Содержание

Введение

В условиях современных образовательных учреждений, характеризующихся высокой концентрацией обучающихся и преподавательского состава, возникает острая проблема организации учебного процесса, связанная с ручным формированием расписания занятий. Большое количество групп, разнообразие предметов, ограниченное количество аудиторий и временные ограничения создают сложности при составлении оптимального расписания. Это приводит к конфликтам в распределении ресурсов, перегрузке преподавателей и студентов, а также снижению эффективности образовательного процесса. Данная ситуация требует автоматизации для минимизации ошибок и повышения удобства использования расписания всеми участниками образовательного процесса. Настоящий дипломный проект посвящен проектированию и разработке информационной системы автоматической генерации учебного расписания, призванной решить эту проблему, обеспечив удобный, гибкий и эффективный способ формирования и управления расписанием для образовательных учреждений любого уровня.

Актуальность темы обусловлена необходимостью поиска инновационных решений для оптимизации процесса организации учебного расписания в условиях ограниченных ресурсов и времени. Проект также предоставляет возможность освоить практические навыки проектирования и реализации реляционных баз данных.

Целью данного проекта является проектирование и разработка информационной системы автоматической генерации учебного расписания для образовательных учреждений, нацеленной на решение проблемы ручного составления расписания и связанных с этим конфликтов. Система должна обеспечить удобный интерфейс для пользователей, эффективное управление расписанием, учет ограничений и возможность экспорта данных для их дальнейшего использования.

Для достижения поставленной цели необходимо решить следующие задачи:

– провести анализ существующей системы организации учебного расписания в образовательных учреждениях, идентифицировать основные проблемы и ограничения;

– разработать функциональные требования к информационной системе, определив необходимый набор функций и возможностей;

– спроектировать базу данных, определив структуру таблиц и связи между ними;

– разработать интерфейс пользователя, обеспечив интуитивно понятное управление системой;

– реализовать функционал автоматической генерации расписания, включая учет ограничений и возможность экспорта данных через API;

– провести тестирование разработанной системы и оценить ее производительность и надежность;

– подготовить техническую документацию, включающую описание архитектуры системы, базы данных и инструкции по использованию.

1 Проектирование информационной системы

1. 1 Анализ предметной области и построение бизнес-процессов

Образовательные учреждения представляют собой комплекс зданий, включающих в себя учебные корпуса, административные помещения и дополнительные объекты, такие как лаборатории, спортивные залы или общежития. Количество студентов очного и дневного отделения варьируется в зависимости от размера учреждения, в среднем составляя 2500 человек, при этом количество сотрудников, включая преподавателей и административный персонал, достигает 250 человек. Организация учебного процесса требует эффективного управления расписанием занятий, учета аудиторий и распределения времени между группами для обеспечения бесперебойной работы всех подразделений. В современном образовательном процессе планирование учебного времени играет ключевую роль, так как многие учебные заведения сталкиваются с трудностями при составлении расписания, учитывая множество факторов, таких как занятость преподавателей, доступность аудиторий и учебных групп. Автоматизация процесса генерации расписания позволяет оптимизировать эту задачу, экономя время административного персонала и минимизируя вероятность ошибок, что особенно важно для обеспечения высокого уровня организации учебного процесса.

Составление учебного расписания — это процесс планирования и организации занятий в учебных заведениях. В этой предметной области необходимо учитывать разнообразие учебных дисциплин, занятость преподавателей, расписание аудиторий, количество учебных групп, а также специфику требований к учебному процессу. Создание расписания также включает возможность настройки дополнительных параметров, таких как приоритетные временные слоты или особые условия для отдельных групп или преподавателей, чтобы максимально удовлетворить потребности всех участников образовательного процесса.

Интерфейс для просмотра и управления данными в формате JSON является дополнительным сервисом, который может быть интегрирован в информационную систему для автоматической генерации учебного расписания. API-интерфейс — это инструмент, предоставляющий возможность взаимодействия с системой через интернет. В контексте данной предметной области, API может использоваться для экспорта данных о расписании, группах, преподавателях и аудиториях, что позволяет легко интегрировать систему с другими платформами или использовать данные для дальнейшей обработки.

В текущем проекте реализовано веб-приложение, ориентированное на автоматическую генерацию учебного расписания и управление данными об учебном процессе.

Задача состоит в создании удобного пользовательского интерфейса, который обеспечит легкую навигацию по системе. Каждый пользователь должен иметь возможность быстро сформировать расписание, просмотреть его или внести необходимые изменения, что позволит сэкономить время. Больше не нужно будет вручную составлять расписания, искать свободные аудитории или согласовывать занятость преподавателей, сталкиваться с ошибками при планировании и тратить ресурсы на корректировку конфликтов в расписании.

В рамках работы с пользователем веб-приложение предоставляет возможность управления учебным расписанием. Пользователь может сгенерировать расписание, указав необходимые параметры в форме: список кабинетов, группы, преподаватели, максимальное количество пар в день, закрепление преподавателя за определенным кабинетом, а также создание предмета для группы и его привязку к конкретному преподавателю. После заполнения формы система автоматически генерирует расписание на основе введенных данных. Пользователь может просмотреть результат на отдельной странице, внести корректировки или экспортировать данные в формате JSON через API.

При проектировании бизнес-процессов для информационной системы автоматической генерации учебного расписания необходимо учитывать следующие основные этапы:

– Создание запроса на генерацию расписания: этот этап включает процесс заполнения формы пользователем через веб-приложение. Пользователь указывает параметры: кабинеты, группы, преподаватели, максимальное количество пар в день, закрепление преподавателя за кабинетом, а также создание предмета для группы и его привязку к преподавателю.

– Подтверждение данных: после отправки формы система автоматически проверяет корректность введенных данных. В процессе проверки система анализирует наличие конфликтов, таких как занятость преподавателя, доступность аудиторий или превышение максимального количества пар в день. При обнаружении несоответствий или конфликтов система уведомляет пользователя о необходимости внести корректировки.

– Генерация расписания: после подтверждения данных система автоматически формирует расписание на основе заданных параметров. Если расписание уже существует, пользователь может обновить его, сохранив предыдущие настройки.

– Просмотр и редактирование расписания: пользователь получает доступ к сгенерированному расписанию на отдельной странице. При необходимости он может внести изменения, добавить или удалить предметы, а также экспортировать данные в формате JSON через API.

– Обратная связь и поддержка: если у пользователя возникают вопросы или проблемы при работе с системой, он может обратиться через страницу обратной связи. Администратор системы рассматривает обращения и предоставляет ответы, что способствует улучшению функционала и повышению удовлетворенности пользователей.

Бизнес-процессы по генерации учебного расписания представлены на рисунке 1.1.

Бизнес-процессы	Описание	Входные данные	Выходные данные	Ответственный	Бизнес-правила
Создание запроса на генерацию расписания	Пользователь заполняет форму через веб-приложение, указывая параметры: кабинеты, группы, преподаватели, максимальное количество пар в день, закрепление преподавателя за кабинетом и привязку предмета.	- Кабинеты - Группы - Преподаватели - Максимальное количество пар в день - Закрепление преподавателя за кабинетом - Предметы и их привязка к группам и преподавателям	- Запрос на генерацию расписания (с указанием всех параметров)	Пользователь	Все поля должны быть корректно заполнены; отсутствие обязательных данных должно вызывать ошибку.
Подтверждение данных	Система проверяет корректность введенных данных, анализирует наличие конфликтов (занятость преподавателей, доступность аудиторий, превышение нагрузки). При обнаружении конфликтов пользователь уведомляется о необходимости внести корректировки.	- Запрос на генерацию расписания	-Подтверждение корректности данных - Уведомления о конфликтах или несоответствиях	Система	Должны быть проверены все ограничения (например, занятость преподавателей, доступность аудиторий, максимальная нагрузка).
Генерация расписания	После подтверждения данных система автоматически формирует расписание на основе заданных параметров. Если расписание уже существует, пользователь может обновить его, сохранив предыдущие настройки.	- Подтвержденный запрос на генерацию расписания	- Сгенерированное расписание (включая даты, время занятий, аудитории, преподаватели, предметы)	Система	Расписание должно соответствовать всем указанным ограничениям и требованиям.
Просмотр и редактирование расписания	Пользователь получает доступ к сгенерированному расписанию на отдельной странице. При необходимости он может вносить изменения, добавлять или удалять предметы, а также экспортировать данные в формате JSON через API.	- Сгенерированное расписание	- Измененное расписание (если пользователь вносит правки) - Экспорт данных в формате JSON	Пользователь	Любые изменения должны сохраняться в системе, и экспорт должен быть доступен в требуемом формате.
Обратная связь и поддержка	Если у пользователя возникают вопросы или проблемы при работе с системой, он может обратиться через страницу обратной связи. Администратор системы рассматривает обращения и предоставляет ответы.	- Сообщение обратной связи от пользователя	- Ответ администратора на обращение	Администратор	Обращения должны обрабатываться в разумные сроки, и пользователь должен получить четкий ответ.

Рисунок 1.1 – Таблица бизнес-процессов

1. 2 Проектирование диаграммы прецедентов и диаграммы деятельности

В ходе разработки информационной системы были спроектированы и разработаны диаграмма прецедентов и диаграмма деятельности.

Диаграмма прецедентов – это диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Акторы представляют собой внешние сущности, которые взаимодействуют с системой. Это могут быть пользователи, другие системы или внешние устройства. Акторы обозначаются на диаграмме в виде человечков или простых фигур, и их роли в системе могут быть различными.

Прецеденты – это специфические действия или функции, которые система выполняет в ответ на запрос акторов. Прецеденты описывают поведение системы с точки зрения пользователя, определяя, что система должна делать, но не как она должна это делать. На диаграмме прецеденты изображаются овальными фигурами, в которых указывается название прецедента.

Диаграмма прецедентов по генерации учебного расписания представлена на рисунке 1.2.

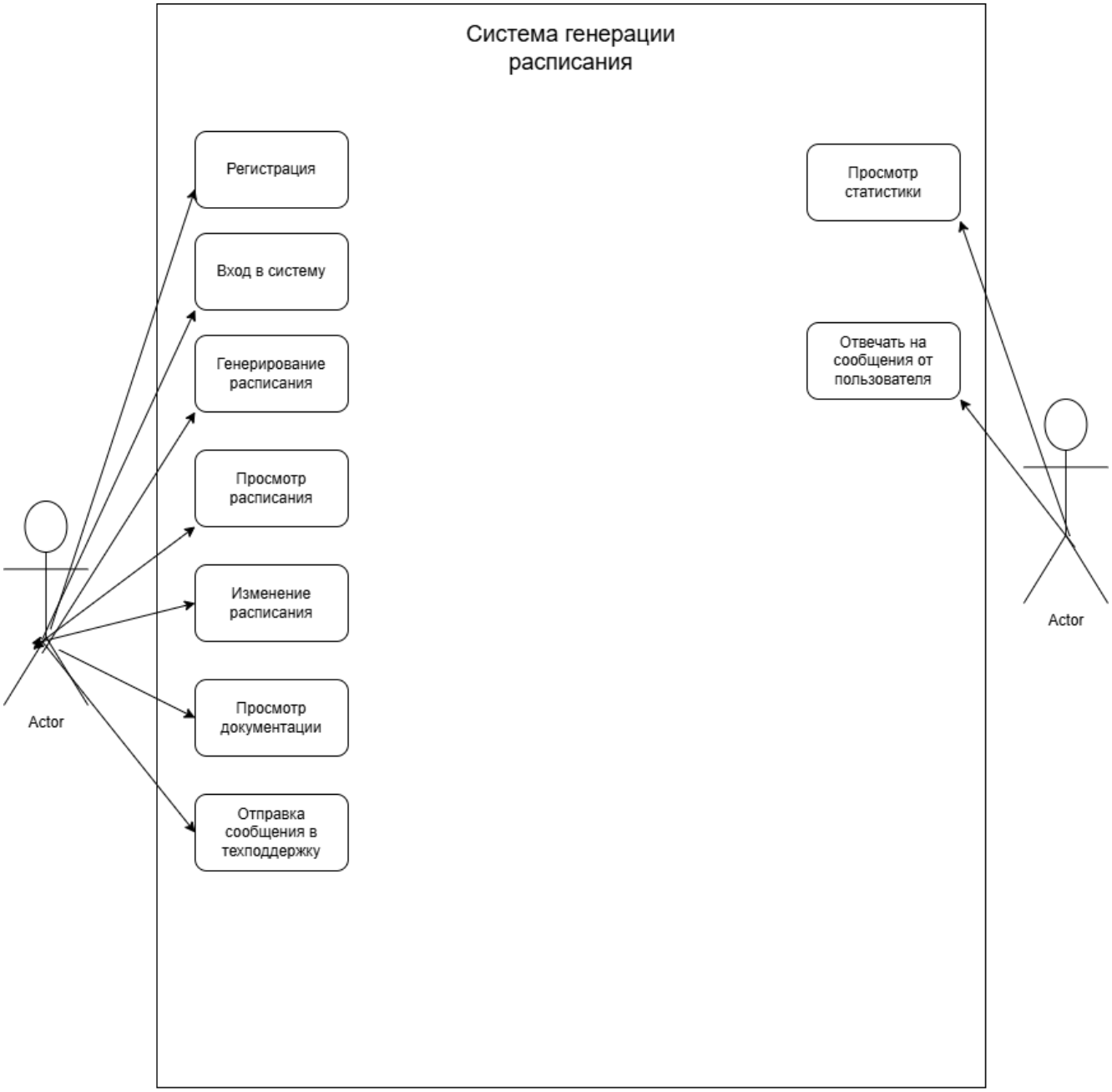

Рисунок 1.2 – Диаграмма прецедентов

Диаграмма деятельности – это диаграмма для демонстрации рабочего процесса некоторой деятельности, основанной на поэтапных действиях и действиях с поддержкой выбора и параллелизма.

Спроектированная диаграмма содержит 2 дорожки: «Пользователь» и «Администратор». Также в ней присутствуют такие элементы, как начальное

состояние, активное состояние, принятие решений и конечное состояние. Чередование состояний сопровождается переходом (стрелкой).

Диаграмма деятельности по генерации учебного расписания представлена на рисунке 1.3.

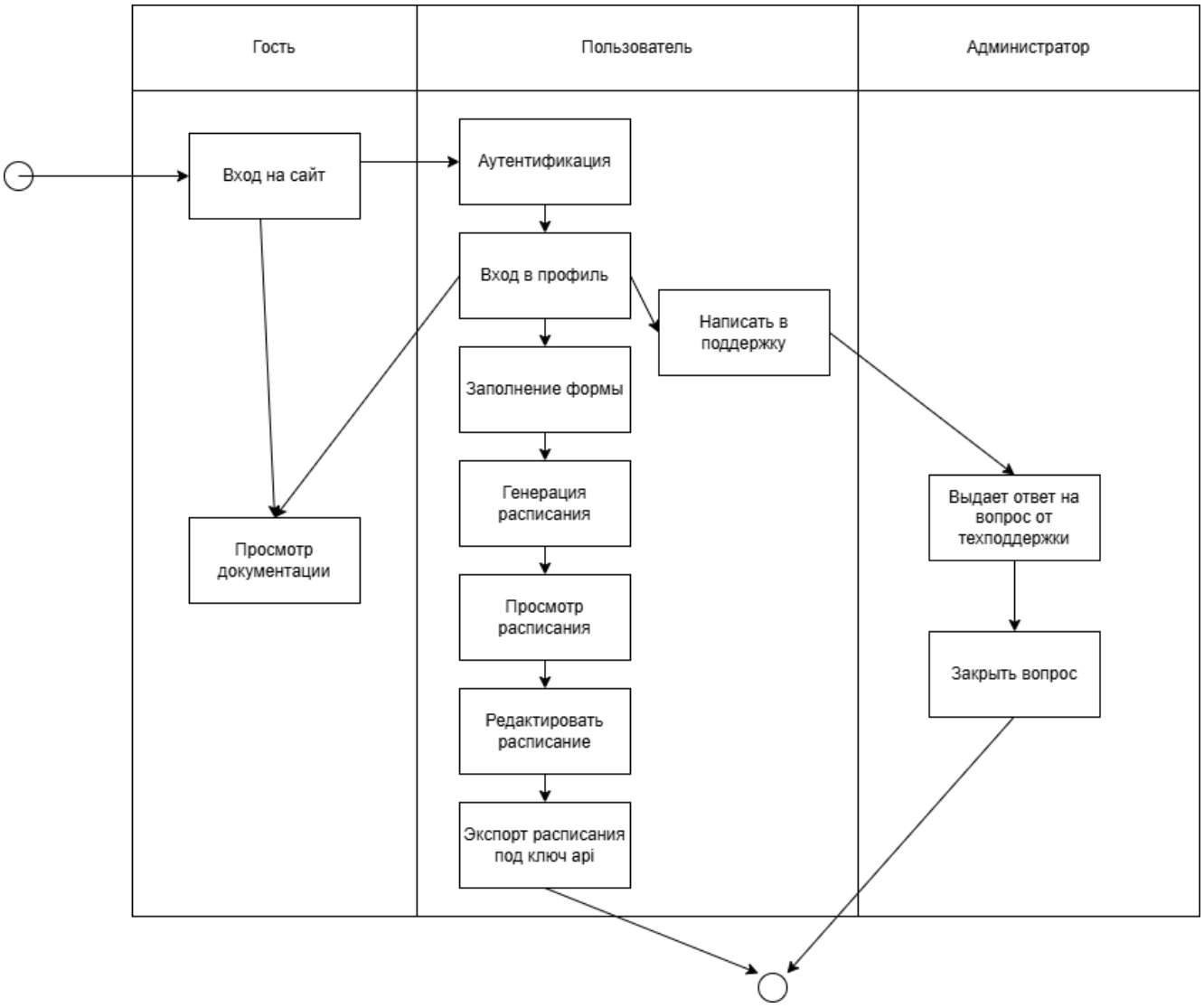

Рисунок 1.3 – Диаграмма деятельности по генерации учебного расписания

Алгоритм — это точное, пошаговое описание процесса решения задачи. Это набор инструкций, понятных исполнителю (человеку или компьютеру), которые, будучи выполнены в определенном порядке, приводят к желаемому результату.

Ключевые характеристики эффективного алгоритма:

– Алгоритм должен завершаться за конечное время; бесконечный цикл или рекурсия без условия остановки делают алгоритм бесполезным; он должен гарантированно завершить работу, даже в худшем случае;

– Каждая инструкция алгоритма должна быть однозначно определена и не допускать неоднозначного толкования; исполнитель должен точно понимать, какое действие нужно выполнить на каждом шаге; нет места для предположений или произвольных решений;

– Алгоритм может принимать входные данные, которые определяют конкретную задачу, которую нужно решить; эти данные могут быть заданы заранее или вводиться в процессе выполнения алгоритма;

– Алгоритм должен производить выходные данные — результат решения задачи; эти данные могут быть выведены на экран, сохранены в файл или использоваться в дальнейших вычислениях;

– Хотя не всегда является строгим требованием, эффективный алгоритм решает задачу, используя оптимальные или близкие к оптимальным ресурсам (время и память).

Алгоритм создания расписания представлена на рисунке 1.4.

