ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Программная инженерия»

УТВЕРЖДАЮ
Академический руководитель
образовательной программы «Программная инженерия»,
старший преподаватель департамента программной инженерии
_____ Н.А. Павлочев
«16» мая _____ 2025 г.

Выпускная квалификационная работа

на тему **Веб-сервис составления расписаний для вузов с использованием искусственного интеллекта: Серверная часть**
по направлению подготовки 09.03.04 «Программная инженерия»

Научный руководитель
НИУ ВШЭ профессор ДПИ
Должность, место работы
к.т.н.
ученая степень, ученое звание
В. В. Шилов
И.О. Фамилия
16 мая 2025г.
Подпись, Дата

Выполнил
студент группы БПИ214
4 курса бакалавриата
образовательной программы
«Программная инженерия»
А. М. Габдуллина
И.О. Фамилия
16 мая 2025г.
Подпись, Дата

Соруководитель
ведущий математик-разработчик
АО «Тераплан»
Должность, место работы
Ph.D.
ученая степень, ученое звание
Д. И. Архипов
И.О. Фамилия
16 мая 2025г.
Подпись, Дата

Москва 2025

Реферат

Данный проект представляет веб-приложение для автоматизированного составления учебных расписаний в университетах с использованием искусственного интеллекта. Система учитывает различные ограничения и предпочтения и позволяет оптимизировать процесс распределения аудиторий, преподавателей и студентов.

Разработанный веб-сервис позволяет оптимизировать процесс составления расписаний с учетом возможных ограничений. Взаимодействие с пользователями осуществляется через веб-интерфейс и чат-бот на базе Yandex GPT API, который анализирует запросы и передает структурированные данные на сервер, чат-бот используется для взаимодействия с пользователем: ответов на его вопросы, добавления данных (в том числе в виде файлов), редактирования или удаления данных. Чат-бот позволяет упростить ввод данных для пользователя и отойти от унифицированного формата файлов. На серверной части, реализованной на Java Spring Boot, реализован алгоритм, формирующий расписание с учетом заданных ограничений.

Решение снижает ручную работу, минимизирует конфликты и улучшает процесс планирования. Использование ИИ делает систему адаптивной и удобной для образовательных учреждений, повышая эффективность управления расписаниями.

Работа содержит 26 страницы, 3 главы, 3 иллюстрации, 14 источников и 0 приложений.

Ключевые слова: *веб-приложение; серверная часть; составление расписаний; университетское расписание.*

Abstract

This project presents a web application for the automated creation of academic schedules in universities using artificial intelligence. The system considers various constraints and preferences, optimizing the allocation of classrooms, instructors, and students.

The project optimizes the scheduling process while accounting for potential constraints. User interaction is facilitated through a web interface and a chatbot based on Yandex GPT API, which analyzes requests and transmits structured data to the server, chat bot is used for user interaction: answer questions, add new data (also in different file format), edit or delete data. Chat bot makes data interaction for user simpler and more flexible. The server-side, implemented in Java Spring Boot, runs an algorithm that generates schedules based on the given constraints.

The solution reduces manual effort, minimizes conflicts, and enhances the planning process. The use of AI makes the system adaptive and user-friendly for educational institutions, improving scheduling efficiency.

The work contains 26 pages, 3 chapters, 3 illustrations, 14 sources, and 0 applications.

Keywords: web application; backend; scheduling; university timetable.

Оглавление

Основные определения, термины и сокращения

API — программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими, включающий в себя набор операций, которые он может выполнить, а также описание входных и выходных данных.

JSON — текстовый формат обмена данными, легко читаемый человеком и удобно структурируемый программой.

Клиент-серверная архитектура — это модель взаимодействия, в которой задачи распределены между клиентами и серверами.

Веб-приложение — клиент-серверное приложение, которое выполняется на веб-сервере.

Солвер (MILP) — программа, реализующая решение задачи целочисленного линейного программирования.

MILP модель — модель целочисленного линейного программирования, формализующая задачу в виде системы линейных уравнений. Может быть решена при помощи солверов.

Введение

В современных университетах составление расписаний является сложной и трудоемкой задачей. Администраторы должны учитывать множество факторов: доступность преподавателей и аудиторий, учебные планы, индивидуальные планы студентов и предпочтения преподавателей. Ручное планирование приводит к значительным временным затратам и возможным конфликтам в расписании, а существующие решения имеют ряд недостатков и подходят не всем.

Разрабатываемая система автоматизированного составления расписаний с использованием искусственного интеллекта решает эту проблему, упрощая процесс планирования. Взаимодействие пользователей с системой осуществляется через веб-интерфейс и чат-бот на базе Yandex GPT API, который анализирует запросы и преобразует их в структурированные данные для дальнейшей обработки. Серверная часть, выполняет расчеты и формирует оптимизированное расписание.

Использование ИИ позволяет существенно снизить нагрузку на администраторов, минимизировать ошибки и повысить гибкость планирования. Благодаря такому подходу, администратору не требуется обладать специальными навыками взаимодействия со сложным приложением, а система не требует строгого формата входных данных. Это значительно упрощает взаимодействие и делает процесс планирования доступным для широкого круга пользователей без необходимости дополнительного обучения.

Целью данной работы является реализация серверной части веб-приложения "Система составления расписания для высших учебных заведений с использованием искусственного интеллекта".

Задачи работы:

- Проанализировать уже существующие программы для составления расписаний в университетах.
- Проанализировать существующие подходы к решению задачи составления расписаний.
- Выбрать методы, модели, средства разработки.
- Спроектировать архитектуру решения и базу данных.
- Создать базу данных для хранения информации.
- Реализовать механизмы выгрузки данных из базы данных и их передачу на UI.
- Продумать контракты взаимодействия клиентской, серверной части и модуля с чат-ботом.

