время на обучении, а также существуют бесплатные инструменты разработки.
* Безопасность: Python использует строгую типизацию данных и встроенные механизмы безопасности, что снижает риск возможных ошибок и уязвимостей в приложении.

Python - высокоуровневый язык программирования, который с приходом новых версий улучшается и становится все более удобным в использовании. Сами основатели языка описывают его как простой в изучении, но мощный в построении инструментов.

Python обладает множеством библиотек и фреймворков, которые можно использовать для написания серверной части веб-приложения. Некоторые из них, такие как Flask [8], широко используются в коммерческих проектах и имеют большие сообщества разработчиков.

Одним из главных преимуществ Python для написания серверной части веб-приложения является его простота и быстрота разработки благодаря простоте синтаксиса и мощному набору инструментов. Кроме того, Python позволяет написать программу в относительно небольшом количестве строк кода, что делает его идеальным выбором для быстрой разработки прототипов.

Но, как и любой язык программирования, у Python есть свои ограничения. Например, он может быть не столь быстрым и неэффективным в обработке больших объемов данных, что может быть важным фактором при выборе языка для создания успешных и популярных веб-приложений.

В целом можно сказать, что Python - отличный выбор при создании серверной части веб-приложения, благодаря своей гибкости, простоте и эффективности. Однако, конечный выбор всегда зависит от особенностей конкретного проекта и условий его разработки.

## Выбор технологий

**Клиентская часть.**

Bootstrap5 - это один из самых популярных и широко используемых фреймворков для фронтенд-разработки веб-приложений [1]. Он был создан на базе HTML [2], CSS [3] и JavaScript [4], что делает его легко доступным и понятным даже для начинающих разработчиков.

Основные причины, по которым для разработки приложения был выбран данный фреймворк:

* Bootstrap5 предлагает простую и эффективную систему сетки, которая помогает упорядочить контент и достичь согласованности дизайна на разных устройствах и платформах. Это значительно упрощает процесс разработки и оптимизации веб-страницы.
* Bootstrap5 предоставляет большой выбор компонентов и элементов дизайна для веб-приложений, таких как кнопки, формы, модальные окна, навигационные меню, карусели и т.д. Они все готовы к использованию и могут быстро интегрироваться в проект.
* Гибкость и расширяемость: он позволяет легко добавлять пользовательские стили и компоненты, что делает его идеальным инструментом для создания уникальных и индивидуальных дизайнов.

Все эти факторы делают Bootstrap5 сильной и надежной платформой для создания клиентской части веб-приложений.

jQuery [5] - это библиотека JavaScript, которая упрощает написание кода. Она позволяет разработчикам быстро и легко манипулировать DOM (объектная модель документа), обрабатывать события, анимировать элементы, работать с AJAX и многими другими задачами. Данная библиотека может значительно упростить обработку пользовательских событий и взаимодействие с пользователями. Также jQuery обладает простым и удобным синтаксисом.

Инструмент Jinja2 [6] был выбран для того, чтобы генерировать HTML-код на стороне сервера. Он облегчает создание динамически генерируемых страниц, и это позволяет использовать шаблоны для повторяющихся элементов сайта. Jinja2 упрощает процесс создания страниц, а также позволяет создавать легко поддерживаемый и модифицируемый код.

Сочетание jQuery и jinja2 является мощным инструментом для создания современных и динамичных веб-приложений.

**Серверная часть.**

Фреймворк Flask [8] - это миниатюрный и легковесный фреймворк для создания серверной части веб-приложений на языке программирования Python [7].

Технология была выбрана Flask по нескольким причинам:

* Является одним из самых популярных фреймворков для создания веб-приложений на Python, что означает, что есть множество библиотек, документации и сообщества, которые могут помочь в разработке.
* Обладает простотой и легковесностью, что позволяет быстро разработать MVP (минимально жизнеспособный продукт) и затем постепенно добавлять новые функции.
* Является гибким фреймворком, который позволяет разработчикам выбирать наиболее подходящие инструменты и библиотеки для своих целей.
* Используя Flask, разработчики могут легко интегрировать другие библиотеки Python, такие как SQLAlchemy [11] для работы с базами данных и Jinja2 [6] для создания HTML-шаблонов [2].

Модуль os [9] (от англ. operating system) - это стандартный модуль языка программирования Python, которая предоставляет функции для работы с операционной системой, такие как доступ к файловой системе, обращение к переменным окружения, управление процессами, создание каталогов, удаление файлов и другое.

Модуль os был выбрана, потому что она предоставляет удобные средства для работы с файловой системой и операционной системой в целом. Она даёт возможность создавать, открывать и закрывать файлы, изменять их содержимое, копировать и перемещать их в нужные каталоги, а также получать информацию о файлах, например, размер, время создания, и многое другое.

## Выбор базы данных

СУБД SQLite [10] — это легкий и быстрый в использовании относительно других СУБД, который является свободным и предоставляет кроссплатформенную поддержку. Она использует файловую систему для хранения данных, в отличие от традиционных СУБД, в которых данные хранятся в отдельном сервере. Одним из основных преимуществ SQLite является его малый размер, что делает его легче и быстрее в установке и настройке.

Данная СУБД была выбрана в связи с тем, что она позволяет хранить данные локально на сервере без необходимости каскадной операции сетью. Это позволяет уменьшить нагрузку на сервер, ускорить работу приложения и упростить его настройку и установку. Кроме того, SQLite обладает прекрасной производительностью, что позволяет быстро получать и обрабатывать данные.

SQLAchemy [11] - это мощный инструмент, который позволяет нам работать с базами данных в приложениях, написанных на Python. Он облегчает работу с базами данных, а также позволяет использовать ORM (объектно-реляционное отображение) для работы с данными в базе.

ORM [14] - это технология, которая позволяет нам работать с базой данных в терминах объектов и классов в Python [7]. Благодаря этому мы можем использовать привычный синтаксис Python для работы с базой, вместо того чтобы использовать SQL язык.

Использование SQLAlchemy и ORM позволяет легко и быстро работать с базой данных. Они упрощают изменение данных, запросы и поиск информации в базе данных. Так же это делает код чище и более легким в понимании, который можно легко поддерживать и расширять в будущем.

## Выбор среды разработки

Выбор среды разработки очень важен для программиста, поскольку работа с ней занимает значительную часть времени. Он во многом зависит от языка программирования, который используется в проекте. Каждая среда обладает определенными функциональными возможностями для работы с определенным языком программирования и набором библиотек. Некоторые среды разработки имеют более простой и интуитивный интерфейс, другие - более продвинутые функциональности. Среда разработки облегчает процесс написания кода, обеспечивает средства отладки и тестирования, предоставляет инструменты для управления версиями и обмена кодом между участниками команды.

**Клиентская часть.**

Для клиентской части был выбран редактор кода Visual Studio Code [12]. Это кроссплатформенный бесплатный редактор кода, созданный компанией Microsoft. Он поддерживает множество языков программирования, включая HTML [2], CSS [3], JavaScript [4], Python [7] и многие другие.

Вот несколько причин, почему Visual Studio Code стоит выбрать для написания клиентской части веб-приложения:

1. Интегрированная среда разработки: Visual Studio Code включает в себя множество полезных функций, таких как автодополнение кода, форматирование кода, отладка и управление версиями. Вам не придется использовать дополнительные инструменты и расширения, чтобы выполнить большую часть своей работы.

2. Множество плагинов и расширений: Visual Studio Code имеет широкую поддержку и активное сообщество, позволяющее использовать различные плагины и расширения для улучшения работы с редактором кода. Вы можете улучшить свой опыт разработки, устанавливая и настраивая различные плагины.

3. Хорошая производительность: Visual Studio Code известен своей быстрой загрузкой времени и выполнением большинства операций в режиме реального времени. Это делает его идеальным инструментом для разработки веб-приложений.

Кратко о функциях VS Code:

* Мощный редактор кода с поддержкой синтаксиса для многих языков программирования;
* Отладчик кода;
* Расширяемость и гибкие настройки;
* Широкая поддержка фреймворков и языков программирования;
* Бесплатный и с открытым исходным кодом.

**Серверная часть.**

Для написания серверной части приложения была выбрана интегрированная среда разработки для языка программирования Python – PyCharm [13]. Она представляет собой мощный и удобный инструмент, который позволяет быстро и просто создавать, отлаживать и разрабатывать приложения.

Что касается разработки серверной части веб-приложений, PyCharm предоставляет ряд преимуществ:

1. Поддержка различных фреймворков: PyCharm поддерживает все популярные фреймворки Python [7] для веб-разработки, включая Flask [8], который будет использован в проекте.

2. Инспектирование кода: PyCharm предоставляет подсказки и автодополнения кода, а также позволяет быстро переходить к определению функций и классов.

3. Отладка: в PyCharm есть удобный и мощный отладчик, который позволяет отслеживать исполнение кода и находить ошибки.

4. Управление проектом: PyCharm предоставляет множество инструментов для управления проектом, включая возможность автоматической установки зависимостей и многое другое.

6. Редактирование файлов: PyCharm предоставляет удобный редактор, который позволяет быстро и удобно редактировать файлы Python, HTML [2], CSS [3] и JavaScript [4].

# РАЗРАБОТКА ПРИЛОЖЕНИЯ

## Архитектурная часть приложения

**Архитектура приложения.**

Архитектура приложений — это структурный принцип, по которому создано приложение. Каждому архитектурному виду свойственны свои характеристики, свойства и отношения между компонентами. Компонент — мелкая или крупная логическая и независимая часть архитектурной системы приложения.

Архитектуру приложений важно продумывать перед стартом разработки. Четко продуманная архитектура — это залог работоспособности и удобства будущей программы. Реакция программы на действия пользователя, возможность справляться с высокими нагрузками, зависания — все эти и другие потенциальные проблемы зависят от того, насколько качественно изначально продумали архитектуру приложения.

В разрабатываемом приложении используется многослойная архитектура.

Многослойная архитектура разделяет программу на следующие слои:

1. Слой представления — отвечает за интерфейс пользователя;
2. Слой бизнес-логики — отвечает за функционал и логику приложения и не отвечает за интерфейс;
3. Слой передачи данных — отвечает за работу с базами данных и обработку информации.

Каждый элемент в приложении «проходит» через все слои. Такая архитектура считается:

1. Простой в реализации;
2. Абстрактной;
3. Защищенной за счет изолированности каждого слоя;
4. Легко управляемой.

Такая архитектура свойственна небольшим приложениям, так как ее реализация возможна при монолитной структуре. Поэтому обслуживание и масштабирование больших приложений на основе этой архитектуры затруднено.

Разрабатываемый проект состоит из двух компонентов.

На стороне сервера используется Flask [8].

Flask – это фреймворк предназначенный для быстрого развертывания веб-приложений. Он не требует конфигурации, а процесс развертывания занимает не более пяти минут.

Общение между приложениями будет осуществляться через API, данные будут передаваться в формате JSON.

На рисунке 1 представлена работа архитектуры приложения.

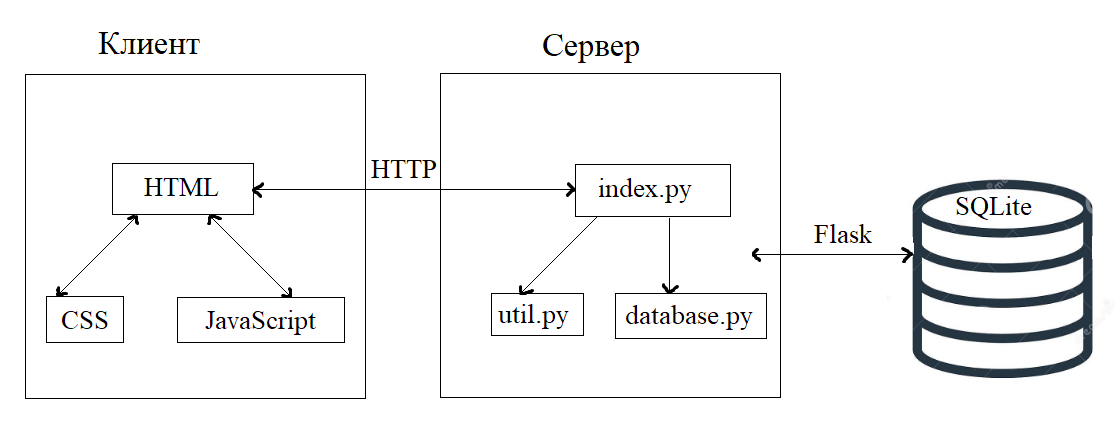


Рисунок 1 – Архитектура приложения

**MVC.**

При разработке приложения использовался паттерн Model View Controller.