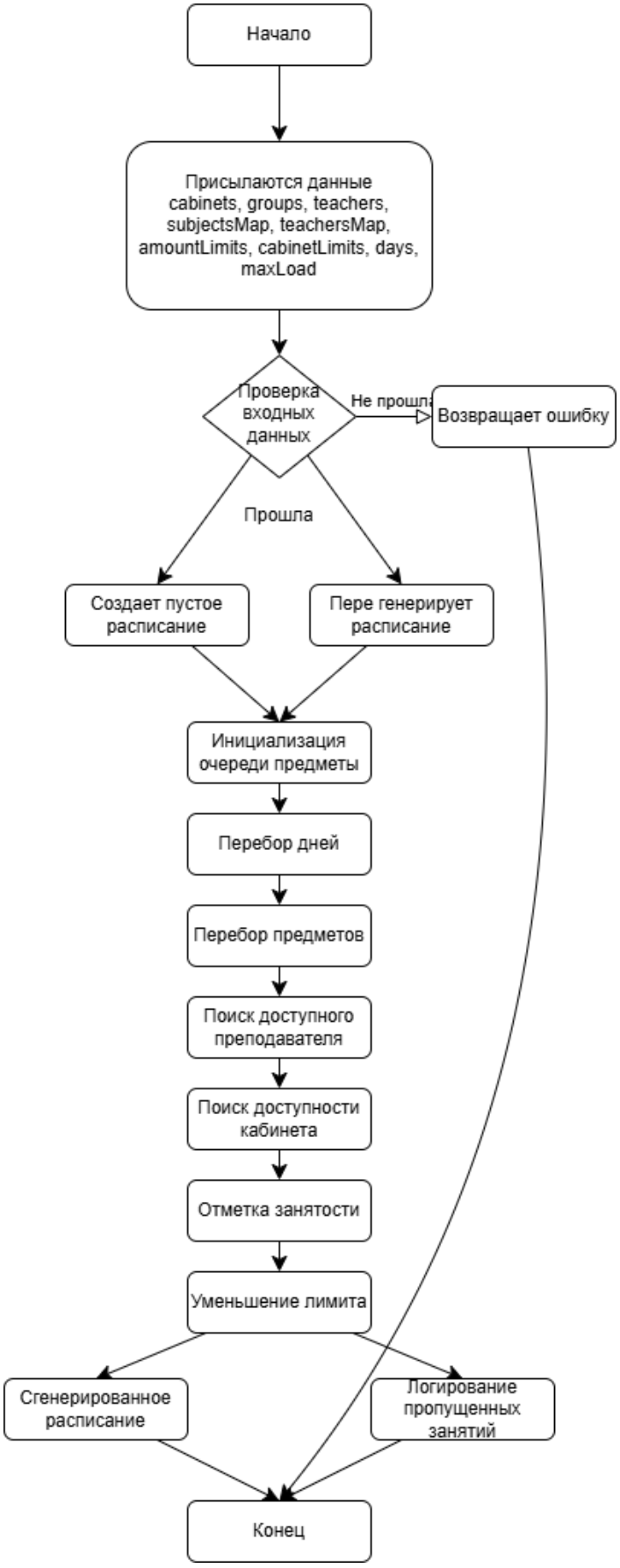

Рисунок 1.4 – Алгоритм создания расписания

1. 3 Проектирование диаграммы «сущность-связь» и словаря данных

Исходя из анализа предметной области, построения бизнес-процессов и диаграммы прецедентов построим диаграмму «сущность-связь».

Диаграмма «сущность-связь» представлена на рисунке 1.5.

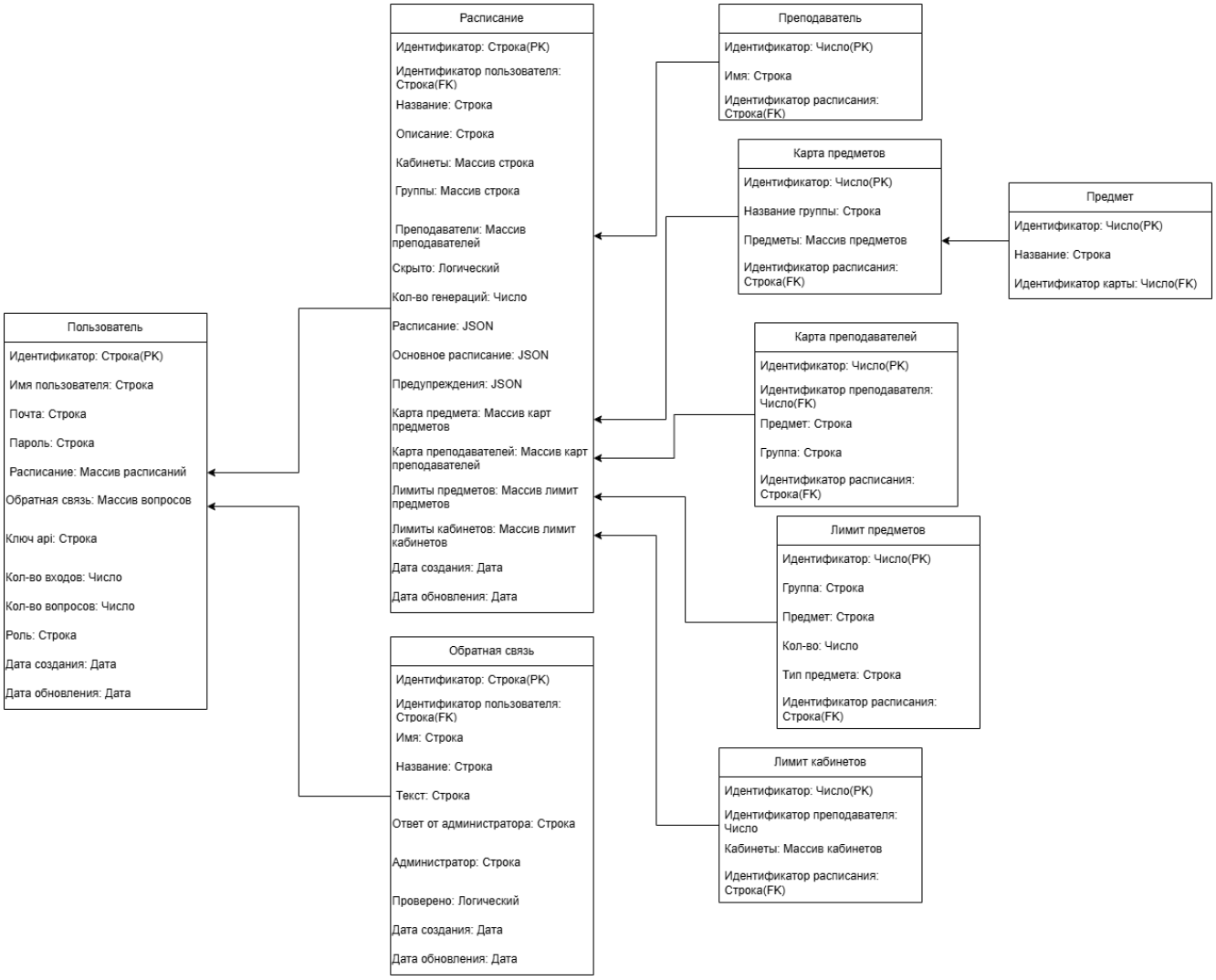

Рисунок 1.5 – Диаграмма «сущность-связь»

В ходе проектирования также был создан словарь данных. Он представляет собой подсистему банка данных, предназначенную для централизованного хранения информации о структурах данных, взаимосвязях файлов базы данных

друг с другом, типах данных и форматах их представления, принадлежности данных пользователям, кодах защиты и разграничения доступа и т. п.

Атрибуты сущности «Пользователь» описаны в таблице 1.1.

Таблица 1.1 – Пользователь

Имя поля	Описание поля	Тип данных	Тип ключа
id	Идентификатор пользователя	STRING	PK
name	Имя пользователя	STRING	
email	Почта	STRING	
password	Пароль	STRING	
schedule	Массив расписаний	Schedule[]	
feedback	Массив обратной связи	Feedback[]	
api_key	Ключ api	STRING	
role	Роль пользователя	STRING	
visits	Количество входов	INT	
feedback_count	Количество вопросов	INT	
created_at	Дата создания	DATE	
updated_at	Дата обновления	DATE	

Атрибуты сущности «Обратная связь» описаны в таблице 1.2.

Таблица 1.2 – Обратная связь

Имя поля	Описание поля	Тип данных	Тип ключа
id	Идентификатор обратной связи	STRING	PK
id_user	Идентификатор пользователя	STRING	FK
title	Имя обратной связи	STRING	
text	Описание обратной связи	STRING	
feedback_admin	Ответ обратной связи	STRING	
admin	От кого ответ	STRING	
isCheck	Проверено ли сообщение	BOOLEAN	
created_at	Дата создания	DATE	
updated_at	Дата обновления	DATE	

Атрибуты сущности «Расписание» описаны в таблице 1.3

Таблица 1.3 - Расписание

Имя поля	Описание поля	Тип данных	Тип ключа
id	Идентификатор расписания	STRING	PK
id_user	Идентификатор пользователя	STRING	FK
title	Имя расписания	STRING	
description	Описание расписания	STRING	
cabinets	Массив кабинетов	STRING[]	
groups	Массив групп	STRING[]	
teachers	Массив преподавателей	Teacher[]	
mapSubjects	Карта предмета	MapSubject[]	
mapTeacher	Карта преподавателя	MapTeacher[]	
amountLimits	Ограничение предмета	AmountLimits[]	
cabinetLimits	Лимит кабинетов	CabinetLimits[]	
isShow	Скрыто ли расписание	BOOLEAN	
schedule	Готовое расписание	JSON	
scheduleMain	Постоянное расписание	JSON	
failed	Предупреждения	JSON	
Schedule_count	Количество генераций	INT	
created_at	Дата создания	DATE	
updated_at	Дата обновления	DATE	

Атрибуты сущности «Карта предмета» описаны в таблице 1.4.

Таблица 1.4 – Карта предмета

Имя поля	Описание поля	Тип данных	Тип ключа
id	Идентификатор карты предмета	INT	PK
scheduleId	Идентификатор расписания	STRING	FK
name_group	Имя группы	STRING	
subjects	Массив предметов	Subject[]	

Атрибуты сущности «Предмет» описаны в таблице 1.5.

Таблица 1.5 - Предмет

Имя поля	Описание поля	Тип данных	Тип ключа
id	Идентификатор предмета	INT	PK
mapSubject_id	Идентификатор карты предмета	STRING	FK
name	Название предмета	STRING	

Атрибуты сущности «Карта преподавателя» описаны в таблице 1.6.

Таблица 1.6 – Карта преподавателя

Имя поля	Описание поля	Тип данных	Тип ключа
id	Идентификатор карты преподавателя	INT	PK
scheduleId	Идентификатор расписания	STRING	FK
tid	Идентификатор преподавателя	INT	
subject	Предмет, который будет ввести преподаватель	STRING	
group	Группа, которую будет ввести преподаватель	STRING	

Атрибуты сущности «Преподаватель» описаны в таблице 1.7.

Таблица 1.7 - Преподаватель

Имя поля	Описание поля	Тип данных	Тип ключа
tid	Идентификатор преподавателя	INT	
name	Имя преподавателя	STRING	
scheduleId	Идентификатор расписания	STRING	FK

Атрибуты сущности «Лимит предметов» описаны в таблице 1.8.

Таблица 1.8 – Лимит предметов

Имя поля	Описание поля	Тип данных	Тип ключа
id	Идентификатор ограничение количество предметов	INT	PK
scheduleId	Идентификатор расписания	STRING	FK
group	Название группы	STRING	
subject	Название предмета	STRING	
count	Количество часов для предмета	INT	
type_subject	Тип предмета	STRING	

Атрибуты сущности «Лимиты кабинетов» описаны в таблице 1.9.

Таблица 1.9 – Лимиты кабинетов

Имя поля	Описание поля	Тип данных	Тип ключа
id	Идентификатор лимитов	INT	PK
id_schedule	Идентификатор расписания	STRING	FK
tid	Идентификатор преподавателя	INT	
cabinets	Массив кабинетов	STRING[]	

1. 4 Проектирование интерфейса информационной системы

В контексте создания информационной системы для автоматической генерации учебного расписания, контент сайта играет ключевую роль. Контент включает в себя текстовые материалы о функционале системы, инструкции по использованию, информацию о возможностях генерации расписания, а также визуальное представление данных в виде таблиц, графиков и интерактивных элементов. Контент является основным информационным наполнением системы и предназначен для передачи информации, взаимодействия с пользователями и достижения целей платформы.

Многостраничные сайты в данном случае обеспечивают возможность структурированного представления информации о функциях системы. Каждая страница может содержать информацию о различных аспектах работы: например, страница "Профиль" предоставляет доступ к настройкам и параметрам создания расписания, страница "Документация" содержит инструкции по интеграции с внешними системами, а страница "Обратная связь" позволяет пользователям оставлять запросы или предложения. Это позволяет пользователям легко находить необходимую информацию и эффективно работать с системой.

При разработке данной информационной системы, визуальное оформление веб-ресурса также является важным аспектом. Макет сайта представляет собой графическое отображение дизайна и структуры веб-страниц, определяющее расположение элементов на сайте и его общий визуальный облик. Макет является первоначальным этапом в процессе разработки веб-сайта, где дизайнер или разработчик создает визуальное представление интерфейса.

При выборе цветовой палитры для интерфейса информационной системы важно учитывать не только эстетические аспекты, но и психологические воздействия на пользователей. Черный цвет ассоциируется с надежностью, профессионализмом и спокойствием, что подчеркивает стабильность и эффективность системы. Белый цвет добавляет элементы чистоты и минимализма,

обеспечивая легкость восприятия интерфейса. Эта комбинация создает гармоничный и привлекательный визуальный облик, который способствует удобству использования системы и улучшает пользовательский опыт

Цветовая палитра - основные цвета #09090B (черный) и #FFFFFF (белый), которые представлены на рисунке 1.6.

Рисунок 1.6 – Основные цвета

Черный часто используется для выделения кнопок и активных элементов, так как он привлекает внимание пользователей и ассоциируется с положительными действиями. Это помогает сделать интерактивные элементы более заметными и поддерживать пользовательскую активность.

Использование черного цвета на главной странице представлено на рисунке 1.7.

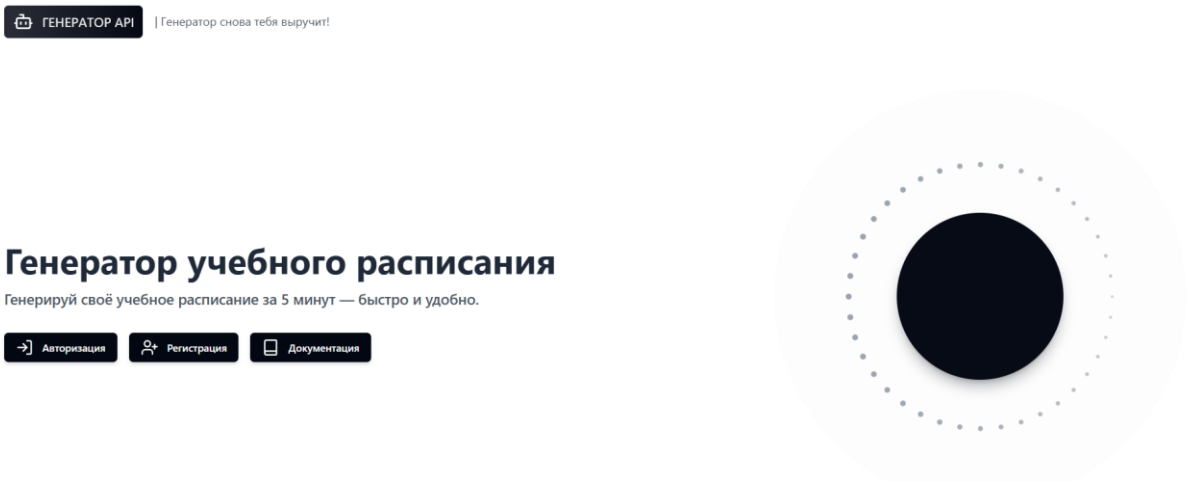

Рисунок 1.7 – Главная страница

Использование черного цвета на странице «Страница авторизации» представлено на рисунке 1.8.

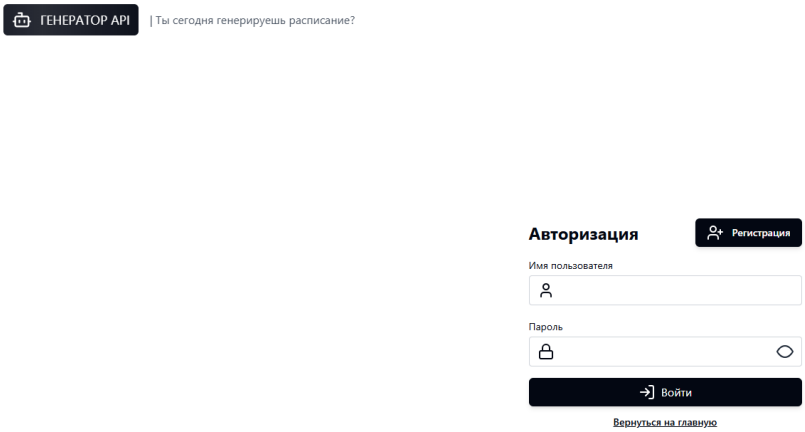

Рисунок 1.8 – Страница авторизации

Использование черного цвета на странице «Страница регистрации» представлено на рисунке 1.9.

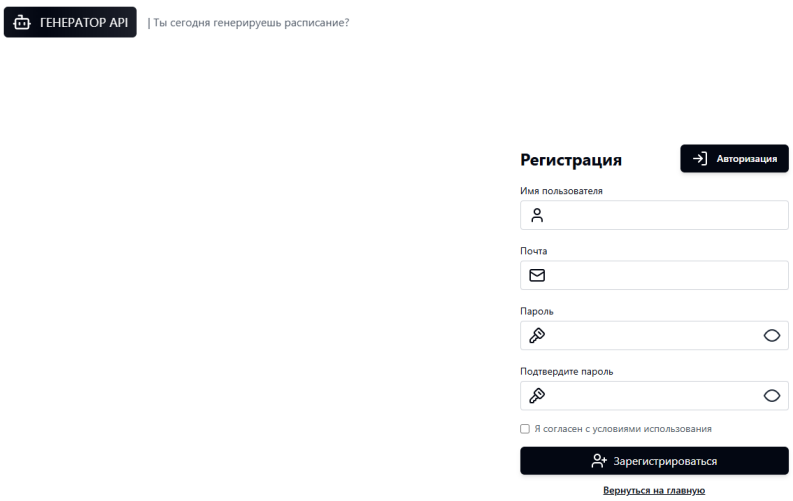

Рисунок 1.9 – Страница регистрации

Белый цвет, напротив, обычно применяется для фона, текста и активных элементов интерфейса, так как он создает ощущение чистоты, простора и минимализма. Это делает контент более легким для восприятия и придает интерфейсу современный и аккуратный вид.

Использование белого цвета в информационной системе представлено на рисунке 1.10.

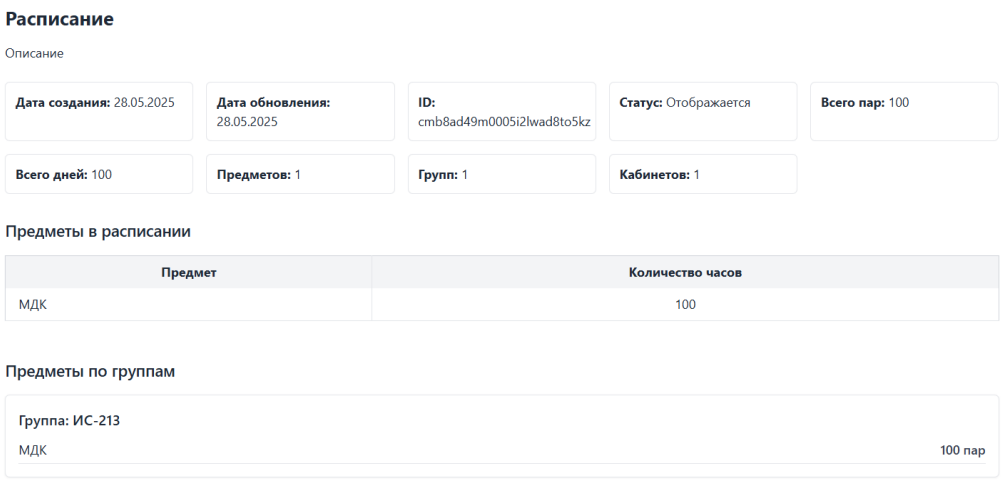

Рисунок 1.10 – Белый цвет в информационной системе

Логотип является ключевым элементом. Черный цвет в логотипе является ключевым элементом, который подчеркивает надежность, профессионализм и технологичность системы. Этот цвет создает ощущение доверия и стабильности, что особенно важно для информационной системы, ориентированной на автоматизацию учебного процесса.

Логотип представлен на рисунке 1.11.

Рисунок 1.11 – Логотип

1. 5 Выбор программных и технических средств разработки информационной системы

При разработке информационной системы было необходимо изучить и применить обширный набор технологий, таких как Typescript, NodeJs, NestJs, NextJs, Prisma, Tailwind и Jotai, VS Code. Эти инструменты совместно обеспечивают мощную архитектуру для создания динамичных и интерактивных веб-приложений, которые соответствуют современным требованиям пользователей. TypeScript стал основным языком программирования в проекте. Этот язык зарекомендовал себя как мощный инструмент для разработки масштабируемых и надежных приложений благодаря своей строгой типизации и совместимости с JavaScript. TypeScript идеально подходит для создания веб-сайтов и приложений, требующих высокой производительности и удобства поддержки кода. Его интуитивно понятный синтаксис и богатая экосистема делают его доступным как для начинающих разработчиков, так и для опытных профессионалов. Кроме того, TypeScript предоставляет широкий набор функций

для работы с различными протоколами и форматами данных, что значительно упрощает процесс разработки.

