МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

# Разработка веб-приложения для отслеживания успеваемости учеников онлайн-школы

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-548.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.,
доцент
_____ А.В. Гренштейн

Автор работы,
студент группы КЭ-402
_____ Н.О. Макрушин

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет**
**(национальный исследовательский университет)»**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

**ЗАДАНИЕ**
**на выполнение выпускной квалификационной работы бакалавра**
студенту группы КЭ-402
Макрушину Никите Олеговичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)

Разработка веб-приложения для отслеживания успеваемости учеников онлайн-школы.

2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.

3. **Исходные данные к работе**

3.1. Дронов В.А. Django 4. Практика создания веб-сайтов на Python. // БХВ-Петербург, 2024. – 791 с.

3.2. Лутц М. Изучаем Python. // Компьютерное издательство «Диалектика», 2019. – 833 с.

3.3. Форсье Д., Биссекс П., Чан. У. Django. Разработка веб-приложений на Python. // Символ-Плюс, 2014. – 456 с.

4. **Перечень подлежащих разработке вопросов**

4.1. Провести анализ предметной области.

4.2. Изучить методы создания веб-приложения на веб-фреймворке Django.

4.3. Спроектировать и реализовать проект.

4.4. Провести тестирование системы.

**5. Дата выдачи задания:** 29.01.2024 г.

**Научный руководитель,**
доцент кафедры СП, к.ф.-м.н., доцент А.В. Гереншtейн

**Задание принял к исполнению** Н.О. Макрушин

**ОГЛАВЛЕНИЕ**

# ВВЕДЕНИЕ

## Актуальность

В современном контексте активного развития онлайн-образования веб-приложение для отслеживания успеваемости становится неотъемлемым инструментом для каждого преподавателя, особенно в онлайн-школах. Это приложение обеспечивает гибкость и эффективность образовательного процесса, превосходя простой ввод оценок.

Важным аспектом является не только упрощение обучения, но и предоставление преподавателям инструментов для индивидуализации подходов в соответствии с уникальными потребностями каждого ученика. Внедрение персонализированной системы помогает преподавателям лучше понимать индивидуальные предпочтения учеников, способствуя развитию их максимального потенциала.

Подчеркивается, что аналитика и отчетность, предоставляемые приложением, не только выявляют общие тенденции, но и предоставляют детальную информацию по каждому ученику. Это обеспечивает прозрачность в образовательном процессе и служит основой для постоянного улучшения методик преподавания.

Особое внимание уделяется тому, что веб-приложение становится ключевым инструментом взаимодействия для родителей, позволяя им отслеживать успехи и трудности своих детей. Это создает не только возможность для поддержки в учебе, но и способствует более глубокому взаимодействию с преподавателями.

В итоге веб-приложение для отслеживания успеваемости учащихся играет ключевую роль в установлении тесного и выгодного взаимодействия между преподавателями, учащимися и их родителями.

## Постановка задачи

Целью данной работы является создание веб-приложения для отслеживания успеваемости учащихся онлайн-школы:

1) проанализировать предметную область;

2) провести анализ аналогичных проектов;
3) спроектировать веб-приложение;
4) реализовать веб-приложение;
5) протестировать веб-приложение.

**Структура и содержание работы**

Работа состоит из введения, четырех глав, заключения и списка литературы. Объем работы составляет 62 страницы, объем списка литературы – 17 источников.

В первом разделе «Анализ предметной области» описываются ключевые функции системы, такие как ведение журнала, управление базой данных и сбор статистики. Также проведен анализ аналогичных образовательных платформ, выявлены их основные возможности и сравнительные преимущества, что позволило определить важные критерии для разработки собственного веб-приложения.

Во втором разделе «Требования к системе» рассматриваются функциональные и нефункциональные требования к создаваемой системе для ведения журнала отчетов о занятиях с учениками. Описаны ключевые функциональные возможности, такие как регистрация пользователей, редактирование данных об учениках и предметах, создание отчетов, а также важные нефункциональные требования, включая удобство использования, масштабируемость и совместимость системы. Представлена диаграмма вариантов использования, иллюстрирующая взаимодействие пользователей с системой, что позволяет более четко понять функциональные аспекты системы и их соответствие потребностям пользователей.

В третьей главе «Архитектура системы» рассмотрен выбор и анализ архитектуры для веб-приложения. Описан выбор фреймворка Django благодаря его преимуществам, таким как мощная система ORM, встроенная административная панель и принцип MVT. Представлена архитектура приложения, включающая модели, представления и шаблоны, а также диаграммы последовательности и деятельности, иллюстрирующие взаимодействие

компонентов системы и упрощающие процесс разработки и дальнейшего развития проекта.

В четвертой главе «Реализация» описываются основные средства разработки, внедрение и настройка СУБД PostgreSQL в веб-приложение. Описана реализация backend части, в которой рассматриваются основные методы представления страниц, а также дополнительный метод авторизации пользователя. Реализация frontend части, в которой описывается создание одного из шаблонов страниц и представления страниц для смартфона и планшета.

В пятой главе «Тестирование системы» подробно описывается функциональное тестирование системы, в котором проводятся тесты на корректность работы функционала системы, а тестирование адаптивной верстки проверяет корректное отображение и работоспособность функционала веб-приложения на различных устройствах.

В приложении А представлены таблицы со спецификациями вариантов использования веб-приложения.

В приложении Б расположены скриншоты аналогичных проектов работы.

В приложении В расположена таблица сущностей базы данных.

В приложении Г представлена исходный код реализации веб-приложения.

В приложении Д представлена схема базы данных.

В приложении Е представлена страница «Расписание».

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Предметная область проекта

Основная задача заключается в создании веб-приложения для заполнения журнала отчетов о проведенных занятиях с учениками и указанием тем занятий по предметам, и сбора статистики по каждому учащемуся.

Веб-приложение предназначено для упрощения процесса ведения учета занятий, обеспечивая учителей средствами для создания отчетов и отслеживания прогресса учащихся. Ниже представлен список ключевых возможностей реализуемого проекта [14].

1. Важная часть предметной области – журнал отчетов, который служит для фиксации проведенных занятий. В нем регистрируются данные о темах занятий, дате проведения и учащихся, участвовавших в занятии.

2. База данных является ключевым компонентом системы. Она хранит информацию об учениках, предметах, темах предметов и отчетах.

3. Система включает в себя данные об учениках, включая их персональные данные. Каждый ученик имеет уникальный идентификатор, который связывает учащегося с проведенными занятиями и отчетами.

4. Информация о предметах и темах предметов используется для классификации проведенных занятий. Учитель может указать тему занятия, а также связать ее с конкретным предметом.

5. Важным аспектом предметной области является сбор статистики по каждому ученику. Это может включать в себя прогресс в обучении, количество посещенных занятий, успеваемость и другие показатели.

6. Для удобства преподавателей предусмотрен интерфейс для заполнения отчетов. Учителя могут выбирать учеников, указывать темы занятий, даты проведения и вносить другую необходимую информацию.

7. Все компоненты предметной области взаимодействуют между собой. Преподаватели используют интерфейс для внесения данных, которые затем сохраняются в базе данных.

## 1.2. Анализ аналогичных проектов

В данном разделе будет проведен анализ аналогичных проектов, предназначенных для отслеживания успеваемости учащихся преподавателями. Будут исследованы основные характеристики и функциональные возможности популярных образовательных платформ и инструментов, направленных на улучшение взаимодействия между учителями, учениками и их родителями. Анализ существующих проектов поможет выявить лучшие практики, идеи и возможности для оптимизации процесса обучения.

**Веб-приложение «Microsoft Teams for Education»**

Данный продукт был создан компанией Microsoft и является частью платформы Microsoft Teams, специально адаптированной для образовательных учреждений, учителей и учеников. Она предоставляет широкий спектр инструментов и функциональности для организации виртуального обучения и взаимодействия в учебной среде. На рисунке 1 приложения А представлен скриншот веб-приложения «Microsoft Teams for Education».

Основные возможности данного продукта:

- создание классов и групп;
- обмен материалами и документами;
- интерактивные инструменты для обучения;
- интеграция с другими сервисами Microsoft.

**Веб-приложение «Google Classroom»**

Образовательная платформа от Google, созданная для облегчения ведения учебного процесса в виртуальной среде. Основная цель «Google Classroom» – предоставить учителям и ученикам инструменты для эффективного обучения, обмена материалами и совместной онлайн работы. На рисунке 2 приложения А предоставлен скриншот веб-приложения «Google Classroom».

Основные возможности данного продукта:

– задание и оценки;
– календарь;
– обмен материалами и документами;
– интеграция с другими сервисами Google.

**Веб-приложение «Schoology»**

Образовательная платформа, которая предоставляет учителям, ученикам и администраторам инструменты для удобного и эффективного управления учебным процессом и взаимодействия в виртуальной среде. На рисунке 3 приложения А находится скриншот веб-приложения «Schoology».

Основные возможности данного продукта:

– создание виртуальных классов;
– календарь и расписание;
– оценки и аналитика;
– задания и оценивание;
– интеграция с другими приложениями (Microsoft, Google).

**Веб-приложение «Moodle»**

Свободная и открытая образовательная платформа управления курсами (LMS, Learning Management System), предназначенная для создания онлайн-курсов и виртуальных обучающихся сообществ. Она разработана для облегчения процесса обучения и предоставления удобных инструментов для создания интерактивных образовательных курсов. На рисунке 4 приложения А предоставлен скриншот веб-приложения «Moodle».

Основные возможности данного продукта:

– оценки и отслеживание процесса;
– блоги и портфолио;
– система уведомлений.

Сравнение аналогичных проектов по основным критерия представлено в таблице 1.

Таблица 1 – Обзор аналогов

| Критерии | Microsoft Teams for Education | Google Classroom | Schoology | Moodle |
|---|---|---|---|---|
| Настройка и администрирования веб-приложения | – | + | – | – |
| Календарь занятий и событий | + | + | + | + |
| Возможность оформления подписки в РФ | + | + | – | + |
| Формирование отчета по ученику | – | – | + | + |

Ниже представлены выводы по таблице 1.

1. Настройка и администрирование веб-приложения. «Google Classroom» проста в настройке и администрировании. «Microsoft Teams for Education», «Schoology», и «Moodle» требуют больше времени и усилий для первоначальной настройки и управления.

2. Календарь занятий и событий. Все рассмотренные платформы предоставляют удобные инструменты для планирования и организации учебного процесса.

3. Возможность оформления подписки в РФ. Приложения «Microsoft Teams for Education», «Google Classroom», и «Moodle» доступны в России. «Schoology» сталкивается с трудностями при оформлении подписки.

4. Формирование отчета по ученику. «Schoology» и «Moodle» предоставляют расширенные инструменты для мониторинга и анализа успеваемости учащихся. «Microsoft Teams for Education» и «Google Classroom» имеют ограниченные возможности в этой области.

**Вывод по первой главе**

В первой главе был проведен анализ предметной области и выделили ключевые возможности проекта. Также были подробно рассмотрены аналогичные проекты, выявлены важные критерии для реализации веб-приложения. Простоту настройки и администрирования, удобство календаря для планирования занятий, доступность в России и необходимость расширенных инструментов для мониторинга успеваемости учеников.

# 2. ТРЕБОВАНИЯ К СИСТЕМЕ

## 2.1. Анализ требований к программной системе

На основе информации, полученной в результате изучения предметной области и обзора приложений-аналогов, были сформированы следующие функциональные и нефункциональные требования к приложению [14].

Функциональные требования к проектируемой системе:

1) регистрация и аутентификация [10]:

– возможность регистрации новых пользователей с указанием логина, адреса электронной почты и пароля;

– вход в приложение с использованием учетных данных пользователя;

2) редактирование учащихся:

– добавление новых учеников в базу данных веб-приложения, с полным указанием всех требуемых данных для профиля учащегося;

– возможность редактирования данных об ученике;

– удаление ученика из базы данных;

3) редактирование предметов:

– добавление нового предмета в базу данных;

– возможность удаления предмета;

4) редактирование тем по каждому предмету:

– добавление новой темы с прикреплением к выбранному предмету, а также заполнение требуемой информации по создаваемой теме;

– изменение требуемых данных по выбранной теме;

5) создание отчетов по проведенному занятию [3]:

– выбор ученика из базы данных;

– выбор предмета и темы занятия;

– добавление домашнего задания к отчету по занятию;

– проставление даты и времени проведенного занятия.

Нефункциональные требования к проектируемой системе:

1) удобство использования:

– интерфейс приложения должен быть максимально простым и понятным, чтобы не перегружать пользователей [11];

– должны быть предусмотрены элементы навигации и подсказки, чтобы облегчить использование системы;

2) безопасность:

– все пользовательские данные должны быть защищены от несанкционированного доступа [10];

– должна быть реализована аутентификация и авторизация пользователей;

– все данные пользователей должны храниться и передаваться в зашифрованном виде с использованием протокола HTTPS [1];

3) масштабируемость:

– веб-приложение должно быть способным масштабироваться для работы с растущим объемом данных и количеством пользователей [4];

– веб-приложение должно быть готово к масштабированию для обработки увеличивающегося числа пользователей и данных;

– должны быть предусмотрены механизмы горизонтального и вертикального масштабирования [4];

4) совместимость:

– веб-приложение должно корректно отображаться в различных браузерах (Chrome, Firefox, Safari, Edge) [3];

– интерфейс веб-приложения должен быть адаптирован под различные форматы устройств (персональные компьютеры, планшеты, смартфоны) [10].

## 2.2. Диаграмма вариантов использования

Для моделирования ранее выделенных функциональных требований к приложению с использованием языка объектного моделирования UML [13]

была создана диаграмма вариантов использования. Эта диаграмма отражает отношения между актерами и прецедентами, представляя собой визуализацию процесса создания веб-приложения и взаимодействия пользователя с основным функционалом проекта.

На рисунке 1 представлена диаграмма вариантов использования, которая иллюстрирует сценарии взаимодействия между пользователями и системой. Графическое представление включает ключевых актеров, таких как пользователь, а также различные варианты использования, представленные прецедентами.

