МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

**федеральное государственное бюджетное образовательное учреждение высшего образования**

**«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**

**филиал ФГБОУ ВО «РГГМУ» в г. Туапсе**

**Кафедра «Экономики и управления на предприятии природопользования»**

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

**(бакалаврская работа)**
**по направлению подготовки 09.03.03 Прикладная информатика**
**(квалификация – бакалавр)**

**На тему Разработка системы автоматического формирования расписания в образовательной организации с использованием веб-технологий»**

**Исполнитель** Чередниченко Давид Владимирович

**Руководитель** к.т.н., Попов Николай Николаевич

**«К защите допускаю»**

**Руководитель кафедры** _____

кандидат экономических наук

Майборода Евгений Викторович

«20» 01 2025 г.

Филиал Российского государственного гидрометеорологического университета в г. Туапсе
НОРМОКОНТРОЛЬ ПРОЙДЕН
«17» 01 2025 г.
ПОДПИСЬ РАСШИФРОВКА ПОДПИСИ

Туапсе
2025

ОГЛАВЛЕНИЕ

# Введение

Разработка систем для автоматического составления расписания в образовательных организациях является актуальной и важной задачей в сфере современных информационных технологий. Веб-системы обладают преимуществом быстрого роста и легко поддаются обновлениям без необходимости установки специального программного обеспечения на устройства пользователей. Это позволяет оперативно внедрять изменения и обновлять систему.

Основным результатом работы стало создание системы автоматического формирования расписания.

Область применения данной системы – автоматизация процессов составления расписания в учебных заведениях.

Эффективность разработки заключается в ускорении процесса и автоматизации формирования расписания.

Выбранная тема выпускной квалификационной работы крайне актуальна в свете внедрения собственного программного обеспечения в образовательные учреждения. Разработка такого продукта позволит заложить начальный функционал, который сможет постепенно развиваться в полноценную систему, отвечающую всем потребностям учебного заведения.

Объектом данного исследования выступает высшее учебное заведение. В выпускной квалификационной работе используются различные методы, включая теоретические, экспериментальные и моделирования.

Предметом исследования является повышение производительности системы автоматического формирования расписания и добавление нового функционала путём доработки кода полученного продукта.

Целью данной работы является создание системы автоматического формирования расписания для высших учебных заведений с применением веб-технологий. Для достижения этой цели необходимо решить ряд задач:

– проанализировать технологии веб-программирования и анализ,

– проанализировать фреймворки и языки программирования,

– описать алгоритмы реализации,

– привести сравнительный анализ и выбор подхода, языка программирования,

– привести сравнительный анализ и выбор фреймворка, алгоритма реализации,

– спроектировать UML-диаграммы,

– составить блок-схему алгоритма,

– описать работу системы и разработать веб-систему автоматического формирования расписания,

– провести тестирование разработанной системы.

Работа включает введение, три раздела, заключение, список использованных источников и приложения.

В первом разделе анализируются технологии веб-программирования, приводится обзор и сравнение аналогичных решений, описываются выбранные фреймворк и язык программирования, а также алгоритмы реализации. Анализируется объект практики – университет, и подводятся итоги.

Во втором разделе представлены UML-диаграммы, описано функционирование системы с помощью блок-схем алгоритмов, осуществлён выбор среды разработки и дано описание работы системы автоматического формирования расписания, а также подведены итоги.

Третий раздел посвящён созданию баз данных для Django, разработке проекта на Django, описывается процесс реализации «жадного алгоритма» и «генетического алгоритма», а также проектирование интерфейса и тестирование системы с выводами по разделу.

В заключении подведены итоги выполненной работы, сделаны окончательные выводы и представлен список использованных источников.

# 1 Аналитическая часть системы автоматического расписания

## 1.1 Анализ технологий программирования и выбор подхода к программированию

Технология программирования включает в себя набор методов и инструментов, применяемых при разработке программного обеспечения. В настоящее время существуют две основные формы приложений: классические и веб-приложения. Классические приложения требуют установки на устройство для их использования и могут быть разработаны на различных языках программирования, ориентируясь на конкретные функциональные потребности. В отличие от них, веб-приложения имеют свои преимущества: они функционируют через интернет, что позволяет получить к ним доступ с любого устройства, имеющего подключение к сети. При этом, такие приложения не требуют установки, их можно запустить на любом гаджете, имеющем доступ к интернету [5].

Веб-разработка подразумевает использование двух ключевых аспектов: frontend и backend. Каждый из них выполняет свою роль в создании полноценного веб-приложения или сайта:

Frontend отвечает за визуальную часть, то есть за внешний вид и взаимодействие с пользователем. Для его реализации используются технологии HTML, CSS и JavaScript, а также различные фреймворки и библиотеки, такие как React.js, Angular и Vue.js.

Backend фокусируется на серверной части, обеспечивая обработку клиентских запросов, управление данными и бизнес-логикой. Здесь применяются языки программирования, такие как Python, PHP, Java и Ruby [10].

Создание собственного приложения требует четкого определения этапов разработки. На первом этапе формулируются требования, касающиеся функциональности, архитектуры, дизайна и других аспектов проекта. Затем осуществляется проектирование системы, на котором определяются основные

компоненты, структура базы данных и интерфейс пользователя [25].

После проектирования наступает этап разработки системы и выбора необходимых технологий, таких как упомянутые языки программирования, фреймворки и библиотеки, упрощающие процесс. Важным шагом является тестирование системы для обеспечения её работоспособности, безопасности и производительности, а также для выявления возможных недостатков [4].

Завершив тестирование и устранив недостатки, систему можно развернуть и запустить. Развёртывание может происходить как на собственном сервере компании, так и в облачных хостинговых сервисах. После запуска системы необходимо следить за её работой и поддерживать актуальность, чтобы гарантировать постоянное соответствие потребностям пользователей [14].

Для эффективной разработки веб-системы крайне важно владеть принципами работы интернета, знать основы работы с базами данных и знать о мерах безопасности. Команда разработчиков должна иметь опыт работы с веб-технологиями, что упрощает процесс реализации проекта. Также важно учитывать масштабируемость, производительность и безопасность системы. Масштабируемость системы подразумевает её возможность эффективно функционировать при увеличении числа пользователей или объёма данных, используя методы распределённых вычислений, кэширования и масштабирования [9].

Производительность веб-приложений является критически важной для обеспечения быстрого отклика и загрузки страниц. Оптимизация кода, использование CDN и сжатие ресурсов — все это помогает повысить производительность и улучшить пользовательский опыт.

Безопасность также занимает значимое место в процессе разработки веб-систем. Защита от угроз, а также важность методов шифрования, аутентификации и контроля доступа не могут быть неучтёнными [4].

Важно понимать, что разработка веб-системы — это сложный процесс, требующий тщательной организации и профессионализма на всех этапах

проекта. В условиях быстрого развития технологий и изменения пользовательских потребностей необходимо также уметь анализировать и сравнивать разные типы приложений для выявления их сильных и слабых сторон.

Сравним классические и веб-приложения по нескольким критериям: 1) Классические приложения, устанавливаемые локально, имеют более ограниченную доступность по сравнению с веб-приложениями, доступными через браузер на любом устройстве с интернетом. 2) Обновления нужны и для классических, и для веб-приложений, но в случае классических приложений пользователям придётся устанавливать обновления вручную, тогда как веб-приложения обновляются автоматически на сервере. 3) Масштабируемость веб-приложений значительно выше, поскольку их можно разворачивать на многих серверах для повышения производительности, в то время как классические приложения могут сталкиваться с ограничениями, связанными с конкретным устройством. 4) Удобство использования веб-приложений обычно выше, так как они доступны через браузер без необходимости установки дополнительных программ, тогда как классические приложения требуют установки и настройки на каждом устройстве.

Таким образом, если важна доступность, простота обновления и будущее расширение системы, выбор в пользу веб-приложений выглядит более привлекательным и перспективным [6], [11].

## 1.2 Анализ существующих аналогов

Рассмотрим несколько аналогов, используемых для разработки системы автоматического формирования расписания и их характеристики:

1. «1С: Автоматизированное составление расписания. Университет» - это система, которая предназначена для оптимизации процесса создания расписания занятий в университете. ПО позволяет автоматически распределять учебные занятия по аудиториям и времени, при этом учитывая занятость

преподавателей, предметы или группы студентов и т.д. Система позволяет сократить время, которое затрачивается на составление расписания, и минимизировать вероятность ошибок [18]. Также она обеспечивает возможность быстрого внесения изменений в расписание и мониторинг его актуальности. В итоге это помогает повысить эффективность управления учебным процессом и обеспечить оптимальное использование ресурсов университета. Рабочее окно программы представлено на рисунке 1.

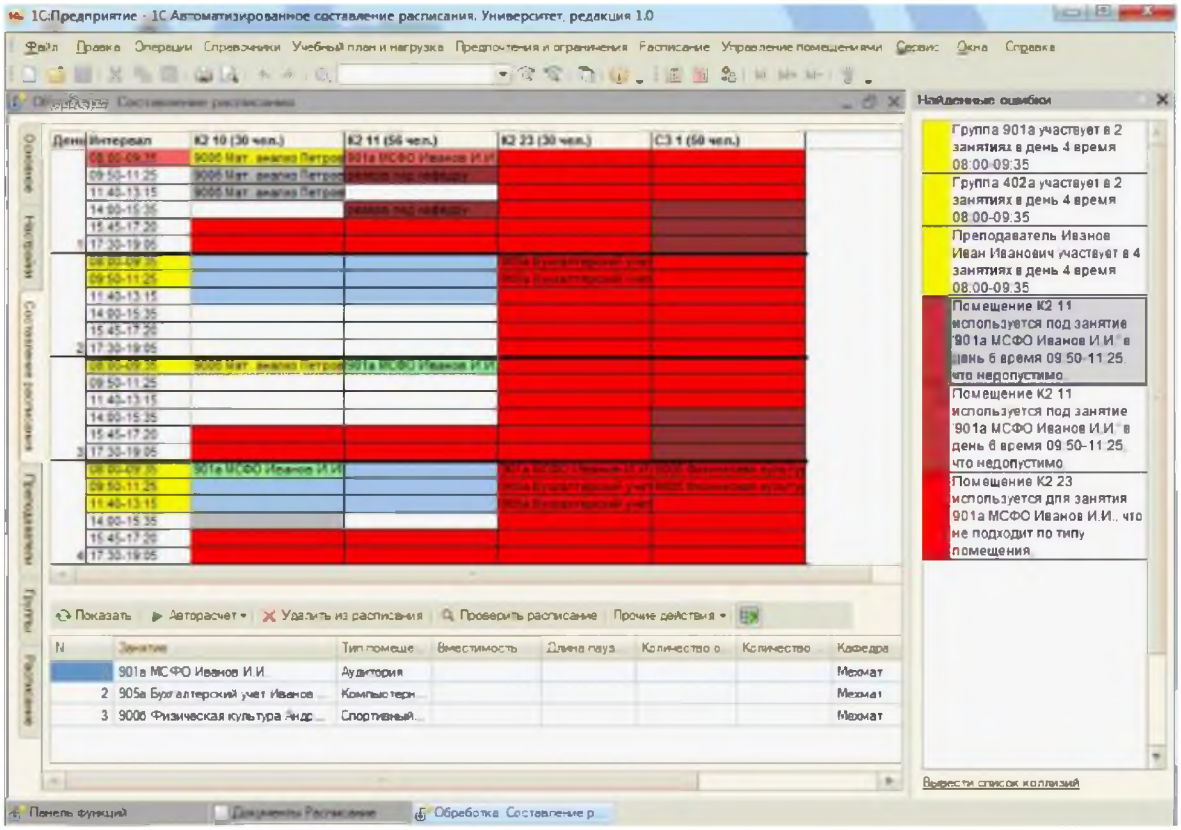


Рисунок 1 – Рабочее окно «1С: Автоматизированное составление расписания для университета»

Преимуществами программы «1С: Автоматизированное составление расписания. Университет» является:

– возможность быстро и точно создавать расписание занятий, учитывая все необходимые ограничения и предпочтения,

– оптимально используются ресурсы университета (аудитории, преподаватели, группы студентов), благодаря автоматическому распределению занятий по аудиториям и времени,

– программа позволяет быстро вносить изменения в расписание и адаптировать его под новые требования или ситуации,

– программа обеспечивает возможность контролировать актуальность расписания и оперативно реагировать на возможные проблемы или конфликты.

Недостатками программы «1С: Автоматизированное составление расписания. Университет»:

– требуется настройка всех параметров и ограничений, что может отразится на времени и специализированных знаний,

– точность и корректность расписания зависит от правильности введенных данных, а также от актуальности информации о доступности преподавателей, аудиторий и т.д.,

– в некоторых случаях программа не удовлетворяет все особенности конкретного учебного заведения или специфики учебного процесса.

Программа «1С: Автоматизированное составление расписания. Университет» имеет множество преимуществ значительно облегчающие и оптимизирующие процесс составления расписания занятий в университете. Но перед ее внедрением необходимо тщательно оценить все плюсы и минусы, чтобы принять обоснованное решение.

2. «Галактика Расписание учебных занятий» это система, которая предоставляет возможность создавать расписания с учетом факторов, таких как доступность преподавателей, аудиторий, учебных групп, предметов и других параметров. Программа автоматически учитывает выбранные предпочтения и ограничения, задающиеся пользователями, и строит оптимальное расписание, которое минимизирует конфликты и пересечения. Рабочее окно программы «Галактика Расписание учебных занятий» представлено на рисунке 2.

С помощью Галактики Расписание можно достаточно быстро и эффективно изменять расписание при необходимости, а также отслеживать актуальные изменения. Программа позволяет сохранять и архивировать расписания, делиться ими с другими пользователями и генерировать отчеты [17].

Преимущества системы «Галактики Расписание учебных занятий»:

– автоматизация сервиса значительно упрощает и ускоряет процесс составления расписания, освобождая время преподавателей и администраторов для других задач,

– гибкость сервиса и возможность настройки параметров позволяет учитывать специфические требования и предпочтения учебного заведения при составлении расписания,

– оптимизация сервиса помогает создать оптимальное расписание, учитывая факторы, такие как равномерное распределение нагрузки, минимизация конфликтов и перерывов между занятиями,

– экономия ресурсов сервиса автоматизация процесса планирования помогает сэкономить время и ресурсы, которые могли бы быть затрачены на ручное составление расписания.

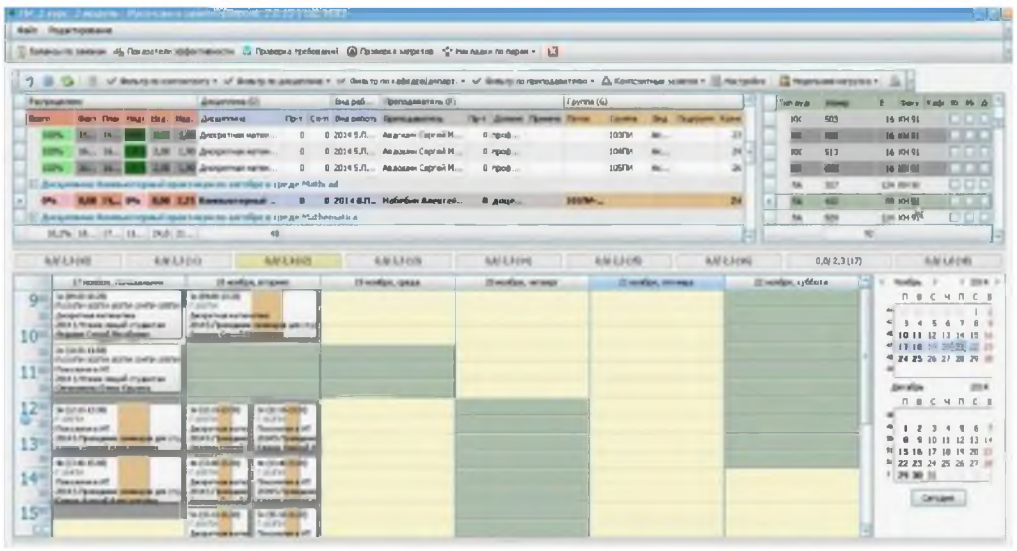


Рисунок 2 - Рабочее окно «Галактика Расписание учебных занятий»

Недостатки системы «Галактики Расписание учебных занятий»:

– зависимость от технических средств для использования сервиса необходим доступ к интернету и соответствующее программное обеспечение, что может ограничить доступность для некоторых пользователей,

– необходимость обучения персонала для эффективного использования сервиса может потребоваться обучение персонала, особенно в случае сложных

настроек и функциональности,

– ограниченные возможности ручного вмешательства в случае нестандартных ситуаций или изменений требуется возможность ручного вмешательства, которая может быть ограничена в автоматизированном сервисе,

– стоимость в зависимости от условий использования, стоимость сервиса изменяется, что может стать дополнительной финансовой нагрузкой для учебного заведения.

«Галактика Расписание учебных занятий» представляет собой полезный инструмент для оптимизации процесса составления расписания, однако следует учитывать плюсы и минусы при принятии решения об его использовании.

3. Система «АВТОРасписания» представляет собой инновационное программное обеспечение, разработанное специально для автоматизации процесса составления и управления расписанием занятий. Она позволяет быстро и эффективно создавать расписания на основе введенных данных о предметах, преподавателях, аудиториях и временных ограничениях [19]. Рабочее окно программы представлено на рисунке 3.

Имеется восемь основных модификаций программы для различных учебных заведений:

– AVTOR School - для средних общеобразовательных школ, лицеев и гимназий,

– AVTOR College - для колледжей, техникумов и профессиональных училищ,

– AVTOR Art College - для училищ искусства и культуры,

– AVTOR High School - для вузов (очная форма обучения),

– AVTOR High School Semestric - для вузов (заочная форма обучения),

– AVTOR M High School Semestric - для военных вузов,

– AVTOR Educational Centers - для учебных центров, УПК и ИПК,

– AVTOR High Shool Pro - для вузов с несколькими удаленными учебными корпусами с учетом времени переезда между ними (очная и заочная

формы обучения, сетевая версия).

Основными преимуществами системы являются:

- быстрота и удобство составления расписания. С помощью «АВТОРасписания» можно легко оптимизировать распределение занятий и избежать конфликтов в расписании,

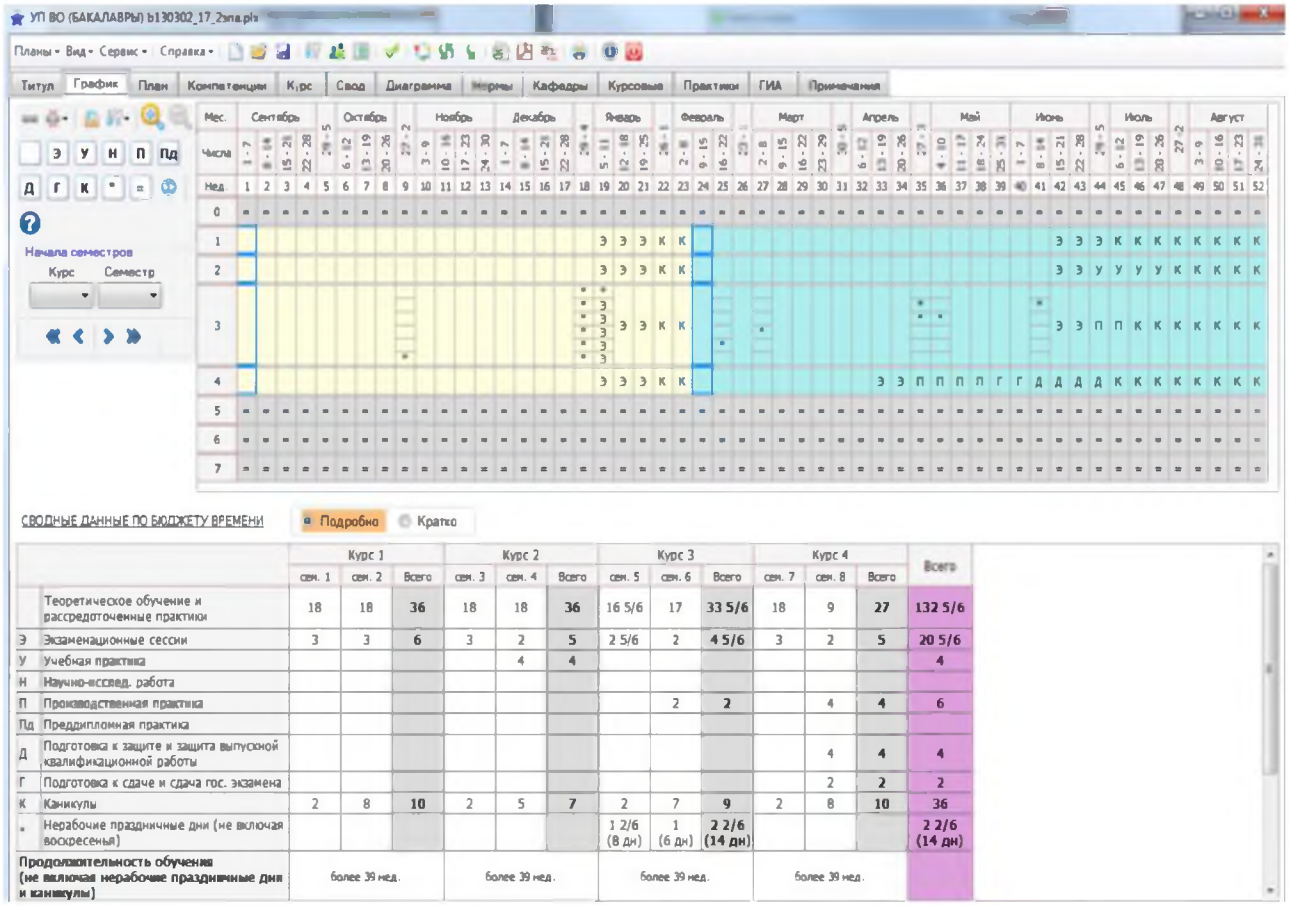



Рисунок 3 – Рабочее окно программы «АВТОРасписания»

– возможность автоматического сопровождения расписания в течение всего учебного года. Система учитывает изменения и позволяет оперативно вносить корректировки,

– возможность составления расписания для высшего и среднего образования одновременно,

– возможность ручного вмешательства,

– гибкость настроек, где пользователь может задать параметры для составления расписания в соответствии с конкретными потребностями.