- Разработать обработчики запросов на сервере для взаимодействия с пользователями и алгоритмами планирования.
- Реализовать алгоритм составления расписания, учитывающий различные ограничения и предпочтения пользователей.
- Реализовать пост-анализ полученного расписания: проверка корректности, предложение нескольких вариантов решения.
- Анализ эффективности алгоритма.

Выпускная квалификационная работа состоит из трех глав.

В первой главе проанализирована предметная область, а также приводится обзор и анализ существующих программных решений и сравнение их с данной работой по нескольким критериям: реализация программы, учет ограничений, необходимость специализированной настройки, автоматизация составления расписаний для администраторов и так далее.

Во второй главе приведено описание основного функционала системы и обзор выбранных для реализации подходов, технических средств и инструментов разработки. Также в данной главе приводится архитектура всего решения в целом, схема серверной части и базы данных.

В третьей главе представлено полное описание разработанного алгоритма составления расписаний, а также проведено его сравнение с конкурентами. Для оценки эффективности рассматриваются различные подходы к генерации расписаний, и проводится их анализ с точки зрения производительности, точности и временных затрат.

В ходе исследования тестируется используемый алгоритм, позволяющий автоматизировать процесс планирования. Производительность алгоритма оценивается по ключевым метрикам, таким как время выполнения, качество сформированных расписаний (исходя из целевой функции) и соответствие заданным ограничениям.

В заключении подводятся итоги работы и предложения для дальнейшего улучшения программы.

Глава 1. Предметная область и существующие решения

1.1. Описание предметной области и актуальность

Предметная область данного дипломного проекта связана с автоматизацией процесса составления расписаний для высших учебных заведений. Составление расписаний является одной из наиболее сложных и трудоемких задач в образовательных учреждениях, так как требует учета множества факторов, таких как доступность аудиторий, расписание преподавателей, учебные планы студентов, а также индивидуальные предпочтения и ограничения.

Данная выпускная квалификационная работа посвящена разработке интеллектуальной системы для автоматизации процесса составления учебных расписаний в вузах. Формирование расписания представляет собой сложную многокритериальную задачу, требующую учета множества ограничений: наличие свободных аудиторий, занятость преподавателей, специфику учебных планов студентов, а также индивидуальные предпочтения участников образовательного процесса. На практике этот процесс зачастую осуществляется вручную, что делает его ресурсоемким, подверженным ошибкам и неэффективным. Существующие программные решения либо предъявляют строгие требования к формату входных данных, либо требуют высокой квалификации от пользователя. Кроме того, не всегда удается достичь высокого качества расписания, способного удовлетворить требования конкретного вуза.

Актуальность разработки системы автоматизированного составления расписаний с использованием искусственного интеллекта обусловлена нехваткой на рынке приложения, сочетающего в себе веб-интерфейс, отсутствие шаблонов для ввода данных и качественный алгоритм составления расписаний, учитывающий предпочтения преподавателей.

1.2. Описание существующих решений

В настоящее время на рынке существует несколько решений для автоматизации составления расписаний в высших учебных заведениях. Ниже приведены описания популярных сервисов, а также их сравнение:

1.2.1. Ректор-ВУЗ

Ректор-ВУЗ [14] — система для управления учебным процессом, которая включает модуль для составления расписаний. Сервис учитывает такие параметры, как доступность аудиторий, расписание преподавателей, учебные планы студентов, а также индивидуальные предпочтения и

ограничения. Система поддерживает автоматическую генерацию расписаний с возможностью ручной корректировки. Программа работает только на компьютерах с оперативной системой Windows.

1.2.2. Экспресс-Расписание ВУЗ

Экспресс-Расписание ВУЗ [15]— программа, которая составляет расписаний в вузах. Сервис учитывает все необходимые параметры для составления расписания, а также индивидуальные предпочтения. Система поддерживает автоматическую генерацию расписаний с возможностью ручной корректировки. Также из преимуществ программы можно выделить поддержку многопользовательского режима и возможность интеграции с другими системами. Программа поддерживается на компьютерах с оперативной системой Windows или Linux. А также требует квалифицированных сотрудников для настройки сервиса.

1.2.3. 1С: Университет

1С: Университет [16]— это комплексная система управления учебным процессом, которая включает в себя в том числе и модуль для составления расписаний. При составлении расписания сервис учитывает доступность аудиторий, расписание преподавателей, учебные планы студентов и индивидуальные предпочтения. Система поддерживает автоматическую генерацию расписаний с возможностью ручной корректировки.

Серверная часть 1С:Университет реализована на платформе 1С:Предприятие. Она предоставляет собой API для интеграции с другими образовательными платформами и системами управления. Это является преимуществом системы, так как позволяет кастомизировать программу под себя, дает гибкость в настройке и возможность подключать сторонние сервисы. Однако, для настройки такой системы нужны квалифицированные специалисты, лицензия с высокой стоимостью и много ресурсов на настройку.

1.2.4. Magellan

Magellan [17] — это облачное решение для составления расписаний, которое учитывает такие параметры, как доступность аудиторий, расписание преподавателей, учебные планы студентов и индивидуальные предпочтения. Сервис также поддерживает автоматическую генерацию расписаний с возможностью ручной корректировки.