Преимущества использования Node.js включают высокую скорость разработки, возможность легкой интеграции с базами данных и доступность огромного количества готовых библиотек и фреймворков через npm (Node Package Manager). Серверные скрипты на Node.js способны обрабатывать множество входящих запросов одновременно благодаря асинхронной событийно-ориентированной архитектуре, что делает его особенно эффективным для приложений реального времени. Гибкость платформы позволяет использовать Node.js в самых различных проектах — от простых веб-сайтов до сложных веб-приложений, API и микросервисов. Благодаря единому языку JavaScript как на сервере, так и на стороне клиента, разработчики могут создавать более согласованные и масштабируемые решения, что значительно упрощает процесс разработки и поддержки кода. Кроме того, активное сообщество и постоянное развитие экосистемы Node.js обеспечивают наличие современных инструментов и решений для любых задач, что делает эту платформу одной из самых востребованных среди веб-разработчиков.

Управление базой данных стало важным аспектом разработки информационной системы. Для этих целей были использованы Prisma и pgAdmin — мощные инструменты для работы с базами данных. Prisma представляет собой современный ORM (Object-Relational Mapping), который предоставляет удобный интерфейс для взаимодействия с базами данных, автоматически генерируя типобезопасный код и упрощая выполнение операций с данными. Благодаря Prisma разработчики могут легко создавать, изменять и запрашивать данные, используя декларативный подход, что значительно ускоряет процесс разработки и снижает количество ошибок. pgAdmin — веб-приложение для администрирования баз данных — предоставляет исчерпывающий функционал для управления. Оно позволяет создавать и редактировать базы данных, выполнять SQL-запросы, управлять пользователями и правами доступа. Благодаря интуитивно понятному

интерфейсу pgAdmin разработчики могут решать задачи без необходимости углубляться в командную строку. Комбинация Prisma и pgAdmin обеспечивает эффективное управление базами данных, делая процесс разработки и администрирования удобным и продуктивным.

NestJS стал важной частью веб-приложения, предоставляя мощный и гибкий фреймворк для создания серверной части на основе Node.js. Эта технология позволяет разрабатывать масштабируемые и хорошо структурированные приложения, следуя принципам модульности и чистой архитектуры. NestJS использует современные подходы, такие как внедрение зависимостей (Dependency Injection) и декораторы, что делает код более организованным и удобным для поддержки. Фреймворк поддерживает интеграцию с различными библиотеками и инструментами, такими как Prisma, TypeORM и GraphQL, позволяя разработчикам создавать сложные API и обрабатывать запросы максимально эффективно. Например, маршруты могут быть легко настроены для обработки как синхронных, так и асинхронных операций, что обеспечивает высокую производительность и отзывчивость сервера.

Next.js стал важной частью веб-приложения, позволяя создавать высокопроизводительные и SEO-оптимизированные веб-сайты. Эта технология предоставляет мощный фреймворк для разработки на React, поддерживающий как серверный рендеринг (SSR), так и генерацию статических страниц (SSG). Next.js позволяет обновлять контент на веб-страницах динамически, что создает более быстрый и отзывчивый пользовательский опыт. Фреймворк используется для эффективной загрузки данных, предварительного рендеринга страниц и уменьшения времени отклика приложения. К примеру, маршрутизация и API-роуты могут быть легко настроены для обработки запросов на стороне сервера, что обеспечивает мгновенную обратную связь и снижает нагрузку на клиентскую часть приложения.

Для стилизации интерфейса информационной системы был использован Tailwind CSS. Этот фреймворк позволяет отделить содержание страницы от её

представления, что делает код более понятным и структурированным. Tailwind CSS упрощает управление макетом страниц, предоставляя готовые utility-классы для быстрой настройки стилей непосредственно в HTML. Благодаря встроенной поддержке адаптивного дизайна и медиазапросов, разработчики могут легко создавать интерфейсы, которые корректно отображаются на устройствах с различными размерами экранов. Кроме того, Tailwind CSS значительно улучшает визуальное восприятие пользовательского интерфейса, позволяя реализовать современные эффекты и анимации без необходимости писать сложные CSS-правила.

Jotai стал важным инструментом для управления состоянием в информационной системе. Он работает на стороне клиента и позволяет разработчикам эффективно внедрять управление глобальным и локальным состоянием, что значительно улучшает пользовательский опыт. Jotai используется для обработки изменений данных в реальном времени, что позволяет мгновенно реагировать на действия пользователей, такие как обновление интерфейса или изменение параметров. С его помощью можно реализовывать сложные сценарии взаимодействия между компонентами, предотвращая дублирование кода и обеспечивая согласованность данных. Благодаря простому и гибкому API, Jotai позволяет создавать динамическое содержимое, обновляя элементы страницы на лету, и поддерживает масштабируемость приложений без потери производительности.

Visual Studio Code — это редактор исходного кода. Он имеет многоязычный интерфейс пользователя и поддерживает ряд языков программирования, подсветку синтаксиса, IntelliSense, рефакторинг, отладку, навигацию по коду, поддержку Git и другие возможности. Многие возможности Visual Studio Code недоступны через графический интерфейс, зачастую они используются через палитру команд или JSON-файлы (например, пользовательские настройки). Палитра команд представляет собой подобие командной строки, которая вызывается сочетанием клавиш.

2 Разработка информационной системы

2.1 Разработка базы данных информационной системы

Для разработки базы данных были использованы ORM Prisma, СУБД PostgreSQL и программа для работы pgAdmin4.

PostgreSQL — это мощная объектно-реляционная система управления базами данных, известная высокой производительностью, широким функционалом и гибкостью. Для взаимодействия с базой данных в проекте используется Prisma — ORM для Node.js, которая обеспечивает удобную работу с данными на уровне объектов и позволяет избежать ручного написания большого количества SQL-запросов.

На рисунке 2.1 представлен интерфейс программного обеспечения PgAdmin4.

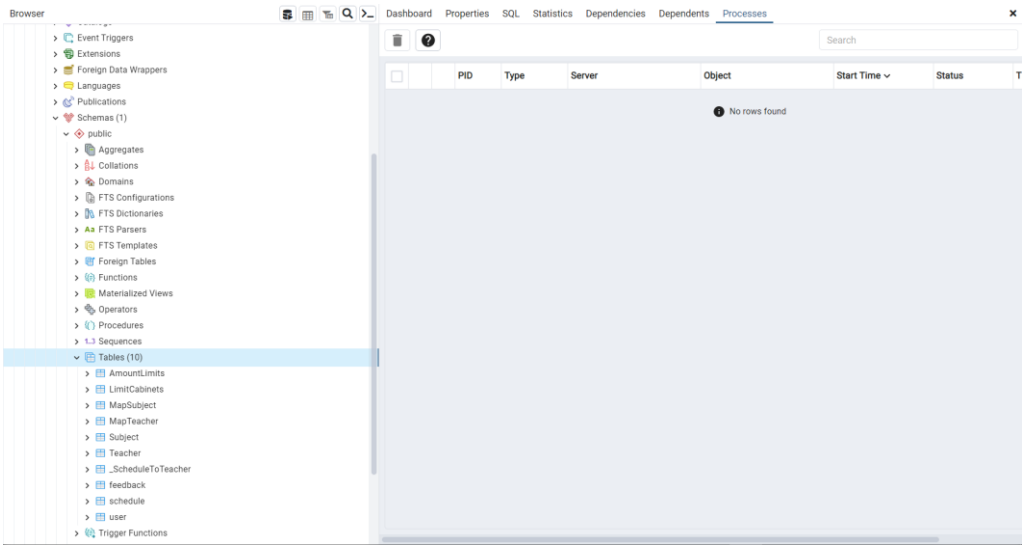

Рисунок 2.1 – Программное обеспечение PgAdmin4

В процессе разработки базы данных было создано 9 таблиц.

Список таблиц в PgAdmin4 представлен на рисунке 2.2.

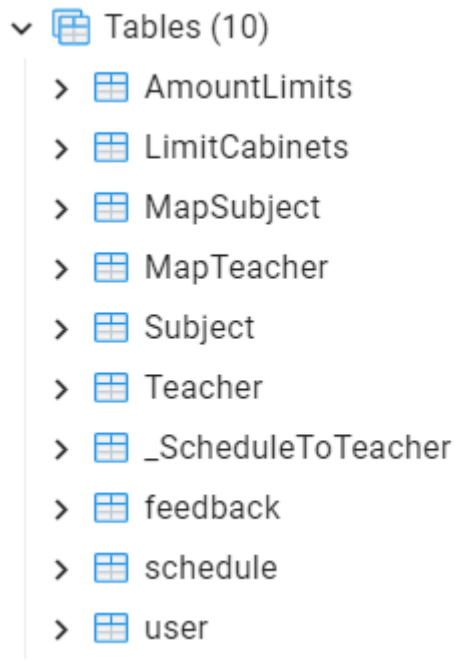

Рисунок 2.2 – Список таблиц

Подробную информацию о структуре и содержании таблиц можно найти в схеме БД. Схема базы данных — это её структура, описанная на формальном языке, поддерживаемом системой управления базами данных (СУБД).

Схема базы данных представлена на рисунке 2.3.

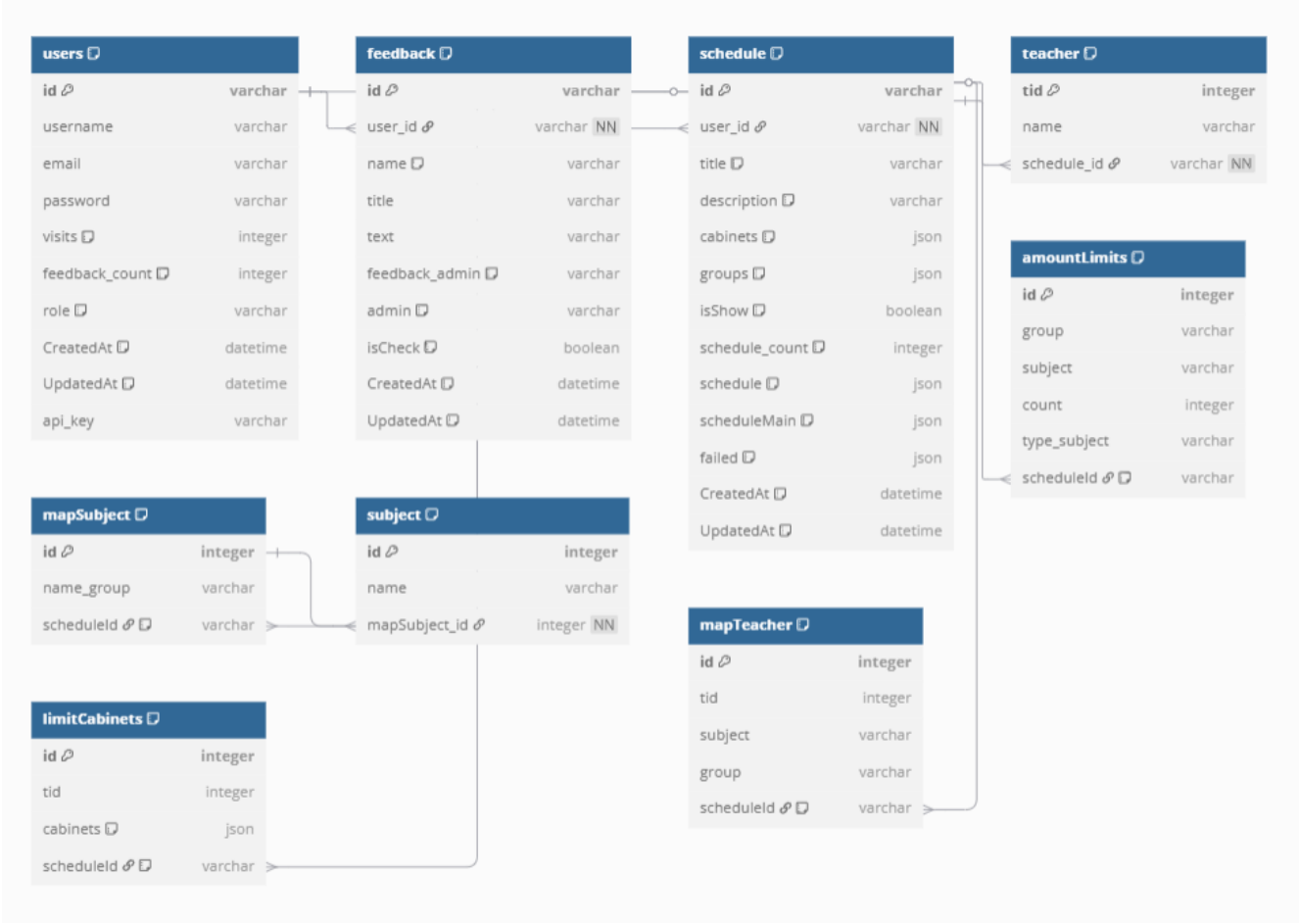

Рисунок 2.3 – Схема базы данных

Для создания таблиц было использована Prisma.

На листингах 2.1, 2.2 и 2.3 представлены Prisma-запросы на создание users (пользователь), schedule (расписание) и feedback (обратная связь).

Листинг 2.1 - Prisma-запрос на создание user (пользователь)

```
// Создайние пользователя
public async create(dto: RegisterAuth): Promise<User> {
return await this.prisma.user.create({
data: {
username: dto.username,
email: dto.email,
password: await hash(dto.password),
schedule: {
create: {
schedule: {},
},
},
},
```

```
include: {
schedule: true,
},
});
}
```

Листинг 2.2 - Prisma-запрос на создание schedule (расписание)

```
// Создание расписания
public async addGeneratedSchedulePrisma(
api_key: string,
data: any,
): Promise<any> {
if (!api_key || api_key.trim() === '') {
throw new Error('API key is required and must not be empty');
}

const user = await this.userService.getByApiKey(api_key);
if (!user) {
throw new Error('Пользователь не найден');
}

const schedule_json = await this.generateSchedule(data);

return await this.prisma.schedule.create({
data: {
schedule: schedule_json,
scheduleMain: { create: [] },
failed: { create: [] },
user_id: user.id,
cabinets: [],
groups: [],
teachers: { create: [] },
amountLimits: { create: [] },
limitCabinets: { create: [] },
isShow: true,
},
});
}
```

Листинг 2.3 - Prisma-запрос на создание таблицы feedback (обратная связь)

```
// Добавить запись
public async add(dto: AddFeedbackDto, user_id: string) {
const { text, title } = dto;

if (!user_id) {
throw new BadRequestException('Идентификатор пользователя не
найден');
}
```

```
const user = await this.userService.getById(user_id);

if (!user) {
throw new BadRequestException('Пользователь не найден');
}

const addFeedback = await this.prisma.feedback.create({
data: {
user_id: user.id,
title,
text,
name: user.username,
},
});

return addFeedback;
}
```

Чтобы подключить базу данных к веб-приложению и работать с данными, было создано 3 файла – это .env, prisma.service.ts и schema.prisma.

– .env – конфигурационный файл с параметрами для подключения к базе данных PgAdmin4.

Листинг 2.4 - Листинг кода .env

```
DATABASE_URL="postgresql://postgres:123@localhost:5432/schedul-
generator
?schema=public"
```

– prisma.service.ts – файл с классом PrismaService, содержит реализацию сервиса для работы с базой данных на основе Prisma ORM , интегрированного в приложение, разработанное с использованием фреймворка NestJS.

Листинг 2.5 - Листинг кода prisma.service.ts

```
import { Injectable, OnModuleInit } from '@nestjs/common';
import { PrismaClient } from '@prisma/client';
@Injectable()
export class PrismaService extends PrismaClient implements
OnModuleInit {
async onModuleInit() {
await this.$connect();
}
}
```

– schema.prisma – файл, в котором описываются настройки подключения к базе данных и генерации клиентской библиотеки Prisma.

Листинг 2.6 - Листинг кода schema.prisma

```
generator client {
provider = "prisma-client-js"
}

datasource db {
provider = "postgresql"
url= env("DATABASE_URL")
}
```

2.2 Разработка интерфейса информационной системы

В процессе разработки интерфейса информационной системы было разработано значительное количество веб-страниц. Ниже приведены некоторые из них.

Страница для генерации расписания представляет собой форму с добавлением преподавателя, группы, кабинета и предмета.

Страница для генерации расписания представлена на рисунке 2.4.

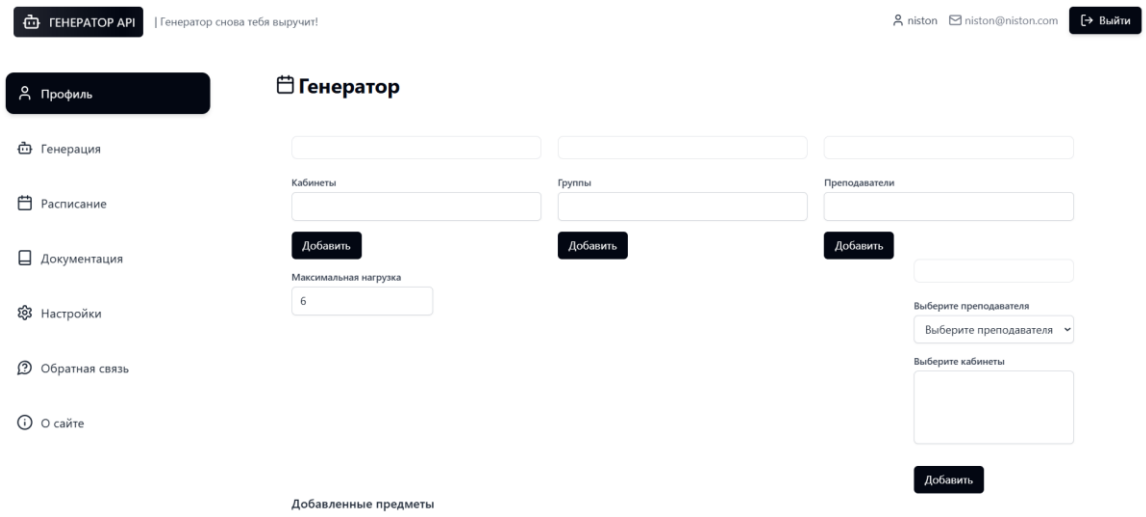

Рисунок 2.4 – Страница генерации расписания

Для создания интерфейса страницы и для генерации расписания были использованы язык разметки HTML и CSS-фреймворк TailwindCSS, который облегчает стилизацию элементов веб-страницы.

Код страницы для генерации расписания представлен на листинге 2.7.

Листинг 2.7 - Код генерации расписания

```
<div className='fixed inset-0 flex items-center justify-center bg-
black bg-opacity-50 z-50'>
<div className='bg-white p-6 rounded-lg shadow-lg w-full h-[90vh]
max-w-[95vw] flex flex-col overflow-hidden'>
<h2 className='text-xl font-bold mb-4'>Форма генерации</h2>
<div className='flex-1 overflow-y-auto pr-4'>
<div className='grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-
6'>
<div className='space-y-6'>
<Cabinets />
<Groups />
<Teachers />
</div>
<div className='space-y-6'>
<MaxLoad />
<CabinetLimits />
<Days />
</div>

<div className='space-y-6'>
<MultiSubject />
</div>
</div>
<div>
<AllSchedule />
</div>
</div>
<div className='flex justify-end gap-2 mt-4'>
<a
href={`http://localhost:5555/api/schedule/generate?api-
key=${data?.api_key}`}
target='_blank'
>
Перейти к расписанию
</a>

<button
type='button'
className='bg-gray-500 text-white px-4 py-2 rounded-md hover:bg-
gray-600'
onClick={() => handlerIsModal()}
>
```

```
Закрыть
</button>
<GenerationSchedule />
<CreateSchedule />
</div>
</div>
</div>
```

Страница сгенерированного расписания служит ключевым элементом информационной системы, поскольку на ней отображается готовое расписание, сформированное на основе введённых данных. На данной странице предусмотрена возможность редактирования уже существующих записей, удаления отдельных занятий, добавления новых элементов, а также повторной генерации определённого временного промежутка между указанными датами. Такой подход обеспечивает гибкость и удобство в управлении учебным графиком.

Пример интерфейса страницы сгенерированного расписания представлен на рисунке 2.5.

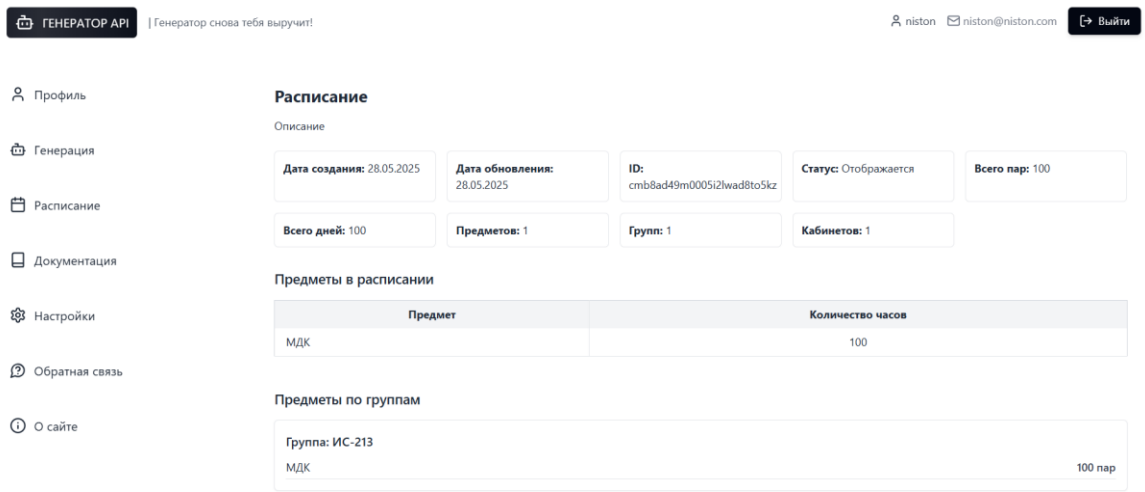

Рисунок 2.5 – Страница сгенерированного расписания

Для создания интерфейса страницы сгенерированного расписания также были использованы HTML и TailwindCSS.