Однако у системы «АВТОРасписания» есть и некоторые недостатки:

– необходимость внедрения и обучения персонала. Для эффективного использования системы требуется подготовка сотрудников и время на освоение функционала,

– возможные технические проблемы. Как и любое программное обеспечение, «АВТОРасписания» может столкнуться с сбоями или ошибками, что потребует дополнительных усилий для исправления.

В целом, система «АВТОРасписания» представляет собой полезный инструмент для оптимизации процесса составления расписания занятий, однако её эффективность зависит от правильной настройки и использования.

4. БИТ.ВУЗ.Расписание – это онлайн-сервис, предназначенный для составления расписания занятий в вузах и других образовательных учреждениях. С помощью этого сервиса можно автоматически составить расписание на основе заданных параметров и ограничений, таких как доступность аудиторий, рабочие нагрузки преподавателей, предпочтения студентов и другие факторы [16]. Рабочее окно программы представлено на рисунке 4.

БИТ.ВУЗ.Расписание обладает приемлемым интерфейсом, который позволяет быстро и эффективно создавать расписание занятий, а также вносить изменения при необходимости.

Сервис предлагает широкий спектр функциональных возможностей для оптимизации процесса планирования учебного процесса и учета всех необходимых параметров, также поддерживает интеграцию с другими системами 1С управления учебным процессом, что позволяет оперативно вносить коррективы в расписание, адаптируясь к изменениям в учебном процессе и требованиям образовательного учреждения.

Плюсы использования сервиса БИТ.ВУЗ.Расписание включают в себя:

– автоматическое составление расписания на основе заданных параметров и ограничений, что экономит время и упрощает процесс планирования,

– возможность учета различных факторов, таких как доступность

аудиторий, рабочие нагрузки преподавателей, предпочтения студентов и другие, что способствует оптимизации расписания,

– широкий спектр функциональных возможностей для оптимизации процесса планирования учебного процесса.

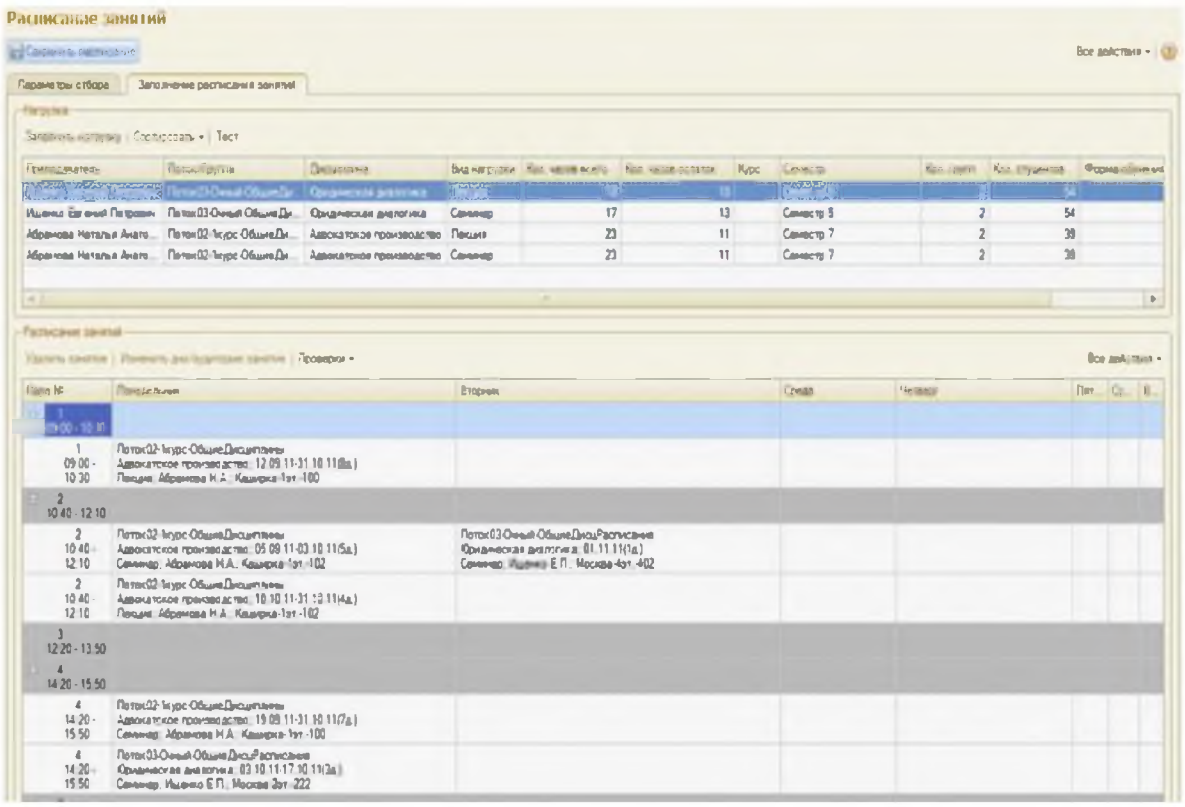



Рисунок 4 – Рабочее окно программы «БИТ.ВУЗ.Расписание»

Однако у сервиса БИТ.ВУЗ.Расписание есть и некоторые минусы:

– не всегда возможно учесть все особенности конкретного учебного заведения или специфические требования к расписанию,

– некоторые пользователи могут испытывать сложности с настройкой параметров и пониманием всех функций сервиса,

– возможны сбои в работе сервиса или проблемы с его доступностью, что может затруднить составление расписания в нужные сроки.

Одной из важных задач является составление расписания занятий, которое должно быть не только оптимизированным с точки зрения ресурсов, но и удобным для всех участников образовательного процесса.

Для решения этой задачи существует некое разнообразие программных решений, каждое из которых обладает своими особенностями и

преимуществами.

В данном сравнительном анализе мы рассмотрим три популярные системы отображения и составления расписаний: «Галактика Расписание учебных занятий», «БИТ.ВУЗ.Расписание», «АВТОРасписания» и «1С: Автоматизированное составление расписания для университета».

Сравним эти системы по ряду критериев, таких как цена, функциональность, удобство и простота использования, интеграция с другими системами и поддержка, и обновления. Сравнение приведено в таблице 1.

Таблица 1 – Сравнение систем автоматического составления расписания по критериям

| Название системы<br>Критерии ↓ | «1С: Автоматизированное составление расписания для университета» | «Галактика Расписание учебных занятий» | «АВТО Расписания» | «БИТ.ВУЗ.Расписание » |
|---|---|---|---|---|
| Цена | Очень высокая | Высокая | Средняя | Очень высокая |
| Функциональность | Высокая | Высокая | Высокая | Высокая |
| Удобство и простота использования | Требует специальных навыков и опыта | Средняя | Довольно простая | Требует специальных навыков и опыта |
| Интеграция с другими системами | С другими системами 1С | Частично | Есть | С другими системами 1С |
| Поддержка и обновления | Есть | Есть | Есть | Есть |

Исходя из проведенного сравнения, наилучшим решением для управления расписанием занятий в образовательном учреждении является разработка собственной системы формирования расписания, так как готовые решения не отвечают всем требованиям организации.

Создание индивидуальной системы позволит адаптировать функционал точно под требования и специфику учебного заведения. Данный подход позволит обеспечить максимальную плавность в управлении процессами составления расписания и учетом особенностей учебного плана, преподавательского состава и инфраструктуры учреждения.

## 1.3 Анализ и выбор фреймворков и языка программирования

Существует огромное разнообразие языков программирования, каждый из которых имеет свои особенности и применения. Технологии и программы для веб-разработки делятся на категории в зависимости от области применения. Некоторые из наиболее популярных языков, используемых для создания интерактивных веб-приложений и динамических сайтов, представлены в таблице 2:

Таблица 2 – Языки программирования и их характеристика

| Язык программирования | Описание |
|---|---|
| HTML (HyperText Markup Language) | в его основе лежит концепция разметки, позволяющая указать браузеру, как содержимое будет отображаться на экране при создании структуры веб-страниц, определяя элементы контента, такие как заголовки, параграфы, списки, изображения и ссылки; |
| CSS (Cascading Style Sheets) | основная роль при создании веб-сайтов и приложений заключается в том, чтобы обеспечить привлекательный внешний вид, а также упростить процесс обновления и изменения страниц; |
| JavaScript | язык программирования, который добавляет интерактивность на веб-страницы и помогает обрабатывать события, изменять содержимое страницы динамически, валидировать данные; |
| PHP | серверный язык программирования, позволяющий разработчику взаимодействовать с базами данных, обрабатывать формы, генерировать персонализированный контент для пользователей и обеспечивать функциональность сайта |
| Python | высокоуровневый интерпретируемый язык программирования, который известен своей сравнительной легкостью, а также имеет множество фреймворков упрощающими и ускоряющими процесс создания веб-сайтов и приложений |

При сравнении языков программирования для разработки системы автоматического расписания с можно выделить такие критерии как «Простота и понятность», «Функциональность», «Возможность работы с базами данных», «Наличие библиотек и фреймворков», «Поддержка сообществом» и «Производительность», анализ которых находится в таблице 3.

Исходя из приведённой ниже таблицы, можно утверждать, что подходящим языком является python. Он достаточно прост и обладает понятным синтаксисом, кроссплатформенностью и универсальностью по сравнению с другими языками для веб-программирования, что ускоряет процесс разработки и позволяет обеспечить удобство при поддержке проекта [13].

Таблица 3- Анализ языков программирования

| Язык программирования<br>Критерии ↓ | HTML | CSS | JavaScript | PHP | Python |
|---|---|---|---|---|---|
| Простота и понятность | + | + | - | +/- | + |
| Функциональность | - | - | + | + | + |
| Возможности работы с базами данных | - | - | + | + | + |
| Наличие библиотек и фреймворков | - | - | + | + | + |
| Поддержка сообществом | - | - | + | + | + |
| Производительность | + | + | + | +/- | + |

Выбор фреймворка может оказать значительное влияние на успешность проекта, поэтому важно тщательно рассмотреть все доступные варианты и выбрать наиболее подходящий для достижения поставленных целей. Фреймворки предлагают готовые инструменты, библиотеки и структуры, что позволяет разработчикам сосредоточиться на бизнес-логике приложения, минимизируя необходимость написания повторяющегося кода.

Ниже представлено сравнение различных фреймворков по нескольким критериям. Уровень абстракции оценивает степень абстракции, которую фреймворк предоставляет, однако это может ограничить возможности его настройки. Скорость разработки рассматривает, насколько фреймворк способен ускорить процесс создания веб-приложений благодаря наличию готовых инструментов, шаблонов и функций, что позволяет снизить затраты ресурсов. Сообщество и экосистема анализируют активность сообщества пользователей и разработчиков, а также наличие сторонних инструментов, плагинов и библиотек, которые могут расширить функциональность фреймворка.

Масштабируемость затрагивает способность фреймворка адаптироваться к растущей нагрузки и поддерживать высокую производительность. Безопасность оценивает уровень защиты данных, предлагаемый фреймворком по умолчанию, а также возможности реализации мер по защите данных и пользователей от различного рода угроз.

Функциональность рассматривает предоставляемый фреймворком набор функций и модулей, а также возможности для добавления сторонних компонентов для расширения функционала [8]. Сравнение фреймворков по упомянутым критериям представлено в таблице 4.

Таблица 4 – Сравнение фреймворков

| Фреймворк→<br>Критерии ↓ | Django | Flask | Pyramid | CherryPy |
|---|---|---|---|---|
| Уровень абстракции | высокий | низкий | средний | низкий |
| Скорость разработки | высокая | средняя | средняя | высокая |
| Сообщество и экосистема | обширное | обширное | среднее | малое |
| Масштабируемость | отличная | высокая | высокая | средняя |
| Безопасность | высокая | высокая | высокая | средняя |
| Функциональность | обширная | обширная | средняя | средняя |

Благодаря обширному сообществу разработчиков, активно обновляемой документации и множеству сторонних библиотек и расширений, Django обеспечивает быструю и эффективную разработку, а также практически неограниченные возможности для роста и масштабирования приложений.

Преимущества использования Django для создания системы автоматического расписания включают: ORM (Object-Relational Mapping), который упрощает взаимодействие с базами данных и облегчает создание сложных запросов; встроенную админ-панель, позволяющую управлять данными приложения без необходимости написания дополнительного кода; масштабируемость, что делает его оптимальным выбором для разработки больших и сложных систем; наличие обширной документации и поддержки, что содействует самостоятельному изучению; а также разнообразие библиотек, которые значительно ускоряют процесс разработки системы автоматического расписания.

## 1.4 Описание и выбор алгоритмов реализации

Создание расписания для образовательных учреждений является сложной задачей, которая требует учета множества факторов. С помощью различных методов и алгоритмов можно эффективно решать задачи по составлению расписания, принимая во внимание особенности организации и запросы пользователей. Ниже представлены алгоритмы, применяемые для автоматизации процесса составления расписаний:

Генетические алгоритмы основаны на принципах естественного отбора и эволюции, что делает их подходящими для оптимизации, включая задачи по составлению расписаний в образовательных учреждениях. Процесс работы такого алгоритма начинается с инициализации популяции случайно сгенерированных расписаний, после чего происходит оценка их эффективности. Далее осуществляется отбор лучших расписаний, последующий процесс включает скрещивание и мутацию. Этот цикл повторяется до тех пор, пока не будет достигнут удовлетворительный результат. Генетические алгоритмы эффективно исследуют пространство возможных решений, находя оптимальные варианты. Схема работы генетического алгоритма представлена на рисунке 5.


Рисунок 5 – Схема работы генетического алгоритма

Алгоритмический подход включает сбор информации о занятости преподавателей, доступности аудиторий, предметах и временных ограничениях. Основные шаги включают определение целей и требований, создание алгоритма, тестирование на реальных данных, оптимизацию и интеграцию в систему управления расписанием, эти шаги обеспечивает

гибкость и адаптируемость расписания, учитывая различные ограничения и требования образовательного учреждения.

Метод снижает количество ошибок и конфликтов, оптимизируя использование ресурсов. Подход требует тщательного планирования и регулярной проверки корректности данных, чтобы обеспечить надежность и точность конечного расписания, адаптированного к динамическим изменениям в образовательной среде

Схема работы алгоритмический подход представлена на рисунке 6.

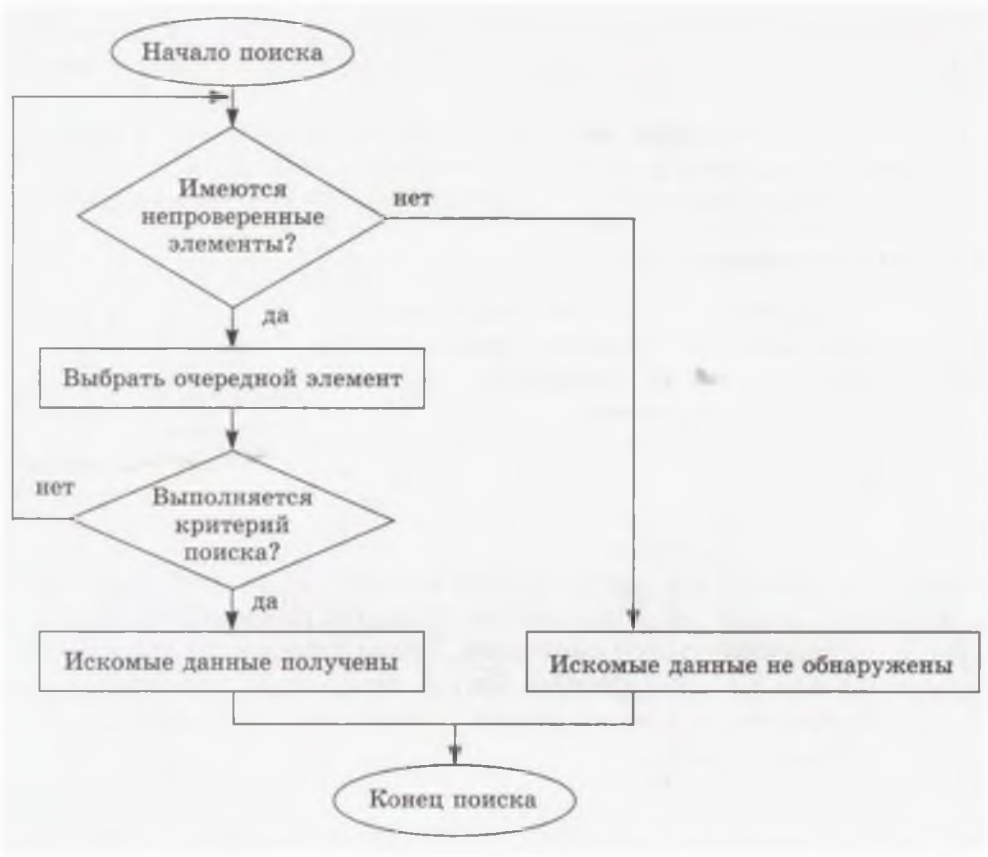



Рисунок 6 – Алгоритмический подход

Интерактивный подход включает разработку интерфейса для ввода данных о преподавателях, аудиториях, группах студентов и других параметрах. Пользователи могут устанавливать приоритеты, редактировать расписания и анализировать результаты. Этот метод позволяет учитывать индивидуальные предпочтения и повышает удовлетворенность пользователей.

Жадный алгоритм выбирает на каждом шаге локально оптимальное решение, надеясь на глобально оптимальный результат. Этот метод прост в

реализации и обладает высокой скоростью работы, но может не учитывать долгосрочные последствия и не справляться с глобальной оптимизацией.

Схема работы жадного алгоритма представлена на рисунке 7.

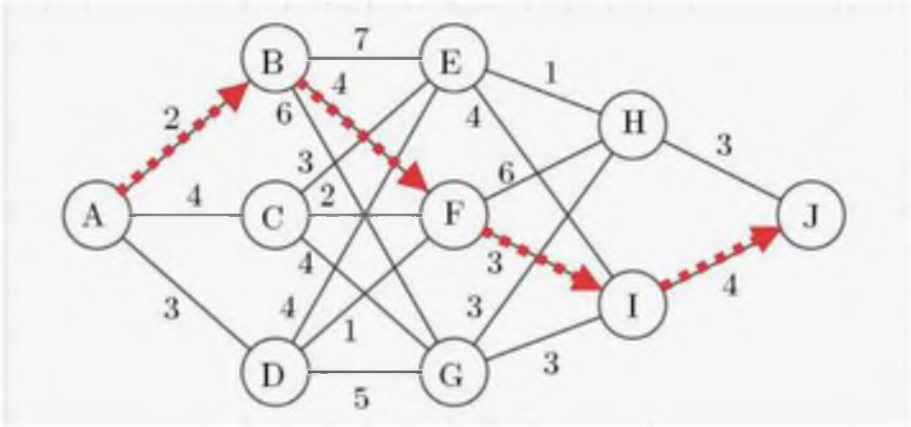



Рисунок 7 – Схема работы жадного алгоритма

Комбинированный подход объединяет различные методы и техники для достижения оптимальных результатов. Он включает применение нескольких алгоритмов для решения отдельных частей задачи, учитывает различные типы ограничений и позволяет пользователям взаимодействовать с системой. Этот метод сочетает преимущества различных подходов, обеспечивая более эффективные решения. Пример комбинированного похода приведён на рисунке 8.