Серверная часть **Magellan** реализована на основе облачных технологий, что обеспечивает высокую доступность и масштабируемость. Данный сервис является не просто программой для

составления расписаний, он содержит 17 модулей**,** охватывающих все участки образовательного процесса: Управление учебным процессом, Единая информационно-образовательная среда, Управление административной деятельностью, Управление хозяйственной частью и другие.

1.2.5. Сравнительный анализ

В результате анализа существующих программных решений для автоматизации расписаний в вузах был сформирован перечень ключевых функциональных требований, обеспечивающих удобство и эффективность использования подобных систем. На основе этих требований проведено сравнение представленных решений, включая разрабатываемое приложение. Сводные результаты представлены в таблице ниже:

Таблица 1. Сравнение сервисов для составления расписаний.

	Schedule. University (наше приложение) [1]	Ректор-ВУЗ [2]	Экспресс-расписание ВУЗ [3]	1С: Автоматизированное составление расписаний. Университет [4]	Magellan [5]
Веб-сервис	+	-	-	-	-
Операционные системы	Веб-сервис	Windows	Windows, Linux	Windows, Linux	Windows, macOS, Linux
Авто составление расписания	+	+	+	+	+
Ручная корректировка расписания	+	+	+	+	+
Пост-анализ решения, вывод подсказок	+	-	-	+	+
Создание нескольких вариантов расписаний одновременно	-	-	-	+	+
Использование искусственного интеллекта	+	-	-	-	-
Учёт пожеланий преподавателей	+	+	-	+	+

Не нужна специализированная настройка	+	-	+	-	-
Учет индивидуальных планов студентов	+	+	-	+	+
Замены преподавателя	-	+	+	-	-
Печать и экспорт	+	+	+	+	+

Проанализировав аналоги и составив сравнительную таблицу, можно сделать вывод, что функционал разработанной системы делает веб-сервис «Scheduling. University» конкурентоспособным продуктом на рынке.

Выводы по главе

В первой главе приведены описание предметной области, а также анализ существующих решений, приведена сравнительная таблица сервисов для составления университетских расписаний, показывающая недочеты существующий решений, в связи с которыми разрабатываемое веб-приложение является актуальным.

Глава 2. Проектирование системы и технологии разработки

Проектирование данного веб-сервиса «Система составления расписаний для вузов с использованием искусственного интеллекта. Серверная часть» включает анализ пользовательских сценариев, разработку архитектуры приложения и обоснование выбора технологий. Каждый этап проектирования направлен на обеспечение функциональности, надежности и удобства использования системы.

2.1. Пользовательские сценарии

Серверная часть системы взаимодействует исключительно с клиентским приложением через REST API, обрабатывая HTTP-запросы и возвращая соответствующие ответы. Пользовательские сценарии описывают типы запросов и их обработку:

- Сценарий генерации расписания

Клиентская часть отправляет POST-запрос на эндпоинт */api/schedule*, содержащий данные о преподавателях, аудиториях, группах студентов и учебных планах. Сервер валидирует входные данные, применяет алгоритмы составления расписания и его оптимизации и возвращает сгенерированное расписание в формате JSON. В случае ошибок (например, некорректные данные) сервер отправляет ответ с кодом 400 и детализацией проблемы.

- Сценарий получения расписания

Клиент отправляет GET-запрос на эндпоинт */api/schedule/{id}*, где id — идентификатор расписания. Сервер извлекает данные из базы, проверяет права доступа и возвращает расписание. Если расписание не найдено, возвращается код 404.

- Сценарий обновления данных

При изменении информации (например, добавлении нового преподавателя) клиент отправляет PUT-запрос на */api/teacher/create* или */api/teacher/delete/*{id} или */api/teacher/import* с обновленными данными. Сервер сохраняет изменения в базе данных и возвращает подтверждение. Транзакции обеспечивают целостность данных при одновременных запросах. Аналогично со всеми типами данных: курсами, группами, аудиториями, программами, студентами и их индивидуальными планами.

- Сценарий интеграции с чат-ботом

Чат бот взаимодействует с сервером через API. По POST-запросу */api/chat/upload* клиент передает сообщения (в том числе содержащие файлы), написанные пользователем в чате. По GET-запросу */api/chat/history* клиентская часть приложения получает историю чата для отображения в боте.

2.2. Архитектура решения

Архитектура системы построена по трехслойной модели клиент-сервер с выделенным внешним ИИ-сервисом. Клиентская часть — это веб-приложение, с которым пользователь взаимодействует через браузер. Серверная часть включает основное серверное приложение, реализующее бизнес-логику, управление данными и взаимодействие с базой данных. Кроме того, отдельным компонентом архитектуры выступает ИИ-модуль, развернутый как автономный HTTP-сервис на другом порту того же сервера. Этот модуль обрабатывает пользовательские запросы с помощью LLM-агентов и возвращает серверу структурированные данные, необходимые для составления расписания.