Код страницы сгенерированного расписания представлен на листинге 2.8.

Листинг 2.8 - Код страницы сгенерированного расписания

```
(
schedule_id && (
<m.div
variants={{
hidden: { opacity: 0, y: 20 },
visible: { opacity: 1, y: 0, transition: { staggerChildren: 0.2 } },
}}
initial='hidden'
animate='visible'
className='p-6 bg-white rounded-lg shadow-md space-y-6 max-w-2xl mx-
auto'
>
{/* Форма для добавления кабинетов */}
<CabinetForm
profile={profile!}
schedule={schedule_id}
onCabinetAdded={newCabinets => {
setCabinets(prev => {
// Удаляем дубликаты и добавляем только уникальные кабинеты
const uniqueCabinets = Array.from(
new Set([...(prev || []), ...newCabinets])
);
return uniqueCabinets;
});
}}
/>

{/* Заголовок расписания */}
<m.div
variants={{
hidden: { opacity: 0, y: 20 },
visible: { opacity: 1, y: 0, transition: { duration: 0.5 } },
}}
>
<h2 className='text-xl font-bold text-gray-800 flex items-center
gap-2'>
<Book size={20} className='text-indigo-600' />
{schedule_id.title}
</h2>
</m.div>

{/* Список кабинетов */}
{cabinets.length > 0 && (
<m.div
variants={{
hidden: { opacity: 0, y: 20 },
visible: { opacity: 1, y: 0, transition: { duration: 0.5 } },
}}
>
```

```
<div className='flex items-center gap-2 mb-2'>
<Building size={18} className='text-gray-600' />
<span className='text-sm font-medium text-gray-700'>
Кабинеты:
</span>
</div>
<div className='flex flex-wrap gap-2'>
{cabinets.map((cabinet, index) => (
<div
key={index}
className='px-3 py-1 bg-gray-100 text-gray-700 rounded-full text-sm
font-medium'
>
{cabinet}
</div>
))}
</div>
</m.div>
)}

{/* Остальной код (преподаватели, группы и т.д.) оставлен без
изменений */}
</m.div>
)
);
```

Через страницу вывода расписаний пользователь может просматривать все сформированные учебные расписания. На этой странице представлена информация о различных расписаниях, включая их названия, даты создания, статусы, а также доступна фильтрация по названию расписания.

Страница вывода расписаний представлена на рисунке 2.6.

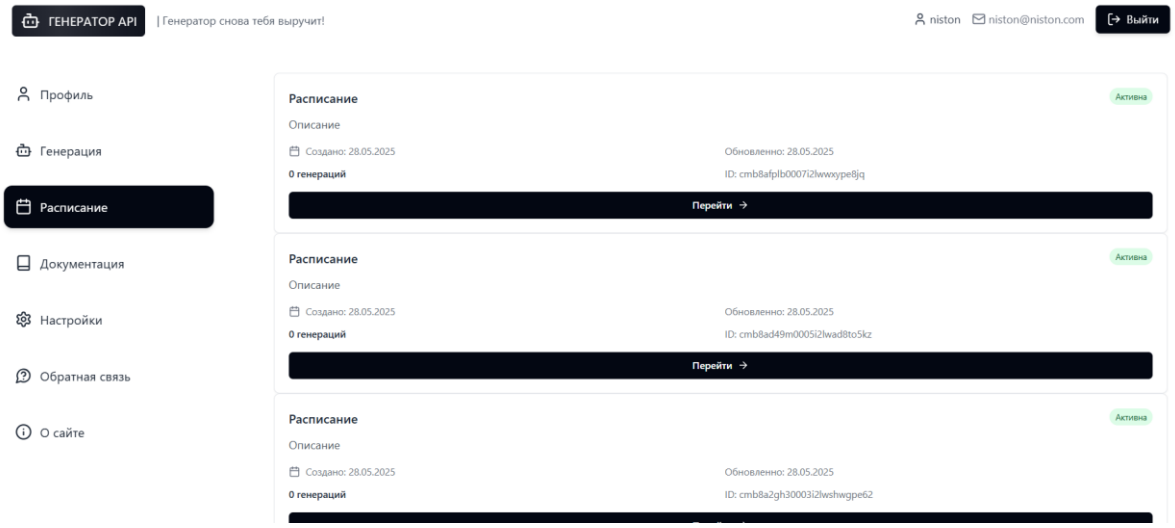

Рисунок 2.6 – Страница вывода расписаний

На странице статистики расписаний предусмотрена возможность фильтрации по временным промежуткам и типу расписания. В зависимости от выбранного интервала отображается информация о количестве сгенерированных расписаний, общем количестве занятий, загруженности преподавателей и аудиторий, а также сравнение текущих данных с предыдущим периодом.

Страница статистики расписаний представлена на рисунке 2.7.

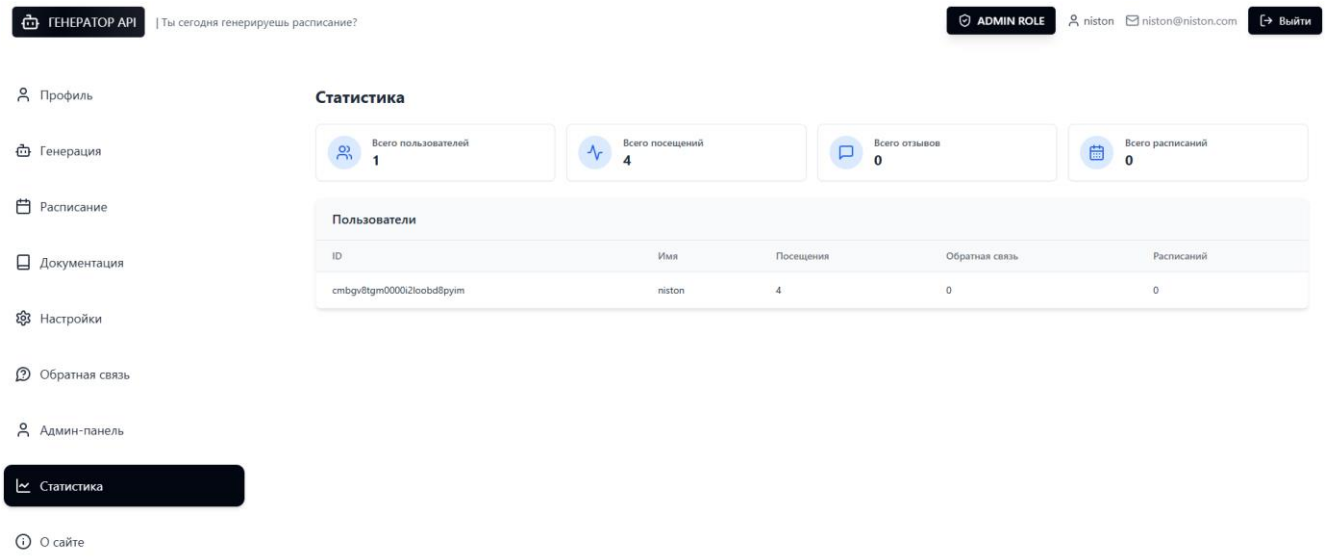

Рисунок 2.7 – Страница статистики

На странице личных данных пользователя предусмотрена возможность изменения логина, адреса электронной почты и пароля.

Страница изменения данных пользователя представлена на рисунке 2.8.

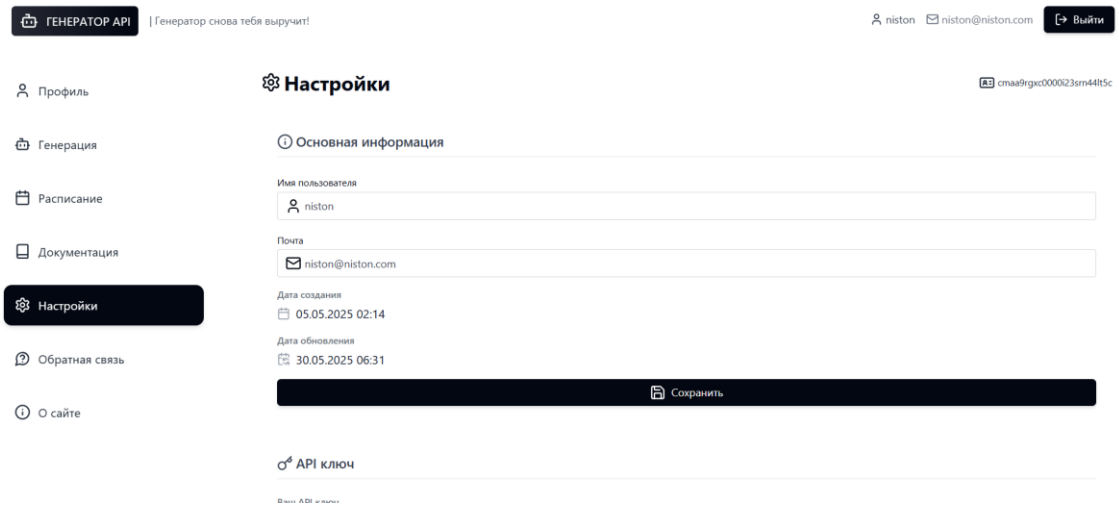

Рисунок 2.8 – Страница изменения данных пользователя

2.3 Разработка функциональной части информационной системы

В форме генерации расписания был реализован метод, который на основе заполненных данных формирует учебное расписание.

Сначала пользователь добавляет кабинеты, группы и преподавателей. Затем указывает максимальную нагрузку в день. Далее происходит привязка преподавателей к кабинетам и выбор дней недели, на которые будет сгенерировано расписание. После этого создаются учебные предметы, для которых указываются: группа, название предмета, преподаватель и тип занятия. На основе этих данных система формирует расписание, учитывая заданные параметры и ограничения.

Реализация данного функционала потребовала написания логики генерации расписания на TypeScript с последующим отображением результата на странице:

1. Объявить переменные:

– cabinets – список доступных аудиторий;

– groups – список учебных групп;

– teachers – список преподавателей;

– subjectsMap – сопоставление предметов по группам;

– teachersMap – привязка преподавателей к предметам;

– amountLimits – ограничения на количество занятий;

– cabinetLimits – привязка кабинетов к преподавателям;

– days – дни недели для генерации расписания;

– maxLoad – максимальная нагрузка;

2. Создать функции:

– createSubjectQueue – формирует очередь предметов для распределения, исходя из лимитов и приоритетов;

– findAvailableTeacherForSubject – ищет свободного преподавателя, который может вести указанный предмет в указанное время;

– findAvailableCabinetForTeacher - находит подходящую аудиторию для преподавателя в выбранный день и пару;

– markTeacherAndCabinetAsBusy – помечает преподавателя и кабинет как занятые в выбранное время;

– decrementAmountLimit – уменьшает оставшееся количество занятий для предмета согласно заданным лимитам.

Код генерации учебного расписания представлен на листинге 2.9.

Листинг 2.9 - Код генерации учебного расписания

```
async generateSchedule(data: any): Promise<any> {
const {
cabinets,
groups,
teachers,
subjectsMap,
teachersMap,
amountLimits,
cabinetLimits,
days,
maxLoad,
} = data;
```

```
if (!Array.isArray(amountLimits) || amountLimits.length === 0) {
throw new Error('Необходимо указать лимиты на количество занятий');
}

if (!Array.isArray(teachersMap) || teachersMap.length === 0) {
throw new Error(
'Необходимо указать соответствие преподавателей и предметов',
);
}

const groupTimetables: Record<string, Record<string, any[]>> = {};
const failedAllocations: {
group: string;
subject: string;
reason: string;
}[] = [];

// Создаем пустое расписание для каждой группы
for (const group of groups) {
groupTimetables[group] = {};
for (const day of days) {
groupTimetables[group][day] = [];
}

// Проверяем, есть ли предметы для группы
const subjects = subjectsMap[group] || [];
if (!Array.isArray(subjects)) {
throw new Error(`Invalid subjects data for group ${group}`);
}

// Создаем очередь предметов
const subjectQueue = this.createSubjectQueue(
subjects,
amountLimits.filter((limit) => limit.group === group),
);

// Инициализация карты доступности преподавателей и кабинетов
const teacherAvailability: Record<string, boolean[]> = {};
const cabinetAvailability: Record<string, boolean[]> = {};

for (const day of days) {
teacherAvailability[day] = Array(maxLoad).fill(true);
cabinetAvailability[day] = Array(maxLoad).fill(true);
}

let dayIndex = 0;
for (const subjectInfo of subjectQueue) {
let attempts = 0;
const maxAttempts = 1000; // Ограничение на количество попыток

while (subjectInfo.remaining > 0) {
```

```
if (attempts >= maxAttempts) {
// Логируем проблему, но продолжаем генерацию
failedAllocations.push({
group,
subject: subjectInfo.subject,
reason: `Не удалось распределить занятие после ${maxAttempts}
попыток`,
});
console.warn(
`Пропущено занятие: ${subjectInfo.subject} для группы ${group}`,
);
break;
}

const currentDay = days[dayIndex % days.length];

// Проверяем доступность преподавателя
const teacherInfo = this.findAvailableTeacherForSubject(
subjectInfo.subject,
teachersMap,
teachers,
group,
teacherAvailability[currentDay],
);
if (!teacherInfo) {
dayIndex = (dayIndex + 1) % days.length;
attempts++;
continue;
}

// Проверяем доступность кабинета
const cabinet = this.findAvailableCabinetForTeacher(
teacherInfo.tid,
cabinetLimits,
cabinets,
cabinetAvailability[currentDay],
groupTimetables[group],
currentDay,
groupTimetables[group][currentDay].length + 1,
maxLoad,
);
if (!cabinet) {
console.warn(
`Нет доступных кабинетов для преподавателя ${teacherInfo.name} (tid:
${teacherInfo.tid}) в день ${currentDay}`,
);
dayIndex = (dayIndex + 1) % days.length;
attempts++;
continue;
}
```

```
// Добавляем занятие в расписание группы
if (groupTimetables[group][currentDay].length < maxLoad) {
groupTimetables[group][currentDay].push([
{
cabinet,
teacher: teacherInfo.name,
subject: subjectInfo.subject,
group,
lessonType: subjectInfo.lessonType,
},
]);
this.markTeacherAndCabinetAsBusy(
teacherInfo.tid,
cabinet,
teacherAvailability[currentDay],
cabinetAvailability[currentDay],
groupTimetables[group][currentDay].length - 1,
);
}

subjectInfo.remaining--;
this.decrementAmountLimit(amountLimits, group, subjectInfo.subject);

dayIndex = (dayIndex + 1) % days.length;
attempts = 0; // Сбрасываем счетчик попыток при успешном
распределении
}
}
}
```

На странице сгенерированного расписания была реализована возможность оставить заявку в техническую поддержку, чтобы задать любой вопрос, предложение или пожелание разработчикам. Для этого предусмотрена кнопка «Сообщить об ошибке» или «Оставить отзыв», при нажатии на которую открывается модальное окно. В этом окне пользователь может выбрать тип обращения (например, «Ошибка», «Предложение», «Общее обращение»), указать тему и описать само сообщение.

Если пользователь авторизован, его данные подставляются автоматически. После заполнения формы и нажатия на кнопку «Отправить» происходит отправка данных на сервер для последующей обработки администратором.

Для реализации данного функционала в файл feedback.service.ts необходимо было добавить TypeScript-код, в котором:

– add(dto: AddFeedbackDto, user_id: string) – добавляет новую заявку от пользователя. Проверяется наличие идентификатора пользователя и его существование. После этого создаётся запись с темой, текстом сообщения и данными автора;

– feedback_admin(dto: AdminFeedbackDto, feedback_id: string) – позволяет администратору ответить на обращение, указав статус, комментарий и имя администратора;

– getId(feedback_id: string) – возвращает конкретное обращение по его идентификатору;

– get(user_id: string) – возвращает все обращения, созданные конкретным пользователем;

– getAll() – извлекает список всех обращений пользователей для просмотра администратором;

– put(dto: ChangeFeedbackDto, feedback_id: string) – редактирует существующее обращение, включая обновление заголовка, текста, ответа администратора и статуса;

– delete(feedback_id: string) - удаляет обращение по его идентификатору.

На листинге 2.10 представлен код сервиса работы с обращениями в техническую поддержку, включающий методы добавления, редактирования, получения и удаления заявок. Также представлены проверки на авторизацию пользователя и корректность переданных данных.

Листинг 2.10 - Код сервиса работы с обращениями в техническую поддержку

```
@Injectable()
export class FeedbackService {
constructor(
private readonly prisma: PrismaService,
private readonly userService: UserService,
) {}

// Добавить запись
public async add(dto: AddFeedbackDto, user_id: string) {
const { text, title } = dto;

if (!user_id) {
```

```
throw new BadRequestException('Идентификтатор пользователя не
найден');
}

const user = await this.userService.getById(user_id);

if (!user) {
throw new BadRequestException('Пользователь не найден');
}

const addFeedback = await this.prisma.feedback.create({
data: {
user_id: user.id,
title,
text,
name: user.username,
},
});

return addFeedback;
}

// Ответить на запись
public async feedback_admin(dto: AdminFeedbackDto, feedback_id:
string) {
const { feedback_admin, isCheck, admin } = dto;

if (!feedback_id) {
throw new BadRequestException('Идентификатор обратной связи не
найден');
}

const feedback = await this.prisma.feedback.update({
where: {
id: feedback_id,
},
data: {
admin,
feedback_admin,
isCheck,
},
});

return feedback;
}

// Вывод одной записи
public async getId(feedback_id: string) {
if (!feedback_id) {
throw new BadRequestException('Запись не найдена');
}
```

```
const feedback = await this.prisma.feedback.findFirst({
where: {
id: feedback_id,
},
});

return feedback;
}

public async get(user_id: string) {
if (!user_id) {
throw new BadRequestException('Идентификатор пользователя не
найден');
}

const user = await this.userService.getById(user_id);

if (!user) {
throw new BadRequestException('Пользователь не найден');
}

const feedback = await this.prisma.feedback.findMany({
where: {
user_id: user.id,
},
});

return feedback;
}

// Вывод всех обращений
public async getAll() {
const feedback = await this.prisma.feedback.findMany();

return feedback;
}

// Изменить запись
public async put(dto: ChangeFeedbackDto, feedback_id: string) {
if (!feedback_id) {
throw new BadRequestException('Идентификатор обратной связи не
найден');
}

const feedback = await this.prisma.feedback.update({
where: {
id: feedback_id,
},
data: {
title: dto.title,
```

```
text: dto.text,
admin: dto.admin,
feedback_admin: dto.feedback_admin,
isCheck: dto.isCheck,
},
});

return feedback;
}

// Удалить запись
public async delete(feedback_id: string) {
const delete_feedback = await this.prisma.feedback.delete({
where: {
id: feedback_id,
},
});

return delete_feedback;
}
}
```

2.4 Разработка подсистемы безопасности информационной системы

Информационная безопасность — это обеспечение защиты информации от несанкционированного доступа, разглашения, модификации или уничтожения. ИБ включает в себя применение технических, организационных и процедурных мер для защиты конфиденциальности, целостности и доступности информации, а также обеспечение ее сохранности в цифровом и физическом виде.

Безопасность веб-приложений относится к защите веб-приложений от различных угроз и атак, направленных на нарушение их функциональности или получение несанкционированного доступа к данным. Это включает в себя реализацию мер безопасности на уровне кода, аутентификации пользователей, защиты от инъекций, управления сеансами и других технических механизмов для предотвращения уязвимостей и обеспечения надежной работы веб-приложений.

В данной информационной системе, с целью обеспечения надежной защиты от атак и взломов, были использованы такие методы обеспечения безопасности, как:

– Хеширование паролей – этот процесс представляет собой преобразование исходного пароля в строку случайных символов фиксированной длины с использованием криптографических хеш-функций. Хеширование паролей необходимо для обеспечения безопасности пользовательских данных.