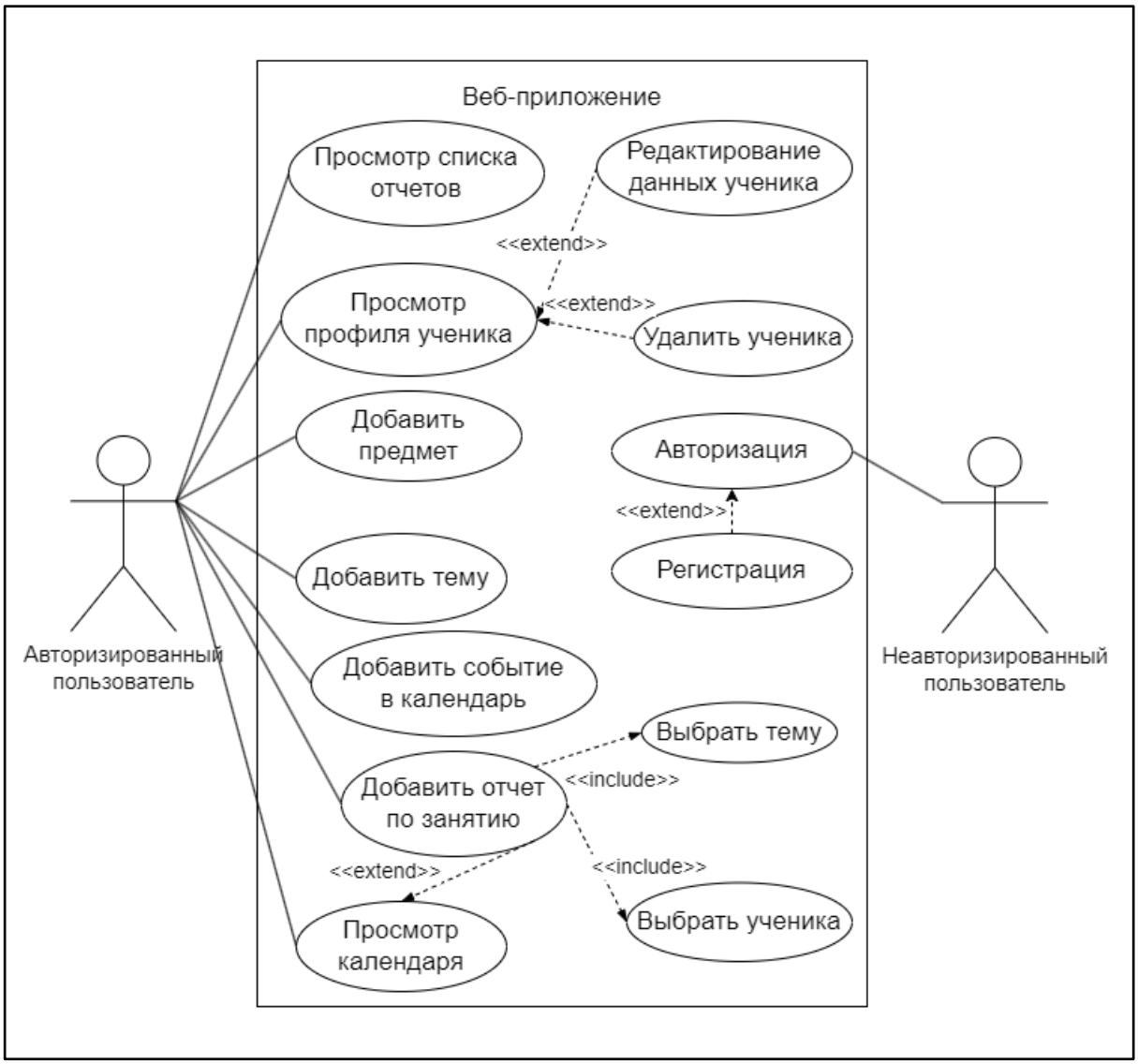


Рисунок 1 – Диаграмма вариантов использования

Эта диаграмма вариантов использования также играет ключевую роль в обеспечении высокой функциональности и удобства пользования разрабатываемым веб-приложением. Путем анализа сценариев взаимодействия пользователей с системой можно выявить потенциальные улучшения и оптимизации, что способствует повышению эффективности и удовлетворенности конечных пользователей. Кроме того, эта диаграмма служит основой для дальнейшего проектирования архитектуры приложения, обеспечивая более глубокое понимание его структуры и взаимосвязей между компонентами.

Также эта диаграмма помогает выявить потенциальные проблемные моменты в процессе взаимодействия пользователя с системой и предоставляет возможность предварительно определить необходимые функции и условия их использования. Кроме того, она служит основой для дальнейшего проектирования архитектуры приложения и разработки его пользовательского интерфейса.

В итоге диаграмма вариантов использования является важным инструментом для обеспечения успешной разработки веб-приложения, которое будет удовлетворять потребностям пользователей.

### 2.3. Описание актеров и их возможностей в системе

В данной системе актерами является неавторизированный пользователь, который может авторизироваться в системе, или зарегистрироваться, если ранее этого не было сделано. Авторизированный пользователь обладает следующим рядом функциональных возможностей.

1. Добавить предмет. Преподаватель может добавить новый предмет в систему. Это может быть, например, название курса, дисциплины или учебного модуля.

2. Добавить тему. После добавления предмета преподаватель может добавлять темы занятий, связанные с выбранным предметом. Это может включать в себя различные темы, разделы или учебные единицы.

3. Добавить отчет по занятию. Преподаватель имеет возможность создавать отчеты по каждому проведенному уроку. При создании отчета выбирается предмет, тему занятия, ученика, а также указывает дату проведения занятия.

4. Просмотр списка отчетов. Преподаватель может просматривать общий список всех отчетов, созданных им по проведенным занятиям.

5. Просмотр профиля ученика. Преподаватель может зайти на страницу профиля ученика, и посмотреть полную информацию о нем, список отчетов по всех проведенным занятиям с учеником, отредактировать какие-то данные ученика или удалить его.

6. Добавить событие в календарь. Пользователь может добавить какое-то событие в календарь (например, преподаватель с 14:00 до 16:00 будет писать тестовое задание по математике), или добавить в календарь занятие с учеником.

7. Просмотр календаря. Пользователь может открыть календарь и посмотреть все события и занятия в календаре на текущий месяц.

В приложении А представлены спецификации основных вариантов использования системы.

**Вывод по второй главе**

В данной главе был проведен детальный анализ функциональных требований к системе. На основе выявленных требований разработана диаграмма вариантов использования, которая иллюстрирует взаимодействие актеров с рассматриваемой системой. Кроме того, в тексте представлено краткое описание данной диаграммы вариантов использования и спецификация отдельных вариантов использования. Это позволяет более четко понять функциональные аспекты системы и определить, как они соотносятся с потребностями пользователей.

# 3. АРХИТЕКТУРА СИСТЕМЫ

## 3.1. Выбор архитектуры системы

В контексте создания веб-приложения для отслеживания успеваемости учащихся онлайн-школы, выбор правильных инструментов и фреймворков играет решающую роль в успешной реализации проекта. На сегодняшний день существует множество технологий, предоставляющих различные инструменты для создания мощных и эффективных веб-приложений. Несколько самых популярных веб-фреймворков представлены ниже.

1. Фреймворк Django. Django является высокоуровневым веб-фреймворком, написанным на языке Python, который предназначен для упрощения процесса создания веб-приложений. Принцип работы Django основывается на модели MVC (Model-View-Controller) или на его вариации, называемой моделью MTV (Model-Template-View). Фреймворк Django предоставляет множество готовых инструментов и функций, что позволяет разработчикам сосредотачиваться на бизнес-логике приложения, а не на рутинных задачах, таких как взаимодействие с базой данных, обработка HTTP-запросов, аутентификация пользователей и другие.

2. Фреймворк Spring. Spring является одним из наиболее популярных фреймворков для создания веб-приложений на языке Java. Данный фреймворк предоставляет обширный функционал, включая инверсию управления, внедрение зависимостей, а также модульные и аспектно-ориентированные функции. Spring имеет множество подпроектов, таких как Spring Boot для быстрой разработки и Spring MVC для построения веб-приложений.

3. Фреймворк Flask. Flask – легковесный фреймворк для веб-разработки на языке Python. Данный фреймворк предоставляет минималистичный набор инструментов, что делает его отличным выбором для небольших и средних проектов. Flask поддерживает расширения, что позволяет разработчикам добавлять функциональность по мере необходимости.

4. Фреймворк ASP.NET. Язык программирования C#. Фреймворк предназначен для веб-разработки, созданный Microsoft, использующий

язык программирования C#. Данный фреймворк поддерживает различные парадигмы программирования, включая веб-формы, MVC (Model-View-Controller), и в последнее время стал более модульным и гибким с появлением ASP.NET Core.

Для реализации данного проекта был выбран фреймворк Django и этот выбор обоснован несколькими ключевыми преимуществами.

1. Мощная система ORM Django значительно упрощает работу с базой данных, что важно для эффективного хранения данных об успеваемости учащихся.

2. Встроенная административная панель Django предоставляет удобный интерфейс для управления данными, сокращая время на администрирование. Это особенно важно для оперативного вмешательства в управление учебным процессом.

3. Принципы MVT способствуют созданию чистого и удобного в обслуживании кода, что облегчает разработку и поддержание приложения.

Django является оптимальным выбором для образовательных проектов, где структурированный и эффективный код играет ключевую роль в долгосрочной устойчивости системы.

### 3.2. Архитектура веб-приложения

В архитектуре MVT (Model-View-Template) [17] Django используется разделение ответственностей между компонентами для эффективной организации кода и обработки запросов. Схема архитектуры представлена на рисунке 2. Каждый из этих компонентов рассмотрен более подробно ниже.

1. Модель (Model). Этот компонент определяет структуру базы данных и содержит логику доступа к данным, описывает объекты данных (например, таблицы в базе данных) и их отношения между собой. Модель в Django используется для выполнения операций с данными, таких как создание, чтение, обновление и удаление (CRUD).

2. Представление (View). Представление обрабатывает запросы пользователя и взаимодействует с моделью для получения данных. Оно содержит логику, определяющую, как данные должны быть обработаны перед передачей их в шаблон. Представление может выполнять дополнительные действия, такие как проверка прав доступа, обработка данных формы и формирование ответа на запрос.

3. Шаблон (Template). Этот компонент содержит HTML-код с встроенными тегами и переменными, предназначенными для отображения данных. Шаблон используется для формирования пользовательского интерфейса и отображения данных. Позволяет разработчику создавать динамические веб-страницы, которые могут изменяться в зависимости от данных из модели и логики из представления.

Схема взаимодействия MVT представлена на рисунке 2.

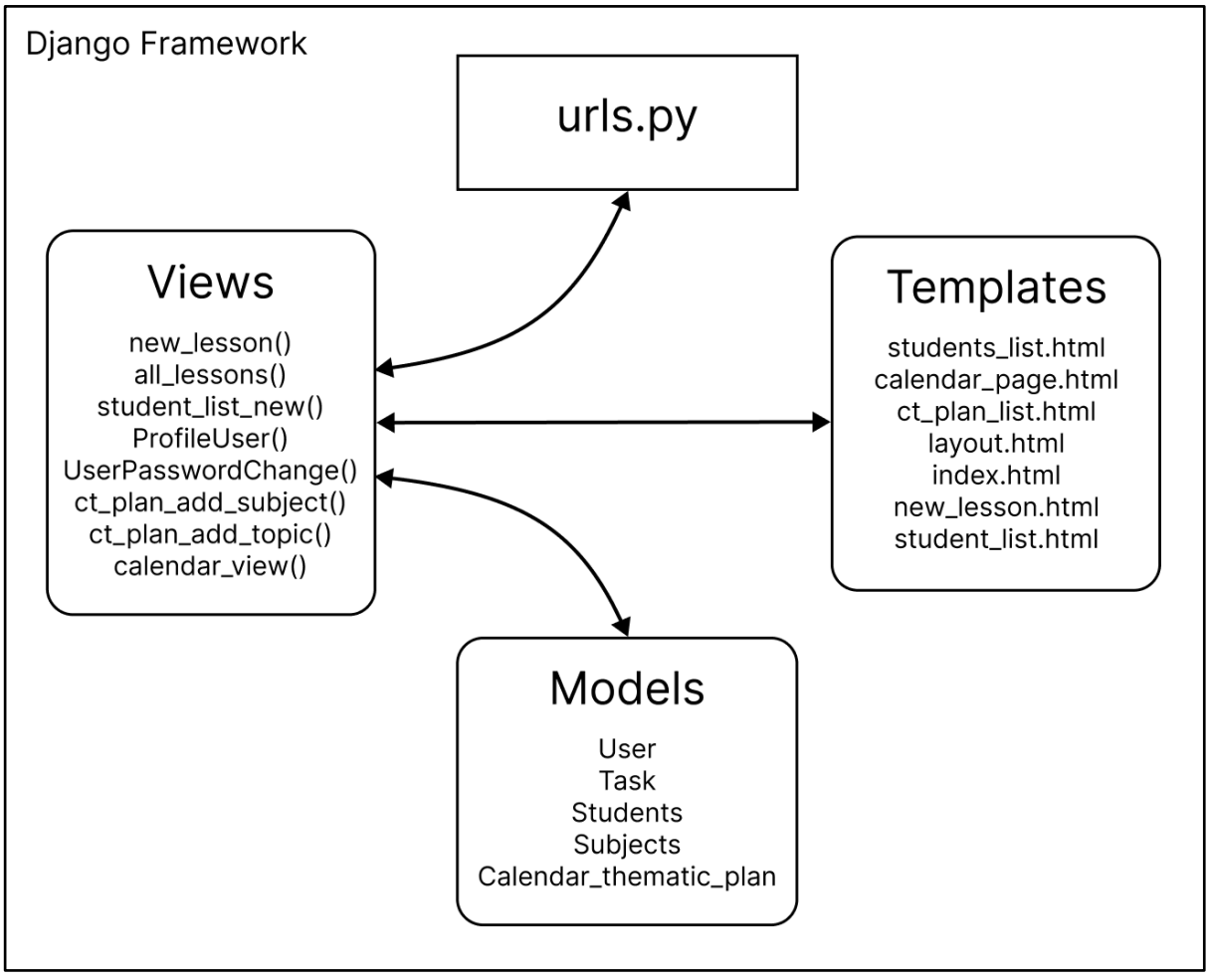


Рисунок 2 – Схема взаимодействия MVT

На практике при создании веб-приложения разработчики определяют модели для представления данных и их структуры в базе. Представления обрабатывают запросы пользователя, обращаясь к моделям для доступа к данным. Полученные данные затем передаются в шаблоны, где они форматируются и отображаются в виде HTML-страниц для пользователя. URL-маршруты определяют связь между URL-адресами и представлениями, которые должны обработать соответствующие запросы.

Ниже описаны реальные примеры каждого из компонентов системы.

1. Модель `Task` содержит записи событий и занятий для отображения на странице «Расписание» в календаре.

2. Шаблон «calendar_page.html» является страницей, для представления календаря с занятиями и событиями на месяц, в котором с помощью шаблонизатора Jinja происходит вывод данных на страницу.

3. Представление `calendar_view`, является методом, который взаимодействует с моделью `Task`, получая все события календаря, доступные для конкретного пользователя системы на текущий месяц, а также передает эти данные в шаблон страницы, который также определяется в этом методе.

4. Адресация страниц прописывается в файлах «urls.py», где определены пути для доступа к различным страницам веб-приложения. Этот файл содержит соответствие URL-запросов к представлениям, что обеспечивает правильную маршрутизацию запросов пользователя к нужным методам и шаблонам.