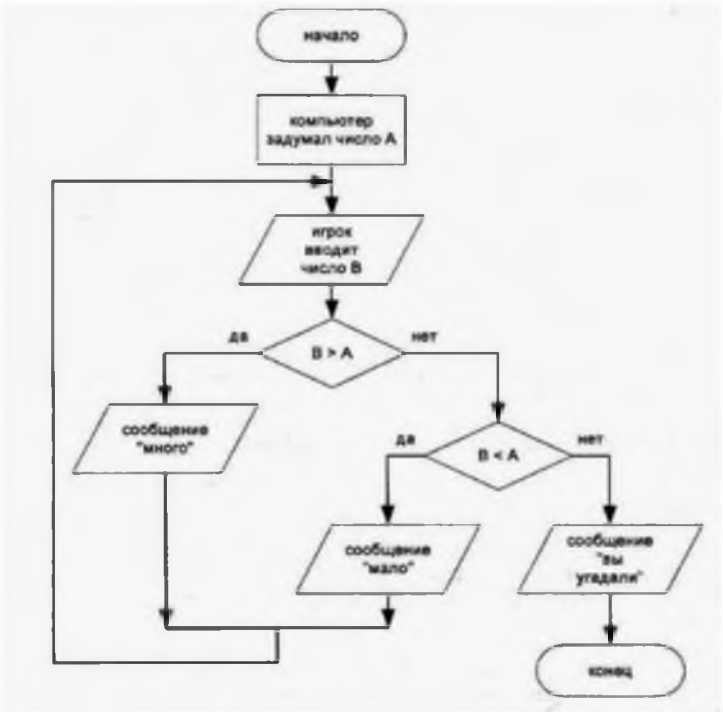



Рисунок 8 – Схема работы комбинированного подхода

Для сравнения подходов и алгоритмов можно отметить, что генетические алгоритмы эффективно находят оптимальные решения для сложных задач, однако они требуют значительных ресурсов. Алгоритмический подход отличается простотой реализации и точности управления процессом, но может быть недостаточно эффективным при решении сложно структурированных задач.

Интерактивный подход ориентирован на учет предпочтений пользователей, однако нуждается в их активном вовлечении. Жадный алгоритм характеризуется простотой и быстротой, но не всегда подходит для достижения глобально оптимальных решений. Комбинированный подход объединяет достоинства различных методов, что приводит к более эффективным решениям.

Генетические алгоритмы и жадный алгоритм могут быть полезными на разных этапах построения и оптимизации расписания. Синтез жадного алгоритма с интерактивным подходом позволяет быстро разработать начальное расписание, которое впоследствии может быть откорректировано пользователями.

Для достижения эффективного формирования расписания в образовательных организациях целесообразно совмещать жадный алгоритм с интерактивным подходом, при этом используя генетические алгоритмы. Такой подход обеспечит учет ограничений и параметров при создании исходного расписания, а затем позволит его оптимизировать, что приведет к более рациональному распределению занятий и повышению качества образовательного процесса. Комбинированная методология увеличивает эффективность использования ресурсов и времени.

Внедрение системы автоматического формирования расписания снижает вероятность ошибок и конфликтов, обеспечивая более плавное и эффективное проведение занятий, а пользователи смогут легко взаимодействовать с системой, внося изменения в расписание в реальном времени в соответствии со своими требованиями.

## 1.5 Анализ объекта исследования

Объектом исследования является учебное заведение высшего образования.

Высшее учебное заведение (ВУЗ) - это образовательное учреждение, которое осуществляет подготовку высококвалифицированных кадров с высшим образованием. Вузы могут быть государственными и частными, они могут иметь филиалы и представительства в других населенных пунктах и странах. ВУЗ имеет устав и является автономным субъектом правовых отношений. В России ВУЗ должен иметь лицензию, которая дает право на образовательную деятельность. Чтобы иметь право выдавать выпускникам диплом государственного образца, ВУЗ должен быть аккредитованным. Виды вузов включают университеты, академии и институты, каждый из которых имеет свою специфику деятельности и направления подготовки специалистов

Расписание в высшем учебном заведении играет важную роль в организации учебного процесса и выполняет несколько ключевых функций:

Организация учебного времени: расписание помогает студентам и преподавателям планировать своё время. Оно определяет, когда и где проходят лекции, семинары, лабораторные занятия и другие виды учебных мероприятий. Это способствует эффективной работе и рациональному использованию ресурсов вуза.

Оптимизация нагрузки: с помощью расписания можно равномерно распределить учебные часы между различными дисциплинами и видами занятий, избегая перегрузки студентов и преподавателей. Это особенно важно при составлении расписаний для нескольких потоков или курсов одновременно.

Синхронизация занятий: в вузах часто проводятся занятия разных форматов – лекции, практические занятия, коллоквиумы и так далее. Расписание обеспечивает синхронизацию этих видов деятельности, чтобы студенты могли посещать все необходимые мероприятия без конфликтов по времени.

Учет аудиторного фонда: расписание учитывает наличие аудиторий, их вместимость и оборудование. Это позволяет эффективно использовать имеющиеся ресурсы и избегать проблем с нехваткой помещений для проведения занятий.

Контроль посещаемости: на основании расписания преподаватели могут контролировать посещаемость студентов, а также корректировать учебный процесс в случае необходимости.

Планирование внеучебных мероприятий: Расписание помогает учитывать время для проведения различных внеучебных мероприятий, таких как конференции, спортивные соревнования, культурные события и др., чтобы они не пересекались с учебными часами.

Обеспечение качества образования: Грамотно составленное расписание способствует созданию комфортных условий для обучения и повышает качество образовательного процесса, обеспечивая оптимальное сочетание теории и практики.

Таким образом, расписание является важным инструментом управления образовательным процессом в высших учебных заведениях, помогающим организовать работу студентов, преподавателей и всей инфраструктуры вуза.

## 1.6 Итоги первого раздела

После проведения анализа технологий веб-программирования можно подвести несколько ключевых выводов. Рассмотрено различие между классическими и веб-приложениями, а также подчеркнута значимость frontend и backend разработки для создания полнофункциональных веб-приложений. Обсуждено, как правильный выбор инструментов и методов разработки непосредственно влияет на качество и удобство использования конечного продукта, а также представлен ряд подходов к программированию.

Далее был проведен анализ и выбор фреймворков и языков программирования, таких как HTML, CSS, JavaScript, PHP, Python, Django,

React, Laravel и Vue.js. Установлено, что каждый из этих инструментов обладает своими уникальными преимуществами и сферами применения, и правильный выбор технологий помогает эффективно реализовывать проекты разной сложности.

Также были исследованы алгоритмы реализации, включая генетические методы, интерактивный подход, алгоритмический подход, жадный алгоритм и комбинированный подход. В результате был выбран один из подходящих алгоритмов для реализации.

Кроме того, был проанализирован объект практики – учебное заведение высшего образования. Выявлено, что методы обучения и подготовки специалистов в области информационных технологий и систем соответствуют современным требованиям рынка труда, что способствует успешной интеграции выпускников в профессиональную среду.

# 2 Проектирование системы автоматического расписания

## 2.1 Проектирование с помощью UML

Унифицированный язык моделирования (UML) предоставляет виды диаграмм, используемые для описания различных аспектов системы. Например, диаграмма классов позволяет описать структуру системы, определяя классы, их атрибуты и методы. Диаграмма вариантов использования помогает выделить основные сценарии использования системы. Диаграммы вариантов использования в UML иллюстрируют сценарии взаимодействия между пользователями и системой. Они демонстрируют, каким образом пользователи будут работать с системой для достижения определенных целей. Эти диаграммы включают актеров (пользователей или системы) и различные сценарии использования (use cases), которые представляют собой конкретные действия или функции системы. Каждый сценарий использования описывает, как актеры могут взаимодействовать с системой для выполнения определённых задач. Диаграммы вариантов использования (рисунок 9) помогают выделить ключевые функциональные требования, определить основных пользователей и понять, как система будет функционировать в реальных условиях.


Рисунок 9 – Диаграмма вариантов использования.

Данные диаграммы являются важными инструментами для анализа и проектирования систем, так как они позволяют наглядно представить сценарии использования и дают четкое представление о том, как система должна

функционировать с точки зрения пользователей.

Диаграмма последовательности позволяет показать взаимодействие различных компонентов системы во времени. «Использование UML при моделировании процесса создания расписания поможет улучшить понимание требований к системе, выделить ключевые компоненты и взаимосвязи между ними, а также обнаружить потенциальные проблемы еще на этапе проектирования[21]. Диаграммы последовательности в UML предназначены для визуализации взаимодействия между объектами или компонентами системы в определённой последовательности действий. Они показывают, как объекты обмениваются сообщениями и взаимодействуют друг с другом по времени. На диаграмме последовательности объекты представлены вертикальными линиями, а сообщения между ними изображаются стрелками, которые указывают направление передачи информации. Эти диаграммы позволяют структурировать порядок выполнения действий с целью анализа взаимодействий объектов в рамках конкретного сценария. Также диаграммы последовательности могут использоваться для выявления возможных проблем или для разработки важных улучшений в процессе проектирования системы. Информация иллюстрируется на рисунках 10-13 [21].

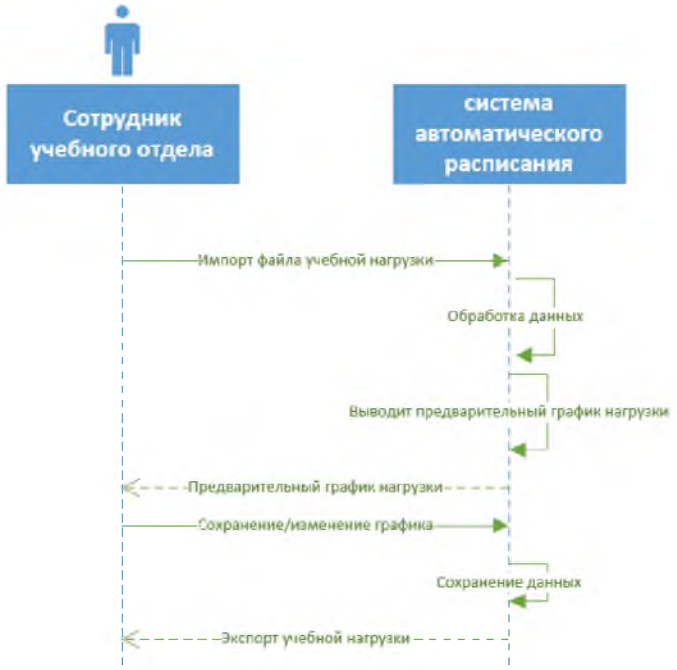



Рисунок 10 - Диаграмма последовательности для варианта использования «экспорт/импорт учебной нагрузки»

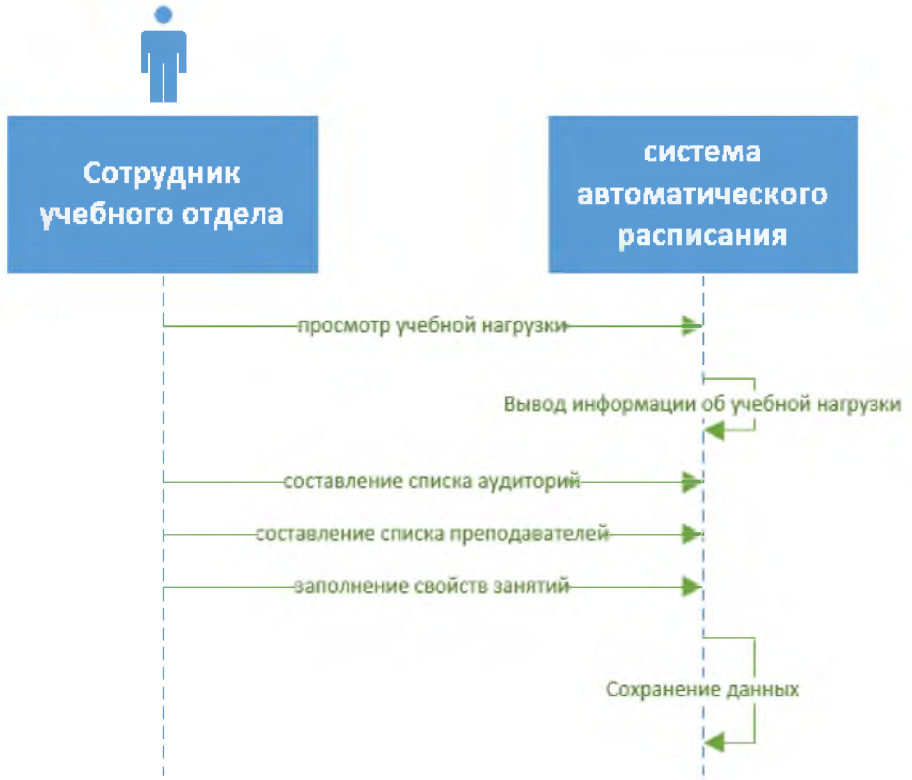



Рисунок 11 - Диаграмма последовательности для варианта использования «заполнение исходных данных»

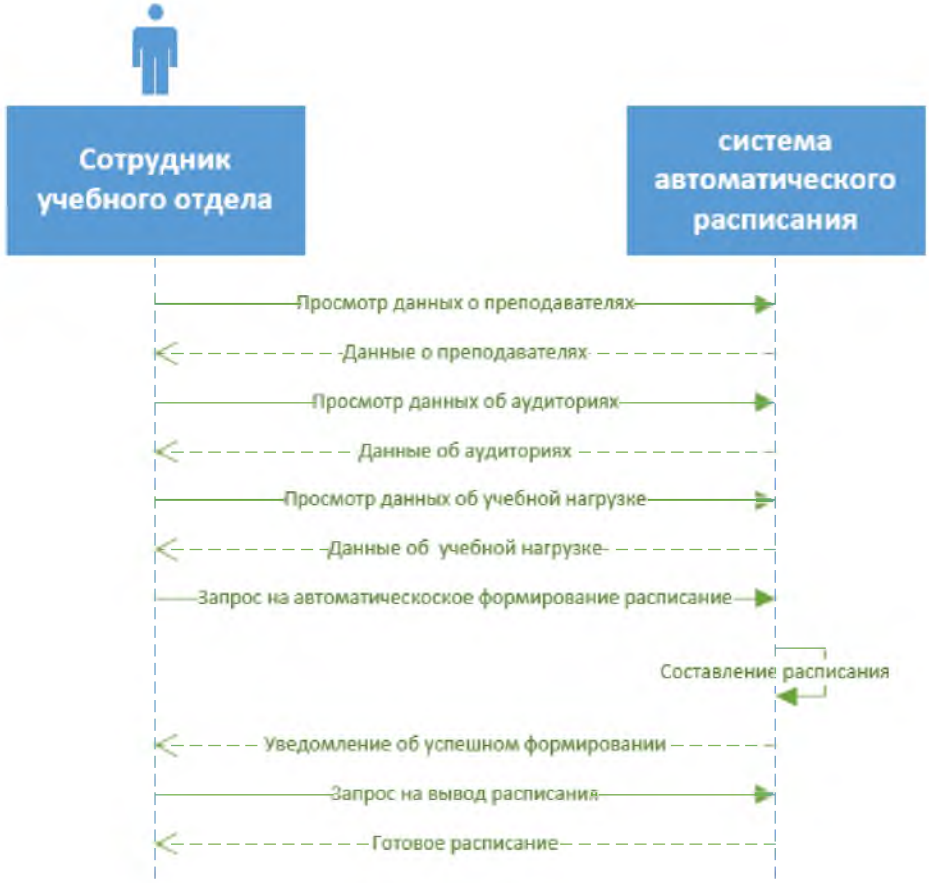



Рисунок 12 - Диаграмма последовательности для варианта использования «формирование расписания»


Рисунок 13 - Диаграмма последовательности для варианта использования «редактирование расписания»

## 2.2 Описание работы алгоритма с помощью блок-схемы

Блок-схема алгоритма представляет собой графическое отображение последовательности шагов для выполнения конкретной задачи или процесса. Она состоит из различных блоков, соединённых стрелками, которые указывают на порядок исполнения операций.

Во-первых, блок-схемы позволяют визуализировать структуру алгоритма или процесса, что делает информацию более доступной и понятной для аудитории.

Во-вторых, такие схемы упрощают анализ и отладку алгоритма; благодаря чёткой структуре информация в блок-схеме становится более ясной, что в конечном итоге положительно сказывается на дальнейшей работе.

Реализация алгоритма в виде блок-схемы показана на рисунке 14, на котором демонстрируется последовательность шагов и взаимосвязи между ними.

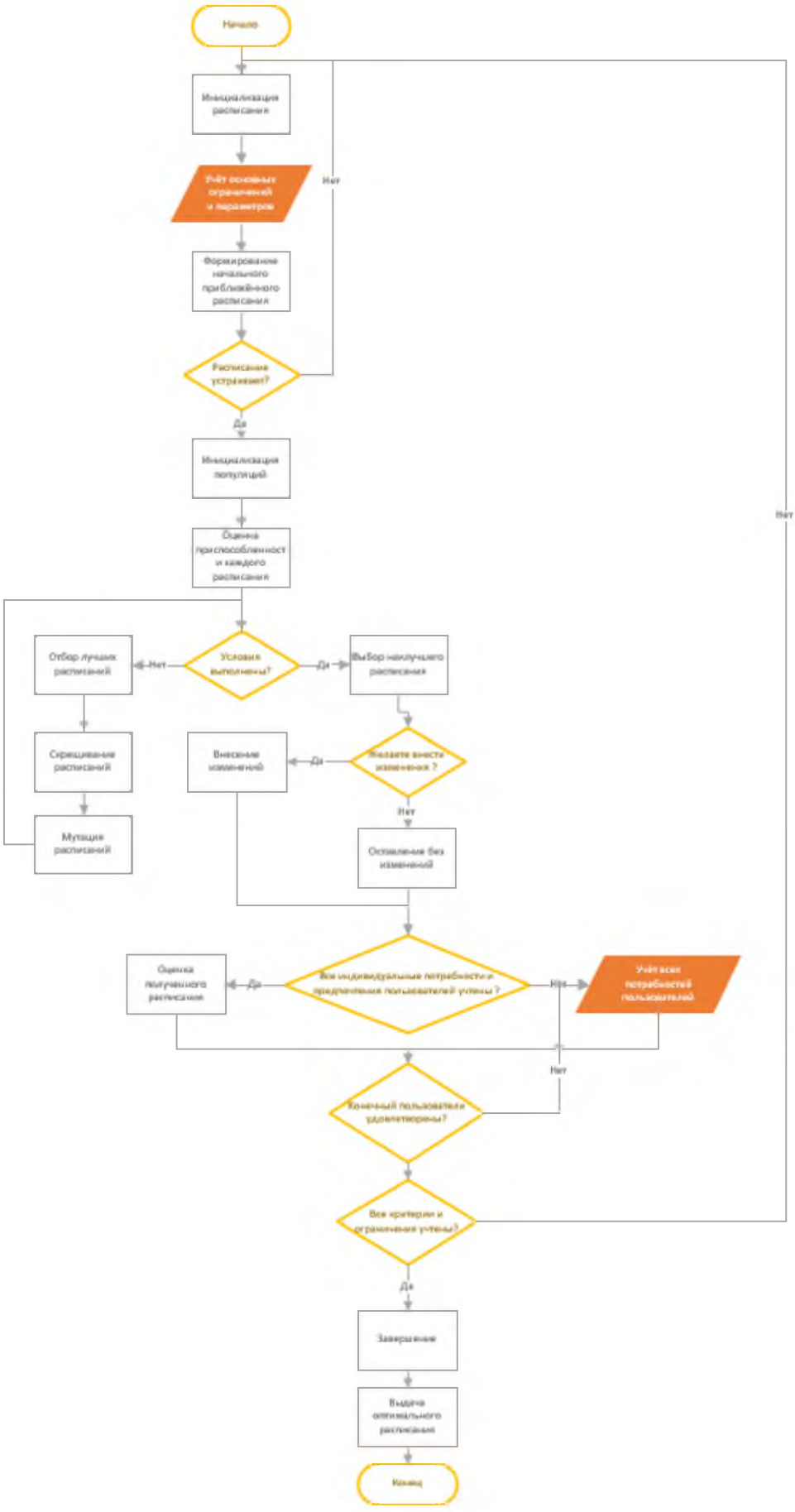



Рисунок 14 – Блок схема работы алгоритма

## 2.3 Сравнение и выбор интегрированной среды разработки

IDE (интегрированная среда разработки) — это среда, которая объединяет набор инструментов для создания программного обеспечения. Обычно такие среды предлагают удобный текстовый редактор с подсветкой синтаксиса, автодополнением и проверкой ошибок. Они также предоставляют инструменты для отладки, управления проектами, сборки и развертывания приложений. IDE интегрируются с различными инструментами разработки, и обеспечивают централизованный интерфейс для облегчения работы разработчиков. Ниже представлены популярные среды разработки для веб-программирования:

PyCharm — одна из самых востребованных IDE для языка Python, разработанная компанией JetBrains. Вот её основные особенности [20]: интуитивно понятный интерфейс, что обеспечивает комфорт в использовании; автодополнение кода, ускоряющее процесс написания и уменьшающее вероятность ошибок; поддержка виртуальных сред; интеграция с фреймворками для веб-разработки; широкие возможности рефакторинга кода; связь с системами контроля версий; поддержка нескольких языков программирования помимо Python. Рабочее окно PyCharm представлено на рисунке 15.

