Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский экономический университет имени Г.В. Плеханова»
Московский приборостроительный техникум

Дипломный проект (работа)

На тему: Разработка программного комплекса генерации и управления аудиторным фондом (на примере ФГБОУ ВО «РЭУ им. Г.В. Плеханова»).

Максименковой Мелисы Сергеевны
Студентка 4 курса группы П50-7-21

по специальности 09.02.07 «Информационные системы и программирование»
для присвоения квалификации: программист
Форма обучения: очная

Руководитель: _____ / Шимбирёв Андрей Андреевич /
(подпись)
«_____»_____2025 г.

Студентка: _____ / Максименкова Мелиса Сергеевна /
(подпись)
«_____»_____2025 г.

Допущена к защите
Приказ от «_____» _____ 2025 г. № _____

2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последние годы информационные технологии стали неотъемлемой частью образовательного процесса, оказывая существенное влияние на организацию и управление учебной деятельностью. Информационные системы обеспечивают повышение эффективности и качества предоставляемых услуг, что особенно важно в сферах, требующих точности и оперативности. Колледжи и университеты, являясь ключевыми учреждениями системы образования, активно внедряют программные решения для оптимизации процессов планирования и использования учебных ресурсов.

В образовательных учреждениях ежедневно проводится множество занятий, что требует чёткого учёта и рационального распределения ресурсов: аудиторий, студенческих групп, преподавателей и видов учебной активности. Традиционные методы, основанные на ручной обработке расписания, становятся неэффективными в условиях высокой плотности учебного графика, множества направлений подготовки и постоянно изменяющихся условий. Это приводит к накладкам, ошибкам и нерациональному использованию аудиторного фонда, что в свою очередь снижает качество учебного процесса.

Особенно остро эти проблемы проявляются в колледжах, где обучение осуществляется по множеству специальностей и профилей, а расписание формируется с учётом практик, мероприятий, подгрупп и дистанционных занятий. В таких условиях составление и поддержание актуального аудиторного фонда вручную требует значительных временных затрат и сопровождается высоким риском ошибок.

В целях повышения точности, скорости и прозрачности распределения ресурсов в данной работе реализована специализированная информационная система, автоматизирующая процесс формирования аудиторного фонда. В рамках дипломного проекта разработано десктопное приложение с графическим

интерфейсом на базе технологии WPF (Windows Presentation Foundation) и языка программирования C#. Приложение интегрируется с серверным API, реализованным на ASP.NET Core, и взаимодействует с базой данных Microsoft SQL Server.

Разрабатываемая система включает следующие ключевые функции: импорт расписания с сайта колледжа с учётом изменений, автоматическое распределение преподавателей по свободным кабинетам с учётом их предпочтений и закреплений, учёт учебных практик, генерация Excel-файлов аудиторного фонда, а также поддержка фильтрации по дню недели, корпусу и типу недели. Дополнительно реализован Telegram-бот, предоставляющий студентам и преподавателям доступ к актуальной информации по аудиториям, расписанию и загруженности кабинетов.

Актуальность разработки обусловлена необходимостью автоматизации процесса составления и сопровождения аудиторного фонда, сокращения временных затрат, минимизации ошибок и повышения эффективности использования учебных помещений. Внедрение такой системы позволит упростить работу сотрудников учебного отдела, повысить прозрачность распределения ресурсов и обеспечить более стабильную организацию образовательного процесса.

1. ОБЩАЯ ЧАСТЬ

1.1. Цель разработки

Цель разработки приложения для генерации и управления аудиторным фондом заключается в упрощении процесса планирования учебного процесса, автоматизации составления аудиторного фонда и улучшении организации занятий в учебном заведении.

1.2. Средства разработки

Для проектирования, разработки и тестирования программного комплекса, реализующего свой функционал на базе платформы Windows, были использованы программные средства, представленные в таблице 1.

Таблица 1 – Программные средства разработки ПО

№	Тип средства	Название средства	Назначение
1	2	3	4
1	Операционная система	Microsoft Windows 11 Pro 23H2	Организация взаимодействия программ и пользователя
2	Инструментальные средства разработки программного решения	Microsoft Visual Studio 2022 Community 17.9.6	Разработка десктопного приложения и API
3	Графическая оболочка	SQL Server Management Studio 20	Разработка базы данных
4	СУБД	Microsoft SQL Server (64-bit)	Разработка базы данных
5	Текстовый редактор	Microsoft Office стандартный 2021	Написание Документации
6	Среда разработки на Python	Visual Studio Code 1.88.0	Разработка Telegram-бота
7	Средство тестирования API	Postman 10.25.2	Тестирование REST-запросов
8	Веб-браузер	Google Chrome ver. 135.0.7049.115	Проверка структуры HTML-страниц и результатов парсинга

В таблице 2 представлены минимальные и рекомендованные технические средства, на базе которых возможно комфортное использование реализуемого программного комплекса.

Таблица 2 – Минимальные и рекомендованные технические средства

№	Тип оборудования	Наименование оборудования
1	2	3
Десктопное приложение		

№	Тип оборудования	Наименование оборудования
1	2	3
Рекомендуемые системные требования		
1	Разрядность	Amd64
2	Процессор	Intel Core i5 и выше
3	Видеопамять	512 МБ
4	Оперативная память	8 ГБ
5	Свободное место на диске	от 500 МБ (под .NET + логи/кэш)
Минимальные системные требования		
1	Рязрядность	Amd64
2	Процессор	Intel Core i3
3	Видеопамять	128 МБ
4	Оперативная память	4 ГБ
Telegram-бот (серверная часть)		
1	Разрядность	Amd64
2	Процессор	Intel Core i3 и выше
3	Оперативная память	от 2 ГБ
4	Свободное место на диске	от 200 МБ
5	Подключение к сети	Стабильное интернет-соединение
6	Операционная система	Linux / Windows Server
7	Интерпретатор Python	Python 3.10 и выше

В качестве средств вычислительной техники при разработке ПО и использования программного комплекса использовался ноутбук Asus Tuf Gaming F15 FX506HC-HN006. Характеристики представлены в таблице 3.

Таблица 3 – Технические характеристики ноутбука при разработке ПО

№	Тип средства	Название средства
1	2	3
Для разработки		
Ноутбук Asus Tuf Gaming F15 FX506HC-HN006		
1	Размер экрана:	15,6
2	Разрешение экрана:	1920x1080
3	Линейка процессора:	Intel Core i5-11400H
4	Количество ядер процессора:	6
5	Оперативная память:	16 ГБ
6	Тип видеокарты:	Дискретная
7	Видеокарта:	GeForce RTX 3050
8	Конфигурация накопителей:	SSD
9	Общий объём всех накопителей:	1168
	Для использования	
	Ноутбук Asus Tuf Gaming F15 FX506HC-HN006	
1	Размер экрана:	15,6
2	Разрешение экрана:	1920x1080
3	Линейка процессора:	Intel Core i5-11400H
4	Количество ядер процессора:	6
5	Оперативная память:	16 ГБ

№	Тип средства	Название средства
1	2	3
6	Тип видеокарты:	Дискретная
7	Видеокарта:	GeForce RTX 3050
8	Конфигурация накопителей:	SSD
9	Общий объём всех накопителей:	1168
1	Размер экрана:	15,6

2. СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1. Постановка задачи

Разработать программный комплекс для генерации и сопровождения аудиторного фонда учебного заведения. Комплекс должен предоставлять пользователю эффективные инструменты для планирования и контроля использования учебных помещений, автоматизации задач, связанных с составлением, обновлением и распространением актуального аудиторного фонда, а также удобный доступ к информации через мессенджер.

Основная цель программного комплекса — автоматизация и оптимизация всех этапов работы с аудиторным фондом: от сбора и анализа расписания до распределения аудиторий и предоставления информации преподавателям и студентам. В результате система позволяет повысить точность планирования, сократить трудозатраты сотрудников и обеспечить более прозрачную и рациональную организацию учебного процесса.

2.1.1. Входные данные

Входные данные программного комплекса представлены в следующем виде:

- Файл расписания: JSON-файл, получаемый из встроенного парсера, содержащий информацию о группах, преподавателях, днях недели, парах, корпусах и типах занятий;
- Данные об изменениях в расписании: день недели, номер пары, группа, преподаватель, описание изменения (отмена, перенос, замена и т. п.); загружаются из дополнительного JSON-файла, полученного с сайта учебного заведения;
- Данные о распределении преподавателя на аудиторию в формате JSON файла;

- Шаблон Excel-файла: структура для формирования финального файла аудиторного фонда;
- Аутентификационные данные пользователя: код доступа, используемый для входа в систему;
- Данные о группах: название, курс, сокращение, квалификация, год поступления, диапазоны ячеек в шаблоне Excel;
- Данные о преподавателях: фамилия, имя, отчество, список закреплённых аудиторий, предпочтения по этажам;
- Данные об аудиториях: номер кабинета, корпус (Нахимовский, Нежинская, Дистанционно), статус (свободна/занята);
- Данные о квалификациях: код и полное наименование специальности;
- Данные о закреплениях: ID преподавателя, ID аудитории, корпус;
- Данные о мероприятиях: дата, аудитория, название события, диапазон пар;
- Данные об учебной практике: дата, группа, преподаватель, аудитория, диапазон пар;
- Тип недели: числитель или знаменатель.

2.1.2. Выходные данные

Выходные данные программного комплекса представлены в следующем виде:

- Сгенерированный Excel-файл аудиторного фонда: содержит информацию о группах, квалификациях, преподавателях, аудиториях, днях недели, типах занятий, месте проведения занятий (корпус), типе недели и дате;
- Информация о закреплённых аудиториях: номер аудитории, корпус, преподаватель, за которым она закреплена;

- Отображение преподавателей: список всех преподавателей в административном интерфейсе, с возможностью фильтрации, редактирования и удаления;
- Отображение аудиторий: список аудиторий с указанием корпуса, поддержкой операций добавления, изменения и удаления;
- Вывод учебной практики: отображение даты, группы, преподавателя, кабинета и диапазона пар в специальной вкладке интерфейса;
- Вывод мероприятий: отображение даты, названия события, аудитории и диапазона пар в разделе событий;
- Распределение по корпусам: обработанные данные из расписания, распределённые по корпусам (Нахимовский, Нежинская, Дистанционно) с учётом типа недели и текущей даты;
- Информация о типе недели: числитель или знаменатель, отображаемая в интерфейсе и используемая при генерации фонда;
- Загруженные данные: визуальное отображение загруженного JSON-файла расписания с возможностью проверки структуры и целостности данных;
- Обновлённые данные после действий пользователя: изменения, внесённые в преподавателей, аудитории, закрепления, практику и мероприятия, моментально отражаются в интерфейсе;
- Ответы Telegram-бота: текстовые сообщения и файлы, формируемые по командам /schedule, /auditory и /teacher, отображающие актуальное расписание, кабинет преподавателя или файл аудиторного фонда.

2.1.3. Подробные требования к проекту

Для обеспечения функциональности, эффективности и удобства работы системы генерации аудиторного фонда необходимо учитывать следующие детализированные требования.

2.1.3.1. Общие требования

- Функциональная полнота: система должна охватывать все ключевые процессы управления аудиторным фондом учебного заведения, включая загрузку расписания и изменений с сайта, его парсинг и обработку, генерацию Excel-шаблона, управление аудиториями, преподавателями, учебной практикой и мероприятиями, автоматическое распределение кабинетов и формирование итогового отчёта;
- Интеграция: программный комплекс должен включать взаимодействие клиентского приложения с серверным API, а также интеграцию с Telegram-ботом для получения актуальной информации о расписании и кабинетах;
- Надёжность и устойчивость: система должна корректно функционировать при больших объёмах входных данных, обеспечивать целостность, согласованность и доступность информации, а также защищать данные от потери и повреждения;
- Удобство интерфейса: пользовательский интерфейс должен быть интуитивно понятным, логически организованным, поддерживать фильтрацию по корпусу, дню недели и типу недели, а также обеспечивать быстрое выполнение операций по управлению и генерации;
- Производительность: система должна обеспечивать высокую скорость выполнения ключевых операций: загрузка и анализ расписания, обработка изменений, генерация Excel-файла, взаимодействие с базой данных через API, распределение кабинетов;
- Безопасность: доступ к функционалу должен быть защищён авторизацией по коду, взаимодействие между модулями должно осуществляться через защищённый API. Передаваемые данные не должны подвергаться риску перехвата или модификации;

• Масштабируемость: архитектура комплекса должна обеспечивать возможность расширения функциональности без необходимости полной переработки существующих компонентов;

• Доступность: Telegram-бот должен быть доступен круглосуточно, обеспечивая пользователям быстрый доступ к расписанию и информации об аудиториях без необходимости запуска основной программы;

• Унификация представления данных: информация, отображаемая в интерфейсе и через Telegram-бота, должна быть согласованной и актуальной, формироваться из одного источника — базы данных, синхронизированной с API.

2.1.3.2. Функциональные требования

Функционал для администратора(пользователя):

• Загрузка расписания в формате JSON с автоматическим распознаванием и отображением имени загруженного файла;

• Загрузка изменений в расписании с сайта и их автоматическая обработка с учётом текущего дня;

• Создание шаблона аудиторного фонда на основе данных о группах, квалификациях и паттернах;

• Управление преподавателями: добавление, редактирование и удаление преподавателей через административный интерфейс;

• Управление аудиториями: добавление, редактирование и удаление аудиторий с привязкой к корпусу (Нахимовский, Нежинская, Дистанционно);

• Привязка аудитории к преподавателю: создание, редактирование и удаление закреплений преподавателя за определённой аудиторией с учётом корпуса;

• Назначение предпочтений преподавателей по этажам с возможностью сохранения и последующего автоматического распределения;

• Добавление и удаление мероприятий: возможность указания даты, аудитории, названия и диапазона пар (отдельно для каждого корпуса);

• Добавление и удаление учебной практики: указание даты, группы, аудитории, преподавателя и диапазона пар (отдельно для корпусов);

• Формирование аудиторного фонда: загрузка шаблонного Excel-файла и автоматическая генерация распределения аудиторий с возможностью применения маркировки по дням недели, корпусу и типу недели (числитель/знаменатель);

• Подстановка кабинетов преподавателя автоматически, в зависимости от закреплений, предпочтений и наличия практики;

• Автоматическая отправка сгенерированного файла аудиторного фонда на сервер после успешной генерации;

• Авторизация пользователя по уникальному коду с возможностью выхода из системы;

• Просмотр и редактирование данных в интерфейсе в реальном времени (все действия моментально отображаются в UI).

Функционал Telegram-бота:

• Получение расписания по группе по команде /schedule;

• Получение Excel-файла с аудиторным фондом по команде /auditory;

• Определение кабинета преподавателя на текущий день по команде /teacher;

• Интерактивный выбор корпуса и уточнение группы/преподавателя при необходимости;

• Автоматическое обращение к API и получение данных в актуальном виде без участия пользователя.

2.2. Внешняя спецификация

2.2.1. Описание задачи

Разработать программный комплекс для управления данными об учебных аудиториях, преподавателях, группах и расписании на базе учебного заведения. Основной компонент комплекса — десктопное приложение, реализованное с использованием технологии WPF, использует реляционную базу данных под управлением СУБД Microsoft SQL Server и взаимодействует с ней через API, разработанное на платформе ASP.NET Core.

Приложение предоставляет пользователю (администратору) удобный интерфейс для загрузки расписания, распределения учебных аудиторий, управления данными о преподавателях, группах, мероприятиях и учебной практике. Пользователю доступны функции добавления, редактирования, удаления и просмотра данных, а также генерация Excel-файлов с актуальной информацией об использовании аудиторного фонда.

При запуске приложения открывается окно авторизации, где необходимо ввести код доступа. После успешной авторизации пользователь попадает на главный экран, где доступны следующие разделы:

- Корпуса — отображение списка корпусов учебного заведения (Нахимовский, Нежинская, Дистанционно);
- Преподаватели — управление списком преподавателей: добавление, редактирование и удаление;
- Аудитории — отображение и управление кабинетами с указанием корпуса;
- Привязка аудиторий — закрепление преподавателей за определёнными кабинетами, просмотр текущих связей, редактирование и удаление;
- События — добавление и удаление мероприятий с указанием даты, аудитории и диапазона пар;

• Учебная практика — создание записей о практике с указанием даты, группы, аудитории и преподавателя;

• Загрузка расписания — импорт файла .json с данными, полученными из встроенного парсера расписания колледжа;

• Маркировка аудиторного фонда — загрузка шаблонного Excel-файла и автоматическое распределение аудиторий с учётом корпуса, дня недели и типа недели;

• Выход — завершение работы с приложением.

Приложение поддерживает следующие ключевые функции:

• Загрузка и обработка расписания из JSON-файла, включая изменения;

• Авторизация по уникальному коду;

• Управление преподавателями и аудиториями;

• Привязка преподавателей к аудиториям с учётом корпуса;

• Создание, редактирование и удаление учебной практики;

• Создание, редактирование и удаление мероприятий;

• Генерация Excel-файлов аудиторного фонда с учётом корпуса, типа недели и закреплений;

• Отображение и фильтрация данных по корпусам и типу недели;

• Мгновенное отображение внесённых изменений в интерфейсе;

• Интеграция с серверным API для централизованного хранения и обработки данных.