Такая архитектура позволяет легко поддерживать и изменять веб-приложение, так как каждый компонент отвечает за конкретный аспект приложения, что делает код более структурированным и понятным, а также обеспечивает разделение ответственности между компонентами, что способствует чистоте кода и облегчает разработку и поддержку веб-приложений.

### 3.3. Диаграмма последовательности

Диаграмма последовательности – это графическое представление последовательности взаимодействия между объектами или компонентами системы в определенном сценарии выполнения, с использованием языка объектного моделирования UML [15]. Она позволяет визуализировать порядок вызова операций и передачу сообщений между объектами, что упрощает понимание логики работы системы и анализ ее поведения.

Пример взаимодействия пользователя и системы на диаграмме последовательности для варианта использования «Добавить предмет» представлен на рисунке 3.

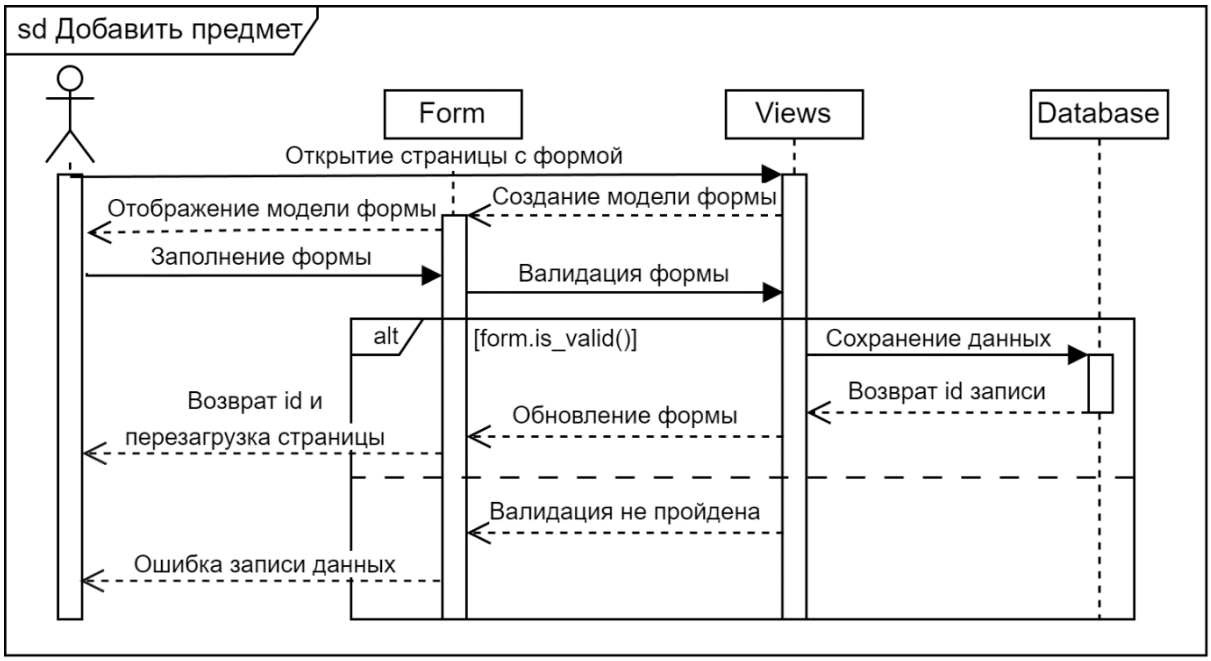


Рисунок 3 – Диаграмма последовательности «Добавить предмет»

Выполнение диаграммы последовательности начинается с того момента, когда пользователь открывает страницу с формой для добавления нового предмета. Это действие инициирует процесс взаимодействия пользователя с системой, в котором ключевую роль играет представление, содержащееся в файле `views`. Оно отвечает за обработку загрузки страницы и загрузку соответствующей формы на страницу, готовой для заполнения [8].

После получения формы пользователь может приступить к заполнению полей и по завершению, нажать кнопку «Отправить». Данные, введенные пользователем, передаются в форму для обработки. Затем форма отправляет запрос модулю `views`, который производит серверную валидацию данных. Этот этап включает в себя проверку допустимости введенной информации, например, ее соответствие определенным правилам или форматам [5].

Если данные прошли серверную валидацию, они сохраняются в базе данных и views получает ответ от сервера в виде идентификатора записи. Затем представление передает запрос форме для обновления ее состояния и происходит перезагрузка формы. Этот шаг важен для того, чтобы пользователь получил обратную связь о том, что его данные успешно сохранены.

В случае, если данные не прошли валидацию, `views` передает пользователю сообщение об ошибке и форма остается открытой с предварительно введенными данными. Это позволяет пользователю легко исправить ошибки и повторно отправить форму для обработки. Таким образом, диаграмма последовательности позволяет наглядно представить весь процесс взаимодействия пользователя с системой при добавлении нового предмета.

### 3.4. Диаграмма деятельности

Диаграмма деятельности – это графическое представление динамического поведения системы, отображающее поток управления и данные между различными действиями или состояниями. Используя язык объектного моделирования UML, диаграммы деятельности помогают визуализировать последовательность шагов и операций, выполняемых системой для достижения определенной цели.

Пример диаграммы деятельности для сценария «Добавить тему», демонстрирующей последовательность действий, выполняемых пользователем, системой и базы данных представлен на рисунке 4.

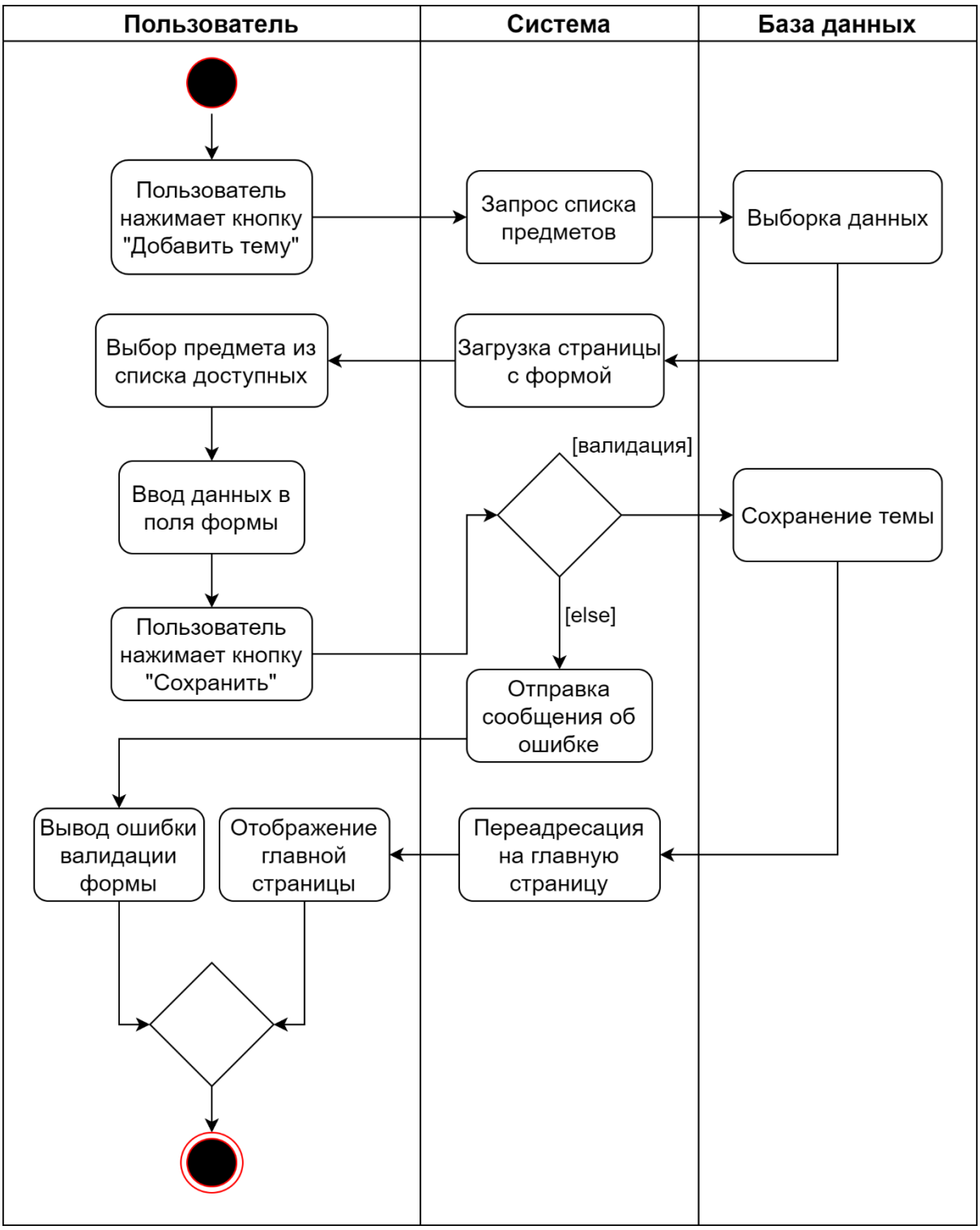


Рисунок 4 – Диаграмма деятельности «Добавить тему»

Процесс начинается с того, что пользователь нажимает кнопку «Добавить тему». Затем система отправляет запрос к базе данных для получения списка доступных предметов, который создал именно этот пользователь, на

что база данных обрабатывает запрос и возвращает список предметов. После этого система отображает страницу с формой для добавления новой темы, включая выпадающий список с предметами.

Пользователь выбирает предмет из предоставленного списка и заполняет необходимые поля формы для добавления темы. Затем пользователь нажимает кнопку «Сохранить», и система проверяет введенные данные на корректность. На этом этапе происходит ветвление, если валидация пройдена, система сохраняет новую тему в базе данных и перенаправляет пользователя на главную страницу. Если валидация не пройдена, система отправляет сообщение об ошибке валидации и пользователь видит это сообщение. Процесс завершается слиянием ветвей и завершением диаграммы деятельности.

**Вывод по третьей главе**

На основе выбранного фреймворка Django для разработки веб-приложения для отслеживания успеваемости учеников онлайн-школы была разработана и обоснована его архитектура, основанная на модели MVT. Django, был выбран благодаря ряду преимуществ, которые существенно упрощают разработку и поддержку приложения.

Прежде всего, мощная система ORM Django значительно упрощает взаимодействие с базой данных, встроенная административная панель Django обеспечивает удобный интерфейс для управления данными, а также принципы MVT способствуют созданию легко поддерживаемого кода, что немало важно для создания нового функционала проекта.

С учетом выбранного шаблона системы, была разработана структура, включающая диаграммы последовательности и деятельности. Эти элементы позволяют наглядно представить организацию приложения, улучшить понимание взаимодействия его составляющих и упростить процесс разработки. Созданная структура также помогает определить логику и поведение системы, что важно для ее дальнейшего развития и поддержки.

# 4. РЕАЛИЗАЦИЯ

## 4.1. Средства инструментов разработки

В контексте создания веб-приложения для отслеживания успеваемости учащихся онлайн-школы, выбор правильных инструментов и фреймворков играет решающую роль в успешной реализации проекта. На сегодняшний день существует множество технологий, предоставляющих различные инструменты для создания мощных и эффективных веб-приложений.

В ходе анализа предметной области был сделан выбор в сторону веб-фреймворка Django. Для разработки веб-приложения и написания кода проекта используется IDE PyCharm Community [16] для языка программирования Python. Преимущества данного IDE следующие:

1) большое количество инструментов разработки;
2) поддержка управления проектами с помощью git;
3) поддержка различных фреймворков и плагинов.

## 4.2. Реализация базы данных

Для реализации базы данных в фреймворке Django базово используется встроенная SQLite3. Данная база данных является легковесной и не требует настройки, но есть и огромный недостаток – это производительность. При большом потоке данных и взаимодействии большого количества пользователей с веб-приложением база данных может не справиться.

Поэтому для реализации базы данных был выбран PostgreSQL. Данная СУБД является мощной, масштабируемой и функционально богатой реляционной системой управления базой данных, которая используется во многих крупных и нагруженных проектах. В отличие от SQLite3, PostgreSQL поддерживает сложные запросы, параллелизм и масштабируемость, что делает ее идеальной для веб-приложений, работающих с большими объемами данных.

Для внедрения выбранной СУБД в систему, была использована библиотека «psycopg2», которая является адаптером PostgreSQL для Python.

**Создание и настройка базы данных**

Первый этапом внедрения СУБД в систему является установка PostgreSQL с официального сайта, создание базы данных и настройка. В листинге 1 приведены требуемые запросы к СУБД.

Листинг 1 – Создание и настройка базы данных

```
CREATE DATABASER tutorws_db;
CREATE ROLE tutorws_admin with password 'password_for_db';
ALTER ROLE "tutorws_admin" WITH LOGIN;
GRANT ALL PRIVILEGES ON DATABASE "tutor_ws" to tutorws_admin;
ALTER USER tutorws_admin CREATEDB;
GRANT ALL ON schema public TO tutorws_admin;
```

Прежде всего создаем базу данных `tutorws_db` для веб-приложения. Далее создается роль пользователя, и прописывается имя пользователя (`tutorws_admin`) и пароль. Следующим шагом даем пользователю возможность входа и все привилегии для взаимодействия с базой данных. Также, для того чтобы, в будущем веб-приложение корректно взаимодействовало с базой данных, нужно дать доступ к схеме `public`. Иначе возникнут ошибки при миграциях.

**Изменение конфигурации базы данных в веб-приложении**

Вторым этапом внедрения СУБД в систему является изменение стандартных настроек конфигурации базы данных, так как изначально проект использует SQLite3. В листинге 2 представлена новая конфигурация базы данных.

Листинг 2 – Конфигурация базы данных

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'tutorws_db',
        'USER': 'tutorws_admin',
        'PASSWORD': 'admin',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

Первым параметром конфигурации указывается библиотека, которая используется для подключения СУБД PostgreSQL. Следующими важными

параметрами являются название базы данных (`tutorws_db`), имя пользователя (`tutorws_admin`) и пароль, которые были созданы ранее. Далее объявляется адрес хоста, на котором запущен сервер базы данных. В данном случае используется локальный хост. Это означает, что база данных запущена на той же машине, что и веб-приложение. Также прописывается порт, на котором PostgreSQL ожидает подключения.

**Описание базы данных**

Для обеспечения требуемого функционала веб-приложения было создано шесть различных моделей. В приложении Г представлена схема базы данных, иллюстрирующая взаимосвязи между этими моделями. Ниже приводится подробное описание каждой модели базы данных.

1. Класс `User`. Этот класс в Django представляет собой модель пользователя, которая наследуется от `AbstractUser`. Класс предоставляет стандартные поля и методы для пользователя, такие, как `username`, `password`, `email`, `first_name`, и `last_name`. Также данная модель была расширена дополнительными полями. Контактный номер телефона и фотография пользователя.