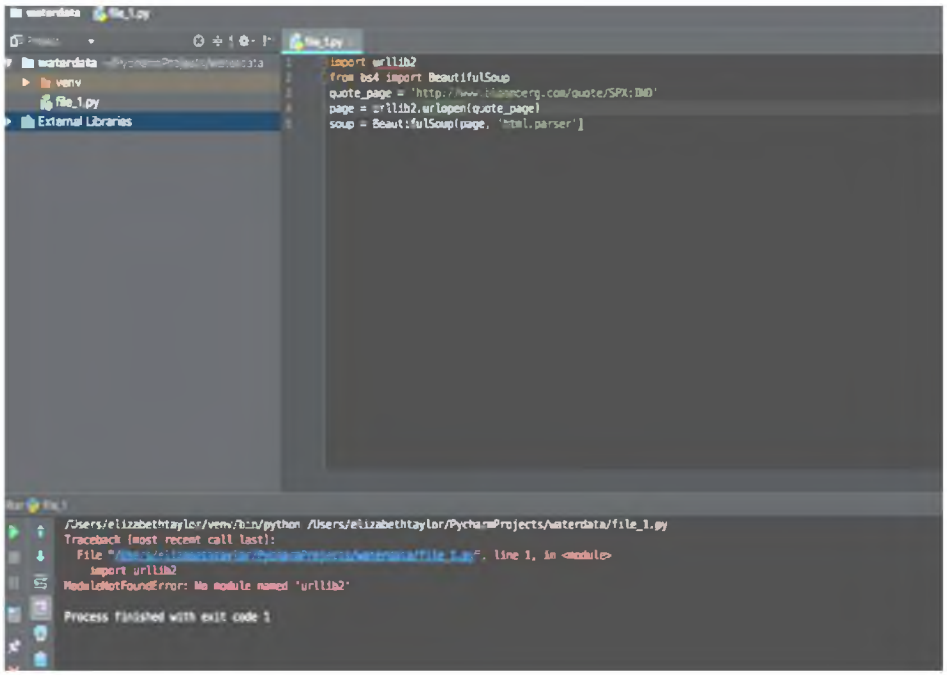



Рисунок 15 – Рабочее окно PyCharm

PyCharm доступен в версиях Community Edition и Professional Edition, где последняя включает дополнительные функции, такие как поддержка баз данных и инструменты для разработки веб-приложений.

Visual Studio Code — бесплатная среда разработки от Microsoft, поддерживающая множество языков и позволяющая создавать различные типы приложений. Рабочее окно Visual Studio Code можно увидеть на рисунке 16.

Основные характеристики этого инструмента: гибкость и расширяемость; встроенная поддержка Git для управления версиями; инструменты для отладки, включая просмотр значений переменных; функция автодополнения кода для упрощения и ускорения написания; терминал для выполнения команд прямо в среде; удобный и понятный интерфейс, подходящий для новичков.

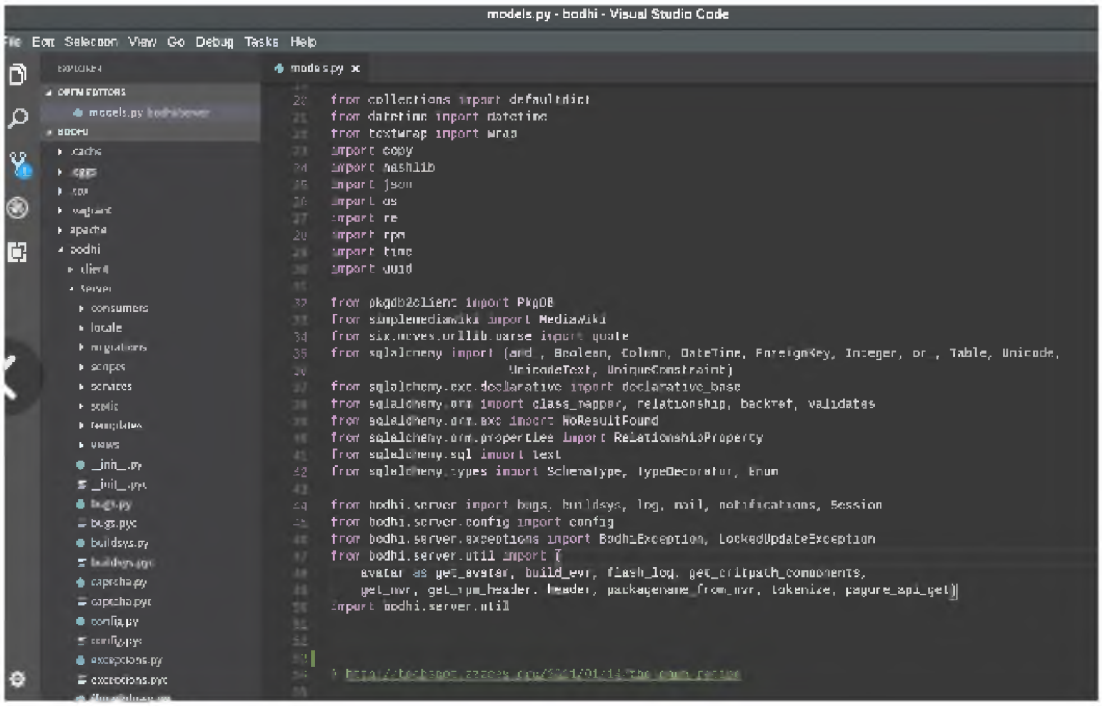



Рисунок 16 – Рабочее окно VS Code

Visual Studio Code за свою функциональность и разнообразие расширений стал одним из самых популярных инструментов разработки программного обеспечения [1].

Atom — это бесплатный текстовый редактор с открытым исходным кодом, созданный GitHub [15]. Рабочее окно Atom представлено на рисунке 17. Ключевые особенности редактора: расширяемость через настройки и плагины; благодаря веб-технологиям (HTML, CSS, JavaScript); интеграция с Git для

удобной работы с репозиториями; поддержка плагинов для добавления функций отладки; функция автодополнения кода для ускорения написания и снижения ошибок; широкий выбор тем оформления и стилей; удобное управление пакетами через встроенный менеджер пакетов apm.

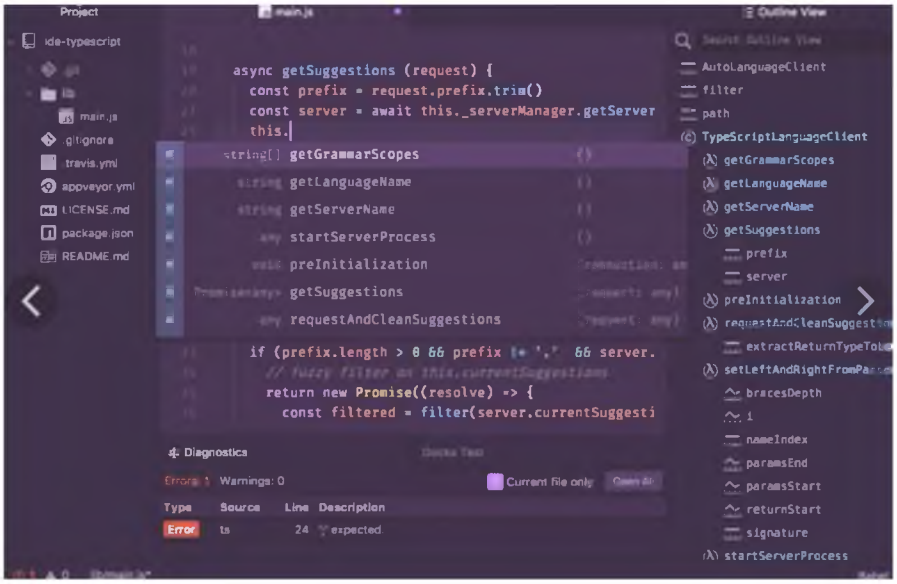



Рисунок 17 – Рабочее окно Atom

Atom является популярным выбором среди разработчиков благодаря своей гибкости, расширяемости и возможности полной настройки под конкретные потребности пользователя.

Таким образом, все три среды разработки имеют свои уникальные особенности и преимущества, что делает их ценными инструментами для разработчиков. Далее, в таблице 5 приведено сравнение всех IDE и выбор подходящего под проект разработки автоматического формирования расписания.

Таблица 5 – Сравнение IDE

| Название IDE → Критерий ↓ | PyCharm | Visual Studio Code | Atom |
|---|---|---|---|
| Функциональность | Очень высокая | Очень высокая | Высокая |
| Производительность | Очень высокая | Высокая | Высокая |
| Интеграция с системами управления версиями | Присутствует | Присутствует | Присутствует |
| Расширяемость и сообщество | Отличная | Отличная | Отличная |
| Плагины | Множество | Множество | Множество |
| Оптимизация под Python | Отличная | Хорошая | Хорошая |

Сравнение показывает, что PyCharm подходит для разработки на Python благодаря своей специализированной функциональности. Эта IDE предлагает функции и инструменты, специально разработанные для Python, что в свою очередь упрощает процесс разработки. PyCharm отличается высокой производительностью и стабильностью, это важный критерий при работе с большими проектами. Поддержка сообщества является преимуществом, предлагая помощь и решения от других пользователей и разработчиков.

## 2.4 Описание работы системы

Система автоматического формирования расписания для образовательных учреждений, разработанная на платформе Django, предназначена для упрощения задач по созданию, управлению и оптимизации учебного расписания.

В ее основе лежит сочетание жадного и генетического алгоритмов, которые позволяют эффективно распределять занятия между предметами, преподавателями, аудиториями и группами. Также предусмотрены интерактивные функции для ручного редактирования расписания.

Для хранения данных используется SQLite, и структура базы данных включает таблицы, отражающие дисциплины, преподавателей, аудитории, группы и само расписание.

Модели данных определяются с помощью Django ORM и охватывают ключевые сущности: Discipline, Teacher, Audience, Group и Schedule. Взаимодействие с данными осуществляется через Django views, где основные функции включают создание расписания, просмотр уже существующих расписаний и редактирование записей.

Для отображения информации пользователю используются Django templates. К основным шаблонам относятся форма для создания расписания и таблица для демонстрации текущего расписания.

Жадный алгоритм применяется для первоначального быстрого создания расписания. Этот алгоритм сортирует занятия по определённым критериям, таким как приоритет или сложность, и последовательно распределяет занятия по доступным аудиториям, начиная с первого элемента списка. При этом производится проверка доступности аудитории, преподавателя и группы в нужное время.

В случае конфликта система пытается найти альтернативное время или аудиторию. Если решение проблемы невозможно, занятие остается без назначения для последующей обработки.

Генетический алгоритм применяется для оптимизации начального расписания, полученного с помощью жадного алгоритма. Он начинает с формирования популяции расписаний на базе оригинального расписания и оценивает их по критериям, таким как количество конфликтов и баланс нагрузки. Наилучшие расписания выбираются для создания нового поколения, между ними происходит обмен частями расписания, а также вносятся случайные изменения для увеличения разнообразия. Процесс продолжается до тех пор, пока не будет достигнуто оптимальное расписание или не завершится определенное количество итераций.

Система позволяет пользователям управлять расписанием вручную через интерфейс Django admin и интерактивные представления. Основные функции включают просмотр расписания, редактирование времени, аудитории и преподавателя для каждого занятия, а также добавление и удаление занятий посредством специальной формы.

Система тщательно проверяет расписание на наличие конфликтов: аудитория не должна быть занята несколькими группами одновременно, преподаватель не может вести несколько занятий в одно время, а у групп не должно быть пересекающихся занятий. При обнаружении конфликтов система предлагает пользователям возможные пути решения или автоматические корректировки.

Таким образом, система автоматического формирования расписания для

образовательных организаций на базе Django представляет собой мощный инструмент для эффективного создания и управления учебным расписанием, гармонично сочетая автоматизацию процесса и интерактивные возможности для пользователей [13].

## 2.5 Итоги второго раздела

Процесс проектирования был осуществлён с применением UML. Разработаны диаграммы вариантов использования с четырьмя представленными случаями.

Для каждого случая были созданы диаграммы последовательностей, что позволяет визуализировать взаимодействие различных элементов системы.

Затем был проведён анализ функционирования алгоритма с использованием блок-схемы.

В работе были описаны генетические алгоритмы, жадный алгоритм и интерактивный метод. В результате была создана блок-схема, демонстрирующая работу алгоритма, что обеспечивает ясное понимание последовательности действий и принципа его функционирования.

Затем был выбран подходящий инструмент разработки, в результате чего для создания системы автоматического формирования расписания выбрана среда разработки PyCharm.

В завершение описана работа системы автоматического формирования расписания для образовательной организации.

# 3 Разработка автоматического формирования расписания

## 3.1 Создание базы данных и проекта для Django

Сначала с помощью «SQLite» создадим и настроим базу данных так как на Django удобнее всего работать именно с «SQLite». Для этого нам потребуется «SQLite Studio». Создадим базу данных с помощью команды «Добавить базу данных» (Рисунок 18).

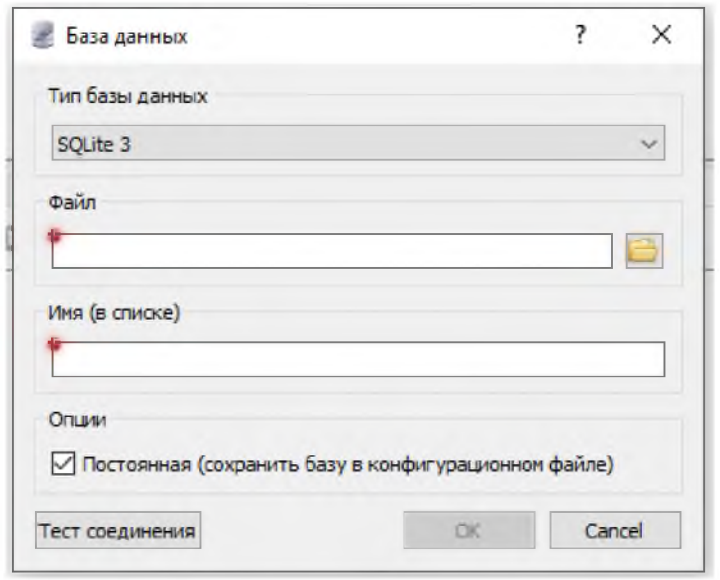



Рисунок 18 – Создание базы данных в SQLite

После это необходимо создать таблицы, в которых будут храниться данные. Делается это с помощью SQL запросов таких как показаны на рисунке 19.

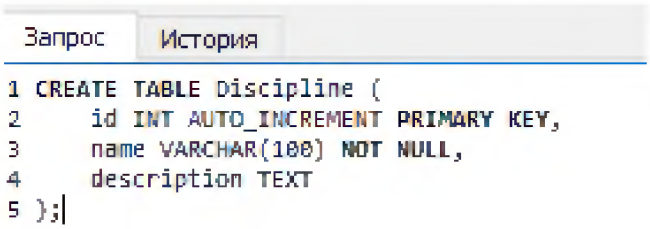



Рисунок 19- Запрос создания таблицы

Таким же образом создаём остальные таблицы в результате чего получается следующие таблицы показанные на рисунке 20.

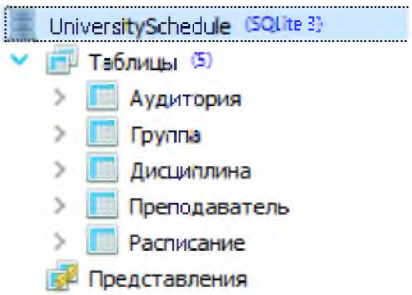



Рисунок 20 – Созданные таблицы

После этого необходимо заполнить таблицы данными. Для этого взяты случайные данные, а полноценная рабочая бд института будет использоваться в дальнейшем после работы с тестовыми данными.

Заполняем таблицы с помощью запросов как показаны на рисунке 21.

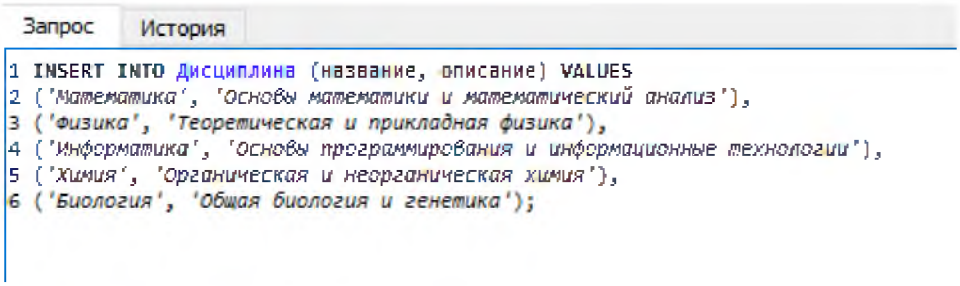



Рисунок 21- Запрос для заполнения таблицы данными

В результате получаем таблицы с заполненными данными (Рисунок 22)

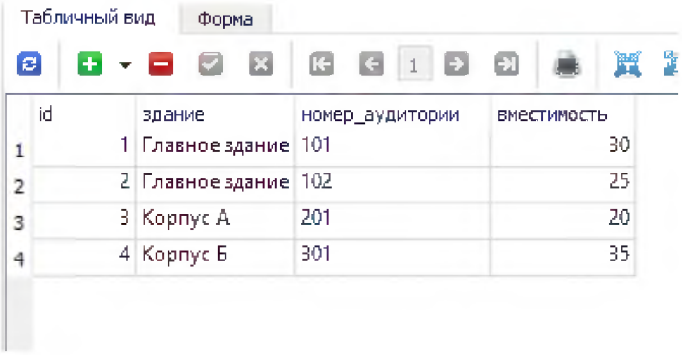



Рисунок 22 – таблица с заполненными данными

После проделываем такую же процедуру для всех таблиц, после чего все таблицы с данными сохраняем для дальнейшего использования в качестве базы данных для Django.

Для начала создадим новый проект Django и настроим его для работы с

базой данных. В этом процессе будут следующие шаги:

– установим Django и создадим новый проект,

– настроим подключение к базе данных SQLite,

– создадим модели для таблиц,

– сгенерируем и применим миграции,

– импортируем начальные данные в базу данных,

– разработаем алгоритмы для автоматического формирования расписания.

Открываем PyCharm и создаём новый проект Django.

Открываем терминал в PyCharm и выполняем следующие команды (Рисунок 23):

```
pip install django

django-admin startproject university_schedule

cd university_schedule

django-admin startapp scheduler
```

Рисунок 23 – Создание проекта на Django

Это первоначальные действия требуемые для создания тела проекта Django которые также создают необходимые для проекта файлы.

Открываем файл settings.py в директории university_schedule и настраиваем базу данных (Рисунок 24):

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Рисунок 24 – Настройка базы данных в Django

После этого необходимо создать модели.

Модели в Django предназначены для определения структуры данных, с которыми будет взаимодействовать приложение.

Они формируют собой классы, которые описывают таблицы в базе данных.

Модели задают поля и поведение данных, которые будут храниться в базе данных, и обеспечивают взаимодействие с ней через объекты на языке Python.

Основные функции моделей Django включают в себя:

– Описание структуры данных: Модели определяют, какие поля (атрибуты) будут у сущностей, их типы, ограничения и связи между ними (такие как отношения один-к-одному, один-ко-многим и многие-ко-многим).

– Создание и редактирование записей: Модели предоставляют возможность добавления новых записей в базу данных и изменения уже существующих.

– Запросы к базе данных: Модели предлагают API для выполнения запросов к базе данных, включая возможности фильтрации, сортировки и агрегации данных.

– Валидация данных: В моделях могут быть реализованы методы для проверки данных перед их сохранением в базе данных.

Готовые модели Django можно увидеть на рисунке 25.

Далее необходимо выполнить миграцию данных.

Миграции в Django используются для управления изменениями в структуре базы данных.

Они позволяют применять изменения в моделях Django к реальной базе данных, чтобы структура базы данных всегда соответствовала определению моделей.

Миграции предоставляют механизм для создания, изменения и удаления таблиц и их полей в базе данных, обеспечивая плавный переход от одной версии схемы базы данных к другой.


Рисунок 25 – модели созданные в Django

Выполняется миграция в терминале с помощью команд представленных на рисунке 26.

```
python manage.py makemigrations
python manage.py migrate
```

Рисунок 26 – Команды для миграции данных

Далее необходимо прописать приложения в файле settings.py. Приложение прописывается в список INSTALLED_APPS в файле settings.py проекта Django для того, чтобы Django знал о его существовании и мог включить его в свою экосистему (рисунок 27).