Также в состав программного комплекса входит Telegram-бот, разработанный на языке Python с использованием библиотеки Aiogram. Бот позволяет пользователям получать актуальное расписание, информацию об аудиториях и кабинетах преподавателей по следующим командам:

• /schedule — получение расписания по выбранной группе;

• /auditory — получение Excel-файла с распределением аудиторий;

• /teacher — отображение кабинета преподавателя на текущий день.

Telegram-бот обращается к серверному API, не требует установки и обеспечивает быстрый доступ к информации как для студентов, так и для преподавателей.

Программный комплекс должен обеспечивать:

• Надёжное и безопасное хранение данных;

• Быструю и устойчивую работу при больших объёмах расписания;

• Интеграцию с внешними компонентами через API;

• Расширяемость — возможность подключения новых модулей и компонентов, таких как мобильные клиенты или расширенные отчёты.

Рисунок 1 демонстрирует сценарии взаимодействия пользователей с разработанной системой.

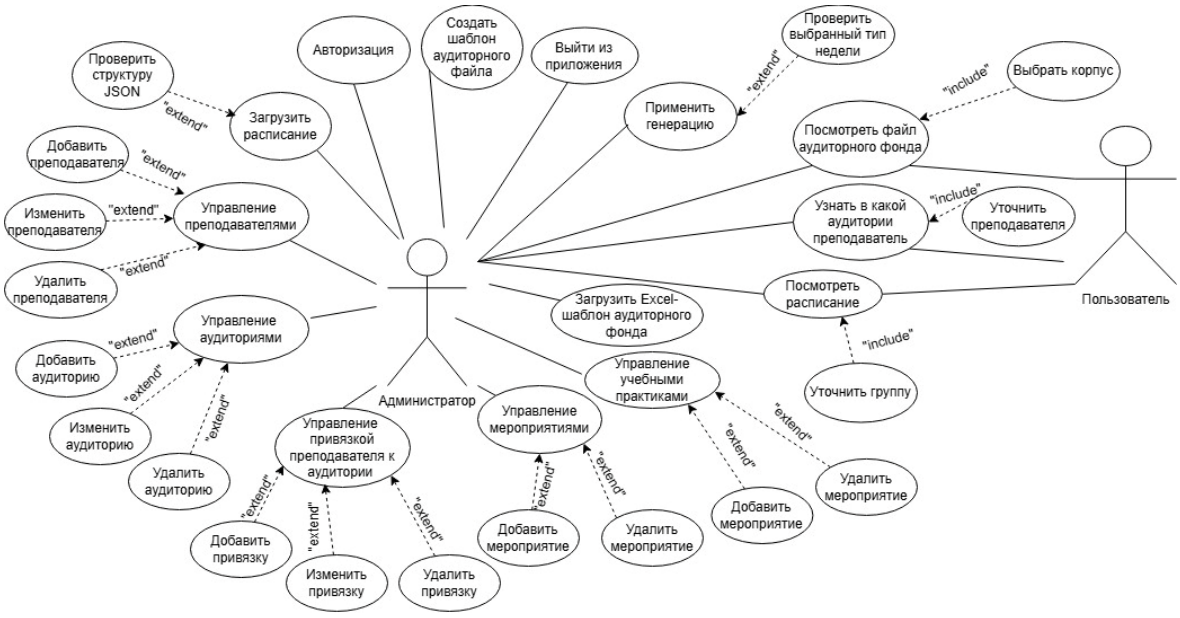

Рисунок 1 – Диаграмма Use Case

Для описания текущего процесса составления аудиторного фонда в колледже была построена IDEF0-диаграмма верхнего уровня. Она отображает все ключевые этапы ручного формирования фонда, включая входную информацию, управляющие воздействия, используемые механизмы и выходные результаты.

На рисунке 2 представлена модель AS-IS, описывающая существующий порядок работы методистов по подготовке таблицы аудиторного фонда. Основная функция — «Формирование аудиторного фонда вручную» — выполняется сотрудником учебного отдела без использования централизованной цифровой системы. Данные о расписании занятий, списки преподавателей, групп и аудиторий, а также сведения об учебной практике и мероприятиях предоставляются в виде Excel-файлов, бумажных листов или сообщений.

Процесс регламентируется внутренними документами колледжа, правилами составления расписания, указаниями методиста, стандартами оформления таблиц и необходимостью учитывать тип недели (числитель или знаменатель). Все эти элементы выступают в роли управляющих воздействий.

В качестве механизмов применяются: Microsoft Excel, где вручную составляется таблица, бумажные документы (например, закрепления), а также методист, выполняющий работу.

Результатом процесса является сформированная вручную таблица аудиторного фонда, которая может содержать ошибки и неточности. Она распечатывается или сохраняется в виде файла и передаётся преподавателям и администрации. Дополнительно могут возникать конфликты и дубли, которые фиксируются отдельно.

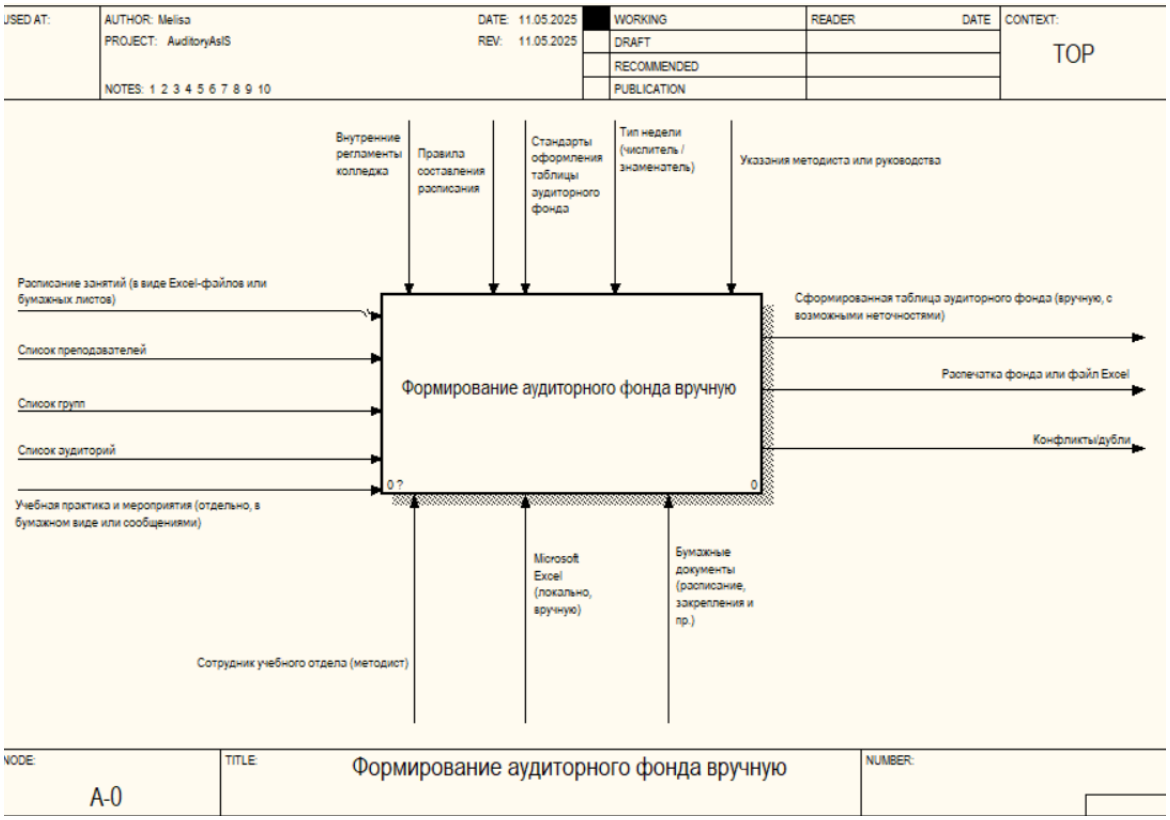

Рисунок 2 – Формирование аудиторного фонда вручную

На рисунке 3 представлена диаграмма IDEF0 уровня A0, описывающая текущий, неавтоматизированный процесс формирования аудиторного фонда в учебном заведении. Основной функцией выступает «Формирование аудиторного фонда вручную», выполняемое сотрудником учебного отдела (методистом) с использованием локальных и бумажных ресурсов.

Процесс начинается со сбора входных данных. Методист получает расписание занятий в виде Excel-файлов или бумажных листов, а также списки преподавателей, учебных групп и аудиторий. Дополнительно предоставляется информация об учебной практике и мероприятиях, которая поступает отдельно — в бумажном виде или в виде сообщений. Все эти сведения формируются вручную, часто из разных источников, что увеличивает вероятность ошибок и затрудняет анализ.

Процесс регулируется внутренними документами колледжа: правилами составления расписания, указаниями методиста или руководства, стандартами оформления таблиц аудиторного фонда, а

также необходимостью учитывать тип недели (числитель или знаменатель). Эти документы определяют порядок и формат сбора, анализа и оформления информации.

На первом этапе происходит сбор и подготовка данных вручную. Методист вручную обрабатывает всю входную информацию, формируя черновые материалы в тетради, Excel или Word. На этом же этапе осуществляется предварительная группировка данных по корпусам и преподавателям, на основе которой будут построены таблицы фонда.

Следующим этапом является анализ и сопоставление данных. Методист вручную проверяет черновые материалы на наличие ошибок, дублирующих или пропущенных данных. Затем выполняется ручное сопоставление преподавателей, групп и аудиторий с учётом корпуса и типа недели. В результате формируется таблица распределения аудиторий по преподавателям и группам, которая переносится в шаблон Excel или оформляется в виде печатной формы.

Финальным шагом является оформление и финализация таблицы. Методист вручную оформляет таблицу согласно установленным стандартам, заполняет недостающие данные, при необходимости вносит правки и завершает работу с документом. После этого формируется итоговая таблица аудиторного фонда, которая может содержать неточности. Она распечатывается либо сохраняется в виде файла Excel и передаётся преподавателям и руководству. При наличии проблем фиксируются конфликты, дубли или другие ошибки, которые могут быть замечены уже после использования таблицы.

Таким образом, процесс формирования аудиторного фонда в текущем виде является трудоёмким, зависит от человеческого фактора и подвержен ошибкам. Он требует значительного времени на сбор, обработку и верификацию данных, не позволяя в полной мере контролировать точность и актуальность распределения учебных помещений.

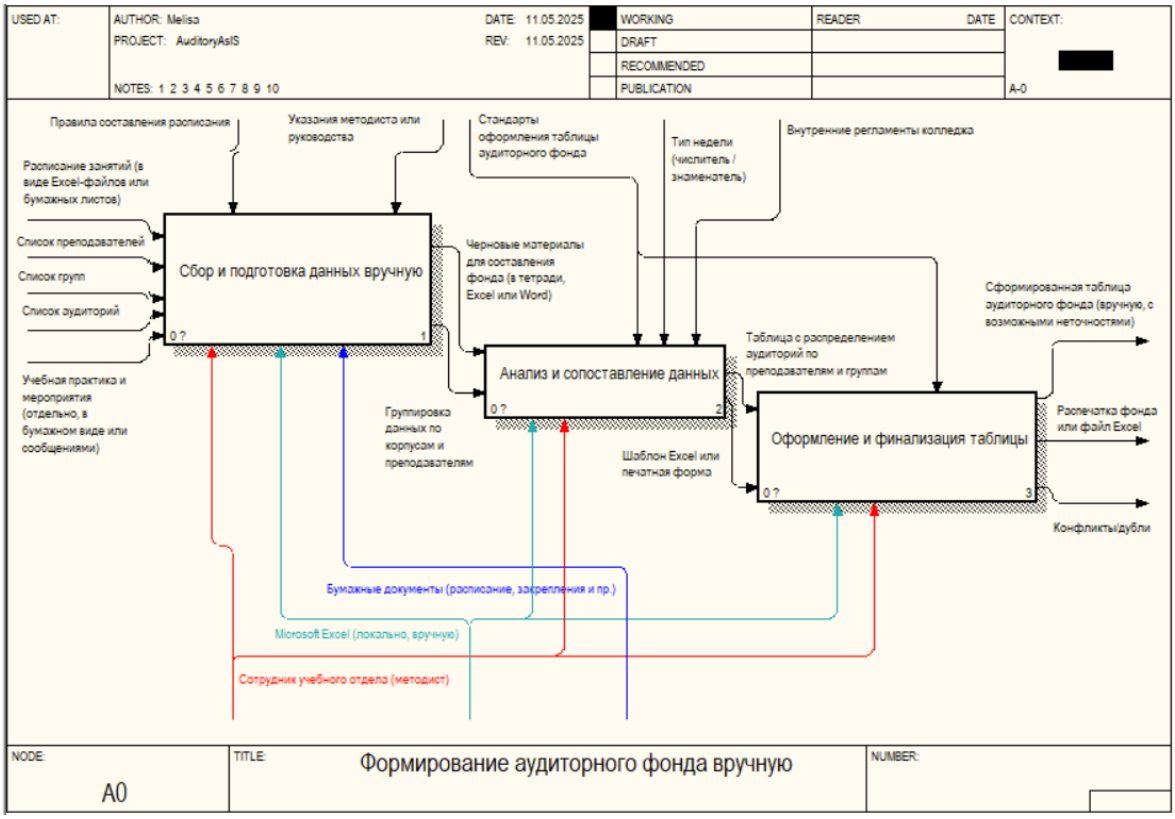

Рисунок 3 – Диаграммы A0 уровня AS-IS

Диаграмма IDEF0 уровня A1 на рисунке 4 детализирует первый этап ручного формирования аудиторного фонда — процесс сбора и подготовки входных данных, выполняемый сотрудником учебного отдела (методистом).

На начальном этапе методист вручную получает расписание занятий в виде Excel-файлов или бумажных листов, списки преподавателей, групп и аудиторий, а также отдельные сведения об учебной практике и мероприятиях. Эти данные часто передаются по частям, в неструктурированном виде и без единых форматов. Помимо основных входов, методист руководствуется правилами составления расписания и указаниями администрации или старших методистов.

Процесс начинается с получения всех этих данных вручную. Методист собирает информацию из разных источников и переносит её на бумажные носители или в локальные Excel-файлы. На этом этапе возникают частично заполненные черновики, в которых информация представлена фрагментарно и с ошибками. Формируются неполные или

несогласованные записи о преподавателях, аудиториях, группах и занятиях.

Следующим шагом является предварительная группировка и структурирование. Методист пытается на основе собранных данных определить распределение по корпусам и преподавателям. При этом информация остается в черновом виде, фиксируется вручную, не проверяется автоматически и не проходит унификацию. Используются бумажные документы (распечатки расписания, закрепления) и Excel-файлы, заполняемые вручную.

На выходе формируются два ключевых продукта: черновые материалы для последующего составления фонда, которые могут представлять собой записи в тетради, Word или Excel, а также первичная группировка по корпусам и преподавателям, являющаяся основой для следующего этапа — анализа и сопоставления данных.

Таким образом, этап сбора и подготовки данных вручную отличается высокой долей ручного труда, отсутствием валидации и сильной зависимостью от человеческого фактора. Это снижает точность на последующих этапах и увеличивает вероятность появления дублирующей или конфликтной информации.

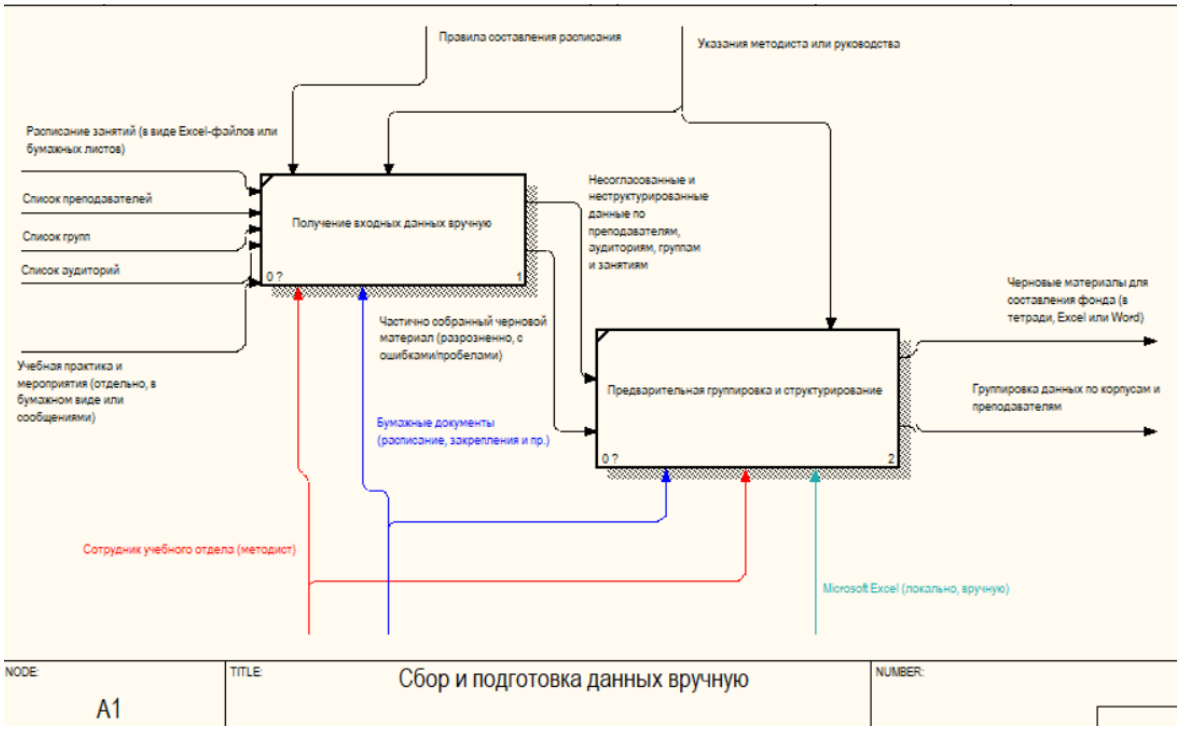

Рисунок 4 – Диаграмме A1 (AS-IS)

На диаграмме IDEF0 уровня A2 представлен этап анализа и сопоставления данных в процессе ручного формирования аудиторного фонда. Данный этап наступает после предварительного сбора и подготовки информации и представляет собой ручную проверку, корректировку и организацию данных, необходимых для распределения аудиторий.