2. Класс `Student`. Этот класс предстадставляет собой модель, в которой хранится основная информация, то есть идентификатор ученика, ФИО ученика, класс, предметы по которым учащийся будет заниматься (поле для хранения связи `ManyToManyField` с моделью `Subject`), его контактный номер телефона и его родителя, ФИО родителя, краткий комметарий по ученику для преподавателя

3. Класс `Subject`. Этот класс представляет собой модель, в которой храниться идентификатор предмета, название самого предмета.

4. Класс `Calendar_themathic_plan`. Модель используется для хранения информации о темах, которые относятся к конкретным предметам. Она включает поле предмет, название темы, навыки, связанные с темой.

5. Класс `Task`. Данная модель предназначена для хранения информации о событиях и занятиях, которые отображаются в календаре.

Она включает поле для указания студента (если требуется), даты и время события, краткого описания урока.

6. Класс `New_lesson`. Модель, предназначенная для хранения подробных отчетов по проведенным занятиям. Она включает поля для указания даты и времени занятия, имени ученика, предмета, темы занятия, навыков и домашнего задания.

Также для каждой модели, кроме `User`, есть дополнительное поле `created_by`, содержащее идентификатор пользователя, создавшего запись в базе данных. В приложении В представлена сводная таблица, которая содержит описание каждого поля, во всех классах.

**Создание модели с использованием Django ORM**

При создании модели в базе данных, в файле «models.py» прописывается класс, который использует Django ORM. В листинге 3 представлен один из классов веб-приложения, который в данном случае создает модель `Student` в базе данных.

Листинг 3 – Создание модели

```
from django.db import models
from ct_plan.models import Subjects
from django.core.validators import RegexValidator
from phonenumber_field.formfields import PhoneNumberField
from users.models import User

class Students(models.Model):
    student_name = models.CharField('ФИО ученика', max_length=50)
    number_class = models.IntegerField('Номер класса', blank=True,
default=0)
    subject = models.ManyToManyField(Subjects,
related_name='students_subject')
    phone_regex = RegexValidator(
        regex=r'^(\+7|7|8)?9\d{9}$',
        message="Номер телефона должен быть в формате: '+79999999999'."
    )
    student_phonenumber = PhoneNumberField(
        label='Контактный телефон ученика',
        region='RU',
        validators=[phone_regex]
    )
    social_link = models.CharField('Ссылка', max_length=50)
    parent_name = models.CharField('ФИО родителя', max_length=50,
blank=True)
    parent_phonenumber = PhoneNumberField(
        label='Контактный телефон ученика',
        region='RU',
        validators=[phone_regex],
    )
```

```
    comment = models.TextField('Комментарий', default='', blank='True')
    created_by = models.ForeignKey(User, on_delete=models.CASCADE,
verbose_name="Пользователь")
    def __str__(self):
        return self.student_name
    def get_absolute_url(self):
        return f'/students/{self.id}'
    class Meta:
        verbose_name = 'Студента'
        verbose_name_plural = 'Студенты'
```

Прежде всего необходимо определить следующие данные и библиотеки.

1. Импорт модуля `models` из `django.db`. Модуль, который предоставляет доступ к различным компонентам Django ORM, включая `models`, который используется для определения моделей базы данных.

2. Импорт модели класса Subject из приложения `ct_plan` и модуля `models`, для дальнейшего взаимодействия с данными этой модели.

3. Импорт класса `RegexValidator` из модуля `django.validators`. Объявление валидатора для проверки данных, введенных пользователем, который позволяет определить регулярное выражение для проверки соответствия введенных данных.

4. Импорт класса `PhoneNumberField` из модуля `phonenumber_field`. Определение из сторонней библиотеки поля, которое автоматически обрабатывает валидацию и форматирование телефонных номеров.

5. Импорт модели класса `User` из приложения `users` и модуля `models`, для дальнейшего взаимодействия с данными модели пользователя.

Далее объявляется класс `Student`, который наследуется от класса `model.Models`, в котором создаются поля для данного класса.

1. Поле `student_name` для хранения полного имени ученика в формате текста (`CharField`).

2. Поле `number_class` для хранения номера класса ученика в формате целого числа (`IntegerField`). Может быть пустым (`blank=True`) и иметь значение по умолчанию (`default=0`).

3. Поле subject, которое имеет Связь «многие-ко-многим» с моделью Subjects, что позволяет ученику изучать несколько предметов.

4. Валидатор phone_regex, задающий шаблон для проверки формата номера телефона. Он гарантирует, что в полях student_phonenumber и parent_phonenumber сохраняются только телефонные номера, соответствующие формату «+79999999999» или «89999999999», с указанием соответствующего сообщения об ошибке при необходимости.

5. Поля student_phonenumber и parent_phonenumber для хранения телефонных номеров ученика и его родителя. Используется PhoneNumberField, который обеспечивает автоматическую валидацию телефонного номера и сохранение в формате, соответствующем региону (region='RU').

6. Поле social_link для хранения ссылки на социальную сеть ученика в формате текста (CharField).

7. Поле parent_name для хранения полного имени родителя ученика в формате текста (CharField).

8. Поле comment для хранения дополнительных комментариев о студенте в формате текста (TextField). По умолчанию может быть пустым (default='').

9. Внешний ключ created_by, связывающий данную модель с моделью User, указывающий на то, кто создал запись о студенте.

Также для правильного дальнейшего отображения названий в панели администратора прописывается Meta класс. Он определяет названия модели (verbose_name и verbose_name_plural). Метод __str__ возвращает ФИО ученика. Метод get_absolute_url генерирует URL для доступа к страницы студента.

### 4.3. Реализация backend части веб-приложения

**Метод представления для страницы «Расписание»**

Для того, чтобы реализовать страницу для расписания, используется метод представления из файла «views.py», который связывает модель `Task` и шаблон страницы календаря. Данный метод представления находится в листинге 1 приложения Г.

Прежде всего, необходимо создать два списка, содержащие название месяцев. Первый список содержит название месяцев для дней недели, а второй названия месяца в именительном падеже. Также, для метода представления календаря необходимо прописать декоратор, который запрещает открытие страницы, если пользователь не авторизован в системе. Далее объявляется метод `calendar_view`, принимающий два параметра `year` и `month`, которые по умолчанию равны `None`. Если эти два параметра указываются, то выбирается первый день месяца. Если нет, то выбираются текущий месяц и первый день месяца.

Следующим шагом, в переменных `prev_month` и `next_month`, определяется предыдущий и следующий месяц, относительно текущей даты. Далее перед созданием списка (`days`) всех задач по текущему месяцу, необходимо определить первый день месяца. Если первый день месяца не понедельник, то добавляются задачи дней из предыдущего месяца, чтобы заполнить календарь до понедельника. Затем, для каждого дня текущего месяца, добавляются задачи, связанные с этим днем. Также необходимо проверить каким днем недели заканчивается текущий месяц. Если последний день месяца не воскресенье, то добавляются дни из следующего месяца, чтобы заполнить календарь до воскресенья.

Далее формируется контекст, в который передаются все необходимые данные, которые были описаны выше, чтобы в дальнейшем использовать эти данные в шаблоне страницы. В конце шаблон «calendar_page.html» рендерится с использованием контекста, сформированного выше.

**Дополнительный класс аутентификации**

Стандартно фреймворк Django использует для аутентификации пользователя в систему только поле `username` и для того, чтобы появилась возможность аутентификации не только по логину, но и по адресу электронной почты необходимо реализовать дополнительный класс аутентификации, который представлен в листинге 4.

Листинг 4 – Класс аутентификации по адресу электронной почты

```
from django.contrib.auth.backends import BaseBackend
from django.contrib.auth import get_user_model

class EmailAuthBackend(BaseBackend):
    def authenticate(self, request, username=None, password=None,
**kwargs):
        user_model = get_user_model()
        try:
            user = user_model.objects.get(email=username)
            if user.check_password(password):
                return user
            return None
        except (user_model.DoesNotExist, user_model.MultipleObjectsRe-
turned):
            return None

    def get_user(self, user_id):
        user_model = get_user_model()
        try:
            return user_model.objects.get(pk=user_id)
        except user_model.DoesNotExist:
            return None
```

Первым этапом требуется определить базовый класс для создания пользовательских бэкендов аутентификации и функцию, возвращающую активную модель пользователя, используемую в проекте. `EmailAuthBackend` наследует от `BaseBackend` и реализует два метода: `authenticate` и `get_user`. Для метода authenticate был взят за основу стандартный метод аутентификации, который представлен в классе `BaseBackend`.

Метод `authenticate` выполняет аутентификацию пользователя. Он принимает объект запроса `request`, имя пользователя `username` (в данном случае, электронная почта), пароль `password`, а также дополнительные аргументы `**kwargs`. Сначала он получает модель пользователя через

get_user_model. Затем пытается найти пользователя по адресу электронной почты с помощью user_model.objects.get(email=username). Если пользователь найден, то проверяется правильность пароля с помощью user.check_password(password). Если пароль верен, то метод возвращает объект пользователя, иначе возвращает None. Возможное исключение user_model.DoesNotExist, которое возникает, если пользователь с указанным адресом электронной почты не найден, и user_model.Multiple-ObjectsReturned, которое возникает, если найдено несколько пользователей с указанным адресом электронной почты.

Метод get_user предназначен для получения пользователя по его идентификатору user_id. Сначала он получает модель пользователя с помощью get_user_model, а затем пытается найти пользователя по первичному ключу user_id, через user_model.objects.get(pk=user_id). Если пользователь с указанным идентификатором не найден, то возникает исключение user_model.DoesNotExist.

**Класс формы «Добавить отчет»**

Класс формы New_lessonForm предназначен для ввода данных о новых занятиях, с возможностью динамически подгружать данные, в зависимости от выбранных значений. Это обеспечивает удобство использования и уменьшает вероятность ошибок при вводе данных. Класс формы для добавления отчета представлен в листинге 2 приложения Г.

Класс New_lessonForm наследует forms.ModelForm, что упрощает создание формы на основе модели New_lesson. Поле date используется для ввода даты занятия и представлено как DateField с виджетом DateInput. Поля start_time и end_time предназначены для ввода времени начала и окончания занятия, реализованы как TimeField с виджетом TimeInput. Поля student_name и subject реализованы как ModelChoiceField для выбора ученика и предмета соответственно. Изначально они имеют пустые списки Students.objects.none и Subjects.objects.none, но они заполняются динамически в зависимости от данных пользователя. Поле

topic_name также является ModelChoiceField и зависит от выбранного предмета. Поле skills является скрытым полем HiddenInput, которое автоматически заполняется в зависимости от выбранной темы, homework представляет собой текстовое поле CharField для ввода домашнего задания.

В классе Meta указано, что форма основана на модели New_lesson и перечислены все поля, которые должны быть включены в форму. Метод __init__ переопределен для настройки формы в зависимости от переданных аргументов: если передан пользователь user, список учеников фильтруется так, чтобы отображались только те, которые созданы этим пользователем. Метод также устанавливает начальные значения полей, на основе переданных данных и обновляет списки выбора queryset, в зависимости от выбранных значений формы.

**Метод представления для страницы «Добавить отчет»**

Для того, чтобы реализовать страницу «Добавить отчет», используется метод представления из файла «views.py», который связывает модель New_lesson, форму New_lessonForm и шаблон страницы. Данный метод представления отображен в листинге 5.

Листинг 5 – Метод представления для страницы «Добавить отчет»

```
@login_required
def new_lesson(request):
    if request.method == 'POST':
        form = New_lessonForm(request.POST, user=request.user)
        if form.is_valid():
            new_lesson = form.save(commit=False)
            new_lesson.created_by = request.user
            new_lesson.save()
            messages.success(request, 'Занятие успешно сохранено!')
            return redirect('home')
        else:
            messages.error(request, 'Пожалуйста, исправьте ошибки в форме.')
    else:
        form = New_lessonForm(user=request.user)
    return render(request, 'new_lesson/new_lesson_1.html', {'form': form})
```

Прежде всего, для метода представления необходимо добавить предмет прописать декоратор, который запрещает открытие страницы, если пользователь не авторизован в системе. При отправке формы методом POST

создается экземпляр формы `New_lessonForm`, который заполняется данными запроса и текущего пользователя. После валидации формы новое занятие сохраняется в базе данных с указанием создателя. В случае успешного сохранения выводится сообщение об успехе, и пользователь перенаправляется на домашнюю страницу. Если форма не прошла валидацию, то пользователю показывается сообщение об ошибке. При `GET` запросе форма инициализируется пустой для отображения на странице создания занятия.

### 4.4. Реализация frontend части веб-приложения

Frontend часть веб-приложения представляет собой интерфейс, с которым взаимодействует пользователь. В процессе разработки frontend необходимо уделить внимание нескольким ключевым аспектам: дизайну, удобству использования и адаптивности интерфейса.

Дизайн должен быть привлекательным и современным, чтобы обеспечить положительное впечатление пользователей. Использование актуальных визуальных стилей, цветов и шрифтов способствует созданию приятного и профессионального внешнего вида приложения. Кроме того, необходимо учитывать особенности сенсорных экранов, таких как удобство нажатия и прокрутки, чтобы пользователи могли комфортно взаимодействовать с приложением на мобильных устройствах.

Удобство использования (usability) подразумевает интуитивно понятный интерфейс, который позволяет пользователям легко выполнять необходимые действия. Навигация должна быть логичной и доступной, кнопки и ссылки – достаточно крупными и заметными. Проведение юзабилити-тестирования на ранних этапах разработки поможет выявить возможные проблемы и улучшить пользовательский опыт.

Адаптивность интерфейса обеспечивает корректное отображение и функционирование на различных устройствах, включая настольные компьютеры, планшеты и смартфоны. Это достигается за счет применения меди-

азапросов CSS и гибкой сеточной системы. Кроме того, необходимо учитывать особенности сенсорных экранов, таких как удобство нажатия и прокрутки, чтобы пользователи могли комфортно взаимодействовать с приложением на мобильных устройствах.

**Шаблон HTML для страницы «Расписание»**

Для реализации страницы «Расписание» используется метод представления, который использует HTML-шаблон «calendar_page.html» для вывода информации по событиям и занятиям из модели `Task`. Исходный код для страницы «Расписание» представлен на листинге 3 приложения Г.

Первой строчкой прописывается базовый шаблон страницы, в котором содержится меню навигации. Для определения частей шаблона используются блоки, определяющие части шаблона, которые могут быть переопределены в дочерних шаблонах. На данной странице переопределяется блок `<title>` для изменения названия страницы, блок `<head>` для определения ссылки на файл CSS, где прописаны все стили для данной страницы, блок `<title-page>`, где прописывается название этой страницы и блок `<content>`, где находится основная разметка страницы.

Первым этапом прописывается блок `<div>`, который содержит стрелки для навигации по месяцам, название месяца и год. В этом блоке объявляется `<table>`. В `<tr>` прописывается первая строка таблицы – названия дней недели. Блок `<td>` содержит строку таблицы, в которой уже прописывается каждый день недели, в котором находятся события и занятия.