MVC – это схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер, таким образом, что модификация каждого компонента может осуществляться независимо. Это очень удобно поскольку если необходимо изменить один из бизнес-процессов, то нет необходимости лезть во все участки кода, а лишь отредактировать изолированный участок. Причем это касается также и двух других компонентов, при современном развитии приложений дизайн устаревает буквально каждый день. И дизайнеры, разработчики могут вносить новые элементы дизайна и обновлять внешний вид страниц, не опасаясь за работоспособность других модулей.

Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.

Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Как уже ранее отмечалось основная цель применения этой концепции состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

К одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;

Не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) — для этого достаточно использовать другой контроллер;

Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

На рисунке 2 представлен схема работы MVC паттерна с точки зрения пользователя.

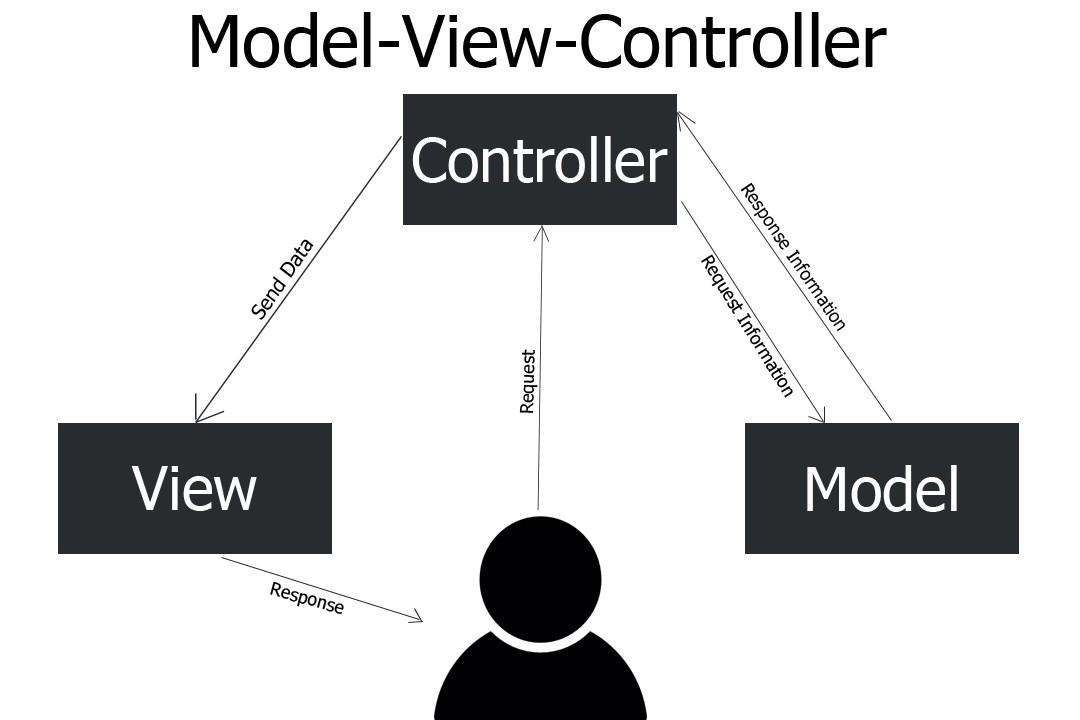


Рисунок 2 – Схема работы MVC

**Структура проекта.**

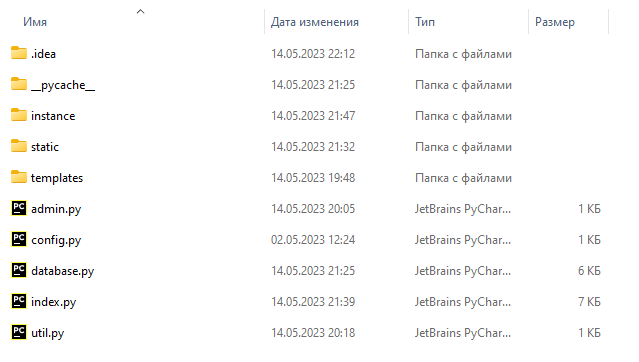


Рисунок 3 – Структура директории проекта

Ниже приводится краткое описание файлов и папок, представленных на рисунке.

.*idea* – Папка, содержащая конфигурационные файлы самой среды разработки. Она не относится непосредственно к работе приложения.

*\_pycache\_* – Кэш работы среды разработки, также не имеет прямого отношения к работе приложения.

*static* – Содержит файлы, а также таблицы стилей для оформления веб-страниц.

*templates* – Папка, содержащая HTML-шаблоны веб-страниц, такие, как страница профиля или страница личной переписки.

*static/upload* – Папка, содержащая все файлы, загружаемые пользователями на сервер.

*instance* – Папка, содержащая файл локальной базы данных SQLite [10]. В ней хранится вся информация, необходимая для работы приложения.

*index.py* – Основной файл проекта. Содержит все функции маршрутизации, а также точку старта программы.

*database.py –* Файл, содержащий ORM представление классов базы данных.

*config.py –* Файл, реализующий конфигурационные настройки приложения.

*util.py –* Файл, содержащий вспомогательные методы, такие, как форматирование строки даты и времени под удобный и читаемый вид.

*admin.py –* Файл, содержащий представление панели администратора, а также ее конфигурацию.

## Информационная часть приложения

**Проектирование базы данных.**

Проектирование баз данных является одной из важнейших частей разработки информационных систем. Этот этап предшествует непосредственной реализации БД и включает в себя определение структуры БД, выбор методов хранения данных, оптимизацию запросов и многое другое.

На первом этапе проектирования определяется структура базы данных. Этот этап включает в себя определение типов данных, которые будут использоваться для хранения информации, а также определение связей между различными таблицами. Также на этом этапе проводится анализ производительности БД, так как структура БД напрямую влияет на ее эффективность.

На втором этапе выбираются методы хранения данных. В определении методов хранения данных учитываются критерии, такие как скорость доступа к данным, размеры БД, требуемый уровень безопасности и т. д. Одним из наиболее распространенных методов хранения данных является использование реляционной СУБД.

На третьем этапе проводится оптимизация запросов к БД. Для получения эффективной работы базы данных необходимо использовать правильные запросы, а также оптимизировать их. Оптимизация запросов включает в себя проектирование индексов, использование правильных типов соединений между таблицами и т. д.

Таким образом, физическое проектирование БД является важным этапом создания информационных систем. В определении структуры БД, выборе методов хранения данных и оптимизации запросов к БД учитываются многие факторы, которые влияют на эффективность и безопасность БД.

ER-диаграмма [15] – это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы.

На рисунке 4 представлена ER-диаграмма спроектированной базы данных.

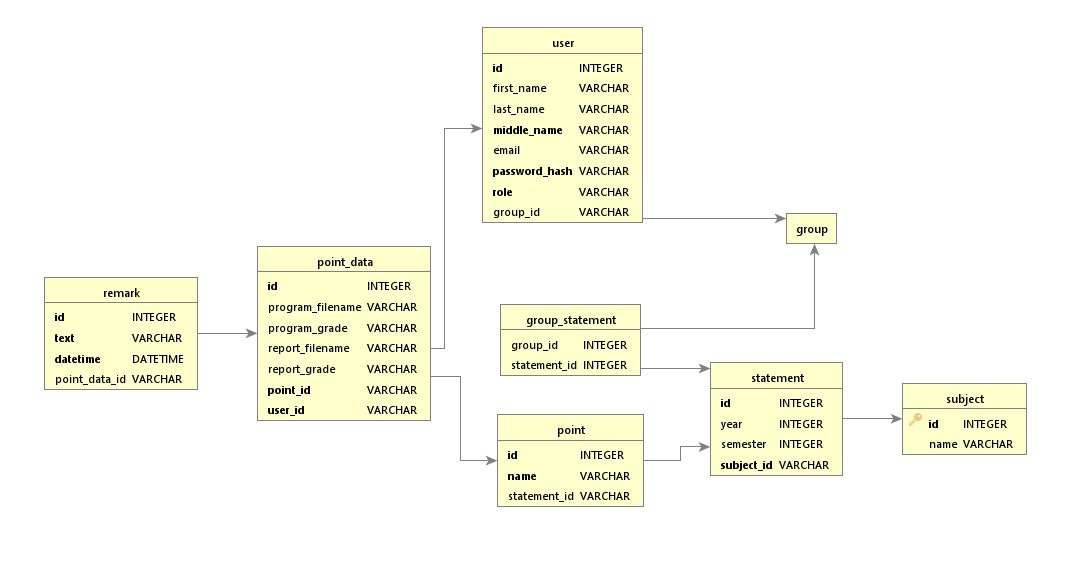


Рисунок 4 – ER-диаграмма

Спроектированная база данных находится в третьей нормальной форме, так как значения всех атрибутов всех сущностей атомарны; все атрибуты, не входящие в первичный ключ, связаны с ним полной функциональной зависимостью; ни один ключевой атрибут функционально не зависит от любого другого не ключевого атрибута.

**Работа с базой данных с помощью технологии ORM.**

Flask [7] для работы с базой данных использует механизм ORM [14].

ORM (Object-Relational Mapping) – это технология программирования которая связывает базы данных с концепциями объектно-ориентированных программ, создавая «виртуальную объектную БД».

Достоинства технологии ORM:

* Возможность оперировать элементами языка программирования, а не элементами реляционной модели данных
* Автоматическое создание SQL-запросов (те самые методы для получения значений полей класса)
* ORM избавляет от необходимости работы с SQL и проработки тонны SQL кода который однообразен и подвержен ошибкам
* ORM изолирует код программы от данных

Недостатки технологии ORM:

* Дополнительная нагрузка на программиста, поскольку ему необходимо изучать этот дополнительный уровень абстракции
* Отсутствие явных SQL запросов мешает в некоторых случаях оптимизировать получение данных
* Если в реализации ORM присутствуют ошибки, то отлавливать их очень трудно.

Механизм работы ORM схематично изображен на рисунке 5.

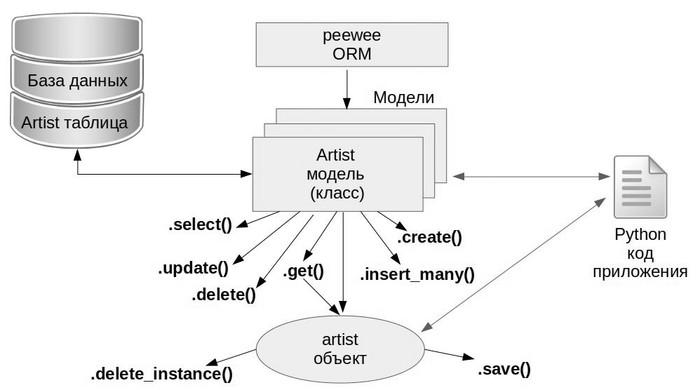


Рисунок 5 – Иллюстрация работы механизма ORM

## Клиентская часть приложения

На рисунке 6 представлен код программы, с помощью которого реализован показ страницы ведомости.



Рисунок 6 – Код страницы ведомости

На рисунках 7, 8 представлены коды, реализующие загрузку файлов программы и отчета со стороны клиента.

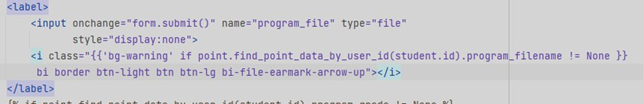


Рисунок 7 – Код загрузки файла программы со стороны клиента



Рисунок 8 – Код загрузки файла отчета со стороны клиента

На рисунке 9 представлен код, реализующий проставление оценки преподавателем студенту.