– В методе create, который отвечает за регистрацию нового пользователя, пароль хешируется перед его сохранением в базе данных. Для этого используется асинхронная функция hash(), которая применяет стойкий алгоритм хеширования argon2. Это гарантирует, что в базе данных будет храниться не сам пароль, а его безопасная хэш-сумма.

На листинге 2.11 представлен код хеширования пароля при регистрации.

Листинг 2.11 - код хеширования пароля при регистрации

```
// Создайние пользователя
public async create(dto: RegisterAuth): Promise<User> {
return await this.prisma.user.create({
data: {
username: dto.username,
email: dto.email,
password: await hash(dto.password),
schedule: {
create: {
schedule: {},
},
},
},
include: {
schedule: true,
},
});
}
```

– Сообщения об ошибках регистрации при вводе простого пароля с фразами «Пароль должен содержать хотя бы одну букву в нижнем регистре и хотя бы одну букву в верхнем регистре. Допускаются только латинские буквы и цифры». Это помогает улучшить безопасность паролей, предотвращая использование слабых комбинаций символов, что снижает риск несанкционированного доступа.

На рисунке 2.9 представлен хешированный пароль в базе данных.

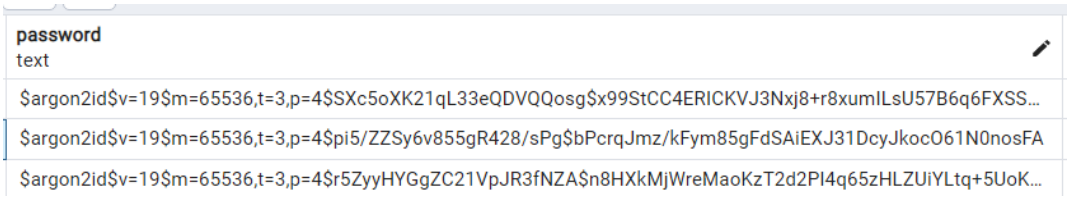

Рисунок 2.9 – Хешированный пароль в базе данных

На рисунке 2.10 представлены ошибки регистрации при вводе простого пароля.

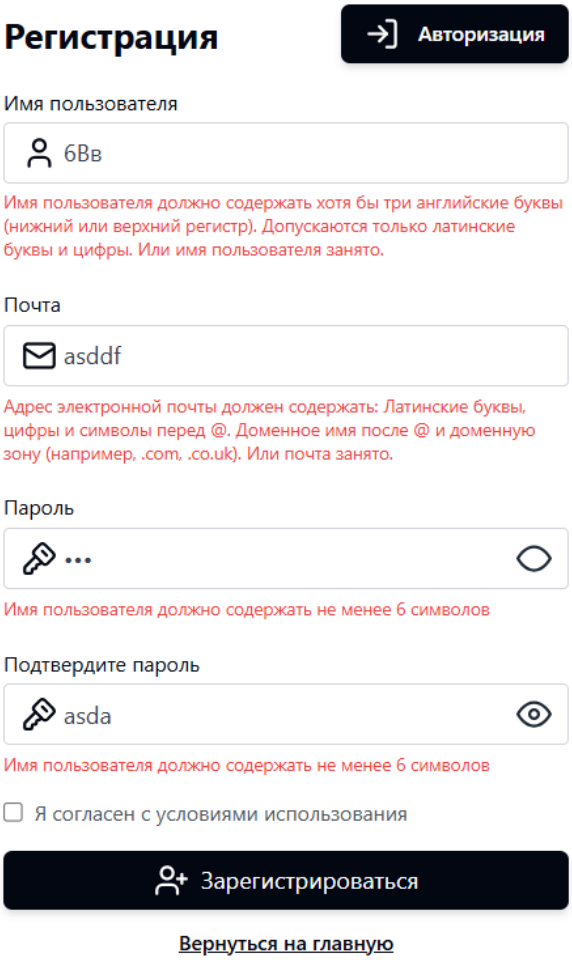

Рисунок 2.10 – Ошибки регистрации при вводе простого пароля

3. Тестирование информационной системы

Тестирование программного обеспечения представляет собой процесс проверки соответствия программного продукта установленным требованиям и выявления возможных ошибок на этапе разработки. Этот процесс включает применение различных методов и подходов, направленных на обеспечение высокого качества программного обеспечения.

В контексте системы автоматической генерации учебного расписания для образовательных учреждений тестирование приобретает особую значимость, поскольку от корректности работы системы напрямую зависит удобство ее использования студентами, преподавателями и административным персоналом, а также общая эффективность образовательного процесса.

Основные виды тестирования:

– Функциональное тестирование ориентировано на проверку соответствия системы заявленным требованиям. Оно охватывает модульное тестирование, которое проверяет отдельные компоненты системы, такие как функция валидации пароля. Интеграционное тестирование оценивает взаимодействие между различными модулями, например, между корзиной заказов и процессом оформления. Системное тестирование предполагает комплексную проверку всей системы в сборе.

– Нефункциональное тестирование сосредоточено на аспектах, не связанных напрямую с функциональностью. Оно включает тестирование удобства использования, направленное на оценку интуитивности интерфейса. Тестирование производительности позволяет оценить поведение системы под нагрузкой, например, при одновременном отправке множество запросов. Тестирование безопасности нацелено на выявление уязвимостей, таких как риск SQL-инъекций.

Особое значение имеет тестирование на корректных и некорректных данных. Позитивное тестирование проверяет работу системы с валидными входными данными, такими как корректный email при регистрации. Негативное тестирование

оценивает обработку системой ошибочных данных, например, ввод пароля недостаточной длины.

Процессы тестирования:

– Планирование тестирования начинается с определения целей и выбора методологии, которая может включать ручное или автоматизированное тестирование. На этом этапе также разрабатываются тест-кейсы.

– Разработка тест-кейсов предполагает создание последовательности действий для проверки конкретного функционала. Каждый тест-кейс содержит предусловия, шаги выполнения, а также ожидаемый и фактический результаты.

– Выполнение тестов может осуществляться вручную, когда тестировщик последовательно проверяет каждый сценарий, или автоматизированно с использованием специализированных инструментов, таких как Selenium для веб-интерфейсов.

– Анализ результатов включает фиксацию выявленных багов с использованием систем управления задачами, таких как Jira или Trello, и формирование отчетов о проведенном тестировании.

Для системы автоматической генерации учебного расписания критически важными являются несколько аспектов: валидация данных обеспечивает корректность ввода информации об аудиториях, преподавателях, группах и временных ограничениях, работа с расписанием включает проверку процессов создания, обновления и редактирования расписания, интеграция с внешними системами требует особого внимания к безопасности и надежности передачи данных, обработка ошибок и исключений гарантирует корректное уведомление пользователей о конфликтах, удобство использования интерфейса обеспечивает простоту заполнения формы для генерации расписания и доступность функций экспорта через API, производительность системы проверяется на способность работать с большими объемами данных.

Выявление ошибок разработчика со стороны программиста представлены в таблице 3.1.

Таблица 3.1 - Тест кейс №1

№	Тест кейс №1
Название	Регистрация пользователя с валидацией полей
Место тестирования	Страница «Регистрация».
Шаги тестирования	Открыть страницу регистрации. Последовательно заполнять поля формы, проверяя каждый раз валидацию. Нажать кнопку «Зарегистрироваться».
Входные данные	Имя пользователя: useruser, Почта: useruser@user.com, Пароль: 123, Подтверждение пароля: 321
Ожидаемый результат	Успешное открытие страницы регистрации Вывод ошибки и обозначение ошибки красным цветом
Фактический результат	Успешная авторизация; Для поле имя пользователя выводятся сообщения «Разрешенные символы (кириллица, пробел и тире)» Для поля Адрес электронной почты выводятся сообщения «Введите правильный адрес электронной почты. Для поля Пароль «Введённый пароль слишком короткий. Он должен содержать как минимум 6 символов. Пароль должен содержать хотя бы одну букву в нижнем регистре и хотя бы одну букву в верхнем регистре. Допускаются только латинские буквы и цифры.» Для поля Подтверждение пароля «Введенные пароли не совпадают.».
Комментарий	Присутствует валидация полей

Выявление ошибок с помощью тест-кейса, который описан в таблице 3.2.

Таблица 3.2 - Тест кейс №2

№	Тест кейс №2
Название	Авторизация пользователя с валидацией полей
Место тестирования	Страница «Авторизация».
Шаги тестирования	1) Открыть страницу авторизации. 2) Последовательно заполнять поля формы, проверяя каждый раз валидацию. Нажать кнопку «Войти».
Входные данные	Имя пользователя: useruser, Пароль: 123,
Ожидаемый результат	Успешное открытие страницы авторизации Вывод ошибки и обозначение ошибки красным цветом

Продолжение таблицы 3.2

Фактический результат	Успешная вход в профиль; Для поле имя пользователя выводятся сообщения «Имя пользователя должно содержать не менее 3 символов» Для поля Пароль «Введённый пароль слишком короткий. Он должен содержать как минимум 6 символов. Пароль должен содержать хотя бы одну букву в нижнем регистре и хотя бы одну букву в верхнем регистре. Допускаются только латинские буквы и цифры.»
Комментарий	Присутствует валидация полей

Тест-кейс на оформление заказа с корзины с валидацией описан в таблице 3.3.

Таблица 3.3 - Тест кейс №3

№	Тест кейс №3
Название	Отправка вопроса в техподдержку с валидацией полей
Место тестирования	Страница «Обратная связь».
Шаги тестирования	Открыть страницу отправки вопроса в техподдержку. Последовательно заполнять поля формы. Нажать кнопку «Зарегистрироваться».
Входные данные	Название: Вопрос, Сообщение: У меня возник вопрос
Ожидаемый результат	Успешное открытие страницы отправки вопроса в техподдержку. Вывод ошибки и обозначение ошибки красным цветом
Фактический результат	Успешное открытие страницы отправки вопроса в техподдержку; Для поле название выводятся сообщения «Поле обязательное» Для поля сообщение выводятся сообщения «Поле обязательное»
Комментарий	При успешной отправке вопроса пользователь получает уведомление о том, что его обращение отправлено в техподдержку

Таблица 3.4 описывает проверку успешной генерации расписания через ввод формы и процесс генерирования. Система должна корректно обрабатывать ввод данных пользователем, проверять их на соответствие требованиям и формировать расписание в соответствии с заданными параметрами (кабинеты, группы, преподаватели, временные ограничения). Успешное прохождение теста подтверждает, что модуль генерации расписания работает в соответствии с требованиями.

Таблица 3.4 - Тест кейс №4:

№	Тест кейс №4
Название	Успешная генерация расписания
Место тестирования	Страница «Генерация расписания»
Шаги тестирования	Авторизоваться в системе как зарегистрированный пользователь Перейти на страницу генерации расписания. Ввести параметры для генерации расписания. Нажать кнопку "Сгенерировать расписание".
Входные данные	Кабинеты: ["101”, “102”, “103”]. Группы: [“Г1”, “Г2”]. Преподаватели: [{ tid: 1, name: 'Иванов' }, { tid: 2, name: 'Петров' }]. Предметы для группы: {'Группа_1': ['Математика', 'Физика'], 'Группа_2': ['Химия']} Соответствие преподавателей и предметов: [{ group: 'Группа_1', subject: 'Математика', tid: 1 }]. Лимиты на количество занятий: [{ group: 'Группа_1', subject: 'Математика', amount: 3 }]. Ограничение по кабинетам у преподавателей: [{ tid: 1, cabinets: ['101'] }]. Дни: [“2025-04-14”, “2025-05-15”]. Максимальное количество пар в день: 4
Ожидаемый результат	После нажатия кнопки «Сгенерировать расписание» система успешно создает расписание. Для каждой группы сформировано расписание с учетом всех введенных параметров. Отображается сообщение «Расписание успешно сгенерировано». Пользователь может просмотреть сгенерированное расписание на странице.
Фактический результат	После нажатия кнопки «Сгенерировать расписание» система успешно создает расписание. Для каждой группы сформировано расписание с учетом всех введенных параметров. Отображения сообщения «Расписание успешно сгенерировано». Пользователь может просмотреть сгенерированное расписание на странице.
Комментарий	Проверка успешной генерации расписания с учетом всех ограничений и параметров.

4 Экономическая эффективность внедрения информационной системы

Применение современных информационных технологий сопряжено с капитальными затратами на приобретение вычислительной техники, на разработку программных продуктов и их внедрение в управленческо-производственный процесс, обучение и подготовку персонала.

Эффективность является сложной экономической категорией, которая складывается на предприятии под влиянием множества факторов: экономических, социальных, правовых и других.

В экономической и научно-технической литературе термин «эффективность» понимается по-разному:

– Как вероятность выполнения поставленных перед системой задач;

– Как отношение реализованного эффекта к максимально возможному.

Экономическую эффективность лучше определять, как меру целесообразности проведения тех или иных мероприятий и выражать ее количественными величинами. Кроме того, под экономической эффективностью обычно понимают отношение между полученными результатами и затратами средств и труда.

Тогда можно дать и такое определение экономической эффективности — это способ действий, обеспечивающий получение в результате осуществляемых усилий и затрат ресурсов максимального (наилучшего) результата.

Понятие эффективности предполагает оценку результатов функционирования системы; это показатель, сопоставляющий в той или иной форме результаты функционирования системы. Общей конечной целью в данном случае является улучшение деятельности предприятия. Частными целями могут быть: снижение затрат на обработку информации; сокращение времени получения результатной информации; получение новой информации, которую без применения ЭВМ получить невозможно.

В современных условиях применяется более широкое понятие социально-экономической эффективности, включающее категории социальных издержек (заболеваемость, загрязнение окружающей среды и т.д.) и социальных благ (здоровье, научный потенциал).

Чтобы выявить в расчетах экономическую эффективность, надо знать, в каких показателях могут быть выражены результаты внедрения информационной системы. Оценка экономической эффективности состоит в определении ряда показателей, характеризующих использование различных видов ресурсов: повышение качества и снижение себестоимости, рост производительности труда управленческих работников и других. Одни из показателей дают оценку прямого эффекта от применения программных продуктов, другие косвенно характеризуют экономическую эффективность.

Общая экономическая эффективность слагается из прямой и косвенной эффективности. Прямая эффективность связана с сокращением затрат труда, с экономией материально-трудовых ресурсов и денежных средств, полученной в результате уменьшения численности персонала, расхода основных средств и вспомогательных материалов. Показатели ее могут быть измерены и выражены в количественных величинах.

Косвенная эффективность проявляется в улучшении работы управленческого персонала, благодаря использованию всесторонней и более качественной информации, что отражается на конечных результатах финансово-хозяйственной деятельности предприятия. Ее критериями могут быть:

– Сокращение сроков формирования документов;

– Повышение качества планово-учетных, контрольных и аналитических операций;

– Сокращение объема документооборота;

– Повышение управленческой культуры;

– Рост производительности труда.

Основным показателем является повышение качества управления, которое, как и при прямой эффективности, ведет к экономии живого и овеществленного труда.

Показатели прямой эффективности подразделяются на первичные и производные и могут быть трудовыми и стоимостными. К таким показателям относят следующие:

– Экономия рабочего времени;

– Индекс экономии затрат труда;

– Индекс производительности труда;

– Коэффициент снижения себестоимости и др.

Обобщающими показателями прямой эффективности, позволяющими судить о целесообразности применения технических средств, являются:

– Годовой экономический эффект;

– Период окупаемости единовременных затрат;

– Коэффициент экономической эффективности (рентабельности) затрат или капитальных вложений.

Экономический эффект – это результат внедрения какого-либо мероприятия, выраженный в стоимостной форме, в виде экономии от его осуществления.

Коэффициент экономической эффективности капитальных вложений показывает величину годового прироста прибыли, образующуюся в результате эксплуатации программного изделия, на один рубль капитальных единовременных вложений.

Период окупаемости (величина обратная коэффициенту эффективности) представляет собой период времени, в течение которого произведенные затраты на программное изделие окупаются полученным эффектом.

Определение эффективности информационной системы основано на принципах экономической оценки производства и использования новой техники.

На различных стадиях жизненного цикла информационной системы и в зависимости от целей расчета рассчитываются и документально оформляются

следующие виды экономического эффекта: предварительный; потенциальный; гарантированный; фактический.

Предварительный экономический эффект рассчитывается до выполнения разработки на основе данных технических предложений и прогноза использования.

Потенциальный экономический эффект рассчитывается на основе достигнутых технико-экономических характеристик разработанной информационной системы. Потенциальный эффект используется при оценке деятельности организаций-разработчиков информационных систем.

Гарантированный экономический эффект рассчитывается в виде эффекта для конкретного объекта внедрения и общего гарантированного внедрения по ряду объектов.

Фактический экономический эффект рассчитывается на основе данных учета и сопоставления затрат и результатов при конкретных применениях информационной системы.

Количественно измерить влияние автоматизированной обработки данных на результаты финансово-хозяйственной деятельности не всегда возможно, так как повышение эффективности зависит не только от применения технических средств, но и от влияния других многочисленных факторов.

Перечень работ по созданию автоматизированной информационной системы приведен в таблице 4.1.

Таблица 4.1 – Перечень работ по стадиям проектирования программного модуля

Наименование этапа	Продолжительность (трудоемкость, чел*час)	Удельный вес, %	Исполнители	
			Руководитель	Программист
Проектирование информационной системы	20	6.25%	+	+
Разработка информационной системы	250	78.13%	-	+
Тестирование информационной системы	30	9.38%	-	+

Продолжение таблицы 4.1

Оформление документов	20	6.25%	+	+
Итог	320	100%	2	4

На графике 4.1 представлен линейный график разработки АИС.

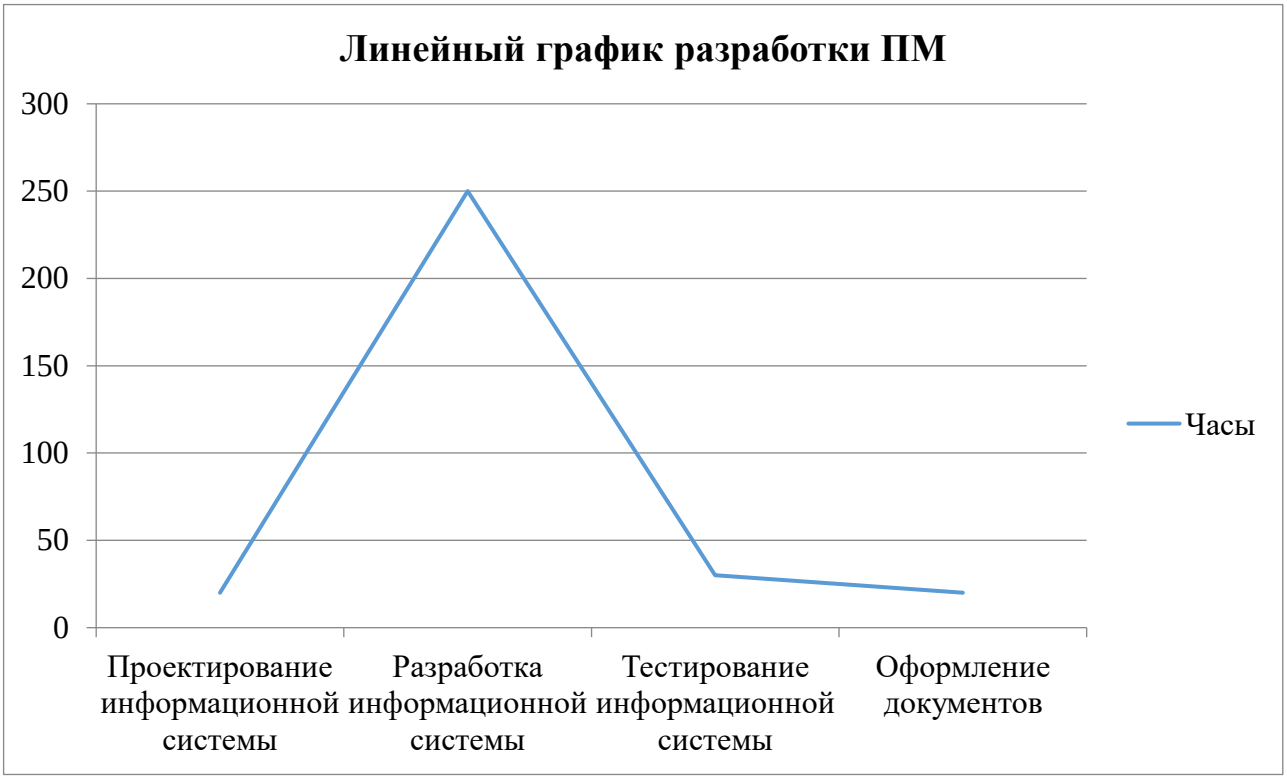

График 4.1 – Линейный график разработки АИС

В результате составленной таблицы наибольшее количествотрудозатрат при разработке программного продукта приходится на разработку информационной системы.