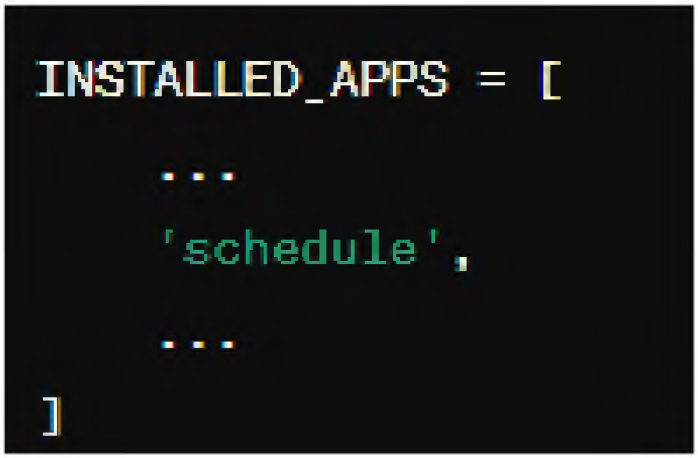



Рисунок 27 – Прописанное приложение

Далее необходимо создать суперпользователя. Суперпользователь в Django — это учетная запись с максимальными привилегиями, предназначенная для управления всем сайтом через административную панель Django (Django Admin).

Суперпользователь имеет доступ ко всем разделам административной панели и может выполнять любые операции, такие как создание, редактирование и удаление записей в базе данных, управление пользователями и настройками приложения. Команды для создания суперпользователя приведены на рисунке 28.

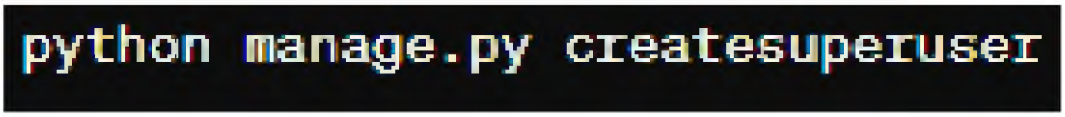



Рисунок 28 – Команда для создания суперпользователя

После введения этой команды терминал попросит ввести имя, электронную почту и пароль пользователя (рисунок 29).

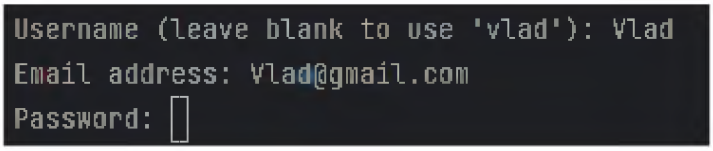



Рисунок 29 – Данные пользователя

Ввод пароля не видно, то есть символы вводятся, но графически не отображаются в терминале. Теперь можно запускать сервер для тестов администрирующих функций (рисунок 30).

```
(.venv) PS C:\Users\Vlad\PycharmProjects\pythonProject1\university_schedule> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 18, 2024 - 20:43:16
Django version 5.0.6, using settings 'university_schedule.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рисунок 30 – Запуск сервера

## 3.2 Разработка жадного алгоритма

Код, представленный в приложении А и обсуждаемый в данном разделе, реализует алгоритм автоматического составления расписания на основе жадного метода. Он извлекает информацию о группах, дисциплинах, преподавателях и аудиториях из базы данных и формирует расписание, стремясь равномерно распределить занятия и избежать конфликтов.

Теперь обратим внимание на ключевые элементы этого кода. Для реализации жадного алгоритма создадим отдельный файл в директории «scheduler», который назовем «greedy_algorithm.py».

Импорт необходимых модулей и моделей в Python (особенно в рамках Django) позволяет включить в текущий файл функции, классы и переменные, определенные в других модулях. Это дает возможность использовать их функциональность без необходимости дублировать код. В контексте Django импорт моделей имеет особое значение, так как компоненты приложения обычно распределены по различным файлам, что облегчает управление и организацию кода. Информация о импорте моделей представлена на рисунке 31.

```
from .models import Расписание, Дисциплина, Преподаватель, Аудитория, Группа
import datetime
```

Рисунок 31 – Импорт моделей

Функция «generate_schedule()» предназначена для автоматического составления расписания занятий на основе данных, хранящихся в базе данных. Она использует жадный алгоритм, чтобы распределить занятия между группами, дисциплинами, преподавателями и аудиториями, стараясь минимизировать конфликты и обеспечить корректность расписания. Рассмотрим шаги и функциональность этой функции более подробно:

1) Очистка текущего расписания: «Расписание.objects.all().delete()».

Перед созданием нового расписания удаляются все существующие записи в таблице Расписание. Это необходимо для того, чтобы начать с чистого листа и избежать дублирования или конфликтов с уже существующими данными.

2) Извлечение данных из базы данных (рисунок 32).

```
группы = Группа.objects.all()
дисциплины = Дисциплина.objects.all()
преподаватели = Преподаватель.objects.all()
аудитории = Аудитория.objects.all()
```

Рисунок 32 - Извлечение данных из базы данных

Функция получает все группы, дисциплины, преподавателей и аудитории из базы данных, чтобы иметь возможность составить расписание для всех возможных комбинаций.

3) Определение дней недели и времени занятий (рисунок 33).

```
дни_недели = ['Понедельник', 'Вторник', 'Среда', 'Четверг',
'Пятница', 'Суббота']
время_занятий = ['08:00-09:30', '09:40-11:10', '11:30-13:00',
'13:10-14:40', '14:50-16:20', '16:30-18:00', '18:10-19:40']
```

Рисунок 33 - Определение дней недели и времени занятий

4) Создание расписания

Этот раздел кода отвечает за фактическое создание расписания, перебирая все возможные комбинации групп, дисциплин, преподавателей и

временных слотов.

Рассмотрим его по частям:

– Создание пустого списка для расписаний «расписания = []». Этот список будет использоваться для хранения объектов расписания перед их сохранением в базу данных,

– перебор всех групп: «for группа in группы:». Цикл проходит по всем группам, полученным из базы данных,

– перебор всех преподавателей: «for преподаватель in преподаватели:»

Для каждой комбинации группы и дисциплины перебираются все преподаватели.

– проверка, преподает ли преподаватель данную дисциплину «if преподаватель_преподает_дисциплину(преподаватель, дисциплина):». Здесь вызывается функция «преподаватель_преподает_дисциплину», которая проверяет, преподает ли данный преподаватель указанную дисциплину. Если да, выполнение продолжается,

– перебор всех дней недели «for день in дни_недели:» Для каждой комбинации группы, дисциплины и преподавателя перебираются все дни недели,

– перебор всех временных слотов «for время in время_занятий:» Для каждого дня недели перебираются все временные слоты,

– проверка возможности добавления занятия в расписание «if можно_добавить_в_расписание(группа, дисциплина, преподаватель, день, время):» Вызывается функция «можно_добавить_в_расписание», которая проверяет наличие конфликтов (например, если у группы или преподавателя уже есть занятие в это время). Если конфликтов нет, выполнение продолжается,

– поиск свободной аудитории: вызывается функция «найти_свободную_аудиторию», которая ищет первую свободную аудиторию для указанного дня и времени. Если аудитория найдена, выполнение продолжается. «аудитория = найти_свободную_аудиторию(день, время), создание объекта расписания и добавление его в список (рисунок 34).

```
    «расписание = Расписание (дисциплина=дисциплина,
преподаватель=преподаватель, аудитория=аудитория,
группа=группа, день_недели=день, время=время
    )
    расписания.append(расписание)
    расписание.save()»
```

Рисунок 34 - Создание объекта расписания и добавление его в список

Создается объект Расписание с заполненными полями, и он добавляется в список расписания. Затем этот объект сохраняется в базе данных с помощью метода save(). Жадный алгоритм готов можно приступать к созданию следующего элемента

## 3.3 Разработка генетического алгоритма

Здесь мы задаем параметры для генетического алгоритма (Рисунок 35):

– число_поколений: количество итераций, которые выполнит алгоритм,

– размер_популяции: количество расписаний в популяции,

– вероятность_мутации: вероятность, с которой каждое расписание будет подвергнуто мутации.

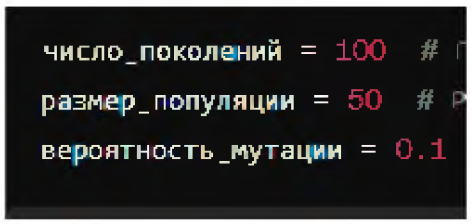



Рисунок 35 – Основные параметры

Функция «genetic_algorithm()» реализует основной цикл генетического алгоритма (Рисунок 36):

– инициализация популяции: создает начальный набор расписаний,

– основной цикл: выполняет селекцию, кроссовер и мутацию для каждого поколения,

– выбор лучшего решения: после завершения всех поколений выбирает лучшее найденное расписание.

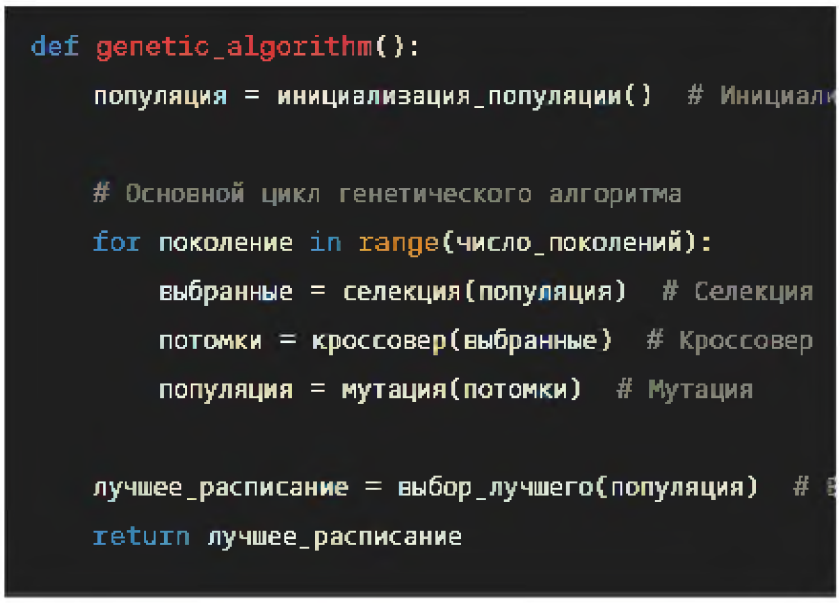



Рисунок 36 - Функция «genetic_algorithm()»

Эта функция создает начальную популяцию расписаний:

1. получение данных: извлекает все группы, дисциплины, преподавателей и аудитории из базы данных; 2. создание расписаний: для каждой группы и дисциплины случайным образом выбирает преподавателя, день, время и аудиторию; 3. формирование популяции: добавляет созданные расписания в популяцию. Код этой функции представлен на рисунке 37.


Рисунок 37 - Функция «инициализация_популяции()»

Функция «селекция(популяция)» (рисунок 38) отбирает наиболее приспособленные расписания:

– сортировка: сортирует популяцию по значению фитнес-функции (от лучшего к худшему),

– отбор: выбирает верхнюю половину расписаний для дальнейшей работы.

```
def селекция(популяция):
    популяция.sort(key=lambda расписание: fitness(расписание), reverse=True)
    return популяция[:размер_популяции // 2]
```

Рисунок 38 - Функция «селекция(популяция)»

Функция «кроссовер(выбранные)» (рисунок 39) выполняет скрещивание выбранных расписаний:

– выбор родителей: Случайным образом выбирает два расписания-родителя из отобранных,

– скрещивание: разделяет родителей в случайном месте и комбинирует части от обоих для создания нового расписания,

– формирование потомков: добавляет созданных потомков в новую популяцию.

```
def кроссовер(выбранные):
    потомки = []
    while len(потомки) < размер_популяции:
        parent1 = random.choice(выбранные)
        parent2 = random.choice(выбранные)
        split = random.randint(1, len(parent1) - 1)
        child = parent1[:split] + parent2[split:]
        потомки.append(child)
    return потомки
```

Рисунок 39 - Функция «кроссовер(выбранные)»

Функция «мутация(потомки)» применяется для внесения случайных изменений в расписания потомков с целью поддержания генетического разнообразия в популяции. Это помогает избежать преждевременной

сходимости к локальному оптимуму и улучшает исследование пространства решений.

Функция «мутация(потомки)» (Рисунок 40) применяет случайные изменения к потомкам:

– Случайная мутация: С вероятностью, заданной параметром «вероятность_мутации», заменяет один элемент расписания на случайный.

```
def мутация(потомки):
    группы = Группа.objects.all()
    дисциплины = Дисциплина.objects.all()
    преподаватели = Преподаватель.objects.all()
    аудитории = Аудитория.objects.all()
    дни_недели = ['Понедельник', 'Вторник', 'Среда', 'Четверг', 'Пятница', 'Суббота']
    время_занятий = ['08:00-09:30', '09:40-11:10', '11:30-13:00', '13:10-14:40', '14

    for расписание in потомки:
        if random.random() < вероятность_мутации:
            группа = random.choice(группы)
            дисциплина = random.choice(дисциплины)
            преподаватель = random.choice(преподаватели)
            день = random.choice(дни_недели)
            время = random.choice(время_занятий)
            аудитория = random.choice(аудитории)
            индекс = random.randint(0, len(расписание) - 1)
            расписание[индекс] = (группа, дисциплина, преподаватель, день, время, аудит

    return потомки
```

Рисунок 40 – Функция «мутация(потомки)»

Функция «выбор_лучшего(популяция)» (Рисунок 41) выбирает лучшее расписание из текущей популяции:

– оценка фитнеса: выбирает расписание с наивысшим значением фитнес-функции.

```
def выбор_лучшего(популяция):
    return max(популяция, key=lambda расписание: fitness(расписание))
```

Рисунок 41 - Функция «выбор_лучшего(популяция)»

Функция «fitness(расписание)» (Рисунок 42) оценивает качество расписания, рассчитывая его «фитнес» — меру того, насколько расписание

соответствует желаемым критериям. В данном случае функция подсчитывает количество конфликтов в расписании и уменьшает оценку за каждый конфликт. Чем меньше конфликтов, тем выше оценка фитнеса, и, соответственно, лучше расписание.

```
def fitness(расписание):
    score = 0
    занятие_по_времени = {}
    for группа, дисциплина, преподаватель, день, время, аудитория in расписание:
        ключ = (группа, день, время)
        if ключ in занятие_по_времени:
            score -= 1
        else:
            занятие_по_времени[ключ] = True
        ключ = (преподаватель, день, время)
        if ключ in занятие_по_времени:
            score -= 1
        else:
            занятие_по_времени[ключ] = True
        ключ = (аудитория, день, время)
        if ключ in занятие_по_времени:
            score -= 1
        else:
            занятие_по_времени[ключ] = True
    return score
```

Рисунок 42 - Функция «fitness(расписание)»

Функция fitness(расписание) оценивает качество расписания:

– подсчет конфликтов: проверяет наличие конфликтов (занятий, назначенных на одно и то же время для одной группы, преподавателя или аудитории) и снижает оценку за каждый конфликт,

– возвращает оценку: чем меньше конфликтов, тем выше оценка расписания.

Этот блок кода (рисунок 43) запускает генетический алгоритм и выводит на экран лучшее найденное расписание, если скрипт выполняется как основная программа.