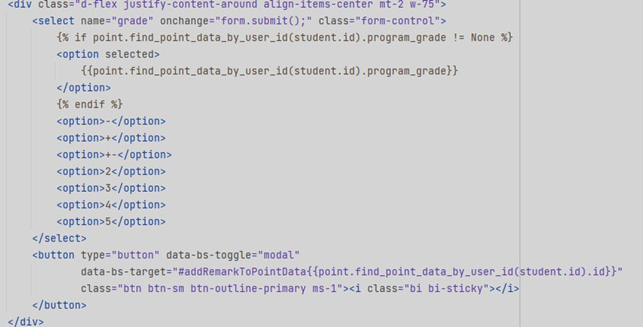


Рисунок 9 – Код проставления оценки со стороны клиента

На рисунке 10 представлен код шаблона, который отвечает за добавление замечания.



Рисунок 10 – Код добавления замечания

## Серверная часть приложения

**Модели данных.**

Для хранения таблицы «Пользователь» создается отдельных класс User, где поля класса соответствуют атрибутам этой таблицы, и у класса должны существовать методы для получения значений из этих полей.

На рисунках 11, 12, 13, 14, 15, 16, 17 представлены классы User, Subject, Group, Statement, Point, PointData, Remark и их методы.

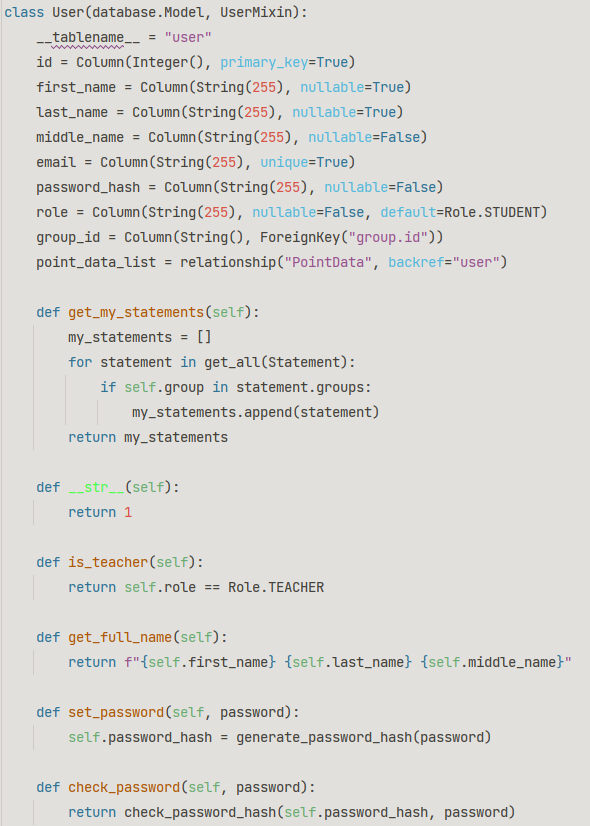


Рисунок 11 – Класс User и его методы

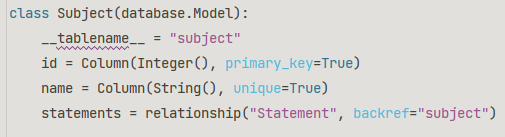


Рисунок 12 – Класс Subject и его методы



Рисунок 13 – Класс Group и его методы



Рисунок 14 – Класс Statement и его методы

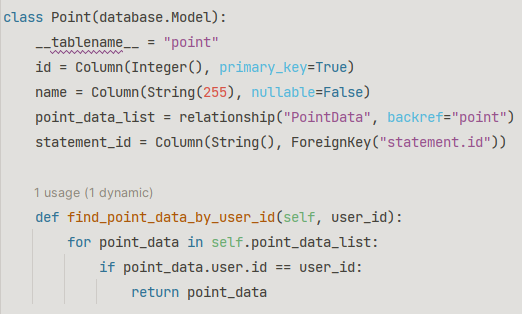


Рисунок 15 – Класс Point и его методы

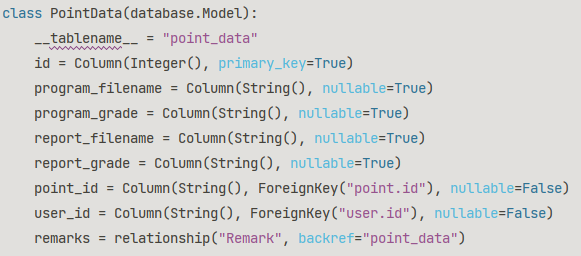


Рисунок 16 – Класс PointData и его методы

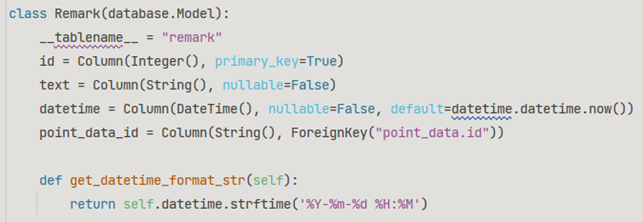


Рисунок 17 – Класс Remark и его методы

На рисунке 18 представлен код, который отвечает за обработку загрузки файлов на сервер.

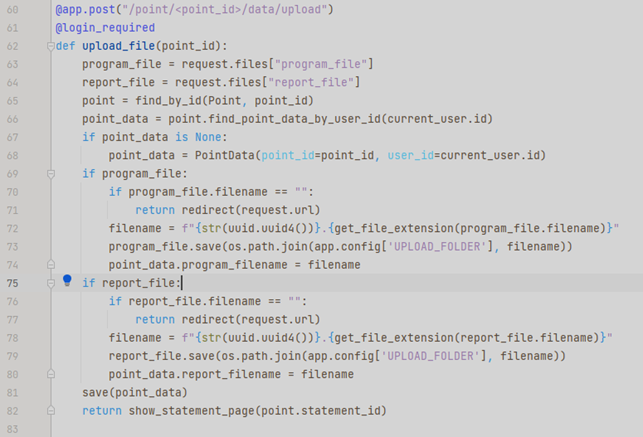


Рисунок 18 – Код загрузки файла на сервер

Ниже на рисунке 19 представлен код, который отвечает за скачивание файлов с сервера.



Рисунок 19 – Код скачивания файла с сервера

На рисунках 20, 21 и 22 представлены коды для обработки запросов на отображение ведомости, а также для выставления оценок и замечаний.



Рисунок 20 – Код, реализующий отображение ведомости

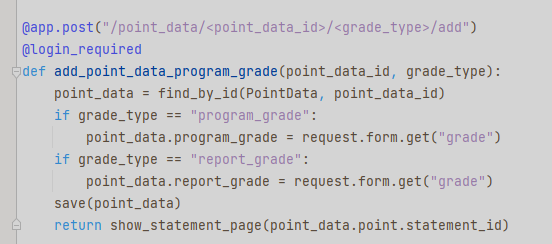


Рисунок 21 – Код, реализующий добавление оценки

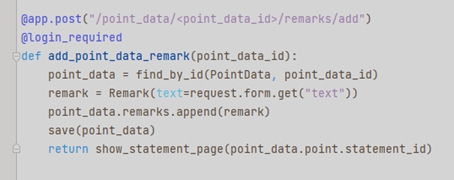


Рисунок 22 – Код, реализующий добавление замечания

## Тестирование приложения

**Авторизация и регистрация.**

Для работы в приложении пользователь должен пройти процесс регистрации или авторизации. На рисунках 23, 24 представлены окна авторизации и регистрации пользователя.