Для оценки экономического эффекта от внедрения веб-приложения в образовательной организации необходимо рассчитать затраты на разработку.

Величина затрат на создание веб-приложения определяется на основе метода калькуляций по отдельным статьям расходов и их последующим суммированием.

Разработка веб-приложения состоит из нескольких стадий проектирования.

В качестве разработчиков системы выступают: руководитель и программист (студент). Таким образом, руководитель участвует 50%, а программист 100% в проекте. Программист принимается за единицу приведенного исполнителя, следовательно, руководитель составляет 0,5 приведенного исполнителя. Т.е. в разработке проекта участвует 1,5 приведенного человека.

Трудоемкость = 320/1,5 = 213.3 чел ∗ час

Срок исполнения, учитывая, что рабочий день составляет 8 ч., получается 27 дней работы над проектом.

В реальности длительность работы над проектом больше примерно на 30%, поэтому можно принять за срок разработки 35 дней.

Постоянные издержки включают в себя амортизационные отчисления на компьютер и программное обеспечение и затраты на текущий ремонт.

Амортизационные отчисления.

Амортизационные отчисления на компьютер и программное обеспечение производятся ускоренным методом с тем условием, что срок морального старения происходит через четыре года.

В таблице 4.2 описан перечень используемого оборудования.

Таблица 4.2 – Перечень используемого оборудования

Наименование изделия	Количество, шт.	Цена за единицу, руб/шт.	Сумма затрат, шт.
Ноутбук Acer nitro 5	1	70000	70000
Принтер	1	45000	45000
Клавиатура	1	3000	3000
Мышка	1	1500	1500
Итог			119500

Балансовая стоимость ЭВМ включает отпускную цену, расходы на транспортировку, монтаж оборудования и его наладку и вычисляется по формуле:

$$A_{об} = \sum \frac{К_{П} \cdot Н_{А} \cdot n}{100\% \cdot 12} \quad (4.1)$$

где $К_П$ – первоначальная стоимость оборудования, руб.;

$Н_А$ – норматив амортизационных отчислений на полное восстановление, % (20%);

n – количество месяцев работы оборудования при создании АИС.

Из таблицы 2 видно, что первоначальная стоимость данного оборудования составляет 119500 руб. Оборудование использовалось 320 часов (160 часов в месяц) или ~2 месяца. Рассчитаем амортизационные отчисления оборудования:

$$А_{об} = (119500 * 0,2 * 2) / 12 = 3983.33 \text{ руб.}$$

Остальные программные средства, используемые для разработки, предоставляются бесплатно.

Затраты на текущий ремонт и обслуживание оборудования рассчитываются по формуле:

$$З_{ро} = \frac{К_п \cdot Н_{ро} \cdot n}{100\% \cdot 12} \quad (4.2)$$

где $К_п$ – первоначальная стоимость оборудования, руб.;

$Н_{ро}$ – норматив отчислений на ремонт и обслуживание оборудования, % (5%);

n – количество месяцев работы оборудования при создании АИС:

Рассчитаем затраты на текущий ремонт и обслуживание оборудования:

$$З_{ро} = (119500 * 0,05 * 2) / 12 = 995.83 \text{ руб.}$$

Полученные данные составляют постоянные издержки и приведены в таблице. Большую долю в постоянных издержках занимают амортизационные отчисления на используемый компьютер и программное обеспечение.

В таблице 4.3 описаны постоянные издержки.

Таблица 4.3 – Постоянные издержки

Вид постоянных издержек	Затраты, руб.	Удельный вес, %
Амортизационные отчисления	3983.33	80
Текущий ремонт	995.83	20
Итого	4979.16	100

Переменные издержки включают в себя:

– Материальные затраты;

– Затраты на потребляемую электроэнергию;

– Затраты на оплату труда;

– Отчисления по налогам.

Материальные затраты.

К материальным затратам относятся отчисления на материалы, которые используются в процессе разработки и внедрения настольного приложения. Использованные для работы материалы представлены в таблице 4.4.

Таблица 4.4 – Материальные расходы

Наименование изделия	Количество, шт.	Цена за единицу, руб./шт.	Сумма затрат, руб.
Катридж для принтера	1	500	500
Обслуживание принтера	1	600	600
Интернет	3	1000	3000
Итог			4100

К затратам на электроэнергию относится стоимость потребляемой электроэнергии ЭВМ за период разработки.

Затраты на электроэнергию рассчитываются по формуле:

$$Э_Т = \sum_{i=1}^{N} Н_{Эi} \cdot t_i \cdot Ц_э \qquad (4.3)$$

где $H_{Эi}$ – норма расхода энергии в единицу времени i-го потребителя, кВт/ч;

t_i – время работы i-го оборудования при создании АИС, ч.;

$Ц_э$ – цена 1 кВт/ч энергии, руб. (по данным на 2024 г. стоимость 1кВт/ч для населения составляет 5,38, данные брались с сайта http://www.energo-consultant.ru);

N – общее количество потребителей энергии.

Расчет затрат на электроэнергию сведем в таблицу 4.5.

Таблица 4.5 – Затраты на электроэнергию

Потребитель	Норма расхода энергии, кВт/ч.	Время работы, ч.	Цена 1 кВт/ч, руб.	Сумма затрат, руб.
Ноутбук	0.117	320	5,38	201.4
Итог				201.4

Расчет основной заработной платы производится на основе доли выполнения работы и величины месячного должностного оклада разработчика.

При расчете основной заработной платы, представленном в таблице 6, за период разработки необходимо учесть, что руководитель участвует в разработке проекта только на этапах системного анализа и анализа требований, которые занимают 10% всего времени, оклад куратора 43000.

Затраты на оплату труда разработчиков программного модуля включают в себя следующие компоненты:

– основная заработная плата разработчиков.

Для этого нам необходимо применить следующую формулу:

$$З_о = \sum_{i=1}^{N} С_ч \cdot t_{шт i} \quad (4.4)$$

где $С_ч$ – часовая тарифная ставка с учетом доплат, руб./ч;

$t_{шт i}$ – трудоемкость выполнения i-го вида работ при создании АИС, ч.

$З_о = 325 * 320 = 104000$ руб.

Дополнительная заработная определяется по формуле:

$$З_д = \frac{З_о \cdot Н_{зд}}{100\%} \quad (4.5)$$

где $Н_{ЗД}$ – норматив дополнительной заработной платы, % (9%);

$З_о$ – основная заработная плата работника.

Рассчитаем затраты на дополнительную заработную плату:

$З_д$ = 104000 * 0,09 = 9360 (руб.) (программист)

$З_д$ = 43000 * 0,09 = 3870 (руб.) (куратор)

Расчет отчислений на социальное страхование ведется по формуле:

$$О_{сс} = \frac{(З_о + З_д) \cdot Н_{сс}}{100\%} \quad (4.6)$$

где $Н_{сс}$ – норматив отчислений на социальное страхование, (30%);

$(З_о + З_д)$ – фонд заработной платы, руб.

Рассчитаем размер отчислений на социальное страхование:

Зсс = (104000 + 9360) * 0,3 = 34008 (руб.) (программист)

Зсс = (43000 + 3870) * 0,3 = 4061 (руб.) (куратор)

Расчет фонда заработной платы за период разработки, представлен в таблице 4.6.

Таблица 4.6 – Расчет фонда заработной платы за период разработки

Должность	Вид заработной платы (руб.)		Сумма, руб.	ФСС, руб.	Удельный вес, %
	ЗП осн.	ЗП доп.			
Программист, руб.	104000	9360	113360	34008	71%
Куратор,руб.	43000	3870	46870	4061	29%
Итого	147000	13230	160230	38069	100%

Накладные расходы рассчитываются по формуле:

$$Р_Н = \frac{З_О \cdot Н_{НР}}{100\%} \quad (4.7)$$

где $Н_{НР}$ – норматив накладных расходов, (50%).

Рассчитаем затраты на накладные расходы:

$$Р_н = 104000 * 0,5 = 52000 \text{ руб.}$$

Переменные издержки составили за период разработки, указаны в таблице 4.7.

Таблица 4.7 – Переменные издержки

Вид переменных издержек	Величина, руб.	Удельный вес, %
Материальные затраты	4100	1.61%
Затраты на электроэнергию	201	0.12%
Затраты на оплату труда	160230	62.91%
Отчисления по налогам	38069	14.95%
Накладные расходы	52000	20.42%
Итого	254600	100%

Далее необходимо рассчитать общие затраты. На эту статью относятся все издержки, которые были произведены при создании настольного приложения.

Полная себестоимость разработки определяется суммированием постоянных и переменных издержек и вычисляется по формуле:

$$З_{об} = З_{пос} + З_{пер} \quad (4.10)$$

где $З_{об}$ – себестоимость приложения;

$З_{пос}$ – постоянные издержки;

$З_{пер}$ – переменные издержки.

Таким образом, себестоимость создаваемого программного модуля равна:

$$З_{об} = 4979.16 + 254600 = 259579 \text{ руб.}$$

В таблице 4.8 приведена структура полных издержек. В результате, при создании программного продукта наибольший удельный вес занимают переменные издержки.

Структура полных издержек за период разработки, представлены в таблице 4.8.

Таблица 4.8 – Структура полных издержек

Вид издержек	Величина, руб.	Удельный вес, %
Постоянные	4979.16	1.92%
Переменные	254600	98.08%
Итого	259579.16	100%

В таблице 4.9 показано процентное соотношение всех категорий затрат на разработку программного модуля.

Таблица 4.9 – Соотношение категорий затрат на разработку программного модуля

Вид затрат	Величина, руб.	Удельный вес, %
Материальные затраты	4100	1.58%
Затраты на электроэнергию	201	0.11%
Затраты на оплату труда	160230	61.7%
Отчисления по налогам	38069	14.66%
Амортизационные отчисления	3983.33	1.53%
Текущий ремонт	995.83	0.38%
Накладные расходы	52000	20.03
Итого	259579.16	100%

Для расчёта коэффициента экономической эффективности и срока окупаемости проекта внедрения информационной системы автоматической генерации учебного расписания в образовательном учреждении используются следующие данные:

– Количество сотрудников учебной части: 10 чел.

– Затраты на разработку: 259 579,16 руб.

– Предполагаемая экономия: Экономия складывается из сокращения времени на составление и корректировку расписания, уменьшения количества ошибок и повторных пересмотров.

Оценка годовой экономии:

Допустим, внедрение системы позволяет сократить время на составление расписания с 80 часов вручную до 10 часов с использованием программы.

– Снижение трудозатрат на одного сотрудника в год: 80−10=70 часов

– Средняя часовая ставка сотрудника учебной части: 400 руб./час

– Экономия на одного сотрудника в год: 70×400=28000 руб.

– Общая годовая экономия: 28000×10=280000 руб.

Срок окупаемости.

Формула:

$$Т_{ОК} = К / Э \quad (4.11)$$

где К – затраты на разработку проекта, руб.;

Э – годовая эффективность, руб.

Срок окупаемости затрат на разработку программного продукта составит:

Ток = 259579.16 / 280000 = 0,93 года = 11 месяцев

Коэффициент экономической эффективности.

Формула:

$$Е_ф = Э / К \quad (4.12)$$

Нормативное значение коэффициента эффективности капитальных вложений Ен = 0,33, если Еф > Ен, то делается вывод об эффективности капитальных вложений.

Коэффициент экономической эффективности разработки (Еф) равен:

Еф = 280000 / 259579.16 = 1.08

Так как *Еф* = 1.08 > *Ен,* то разработка и внедрение разрабатываемого продукта является эффективным, т.е. эффект от использования настольного приложения окупает все затраты, связанные с проектированием и эксплуатацией.

В таблице 4.10 приведены сводные данные экономического обоснования разработки и внедрения проекта.

Таблица 4.10 – Таблица показателей экономической эффективности

Показатель	Величина
Трудозатраты на разработку проекта, чел*час	320
Затраты на разработку проекта, руб.	259579.16
Экономическая эффективность проекта, руб.	280000
Коэффициент экономической эффективности	1.08
Срок окупаемости, мес.	11

Таким образом, произведенный экономический анализ эффективности создания и эксплуатации приложения доказывает целесообразность его использования в организации.

Заключение

В период разработки дипломного проекта была создана информационная система, направленная на решение актуальной проблемы организации учебного процесса в условиях современных образовательных учреждений, характеризующихся высокой концентрацией обучающихся и преподавательского состава, а также ограниченными ресурсами.

При проектировании и разработке информационной системы были решены следующие задачи:

– изучены текущие процессы организации учебного расписания в образовательных учреждениях, выявлены основные проблемы и ограничения;

– определен необходимый набор функций и возможностей системы, включающих в себя удобный интерфейс для пользователей;

– спроектирована реляционная база данных, определена структура таблиц и связи между ними, обеспечивающая хранение данных о пользователях;

– разработаны и внедрены модули для автоматической генерации расписания, учета ограничений и экспорта данных через API;

– выполнены тесты для оценки производительности и надежности системы, а также для проверки корректности работы всех основных функций.

В процессе разработки информационной системы были применены современные технологии и инструменты, такие как: TypeScript, Node.js, NestJS, NextJS, Prisma, Tailwind, Jotai, а также среда разработки VS Code .

В заключение можно сделать вывод, что данный дипломный проект продемонстрировал возможность эффективного решения практических задач с использованием современных технологий. Разработанная система может быть внедрена в эксплуатацию и способствовать повышению качества организации учебного процесса, минимизации ошибок и создания более комфортной среды в образовательных учреждениях. В дальнейшем планируется доработка системы и реализация дополнительных функций.

Список используемых источников

1. Ананьева, Т. Н. Стандартизация, сертификация и управление качеством программного обеспечения: учебное пособие / Т.Н. Ананьева, Н.Г. Новикова, Г.Н. Исаев. — Москва: ИНФРА-М, 2021. — 232 с. — (Среднее профессиональное образование). — ISBN 978-5-16-014887-8. — Текст: электронный. — URL: https://znanium.com/catalog/product/1413308 – Режим доступа: по подписке.

2. Варфоломеева, А. О. Информационные системы предприятия: учебное пособие / А.О. Варфоломеева, А.В. Коряковский, В.П. Романов. — 2-е изд., перераб. и доп. — Москва: ИНФРА-М, 2023. — 330 с. — (Среднее профессиональное образование). — ISBN 978-5-16-014729-1. — Текст: электронный. — URL: https://znanium.ru/catalog/document?id=425518 – Режим доступа: по подписке.

3. Волк, В. К. Базы данных. Проектирование, программирование, управление и администрирование / В.К. Волк. — 2-е изд., стер. — Санкт-Петербург: Лань, 2022. — 340 с. — ISBN 978-5-8114-9682-2. — Текст: электронный // Лань. — URL: https://e.lanbook.com/book/198584 — Режим доступа: для авториз. пользователей.

4. Гвоздева, Т. В. Проектирование информационных систем. Основы управления проектами. Лабораторный практикум / Т.В. Гвоздева, Б.А. Баллод. — 2-е изд., стер. — Санкт-Петербург: Лань, 2022. — 120 с. — ISBN 978-5-507-44958-3. — Текст: электронный // Лань. — URL: https://e.lanbook.com/book/250811 — Режим доступа: для авториз. пользователей.

5. Гилязова, Р. Н. Информационная безопасность. Лабораторный практикум: учебное пособие для спо / Р.Н. Гилязова. — 3-е изд., стер. — Санкт-Петербург: Лань, 2022. — 44 с. — ISBN 978-5-8114-9138-4. — Текст: электронный // Лань. — URL: https://e.lanbook.com/book/187645 — Режим доступа: для авториз. пользователей.

6. Голицына, О. Л. Информационные системы: учебное пособие / О.Л. Голицына, Н.В. Максимов, И.И. Попов. — 2-е изд. — Москва: ФОРУМ: ИНФРА-М, 2022. — 448 с. — ISBN 978-5-91134-833-5. — Текст: электронный. — URL: https://znanium.com/catalog/product/1832410 – Режим доступа: по подписке.

7. Жуков, Р. А. Язык программирования Python. Практикум / Москва: ДМК Пресс, 2024. — 216 с. — ISBN 978-5-16-015638-5. — Текст: электронный. — URL: https://znanium.ru/catalog/document?id=439174 – Режим доступа: по подписке.

8. Игнатьев, А. В. Тестирование программного обеспечения / А.В. Игнатьев. — 3-е изд., стер. — Санкт-Петербург: Лань, 2023. — 56 с. — ISBN 978-5-507-45426-6. — Текст: электронный // Лань. — URL: https://e.lanbook.com/book/269876 — Режим доступа: для авториз. пользователей.

9. Мартишин, С. А. Базы данных: Работа с распределенными базами данных и файловыми системами на примере MongoDB и HDFS с использованием Node.js, Express.js, Apache Spark и Scala. Учебное пособие / Москва: ИНФРА-М, 2023. — 235 с. — ISBN 978-5-16-015643-9. — Текст: электронный. — URL: https://znanium.ru/catalog/document?id=435831 – Режим доступа: по подписке.

10. Миронов, А. И. Тестирование и верификация программного обеспечения: Практикум: учебное пособие / А.И. Миронов, С.М. Трушин, А.А. Петренко. — Москва: РТУ МИРЭА, 2022. — 65 с. — Текст: электронный // Лань. — URL: https://e.lanbook.com/book/240095 — Режим доступа: для авториз. пользователей.

11. Митяков, Е. С. Искусственный интеллект и машинное обучение: учебное пособие для СПО / Е.С. Митяков, А.Г. Шмелева, А.И. Ладынин. — Санкт-Петербург: Лань, 2025. — 252 с. — ISBN 978-5-507-51466-3. — Текст: электронный // Лань. — URL: https://e.lanbook.com/book/450830 — Режим доступа: для авториз. пользователей.

12. Никифоров, С. Н. Методы защиты информации. Пароли, скрытие, шифрование: учебное пособие для СПО / С.Н. Никифоров. — 2-е изд., стер. — Санкт-Петербург: Лань, 2021. — 124 с. — ISBN 978-5-8114-8256-6. — Текст:

электронный // Лань. — URL: https://e.lanbook.com/book/173803 — Режим доступа: для авториз. пользователей.

13. Петренко, В. И. Защита персональных данных в информационных системах. Практикум: учебное пособие для СПО / В.И. Петренко, И.В. Мандрица. — 2-е изд., стер. — Санкт-Петербург: Лань, 2022. — 108 с. — ISBN 978-5-8114-9038-7. — Текст: электронный // Лань. — URL: https://e.lanbook.com/book/183744 — Режим доступа: для авториз. пользователей.

14. Полищук, Ю. В. Базы данных и их безопасность : учебное пособие / Ю.В. Полищук, А.С. Боровский. — Москва: ИНФРА-М, 2022. — 210 с. — ISBN 978-5-16-016151-8. — Текст: электронный. — URL: https://znanium.com/catalog/product/1864071 – Режим доступа: по подписке.

15. Рогов, Е. В. PostgreSQL 15 изнутри / Е.В. Рогов. — Москва: ДМК Пресс, 2023. — 664 с. — ISBN 978-5-93700-178-8. — Текст: электронный. — URL: https://znanium.ru/catalog/product/2150531 – Режим доступа: по подписке.

16. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: учебное пособие / Г.Н. Федорова. — Москва: КУРС: ИНФРА-М, 2022. — 336 с. — ISBN 978-5-906818-41-6. — Текст: электронный. — URL: https://znanium.com/catalog/product/1858587 – Режим доступа: по подписке.

17. Черников, Б. В. Управление качеством программного обеспечения: учебник / Б.В. Черников. — Москва: ФОРУМ: ИНФРА-М, 2022. — 240 с. — ISBN 978-5-8199-0902-7. — Текст: электронный. — URL: https://znanium.com/catalog/product/1850732 – Режим доступа: по подписке.

18. Шитов, В. Н. Обработка отраслевой информации: учебное пособие / В.Н. Шитов. — Москва: ИНФРА-М, 2022. — 184 с. — DOI 10.12737/1846131. — ISBN 978-5-16-017373-3. — Текст: электронный. — URL: https://znanium.com/catalog/product/1846131 – Режим доступа: по подписке.

Приложение А – Инструкция пользователя и программиста

Инструкция пользователя

Чтобы создать учетную запись пользователя нужно нажать на кнопку «Регистрация», если учетная запись уже есть нажмите на кнопку «Авторизация».

На рисунке А1 представлена главная страница.