```
if __name__ == '__main__':
    schedule = genetic_algorithm()
    for entry in schedule:
        print(entry)
```

Рисунок 43 - Основной блок выполнения

Код представленный в приложении Б выполняет полную реализацию генетического алгоритма для автоматического составления расписания. Он включает функции для инициализации популяции, селекции, кроссовера, мутации и оценки фитнеса. Эти функции работают вместе для создания, улучшения и выбора лучших расписаний в рамках заданного числа поколений. В результате получается расписание с минимальным количеством конфликтов, что делает его пригодным для использования в реальных условиях.

Генетический алгоритм помогает автоматизировать и оптимизировать процесс составления расписания, экономя время и усилия, а также повышая качество расписания.

## 3.4 Разработка интерфейса

Для создания интерфейса был использован онлайн-конструктор «Ninjamock» благодаря его простому синтаксису и удобному графическому инструментарию. В будущем планируется перенести интерфейс с конструктора на фреймворк Django путем экспорта в HTML.

Макет расписания групп с некоторыми функциями показан на рисунке 44. Слева на странице расположено меню, где представлены различные функции, включая «Выбор группы». Группа может быть выбрана из выпадающего списка или с помощью функции поиска. После выбора группы для отображения её расписания необходимо нажать кнопку «Отобразить». Справа от выборки групп располагаются функции: «Редактировать», «Сохранить», «Экспорт», «Импорт», «Печать» и «Удалить». Также предусмотрен переход к страницам с расписанием преподавателей (рисунок 45) и аудиторий (рисунок 46).

Функция «Редактировать» позволяет изменять содержимое расписания для выбранных групп, а также редактировать список групп. Подобный функционал доступен и на страницах с расписаниями преподавателей и аудиторий. Функция «Сохранить» отвечает за сохранение внесённых изменений. Функция «Экспорт» позволяет экспортировать расписание, предоставляя возможность выбрать нужный формат. Функция «Импорт» служит для загрузки расписания из заранее определённых форматов файлов. Функция «Печать» отправляет расписание на печать. Наконец, функция «Удалить» очищает расписание для выбранной группы.

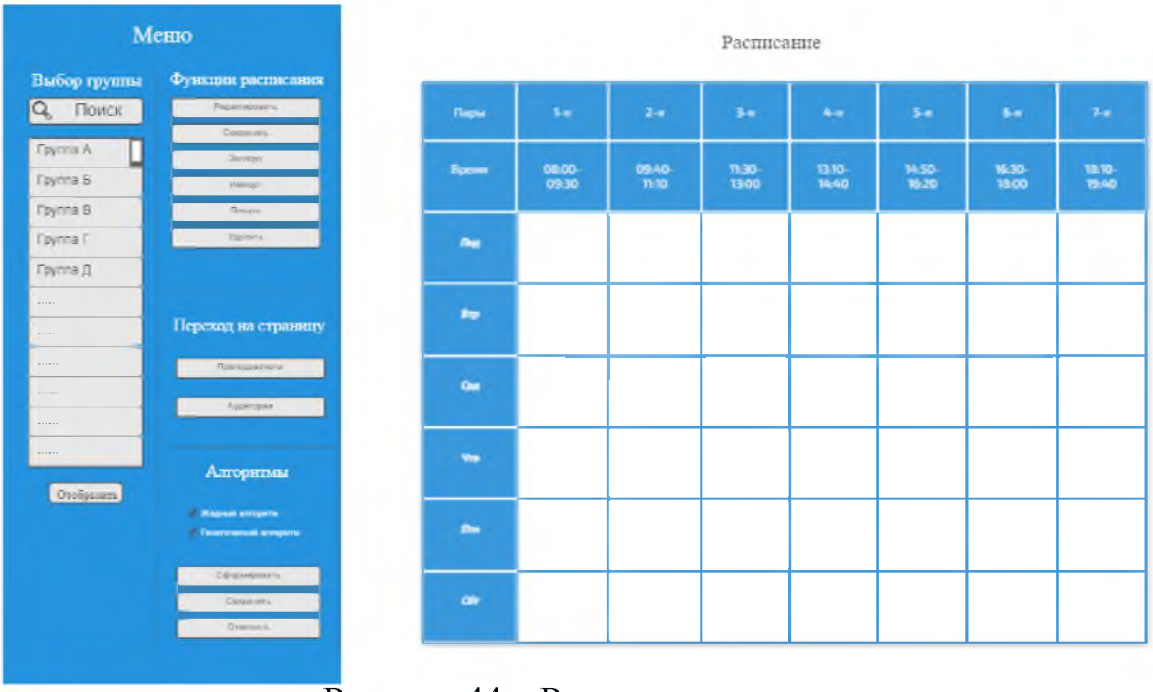



Рисунок 44 – Расписание групп

Также на странице с каждым расписанием представлены алгоритмы для возможности автоматического формирования расписания с помощью «Жадного» или «Генетического» алгоритмов, которые можно применить, поставив «галочку» напротив нужного алгоритма, либо напротив обоих алгоритмов. Для формирования расписания в автоматическом режиме после выбора советующих алгоритмов необходимо нажать кнопку «Сформировать», если результат устраивает, то можно нажать кнопку «Сохранить», а если же нет можно повторно сформировать расписание или же отменить изменения нажав кнопку «Отменить»

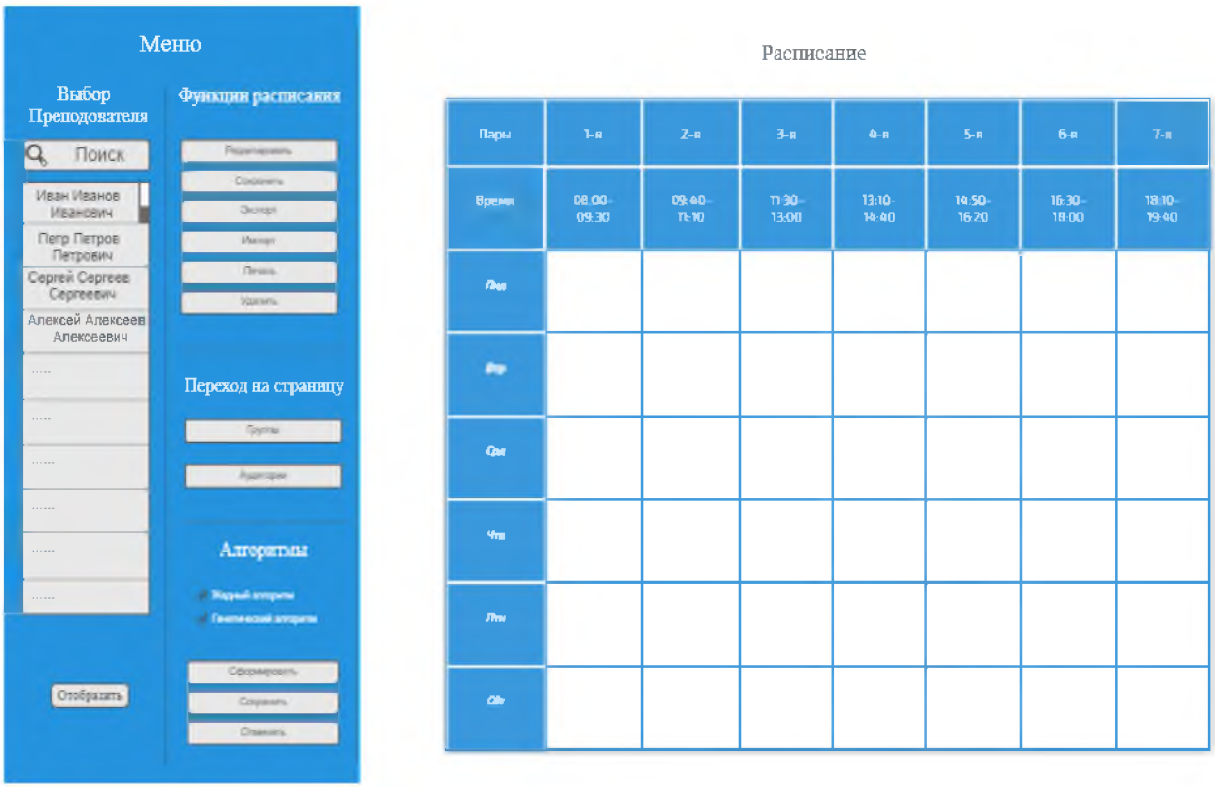



Рисунок 45 – Расписание преподавателей

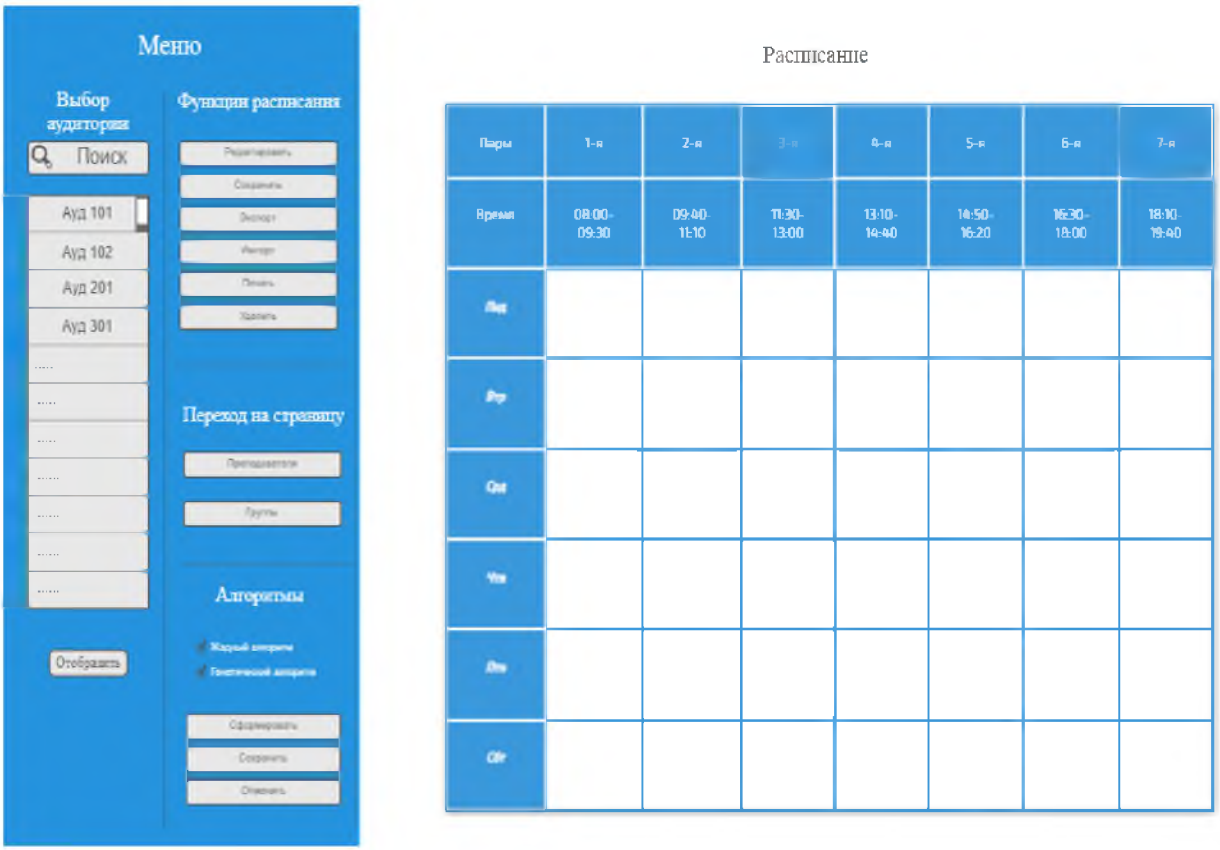



Рисунок 46 – Расписание аудиторий

На данный момент в работе будет использоваться тестовый интерфейс, который можно будет увидеть на рисунках далее.

## 3.5 Тестирование, оптимизация алгоритмов и исследование производительности

На данном этапе необходимо провести тестирование разработанного решения системы автоматического формования расписания.

При запуске системы мы попадаем на главную страницу (рисунок 47), где на выбор предлагается две кнопки «Меню администратора» и «Расписание».

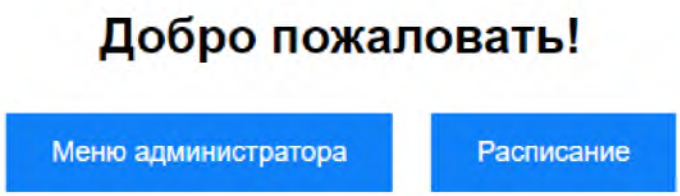



Рисунок 47 – Главная страница

При нажатии на кнопку «Меню администратора» мы попадаем в меню администрирования системы, показанное на рисунке 48.

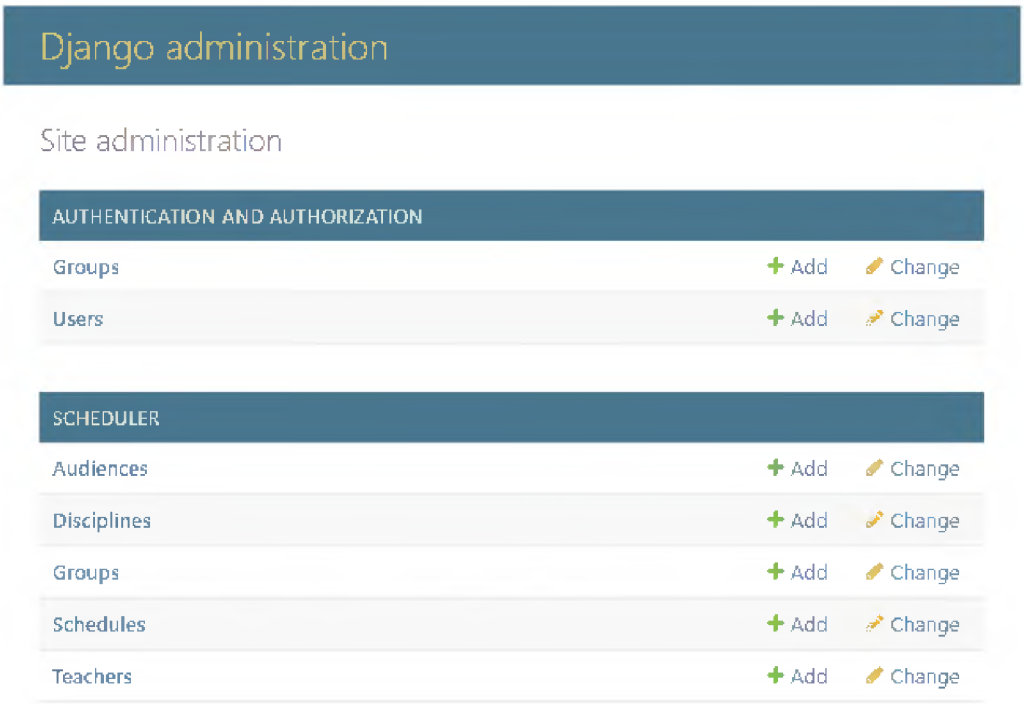



Рисунок 48 – Меню администрирования

Через это меню есть возможность настраивать данные, связанные с расписанием и управлять содержимым базы данных.

При нажатии на главной странице кнопки «Расписание» мы попадаем на страницу с расписанием (рисунок 49).

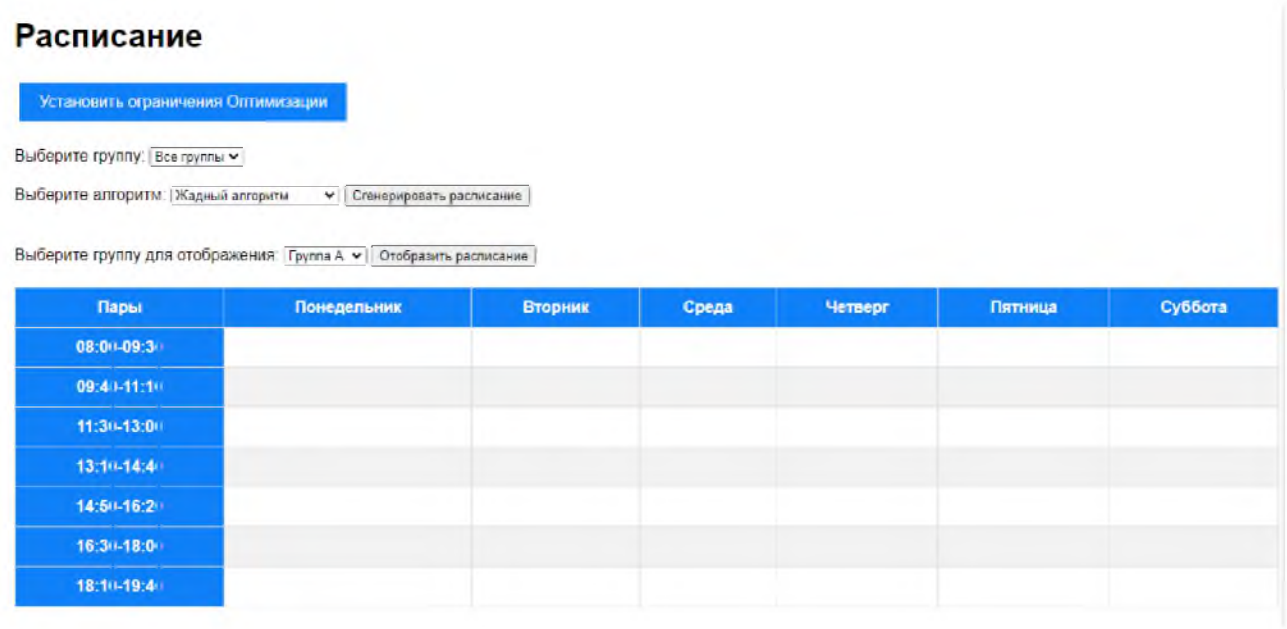



Рисунок 49 – Страница «Расписание»

На данной странице мы наблюдаем общий интерфейс системы, на котором генерируется и отображается расписание. На рисунке 49 мы также видим кнопку «Установить ограничения Оптимизации», при нажатии на которую появляется всплывающее окно, показанное на рисунке 50.

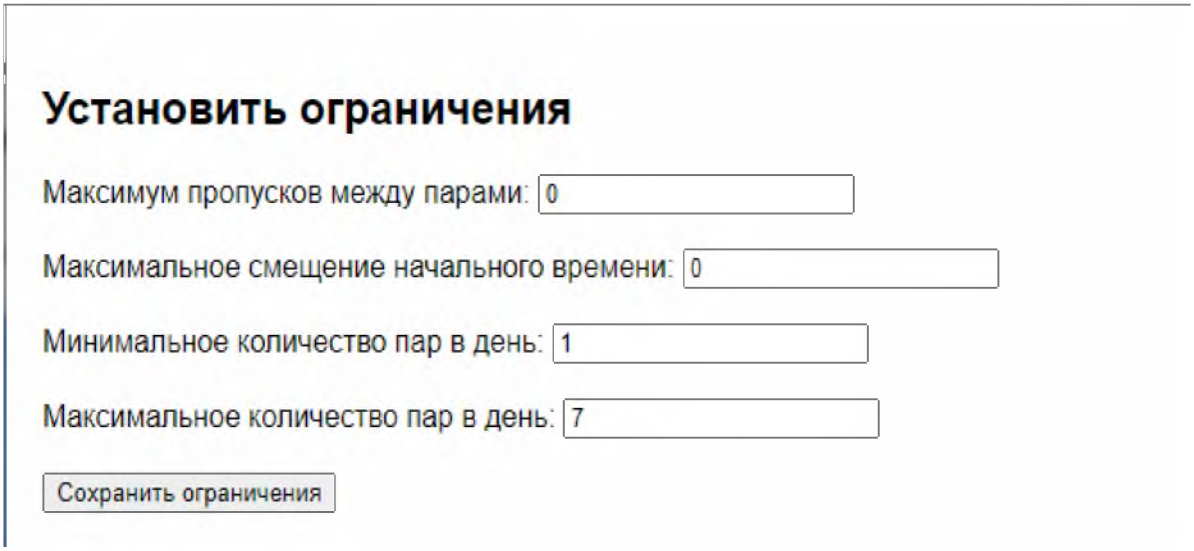



Рисунок 50 – Установка ограничений

На рисунке 50 также можно наблюдать такие ограничения как

«Максимум пропусков между парами», «Максимальное смещение начального времени», «Минимальное количество пар в день», «Максимальное количество пар в день».

Далее на рисунке 51 мы наблюдаем выпадающий список, где можно выбрать либо все группы, либо отдельную.

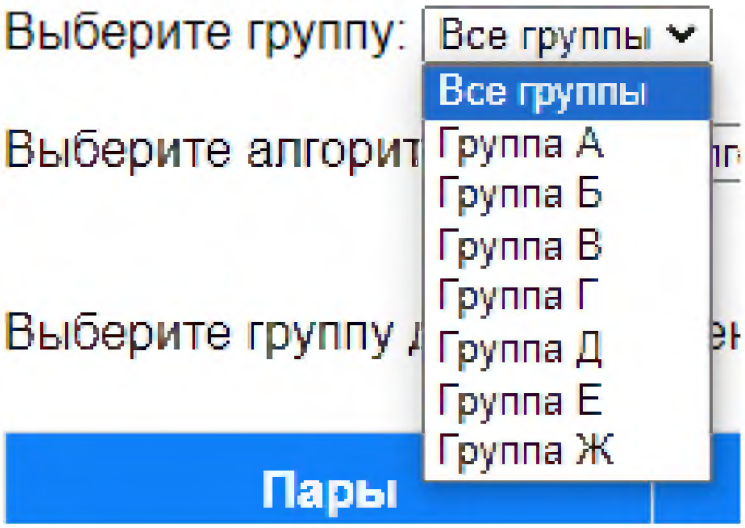



Рисунок 51 – Список групп

Следующий выпадающий список и кнопку «Сгенерировать расписание» представлены на рисунке 52. Данный выпадающий список отвечает за выбор алгоритма.

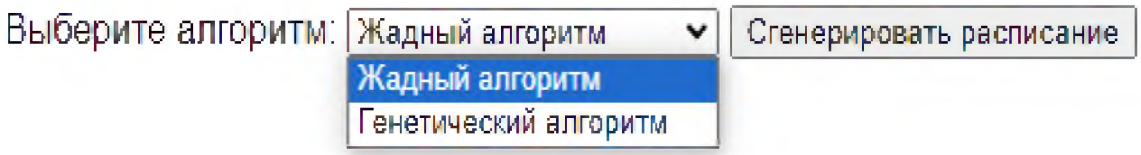



Рисунок 52 – Выпадающий список алгоритмов и кнопка генерации расписания

После выбора группы и алгоритма нажимаем на кнопку «Сгенерировать расписание» после чего появляется сгенерированное расписание с помощью выбранного алгоритма (рисунок 53) по умолчанию, показывающего расписание, созданное для группы А.

Для просмотра расписания другой группы необходимо выбрать группу для отображения с помощью выпадающего списка и нажать кнопку «Отобразить расписание» (рисунок 54).

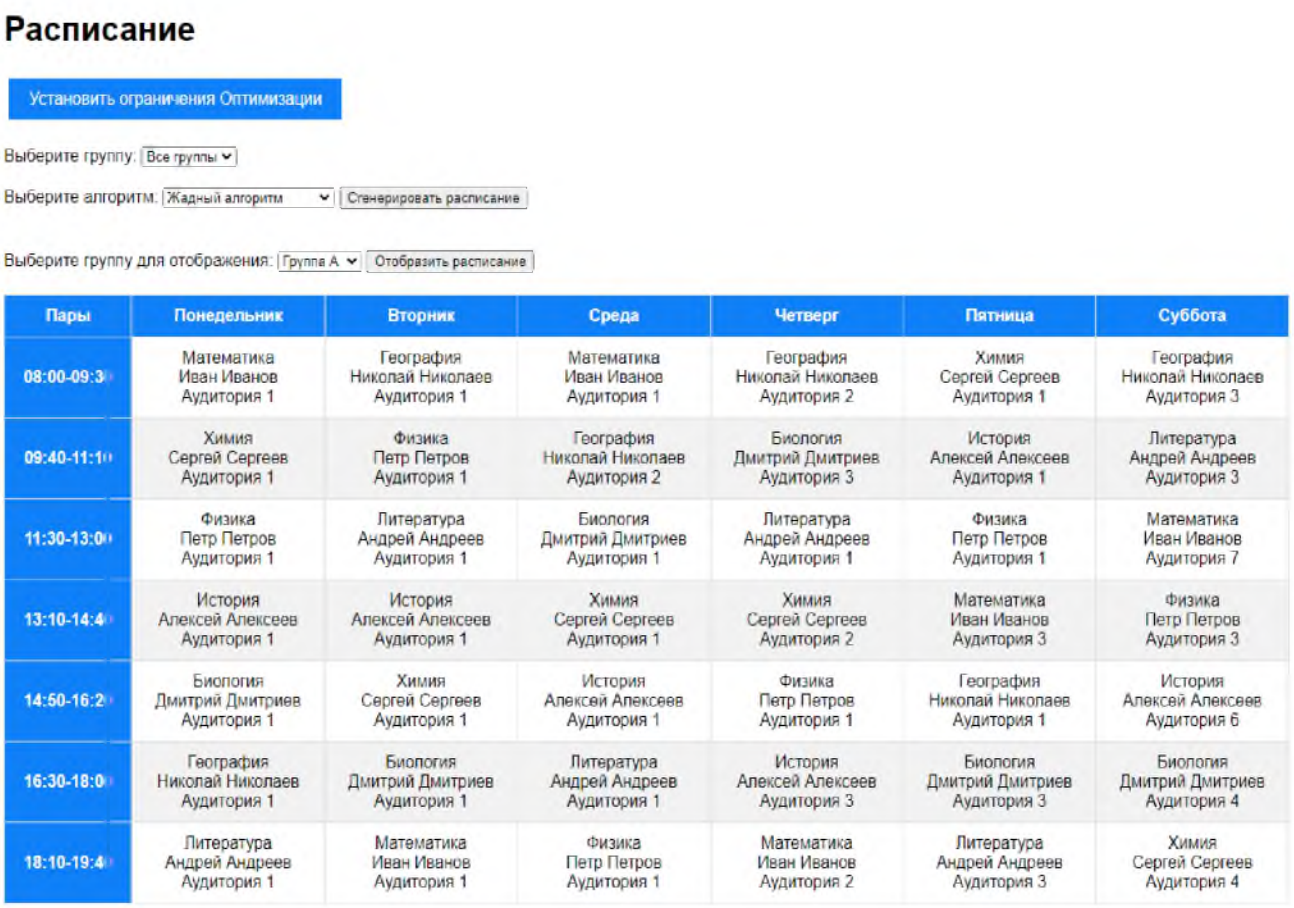



Рисунок 53 – Расписание созданное «Жадным алгоритмом»

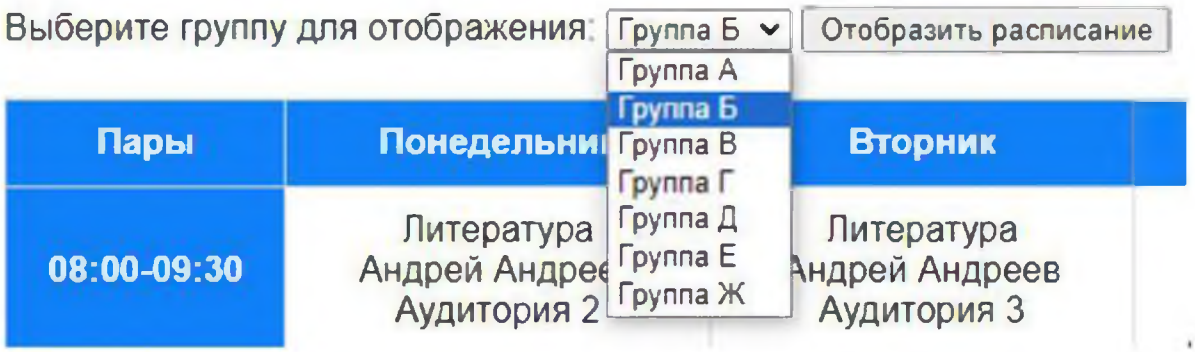



Рисунок 54 – Выбор группы для отображения

За создание начального расписание отвечает «Жадный алгоритм», а за его оптимизацию отвечает «Генетический алгоритм». После построения расписания для его оптимизации выставляем ограничения как показано на рисунке 55. После этого также выбираем алгоритм как показано на рисунке 52 и нажимаем кнопку «Сгенерировать расписание». Генетический алгоритма начнёт оптимизацию расписания с учётом ограничений. После начала оптимизации запустится таймер, отсчитывающий оставшееся время

оптимизации (рисунок 56). Итог оптимизации генетического алгоритма для одной из групп показан на рисунке 57.


Рисунок 55 – Выставленные ограничения


Рисунок 56 – Таймер с оставшимся временем

| Пары | Понедельник | Вторник | Среда | Четверг | Пятница | Суббота |
|---|---|---|---|---|---|---|
| 08:0(-09:3( | | | Физика<br>Петр Петров<br>Аудитория 1 | | | |
| 09:4(-11:1( | | Биология<br>Дмитрий Дмитриев<br>Аудитория 7 | Физика<br>Петр Петров<br>Аудитория 6 | Химия<br>Сергей Сергеев<br>Аудитория 1 | | Химия<br>Сергей Сергеев<br>Аудитория 7 |
| 11:3(-13:0( | Биология<br>Дмитрий Дмитриев<br>Аудитория 6 | Химия<br>Сергей Сергеев<br>Аудитория 2 | География<br>Николай Николаев<br>Аудитория 1 | Литература<br>Андрей Андреев<br>Аудитория 6 | Биология<br>Дмитрий Дмитриев<br>Аудитория 1 | |
| 13:1(-14:4( | | Физика<br>Петр Петров<br>Аудитория 1 | Биология<br>Дмитрий Дмитриев<br>Аудитория 7 | | Физика<br>Петр Петров<br>Аудитория 1 | География<br>Николай Николаев<br>Аудитория 1 |
| 14:5(-16:2( | История<br>Алексей Алексеев<br>Аудитория 1 | | | Биология<br>Дмитрий Дмитриев<br>Аудитория 2 | | География<br>Николай Николаев<br>Аудитория 1 |
| 16:3(-18:0( | Химия<br>Сергей Сергеев<br>Аудитория 6 | Физика<br>Петр Петров<br>Аудитория 1 | География<br>Николай Николаев<br>Аудитория 1 | География<br>Николай Николаев<br>Аудитория 1 | География<br>Николай Николаев<br>Аудитория 1 | Физика<br>Петр Петров<br>Аудитория 5 |
| 18:1(-19:4( | | | География<br>Николай Николаев<br>Аудитория 2 | | | Физика<br>Петр Петров<br>Аудитория 1 |

Рисунок 57 – Итог оптимизации Генетическим алгоритмом

Итоговое расписание соответствует заданным ограничениям из чего можно сделать вывод что алгоритмы работают корректно. В системе также предусмотрено ручное редактирования расписания. При нажатии на ячейку с появляется всплывающее окно как показано на рисунке 58.

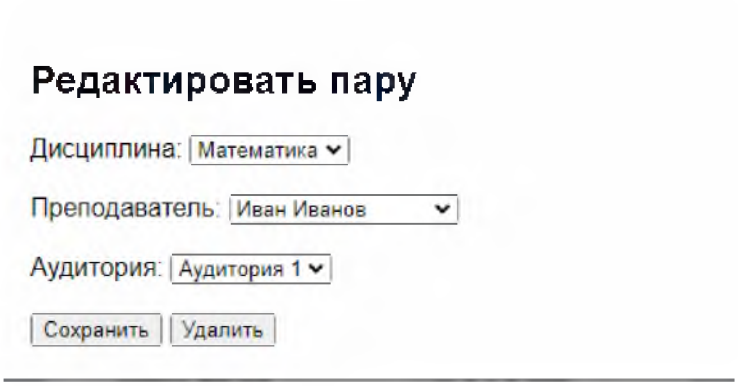



Рисунок 58 – Редактирование занятий

Стоит отметить, что дисциплины привязаны к преподавателям и при выборе дисциплины преподаватель к этой дисциплине выбирается автоматически. При формировании расписания используется жадный алгоритм, а при оптимизации используется генетический алгоритм. Формирования расписания не занимает много времени, так как ничего сложного там не просчитывается, а другое же дело оптимизация, которая при введении ограничений выполняется в среднем до 20 минут как будет в дальнейшем показано. Улучшить оптимизацию кода можно разными способами, например улучшение производительности отдельных функций, использование более эффективных данных и алгоритмов и оптимизация кода для многопоточности или использования GPU или CPU. Предлагается внести исправления в коде как описано ниже. Одно из улучшений это распараллеливание вычислений, чтобы ускорить выполнение, как показано на рисунке 59.