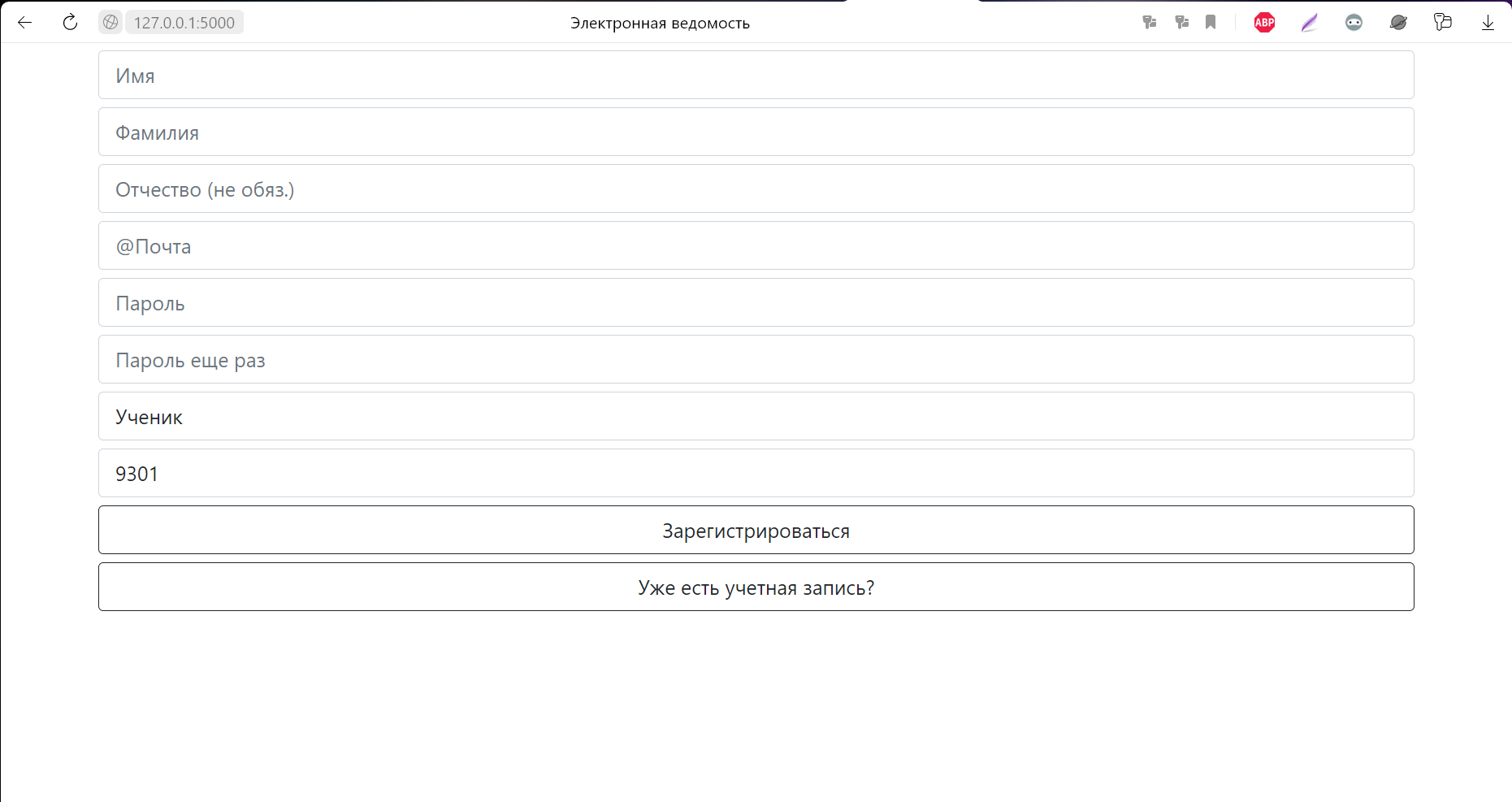


Рисунок 23 – окно регистрации пользователя

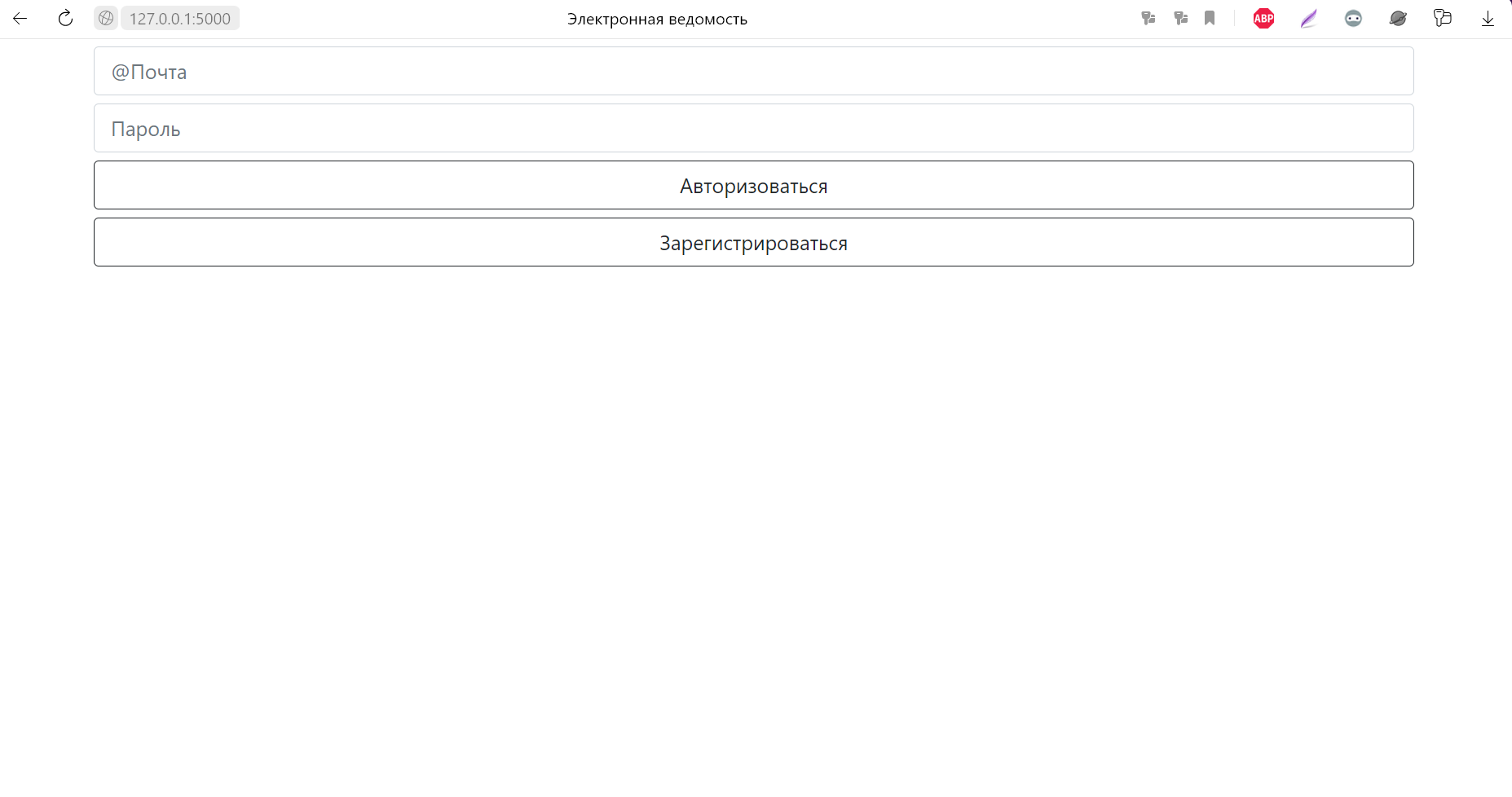


Рисунок 24 – окно авторизации пользователя

**Основной интерфейс приложения.**

В приложении предусмотрено два типа работы в зависимости от роли пользователя:

* Студент;
* Преподаватель.

При входе в аккаунт у преподавателя открывается экран, на котором отображены все существующие ведомости, а также доступно три кнопки:

* «Главная»;
* «Добавить ведомость»;
* «Выйти из учетной записи».

На рисунке 25 представлен экран преподавателя после входа в аккаунт.

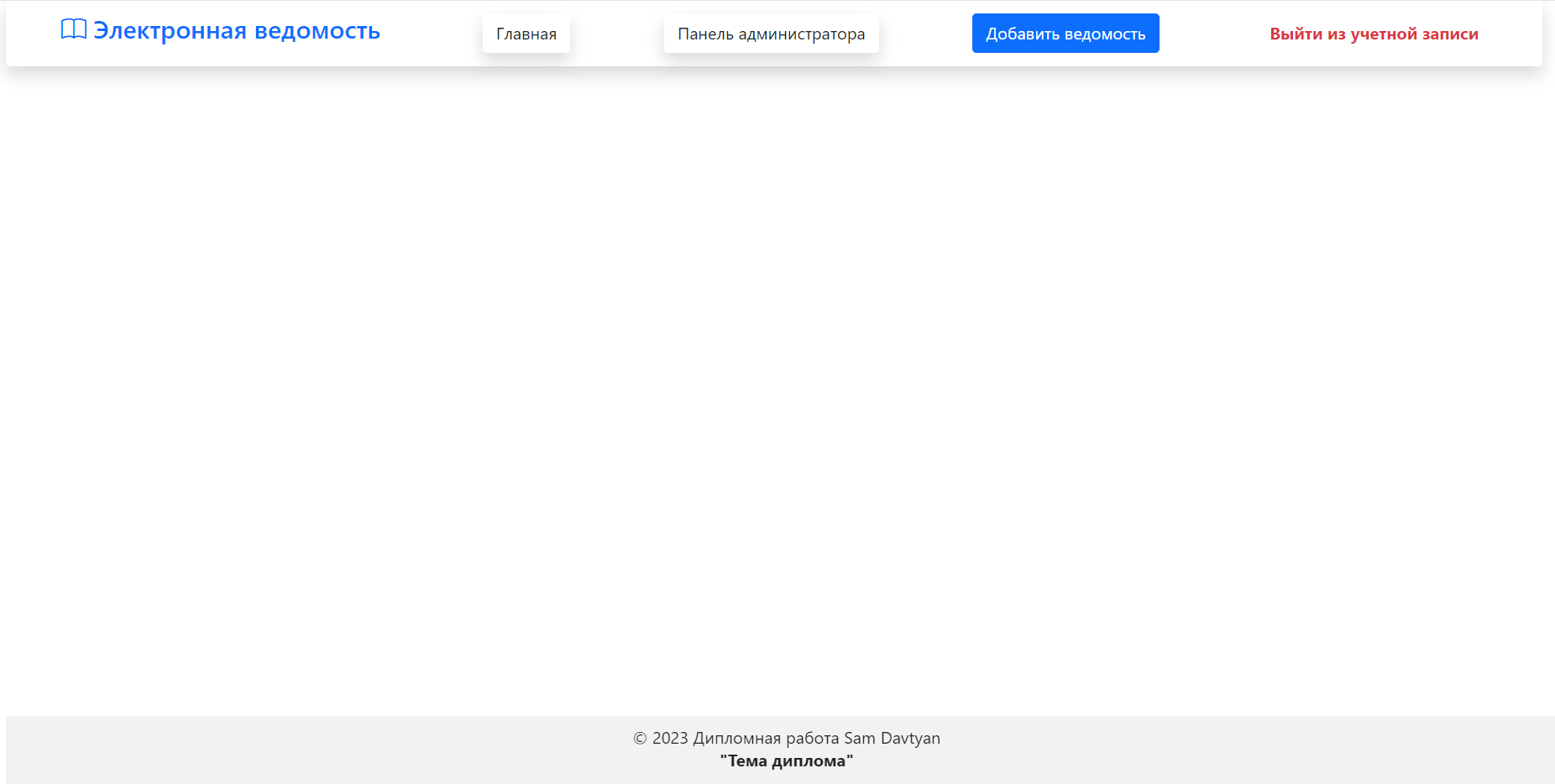


Рисунок 25 – экран преподавателя после входа в аккаунт

Для студента также открывается экран с доступными ему ведомостями и две кнопки:

* «Главная»;
* «Выйти из учетной записи».

На рисунке 26 представлен экран студента после входа в аккаунт.

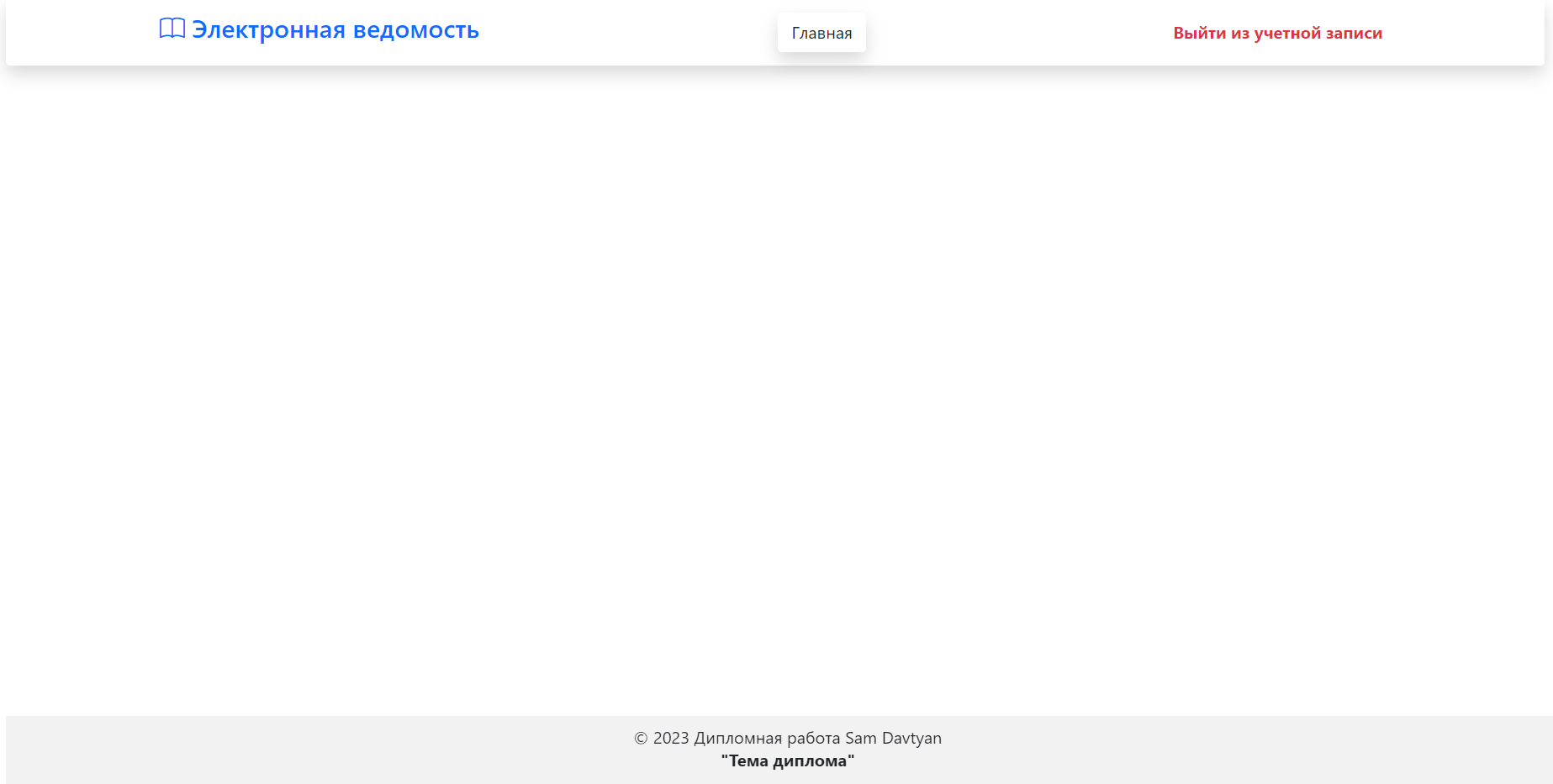


Рисунок 26 – экран преподавателя входа в аккаунт

Далее для преподавателя доступна возможность добавления ведомости. При нажатии на кнопку «Добавить ведомость» появляется всплывающее окно, в котором преподаватель может указать:

* «Семестр» – текстовое поле, в которое преподаватель может вписать номер текущего семестра, по которому создается ведомость.
* «Год» – поле с календарем, в котором преподаватель может выбрать текущий учебный год.
* «Дисциплина» – выпадающее окно, в котором преподаватель может выбрать дисциплину, по которой будет создана ведомость. Дисциплины заранее должны быть добавлены администратором.