Прежде всего, в блоке `<td>` с помощью for-тега `{% for day, tasks in days %}` перебирается список всех дней и задач, которые были переданы, как список days из метода представления этой страницы `calendar_view`. Далее прописывается три разные конструкции для разделения дней прошлого месяца, текущего месяца и следующего месяца, чтобы реализовать различные стили для разных дней недели. А именно, серым цветом будут выделены дни, если они не является днями этого месяца и белым, если это дни текущего месяца.

В ячейке каждого дня выводится день и месяц `{{ day|date:"d" }}` `{{ past_month }}`, а также события и занятия, которые есть в расписании на день. Для каждого события выводится время, в которое запланировано событие.

Для данной страницы также были прописаны стили в файле «calendar_page.css», исходный код которой представлен на листинге 4 приложения Г.

CSS-код стилизует календарь на веб-странице. Основной контейнер `calendar-container` задает ширину, отступы и позицию.

Навигация `navigation` выравнивает элементы по центру и распределяет их пространство, а стрелки .nav-arrow центрируются и отображаются с помощью псевдоэлементов.

Таблица имеет фиксированную ширину, убирает пробелы между ячейками и использует внешнюю границу. Заголовки `th` центрируются, имеют серый цвет, жирный шрифт и соответствующий фон. Ячейки `td` имеют белый фон и левое выравнивание текста. Прошлые и будущие дни выделяются серым фоном. Ссылки внутри ячеек черные и жирные. Контейнер задач `task-container` использует сетку для равномерного распределения, а задачи `task` имеют полупрозрачный фон и скругленные углы.

Такой дизайн позволяет легко ориентироваться в календаре, предоставляя пользователю интуитивно понятный и визуально привлекательный интерфейс. Итоговое отображение страницы «Расписание» расположено на рисунке 5 приложения Е.

Страница «Расписание» содержит базовый шаблон, который расширен шаблоном для отображения расписания преподавателя. Шаблон базовой страницы включает стандартную разметку, боковое меню с основными ссылками на страницы веб-приложения, а также блок, где отображается основная информация о текущем авторизованном пользователе.

Боковое меню состоит из разделов, таких как «Главная», «Расписание», «Ученики», «КТП», «Добавить отчет» и «О разработчике». В разделе

«Расписание» помимо самого расписания преподавателя также есть ссылки на страницы добавления занятий и просмотра всех проведенных занятий. В разделе «Ученики» можно просматривать список учеников и добавлять новых. В разделе «КТП» пользователь может создать новую тему для выбранного предмета, а также добавить новый предмет.

Шаблон страницы «Расписание» расширяет базовый шаблон, добавляя специфический контент. В блоке `home-section` расположен календарь с расписанием событий и занятий преподавателя. Календарь имеет возможность навигации по разным месяцам, что позволяет преподавателю легко просматривать свое расписание на ближайшие и прошлые месяцы.

При отображении расписания каждый день месяца представлен ячейкой календаря, в которой указана дата и список занятий. Занятия отображаются с указанием времени начала и окончания, а также именем ученика. Ячейки, которые относятся к дням предыдущего или следующего месяца имеют отличительный стиль, чтобы визуально отделить их от текущих и будущих дней.

**Адаптивная верстка веб-приложения**

Данное веб-приложение предназначено для широкого использования, поэтому верстка страниц данного веб-приложения адаптивна, то есть элементы страниц изменяются в соответствии с размером экрана устройства, на котором открыто веб-приложение. Это обеспечивает удобство работы пользователям, независимо от того, используют ли они смартфон, планшет или настольный компьютер.

Благодаря адаптивной верстке веб-приложение автоматически подстраивается под различные разрешения экрана, сохраняя при этом функциональность и эстетичность интерфейса. Несколько примеров страниц с адаптивной версткой представлены на рисунках 6–7.

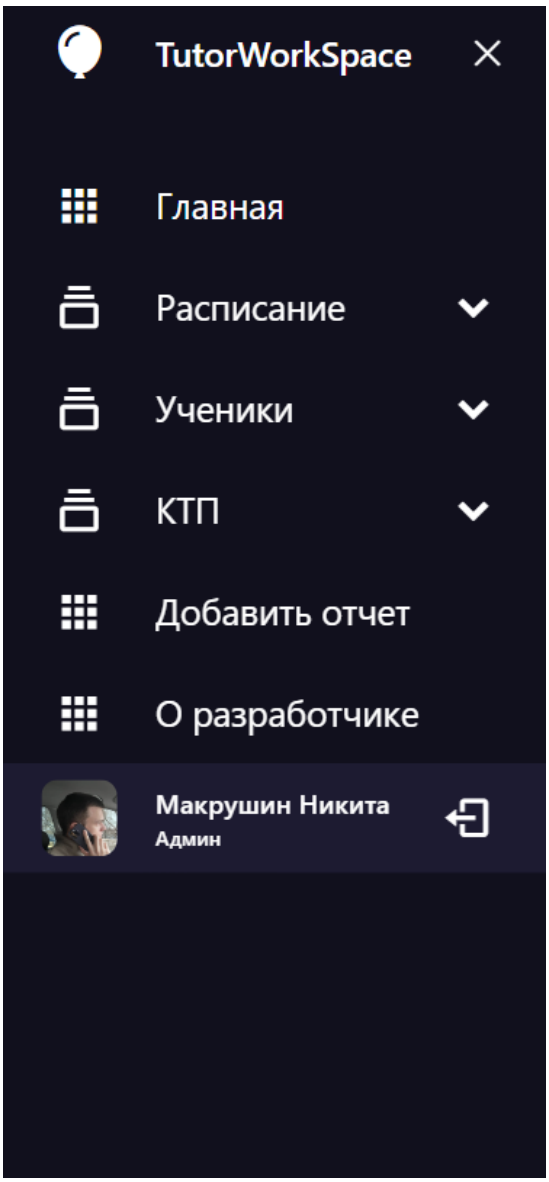


Рисунок 6 – Меню веб-приложения

При реализации мобильной версии верстки веб-приложения было принято решение переработать боковое меню для более удобного взаимодействия пользователя с веб-приложением. Теперь, при открытии веб-приложения пользователь не видит меню, но при нажатии на кнопку, расположенную в правом верхнем углу, меню раскрывается на полный экран. Вся функциональность меню навигации сохранена в полном объеме.

Страница с формой добавления отчета по проведенному занятию с шириной экрана планшета представлена ниже на рисунке 7.

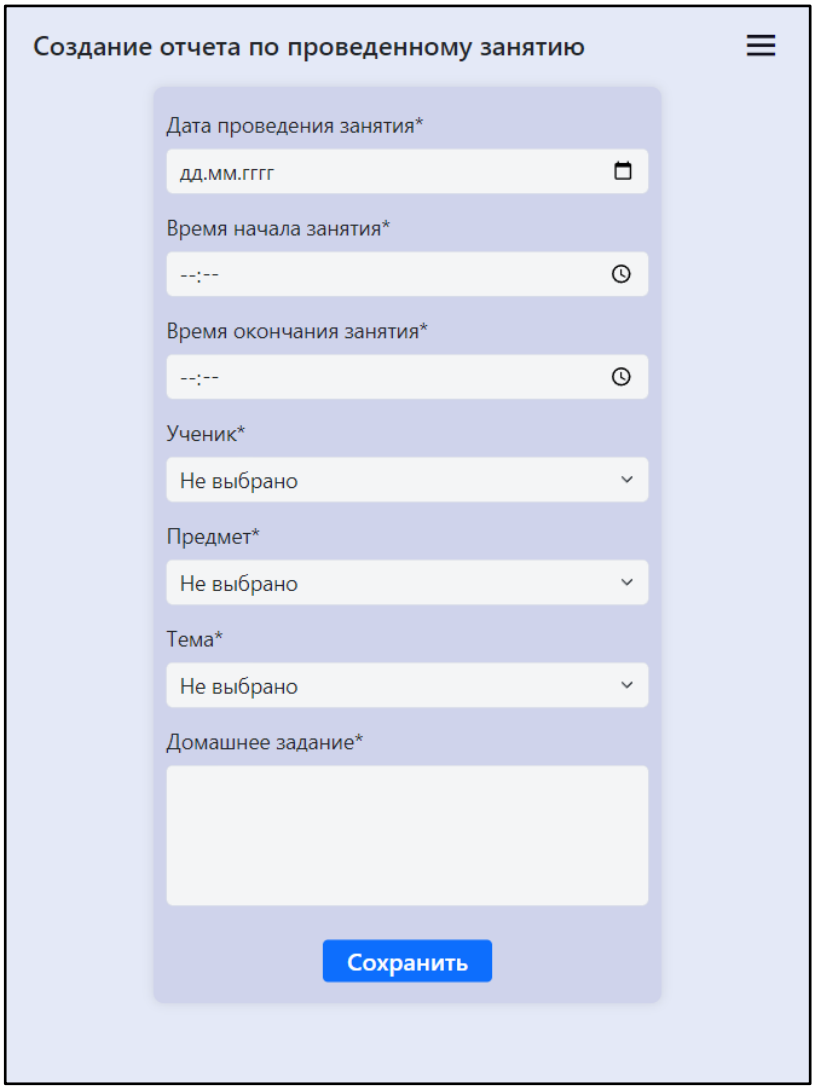


Рисунок 7 – Страница «Добавить отчет»

Для данной страницы было принято решение адаптировать размеры формы и содержимое с полным сохранением функциональности.

**Вывод по четвертой главе**

В результате реализации было разработано веб-приложение для отслеживания успеваемости учеников онлайн-школы. Было реализовано внедрение СУБД PostgreSQL, которая будет перекрывать все потребности в производительности и дальнейшем масштабировании системы. Схема базы данных представлена в приложении Д. Далее были разработаны 6 сущностей базы данных с использованием ORM Django. Успешно реализованы методы представления для различных страниц и разработаны шаблоны для различных страниц, с учетом отображения на различных устройствах.

# 5. ТЕСТИРОВАНИЕ

В данной главе описано тестирование веб-приложения. Тестирование является неотъемлемой частью создания веб-приложения, потому что именно на данном этапе выявляются основные ошибки, проблемы и недочеты в работе системы и ее взаимодействии с пользователем. Тестирование состоит из нескольких частей:

1) функциональное тестирование системы;
2) тестирование адаптивной верстки веб-приложения.

## 5.1. Функциональное тестирование системы

Функциональное тестирование является ключевым этапом в обеспечении корректной работы веб-приложения. Главной целью является проверка всех функций и возможностей системы в соответствии с требованиями и ожиданиями пользователей. В рамках этого тестирования проверяются прежде всего все варианты использования веб-приложения, включая регистрацию, аутентификацию, управление данными и взаимодействие пользователя с различными модулями. Это позволит выявить и исправить возможные ошибки до запуска веб-приложения в эксплуатацию. Тестовые случаи и результаты функционального тестирования, которые расположены ниже в таблице 2.

Таблица 2 – Функциональное тестирование системы

| № | Тестовый случай | Описание теста | Шаги | Ожидаемый результат | Статус |
|---|---|---|---|---|---|
| 1 | Регистрация пользователя | Проверка возможности регистрации нового пользователя | 1. Открыть страницу регистрации<br>2. Ввести данные (логин, пароль, подтверждение пароля, номер телефона, имя, фамилия, email)<br>3. Нажать «Регистрация» | Новый пользователь успешно зарегистрирован и перенаправлен на страницу профиля | Успешно |

Продолжение таблицы 2

| № | Тестовый случай | Описание теста | Шаги | Ожидаемый результат | Статус |
|---|---|---|---|---|---|
| 2 | Вход в систему | Проверка авторизации пользователя | 1.Открыть страницу входа<br>2. Ввести логин или email и пароль<br>3. Нажать «Войти» | Пользователь успешно авторизован и перенаправлен на главную страницу | Успешно |
| 3 | Восстановление пароля | Проверка восстановления пароль, через ссылку, которая приходит на почту | 1. Открыть страницу входа<br>2. Нажать «Забыли пароль»<br>3. Отрыть почту и перейти по ссылке<br>4. Ввести новый пароль и подтвердить его.<br>5. Открыть страницу входа и войти уже с новым паролем | Пользователь успешно восстановил пароль, перейдя по ссылке из почты, и вошел уже с новым паролем в систему. | Успешно |
| 4 | Изменить пароль пользователя | Изменение пароля пользователя на новый | 1. Открыть страницу профиля<br>2. Нажать кнопку изменить пароль<br>3. Ввести старый и новый пароль, и повтор нового пароля | Пользователь успешно изменил пароль | Успешно |
| 5 | Добавление ученика | Добавить нового ученика в систему | 1. Открыть страницу «Добавить ученика»<br>2. Ввести все данные ученика и выбрать предмет(ы)<br>3. Нажать кнопку «Сохранить» | Пользователь успешно добавил ученика в систему | Успешно |
| 6 | Добавление предмета | Добавление нового предмета | 1. Открыть страницу «Добавить предмет»<br>2. Ввести название предмета<br>3. Нажать кнопку «Сохранить» | Пользователь успешно добавляет предмет в систему | Успешно |
| 7 | Редактирование ученика | Изменить какие-либо данные по выбранному ученику | 1. Открыть профиль ученика<br>2. Нажать кнопку «Редактировать»<br>3. Изменить данные<br>4. Нажать кнопку «Сохранить» | Пользователь успешно изменил данные выбранного ученика и перенаправлен на профиль ученика | Успешно |

Окончание таблицы 2

| № | Тестовый случай | Описание теста | Шаги | Ожидаемый результат | Статус |
|---|---|---|---|---|---|
| 8 | Удаление ученика | Удаление выбранного ученика | 1. Открыть профиль ученика<br>2. Нажать кнопку «Удалить» | Пользователь успешно удалил ученика и был перенаправлен на страницу списка учеников | Успешно |
| 9 | Добавление темы | Добавить новую тему для конкретного предмета | 1. Открыть страницу «Добавить тему»<br>2. Выбрать предмет<br>3. Ввести необходимые данные для темы (название, навыки, краткая информация) | Пользователь успешно добавил новую тему по выбранному предмету | Успешно |
| 10 | Добавление события или занятия | Добавить новое событие или занятие в расписание | 1. Открыть страницу «Добавить событие»<br>2. Ввести дату, время, краткое описание, и если нужно выбрать ученика | Пользователь успешно добавил событие в расписание | Успешно |

В результате функционального тестирование проведены тесты по проверке всех вариантов использования системы. Все тесты прошли успешно. Проблем в результате тестирования выявлено не было.

## 5.2. Тестирование адаптивной верстки веб-приложения

Тестирование адаптивной верстки веб-приложения направлено на проверку корректного отображения и функционирования интерфейса на различных устройствах и экранах. Адаптивность является важным аспектом современного веб-дизайна, поскольку пользователи могут использовать веб-приложение на смартфонах, планшетах, ноутбуках и настольных компьютерах. В ходе тестирования проверяется насколько корректного элементы подстраиваются и отображаются на страницах, сохраняя при этом свою функциональность. В данном разделе будут описаны тестовые случаи и результаты тестирования адаптивной верстки, которые представлена ниже в таблице 3.