2.2.1. Схема базы данных

База данных спроектирована и размещена на сервере “jdbc:postgresql://82.97.244.207:5432/university”

Схема базы данных представлена на рисунке 1:

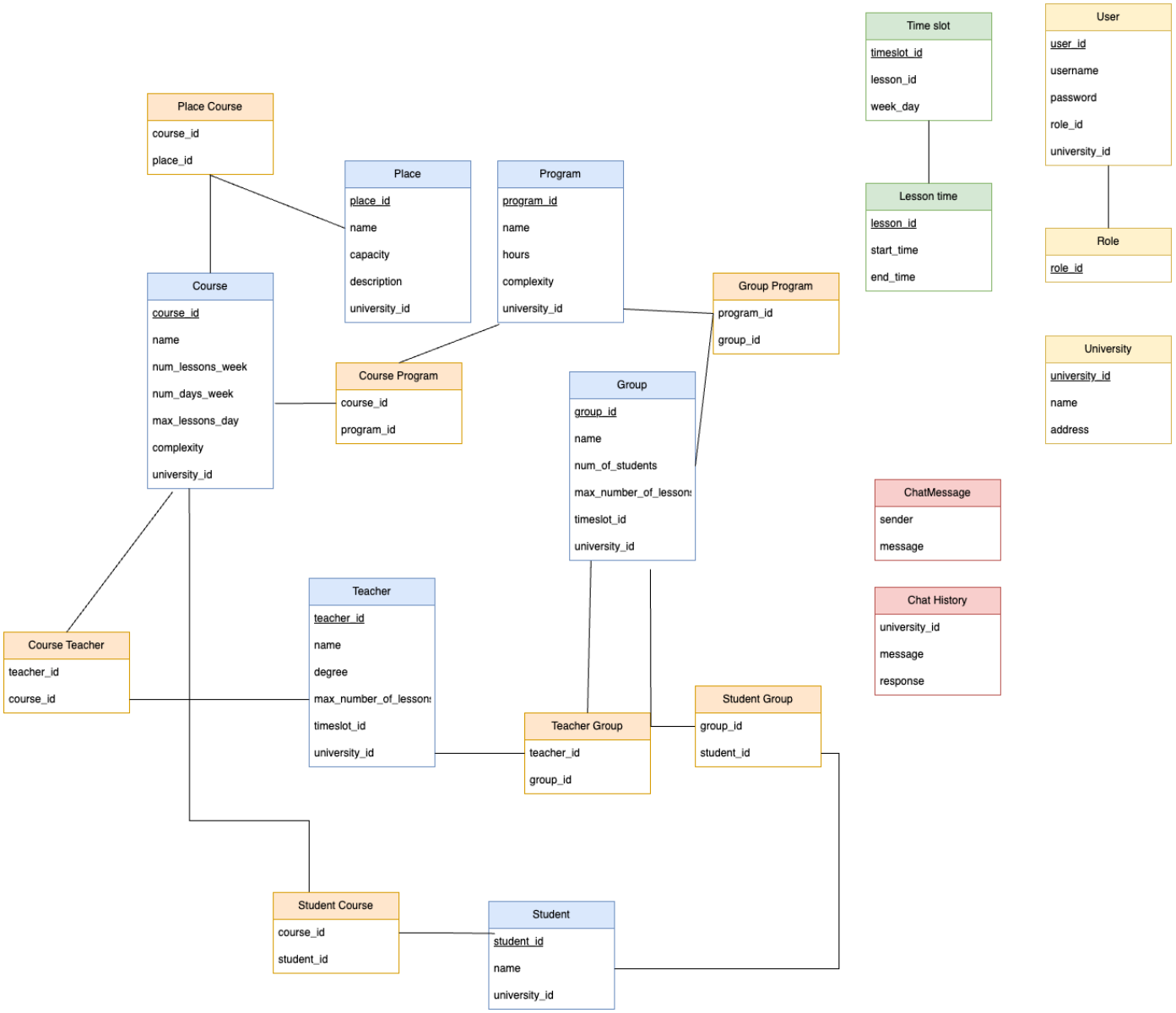

Рисунок 1 – Схема базы данных.

2.2.2. Архитектура приложения

Серверная часть приложения спроектирована с четким разделением ответственности между слоями. Она содержит 4 слоя, каждый из которых выполняет свою функцию:

1. Контроллеры – слой, отвечающий за принятие запросов с клиента, их обработку и отправку ответов.

2. Планировщик – слой, отвечающий за построение расписания.

3. Репозиторий – слой, взаимодействующий с базой данных.

4. ИИ – слой, взаимодействующий с Yandex GPT API для получения и анализа запросов с

клиента/сервера и осуществляющий пред-обработку данных и пост-обработку ответов от сервера для презентации их на клиенте в user-friendly формате. Этот слой представляет собой отдельный микро-сервис на Python.

Выбранный подход в архитектуре разграничивает ответственность между слоями, что способствует более простому добавлению функционала.

Общий принцип работы программы:

1. Запрос от клиентского приложения направляется на сервер через веб-приложение.
2. Если этот запрос был из чат-бота, ИИ модуль анализирует полученный запрос и группирует его на отдельные категории: замена данных, добавление/удаление ограничений и т.д. Далее запрос передается в основной модуль серверной части.
3. Сервер принимает запрос и обращается к базе данных для получения необходимой информации, редактирования или записи данных в нее.
4. База данных выполняет запрос и возвращает результаты на сервер.
5. Данные, полученные от базы данных обрабатываются сервером, который формирует ответ на запрос. В случае запроса расписания, запускается алгоритм составления расписания, генерирующий расписание по введенным данным.
6. Если исходный вопрос был из чат-бота, снова запускается ИИ-модуль, отвечающий на него и генерирующий ответ для пользователя.
7. Итоговый ответ отправляется обратно в клиентское приложение, где данные отображаются в соответствующем формате на веб-странице клиента (либо в виде ответа в чат-боте, либо в виде изменения введенных данных).