На рисунке 27 представлено всплывающее окно для преподавателя, с помощью которого создается ведомость.

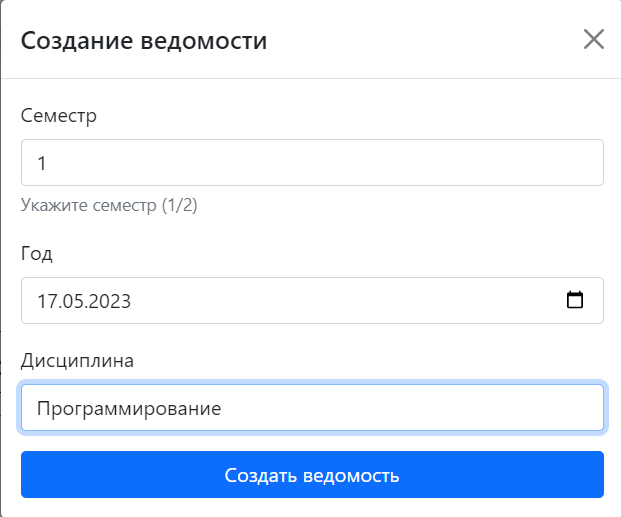


Рисунок 27 – всплывающее окно создания ведомости

После нажатия кнопки «Создать ведомость» на экране преподавателя и студентам появляется созданная ведомость и запись о том, что она успешно добавлена.

На рисунке 28 представлен экран преподавателя после создания ведомости.

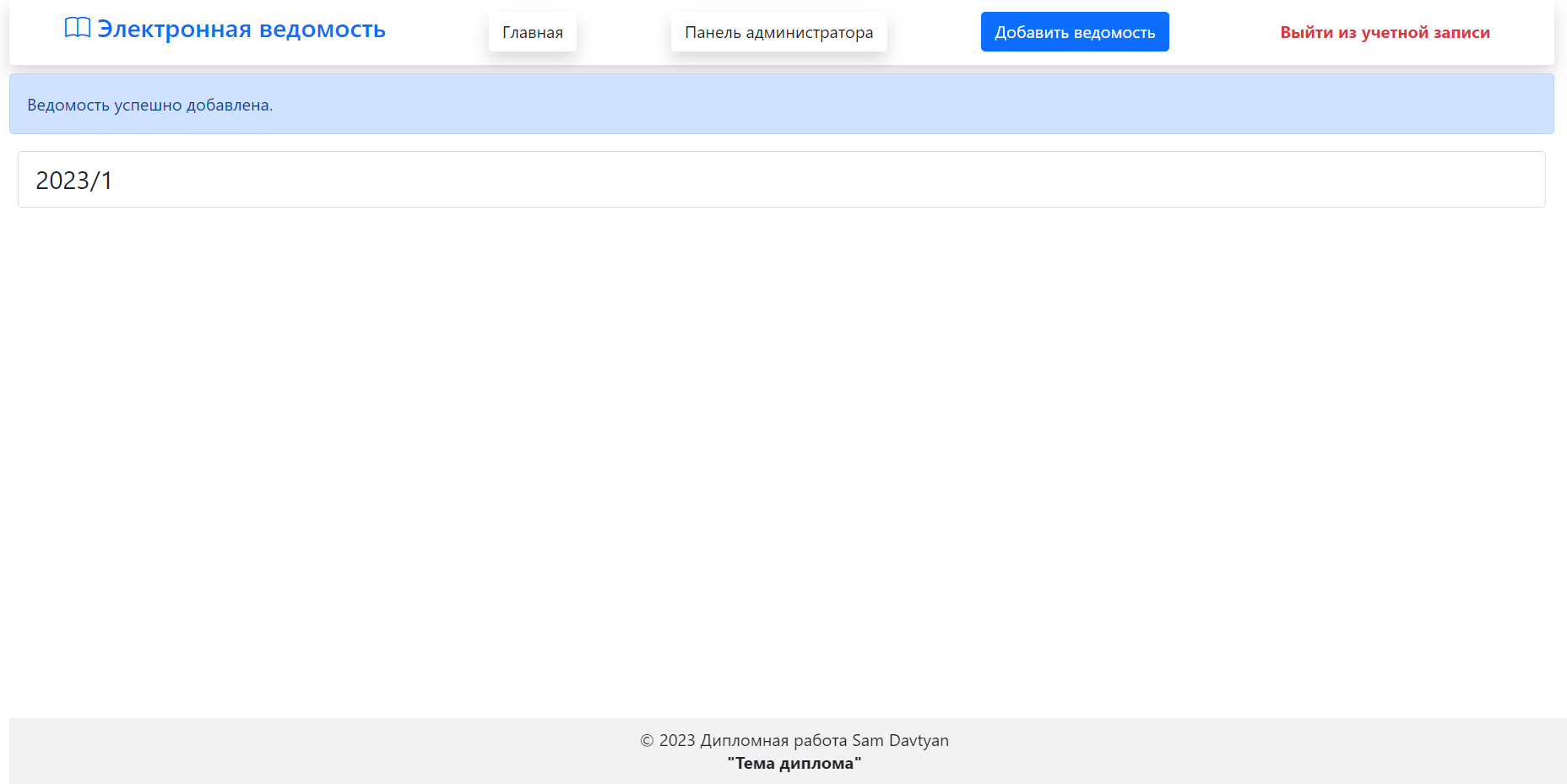


Рисунок 28 – Экран преподавателя после добавления новой ведомости

После нажатия на ведомость, преподаватель попадает на экран ведомости. Изначально он состоит из страницы, на которой указано:

* название дисциплины;
* список добавленных групп;
* пустая ведомость.

На рисунке 29 представлен экран преподавателя после перехода в только что созданную ведомость.

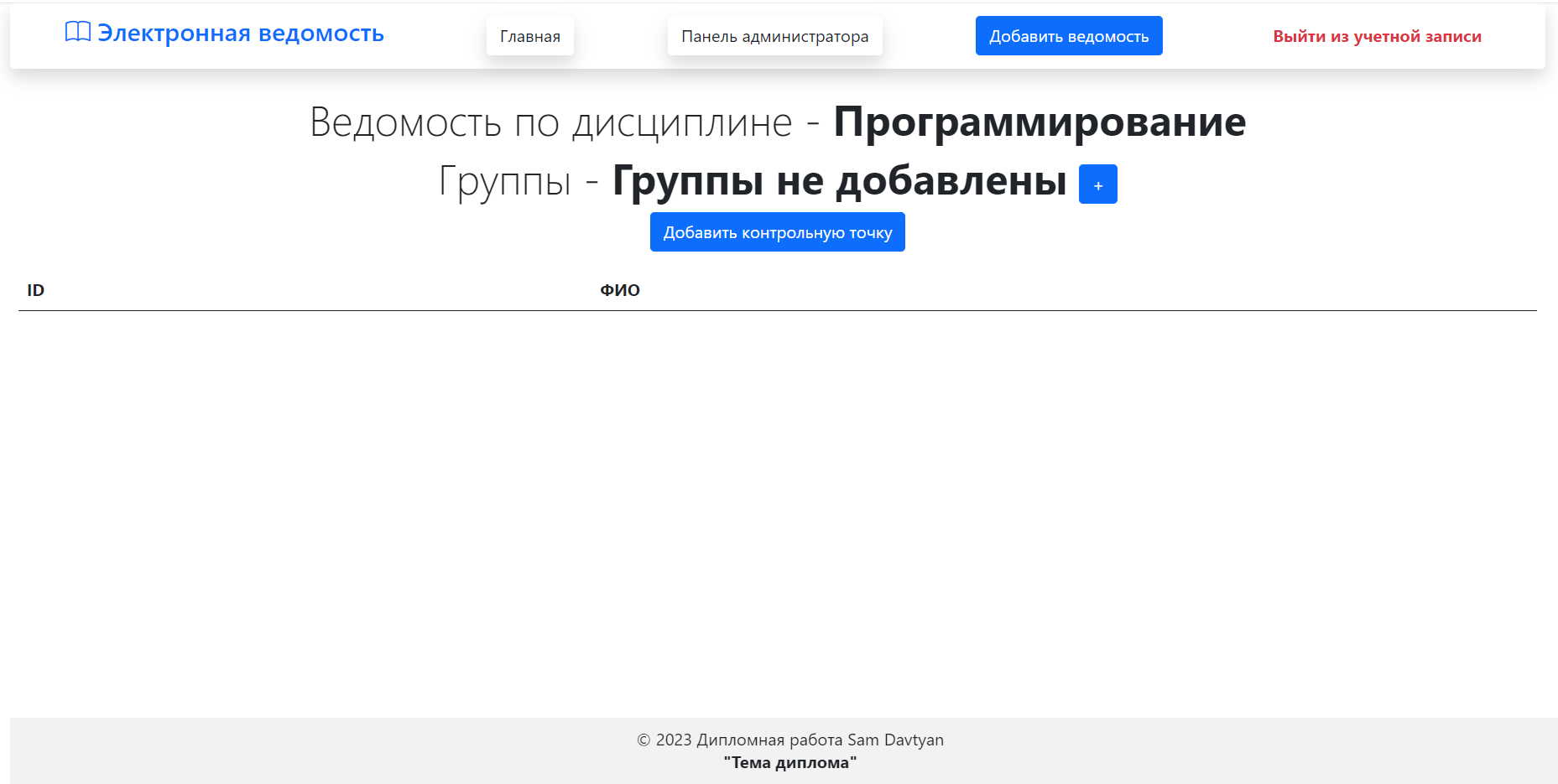


Рисунок 29 – Страница только что созданной ведомости

Преподаватель может добавить группы и контрольные точки. Для добавления группы он должен нажать на кнопку «+». При нажатии данной кнопки появляется всплывающее окно, с помощью которого преподаватель может добавить группу. Всплывающее окно состоит из одного выпадающего списка, в котором преподаватель выбирает номер группы. Номера доступных групп добавляются администратором.

На рисунках 30, 31 представлено всплывающее окно добавления группы.

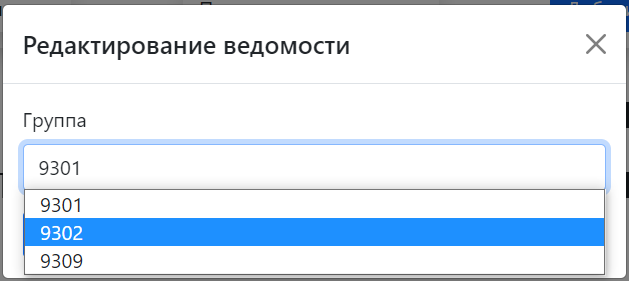


Рисунок 30 – выпадающий список для выбора номера группы

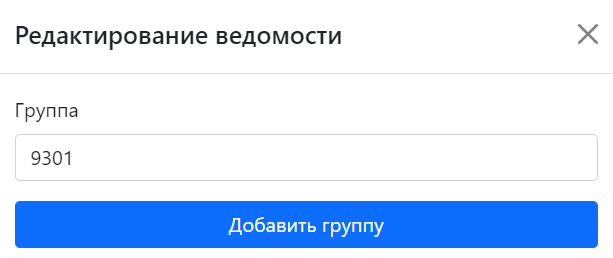


Рисунок 31 – всплывающее окно для добавления группы

После того, как преподаватель добавляет группу, в ведомости появляются все студенты, обучающиеся в данной группе.

На рисунке 32 представлена ведомость после добавления в нее группы 9302. Видно, что обучающиеся в группе 9302 и имеющие аккаунт в данном приложении студенты появились в ведомости.