Входными данными для процесса выступают черновые материалы, полученные после этапа сбора: тетради, локальные Excel-файлы или распечатки, содержащие информацию о группах, преподавателях, аудиториях и практике. Также в работу поступают предварительно сгруппированные данные по корпусам и преподавателям, которые методист формирует в ходе подготовки. Этап выполняется в соответствии со стандартами оформления таблицы аудиторного фонда, внутренними регламентами колледжа и с учётом текущего типа недели (числитель или знаменатель).

Первым подэтапом является ручная проверка черновиков. Методист последовательно просматривает все полученные материалы, выделяя явные ошибки, дубли, противоречивую или устаревшую

информацию. Отмечаются конфликты — например, ситуации, при которых преподаватель назначен сразу в несколько аудиторий или у группы оказывается более одной пары в одно и то же время. Черновики дополняются или исправляются вручную. В процессе проверки нередко приходится обращаться к бумажным документам — оригиналам расписания, закреплений или дополнительным комментариям от руководства.

После этого переходит ко второму подэтапу — ручному сопоставлению групп, преподавателей и аудиторий. Методист вручную подбирает аудитории под пары, стараясь учесть ограничения по корпусам, соответствие требованиям дисциплин и предварительные закрепления. Здесь также производится финальное распределение с учётом типа недели, чтобы исключить конфликтные назначения. Окончательные данные фиксируются в локальном Excel-файле или печатной таблице, которая используется в следующем этапе — оформлении.

Результатом этапа является подготовленная таблица с распределением аудиторий по преподавателям и группам. Также формируется шаблон Excel или его печатная версия, отражающая вручную составленные данные.

Таким образом, этап анализа и сопоставления данных является одним из самых ответственных в ручной системе формирования фонда. Он требует внимательности, многократной перепроверки и постоянного участия методиста, поскольку любые ошибки на этом этапе напрямую влияют на точность и корректность итогового фонда.

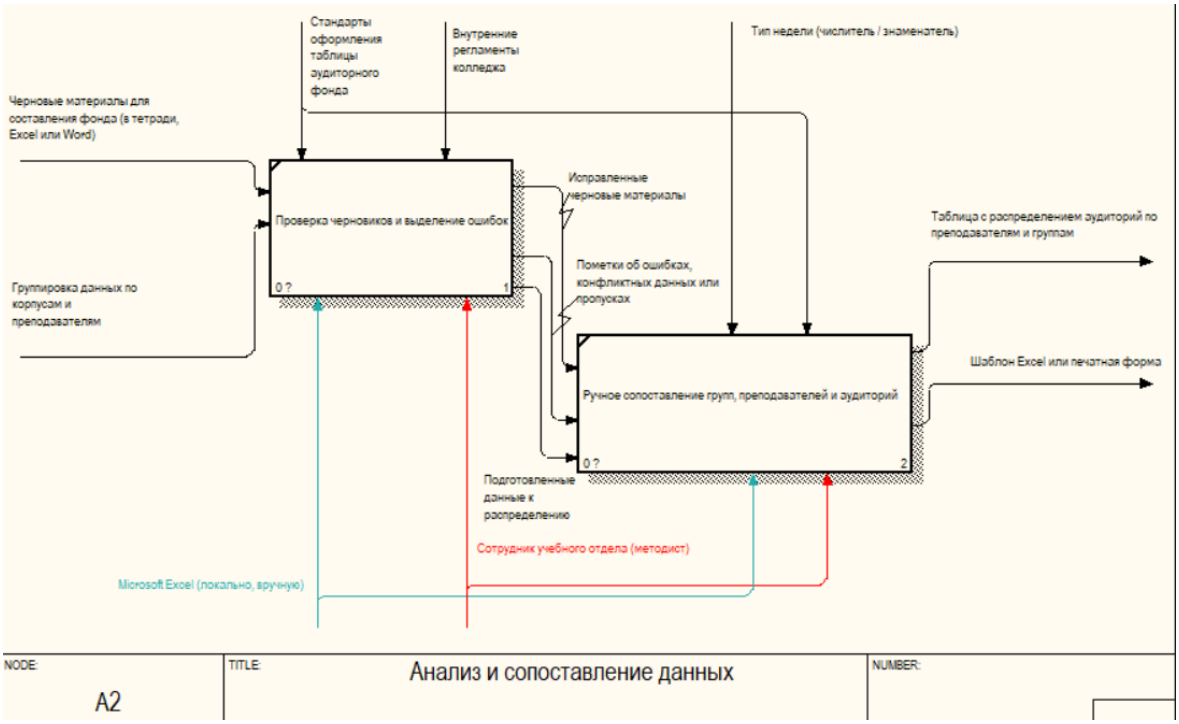

Рисунок 5 – Диаграмма A2 уровня AS-IS

Диаграмма IDEF0 уровня A3 описывает заключительный этап ручного формирования аудиторного фонда — оформление итоговой таблицы и передача результата в виде файла или распечатки. Данный процесс базируется на данных, подготовленных ранее при сопоставлении групп, преподавателей и аудиторий.

На входе используются две ключевые составляющие: таблица с распределением, составленная вручную на предыдущем этапе, и шаблон Excel или печатная форма, которую методист заполняет. В качестве управляющих воздействий выступают стандарты оформления таблицы аудиторного фонда, регламентирующие структуру, стилистику и допустимые обозначения.

Первым подэтапом является заполнение финальной таблицы. Методист вручную переносит информацию в шаблон Excel, формируя итоговую таблицу фонда. На этом этапе велика вероятность появления ошибок, связанных с человеческим фактором — пропущенные пары, некорректно скопированные данные, несоответствие названий и т.д. По мере заполнения фиксируются обнаруженные дубли, несостыковки или неясные фрагменты. В некоторых случаях создаются временные

черновики для повторной перепроверки и корректировки данных перед финальной фиксацией.

Далее осуществляется сохранение и передача результатов. Финальная таблица может быть распечатана или сохранена в виде файла Excel. Важно отметить, что данный файл не всегда проходит верификацию, и в нём могут сохраняться ошибки. Вместе с основным результатом сохраняется информация о конфликтах и дублях, которую методист использует при устной передаче или пояснениях.

Механизмы выполнения представлены сотрудником учебного отдела, который вручную выполняет все действия, а также программой Microsoft Excel, используемой для ручного редактирования. Результатом этапа является сформированная таблица аудиторного фонда — в печатной или электронной форме — готовая к передаче преподавателям и руководству.

Таким образом, этап оформления и финализации в ручной системе отличается высокой трудоёмкостью и уязвимостью к ошибкам. Отсутствие автоматической проверки и единых форматов снижает надёжность данных и требует постоянного внимания со стороны исполнителя.

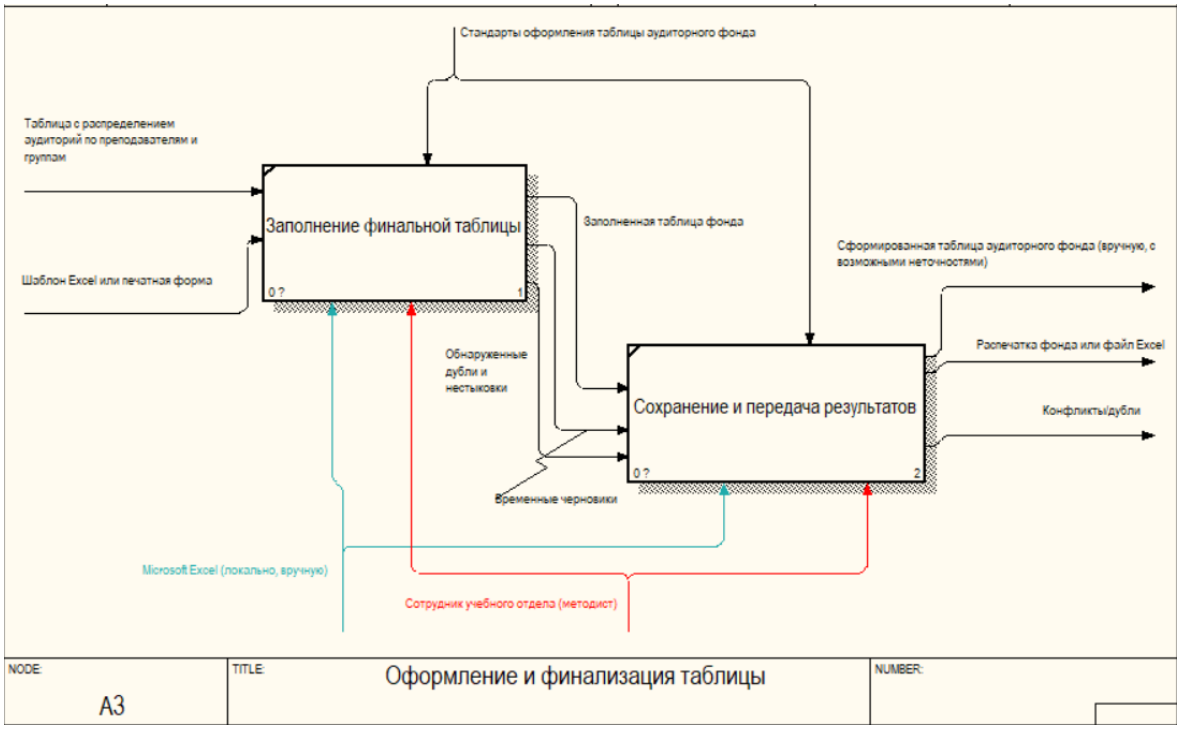

Рисунок 6 – Диаграмма A3 (AS-IS)

Диаграмма IDEF0 верхнего уровня (A0) отражает автоматизированный процесс формирования аудиторного фонда в учебном заведении. В отличие от предыдущей (ручной) модели, здесь используются цифровые технологии, обеспечивающие повышение точности, скорости и прозрачности операций.

На вход в систему подаются структурированные JSON-файлы, полученные из парсера расписаний и изменений: данные о преподавателях, группах, аудиториях, учебной практике и мероприятиях. Эти входные данные соответствуют утверждённой структуре, проверяются согласно правилам валидации и фильтруются в зависимости от заданных параметров (корпус, день недели, числитель/знаменатель). В качестве управляющих воздействий применяются: шаблон Excel-файла, правила генерации, внутренний регламент колледжа, а также правила формирования ответов Telegram-бота.

Главная функция блока — автоматически обработать и связать все входные данные между собой, сформировать распределение аудиторий и

вывести несколько результирующих файлов. Сотрудник (администратор системы) взаимодействует с системой через десктопное WPF-приложение, которое передаёт данные в API ASP.NET Core, осуществляющее основную логику. База данных размещена на Microsoft SQL Server, а результаты отправляются как на Telegram-бот (в режиме чтения), так и в виде Excel-файлов через библиотеку ClosedXML.

На выходе получаем:

- готовый Excel-файл аудиторного фонда;
- структурированную информацию по аудиториям и преподавателям (для Telegram-бота);
- распределения по корпусам и неделям;
- текстовые ответы на команды Telegram-бота: /auditory, /teacher, /schedule.

Таким образом, процесс полностью исключает ручной ввод и сопоставление данных, минимизируя ошибки и ускоряя формирование актуального фонда. Интеграция с Telegram позволяет также предоставить преподавателям и студентам быстрый доступ к данным.

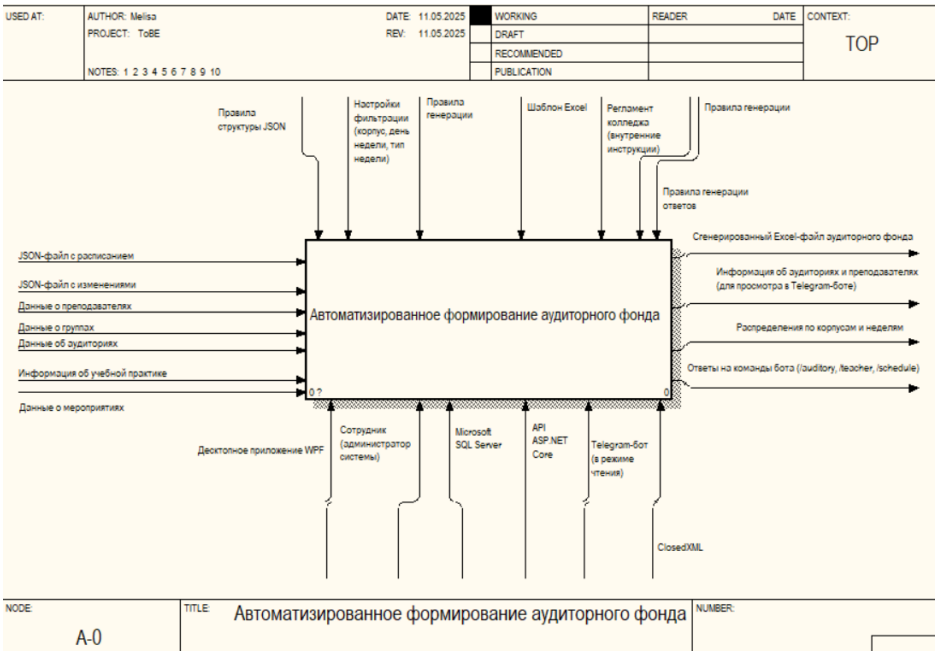

Рисунок 7 – Автоматизированное формирование аудиторного фонда

Диаграмма A0 отражает автоматизированный процесс формирования аудиторного фонда. Система получает входные данные

(JSON-файлы с расписанием, изменениями, преподавателями, группами, аудиториями, практикой и мероприятиями), которые проходят валидацию и фильтрацию. Затем происходит логическая обработка и распределение аудиторий по корпусам и преподавателям с учётом регламентов и предпочтений.

Готовая информация экспортируется в Excel по шаблону и передаётся Telegram-боту. Результаты включают: сформированный Excel-файл фонда, ответы на команды бота, распределения по корпусам и недели, а также структурированные данные о преподавателях и аудиториях. Все действия выполняются через WPF-приложение с взаимодействием по API.

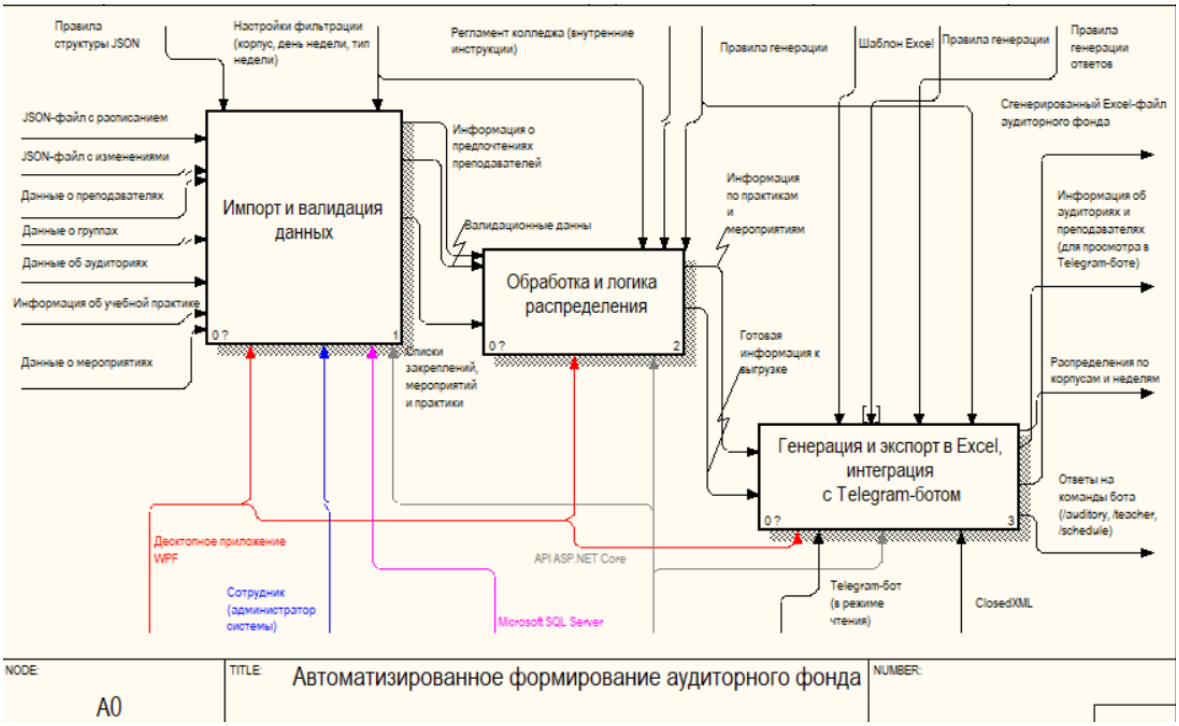

Рисунок 8 – Диаграмма A0

Диаграмма A1 представляет этап «Импорт и валидация данных» в процессе автоматизированного формирования аудиторного фонда.