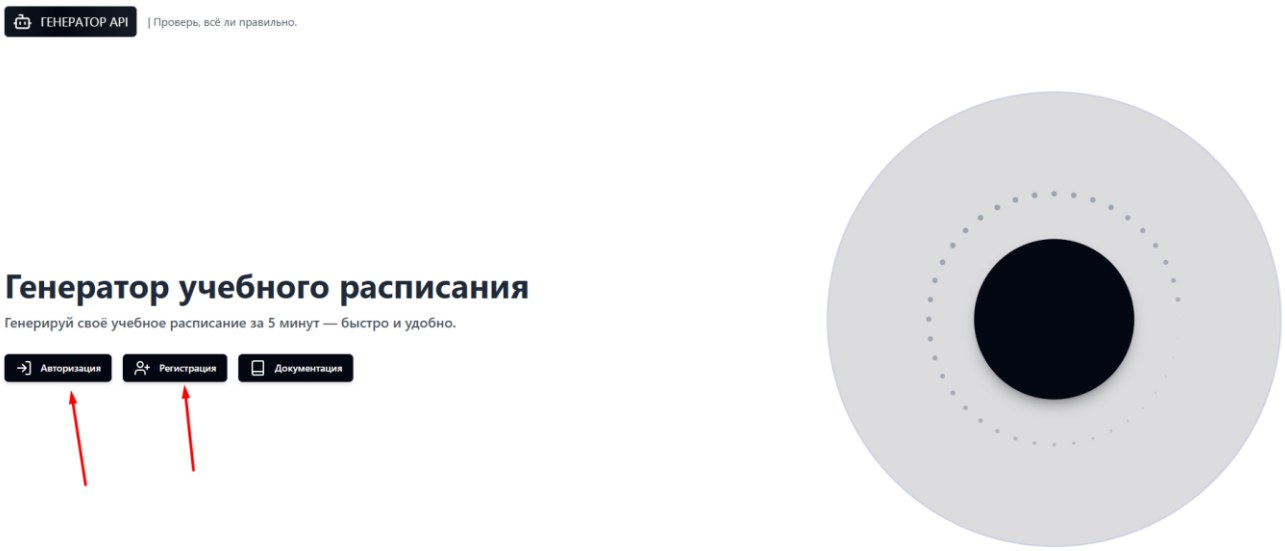

Рисунок А1 – Главная страница

На странице авторизации заполняются данные, что использовали при регистрации, а именно логин и пароль, затем следует нажать на кнопку «Войти».

На рисунке А2 представлена страница авторизации.

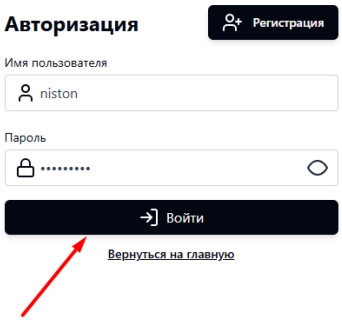

Рисунок А2 – Страница авторизации

На странице регистрации нужно заполнить все необходимые поля, нажать на флажок «Я согласен с условиями использования» и затем на кнопку «Зарегистрироваться».

На рисунке А3 представлена страница регистрации.

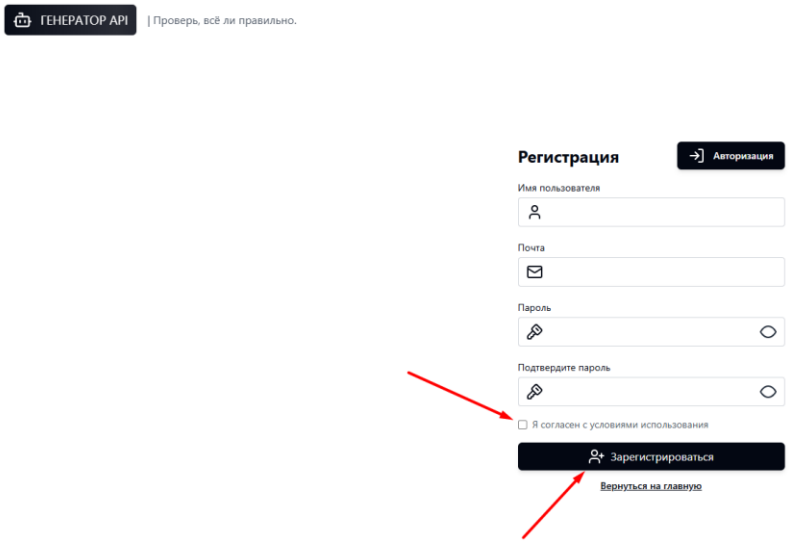

Рисунок А3 – Страница регистрации

После успешной регистрации и авторизации происходит перенаправление на профиль. На главной странице профиля пользователя встречает форма, первые три

поля которой отвечают за добавление в свое расписание ключевых параметров: кабинеты, группы и преподаватели. После ввода необходимых данных пользователь должен нажать на кнопку «Добавить», чтобы сохранить выбранные параметры.

На рисунке А4 представлены поля кабинеты, группы и преподаватели.

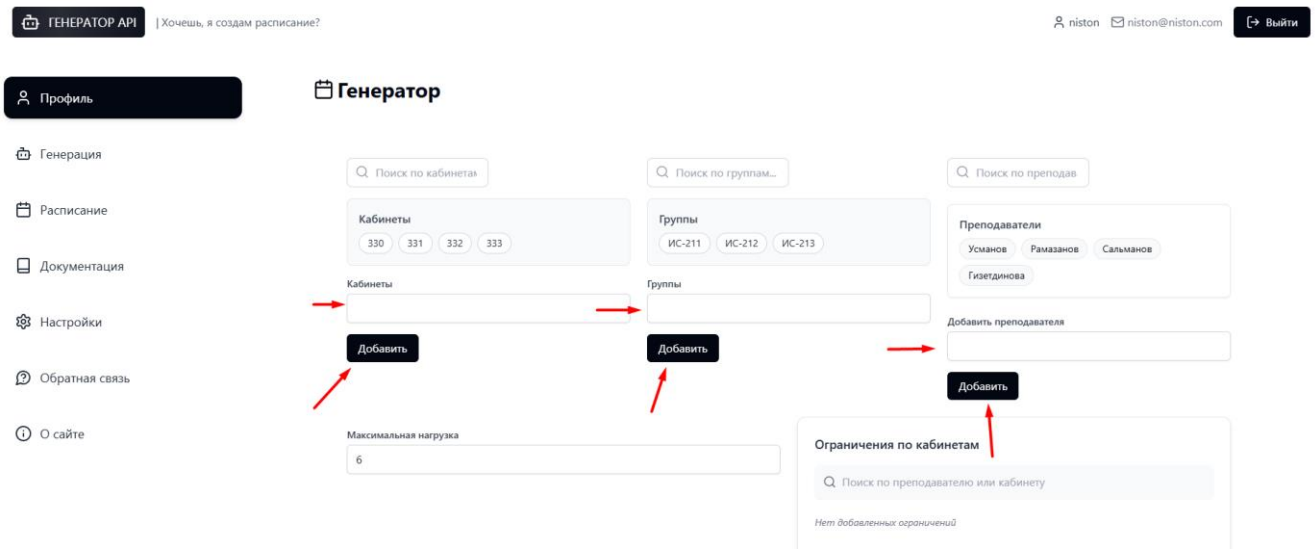

Рисунок А4 – Поля кабинеты, группы и преподаватели

После добавления кабинетов, групп и преподавателей пользователь задает максимальную нагрузку (количество пар в день для каждой группы) и ограничения по кабинетам, выбирая преподавателя и указывая кабинет; если ограничения не указаны, занятия могут проходить в любом доступном кабинете.

На рисунке А5 представлены поля максимальной нагрузки и форма ограничений по кабинетам.

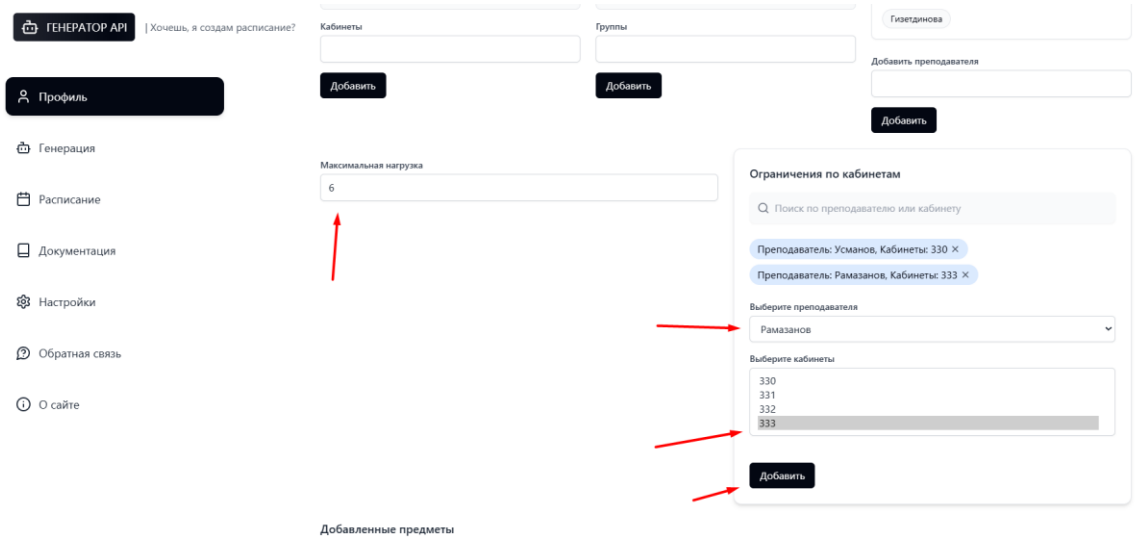

Рисунок А5 – Поля максимальной нагрузки и форма ограничений по кабинетам

Далее добавляем предмет, указываем его название, выбираем преподавателя, вводим количество часов для лекций и при необходимости задаем часы для первой и второй подгрупп, после чего нажимаем кнопку «Добавить предмет».

На рисунке А6 представлена форма добавления предмета.

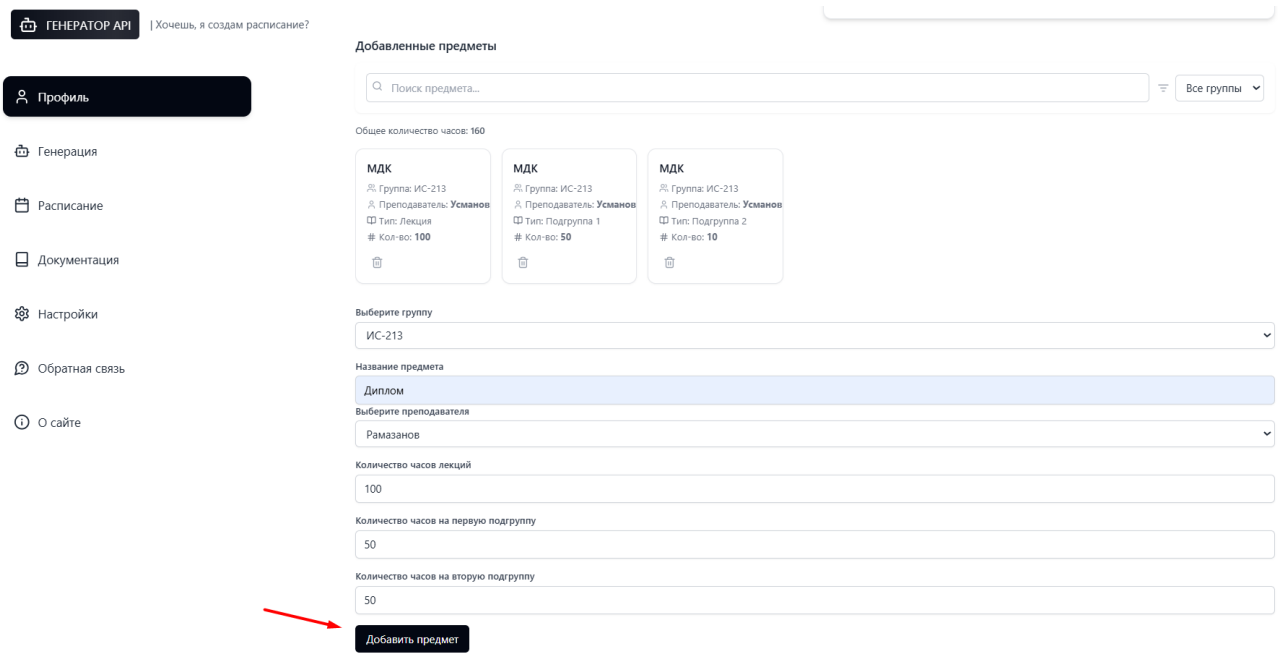

Рисунок А6 – Форма добавления предмета

В блоке «Исключить дни недели» выделяются дни, в которые расписание генерироваться не будет, выбор дней имеет три состояния: первое состояние - режим дней, где при зажатом Shift и правой кнопке мыши можно выбрать несколько дней, второе состояние - дни недели, где пользователь выбирает дни недели для определенного месяца, третье состояние - месяцы, где пользователь выбирает месяц для генерации расписания, также доступны две кнопки: «Сгенерировать» - перегенерирует выделенное расписание на основе выбранных параметров и «Создать пустое расписание» - создает новое расписание с нуля.

На рисунке А7 изображен режим дня.

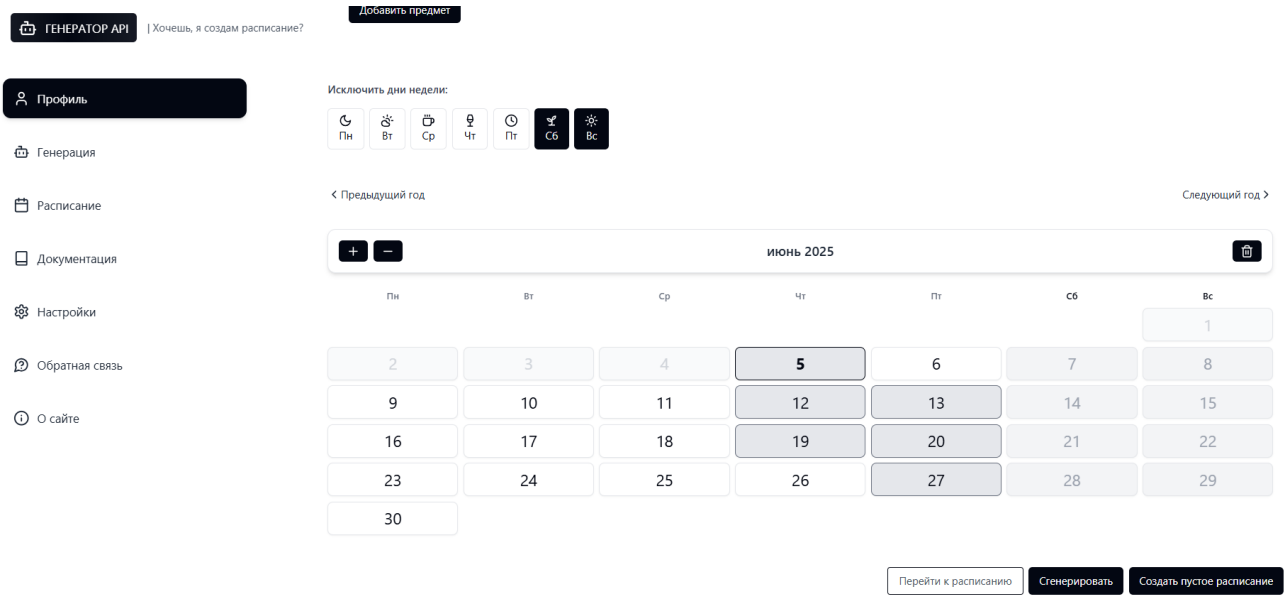

Рисунок А7 – Режим дня

На рисунке А8 изображен режим недели.

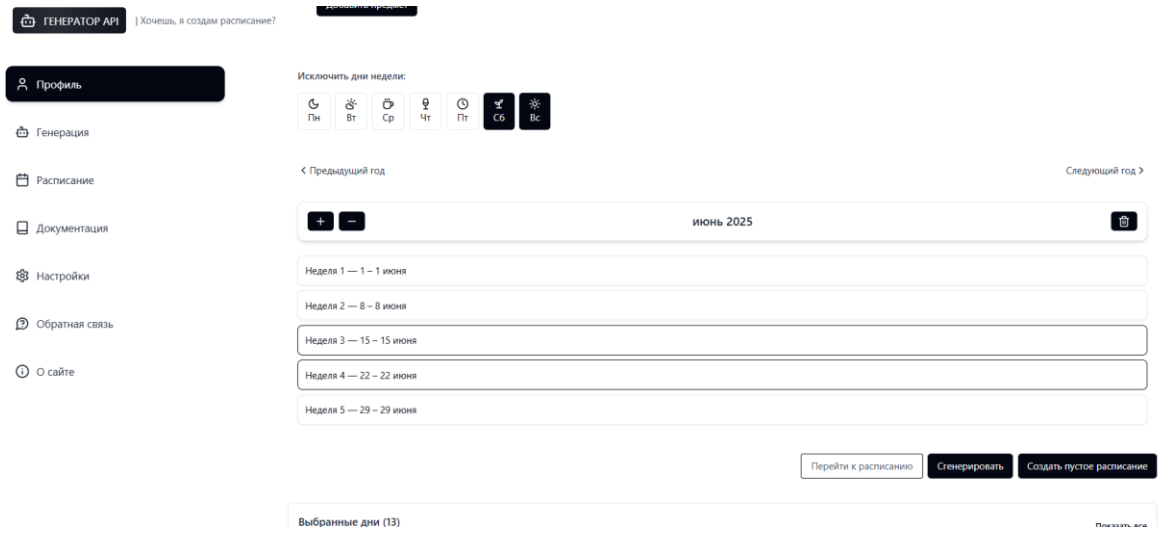

Рисунок А8 – Режим недели

На рисунке А9 изображен режим месяца.

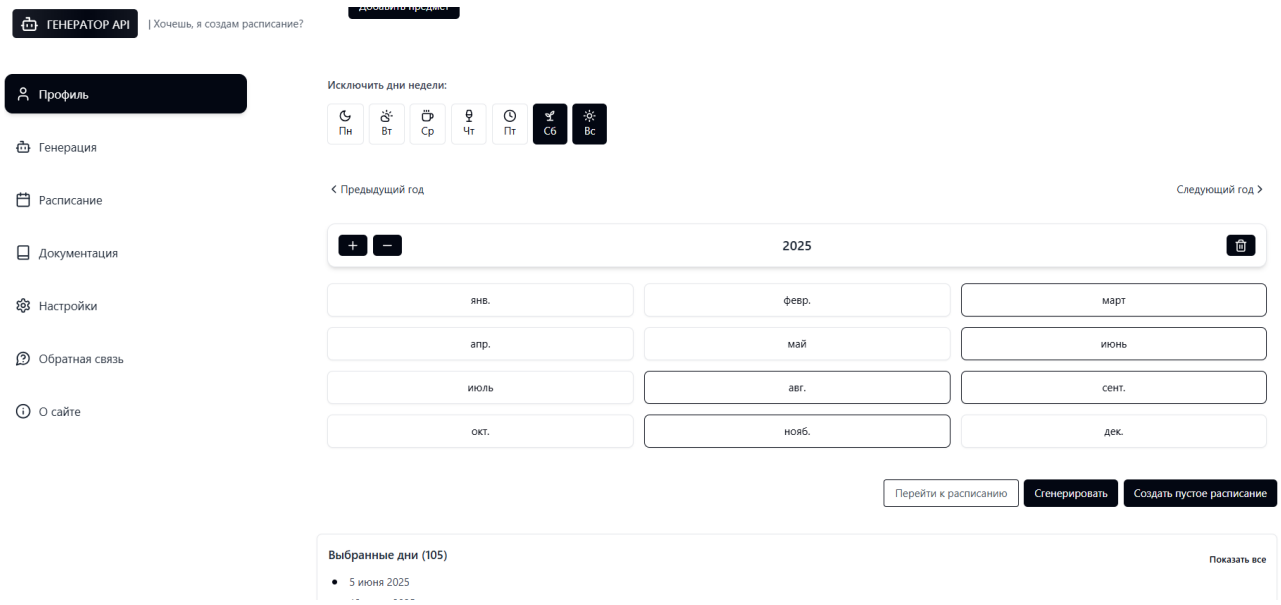

Рисунок А9 – Режим месяца

После успешной генерации расписания появляется уведомление о том, что генерация прошла успешно, затем пользователь переходит на страницу «Расписание», где может выбрать свое сгенерированное расписание для просмотра или дальнейшего редактирования.

На рисунке А10 изображена успешная генерация расписания.