Архитектура приложения представлена на рис.2:

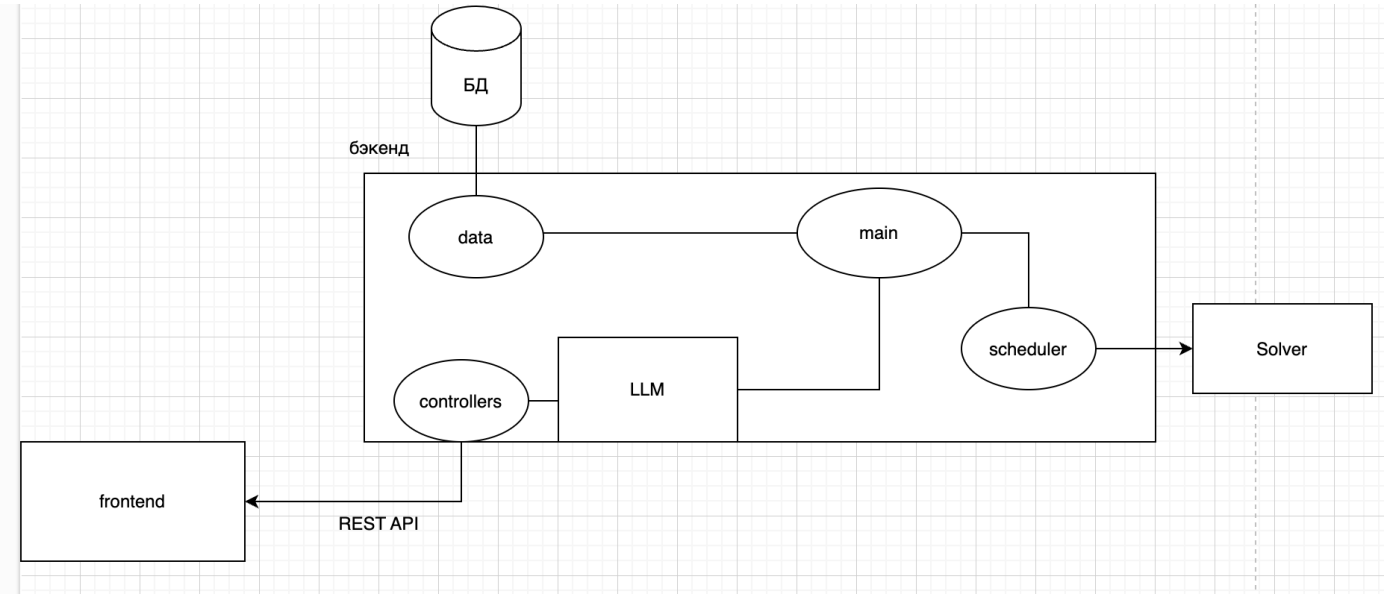

Рисунок 2 – Архитектура веб-сервиса.

2.3. Выбор технологий для реализации и принцип работы приложений

2.3.1. Серверная часть

Серверная часть системы реализована на языке программирования Java 17. Для упрощения создания и настройки приложения используется фреймворк Spring Boot версии 2.6.2, предоставляющий встроенный сервер Tomcat, автоматическую конфигурацию компонентов и инструменты для интеграции с базами данных. Взаимодействие с базой данных реализуется через Spring Data JPA и ORM-библиотеку Hibernate, что минимизирует необходимость ручного написания SQL-запросов и обеспечивает удобный маппинг объектов Java на реляционные структуры.

Архитектура REST API строится на основе Spring Web MVC, обеспечивающего обработку HTTP-запросов и формирование ответов в формате JSON. Аутентификация и авторизация реализованы через JSON Web Token (JWT), интегрированный с Spring Security, что обеспечивает проверку прав доступа к защищенным эндпоинтам.

Для генерации расписаний применяются алгоритмы локального поиска, а именно двухуровневый локальный поиск и поиск с ограничениями.

Функциональное тестирование реализовано на базе фреймворка JUnit 5, оно включает в себя модульные тесты для основных компонентов системы: сервисов, управляющих курсами, преподавателями, группами и временем занятий. Экспериментальное тестирование алгоритма

проводилось вручную для оценки его производительности и устойчивости к росту входных данных и подробнее описано в главе 3.

Сборка проекта выполняется через Maven 3.8 с упаковкой в исполняемый JAR-файл, а развертывание осуществляется на облачном сервере TimeWeb Cloud.

2.3.2. База данных

Для организации данных, необходимых для составления расписаний (фактов, ограничений), а также для хранения информации о пользователях (ролях и университетах), применяется реляционная база данных PostgreSQL. Взаимодействие с базой данных осуществляется посредством Java JDBC фреймворка и SQL запросов, написанных на диалекте PostgreSQL.

2.3.3. Коммуникация между клиентом и сервером

Чтобы клиентская часть приложения могла эффективно обрабатывать полученные данные, выходные данные должны предоставлять формат JSON для возвращаемых данных, чтобы гарантировать унификацию данных и их удобную обработку веб-приложением.

Приложением предоставляются данные в формате HTTP-ответов на запросы, отправляемые веб-приложением со стороны клиента. Ответы состоят из кода ответа, заголовков, текстового содержимого, оформленного в формате JSON, а также логи, фиксирующие деятельность приложения.

Выводы по главе

Во второй главе описаны функциональные характеристики разработанной системы и ее архитектура. Также рассмотрены использованные при разработке технологии с объяснением причин выбора каждого инструмента, приведена схема базы данных и описаны принципы работы выбранных инструментов.