Процесс начинается с загрузки данных: JSON-файлов с расписанием, изменениями, списками преподавателей, групп, аудиторий, практики и мероприятий. Эти данные поступают из WPF-приложения, заполняются администратором и сохраняются в локальную систему или

в базу данных. Также на данном этапе учитываются правила структуры JSON и настройки фильтрации (корпус, день недели, тип недели).

На первом подэтапе происходит загрузка и структуризация данных — система выделяет и преобразует информацию в читаемый формат, распределяя её по категориям: расписание, группы, аудитории, преподаватели. На выходе формируется упорядоченный набор структурированных данных.

Во втором подэтапе проводится валидация: данные проходят проверку на соответствие фильтрам и структурам, выявляются ошибки, дубликаты и несоответствия. По результатам формируются три выходных потока:

- информация о предпочтениях преподавателей (например, по этажам),
- валидационные данные, готовые к распределению,
- списки закреплений, мероприятий и практик для использования в последующих этапах.

В работе участвуют: сотрудник-администратор, система хранения (Microsoft SQL Server) и сервер взаимодействия (API ASP.NET Core).

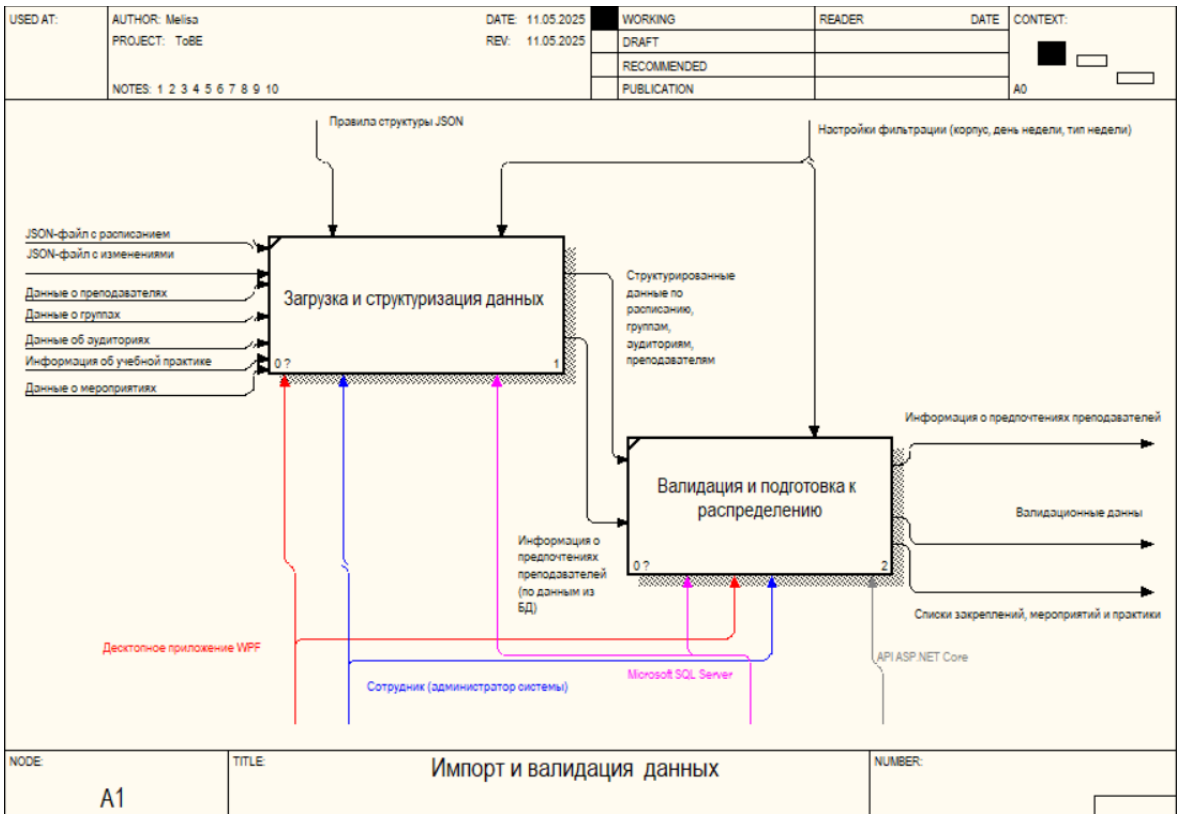

Рисунок 9 – Диаграмма A1 уровня ToBE

Диаграмма A2 описывает этап «Обработка и логика распределения» в автоматизированной системе формирования аудиторного фонда.

Процесс начинается с блока обработки валидированных данных и закреплений. На этом этапе в систему поступают: валидационные данные (структурированные сведения о преподавателях, группах, аудиториях и занятиях), списки закреплений, мероприятий и практик, информация о предпочтениях преподавателей (например, этаж, кабинет), параметры фильтрации (корпус, день недели, тип недели), данные от WPF-приложения и администратора.

Система объединяет и анализирует все эти данные, производит сверку закреплений и предпочтений, а также подготавливает информацию к последующему распределению. На выходе формируются частично распределённые данные, готовые для логической генерации.

На следующем этапе — «Применение логики распределения и подготовка к выгрузке» — данные проходят по правилам распределения: подстановка кабинетов преподавателей, предпочтений по этажам, учёт

закреплений и загрузка информации о мероприятиях и практике. При этом учитываются внутренние регламенты колледжа и правила генерации (в том числе — правила ответов в Telegram-боте).

Результатом этапа являются: финальные данные для выгрузки, информация по практикам и мероприятиям, промежуточные данные, подготовленные к экспорту в Excel и Telegram-бот.

Процесс активно взаимодействует с API ASP.NET Core и WPF-приложением, обеспечивая адаптивную, централизованную обработку всех компонентов распределения.

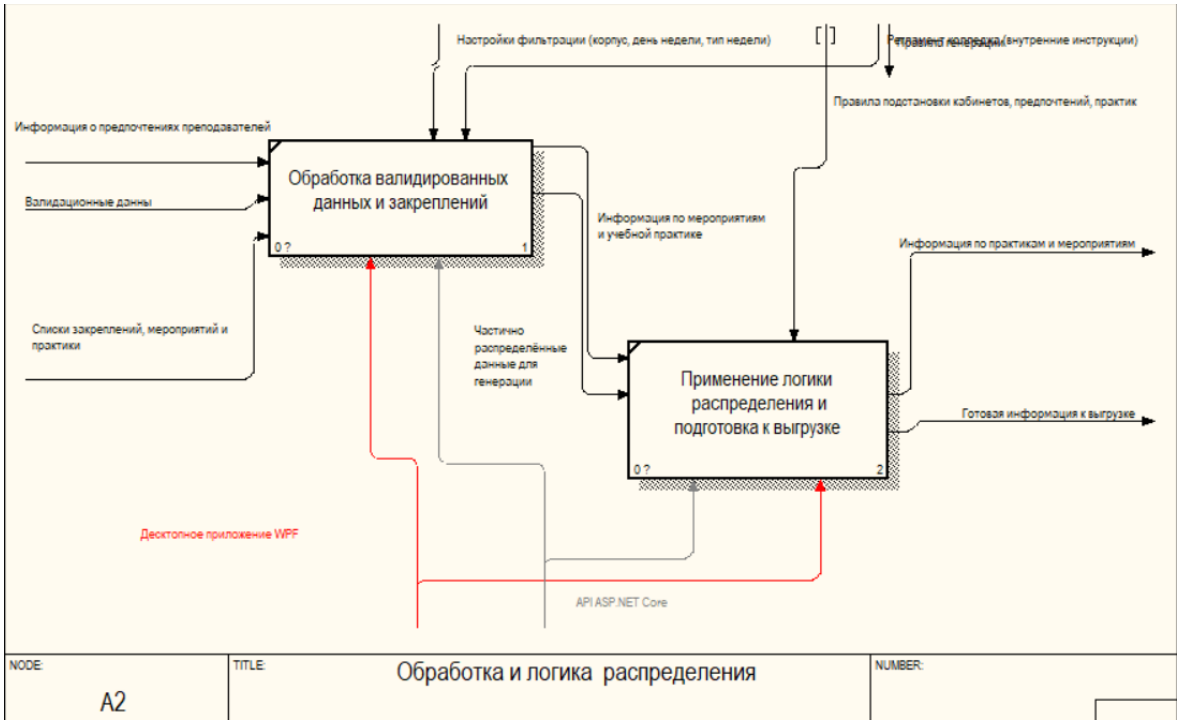

Рисунок 10 - Диаграмма A2 уровня ToBE

Диаграмма A3 описывает заключительный этап процесса — «Генерация и экспорт в Excel, интеграция с Telegram-ботом».

На первом этапе выполняется генерация Excel-файла. В качестве входных данных используются: шаблон Excel-документа, готовая к выгрузке информация (группы, аудитории, преподаватели, тип недели), информация по учебным практикам и мероприятиям, правила генерации и логика отображения (например, цветовая разметка или формат записи).

Процесс реализуется с помощью библиотеки ClosedXML и инициируется из WPF-приложения. Сформированный Excel-файл

содержит распределение преподавателей по кабинетам и визуальное представление аудиторного фонда. Он используется не только для отчётности, но и для дальнейшей интеграции в Telegram-бота.

На следующем шаге запускается модуль «Интеграция с Telegram-ботом». Он принимает: готовый Excel-файл, данные о распределении по корпусам и неделям, правила генерации ответов.

Telegram-бот, находящийся в режиме только чтения, предоставляет пользователю следующие результаты: информацию об аудиториях и преподавателях по запросу, распределение по корпусам, ответы на команды /auditory, /teacher и /schedule.

Таким образом, результат генерации автоматически доступен в мессенджере, обеспечивая студентам и сотрудникам быстрый доступ к актуальной информации без необходимости вручную открывать Excel-файл.

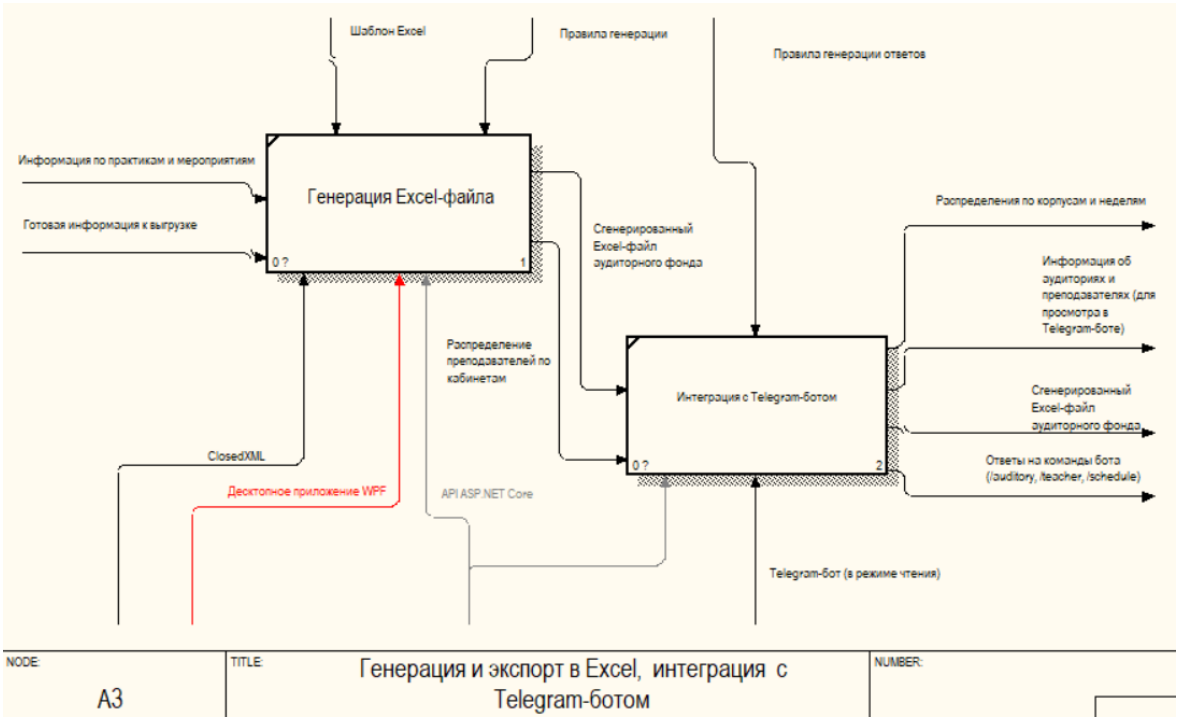

Рисунок 11 - Диаграмма A3 уровня ToBE

2.2.2. Входные и выходные данные

В таблице 4 представлены входные данные.

Таблица 4 – Входные данные

Имя	Тип	Ограничение	Формат ввода	Описание
1	2	3	4	5
Входные данные				
Формы аутентификации				
AuthCode	Строка	^[A-Za-z0-9]{6,20}$	Поле ввода	Аутентификация — код доступа для входа
Импорт расписания				
ScheduleFile	JSON-файл	^.+\.json$	Импорт файла	Импорт файла расписания с сайта, содержащего информацию о группах, преподавателях, днях недели, парах, корпусах и типах занятий
Импорт шаблона Excel				
ExcelTemplate	Excel-файл	^.+\.xlsx$	Импорт файла	Импорт шаблона Excel-файла для генерации таблицы аудиторного фонда
Добавление преподавателя				
TeacherFullName	Строка	^[A-Za-zA-Яа-яЁё\\-\\.\\s]{5,100}$	Поле ввода	ФИО преподавателя
Закрепление кабинета за преподавателем				
RoomId	Целое число	^(\d{1,3})(,\d{1,3})*$	Поле выбора	Список ID аудиторий
TeacherId	Целое число	^(\d{1,3})(,\d{1,3})*$	Поле выбора	Список ID преподавателей
Добавление аудитории				

Имя	Тип	Ограничение	Формат ввода	Описание
1	2	3	4	5
RoomNumber	Строка	^\d{1,5}$	Поле ввода	Номер учебной аудитории (например, 306)
CampusId	Целое число	^(\d{1,3})(,\d{1,3})*$	Поле выбора	Корпус, к которому относится аудитория
Ввод информации о мероприятии:				
EventDate	Дата	^\d{2}\.\d{2}\.\d{4}$	Поле выбора	Дата проведения мероприятия
EventRoom	Целое число	^\d+$	Поле выбора	Аудитория, в которой проводится мероприятие
EventName	Строка	^[A-Za-zA-Яа-яЁё0-9\\-\\.\\s]{1,100}$	Поле ввода	Название мероприятия
EventLessons	Строка	^([1-8])(-[1-8])?$	Поле ввода	Диапазон пар, на которые попадает мероприятие (например, 1-4)
Ввод информации об учебной практике				
PracticeDate	Дата	^\d{2}\.\d{2}\.\d{4}$	Поле выбора	Дата проведения учебной практики
PracticeGroup	Целое число	^\d+$	Поле выбора	Идентификатор группы, участвующей в практике
PracticeTeacher	Целое число	^\d+$	Поле выбора	Идентификатор преподавателя, ответственного за практику
PracticeRoom	Целое число	^\d+$	Поле выбора	Идентификатор аудитории, закреплённо

Имя	Тип	Ограничение	Формат ввода	Описание
1	2	3	4	5
				й под практику
PracticeLessons	Строка	^([1-8])(-[1-8])?$	Поле ввода	Диапазон пар, в которые проходит практика (например, 2-4)
Выбор типа недели				
WeekType	Строка	^(Числитель|Знаменатель)$	Поле выбора	Тип недели, используемый при составлении расписания
Настройки фильтрации				
FilterCampus	Строка	^(Нахимовский|Нежинская|Дистанционно)$	Поле выбора	Корпус, выбранный для генерации аудиторного фонда
FilterWeekType	Строка	^(Числитель|Знаменатель)$	Поле выбора	Тип недели, выбранный при генерации
FilterDayOfWeek	Строка	^(Понедельник|Вторник|Среда|Четверг|Пятница|Суббота)$	Поле выбора	День недели для распределения аудиторий
Загрузка изменений в расписании				
ChangesFile	JSON-файл	^.+\.json$	Импорт файла	JSON-файл с изменениями в расписании (отмена занятий, замены преподавателей и др.)
Предпочтения преподавателей				
TeacherPreference	Целое число	^[1-9]$	Поле выбора	Предпочтительный этаж для преподавателя

Имя	Тип	Ограничение	Формат ввода	Описание
1	2	3	4	5
Команды Telegram-бота (как входные события)				
BotCommand	Строка	^/(auditory|teacher|schedule)$	Ввод команды	Команда, запрошенная пользователем в Telegram-боте

В таблице 5 представлены выходные данные.