```
with Pool() as pool:
    fitness_scores = pool.starmap(fitness, [(individual, groups, days_of_week, time_slots, min_classes_per_day, max_classes_per_day, max_gaps) for individual in population])
```

Рисунок 59 - Распараллеливание функции оценки

В фрагменте кода показанного на рисунке 59 используется способ называемый «multiprocessing.Pool» для параллельного вычисления приспособленности каждой особи в популяции, «pool.starmap» позволяет

распараллелить выполнение функции «fitness», которая оценивает каждую особь это позволяет улучшить эффективность и тем самым сократить время на оценку. В коде показанном на рисунке 60 «фитнес» функция вычислялась последовательно для каждой особи.

```
fitness_scores = [fitness(individual, groups, days_of_week, time_slots, min_classes_per_day, max_classes_per_day, max_gaps) for individual in population]
```

Рисунок 60 – код до распараллеливания

Улучшение функции мутации для ускорения процесса вычислений, функция до внесения изменений показана на рисунке 61 после изменений показана на рисунке 62.

```
def mutate(schedule):
    if random.random() < mutation_rate:
        index = random.randint(0, len(schedule) - 1)
        schedule[index] = random.choice(new_value)
```

Рисунок 61 – функция мутации до изменений

```
def mutate(schedule, groups, days_of_week, time_slots, max_gaps, max_classes_per_day):
    if random.random() < 0.1:  # mutation_rate
        group = random.choice(groups)
        day = random.choice(days_of_week)
        time_slot = random.choice(time_slots)
        discipline = random.choice(group.disciplines.all())
        teacher = Teacher.objects.filter(discipline=discipline).first()
        audience = find_available_audience(day, time_slot)
        if audience and can_schedule(group, discipline, teacher, day, time_slot, max_gaps,
            Schedule.objects.create(
                discipline=discipline,
                teacher=teacher,
                audience=audience,
                group=group,
                day_of_week=day,
                time_slot=time_slot
            )
```

Рисунок 62 - Изменённая функция мутации

На рисунке 62 код работает следующим образом: выбирается случайная группа, день, временной интервал и дисциплина, проверяется возможность назначения занятия с учетом ограничений (максимальных пропусков и

максимального количества занятий в день и т.д.) с помощью функции «can_schedule». Если все условия соблюдены, создается новая запись в расписании.

В новом коде используется вероятность мутации (mutation rate), которая была установлена на 10% и теперь она закреплена, что ограничивает количество мутаций в каждом поколении, снижая вероятность внесения случайных изменений, которые могут ухудшить качество решений, но при этом 10% - это достаточно высокая вероятность, чтобы обеспечить достаточное количество мутаций для исследования пространства решений.

Уменьшение размера выборки для кроссовера и мутации также одно из нововведений. Код до изменения показан на рисунке 63, а после изменений на рисунке 64.

```
while len(new_population) < population_size:
    parent1, parent2 = random.sample(population, 2)
    child1, child2 = crossover(parent1, parent2)
    new_population.append(child1)
    new_population.append(child2)
```

Рисунок 63– Код выборки популяции до изменений