Глава 3. Алгоритмы и методы составления расписаний и развертывание веб-приложения

Задача составления расписаний в университетах — одна из классических задач комбинаторной оптимизации, относящаяся к NP-трудным. Это означает, что точный перебор всех возможных решений занимает экспоненциальное время, что делает его непрактичным для больших данных. Поэтому на протяжении десятилетий предлагались различные методы решения: от математического программирования до эвристических и гибридных подходов. Подробный обзор результатов по данной теме представлен в статье [18]. Также в данной работе авторами было предложено решение с MILP постановкой и учетом обязательных и предпочтительных ограничений. В статье [19] представлено еще одно решение, решающее MILP модель, без учета предпочтений, но с большей размерностью задачи и большим временем работы программы. В статье [20] авторы приводят решение с реализацией генетического алгоритма, учитывающее только часть обязательных ограничений. Обзор существующих решений на русском языке представлено в статье Самсоновой Н.В. «Составление расписания в высшем учебном заведении: математические методы и программные продукты» [21]. В этой главе приведены методы оптимизации, применимые к рассматриваемой задаче. Описаны модель целочисленного линейного программирования и алгоритм локального поиска с запретами, предложенный для решения задачи. а также результаты тестирования.

3.1. Решение задачи составления расписаний

Цель задачи — определить, каким образом назначить учебные занятия множеству академических групп, преподавателей и аудиторий на ограниченное количество временных слотов так, чтобы были выполнены все жёсткие (обязательные) и по возможности мягкие (предпочтительные) ограничения.

Входными данными задачи составления расписания в университете являются:

• Множество курсов C, каждый курс $c \in C$ характеризуется: числом занятий в неделю, длительностью занятия, сложностью, принадлежностью к образовательной программе.

• Множество преподавателей P, каждый преподаватель $p \in P$ имеет ограничения по доступным временным слотам, квалификацию (какие курсы он может вести), и максимальное число занятий в неделю.

• Множество учебных групп G, каждая группа $g \in G$ состоит из заданного количества студентов и привязана к определённой программе обучения.

• Множество аудиторий R, каждая аудитория r ∈ R имеет вместимость, тип (лекционная, компьютерная и т.п.), а также расписание доступности.

• Множество временных слотов T, представляющих дискретные интервалы времени (например, понедельник 09:00–10:30).

Дополнительно заданы бинарные соответствия: курс c может вестись преподавателем p, изучаться группой g, проводиться в аудитории r; преподаватель p, аудитория r и группа g доступны в слоте t.

Цель — распределить занятия по временным слотам и аудиториям, назначив преподавателей, и при этом:

• удовлетворить все жёсткие ограничения: отсутствие пересечений по времени у групп, преподавателей и аудиторий, соответствие доступности и связей;

• минимизировать неудобства, описанные мягкими ограничениями: количество окон в расписании, занятия в нежелательные часы у групп и преподавателей и т.п.

3.1.1. Целочисленное линейное программирование (MILP) и программирование в ограничениях (CP)

MILP-методы формулируют задачу как систему линейных уравнений и неравенств с целочисленными переменными. Это позволяет найти оптимальное расписание (либо расписание с неким отрывом от оптимального), но за счёт высокой вычислительной сложности решение таких моделей при масштабировании задачи становится проблематичным. Тем не менее, MILP остаётся стандартным подходом в исследованиях и коммерческих системах, использующих солверы (Gurobi [23], CPLEX [24], SCIP [25], CBC [26]). В данной работе была построена MILP модель, при решении были использованы солверы CBC и Gurobi. Ниже приведена формулировка задачи в терминах целочисленного линейного программирования.

Переменные:

$x_{\{c,t,r\}} \in \{0,1\}$ — бинарная переменная, равна 1, если курс c проводится в момент времени t в аудитории r ; 0 иначе.

$y_{\{c,t,p\}} \in \{0,1\}$ — бинарная переменная, равна 1, если курс c ведёт преподаватель p в момент времени t; 0 иначе.

$z_{\{g,c,t\}} \in \{0,1\}$ — бинарная переменная, равна 1, если группа g посещает курс c в момент времени t; 0 иначе.

$a_{\{t,p\}} \in \{0,1\}$ — параметр, равен 1, если преподаватель p доступен в момент времени t; 0 иначе.

$b_{\{t,r\}} \in \{0,1\}$ — параметр, равен 1, если аудитория r доступна в момент времени t; 0 иначе.

$s_{\{g,t\}} \in \{0,1\}$ — бинарная переменная, равна 1, если у группы g есть занятие в момент времени t; используется для минимизации "окон" в расписании.

Ограничения:

Каждое занятие должно быть запланировано один раз:

$$\sum_{\{t\}} \sum_{\{r\}} x_{\{c,t,r\}} = 1, \quad \forall c$$

Аудитория не может быть занята двумя курсами одновременно:

$$\sum_{\{c\}} x_{\{c,t,r\}} \leq 1, \quad \forall t, r$$

Преподаватель не может вести два курса одновременно:

$$\sum_{\{c\}} y_{\{c,t,p\}} \leq 1, \quad \forall t, p$$

Группа не может посещать два курса одновременно:

$$\sum_{\{c\}} z_{\{g,c,t\}} \leq 1, \quad \forall g, t$$

Связь между x и y (курс должен быть и в аудитории, и с преподавателем):

$x_{\{c,t,r\}} = y_{\{c,t,p\}}$, если преподаватель p ведёт курс c в аудитории r в момент t

Преподаватель должен быть доступен в указанное время:

$$y\_\{c,t,p\} \leq a\_\{t,p\}, \quad \forall c,t,p$$

Аудитория должна быть доступна в указанное время:

$$x\_\{c,t,r\} \leq b\_\{t,r\}, \quad \forall c,t,r$$

Целевая функция:

Минимизировать суммарное неудобство расписания, которое зависит от предпочтений пользователей, выбранных в системе, и может включать в себя количество окон, занятий в нежелательное время у преподавателей и групп.