Таблица 5 – Выходные данные

Имя	Тип	Ограничение	Описание
1	2	3	4
Генерация Excel-файла			
AuditoryFundFile	XLSX-файл	^.+\.xlsx$	Формирование и экспорт Excel-файла с распределением
AutoRoomAssignment	Строка	^.+$	Автоматическая подстановка кабинета преподавателя в Excel-файл по закреплениям
PracticeRoomAssignment	Строка	^.+$	Подстановка аудитории на основе учебной практики
ExcelHighlighting	Цветовая метка	^(#(?:[0-9a-fA-F]{6}))$	Заливка ячеек Excel-файла в зависимости от типа занятия
UpdatedAuditoryFundFile	XLSX-файл	^.+\.xlsx$	Обновлённый файл после генерации с учётом закреплений, фильтров и практики
Отображение закреплённых аудиторий			
AssignedRoomsView	Строка	^[A-Za-zA-Яа-яЁё0-9\\-\\.\\s]{1,100}$	Просмотр закреплённых аудиторий
TeachersList	Список объектов	^.+$	Список преподавателей
RoomsList	Список объектов	^.+$	Список аудиторий

Имя	Тип	Ограничение	Описание
1	2	3	4
PracticeDisplay	Список объектов	^.+$	Отображение учебной практики
EventsDisplay	Список объектов	^.+$	Отображение мероприятий
ScheduleJsonParsed	JSON-объект	^.+$	Просмотр загруженного расписания
FilteredWeekView	Список объектов	^.+$	Отображение данных по выбранной неделе и дню
Работа Telegram-бота			
TelegramBotResponse	JSON-объект	^.+$	Ответ Telegram-бота на команды /auditory, /teacher, /schedule
Обработка изменений в расписании			
ScheduleChangesParsed	JSON-объект	^.+$	Результат обработки изменений в расписании (отмены, замены и переносы)
ChangesAppliedExcel	Строка	^.+$	Отражение изменений в расписании в Excel-файле аудиторного фонда
ChangesDisplay	Список объектов	^.+$	Отображение обработанных изменений в интерфейсе администратора

2.2.3. Методы

База данных проекта нормализована до третьей нормальной формы (3НФ), что позволяет минимизировать избыточность информации и обеспечить логическую целостность данных. Вся бизнес-логика обработки данных реализуется на стороне сервера, полностью изолированной от клиентского интерфейса, что обеспечивает безопасность и независимость уровней приложения.

Для взаимодействия между клиентским приложением и сервером используется программный интерфейс (API), разработанный на

платформе ASP.NET Core. Через API выполняются все операции: загрузка, обновление, удаление и выборка данных из базы, что обеспечивает гибкость архитектуры, возможность масштабирования и упрощение подключения дополнительных клиентов, включая Telegram-бот.

Клиентская часть реализована в виде WPF-приложения с применением объектно-ориентированного подхода. Все модули разделены по функциональности, что обеспечивает модульность, повторное использование кода и упрощённую поддержку. Используется принцип инкапсуляции: все данные и методы скрыты внутри классов и доступны через строго определённые интерфейсы.

В интерфейсе приложения применяются элементы реактивного программирования: при изменении входных данных, например фильтрации по типу недели или дню, автоматически обновляются связанные представления, что обеспечивает мгновенный отклик интерфейса и удобство пользователя. Такой подход повышает отзывчивость системы и делает интерфейс интуитивно понятным.

В процессе разработки приложения использовались методы рефакторинга и оптимизации, направленные на повышение качества, читаемости и производительности программного кода. Рефакторинг позволил устранить дублирование логики, улучшить структуру методов и повысить удобство сопровождения проекта. Оптимизация обеспечила снижение нагрузки на интерфейс, ускорение отклика при взаимодействии с пользователем и уменьшение количества лишних вычислений.

На рисунке 12 представлен исходный фрагмент кода до применения рефакторинга. В данной версии каждый метод загрузки данных содержал повторяющуюся логику по обращению к API, чтению ответа и десериализации JSON. Такое дублирование усложняло поддержку и масштабирование проекта.

После рефакторинга, представленного на рисунке 13, общая логика получения и обработки данных была вынесена в универсальный метод FetchData<T>(), что позволило сократить количество повторений, повысить читаемость кода и упростить добавление новых типов запросов.

```
private async void LoadTeachers()
{
    try
    {
        var response = await _httpClient.GetAsync($"{_teachersUrl}");
        response.EnsureSuccessStatusCode();
        var json = await response.Content.ReadAsStringAsync();

        allTeachers = JsonSerializer.Deserialize<List<Teacher>>(json, new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true
        });

        TeacherComboBox.ItemsSource = allTeachers;
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка при загрузке преподавателей: {ex.Message}");
    }
}
```

Рисунок 12 – Код до применения рефакторинга

```
Ссылок: 1
private async Task<List<T>> FetchData<T>(string url)
{
    var response = await _httpClient.GetAsync(url);
    response.EnsureSuccessStatusCode();
    var json = await response.Content.ReadAsStringAsync();
    return JsonSerializer.Deserialize<List<T>>(json, new JsonSerializerOptions
    {
        PropertyNameCaseInsensitive = true
    });
}
Ссылок: 1
private async void LoadTeachers()
{
    try
    {
        allTeachers = await FetchData<Teacher>(_teachersUrl);
        TeacherComboBox.ItemsSource = allTeachers;
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка при загрузке преподавателей: {ex.Message}");
    }
}
```

Рисунок 13 – Код после применения рефакторинга

На рисунке 14 продемонстрирован код до оптимизации. В нем при каждом вводе символа в поле выбора преподавателя производился пересчёт всего списка и обновление источника данных, что вызывало лишние перерисовки интерфейса и снижало отзывчивость при быстром вводе.

В оптимизированной версии, представленной на рисунке 15, была добавлена простая проверка на изменение текста поиска. Это позволило избежать повторной фильтрации, если значение не изменилось, и тем самым снизить нагрузку на интерфейс. Кроме того, использовалась более

безопасная проверка с приведением к нижнему регистру для нечувствительности к регистру, что улучшило пользовательский опыт.

```
Ссылок: 1
private void TeacherComboBox_PreviewKeyUp(object sender, KeyEventArgs e)
{
    var comboBox = (ComboBox)sender;
    var searchText = comboBox.Text.ToLower();

    var filteredTeachers = allTeachers
        .Where(t => t.FullName.ToLower().Contains(searchText))
        .ToList();

    comboBox.ItemsSource = filteredTeachers;
    comboBox.IsDropDownOpen = true;
}
```

Рисунок 14 – Код до оптимизации

```
private string _lastSearch = "";

private void TeacherComboBox_PreviewKeyUp(object sender, KeyEventArgs e)
{
    var comboBox = (ComboBox)sender;
    var searchText = comboBox.Text.ToLower();

    if (searchText == _lastSearch) return;
    _lastSearch = searchText;

    var filtered = allTeachers
        .Where(t => t.FullName != null && t.FullName.ToLower().Contains(searchText))
        .ToList();

    comboBox.ItemsSource = filtered;
    comboBox.IsDropDownOpen = true;
}
```

Рисунок 15 – Код после оптимизации

2.2.4. Тесты

1. По формальности тестирования.

Тестирование выполнялось на основе заранее составленных тест-кейсов и направлено на проверку функций десктопного приложения и Telegram-бота на соответствие заданным сценариям. Это позволило системно подтвердить корректность работы отдельных модулей и общего поведения системы.

2. По исполнению кода.

Применялось динамическое тестирование, направленное на проверку компонентов системы во время их выполнения. Оно использовалось для выявления ошибок в логике распределения аудиторий, генерации Excel-файлов и отправке ответов от Telegram-бота.

3. По уровню тестирования.

Системное тестирование охватывало всю систему в целом, включая клиентскую часть, API и базу данных. Проверялась согласованность

работы между модулями, корректность обработки входных данных и стабильность генерации выходных файлов.

4. По целям.

Функциональное тестирование было направлено на проверку реализации всех заявленных функций: импорта расписания, редактирования данных, автоматического распределения и отображения информации в Telegram-боте. Все тесты соответствовали ожидаемому поведению системы.

5. По степени автоматизации.

В проекте использовалось ручное тестирование через пользовательский интерфейс и вызовы API. Такой подход позволил проверить реализацию с точки зрения конечного пользователя и убедиться в стабильности поведения при различных условиях.

6. По позитивности сценария.

Проводились как позитивные, так и негативные тесты. Позитивные сценарии включали стандартные действия пользователя, такие как загрузка корректного JSON-файла или генерация Excel. Негативные тесты — работу с повреждёнными файлами, пустыми данными и некорректными параметрами, что позволило выявить и обработать потенциальные ошибки.

7. По знанию системы.

Применялось тестирование «белого ящика» на этапе разработки логики распределения и API. Это позволило провести отладку внутренних алгоритмов, проанализировать структуру кода и убедиться в корректности работы всех модулей.

8. По разработке тестовых испытаний.

Проводилось тестирование на основе требований, изложенных в техническом задании. Каждая функция сравнивалась с заявленной, чтобы убедиться в соответствии конечного продукта ожиданиям пользователей и спецификации.

2.2.5. Контроль целостности данных

В процессе разработки были реализованы методы валидации входных данных в информационной системе, которые обеспечивают целостность этих входных данных.

Контроль целостности, описывающих ситуации и реакции приложения на выполнения функций представлен в таблице 6.

Таблица 6 – Ситуации и аномалии

№	Ситуация	Аномалия	Реакция	Примечание
1	2	3	4	5
1	Загрузка файла расписания	Файл повреждён или не соответствует ожидаемой структуре JSON	Программа выводит ошибку «Некорректный формат файла расписания» и отклоняет импорт	Допускается только строго структурированный JSON-файл, соответствующий требованиям системы
2	Создание таблицы аудиторного фонда	Отсутствуют данные о группах или преподавателях	Программа останавливает генерацию таблицы и сообщает: «Недостаточно данных для формирования аудиторного фонда»	Генерация невозможна без обязательных сущностей: групп и преподавателей
3	Запись данных в Excel-файл	Указанный путь сохранения не существует или недоступен	Программа выводит сообщение: «Не удалось сохранить файл. Проверьте путь и права доступа»	Пользователь должен выбрать корректную директорию с правами на запись
4	Указание диапазона ячеек для группы	Диапазон выходит за границы допустимой области шаблона Excel	Программа выводит ошибку: «Недопустимый диапазон ячеек для группы»	Для корректного отображения данные не должны выходить за границы таблицы
5	Указание закреплённой аудитории	Преподаватель уже привязан к другой аудитории на то же время	Программа сообщает об ошибке конфликта: «Преподаватель уже имеет закреплённую	Один преподаватель не может быть закреплён одновременно за несколькими аудиториями

№	Ситуация	Аномалия	Реакция	Примечание
1	2	3	4	5
			аудиторию в это время»	
6	Вывод мероприятий	Дата мероприятия указана в неверном формате или отсутствует	Программа выводит предупреждение: «Неверный формат даты мероприятия» и не отображает событие	Дата должна быть в формате ГГГГ-ММ-ДД
7	Определение типа недели	Тип недели не выбран	Программа не позволяет продолжить генерацию, выводит сообщение: «Выберите тип недели — числитель или знаменатель»	Тип недели необходим для корректной генерации расписания
8	Запрос от Telegram-бота	Указана несуществующая группа или преподаватель	Бот отправляет сообщение: «Информация не найдена. Проверьте корректность запроса»	Обрабатываются только существующие группы и преподаватели
9	Команда /auditory или /teacher в нерабочий день	На текущий день нет расписания или распределений	Бот отправляет сообщение: «На сегодня занятий нет» или «Аудиторный фонд не сформирован»	Проверяется дата и наличие данных перед отправкой ответа
10	Запрос к боту при недоступности API	Сервер не отвечает или недоступен	Бот отправляет сообщение: «Ошибка получения данных. Повторите попытку позже»	При недоступности API бот не завершает работу, а сообщает об ошибке

2.3. Проектирование

2.3.1. Схема архитектуры приложения

Компонент пользовательский интерфейс виден пользователю и взаимодействует с ним. В зависимости от действия пользователя

осуществляются запросы к базе данных, которая возвращает ответы на запрос в виде данных.

На Рисунке 16 представлена архитектурная схема приложения.

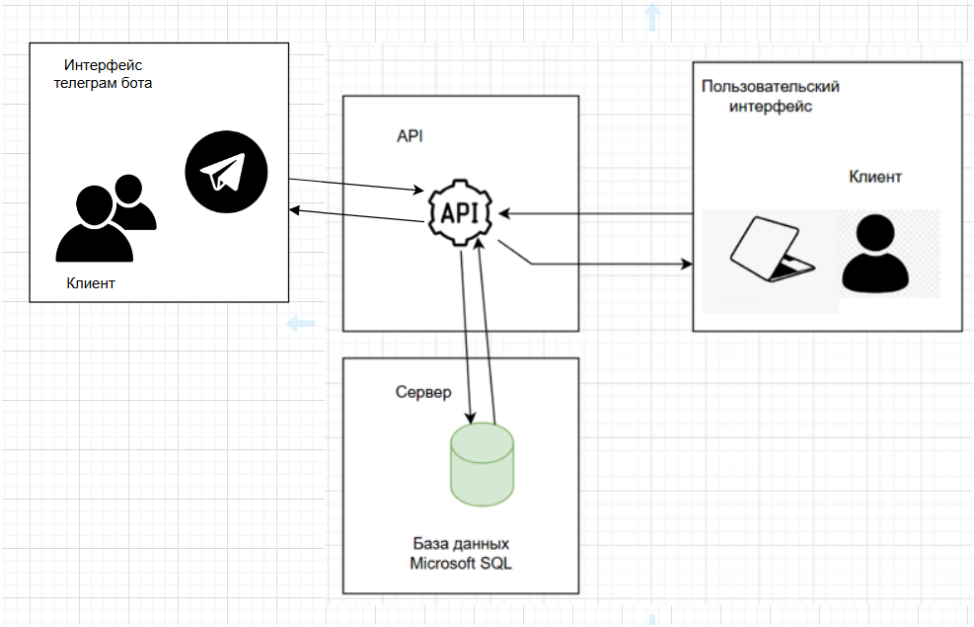

Рисунок 16 - Архитектурная схема программного комплекса

2.3.2. Логическая схема данных

Разработанная логическая модель базы данных, представленная на Рисунке 17, послужила основой для реализации логики манипуляции данными в проектируемой базе данных.

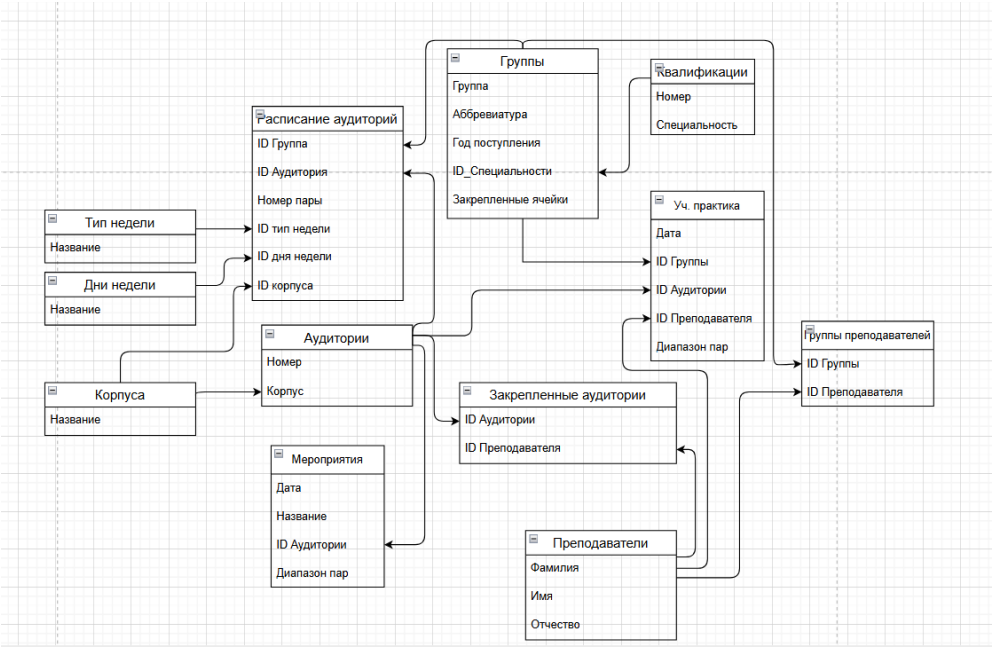

Рисунок 17 – Логическая схема данных

2.3.3. Физическая схема данных

Разработанная физическая модель базы данных, изображенная на Рисунке 18, послужила основой для реализации логики хранения и защиты данных в проектируемой базе данных.