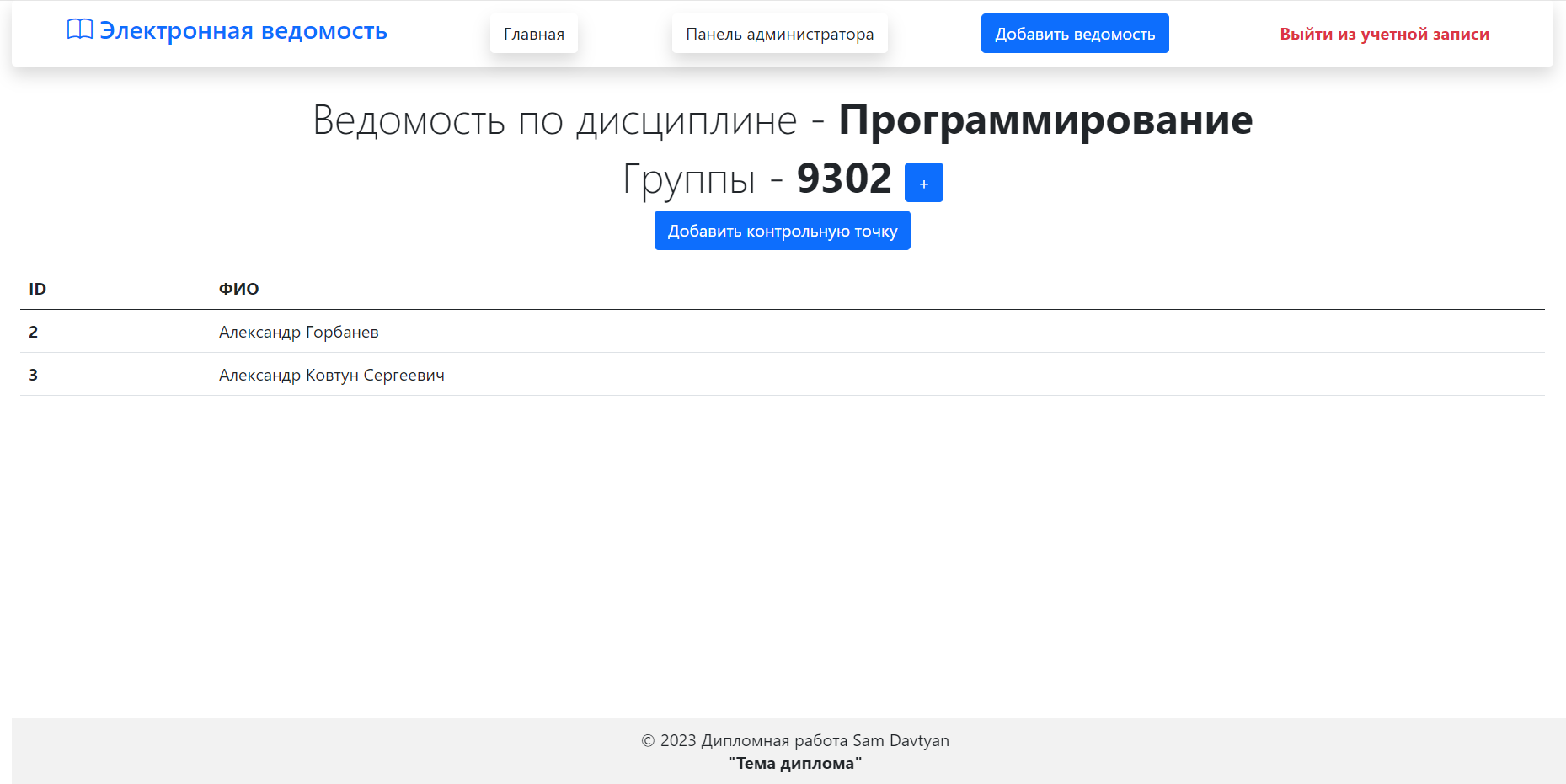


Рисунок 32 – ведомость после добавления группы

Далее преподаватель может добавить контрольные точки. Для этого он может нажать на кнопку «Добавить контрольную точку». После этого появится всплывающее окно, в котором преподаватель в текстовом поле может вписать название контрольной точки, которая в виде столбика появится в ведомости.

На рисунке 33 представлено всплывающее окно для добавления новой контрольной точки.

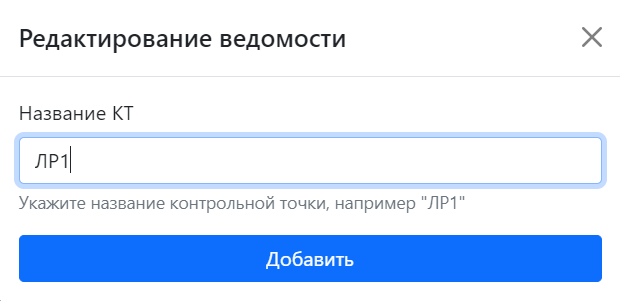


Рисунок 33 – всплывающее окно добавления контрольной точки

После добавления контрольной точки, она появляется в виде столбика таблицы. После добавления нескольких контрольных точек у ведомости появляется вид таблицы.

На рисунках 34, 35 представлена ведомость после добавления контрольных точек.

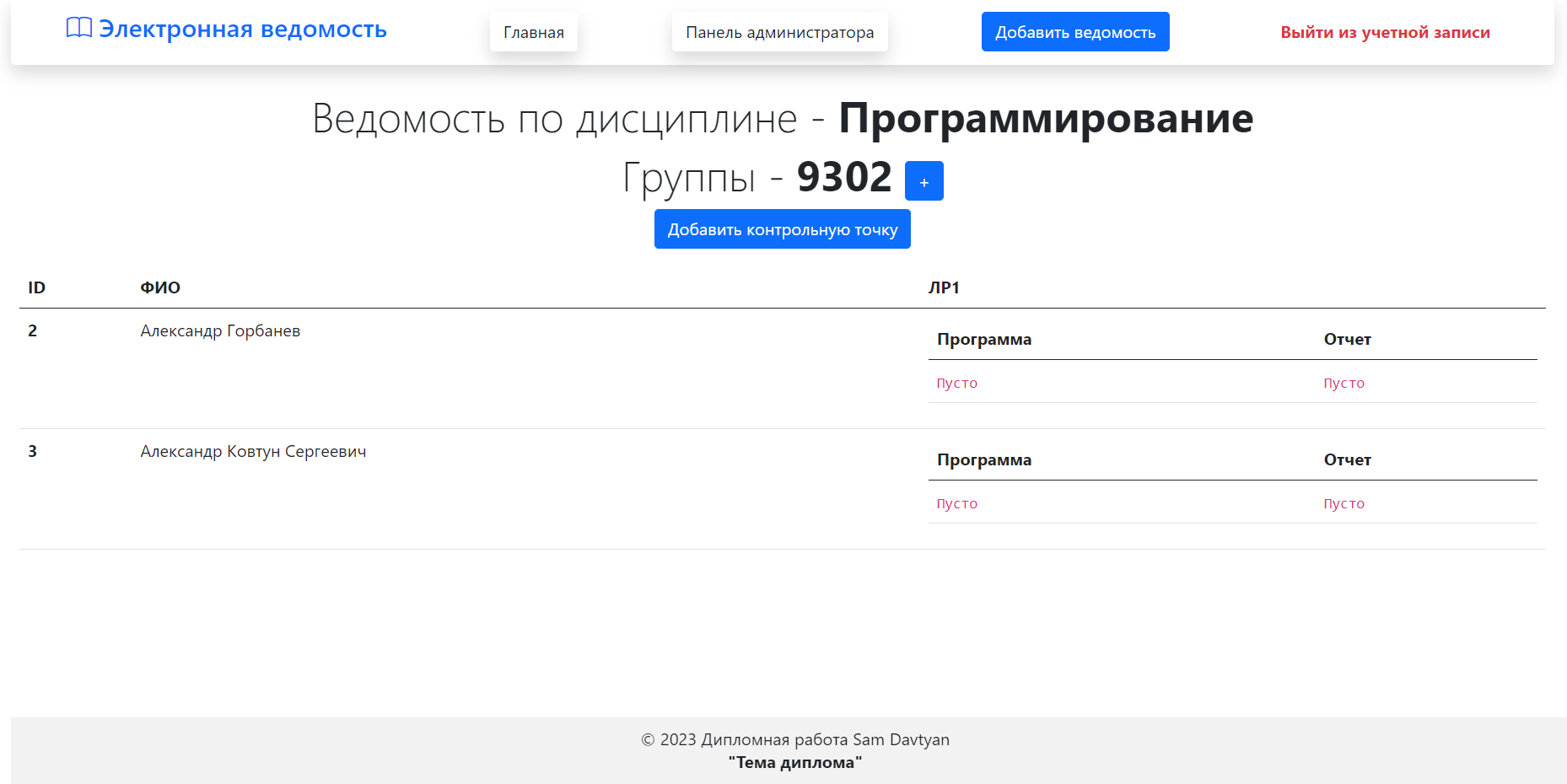


Рисунок 34 – ведомость после добавления первого столбца

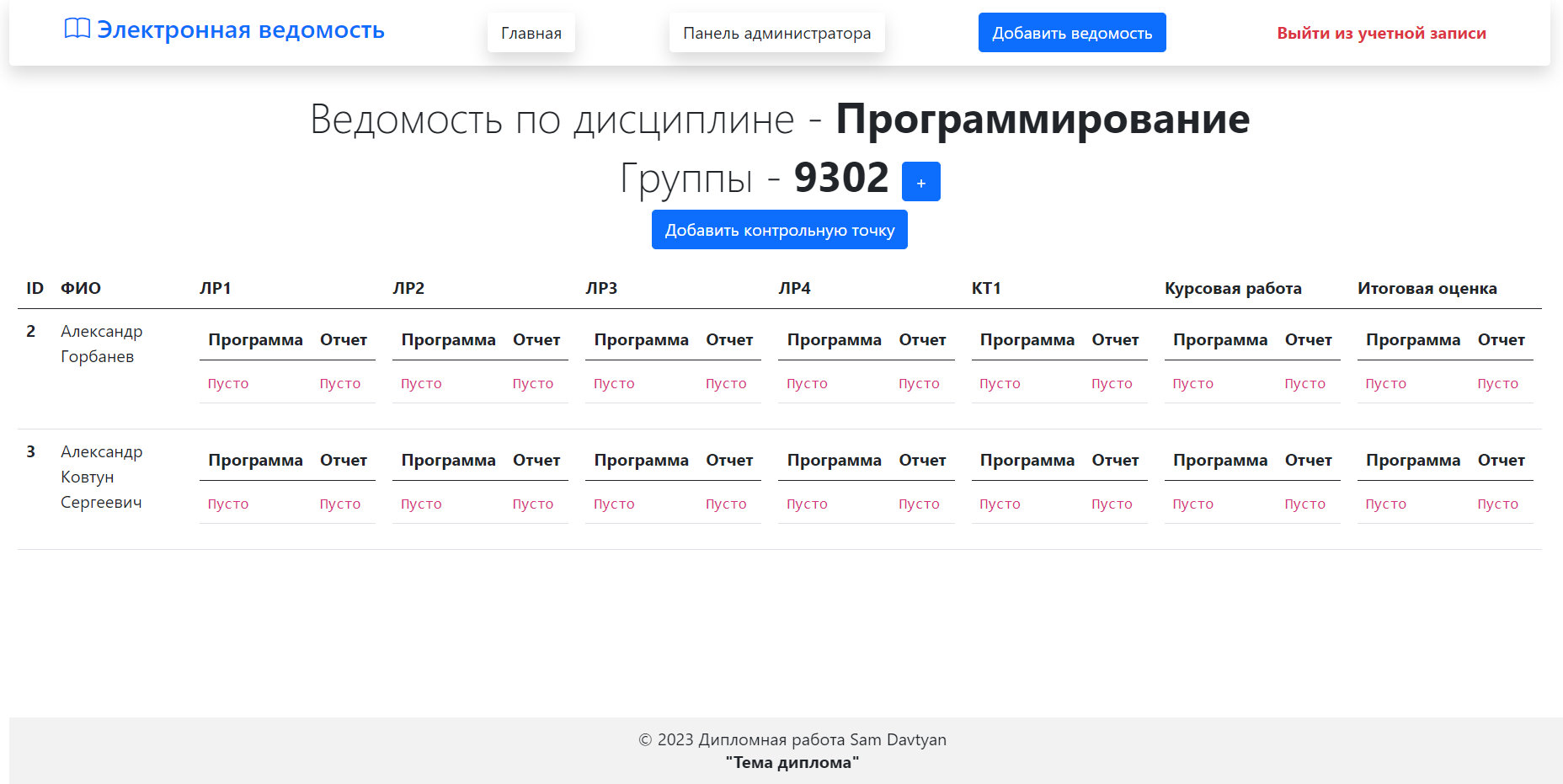


Рисунок 35 – ведомость после добавления всех столбцов

В каждом столбце присутствуют две ячейки: «Программа» и «Отчет». Изначально они пустые, потому что пока что студенты ничего не добавляли.