3.1.2. Эвристические и метаэвристические методы

Эвристики строят расписание за полиномиальное время, жертвуя оптимальностью ради скорости. К популярным методам относятся:

Жадные алгоритмы — последовательное назначение занятий в первую доступную ячейку. Они быстрые, но могут привести к плохим решениям при сложных ограничениях.

Локальный поиск (Local Search, Tabu Search, Simulated Annealing) — итеративное улучшение расписания путём перестановки элементов.

Генетические алгоритмы — эволюционные методы, использующие механизмы мутаций и скрещивания для поиска хороших решений.

3.1.3. Гибридные подходы

Гибридные методы комбинируют точные и приближённые алгоритмы. Например, MILP для распределения аудиторий, а эвристики — для основной части планирования. Также используют ML для подбора стартовых точек или для тюнинга параметров (например, коэффициентов в целевой функции или в ограничениях).

3.2. Разработанный алгоритм составления расписаний

Основываясь на анализе методов, был выбран двухуровневый гибридный подход, сочетающий поиск с ограничениями (Tabu search) и локальный поиск [22]. Данный подход позволяет учитывать ограничения различного вида и решать задачу за разумное время. Он отлично зарекомендовал себя для решения задачи оптимального назначения сотрудников на обслуживание сетевого оборудования. Алгоритм включает несколько этапов:

Формирование начального решения — жадный алгоритм распределяет занятия по доступным слотам. Получаем все незапланированные занятия (для учителей, групп и студентов), перебираем все незапланированные занятия и назначаем занятие на первый доступный таймслот

без конфликтов (если таких не осталось, выбираем любой - с конфликтом) и обновляем лимиты. В результате получаем расписание со всеми необходимыми парами, но с конфликтами.

Внешний локальный поиск – генерация расписания путем изменения расписания перестановками с удовлетворением всех жестких ограничений.

Внутренний локальный поиск с запретами – улучшение расписания с учетом мягких ограничений, выход из локальных оптимумов, запрет на повторение комбинаций (табу лист хранит состояния объектов, которые встречались за последние n шагов и запрещает возвращаться к ним). В результате комбинации с внешним локальным поиском получаем расписание без конфликтов и с по максимуму удовлетворенными мягкими ограничениями. В случае если алгоритм не смог выйти на расписание без конфликтов, пользователю возвращается конфликтное расписание с соответствующим сообщением и указанием мест, где не удалось решить конфликты.

Выбранный подход обеспечивает баланс между скоростью работы и качеством расписания, а также помогает построить расписание, удовлетворяющее всем ограничениям и учитывающее предпочтения и необязательные условия.

3.3 Тестирование алгоритма

Для оценки эффективности работы алгоритма генерации расписаний были проведены функциональные и производительные тесты. Цель тестирования заключалась в проверке способности системы обрабатывать входные данные различного масштаба в разумное время, обеспечивая при этом удовлетворение всех жёстких ограничений (таких как занятость преподавателей, вместимость аудиторий, доступность групп и пр.).

Алгоритм был протестирован на двух наборах входных данных:

Тест 1 (сгенерированный набор данных):

- 30 преподавателей;
- 40 учебных аудиторий;
- 2 образовательные программы;
- 10 учебных групп (по 30 студентов);
- 30 курсов.

Время выполнения: менее **3 минут**.

Тест 2. Набор данных МОУ СОШ №33 города Ярославля, адаптированный под модель университета:

- 60 преподавателей;

- 40 учебных аудиторий;

- 4 образовательные программы;

- 20 учебных групп (по 30 студентов);

- 30 курсов.

Время выполнения: до **5 минут.**

Тест 3. Набор данных математического факультета ЯрГУ им.Демидова:

- 100 преподавателей;

- 30 учебных аудиторий;

- 5 образовательных программ;

- 26 учебных групп (по 20-30 студентов);

- 90 курсов.

Время выполнения: от **15** до **20 минут.**

Во всех случаях алгоритм успешно завершал построение расписания, обеспечивая соблюдение жёстких ограничений и минимизацию количества нарушений мягких ограничений (например, предпочтений по времени или аудиториям).

Временные показатели указывают на хорошую масштабируемость алгоритма при увеличении входных данных. Это позволяет использовать его в реальных условиях, в том числе для крупных образовательных учреждений.

Также в процессе тестирования производилась верификация корректности построенных расписаний, что подтверждало отсутствие конфликтов и логических ошибок.

3.4 Развертывание сервиса

Сервер работает на виртуальной машине, находящейся на облачной платформе TimeWeb Cloud. Виртуальный хост обладает гарантированными ресурсами: 1 x 3.3 ГГц CPU, 2 ГБ RAM и 30

ГБ NVMe. Такой набор характеристик обеспечивает достаточную производительность для данной программы. Операционная система на сервере — Ubuntu 22.04.

ИИ-модуль, реализующий интеллектуальную обработку пользовательских запросов, развернут на том же сервере, но прослушивает отдельный порт — 5001. Он реализован на языке Python с использованием фреймворка Flask. Вся логика модуля построена на агентной архитектуре: используются агенты парсинга, валидации и генерации диалога. Они взаимодействуют друг с другом через класс-оркестратор, обеспечивая пошаговую обработку естественного языка и генерацию осмысленного ответа. Модуль обращается к внешнему Yandex GPT API, что позволяет использовать языковую модель без локального обучения. Flask-приложение принимает HTTP-запросы от основного сервиса и возвращает готовые ответы в формате JSON. Такой подход позволяет гибко масштабировать ИИ-компонент без изменения архитектуры всего сервиса.