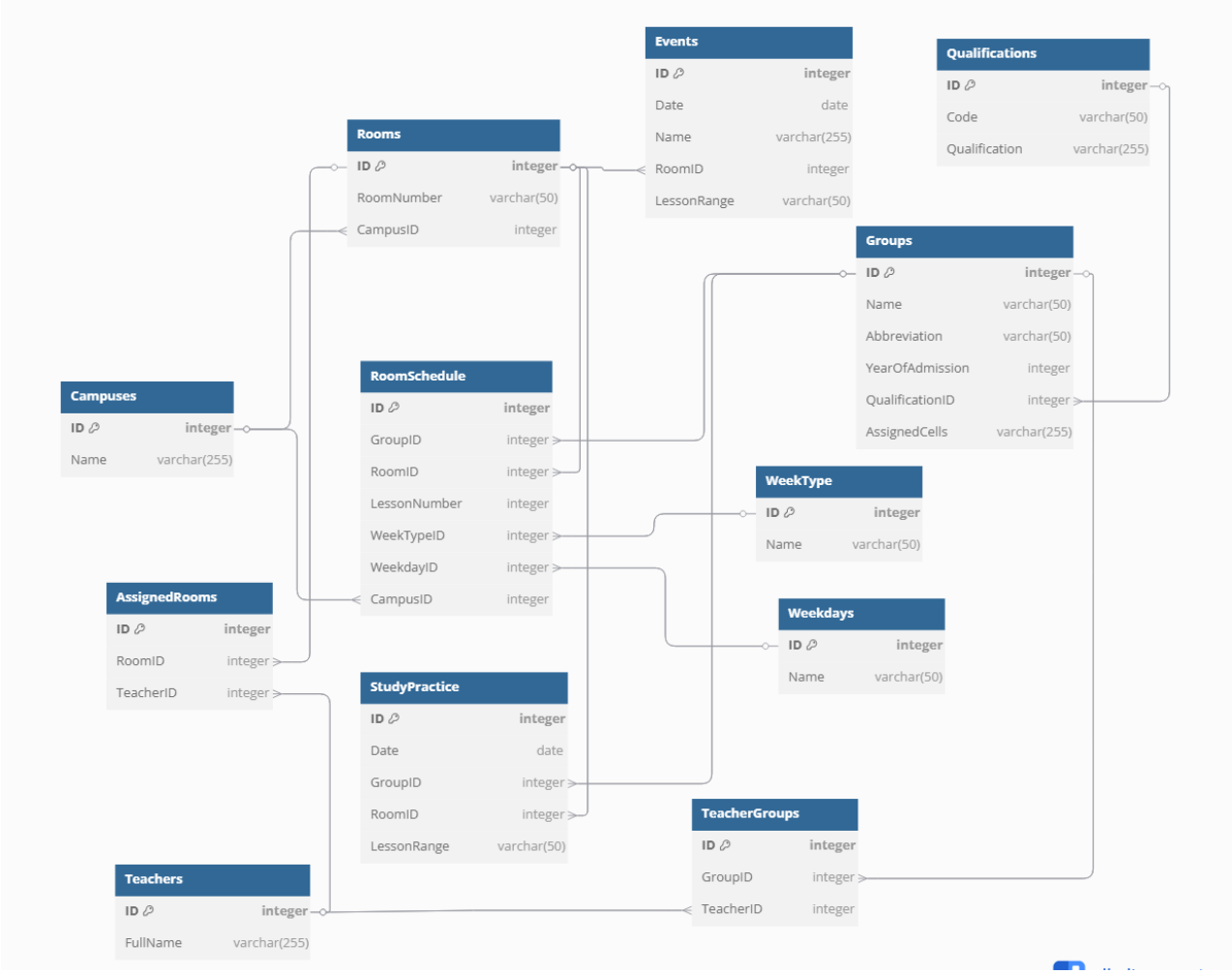

Рисунок 18 – Физическая схема данных

В проектируемой базе данных, разработанной для автоматизации процесса формирования аудиторного фонда колледжа, выделено 12 сущностей (таблиц). Каждая таблица содержит атрибуты, необходимые для хранения информации об учебных группах, преподавателях, аудиториях, корпусах, квалификациях, расписании, практике и мероприятиях.

Во всех таблицах предусмотрен уникальный идентификатор (ID), выступающий в роли первичного ключа. Этот атрибут используется для построения связей между таблицами с помощью внешних ключей, что

обеспечивает целостность и логическую связанность данных. Такая структура базы данных позволяет эффективно обрабатывать входную информацию и использовать её при генерации итогового Excel-документа аудиторного фонда.

В таблице 7 представлен словарь данных реализуемой базы данных для данного приложения.

Таблица 7 – Словарь базы данных

Ключ	Поле	Тип данных	Обязательность заполнения	Описание
1	2	3	4	5
Таблица «Campuses»				
PK	CampusID	int	Not null	Уникальный идентификатор корпуса
–	Name	NVARCHAR(255)	Not null	Название корпуса (например, Нахимовский, Нежинская)
Таблица «Rooms»				
PK	RoomID	int	Not null	Уникальный идентификатор аудитории
–	RoomNumber	NVARCHAR(50)	Not null	Номер аудитории
FK	CampusID	int	Not null	Внешний ключ, указывающий на корпус (Campuses)
Таблица «Qualifications»				
PK	QualificationID	int	Not null	Уникальный идентификатор квалификации
–	Code	NVARCHAR(50)	Not null	Код специальности
–	Qualification	NVARCHAR(255)	Not null	Полное название квалификации
Таблица «Groups»				
PK	GroupID	int	Not null	Уникальный идентификатор группы
–	Name	NVARCHAR(50)	Not null	Название группы
–	Abbreviation	NVARCHAR(50)	Not null	Сокращение группы

Ключ	Поле	Тип данных	Обязательность заполнения	Описание
1	2	3	4	5
–	YearOfAdmission	int	Not null	Год поступления
FK	QualificationID	int	Not null	Внешний ключ, указывающий на квалификацию
–	AssignedCells	NVARCHAR(255)	Null	Диапазоны ячеек в шаблоне Excel
Таблица «Teachers»				
PK	TeacherID	int	Not null	Уникальный идентификатор преподавателя
–	FullName	NVARCHAR(255)	Not null	ФИО преподавателя
Таблица «AssignedRooms»				
PK	AssignedRoomID	int	Not null	Уникальный идентификатор привязки аудитории
FK	RoomID	int	Not null	Внешний ключ на таблицу Rooms
FK	TeacherID	int	Not null	Внешний ключ на таблицу Teachers
Таблица «Weekdays»				
PK	WeekdayID	int	Not null	Уникальный идентификатор дня недели
–	Name	NVARCHAR(50)	Not null	Название дня недели (например, Понедельник)
Таблица «WeekType»				
PK	WeekTypeID	int	Not null	Уникальный идентификатор типа недели
–	Name	NVARCHAR(50)	Not null	Тип недели (Числитель/Знаменатель)
Таблица «RoomSchedule»				
PK	ScheduleID	int	Not null	Уникальный идентификатор записи расписания
FK	GroupID	int	Not null	Внешний ключ на группу

Ключ	Поле	Тип данных	Обязательность заполнения	Описание
1	2	3	4	5
FK	RoomID	int	Not null	Внешний ключ на аудиторию
–	LessonNumber	int	Not null	Номер пары (1–7)
FK	WeekTypeID	int	Not null	Тип недели (числитель/знаменатель)
FK	WeekdayID	int	Not null	День недели
FK	CampusID	int	Not null	Корпус, в котором проходит пара
Таблица «StudyPractice»				
PK	PracticeID	int	Not null	Уникальный идентификатор учебной практики
–	Date	DATE	Not null	Дата проведения практики
FK	GroupID	int	Not null	Внешний ключ на группу
FK	RoomID	int	Not null	Внешний ключ на аудиторию
–	LessonRange	NVARCHAR(50)	Null	Диапазон пар (например, 1-4)
Таблица «Events»				
PK	EventID	int	Not null	Уникальный идентификатор мероприятия
–	Date	DATE	Not null	Дата мероприятия
–	Name	NVARCHAR(255)	Not null	Название мероприятия
FK	RoomID	int	Not null	Внешний ключ на аудиторию
–	LessonRange	NVARCHAR(50)	Null	Диапазон пар
Таблица «TeacherGroups»				
PK	TeacherGroupID	int	Not null	Уникальный идентификатор связи преподавателя с группой
FK	GroupID	int	Not null	Внешний ключ на группу
FK	TeacherID	int	Not null	Внешний ключ на преподавателя

2.3.4. Структурная схема

На Рисунке 19 представлен общий план структурной схемы приложения, в которой графически показано взаимодействие представлений, моделей и их взаимодействие друг с другом.

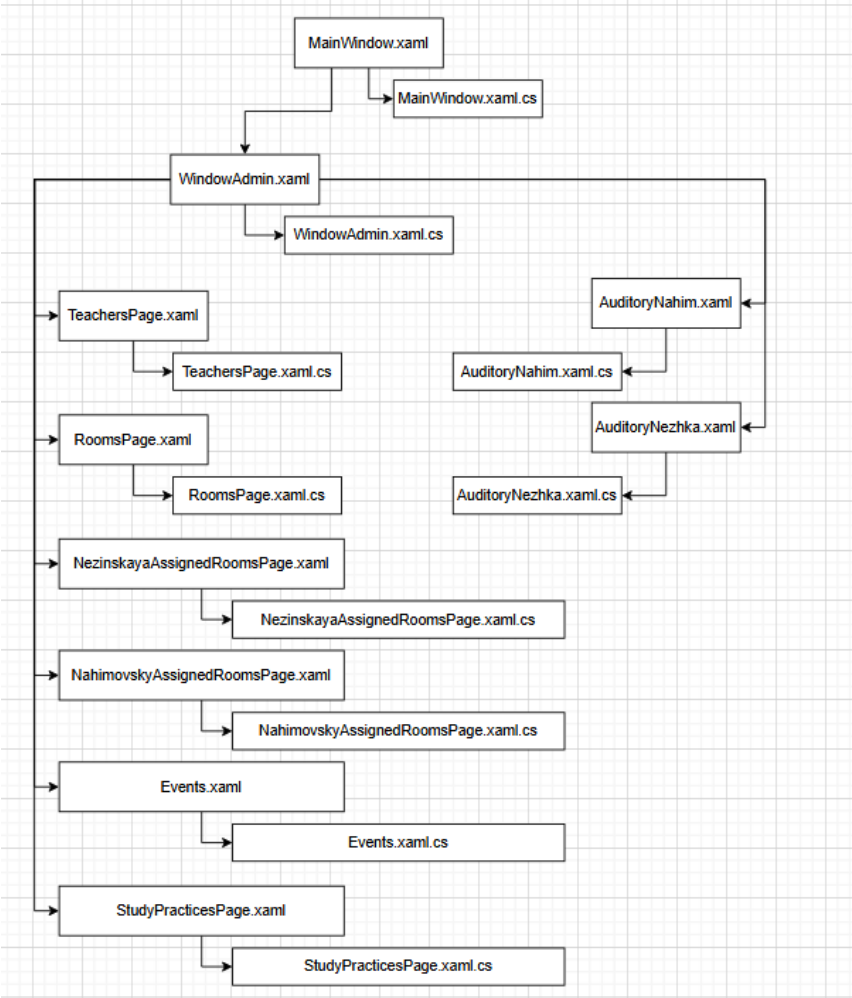

Рисунок 19 – Структурная схема

В таблице 8 представлено описание каждого модуля структурной схемы.

Таблица 8 – Описание модулей

№	Наименование модуля	Описание
1	2	3
Окна		
1	MainWindow.xaml	Окно авторизации пользователя
2	MainWindow.xaml.cs	Обработка логики авторизации и перехода к роли администратора
3	WindowAdmin.xaml	Главное окно администратора с навигацией

№	Наименование модуля	Описание
1	2	3
4	WindowAdmin.xaml.cs	Реализация переходов между страницами и обработка событий администратора
5	TeachersPage.xaml	Разметка страницы со списком преподавателей
6	TeachersPage.xaml.cs	Загрузка, отображение, добавление и редактирование преподавателей
7	RoomsPage.xaml	Разметка страницы со списком аудиторий
8	RoomsPage.xaml.cs	Загрузка, отображение, добавление и редактирование аудиторий
9	NezinskayaAssignedRoomsPage.xaml	Разметка страницы привязок аудиторий корпуса Нежинская
10	NezinskayaAssignedRoomsPage.xaml.cs	Загрузка привязок, проверка дубликатов, отправка на сервер
11	NahimovskyAssignedRoomsPage.xaml	Разметка страницы привязок аудиторий корпуса Нахимовский
12	NahimovskyAssignedRoomsPage.xaml.cs	Функционал аналогичен Нежинской, но для другого корпуса
13	AuditoryNahim.xaml	Разметка таблицы аудиторного фонда (Нахимовский)
14	AuditoryNahim.xaml.cs	Генерация и отображение Excel-файла с аудиториями по расписанию
15	AuditoryNezhka.xaml	Разметка таблицы аудиторного фонда (Нежинская)
16	AuditoryNezhka.xaml.cs	Генерация и отображение Excel-файла с аудиториями по расписанию
17	Events.xaml	Разметка страницы управления мероприятиями
18	Events.xaml.cs	Добавление, редактирование и удаление мероприятий
19	StudyPracticesPage.xaml	Разметка страницы управления учебной практикой
20	StudyPracticesPage.xaml.cs	Добавление, редактирование и удаление записей практики

2.3.5. Функциональная схема

На Рисунке 20 представлена функциональная схема настольного приложения, на котором графически изображены возможности пользователя в процессе эксплуатации.

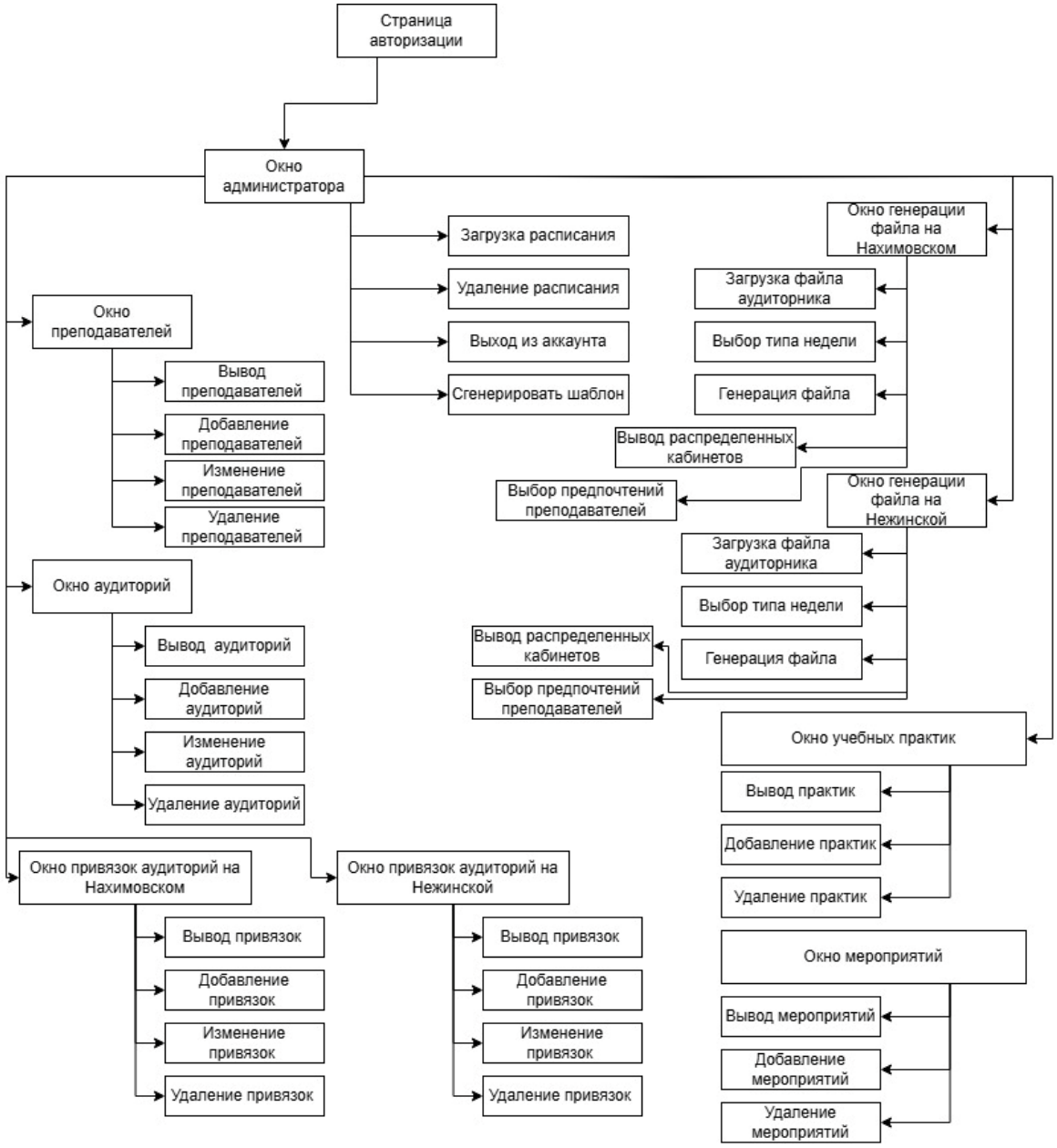

Рисунок 20 – Функциональная схема

Telegram-бот в рамках разработанной системы реализует вспомогательный функционал для пользователей, не обладая графическим пользовательским интерфейсом. Он предоставляет доступ к информации о расписании, аудиториях и преподавателях через команды и текстовые ответы. Бот не входит в структуру окон WPF-приложения и

не отображается в функциональной схеме интерфейса, поскольку взаимодействие с ним осуществляется исключительно через Telegram, без визуальных окон. Его архитектурная роль отражена в диаграмме взаимодействия компонентов, где он представлен как отдельный клиент, обращающийся к API.