Таблица 3 – Тестирование адаптивной верстки веб-приложения

| № | Тестовый случай | Описание теста | Шаги | Ожидаемый результат | Статус |
|---|---|---|---|---|---|
| 1 | Навигационное меню (мобильное устройство) | Проверка отображения и корректной работы навигационного меню на мобильном устройстве | 1. Открыть любую страницу на устройстве с шириной экрана < 600px<br>2. Нажать кнопку «Меню» | Меню корректно отображается, все элементы видны и отображаются корректно | Успешно |
| 2 | Форма регистрации (планшет) | Проверка отображения формы регистрации на планшете | Открыть страницу регистрации на устройстве с шириной экрана 600px-900px | Форма отображается корректно, все выпадающие списки, поля и кнопки отображаются и функционируют корректно | Успешно |
| 3 | Отображение списка учеников (мобильное устройство) | Проверка отображения списка учеников на мобильном устройстве | Открыть страницу списка учеников на устройстве с шириной экрана < 600px | Список отображается и функционирует корректно | Успешно |
| 4 | Страница «Профиль пользователя» (настольный компьютер) | Проверка отображения страницы профиля пользователя на настольном компьютере | Открыть страницу профиля пользователя на устройстве с шириной экрана > 900px | Страница профиля пользователя отображается и функционирует корректно. | Успешно |
| 5 | Страница «Расписание» (настольный компьютер) | Проверка отображения расписания пользователя на настольном компьютере | Открыть страницу «Расписание» на устройстве с шириной экрана 900px> | Навигационное меню и календарь отображается и функционирует корректно | Успешно |
| 6 | Страница «Добавить отчет» (планшет) | Проверка отображения страницы «Добавить отчет» на планшете | 1. Открыть страницу «Добавить отчет» на устройстве, с шириной экрана 600px-900px<br>2. Заполнить поля формы<br>3. Нажать кнопку «Сохранить» | Форма добавления отчета по проведенному занятию отображается корректно. Выпадающие списки функционируют корректно. | Успешно |

Окончание таблицы 2

| № | Тестовый случай | Описание теста | Шаги | Ожидаемый результат | Статус |
|---|---|---|---|---|---|
| 7 | Страница «Профиль ученика» (настольный компьютер) | Проверка отображения страницы «Профиль ученика» на настольном компьютере | 1. Открыть страницу «Профиль ученика» на устройстве, с шириной экрана >900px.<br>2. Просмотреть отчеты по проведенным занятиям. | Страница и данные отображаются корректно, все кнопки функционируют, список с отчетами листается корректно | Успешно |
| 8 | Страница «Вход» (мобильное устройство) | Проверка отображения страницы «Вход» на мобильном устройстве | 1. Открыть страницу «Вход» на устройстве, с шириной экрана <600px | Форма отображается и функционирует корректно | Успешно |

Проведение тестирования адаптивной верстки показало, что все тесты были пройдены успешно, страницы на разных размерах экранов устройств отображаются корректно, функциональные возможности были сохранены.

**Вывод по пятой главе**

Проведенное тестирование подтвердило функциональную и визуальную корректность веб-приложения. Функциональное приложение показало, что все ключевые возможности системы работают без сбоев, что обеспечивает правильное взаимодействие пользователя с системой. Тестирование адаптивной верстки приложения показало, что интерфейс веб-приложения корректно отображается на разных экранах устройств, функционал при изменениях размера экрана также сохраняется. Таким образом веб-приложение готово к запуску и дальнейшей эксплуатации.

## ЗАКЛЮЧЕНИЕ

В ходе данной работы было разработано веб-приложения для отслеживания успеваемости учеников онлайн-школы, которое позволяет пользователям вести учет и статистику по ученикам и проведенным занятиям. В результате выполнения работы были выполнены следующие задачи:

1) проведен анализ предметной области веб-приложения;
2) выявлены основные требования системы;
3) выбран фреймворк и спроектирована архитектура веб-приложения;
4) выполнена реализация веб-приложения:

– создана база данных с использованием PostgreSQL и встроенным ORM Django;

– разработаны формы для заполнения данных пользователем, с использованием встроенных виджетов;

– реализованы функции представления страниц;

– созданы HTML шаблоны страниц веб-приложения с использованием встроенных инструментов Django и файлов CSS для настройки стилей страниц;

5) проведено успешное функциональное тестирование системы и тестирование адаптивной верстки веб-приложения.

В дальнейшем планируется продолжить разработку данного проекта, и внедрить данное веб-приложение в онлайн-школу. Также составлен план дальнейшего развития и модернизации проекта:

1) внедрить отправку расписания занятий и отчетов по проведенным занятиям ученикам и родителям;
2) внедрить google календарь, для отображения расписания занятий не только в веб-приложении, но в других приложениях;
3) внедрить телеграмм бота ,связанного с базой данных, в котором ученики будут отслеживать необходимую информацию по занятиям.

**ЛИТЕРАТУРА**

1. Django Docs. [Электронный ресурс] URL: https://djangodoc.ru/3.2/ (дата обращения: 01.04.2024 г.).

2. Django MTV (Model-Template-View) Pattern. [Электронный ресурс] URL: https://docs.djangoproject.com/en/3.2/faq/general/#django-uses-an-mtv-architecture (дата обращения: 05.04.2024 г.).

3. JetBrains. PyCharm Community Edition. [Электронный ресурс] URL: https://www.jetbrains.com/pycharm/download/#section=windows (дата обращения: 20.03.2024 г.).

4. Jinga Docs. [Электронный ресурс] URL: https://sufangmu.github.io/resource/python/jinja-palletsprojects-com-en2.11.x.pdf
(дата обращения: 03.03.2024 г.).

5. Python Docs. [Электронный ресурс] URL: https://docs.python.org/3/ (дата обращения: 05.03.2024 г.).

6. Ганди Р. Head First. Git. // Издательство «БХВ-Петербург», 2023. – 464 с.

7. Дауни А.Б. Основы Python. Научиться думать, как программист. // МИФ, 2023. – 302 с.

8. Документация по GitHub. [Электронный ресурс] URL: https://docs.github.com/ru (дата обращения: 16.03.2024 г.).

9. Дронов В.А. Django 4. Практика создания веб-сайтов на Python. // Издательство «БХВ-Петербург», 2024. – 791 с.

10. Кугаевский А.В. Проектирование информационных систем. Системная и бизнес-аналитика. // Новосибирский государственный технический университет (НГТУ), 2019. – 50 с.

11. Лутц М. Изучаем Python. // Компьютерное издательство «Диалектика», 2019. – 833 с.

12. Постолит А.В. Python, Django и Bootstrap для начинающих. // Издательство «БХВ-Петербург», 2023. – 624 с.

13. Рамальо Л. Python. К вершинам мастерства. // ДМК-Пресс, 2022. – 768 с.

14. Седер Н. Python. Экспресс-курс. // Издательский дом «Питер», 2021. – 480 с.

15. Фаулер М. UML. Основы, 3-е издание. // Символ-Плюс, 2019. – 456 с.

16. Форсье Д., Биссекс П., Чан. У. Django. Разработка веб-приложений на Python. // Символ-Плюс, 2020. – 440 с.

17. Яблонски Д. Законы UX-дизайна. // Издательство «БХВ-Петербург», 2022. – 160 с.

# ПРИЛОЖЕНИЯ

## Приложение А. Спецификация вариантов использования

Спецификация вариантов использования (ВИ) системы приведена в таблицах 1–4.

Таблица 1 – Спецификация ВИ «Просмотр списка отчетов»

| |
|---|
| Прецедент: Просмотр списка отчетов |
| ID: 1. |
| Краткое описание:<br>Просмотр списка отчетов по всем проведенным занятиям |
| Главные актеры:<br>Пользователь |
| Второстепенные актеры:<br>Нет |
| Предусловия: Открыта главная страница. |
| Основной поток:<br>1. Вариант использования начинается, когда пользователь открывает главную страницу.<br>2. Система выведет список отчетов (тема, предмет, имя ученика, и дату).<br>3. Пользователь может также отсортировать по выбранным данным). |
| Постусловия*:*<br>1. Переход на профиль определенного ученика или тему. |
| Альтернативные потоки:<br>Нет |

Таблица 2 – Спецификация ВИ «Добавить предмет»

| |
|---|
| Прецедент: Добавить предмет |
| ID: 2. |
| Краткое описание:<br>Добавить новый предмет в базу данных. |
| Главные актеры:<br>Пользователь |
| Второстепенные актеры:<br>Нет |
| Предусловия*:*<br>Пользователь переходит на страницу добавление нового предмета. |
| Основной поток:<br>1. Вариант использования начинается, когда пользователь переходит на страницу «Добавить предмет».<br>2. Система открывает страницу с формой для добавления предмета.<br>3. Пользователь вводит название нового предмета.<br>4. Пользователь нажимает кнопку «Сохранить».<br>5. Система перенаправляет пользователя на главную страницу. |
| Постусловия:<br>1. Новый предмет в базе данных. |
| Альтернативные потоки:<br>Нет. |

Таблица 3 – Спецификация ВИ «Добавить отчет по занятию»

| |
|---|
| Прецедент: Добавить отчет по занятию |
| ID: 3. |
| Краткое описание:<br>Добавить новый отчет по проведенному занятию с учеником в базу данных |
| Главные актеры:<br>Пользователь |
| Второстепенные актеры:<br>Нет |
| Предусловия: Открыта страница создания отчетов. |
| Основной поток:<br>1. Вариант использования начинается, когда пользователь переходит на страницу «Добавить отчет»<br>2. Система открывает страницу с формой для добавления отчета по проведенному уроку.<br>2. Система выведет список отчетов (тема, предмет, имя ученика, и дату).<br>3. Пользователь вводит данные (дату и время, домашнее задание) и выбирает (ученика, предмет, тему) из имеющихся в системе данных.<br>4. Пользователь нажимает кнопку «сохранить».<br>5. Система перенаправляет пользователя на главную страницу. |
| Постусловия*:*<br>1. Добавлен отчет по проведенному уроку в базу данных. |
| Альтернативные потоки:<br>Нет |

Таблица 4 – Спецификация ВИ «Просмотр профиля ученика»

| |
|---|
| Прецедент: Редактирование темы |
| ID: 4. |
| Краткое описание:<br>Просмотр полной информации по ученику, списка всех отчетов по проведенным занятиям, редактирование информации или удаление ученика. |
| Главные актеры:<br>Пользователь |
| Второстепенные актеры:<br>Нет |
| Предусловия*:* Пользователь переходит на страницу профиля выбранного ученика. |
| Основной поток:<br>1. Вариант использования начинается, когда пользователь переходит на страницу для редактирования темы.<br>2. Пользователь просматривает общую информацию ученика. |
| Постусловия:<br>1. Отредактированная или удаленная тема. |
| Альтернативные потоки:<br>I. Пользователь нажимает кнопку удалить, и система удаляет ученика и перенаправляет пользователя на главную страницу.<br>II.Пользовать нажимает кнопку редактировать, и система перенаправляет пользователя на страницу редактирования данных об ученике. |

## Приложение Б. Примеры аналогичных проектов

На рисунках 1–4 представлены скриншоты аналогичных приложений.

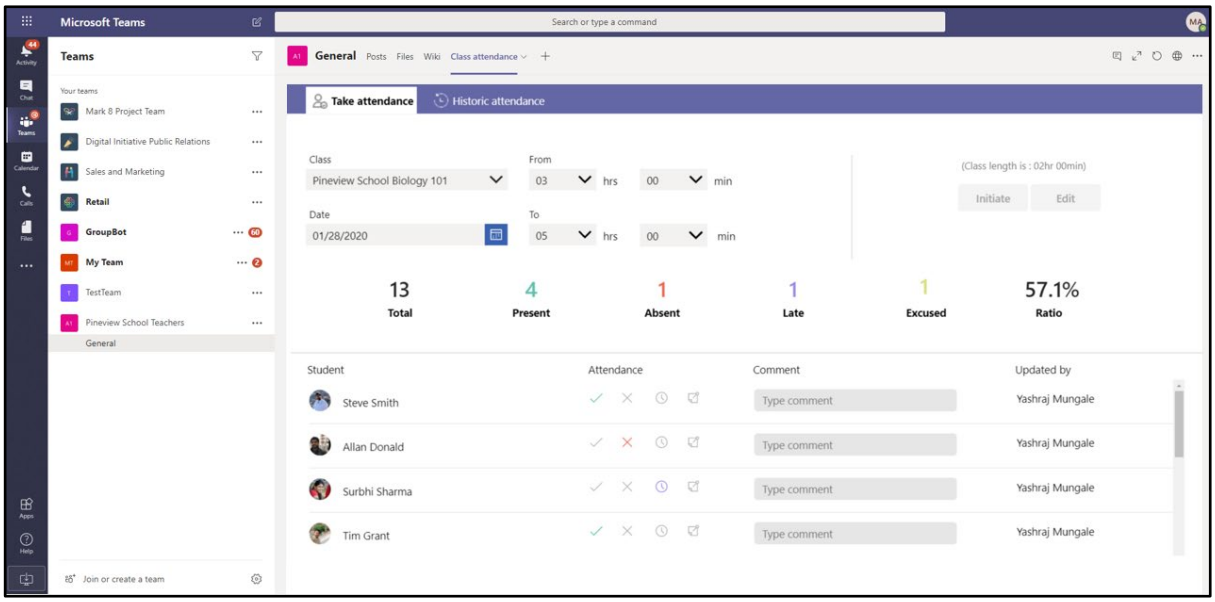


Рисунок 1 – Скриншот веб-приложения «Microsoft Teams for Education»