Ведомость появляется на странице у студентов только после того, как преподаватель добавит их группу с помощью кнопки «+».

На рисунке 36 представлена страница, которую видят студенты после того, как преподаватель предоставил им доступ к ведомости.

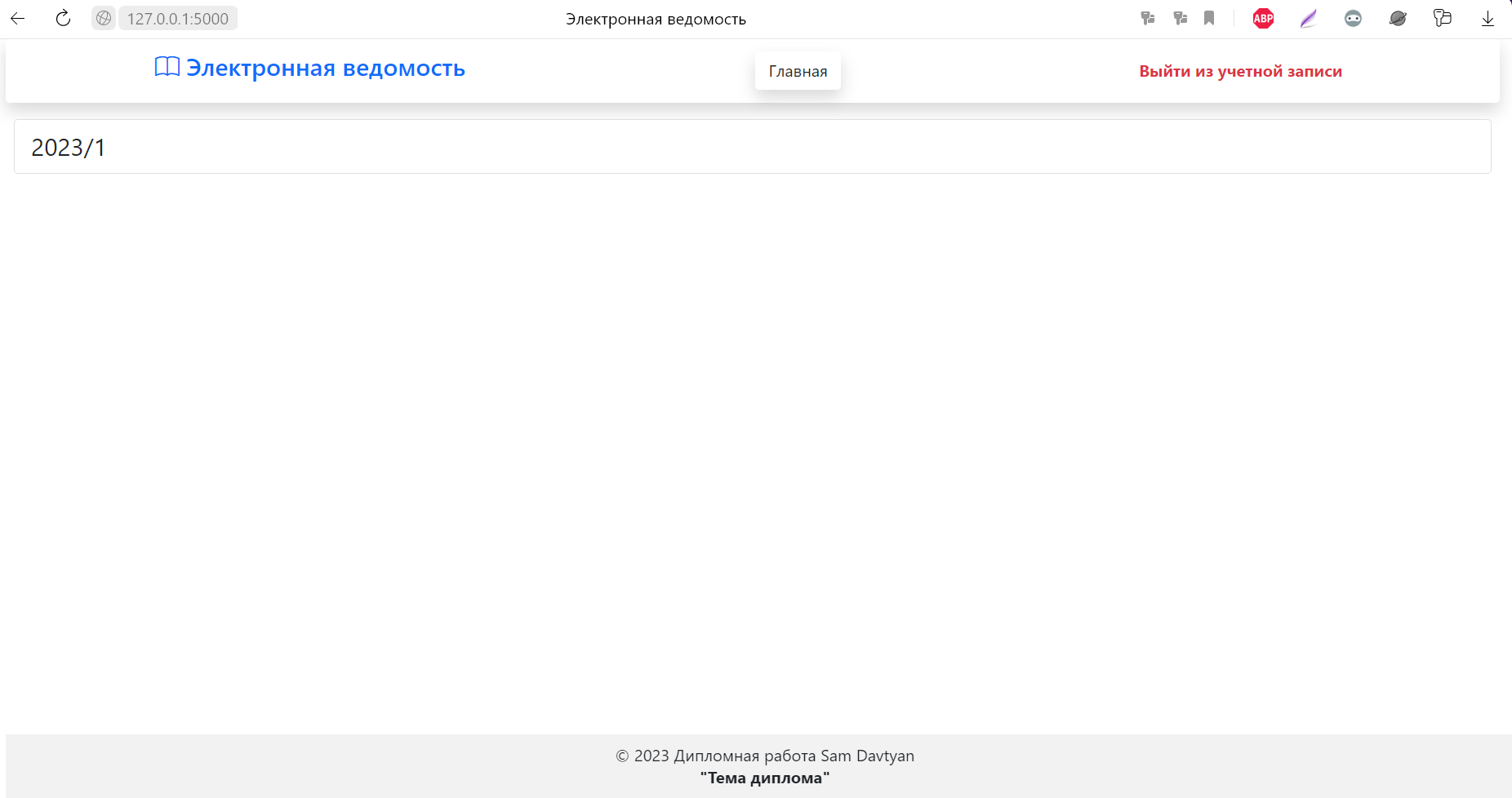


Рисунок 36 – экран студентов после предоставления им доступа к ведомости

После нажатия на ведомость, студент переходит на страницу с ведомостью, на которой указано:

* название дисциплины;
* номера групп, которым предоставлен доступ;
* ведомость в виде таблицы.

Авторизованный пользователь может добавить файлы программы и отчета к каждой контрольной точке в своей строчке. К строчкам, относящимся к другим студентам, доступа сейчас нет, в дальнейшем студенты смогут просматривать оценки друг друга.

На рисунке 37 представлена страница ведомости, которую видит студент.

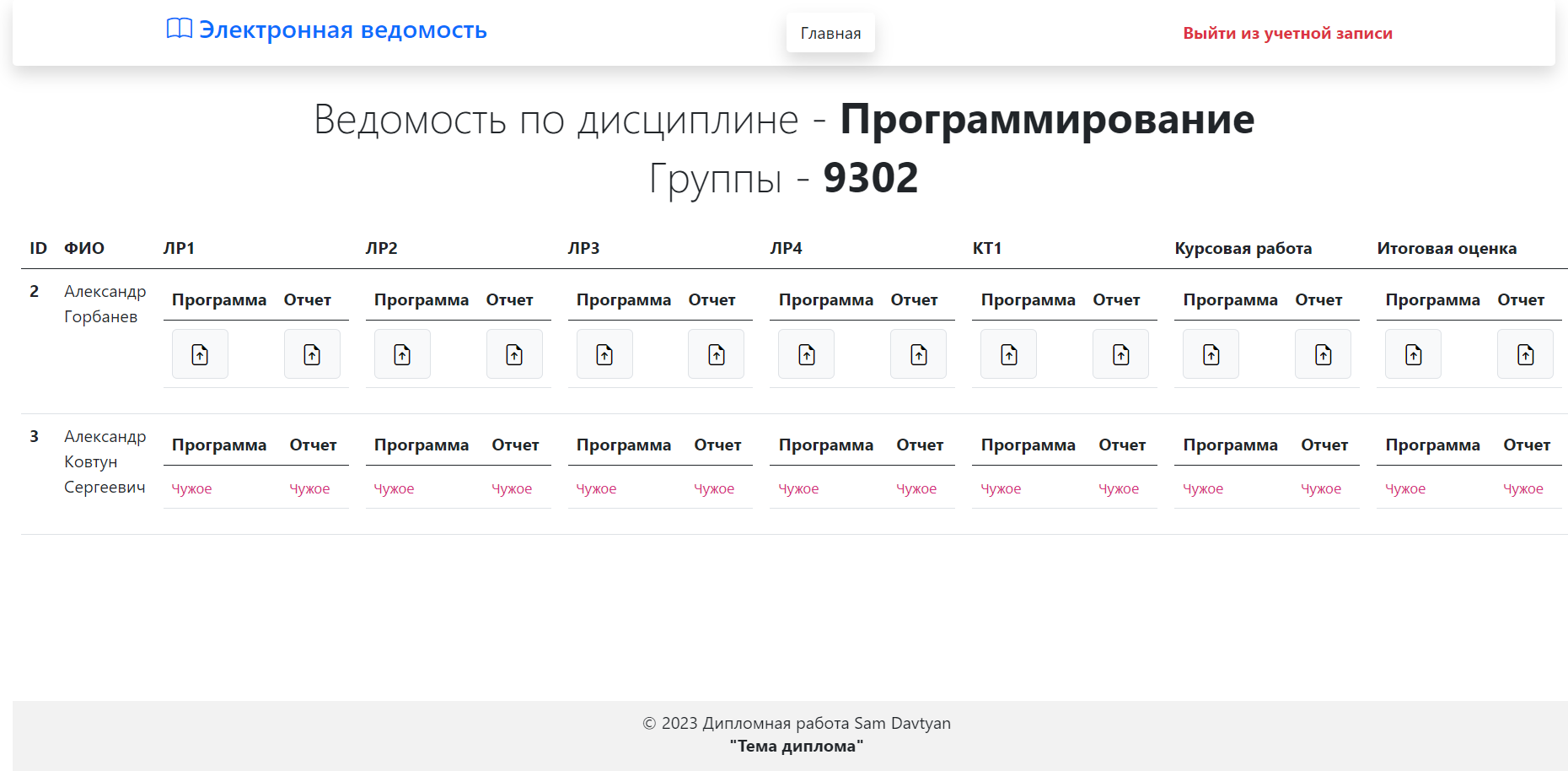


Рисунок 37 – страница ведомости для авторизованного пользователя

Для студентов под каждой ячейкой с программой и отчетом появляется иконка, нажав на которую, они могут загрузить файлы.

На рисунке 38 представлено окно, которое открывается после нажатия на кнопку.

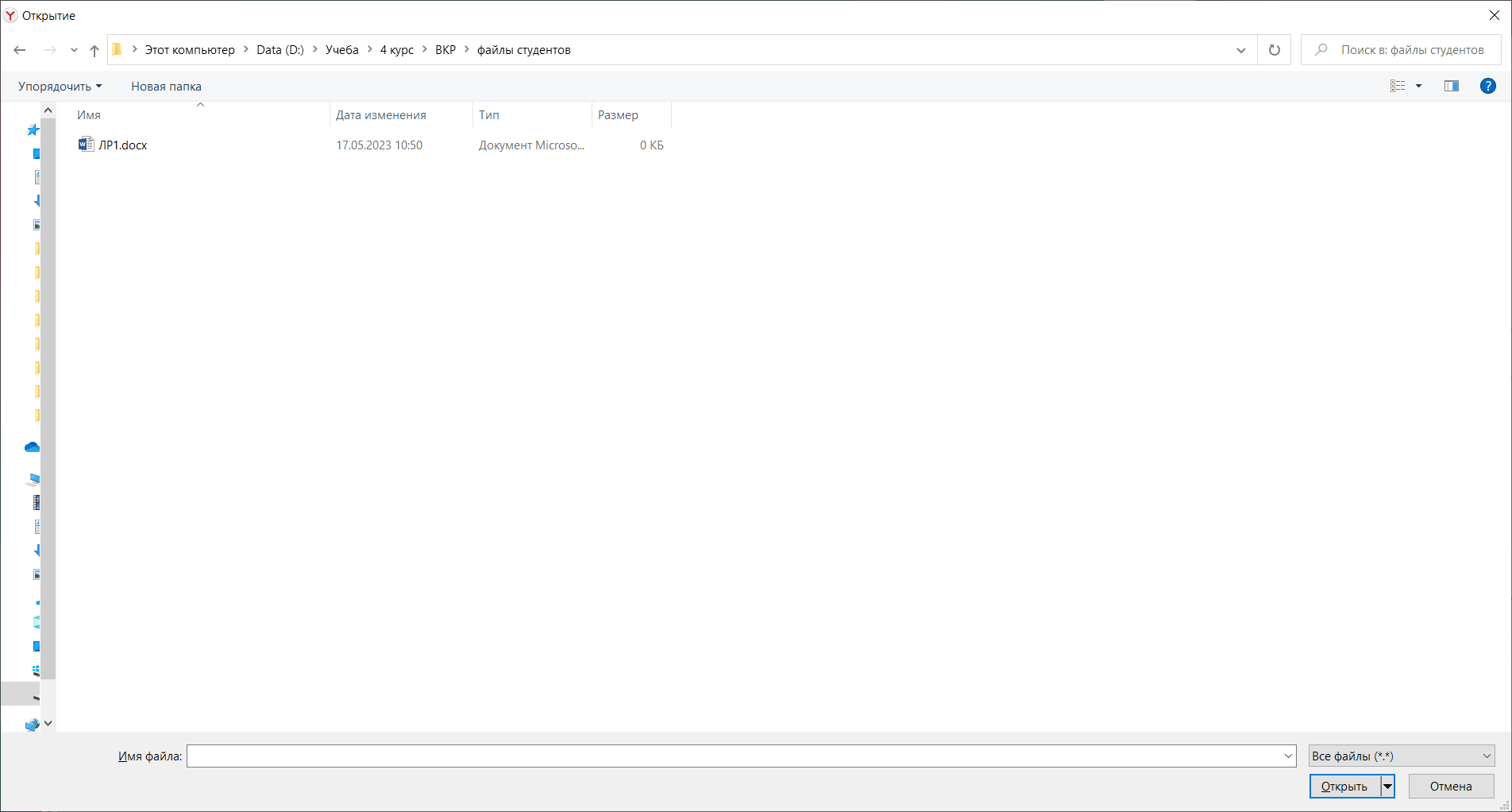


Рисунок 38 – окно с загрузкой файлов

После загрузки файла иконка подсвечивается желтым цветом, чтобы видно было, что работа ожидает проверки. В дальнейшем будет реализовано, что при проверке преподавателя и проставлении оценки, иконка будет подсвечиваться зеленым цветом, а при отправлении на доработку будет подсвечено красным.