Для мониторинга используется встроенная система TimeWeb Cloud, позволяющая отслеживать загрузку процессора, оперативной памяти, дисковую активность и сетевой трафик в реальном времени.

База данных PostgreSQL также размещена на облачном сервере платформы TimeWeb Cloud. Настроено автоматическое резервное копирование для предотвращения потери данных. Доступ к базе осуществляется через ORM Hibernate, обеспечивающий удобную работу с данными.

Выводы по главе

В данной главе представлен анализ задачи составления расписаний, рассмотрены методы ее решения, описано использующееся в программе решение, а также приведены вычислительные результаты работы алгоритма. Кроме того, описан процесс развертывания сервиса с указанием всех деталей.

Заключение

В данной выпускной квалификационной работе была рассмотрена задача автоматизированного составления расписаний для высших учебных заведений, а также создана серверная часть для веб-приложения, составляющего данные расписания. В ходе исследования были проанализированы существующие методы решения данной задачи, проведён сравнительный анализ различных алгоритмических подходов, а также предложено и реализовано собственное решение на основе двухуровневого поиска.

В первой главе была рассмотрена предметная область, определены ключевые проблемы традиционного составления расписаний и проанализированы существующие программные решения. Было выявлено, что большинство существующих систем либо обладают высокой вычислительной сложностью, либо требуют слишком индивидуализированную настройку или жёстких входных данных, что затрудняет их использование на практике.

Во второй главе представлено проектирование системы, включая архитектуру серверной части, базу данных и взаимодействие с клиентским приложением. Описаны ключевые технологические решения.

Третья глава посвящена алгоритмической части работы и практической его развертке. Был проведён обзор классических методов составления расписаний, рассмотрены аспекты развертывания серверной части и базы данных, а также интеграция системы с чат-ботом.

В целом, проделанная работа позволила создать универсальное, эффективное и гибкое решение для автоматизации составления университетских расписаний, подходящее небольшим университетам, желающих автоматизировать процессы. Разработанная система минимизирует ручной труд, снижает вероятность конфликтов и упрощает процесс планирования. В дальнейшем возможны улучшения алгоритма за счёт более сложных методов и дополнительных механизмов адаптации к специфическим требованиям учебных заведений.

СПИСОК ИСТОЧНИКОВ

1. ГОСТ 19.101-77 Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.103-77 Обозначения программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.104-78 Основные надписи. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.105-78 Общие требования к программным документам. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.603-78 Общие правила внесения изменений. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. Системные требования браузера Google Chrome [Электронный ресурс] / Google. Режим доступа: https://support.google.com/chrome/a/answer/7100626?hl=ru, свободный (Дата обращения: 01.10.2024).
10. PostgreSQL [Электронный ресурс]. URL: https://www.postgresql.org, свободный (Дата обращения: 01.10.2024).
11. Spring Boot: [Электронный ресурс]. URL: https://spring.io/projects/spring-boot. (Дата обращения: 01.10.2024).
12. IntelliJ IDEA Edu: [Электронный ресурс]. URL: https://www.jetbrains.com/ru-ru/idea/. (Дата обращения: 01.10.2024).
13. Maven: [Электронный ресурс]. URL: https://maven.apache.org. (Дата обращения: 01.10.2024).
14. Ректор-ВУЗ: [Электронный ресурс]. URL: https://rector.spb.ru/raspisanie-vuz-4u.php. (Дата обращения: 01.10.2024).
15. Экспресс-Расписание ВУЗ: [Электронный ресурс]. URL: https://pbprog.ru/docs/raspisv/. (Дата обращения: 01.10.2024).
16. 1С:Университет: [Электронный ресурс]. URL: https://solutions.1c.ru/catalog/university. (Дата обращения: 01.10.2024).

17. Magellan: [Электронный ресурс]. URL: https://magellanius.ru/modules/schedule/. (Дата обращения: 01.10.2024).
18. Mo Chen, Frank Werner, Mohammad Shokouhifar “Mathematical Modeling and Exact Optimizing of University Course Scheduling Considering Preferences of Professors” – 2023. – DOI: 10.20944/preprints202303.0139.v1
19. Camilo Torres-Ovalle, Jairo R. Montoya-Torres, Carlos Quintero-Araujo “University Course Scheduling and Classroom Assignment”. – 2014. – DOI: 10.11144/JavEriana.iYU18-1.phaa
20. Mohammed Aldasht, Mahmoud Alsaheb, Safa Adi, Mohammad Abu Qbeitah “University Course Scheduling Using Evolutionary Algorithms”. – 2009. – DOI: 10.1109/ICCGI.2009.15
21. Самсонова Н.В. Симонов А.Б. «Составление расписания в высшем учебном заведении: математические методы и программные продукты». – 2018. – DOI: https://doi.org/10.26425/2658-3445-2018-1-60-69
22. Давыдов И., Габдуллина А., Шевцова М., Архипов Д. “Tabu Search for a Service Zone Clustering Problem” // Lecture Notes in Computer Science. – 2024. – Т. 14766. – С. 75–88. – DOI: 10.1007/978-3-031-62792-7_6
23. Gurobi: [Электронный ресурс]. URL: https://www.gurobi.com. (Дата обращения: 01.10.2024).
24. CPLEX: [Электронный ресурс]. URL: https://www.ibm.com. (Дата обращения: 01.10.2024).
25. SCIP: [Электронный ресурс]. URL: https://www.scipopt.org. (Дата обращения: 01.10.2024).
26. CBC: [Электронный ресурс]. URL: https://www.coin-or.org. (Дата обращения: 01.10.2024).