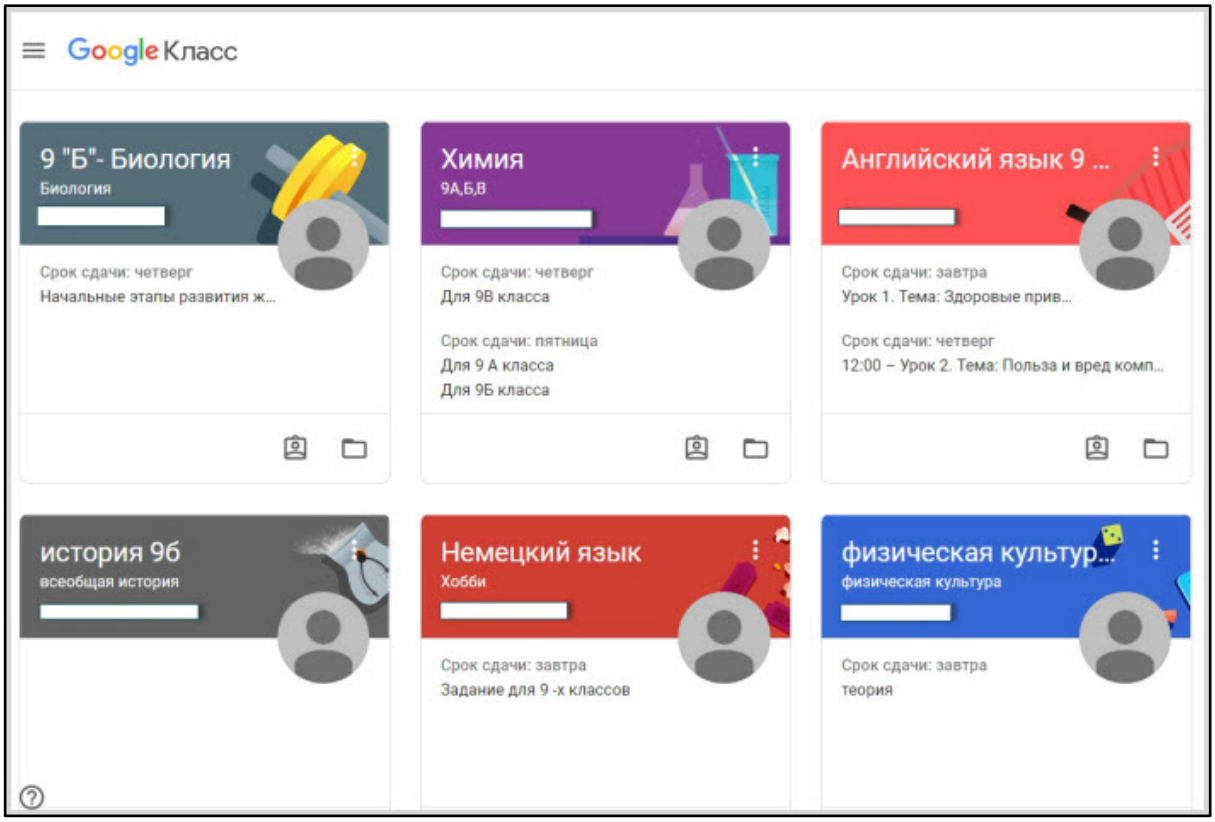


Рисунок 2 – Скриншот веб-приложения «Google Classroom»

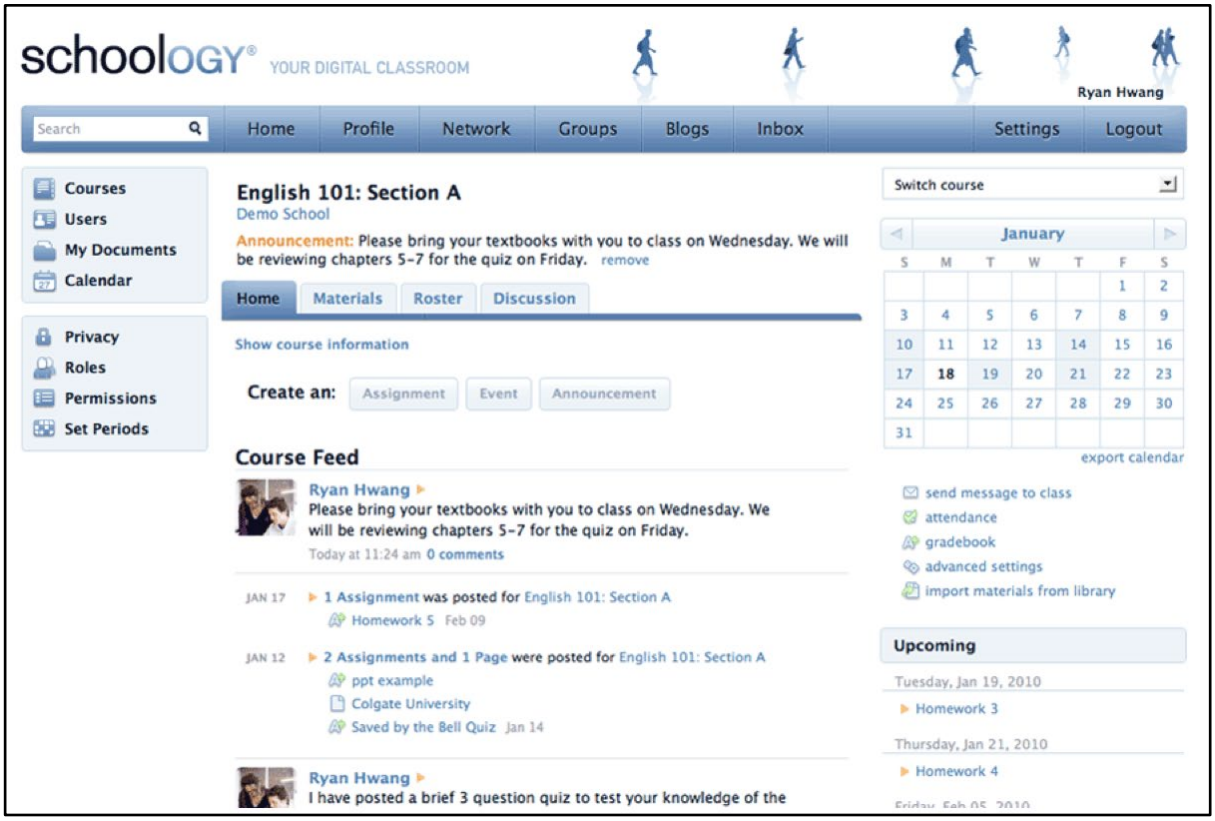


Рисунок 3 – Скриншот веб-приложения «Schoology»

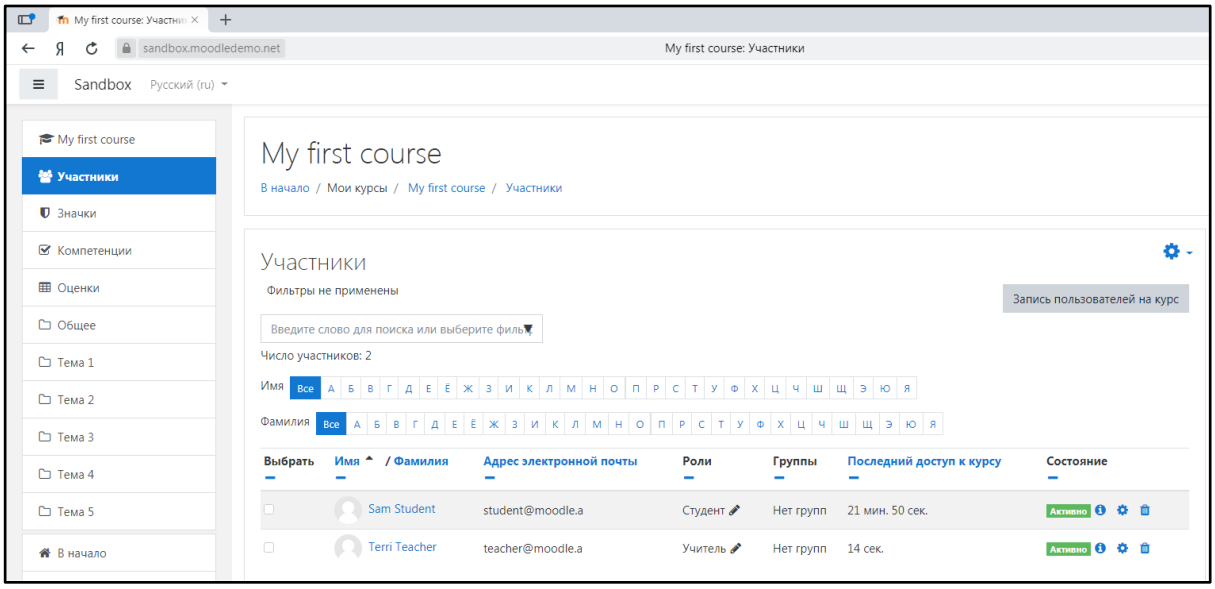


Рисунок 4 – Скриншот веб-приложения «Moodle»

## Приложение В. Сущности базы данных

Таблица 5 – Сущности базы данных

| № | Сущность | Поля |
|---|---|---|
| 1 | Student | id – идентифтикатор студента<br>student_name – ФИО ученика<br>number_class – номер класса<br>social_link – ссылка на социальную сеть<br>parent_name – ФИО родителя ученика<br>comment – комментарий преподавателя по ученику<br>created_by_id – идентификатор пользователя, добавившего ученика |
| 2 | User | id – идентификатор пользователя<br>password – пароль пользователя<br>last_login – время последнего входа пользователя в систему<br>is_superuser – значение статуса пользователя в системе(администратор или стандартный пользователь)<br>username – логин пользователя<br>first_name – имя пользователя<br>last_name – фамилия пользователя<br>email – адрес электронной почты<br>is_staff – булевое поле, указывающее, является ли пользователь членом персонала<br>is_active – булевое поле, указывающее, активен ли пользователь<br>date_joined – дата и время создания учетной записи пользователя<br>image – фото пользователя<br>phone_number – контактный номер телефона пользователя |
| 3 | Task | id – идентификатор события<br>date – дата создания события<br>start_time – время начала события<br>end_time – время окончания события<br>description – описание события<br>created_by – идентификатор пользователя, создавшего событие<br>student_id – идентификатор пользователя |
| 4 | Subject | id – идентификатор предмета<br>subject – название предмета<br>created_by – идентификатор пользователя, создавшего предмет |
| 5 | Calendar_thematic_plan | id – идентификатор темы<br>subject_id – идентификатор предмета<br>topic_name – название темы<br>skills – навыки<br>created_by – идентификатор пользователя, создавшего предмет |

Окончание таблицы 5 приложения В

| № | Сущность | Поля |
|---|---|---|
| 6 | New_lesson | id – идентификатор отчета по занятию<br>date – дата проведения занятия<br>start_time – время начала занятия<br>end_time – время окончания занятия<br>student_name_id – идентификатор ученика<br>subject_id – идентификатор предмета<br>topic_name_id – идентификатор темы<br>skills – навыки полученные при изучении темы<br>homework – домашнее задание<br>created_by – идентификатор пользователя, создавшего отчет по занятию |
| 7 | Student_subject | id – идентификатор поля таблицы связей<br>student_id – идентификатор ученика<br>subject_id – идентификатор предмета |

## Приложение Г. Исходный код реализации веб-приложения

Исходный код реализации веб-приложения представлен в листингах 1–4.

Листинг 1 – Метод представления страницы календаря

```
month_for_date = ['Января', 'Февраля', 'Марта',
                  'Апреля', 'Мая', 'Июня',
                  'Июля', 'Августа', 'Сентября',
                  'Октября', 'Ноября', 'Декабря']

month_list = ['Январь', 'Февраль', 'Март',
              'Апрель', 'Май', 'Июнь',
              'Июль', 'Август', 'Сентябрь',
              'Октябрь', 'Ноябрь', 'Декабрь']

@login_required
def calendar_view(request, year=None, month=None):
    if year is None or month is None:
        today = datetime.today()
        year = today.year
        month = today.month
    else:
        today = datetime(year, month, 1)

    # Определяем предыдущий и следующий месяцы
    prev_month = today - relativedelta(months=1)
    next_month = today + relativedelta(months=1)

    # Формируем список дней и задач для каждого дня
    first_day_of_month = datetime(year, month, 1)
    weekday_of_first_day = first_day_of_month.weekday()  # День недели пер-
вого дня месяца (0 - понедельник, 6 - воскресенье)
    days = []
    if weekday_of_first_day != 0:  # Если первый день месяца не понедельник
        # Добавляем дни прошлого месяца в начало текущего месяца
        prev_month_last_day = (first_day_of_month - timedelta(days=1)).day
        for day in range(prev_month_last_day - weekday_of_first_day + 1,
prev_month_last_day + 1):
            prev_month_date = datetime(year, month - 1, day)
            tasks = Task.objects.filter(date=prev_month_date, cre-
ated_by=request.user)
            days.append((prev_month_date, tasks))
    # Формируем список дней и задач для текущего месяца
    num_days = monthrange(year, month)[1]
    for day in range(1, num_days + 1):
        current_date = first_day_of_month + timedelta(days=day - 1)
        tasks = Task.objects.filter(date=current_date, created_by=re-
quest.user)
        days.append((current_date, tasks))
    last_day_of_month = first_day_of_month + timedelta(days=num_days - 1)
    if last_day_of_month.weekday() != 6:  # Если последний день месяца не
воскресенье
        days_to_sunday = 6 - last_day_of_month.weekday()
        # Добавляем дни следующего месяца в конец текущего месяца
        for day in range(1, days_to_sunday + 1):
            next_month_date = last_day_of_month + timedelta(days=day)
            tasks = Task.objects.filter(date=next_month_date, cre-
ated_by=request.user)
            days.append((next_month_date, tasks))
    # Формируем контекст для передачи в шаблон
```

```
    context = {
        'days': days,
        'past_month': month_for_date[prev_month.month - 1],
        'current_month': month_for_date[month - 1],
        'future_month': month_for_date[next_month.month - 1],
        'month': month,
        'year': year,
        'month_name': month_list[month - 1],
        'prev_year': prev_month.year,
        'prev_month': prev_month.month,
        'next_year': next_month.year,
        'next_month': next_month.month,
    }
    return render(request, 'class_calendar/calendar_page.html', context)
```