На рисунке 39 представлена страница с ведомостью после загрузки файла студентом.

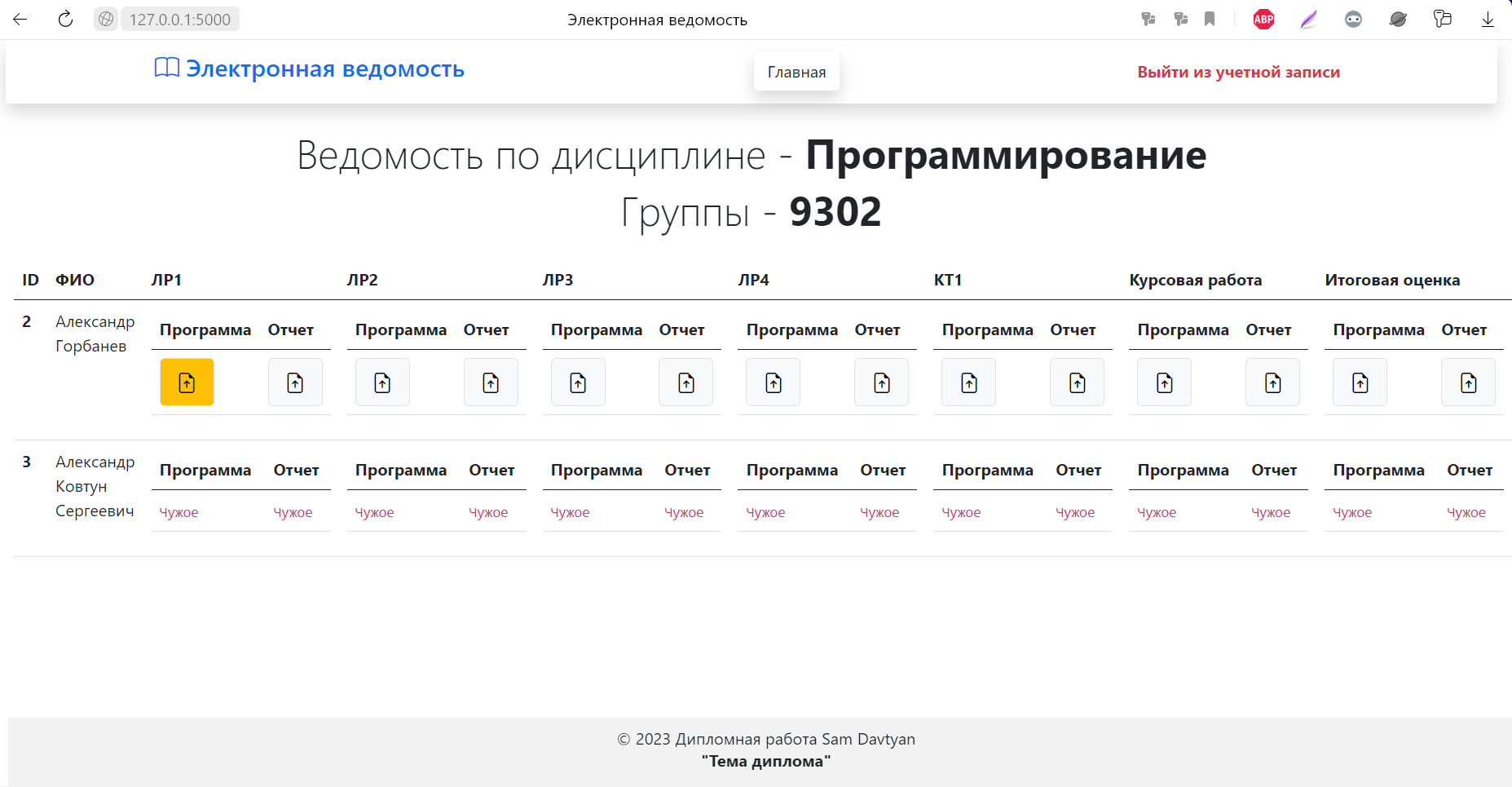


Рисунок 39 – страница ведомости после загрузки файла

После этого у преподавателя высветится, что данный студент загрузил работу. Также под иконкой загрузки файла появятся выпадающий список с возможными оценками и кнопка, нажав на которую, преподаватель может оставить комментарий. На рисунках 40, 41, 42 представлены выпадающий список с возможными оценками, кнопка для оставления комментария и сама страница, которую видит преподаватель после загрузки файла студентом.

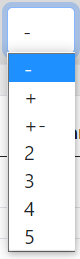


Рисунок 40 – выпадающий список с возможными оценками



Рисунок 41 – кнопка для оставления замечания

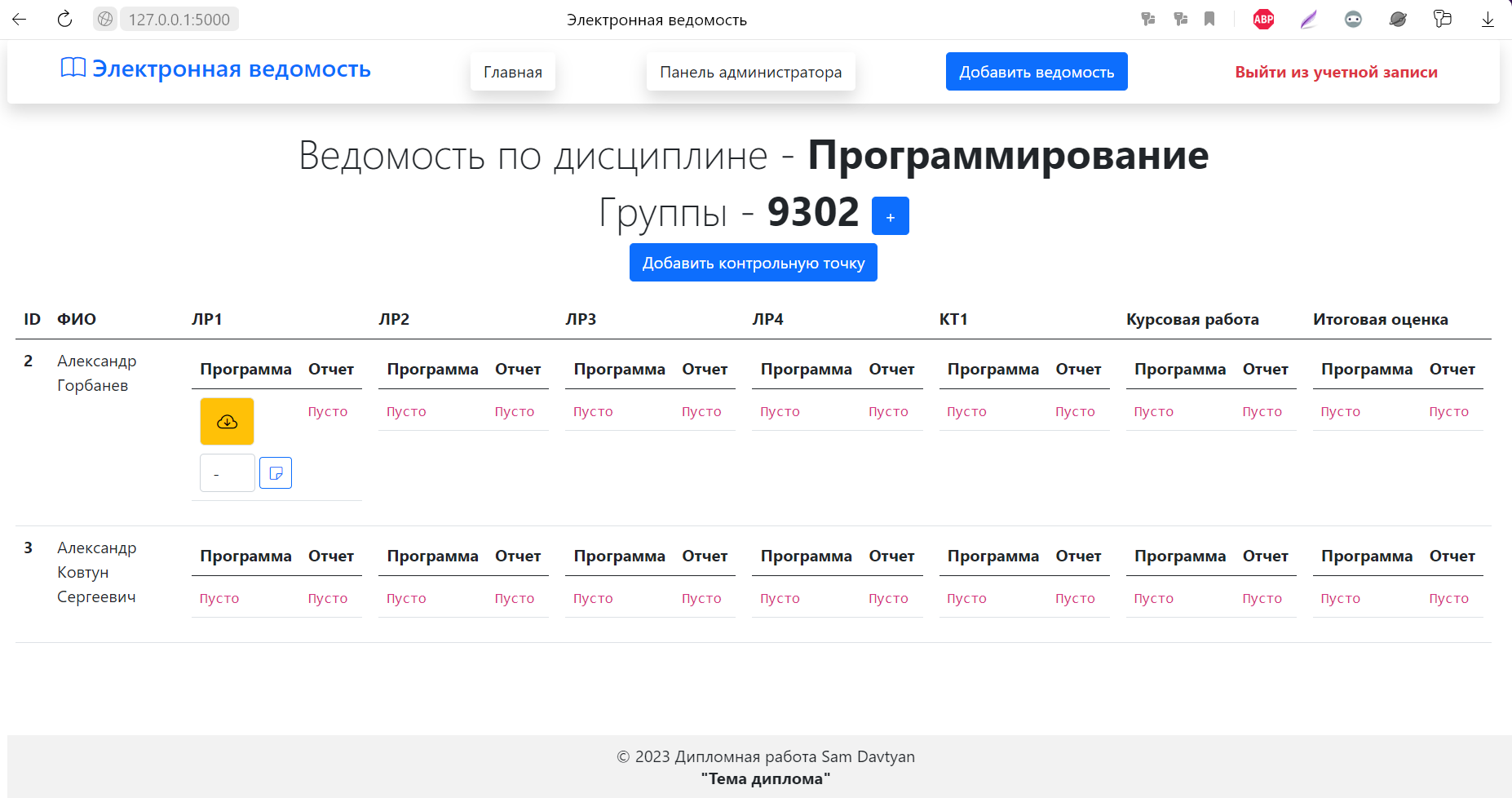


Рисунок 42 – страница ведомости у преподавателя после загрузки файла студентом

ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной квалификационной работы было создано веб-приложение для ведения ведомостей текущего контроля.

Главное ее отличие от аналогов состоит в интуитивно понятном интерфейсе, где в одном месте собрано как загрузка файлов студентами, так и обратная связь от преподавателя.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bootstrap5 // Build fast, responsive sites with Bootstrap. URL: <https://getbootstrap.com/> (дата обращения: 16.05.2023 г.).
2. HTML // Википедия. URL: <https://ru.wikipedia.org/wiki/HTML> (дата обращения: 13.05.2023 г.).
3. CSS // Википедия. URL: <https://ru.wikipedia.org/wiki/CSS>(дата обращения: 13.05.2023 г.).
4. JavaScript // Википедия. URL: <https://ru.wikipedia.org/wiki/JavaScript> (дата обращения: 13.05.2023 г.).
5. JQuery // jQuery. URL: <https://jquery.com/> (дата обращения: 16.05.2023 г.).
6. Jinja2 // Integration // Jinja Documentation. URL: <https://jinja.palletsprojects.com/en/3.1.x/integration/#flask> (дата обращения: 16.05.2023 г.).
7. Python // Welcome to Python. URL: <https://www.python.org/> (дата обращения: 16.05.2023 г.).
8. Flask // Welcome to Flask // Flask Documentation. URL: <https://flask.palletsprojects.com/> (дата обращения: 16.05.2023 г.).
9. Os // Miscellaneous operating system interfaces. URL: <https://docs.python.org/3/library/os.html?highlight=os#module-os> (дата обращения: 16.05.2023 г.).
10. SQLite // SQLite Home Page. URL: <https://www.sqlite.org/> (дата обращения: 16.05.2023 г.).
11. SQLAlchemy // The Python SQL Toolkit and Object Relational Mapper. URL: <https://www.sqlalchemy.org/> (дата обращения: 16.05.2023 г.).
12. VS Code // Code editing. Redefined. URL: <https://code.visualstudio.com/> (дата обращения: 16.05.2023 г.).
13. PyCharm // The Python IDE for Professional Developers. URL: <https://www.jetbrains.com/pycharm/> (дата обращения: 16.05.2023 г.).
14. ORM // Википедия. URL: <https://ru.wikipedia.org/wiki/ORM> (дата обращения 16.05.2023 г.).
15. ER-диаграмма // ER-модель данных // Википедия. URL: <http://wiki-org.ru/wiki/ER-модель_данных> (дата обращения 16.05.2023)