```
new_population = population[:10]
while len(new_population) < population_size:
    parent1, parent2 = random.sample(population[:20], 2)
    child1, child2 = crossover(list(parent1), list(parent2))
    new_population.append(child1)
    new_population.append(child2)
```

Рисунок 64 – Изменённый код выборки популяции

До изменения кода родители для скрещивания выбирались случайным образом из всей популяции, что могло привести к плохим потомкам.

После изменения родители выбираются из верхних 20 лучших особей, что увеличивает шансы на получение более приспособленных потомков.

Далее приведены варианты системы без внедрения оптимизации в код и после внедрения которые описаны в таблице 6 предложенных изменений.

Таблица 6 – Варианты системы до оптимизации кода и после.

| Название изменения | До оптимизации | После оптимизации |
|---|---|---|
| Улучшение функции мутации | Мутация выполнялась случайно без учета текущего состояния расписания и ограничений. | Мутация выполняется с учетом доступных ресурсов и ограничений, что предотвращает создание конфликтов. |
| Параллелизация | Оценка в функции «фитнес» вычислялась последовательно | Оценка в функции «фитнес» теперь вычисляется параллельно |
| Уменьшение размера выборки | Родители для кроссинговера выбирались случайным образом из всей популяции, что могло привести к плохим потомкам. | Родители выбираются из верхних 20 лучших особей, что увеличивает шансы на получение более приспособленных потомков. |

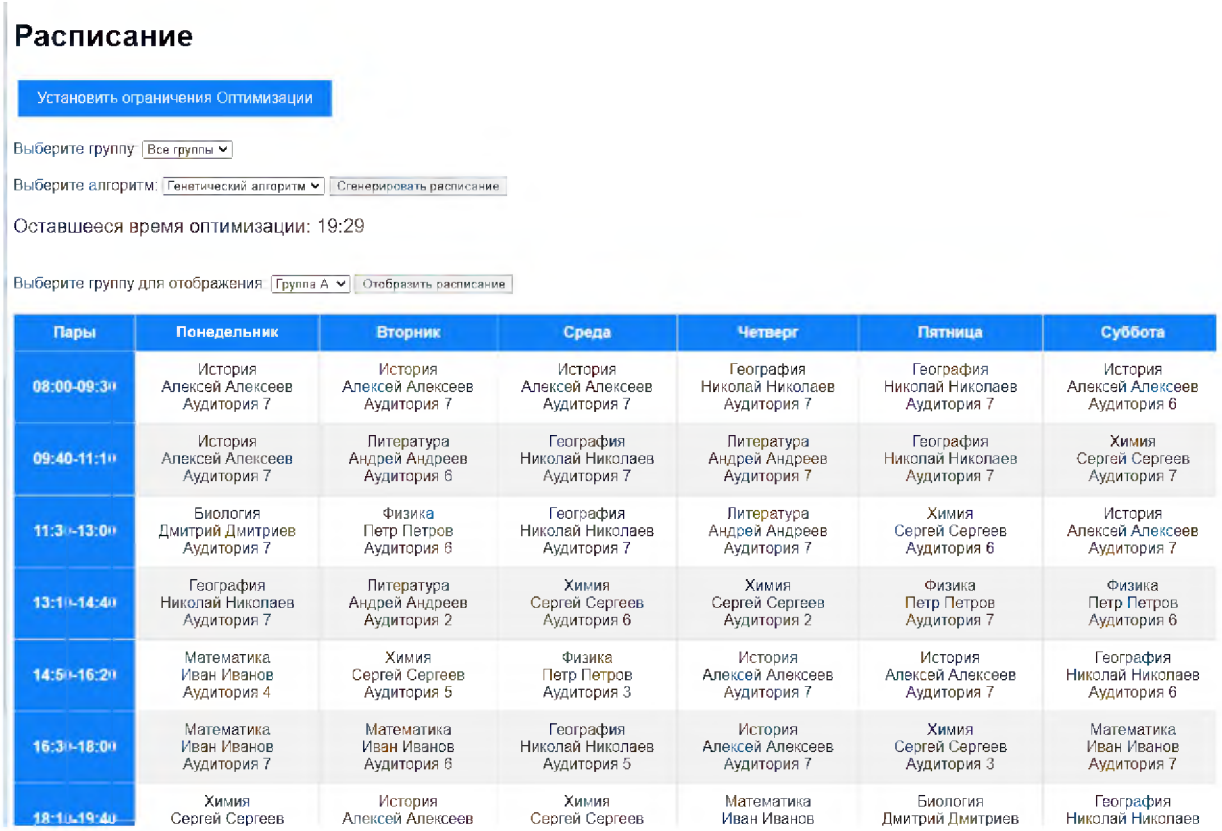
Далее в код генетического алгоритма были введены предложенные исправления. После этого необходимо протестировать систему и выявить произошло ли улучшение производительности за счёт внедрения нововведений.

Система тестировалась до оптимизации кода и после, время за которое генетический алгоритм оптимизировал составленное расписание полученные за 5 попыток можно увидеть в таблице 7.

Таблица 7 – Время попыток

| Номер попытки | Время попытки до внесения улучшений | Время попытки после внесения улучшений |
|---|---|---|
| 1 | 20 минут 43 секунды | 11 минут 14 секунд |
| 2 | 19 минут 20 секунда | 9 минут 56 секунд |
| 3 | 23 минуты 07 секунд | 10 минут 17 секунд |
| 4 | 21 минута 18 секунд | 8 минут 55 секунд |
| 5 | 20 минут 17 секунд | 10 минут 13 секунды |
| Среднее время | 20 минут 48 секунды | 10 минут 9 секунд |

Также на рисунке 65 можно увидеть время, требуемое для работы генетического алгоритма до оптимизации кода.




| Пары | Понедельник | Вторник | Среда | Четверг | Пятница | Суббота |
|---|---|---|---|---|---|---|
| 08:00-09:30 | История<br>Алексей Алексеев<br>Аудитория 7 | История<br>Алексей Алексеев<br>Аудитория 7 | История<br>Алексей Алексеев<br>Аудитория 7 | География<br>Николай Николаев<br>Аудитория 7 | География<br>Николай Николаев<br>Аудитория 7 | История<br>Алексей Алексеев<br>Аудитория 6 |
| 09:40-11:10 | История<br>Алексей Алексеев<br>Аудитория 7 | Литература<br>Андрей Андреев<br>Аудитория 6 | География<br>Николай Николаев<br>Аудитория 7 | Литература<br>Андрей Андреев<br>Аудитория 7 | География<br>Николай Николаев<br>Аудитория 7 | Химия<br>Сергей Сергеев<br>Аудитория 7 |
| 11:30-13:00 | Биология<br>Дмитрий Дмитриев<br>Аудитория 7 | Физика<br>Петр Петров<br>Аудитория 6 | География<br>Николай Николаев<br>Аудитория 7 | Литература<br>Андрей Андреев<br>Аудитория 7 | Химия<br>Сергей Сергеев<br>Аудитория 6 | История<br>Алексей Алексеев<br>Аудитория 7 |
| 13:10-14:40 | География<br>Николай Николаев<br>Аудитория 7 | Литература<br>Андрей Андреев<br>Аудитория 2 | Химия<br>Сергей Сергеев<br>Аудитория 6 | Химия<br>Сергей Сергеев<br>Аудитория 2 | Физика<br>Петр Петров<br>Аудитория 7 | Физика<br>Петр Петров<br>Аудитория 6 |
| 14:50-16:20 | Математика<br>Иван Иванов<br>Аудитория 4 | Химия<br>Сергей Сергеев<br>Аудитория 5 | Физика<br>Петр Петров<br>Аудитория 3 | История<br>Алексей Алексеев<br>Аудитория 7 | История<br>Алексей Алексеев<br>Аудитория 7 | География<br>Николай Николаев<br>Аудитория 6 |
| 16:30-18:00 | Математика<br>Иван Иванов<br>Аудитория 7 | Математика<br>Иван Иванов<br>Аудитория 6 | География<br>Николай Николаев<br>Аудитория 5 | История<br>Алексей Алексеев<br>Аудитория 7 | Химия<br>Сергей Сергеев<br>Аудитория 3 | Математика<br>Иван Иванов<br>Аудитория 7 |
| 18:10-19:40 | Химия<br>Сергей Сергеев | История<br>Алексей Алексеев | Химия<br>Сергей Сергеев | Математика<br>Иван Иванов | Биология<br>Дмитрий Дмитриев | География<br>Николай Николаев |

Рисунок 65 – Время оптимизации до улучшения

А на рисунке 66 можно увидеть время, полученное после оптимизации кода генетического алгоритма.

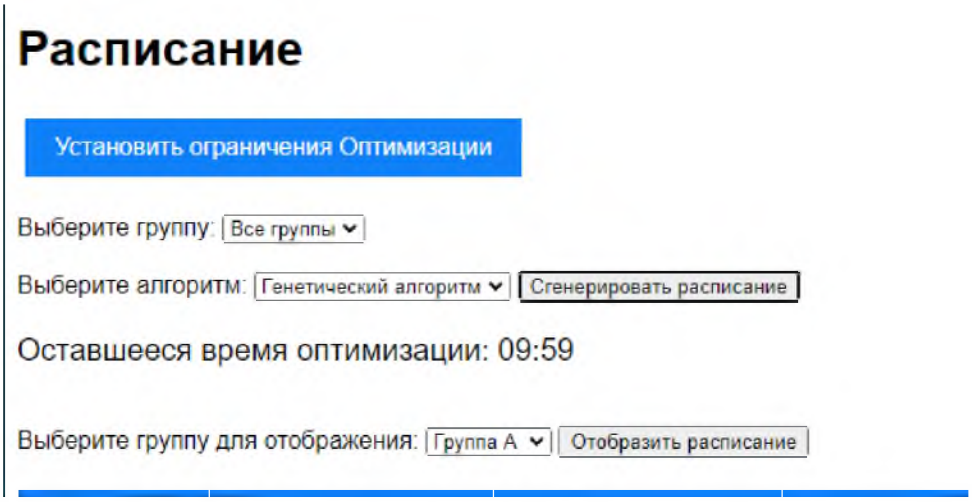



Рисунок 66 – Время оптимизации после внесения улучшений

Проанализировав данные тестов в таблице 6, и практические результаты показанные на рисунке 65 и рисунке 66 можно утверждать, что в среднем время

на оптимизацию сократилось в 2 раза это означает что общее улучшение производительности составило около 50%, что является хорошим показателем.

### 3.6 Итоги третьего раздела

В данном разделе представлены создание базы данных для проекта Django. Представлено создание и подготовка для работы проекта Django. Представлена разработка «Жадного алгоритма», полностью описан код и поэтапно описаны все функции, выполняемые этим кодом.

Также описан «Генетический алгоритм» и его код.

Поэтапно описаны все функции кода этого алгоритма. Далее представлен макет интерфейса с некоторым функционалом, который в дальнейшем будет перенесён на проект Django.

В итоге проведено тестирование системы с описанием функционала, в результате которого, можно сделать вывод о правильной работе алгоритмов, также проведён эксперимент по использованию алгоритмов.

# Заключение

В процессе прохождения выпускной квалификационной работы сформированы цели и задачи практики. Выполнено формирование цели и задач исследования. Определен объект и предмет исследования. Обоснована актуальность исследования.

В результате выполнения выпускной квалификационной работы представлен анализ веб технологий, анализ аналогов и их сравнение после чего было принято решение о разработке собственной системы.

Описаны фреймворки, языки программирования и алгоритмы реализации, в результате чего в качестве фреймворка выбран Django, в качестве языка программирования выбран Python, а в качестве алгоритмов реализации была выбрана комбинация жадного и генетического алгоритма с интерактивным подходом.

Представлен анализ объекта практики – университета, проведен сравнение и выбор подходов и была выбрана веб-разработка как более перспективная. Представлено проектирование UML-диаграмм, а именно диаграммы вариантов использования и диаграммы последовательности, а также представлено описание работы алгоритма при помощи блок-схемы, описана работа системы с работой алгоритмов, выбранных ранее для построения системы, проведено сравнение IDE и выбран PyCharm потому, что он наиболее подходящий для работы с фреймворком Django, представлен проект интерфейса будущей системы.

Создана база данных, созданы модели Django для работы с базой данных, написан код алгоритмов на Python, и другие элементы системы, созданной с помощью Django. Проведено тестирование системы с положительным результатом, а также проведён эксперимент по использованию алгоритмов.

В результате выполнения поставленных задач удалось получить ценные практические и теоретические навыки, которые могут быть использованы для оптимизации процессов управления расписанием в образовательных

учреждениях. Актуальность данного исследования заключается в повышения эффективности организации образовательного процесса и улучшения условий для студентов и преподавателей.

Заключительные выводы работы позволяют сделать предположения о возможных направлениях дальнейших исследований и реализации в данной области, а также подчеркивают важность развития информационных технологий для совершенствования управления учебными процессами.

Список литературы

1. Бедердинова, О. И. Создание приложений баз данных в среде Visual Studio: учебное пособие / О.И. Бедердинова, Т.А. Минеева, Ю.А. Водовозова. — Москва: ИНФРА-М, 2021. — 94 с. - ISBN 978-5-16-109411-2. - Текст: электронный. - [Электронный ресурс].URL: https://znanium.com/catalog/product/1243816 (дата обращения: 23.12.2023)

2. Белов, В. В. Алгоритмы и структуры данных: учеб. / В.В. Белов, В.И. Чистякова. — Москва: КУРС: ИНФРА-М, 2023. — 240 с. — (Бакалавриат). - ISBN 978-5-906818-25-6. - Текст: электронный. -[Электронный ресурс]. URL: https://znanium.com/catalog/product/2110058 (дата обращения: 05.10.2024).

3. Буч, Г. Язык UML. Руководство пользователя: практическое руководство / Г. Буч, Дж. Рамбо, И. Якобсон; пер. с англ. Н. Мухина. — 3-е изд. - Москва: ДМК Пресс, 2022. - 495 с. - ISBN 978-5-89818-247-2. - Текст: электронный. -[Электронный ресурс]. URL: https://znanium.com/catalog/product/2110008 (дата обращения: 05.10.2024).

4. Вагин, Д. В. Современные технологии разработки веб-приложений: учебное пособие / Д. В. Вагин, Р. В. Петров. - Новосибирск: Изд-во НГТУ, 2019. - 52 с. - ISBN 978-5-7782-3939-5. - Текст: электронный. - [Электронный ресурс].URL: https://znanium.com/catalog/product/1866926 (дата обращения: 05.10.2024).

5. Голицына, О. Л. Программное обеспечение: учеб. пособие / О. Л. Голицына, Т. Л. Партыка, И. И. Попов. - 4-е изд., перераб. и доп. - Москва: ФОРУМ: ИНФРА-М, 2021. - 448 с.: ил. - (Профессиональное образование). - ISBN 978-5-91134-711-6. - Текст: электронный. - [Электронный ресурс].URL: https://znanium.ru/catalog/product/1189345 (дата обращения: 05.10.2024).

6. Голицына, О. Л. Языки программирования: учебное пособие / О.Л. Голицына, Т.Л. Партыка, И.И. Попов. — 3-е изд., перераб. и доп. — Москва: ФОРУМ: ИНФРА-М, 2023. — 399 с. — ISBN 978-5-00091-613-1. - Текст: электронный. - URL: https://znanium.com/catalog/product/1941740 (дата

обращения: 05.10.2024).

7. Гома, Х. UML. Проектирование систем реального времени, параллельных и распределенных приложений: практическое руководство / Х. Гома; пер. с англ. А. А. Слинкина. - 2-е изд. - Москва: ДМК Пресс, 2023. - 701 с. - (Объектно-ориентированные технологии в программировании). - ISBN 978-5-89818-574-9. - Текст: электронный. - [Электронный ресурс].URL: https://znanium.com/catalog/product/2107936 (дата обращения: 05.10.2024).

8. Гуриков, С. Р. Основы алгоритмизации и программирования на Python: учебное пособие / С.Р. Гуриков. — Москва: ИНФРА-М, 2023. — 343 с. — (Среднее профессиональное образование). - ISBN 978-5-16-016906-4. - Текст: электронный. - [Электронный ресурс].URL: https://znanium.ru/catalog/product/1927269 (дата обращения: 05.10.2024).

9. Колокольникова, А. И. Базовый инструментарий Moodle для развития системы поддержки обучения / А. И. Колокольникова. - 2-е изд. - Москва: Директ-Медиа, 2020. - 291 с. - ISBN 978-5-4499-1543-6. - Текст: электронный. -[Электронный ресурс]. URL: https://znanium.com/ catalog/product/1979900 (дата обращения: 05.10.2024).

10. Козловский, П. Разработка веб-приложений с использованием AngularJS: практическое руководство / П. Козловский, П. Б. Дарвин; пер. с англ. А. Н. Киселёва. - 2-е изд. - Москва: ДМК Пресс, 2023. - 395 с. - ISBN 978-5-89818-539-8. - Текст: электронный. - [Электронный ресурс].URL: https://znanium.com/catalog/product/2107216 (дата обращения: 05.10.2024).

11. Ландовский, В. В. Алгоритмы обработки данных: учеб. пособие / В. В. Ландовский. - Новосибирск: Изд-во НГТУ, 2018. - 67 с. - ISBN 978-5-7782-3645-5. - Текст: электронный. - [Электронный ресурс].URL: https://znanium.com/catalog/product/1869248 (дата обращения: 05.10.2024).

12. Лион, У. Разработка веб-приложений GraphQL с React, Node.js и Neo4j: практическое руководство / У. Лион; пер. с англ. А. Н. Киселева. - Москва: ДМК Пресс, 2023. - 262 с. - ISBN 978-5-93700-185-6. - Текст: электронный. - [Электронный ресурс].URL: https://znanium.com/catalog/product

/2109522 (дата обращения: 05.10.2024).

13. Меле, А. Django 4 в примерах / А. Меле. - Москва: ДМК Пресс, 2023. - 801 с. - ISBN 978-5-93700-204-4. - Текст: электронный. - [Электронный ресурс].URL: https://znanium.ru/catalog/product/2150534 (дата обращения: 05.10.2024).

14. Миковски, М. Разработка одностраничных веб-приложений: практическое руководство / М. Миковски, Дж. К. Пауэлл; пер. с англ. А. А. Слинкина. - 2-е изд - Москва: ДМК Пресс, 2023. - 514 с. - ISBN 978-5-89818-353-0. - Текст: электронный. - [Электронный ресурс].URL: https://znanium.com/catalog/product/2103586 (дата обращения: 05.10.2024).

15. Официальный сайт «Atom» - [Электронный ресурс]- URL: https://atom-editor.cc/ (дата обращения: 23.12.2024).

16. Официальный сайт «ПервыйБит» БИТ.ВУЗ.Расписание – [Электронный ресурс] - URL: https://www.1cbit.ru/1csoft/bit-vuz-raspisanie/ (дата обращения: 15.01.2023).

17. Официальный сайт «ГалактикаИТ» «Галактика Расписание учебных занятий» - [Электронный ресурс]- URL: https://galaktika-it.ru/spb/ruz (дата обращения: 05.10.2024).

18. Официальный сайт Отраслевые и специализированные решения 1С: Предприятие, «1С: Автоматизированное составление расписания.Университет» - [Электронный ресурс] - URL https://solutions.1c.ru/catalog/asp_univer/features (дата обращения: 05.10.2024).

19. Официальный сайт «Лаборатория ММИС», «АВТОРасписание» [Электронный ресурс]. – URL https://www.mmis.ru/programs/avtor (дата обращения: 05.10.2024).

20. Официальный сайт «PyCharm: IDE» - [Электронный ресурс] - URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата обращения: 23.12.2024).

21. Постольник В.С., Концепция автоматизированной системы составления расписания образовательной организации / В.С Постольник., К.Н. Цебренко // Международный журнал гуманитарных и естественных наук:

Технические науки. – 2024. – №5-4(92). – С. 43-48.

22. Титов, А. Н. Python. Обработка данных: учебно-методическое пособие / А. Н. Титов, Р. Ф. Тазиева; Минобрнауки России, Казан. нац. исслед. технол. ун-т. - Казань: Изд-во КНИТУ, 2022. - 104 с. - ISBN 978-5-7882-3171-6. - Текст: электронный. - [Электронный ресурс]. URL: https://znanium.com/catalog/product/2069264 (дата обращения: 05.10.2024).

23. Фаррелл, Б. Веб-компоненты в действии: практическое руководство / Б. Фаррелл; пер. с англ. Д. А. Беликов. - Москва: ДМК Пресс, 2021. - 462 с. - ISBN 978-5-97060-856-2. - Текст: электронный. - [Электронный ресурс]. URL: https://znanium.com/catalog/product/1210671 (дата обращения: 05.10.2024).

24. Хорошева, А. В. Сетевые информационные ресурсы и технологии в профессиональной деятельности сотрудников уголовно-исполнительной системы: учебное пособие / А. В. Хорошева. – Владимир: ВЮИ ФСИН России, 2020. - 68 с. – ISBN 978-5-93035-693-9. - Текст: электронный. -[Электронный ресурс]. URL: https://znanium.com/catalog/product/1863398 (дата обращения: 05.10.2024).

25. Хэррон, Д. Node.js. Разработка серверных веб-приложений на JavaScript: практическое руководство / Д. Хэррон; пер. с англ. А. А. Слинкина. - 2-е изд. - Москва: ДМК Пресс, 2023. - 145 с. - ISBN 978-5-89818-632-6. - Текст: электронный. - [Электронный ресурс].URL: https://znanium.com/catalog/product/2108525 (дата обращения: 05.10.2024).

26. Хортон, А. Разработка веб-приложений в ReactJS: практическое руководство / А. Хортон, Р. Вайс; пер. с англ. Р. Н. Рагимова. - 2-е изд. - Москва: ДМК Пресс, 2023. - 255 с. - ISBN 978-5-89818-503-9. - Текст: электронный. - [Электронный ресурс]. URL: https://znanium.com/catalog/product/2107174 (дата обращения: 05.10.2024).

27. Чэпел, Э. AutoCAD® Civil 3D® 2014. Официальный учебный курс : учебное пособие / Э. Чэпел ; пер. с англ. А. В. Снастина, С. П. Ивженко. - 2-е изд. - Москва : ДМК Пресс, 2023. - 441 с. - ISBN 978-5-89818-355-4. - Текст : электронный. - [Электронный ресурс].URL: https://znanium.ru/catalog/product/

2103588 (дата обращения: 10.01.2025)

28. Черняховская, Ю. И. Проектирование информационной системы по формированию расписания занятий в учебном заведении / Ю. И. Черняховская, Д. И. Дударев, О. В. Дударева // Информатизация и виртуализация экономической и социальной жизни: материалы VI Студенческой научно-практической конференции с международным участием, Иркутск, 14 мая 2019 года. – Иркутск: Иркутский национальный исследовательский технический университет, 2019. – С. 104-108. – EDN HDDLLU.

29. Яковлева, Н.Ф. Проектная деятельность в образовательном учреждении: учеб. пособие / Н.Ф. Яковлева. - 3-е изд., стер. - Москва: ФЛИНТА, 2020. - 144 с. - ISBN 978-5-9765-1895-7. - Текст: электронный. - [Электронный ресурс]. URL: https://znanium.com/catalog/product/1042547 (дата обращения: 05.10.2024).

30. Янбердин, Р. Р. Управление командой разработки программного обеспечения на no-code платформе / Р. Р. Янбердин, О. Б. Назарова // Управление проектами : сборник статей по материалам Всероссийской научной конференции, Магнитогорск, 14–15 декабря 2022 года. – Магнитогорск: «ФГБОУ ВО «МГТУ им. Г.И. Носова»», 2023. – С. 184-188. – EDN FRSSYZ.

# Приложение 1

## Python код «жадного» алгоритма

```
import random
from .models import Schedule, Discipline, Teacher, Audience, Group

def generate_random_schedule(group_id=None, constraints=None):
    # Удаляем существующее расписание для указанных групп
    Schedule.objects.all().delete() if group_id == 'all' else
Schedule.objects.filter(group_id=group_id).delete()

    # Получаем все группы или конкретную группу
    groups = list(Group.objects.all() if group_id == 'all' else Group.objects.filter(id=group_id))
    days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
    time_slots = ['08:00-09:30', '09:40-11:10', '11:30-13:00', '13:10-14:40', '14:50-16:20', '16:30-
18:00', '18:10-19:40']

    # Создаем словарь для отслеживания числа назначенных пар для каждой группы в каждый
день
    group_schedule_count = {group.id: {day: 0 for day in days_of_week} for group in groups}

    def find_available_audience(day, time_slot):
        for audience in Audience.objects.all():
            if not Schedule.objects.filter(audience=audience, day_of_week=day,
time_slot=time_slot).exists():
                return audience
        return None

    # Генерируем расписание по круговому принципу
    for day in days_of_week:
        for time_slot in time_slots:
            for group in groups:
                if group_schedule_count[group.id][day] >= len(time_slots):
                    continue  # Переходим к следующей группе, если у этой группы уже заполнены
все слоты на день

                disciplines = list(group.disciplines.all())
                random.shuffle(disciplines)

                assigned = False
                for discipline in disciplines:
                    teacher = Teacher.objects.filter(discipline=discipline).first()
                    if not teacher:
                        continue

                    # Проверка на занятость преподавателя в это время
                    if Schedule.objects.filter(teacher=teacher, day_of_week=day,
time_slot=time_slot).exists():
                        continue

                    # Проверка, что не более 2 одинаковых пар в день
```

```
                if Schedule.objects.filter(group=group, discipline=discipline,
day_of_week=day).count() >= 2:
                    continue

                audience = find_available_audience(day, time_slot)
                if audience:
                    schedule_entry = Schedule(
                        discipline=discipline,
                        teacher=teacher,
                        audience=audience,
                        group=group,
                        day_of_week=day,
                        time_slot=time_slot
                    )
                    schedule_entry.save()
                    group_schedule_count[group.id][day] += 1
                    assigned = True
                    break

            if not assigned:
                print(f"Could not assign a schedule for group {group.name} at {day} {time_slot}")

    # Заполняем оставшиеся пустые ячейки случайными парами
    for day in days_of_week:
        for time_slot in time_slots:
            for group in groups:
                if not Schedule.objects.filter(group=group, day_of_week=day,
time_slot=time_slot).exists():
                    disciplines = list(group.disciplines.all())
                    random.shuffle(disciplines)

                    for discipline in disciplines:
                        teacher = Teacher.objects.filter(discipline=discipline).first()
                        if not teacher:
                            continue
                # Проверка на занятость преподавателя в это время
            If Schedule.objects.filter(teacher=teacher, day_of_week=day, time_slot=time_slot).exists():
                continue

                        audience = find_available_audience(day, time_slot)
                        if audience:
                            schedule_entry = Schedule(
                                discipline=discipline,
                                teacher=teacher,
                                audience=audience,
                                group=group,
                                day_of_week=day,
                                time_slot=time_slot
                            )
                            schedule_entry.save()
                            break
```

# Приложение 2

## Python код «генетического» алгоритма

```
import random
import numpy as np
from .models import Schedule, Discipline, Teacher, Audience, Group

def generate_random_schedule(group_id=None):
    groups = Group.objects.all() if group_id == 'all' else Group.objects.filter(id=group_id)
    days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
    time_slots = ['08:00-09:30', '09:40-11:10', '11:30-13:00', '13:10-14:40', '14:50-16:20',
'16:30-18:00', '18:10-19:40']

    for group in groups:
        Schedule.objects.filter(group=group).delete()
        all_disciplines = list(group.disciplines.all())
        random.shuffle(all_disciplines)

        for discipline in all_disciplines:
            teacher = Teacher.objects.filter(discipline=discipline).first()
            assigned_slots = set()

            for _ in range(7):
                while True:
                    day = random.choice(days_of_week)
                    time_slot = random.choice(time_slots)
                    if (day, time_slot) not in assigned_slots:
                        audience = find_available_audience(day, time_slot)
                        if audience:
                            Schedule.objects.create(
                                discipline=discipline,
                                teacher=teacher,
                                audience=audience,
                                group=group,
                                day_of_week=day,
                                time_slot=time_slot
                            )
                            assigned_slots.add((day, time_slot))
                            break

def can_schedule(group, discipline, teacher, day, time, max_gaps, max_classes_per_day):
    schedules = Schedule.objects.filter(group=group,
day_of_week=day).order_by('time_slot')
    time_slots = ['08:00-09:30', '09:40-11:10', '11:30-13:00', '13:10-14:40', '14:50-16:20',
'16:30-18:00', '18:10-19:40']

    if schedules.exists():
        last_schedule_time = schedules.last().time_slot
        last_schedule_index = time_slots.index(last_schedule_time)
        current_time_index = time_slots.index(time)
```

```
        if current_time_index - last_schedule_index - 1 > max_gaps:
            return False
    return not Schedule.objects.filter(group=group, day_of_week=day,
time_slot=time).exists() and \
           not Schedule.objects.filter(teacher=teacher, day_of_week=day,
time_slot=time).exists() and \
           schedules.count() < max_classes_per_day

def find_available_audience(day, time):
    for audience in Audience.objects.all():
        if not Schedule.objects.filter(audience=audience, day_of_week=day,
time_slot=time).exists():
            return audience
    return None

def genetic_algorithm(group_id=None, constraints=None):
    if constraints is None:
        constraints = {'max_gaps': 0, 'max_shift': 0, 'min_classes_per_day': 4,
'max_classes_per_day': 7}

    max_gaps = int(constraints.get('max_gaps', 0))
    max_shift = int(constraints.get('max_shift', 0))
    min_classes_per_day = int(constraints.get('min_classes_per_day', 4))
    max_classes_per_day = int(constraints.get('max_classes_per_day', 7))

    generate_random_schedule(group_id)

    groups = Group.objects.all() if group_id == 'all' else Group.objects.filter(id=group_id)
    days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
    time_slots = ['08:00-09:30', '09:40-11:10', '11:30-13:00', '13:10-14:40', '14:50-16:20',
'16:30-18:00', '18:10-19:40']

    population_size = 50
    generations = 1000
    mutation_rate = 0.1

    def fitness(schedule):
        fitness_score = 0
        for group in groups:
            for day in days_of_week:
                daily_schedules = schedule.filter(group=group, day_of_week=day)
                num_classes = daily_schedules.count()
                if num_classes < min_classes_per_day or num_classes > max_classes_per_day:
                    fitness_score -= abs(num_classes - ((min_classes_per_day +
max_classes_per_day) / 2))
                last_schedule_time = None
                for sched in daily_schedules.order_by('time_slot'):
                    if last_schedule_time:
                        last_schedule_index = time_slots.index(last_schedule_time)
                        current_time_index = time_slots.index(sched.time_slot)
                        gaps = current_time_index - last_schedule_index - 1
                        if gaps > max_gaps:
```

```
                    fitness_score -= gaps
                last_schedule_time = sched.time_slot
        return fitness_score

    def crossover(parent1, parent2):
        crossover_point = random.randint(0, len(parent1))
        child1 = parent1[:crossover_point] + parent2[crossover_point:]
        child2 = parent2[:crossover_point] + parent1[crossover_point:]
        return child1, child2

    def mutate(schedule):
        if random.random() < mutation_rate:
            group = random.choice(groups)
            day = random.choice(days_of_week)
            time_slot = random.choice(time_slots)
            discipline = random.choice(group.disciplines.all())
            teacher = Teacher.objects.filter(discipline=discipline).first()
            audience = find_available_audience(day, time_slot)
            if audience and can_schedule(group, discipline, teacher, day, time_slot, max_gaps,
max_classes_per_day):
                schedule_entry = Schedule(
                    discipline=discipline,
                    teacher=teacher,
                    audience=audience,
                    group=group,
                    day_of_week=day,
                    time_slot=time_slot
                )
                schedule_entry.save()

    population = []
    for _ in range(population_size):
        generate_random_schedule(group_id)
        population.append(Schedule.objects.all())

    for _ in range(generations):
        population = sorted(population, key=fitness, reverse=True)
        new_population = population[:10]
        while len(new_population) < population_size:
            parent1, parent2 = random.sample(population[:20], 2)
            child1, child2 = crossover(list(parent1), list(parent2))
            new_population.append(child1)
            new_population.append(child2)
        for individual in new_population:
            mutate(individual)
        population = new_population

    best_schedule = population[0]
    Schedule.objects.all().delete()
    for sched in best_schedule:
        sched.save()
```