Листинг 2 – Класс формы для добавления отчета

```
class New_lessonForm(forms.ModelForm):
    date = forms.DateField(
        label='Дата проведения занятия',
        widget=forms.DateInput(format='%Y-%m-%d', attrs={'type': 'date'}))
    start_time = forms.TimeField(
        label='Время начала занятия',
        widget=forms.TimeInput(attrs={'type': 'time'}))
    end_time = forms.TimeField(
        label='Время окончания занятия',
        widget=forms.TimeInput(attrs={'type': 'time'}))
    student_name = forms.ModelChoiceField(
        queryset=Students.objects.none(),
        widget=forms.Select(attrs={
            'hx-get': 'load_subjects/',
            'hx-target': '#id_subject'
        }),
        empty_label='Не выбрано', label='Ученик')
    subject = forms.ModelChoiceField(
        queryset=Subjects.objects.none(),
        widget=forms.Select(attrs={
            'hx-get': 'load_topics/',
            'hx-target': '#id_topic_name',
            'hx-trigger': 'change'
        }),
        empty_label='Не выбрано', label='Предмет')
    topic_name = forms.ModelChoiceField(
        queryset=Calendar_thematic_plan.objects.none(),
        widget=forms.Select(attrs={
            'hx-get': '/new_lesson/load_info/',
            'hx-target': '#skills-container',
            'hx-trigger': 'change'
        }),
        empty_label='Не выбрано', label='Тема')
    skills = forms.CharField(
        required=True,
        widget=forms.HiddenInput(attrs={'id': 'id_skills_hidden'}),
        label='Навыки'
    )
    homework = forms.CharField(
        widget=forms.Textarea(attrs={'rows': '4', 'style': 'resize:
none;'}),
        label='Домашнее задание', empty_value='Домашнее задание напишите'
    )
```

```
    class Meta:
        model = New_lesson
        fields = [
            'date',
            'start_time',
            'end_time',
            'student_name',
            'subject',
            'topic_name',
            'skills',
            'homework'
        ]
    def __init__(self, *args, **kwargs):
        user = kwargs.pop('user', None)
        print(f"Пользователь: {user}")
        super(New_lessonForm, self).__init__(*args, **kwargs)
        if user:
            self.fields['student_name'].queryset = Students.objects.fil-
ter(created_by=user)
            print(f"Студенты: {self.fields['student_name'].queryset}")
        else:
            print('Пользователь не найден!')
        if 'date' in self.initial:
            print(f"self.initial['date']) = {str(self.initial['date'])}")
            self.fields['date'].initial = self.initial['date']
            print(f"self.fields['date'].initial =
{str(self.fields['date'].initial)}")
        if 'start_time' in self.initial:
            self.fields['start_time'].initial = self.initial['start_time']
        if 'end_time' in self.initial:
            self.fields['end_time'].initial = self.initial['end_time']
        if 'student_name' in self.initial:
            self.fields['student_name'].initial = self.initial['stu-
dent_name']
        if 'student_name' in self.data:
            try:
                student_id = int(self.data.get('student_name'))
                subjects = Students.objects.get(id=student_id).sub-
ject.all()
                self.fields['subject'].queryset = subjects
            except (ValueError, TypeError, Students.DoesNotExist):
                pass
        if 'subject' in self.data:
            try:
                subject_id = int(self.data.get('subject'))
                topics = Calendar_thematic_plan.objects.filter(sub-
ject_id=subject_id)
                self.fields['topic_name'].queryset = topics
            except (ValueError, TypeError, Calendar_thematic_plan.DoesNo-
tExist):
                pass
        if 'topic_name' in self.data:
            try:
                topic_id = int(self.data.get('topic_name'))
                topic = Calendar_thematic_plan.objects.get(id=topic_id)
                self.initial['skills'] = topic.skills
            except (ValueError, TypeError, Calendar_thematic_plan.DoesNo-
tExist):
                pass
```

Листинг 3 – Исходный код шаблона страницы «Расписание»

```
{% extends 'main/layout_1.html' %}
{% load static %}
{% block head %}
    <link rel="stylesheet" type="text/css" href="{% static 'class_calen-
dar/css/calendar_page.css' %}">
{% endblock %}
{% block title %}Tutor Work Space - Расписание{% endblock %}
{% block title-page %}Расписание{% endblock %}
{% block content %}
<div class="calendar-container">
    <div class="navigation">
        <a href="{% url 'calendar_with_date' year=prev_year
month=prev_month %}" class="nav-arrow left-arrow"></a>
        <h3 class="month-name">{{ month_name }} {{ year }} года</h3>
        <a href="{% url 'calendar_with_date' year=next_year
month=next_month %}" class="nav-arrow right-arrow"></a>
    </div>
    <table>
        <tr>
            <th>Понедельник</th>
            <th>Вторник</th>
            <th>Среда</th>
            <th>Четверг</th>
            <th>Пятница</th>
            <th>Суббота</th>
            <th>Воскресенье</th>
        </tr>
            {% for day, tasks in days %}
                {% if day.weekday == 0 %}
                    <tr>
                {% endif %}
                    {% if day.month < month %}
                        <td class="day past">
                            <strong class="date">{{ day|date:"d" }} {{
past_month }}</strong><br>
                            <div class="task-container">
                                {% for task in tasks %}
                                    <div class="task past">
                                        {{ task.start_time|date:"H:i" }} -
{{ task.end_time|date:"H:i" }}<br>
                                        <a href="{% url 'task_de-
tail_or_create' task.id %}">{{ task.student.student_name }}</a>
                                    </div>
                                {% endfor %}
                            </div>
                        </td>
                    {% endif %}
                    {% if day.month == month %}
                        <td class="day">
                            <strong class="date">{{ day|date:"d" }} {{ cur-
rent_month }}</strong><br>
                            <div class="task-container">
                                {% for task in tasks %}
                                    <div class="task">
                                        {{ task.start_time|date:"H:i" }} -
{{ task.end_time|date:"H:i" }}<br>
                                        <a href="{% url 'task_de-
tail_or_create' task.id %}">{{ task.student.student_name }}</a>
                                    </div>
                                {% endfor %}
```

Окончание листинга 3 приложения Г

```
                            </div>
                        </td>
                    {% endif %}
                    {% if day.month > month %}
                        <td class="day future">
                            <strong class="date">{{ day|date:"d" }} {{ fu-
ture_month }}</strong><br>
                            <div class="task-container">
                                {% for task in tasks %}
                                    <div class="task future">
                                        {{ task.start_time|date:"H:i" }} -
{{ task.end_time|date:"H:i" }}<br>

                                        <a href="{% url 'task_de-
tail_or_create' task.id %}">{{ task.student.student_name }}</a>
                                    </div>
                                {% endfor %}
                            </div>
                        </td>
                    {% endif %}
                {% if day.weekday == 6 %}
                    </tr>
                {% endif %}
            {% endfor %}
    </table>
</div>
{% endblock %}
```

Листинг 4 – Исходный код стилей страницы расписания

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
.calendar-container {
    width: 86%;
    margin-left: 7%;
    margin-top: 2%;
    position: relative;
}
.navigation {
    display: flex;
    vertical-align: center;
    justify-content: space-between;
    margin-bottom: 10px;
}
.nav-arrow {
    text-decoration: none;
    font-size: 24px;
    color: #777;
    width: 24px;
    height: 24px;
    display: flex;
    justify-content: center;
    align-items: center;
}
.nav-arrow.left-arrow::before {
    content: '\2190'; /* left arrow */
}
```

```
.nav-arrow.right-arrow::before {
    content: '\2192'; /* right arrow */
}
table {
    table-layout: fixed;
    width: 100%;
    border-collapse: collapse;
    border-spacing: 0;
    border: 15px solid rgba(207, 211, 235, 1);
    border-top: none;
}
table th {
    vertical-align: middle;
    color: #777777;
    text-align: center;
    font-weight: bold;
    padding: 10px 5px;
    background: rgba(207, 211, 235, 1);
    border-bottom: 5px solid rgba(207, 211, 235, 1);
}
table tr th  {
    font-size: 16px;
}
table tr .date{
    font-size: 15px;
    color: #777777;
}
table td {
    font-size: 15px;
    background-color: #fff;
    text-align: left;
    padding: 10px 5px;
    border: 1px solid rgba(207, 211, 235, 1);
    height: 117px;
}
table td.day {
    border: 1px solid rgba(207, 211, 235, 1);
    vertical-align: top;
}
table td a {
    color: black;
    font-size: 15px;
    font-weight: 500;
    text-decoration: none;
    box-sizing: border-box;
}
.task-container {
    top: 5px;
    display: grid;
    gap: 5px; /* Расстояние между строками */
}
.task {
    background-color: rgba(207, 211, 235, 0.5);
    padding: 5px;
    border-radius: 5px;
}
```

## Приложение Д. Схема базы данных

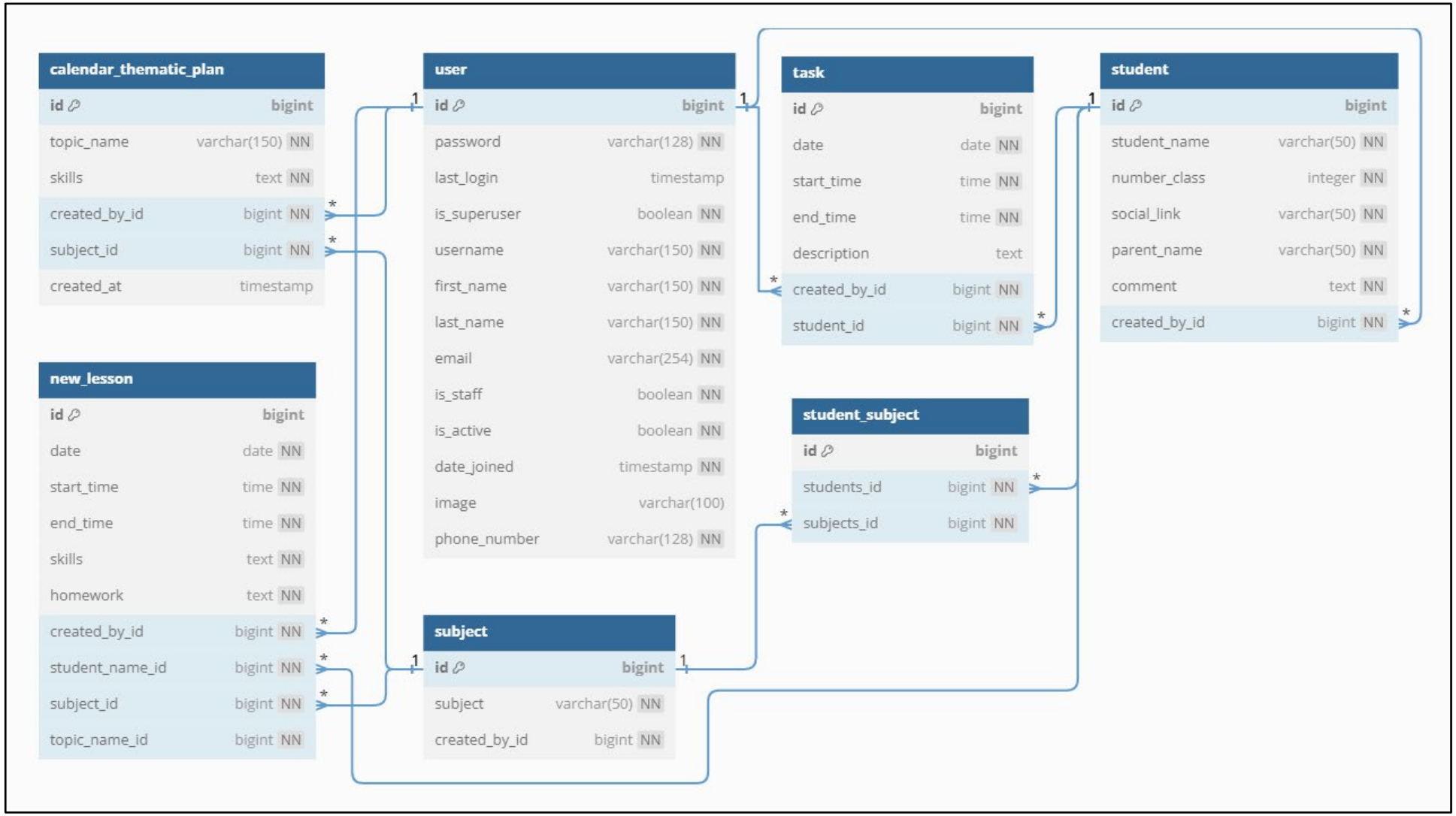


Рисунок 5 – Схема базы данных веб-приложения

**Приложение Е. Страница «Расписание»**

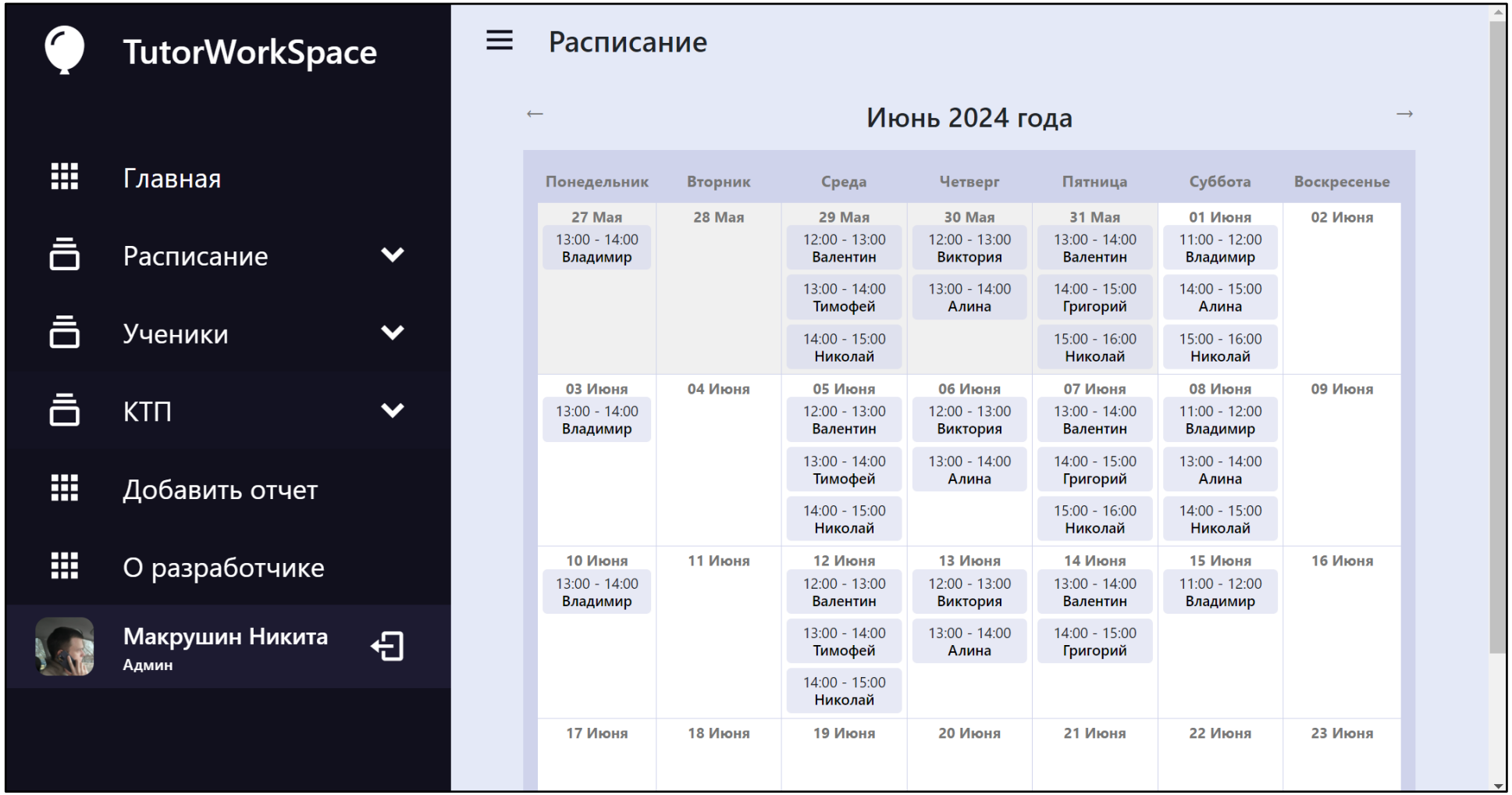


Рисунок 6 – Страница «Расписание»