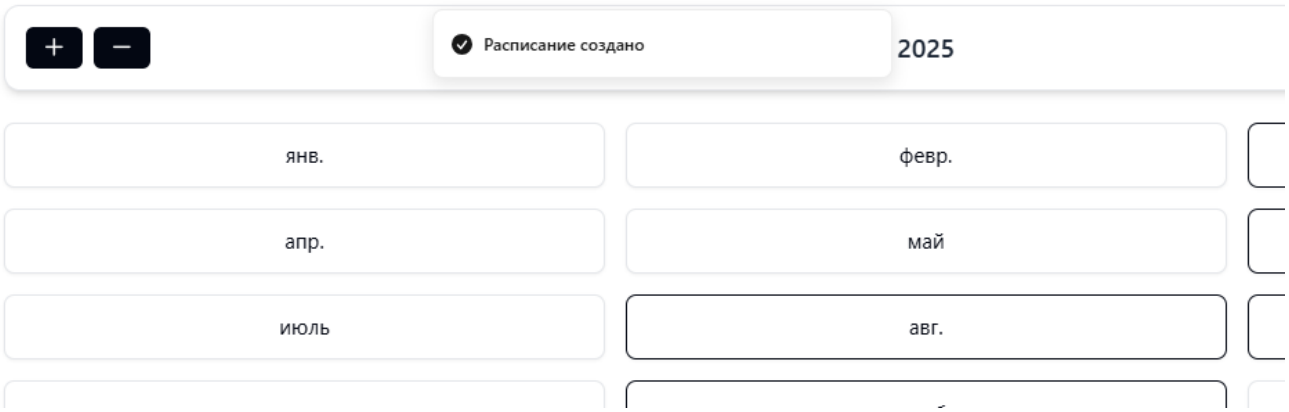

Рисунок А10 – Уведомление об успешной генерации расписания

На рисунке А11 изображена страница «Расписание».

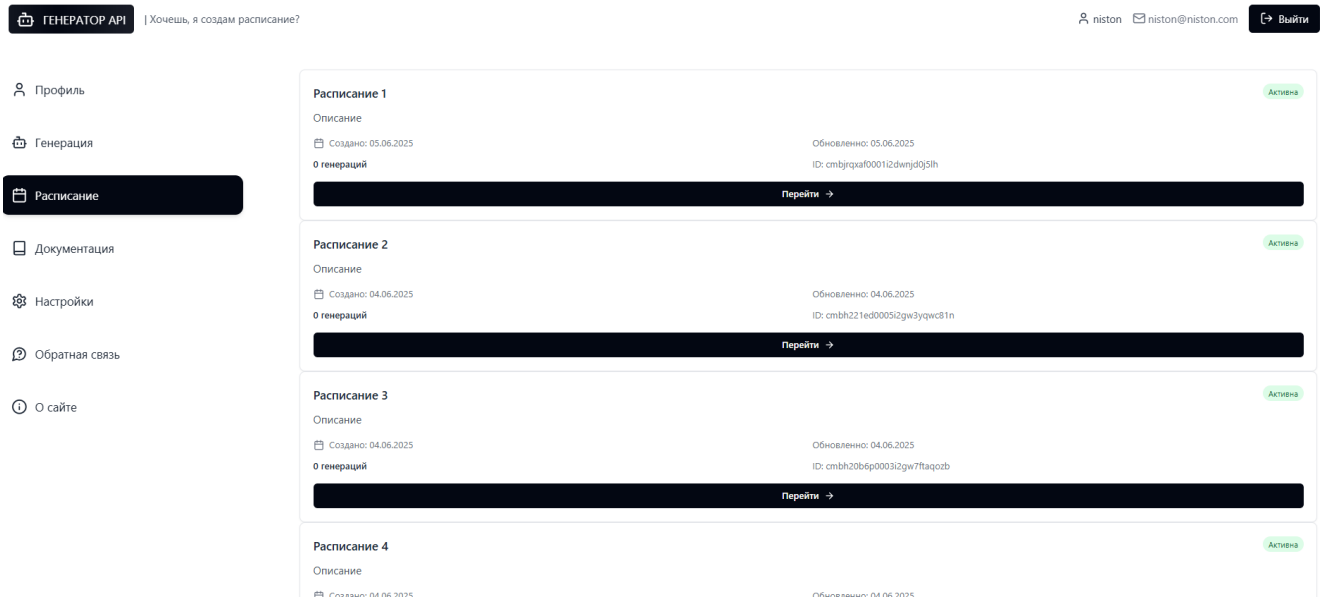

Рисунок А11 – Страница «Расписание»

При переходе на сгенерированное расписание отображается статистика, включающая дату создания и обновления расписания, личный идентификатор, статус, общее количество часов, количество дней, на которые было сгенерировано расписание, количество предметов, групп и кабинетов. При необходимости можно

получить более подробную информацию, например, при нажатии на данные о количестве кабинетов отображается их полный список.

На рисунке А12 изображено сгенерированное расписание.

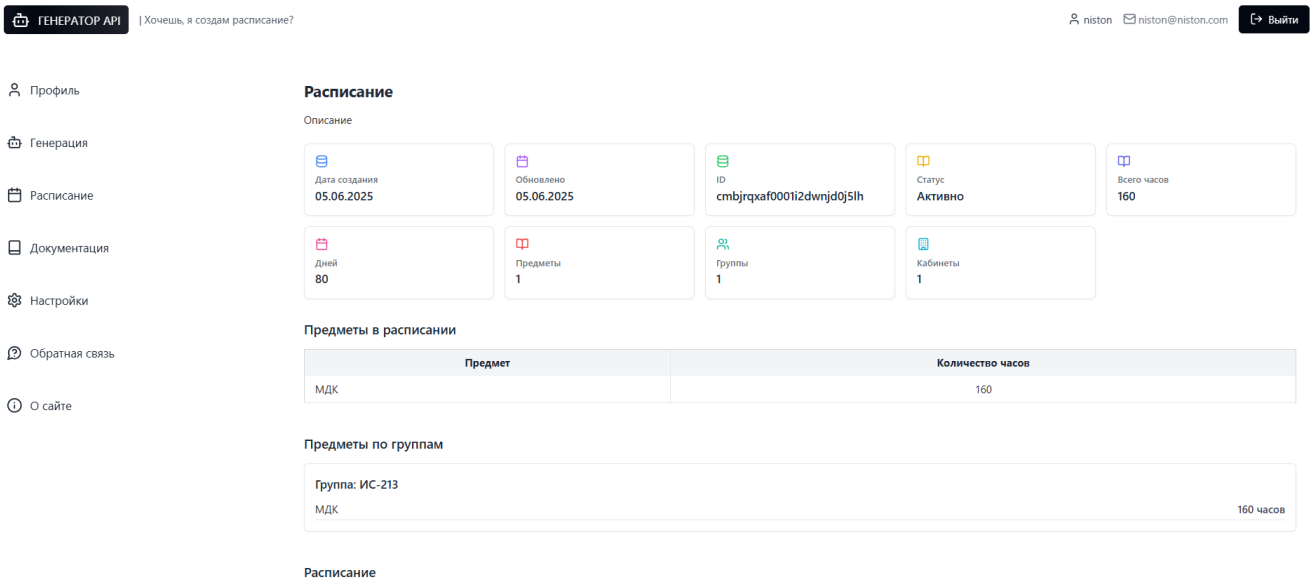

Рисунок А12 – Сгенерированное расписание

На рисунке А13 изображены расширенные данные о расписании.

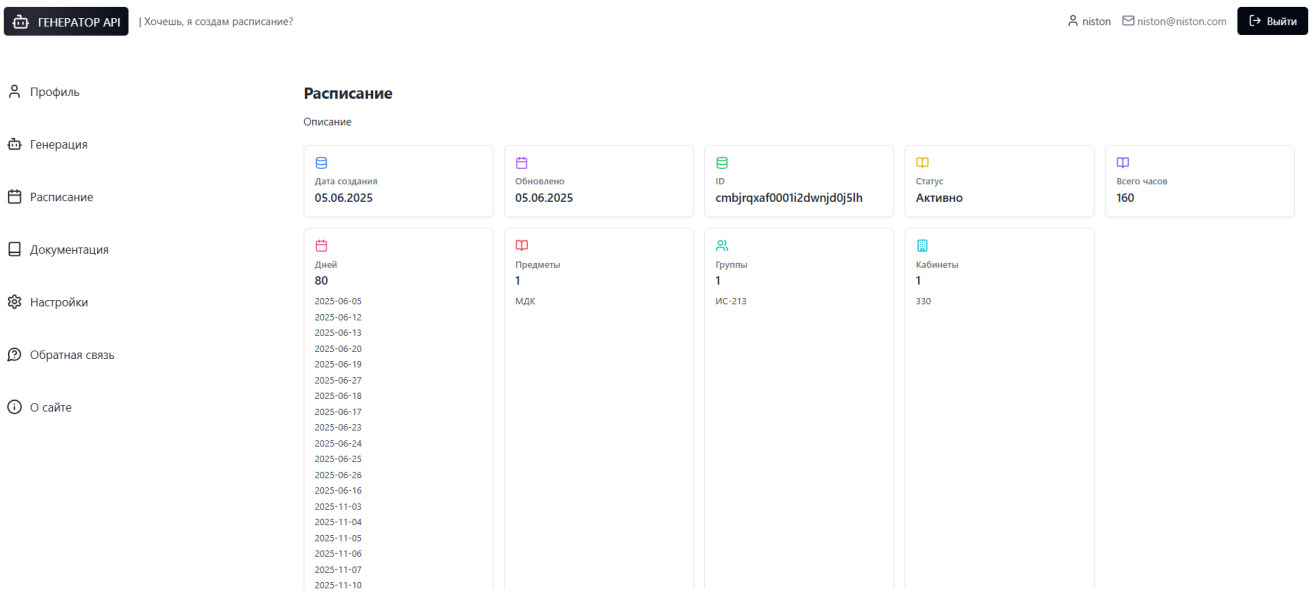

Рисунок А13 – Расширенные данные о расписание

На странице «Обратная связь» можно задать любой вопрос, заполнив поля «Название» и «Сообщение». После ввода текста необходимо нажать на кнопку «Отправить», чтобы отправить ваш запрос в техподдержку.

На рисунке А14 изображена страница «Обратная связь».

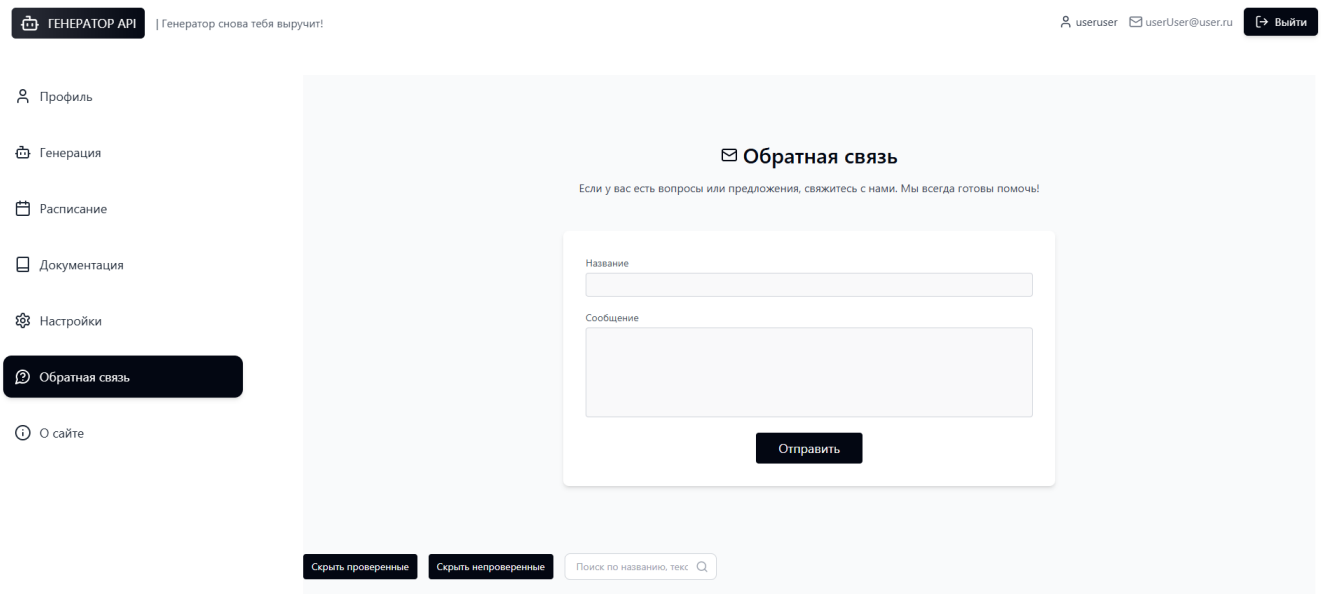

Рисунок А14 – Страница «Обратная связь»

Инструкция программиста

Перейти в папку server

```
cd ./server
```

Установите пакеты

```
npm i
```

Примените миграции базы данных с помощью Prisma

```
npx prisma db push
```

Создайте миграции для обновления структуры базы данных

```
npx prisma generate
```

Запустите сервер

```
npm run start:dev
```

После запуска сервер будет доступен по адресу

```
http://localhost:5555
```

Перейти в папку client

```
cd ./client
```

Установите пакеты

```
npm i
```

Запустите сервер

```
npm run dev
```

После ввода данной строчки в терминале, следует открыть в браузере запущенный локальный сервер, для этого в адресной строке нужно написать http://localhost:3000

Результатом будет, то что изображено на рисунке А15.

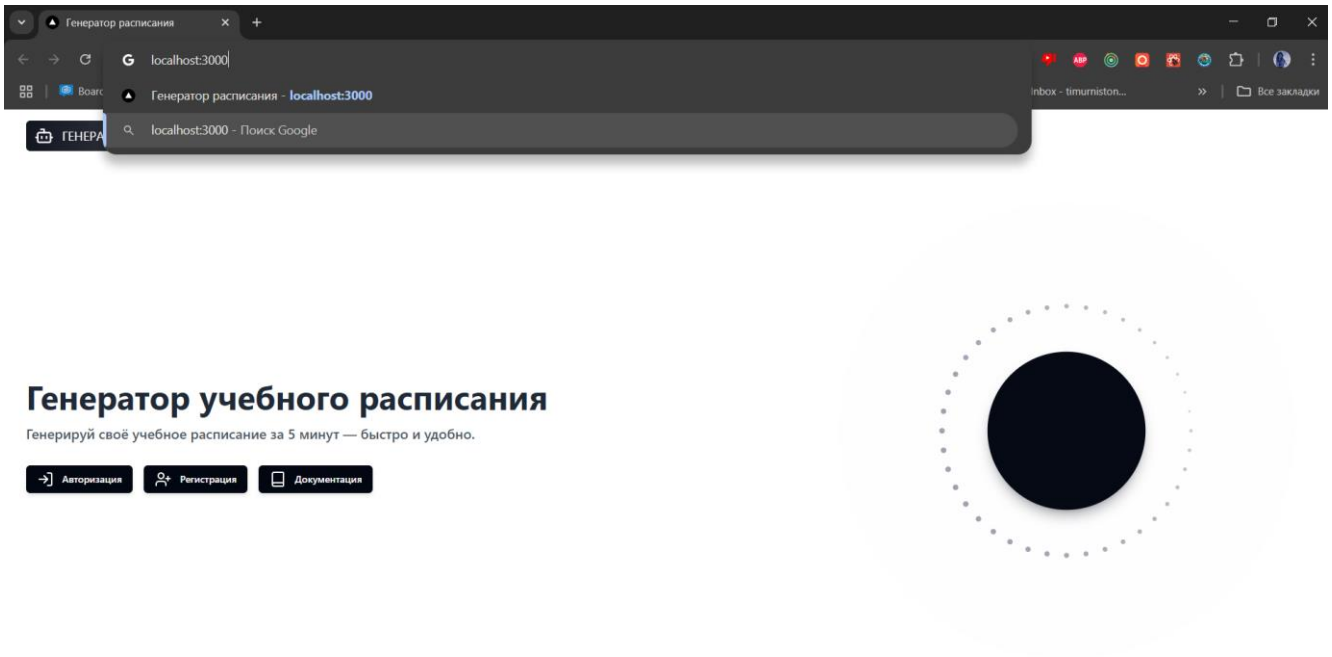

Рисунок А15 – Главная страница ИС

Далее нам необходимо зайти под учетной записью администратора. Ниже будут данные для входа:

– Логин: admin;

– Пароль: adminA;

На рисунке А16 изображена страница «Админ панель».

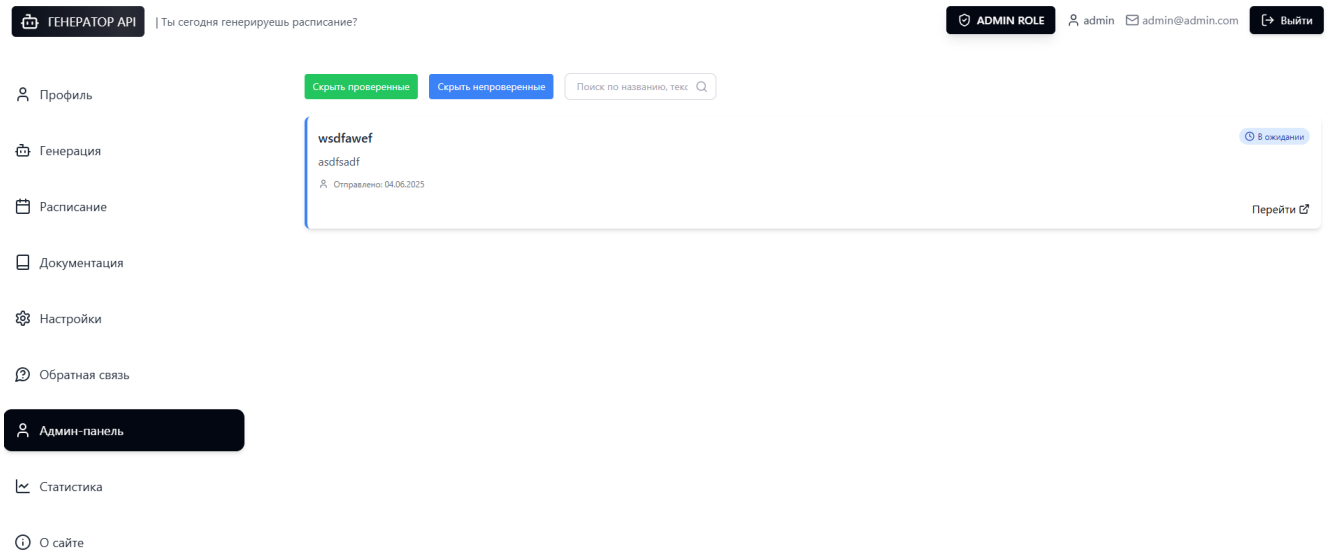

Рисунок А16 – Страница «Админ-панель»

На рисунке А17 изображена страница «Статистика».

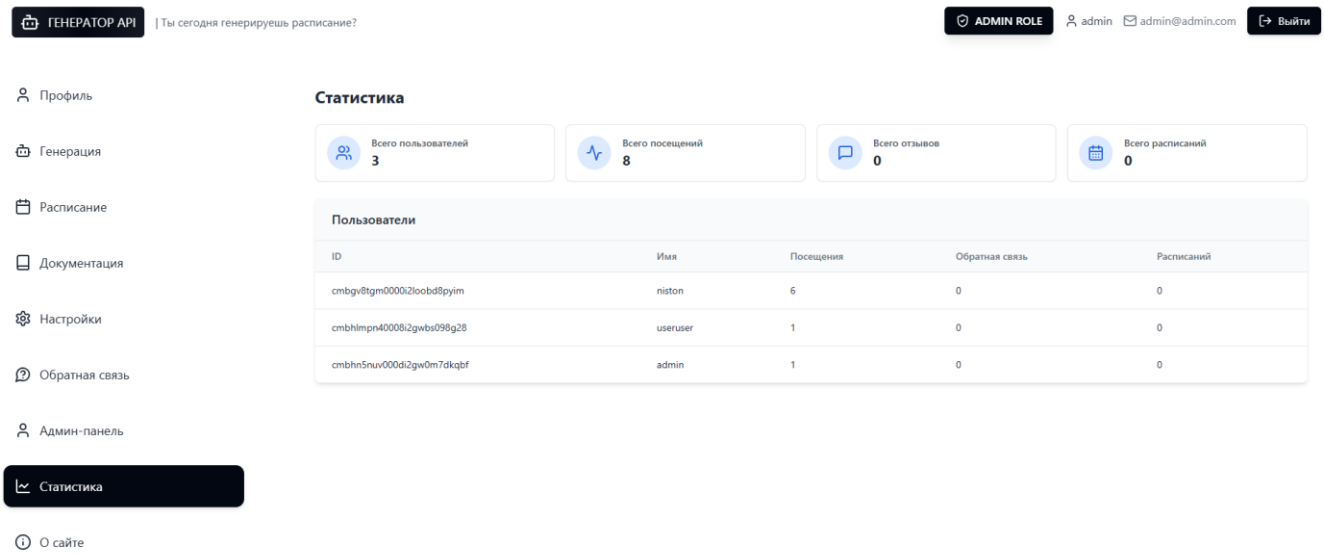

Рисунок А17 – Страница «Статистика»

Приложение Б – Электронный формат пояснительной записки и информационной системы