2.3.6. Диаграмма классов

На Рисунках 21-26 представлена диаграмма классов приложения и телеграм бота по частям. На ней графически изображены классы, которые состоят из полей, методов и свойств.

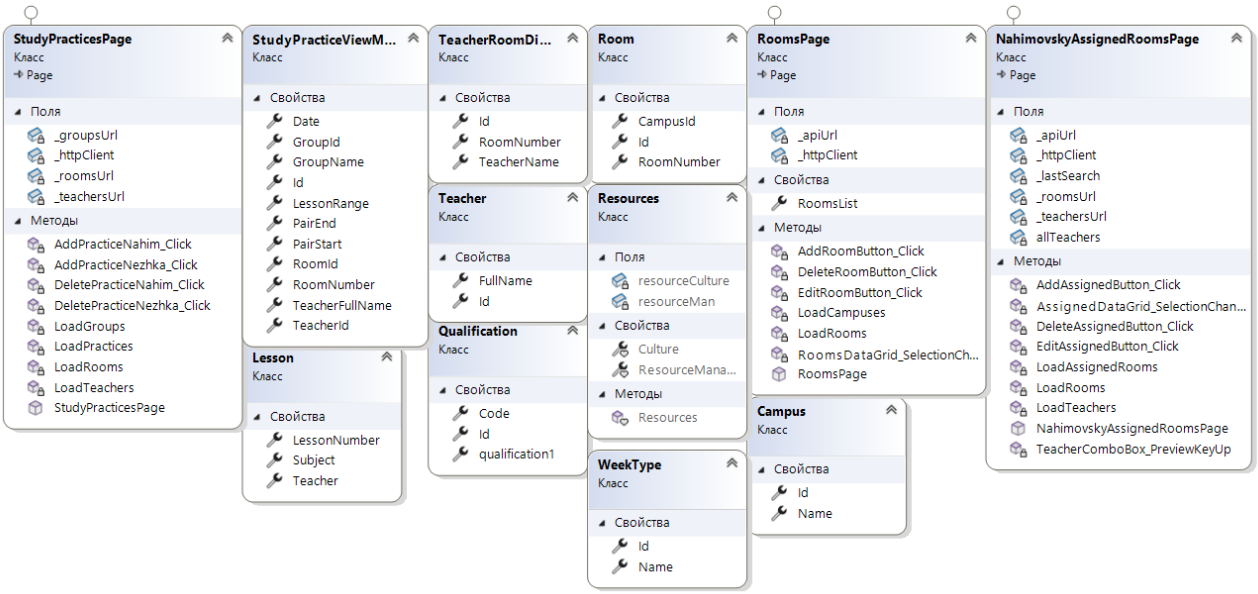

Рисунок 21 – Диаграмма классов 1 часть

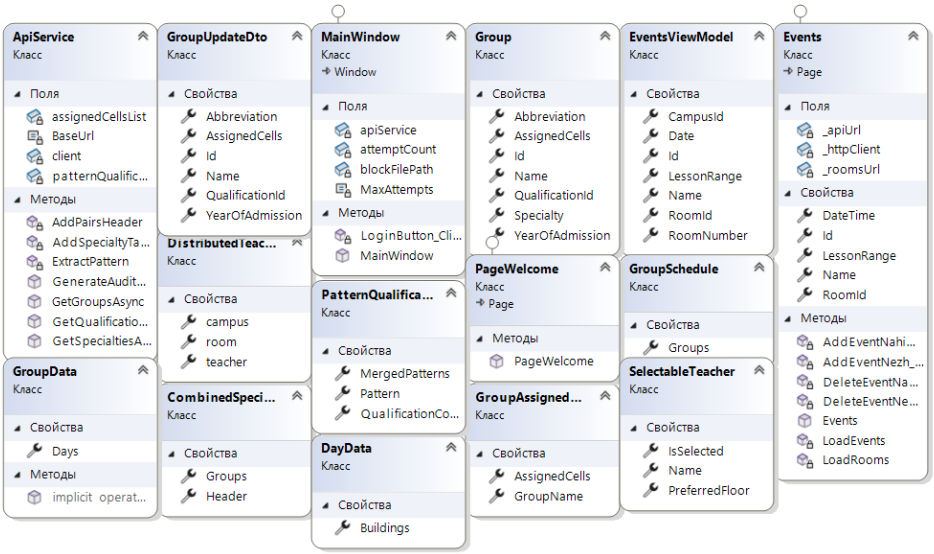

Рисунок 22 – Диаграмма классов 2 часть

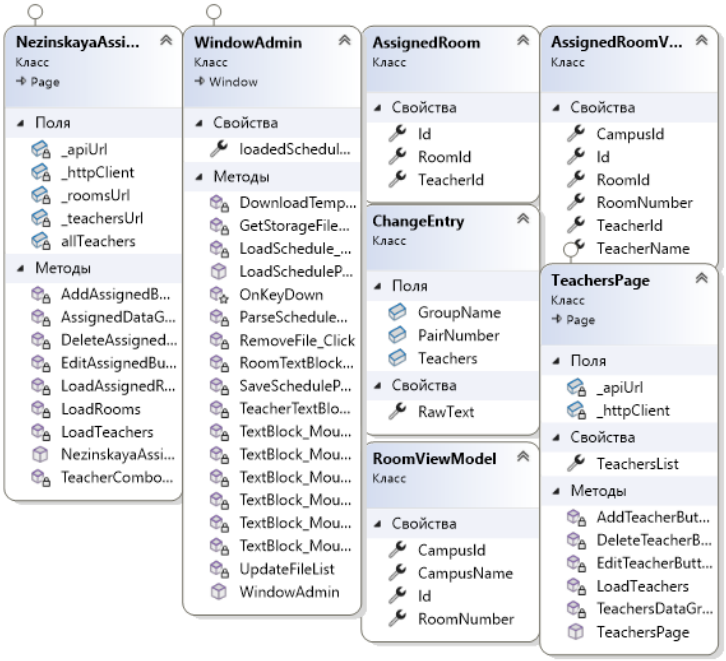

Рисунок 23 – Диаграмма классов 3 часть

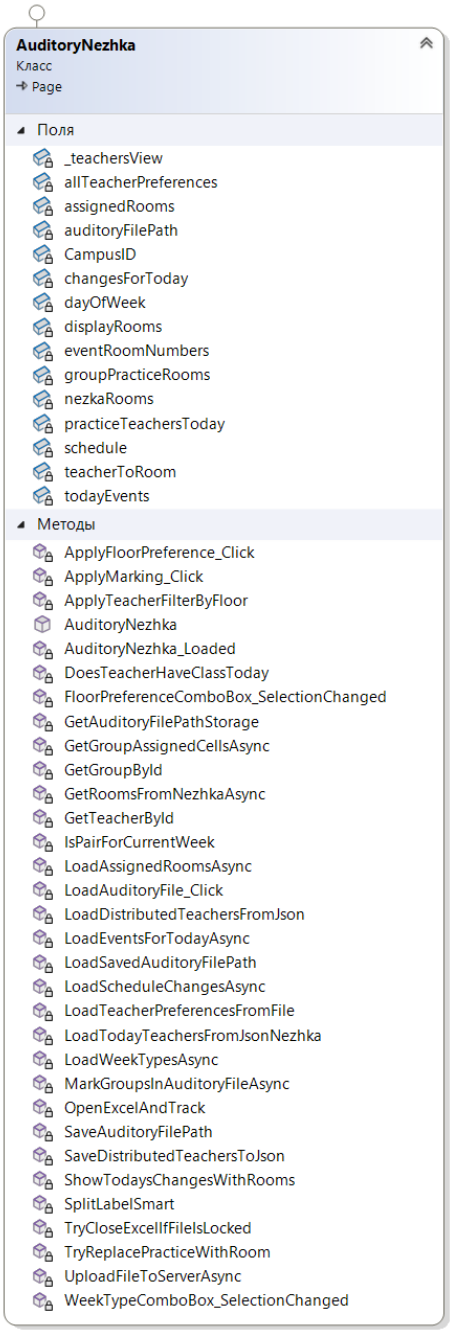

Рисунок 24 – Диаграмма классов 4 часть

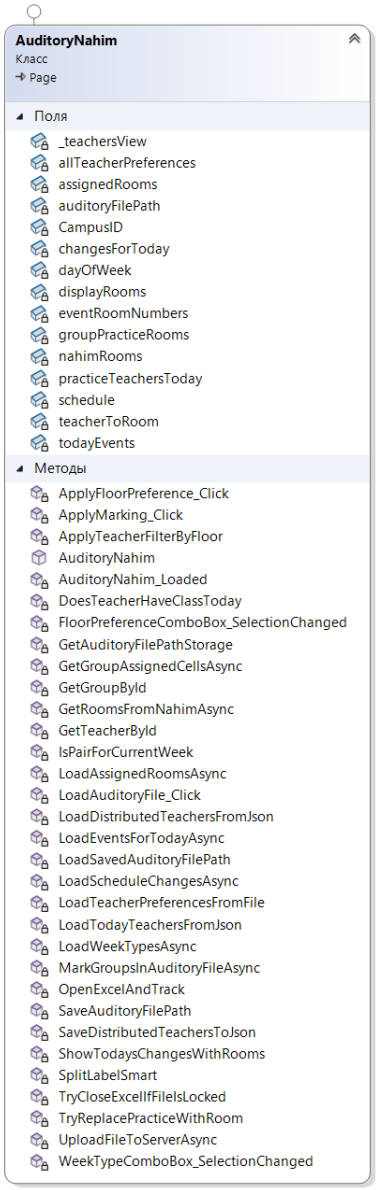

Рисунок 25 – Диаграмма классов 5 часть

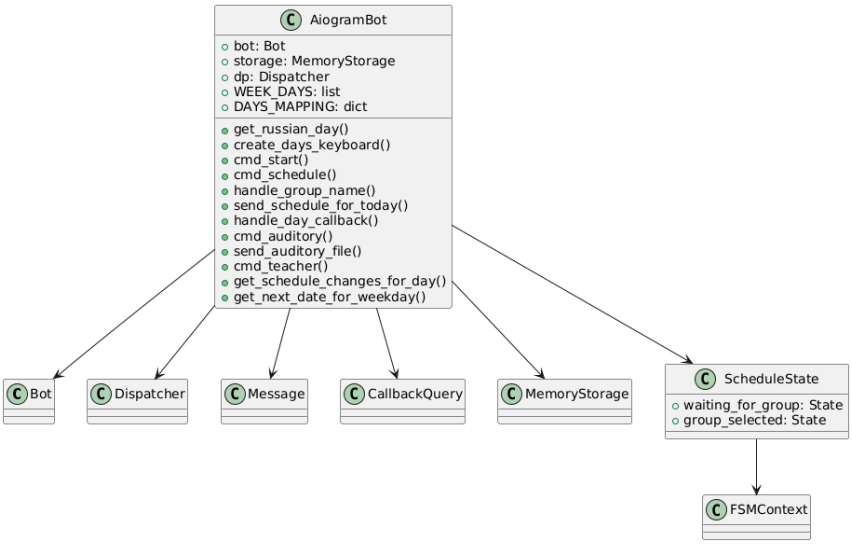

Рисунок 26 – Диаграмма классов телеграмм бота

В таблице 9 представлено описание некоторых классов продемонстрированной диаграммы классов.

Таблица 9 – Описание диаграммы классов

№	Наименование	Описание
1	2	3
1	MainWindow.xaml.cs	Класс, отвечающий за обработку логики авторизации и переход к роли администратора
2	WindowAdmin.xaml.cs	Класс, отвечающий за навигацию между модулями и обработку событий в интерфейсе администратора
3	TeachersPage.xaml.cs	Класс, отвечающий за загрузку, отображение, добавление, редактирование и удаление преподавателей
4	RoomsPage.xaml.cs	Класс, отвечающий за управление списком аудиторий: загрузка, отображение, добавление, редактирование и удаление
5	NezinskayaAssignedRoomsPage.xaml.cs	Класс, отвечающий за загрузку и отображение закреплённых аудиторий на Нежинской, проверку дубликатов и отправку данных на сервер
6	NahimovskyAssignedRoomsPage.xaml.cs	Класс, отвечающий за аналогичную работу с закреплёнными аудиториями, но для корпуса Нахимовский
7	AuditoryNahim.xaml.cs	Класс, отвечающий за генерацию Excel-файла по корпусу Нахимовский на основе расписания
8	AuditoryNezhka.xaml.cs	Класс, отвечающий за генерацию Excel-файла по корпусу Нежинская на основе расписания
9	Events.xaml.cs	Класс, отвечающий за управление мероприятиями: добавление, редактирование и удаление
10	StudyPracticesPage.xaml.cs	Класс, отвечающий за добавление, редактирование и удаление записей учебной практики

2.3.7. Схема тестирования

На рисунке 27 представлена схема тестирования, по которой проводилось тестирование приложения.

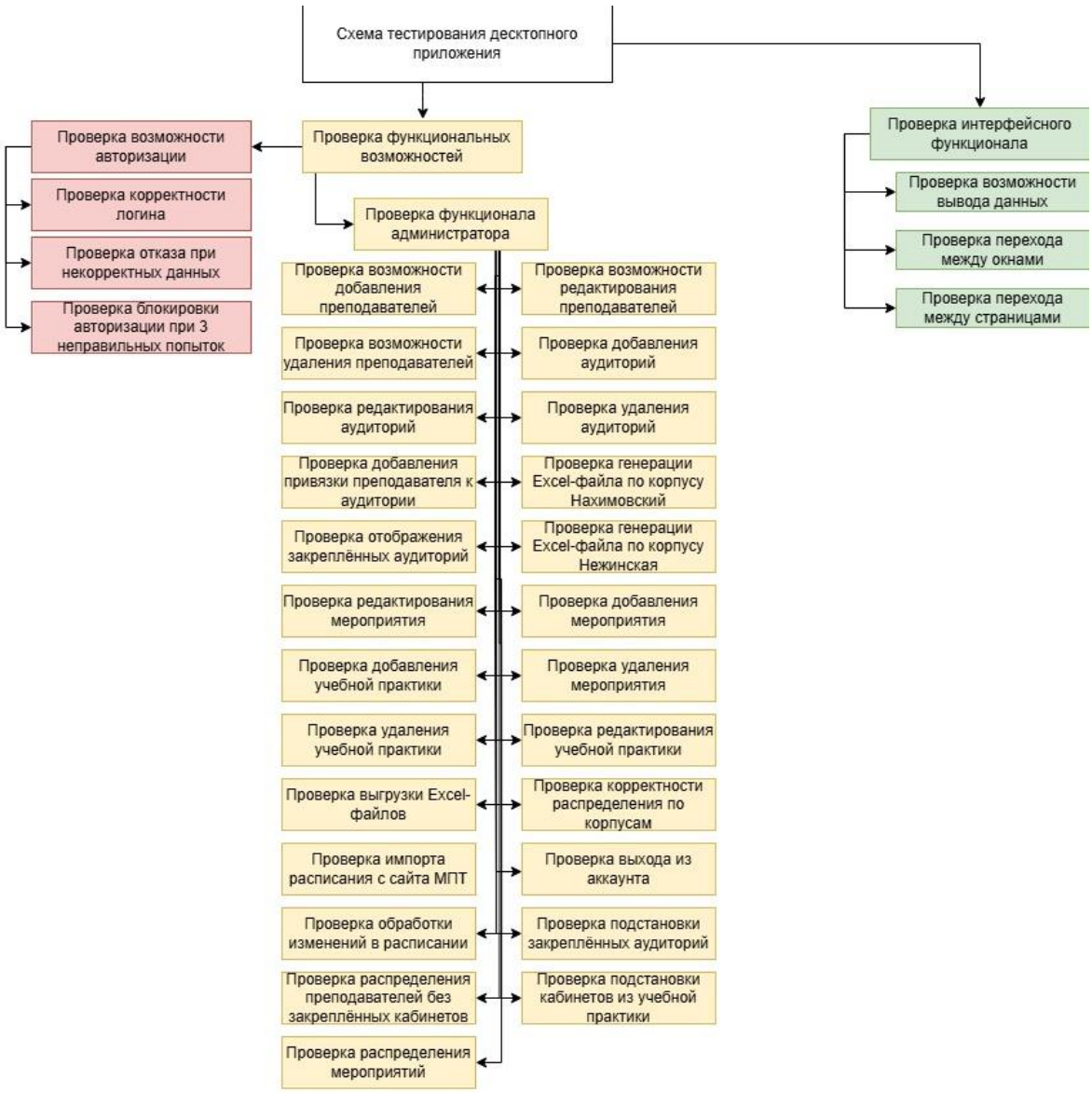

Рисунок 27 – Схема тестирования десктопного приложения

На рисунке 28 представлена схема тестирования Telegram-бота:

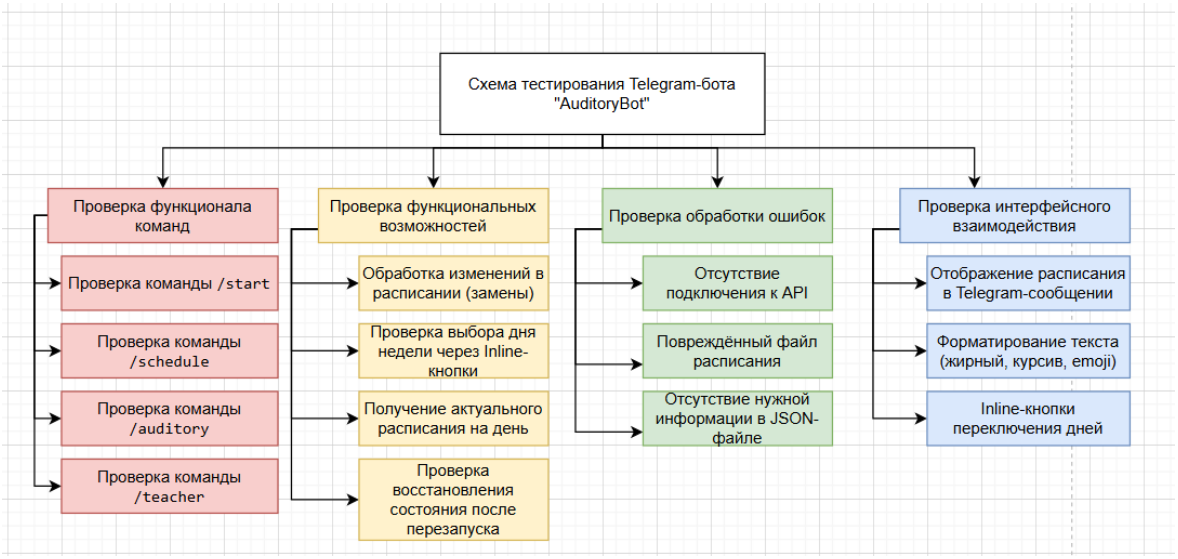

Рисунок 28 – Схема тестирования Telegram-бота

2.3.8. Схема пользовательского интерфейса

На рисунке 29 представлена схема пользовательского интерфейса административного приложения.

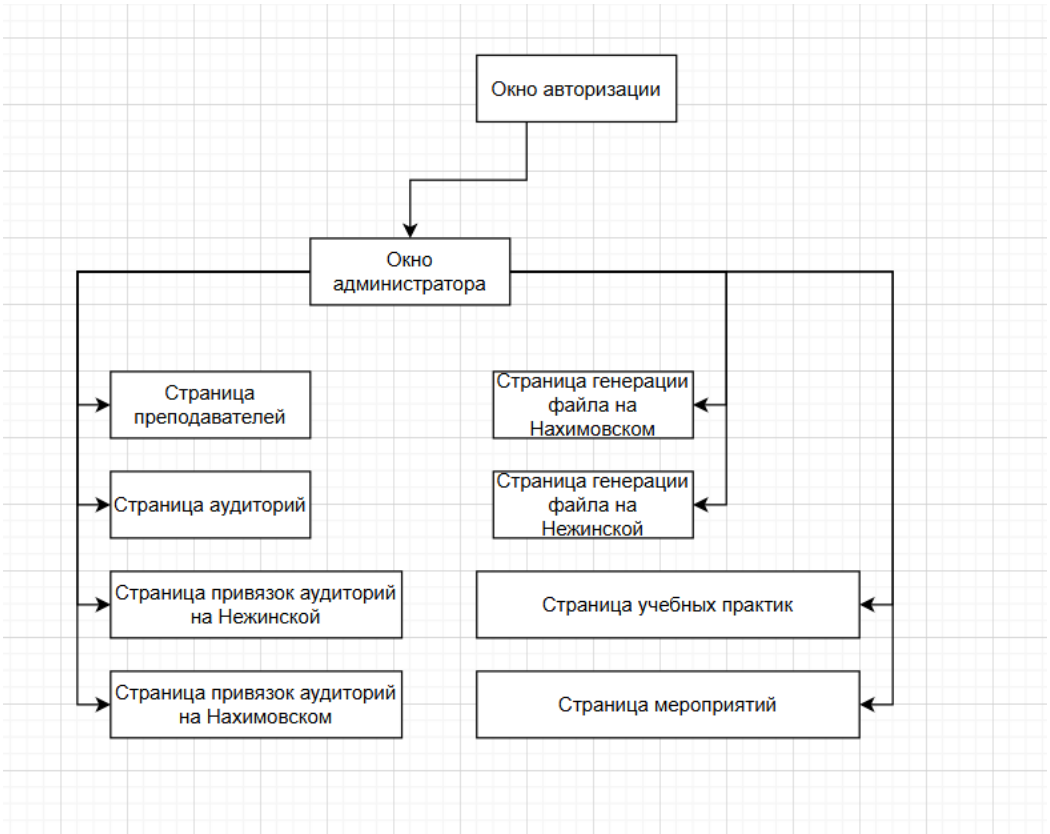

Рисунок 29 – Схема пользовательского интерфейса

2.4. Результат работы программы

В результате поставленной задачи был разработан программный комплекс в виде десктопного приложения и Telegram-бота.

На Рисунках 19-20 представлен некоторый результат работы программы. Более развёрнуто результат работы программы описан в Приложении Г «Руководство пользователя».

На Рисунке 30 представлен экран администратора.

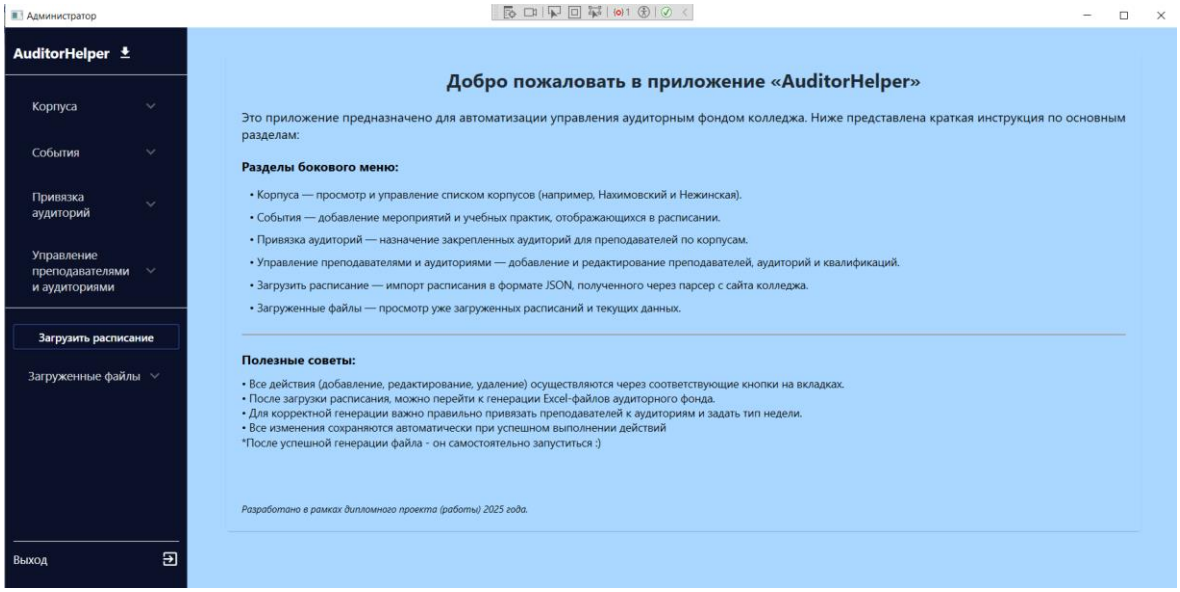

Рисунок 30 – Окно администратора с навигацией

На Рисунке 31 представлена страница для привязки аудиторий к преподавателям.

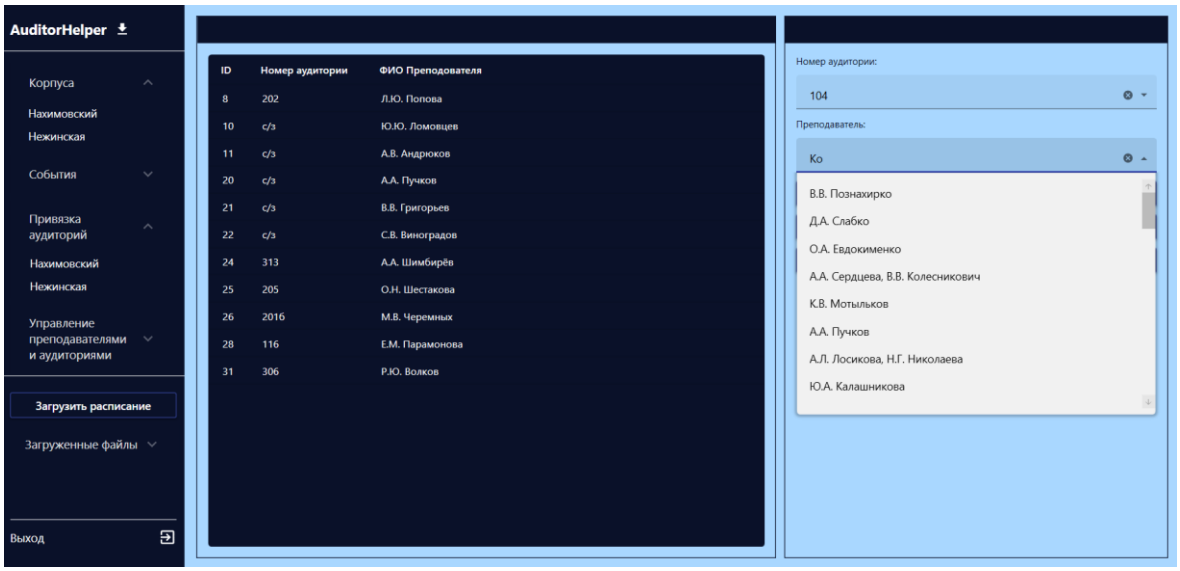

Рисунок 31 – Окно с выводом данных

3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1. Инструментальные средства

Разработка десктопного приложения для автоматизации формирования аудиторного фонда колледжа осуществлялась с использованием технологии WPF (Windows Presentation Foundation) в среде Visual Studio 2022. WPF позволяет создавать гибкий и масштабируемый пользовательский интерфейс, а Visual Studio предоставляет обширный инструментарий — от XAML-дизайнера до встроенного дебаггера и расширений NuGet.

Клиентская часть реализована на языке программирования C# — строго типизированном объектно-ориентированном языке, предоставляющем удобный синтаксис, высокую безопасность выполнения и широкую поддержку библиотек. Логика приложения структурирована с разделением по слоям: данные, логика и представление.

Для взаимодействия с базой данных используется API, реализованный на ASP.NET Core. Это даёт возможность централизованно обрабатывать бизнес-логику, снизить связность компонентов и повысить безопасность за счёт серверной фильтрации и валидации.

В качестве СУБД использовалась Microsoft SQL Server, благодаря своей стабильности и производительности. Управление структурой таблиц и данными осуществлялось через SQL Server Management Studio (SSMS). Для сериализации и десериализации данных применялась библиотека Newtonsoft.Json, а для генерации Excel-документов — ClosedXML.

Дополнительно реализован Telegram-бот на Python с использованием библиотеки Aiogram. Бот взаимодействует с API и сервером FirstByte, обеспечивая удобный доступ к актуальному расписанию и аудиторной информации через команды.

Такой набор инструментов позволяет легко масштабировать проект, добавлять новые функции (например, уведомления или экспорт статистики) и интегрировать внешние сервисы.

3.2. Отладка программы

Отладка велась с использованием средств Visual Studio: точки останова, отслеживание переменных и стеков вызовов, просмотр локальных и глобальных значений. Это позволяло точно выявлять участки, вызывающие сбои, и своевременно их исправлять.

При интеграции с внешними API и базой данных возникали проблемы подключения, связанные с неверными строками конфигурации, отсутствием прав доступа или сбоем на стороне сервера. Эти ошибки обрабатывались через конструкции try...catch с выводом подробных сообщений пользователю.

Для Telegram-бота отладка проводилась через логирование, асинхронный режим обработки событий и тестирование сценариев в закрытых чатах. Дополнительно применялись методы имитации API-ответов для проверки логики бота без обращения к серверу.

Источниками информации при устранении проблем служили официальные документы по .NET, SQL Server, Aiogram и сообщества Stack Overflow, Microsoft Learn и другие.

3.3. Защитное программирование

В программном коде реализованы механизмы защиты от ошибок и неожиданных сбоев приложения, используя конструкции if...else и try...catch.

На Рисунке 32 представлен блок if…else.

```
if (response.IsSuccessStatusCode)
{
    var rooms = await response.Content.ReadAsAsync<List<Room>>();
    nahimRooms = rooms.Where(r => r.CampusId == 1).ToList();
    return nahimRooms;
}
else
{
    MessageBox.Show("Ошибка при получении списка аудиторий.");
    return new List<Room>();
}
```

Рисунок 32 – Блок if…else

На Рисунке 33 – блок try…catch.

```
try
{
    var selectedRoom = (RoomViewModel)RoomComboBoxNezh.SelectedItem;

    var newEvent = new EventsViewModel
    {
        RoomId = selectedRoom.Id.Value,
        CampusId = selectedRoom.CampusId,
        Date = EventDateNezh.SelectedDate.Value.Date,
        Name = EventNameNezh.Text,
        LessonRange = LessonRangeNezh.Text
    };

    var json = JsonConvert.SerializeObject(newEvent, new JsonSerializerSettings
    {
        DateFormatString = "yyyy-MM-dd",
        NullValueHandling = NullValueHandling.Ignore
    });

    Console.WriteLine($"Отправляем JSON: {json}");

    var content = new StringContent(json, Encoding.UTF8, "application/json");
    var response = await _httpClient.PostAsync(_apiUrl, content);
    response.EnsureSuccessStatusCode();

    MessageBox.Show("Мероприятие успешно добавлено!");
    LoadEvents(2, EventsDataGridNezh); // Загружаем данные для Нежинской (CampusId = 2)
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка при добавлении мероприятия: {ex.Message}");
}
```

Рисунок 33 – Try…catch

3.4. Характеристики программы

Разработанное приложение работает на ОС Windows 10 и выше. Telegram-бот не зависит от платформы и может быть запущен на любом устройстве с установленным Python и доступом к сети. Минимальные

требования: .NET 6, установленный SQL Server (или подключение к удалённому серверу), доступ к API и Telegram.

Характеристики и состав модулей приведены в Приложении А «Текст программы» в таблице 1 «Модули».

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта было разработано программное решение для автоматизированного формирования аудиторного фонда колледжа, включающее десктопное приложение и Telegram-бот. Проект охватывает полный цикл обработки расписания — от импорта и анализа данных до генерации Excel-файла и интеграции с пользовательскими интерфейсами.

На этапе анализа были выделены ключевые проблемы существующей системы: высокая трудоёмкость ручного составления фонда, вероятность ошибок и дублирующихся закреплений, отсутствие единого хранилища данных. Это позволило сформировать обоснованные требования к функциональности и архитектуре системы.

В рамках проекта были реализованы следующие компоненты:

- Интуитивный WPF-интерфейс для администратора с функциями загрузки расписания, редактирования преподавателей, аудиторий, практик и мероприятий;
- База данных Microsoft SQL Server, отражающая реальную структуру колледжа: группы, аудитории, преподаватели, закрепления и события;
- API на ASP.NET Core для обработки запросов, валидации и взаимодействия между клиентской частью и сервером;
- Модуль импорта расписания и изменений с сайта колледжа (mpt.ru) в формате JSON;
- Генерация Excel-документа с учётом корпусов, типа недели и текущего дня;
- Автоматическое распределение преподавателей по аудиториям (в том числе по предпочтениям и закреплениям);

• Интеграция Telegram-бота, позволяющая пользователям получать расписание, файлы аудиторного фонда и искать кабинеты преподавателей по фамилии.

На этапе тестирования проведена проверка следующих аспектов:

• Работоспособности команд Telegram-бота;

• Корректности импорта расписания и генерации Excel-файлов;

• Реакции интерфейса на пользовательские ошибки;

• Выполнения всех операций с данными (добавление, изменение, удаление);

• Устойчивости при сбоях подключения к API или при некорректных входных данных.

Все выявленные ошибки были своевременно устранены. Приложение показало стабильную работу в рамках типичных сценариев использования.

Разработка проекта позволила автору углубить практические навыки в областях:

• Объектно-ориентированного и событийно-ориентированного программирования на C# и Python;

• Создания клиент-серверной архитектуры;

• Интеграции Telegram-ботов с API и базами данных;

• Автоматизации документооборота на основе структурированных данных расписания.

Разработанная система соответствует поставленным целям, обеспечивает прозрачность и автоматизацию формирования аудиторного фонда, а также легко адаптируется под новые требования колледжа.

Возможные направления развития:

• Расширение Telegram-бота: добавление расписания по преподавателю, поддержка inline-режима, фильтрация по типу недели;

• Разграничение прав доступа: методист, преподаватель, гость;

- Поддержка мобильной версии или web-интерфейса;
- Хранение истории изменений закреплений и генераций фонда;
- Автоматическая отправка уведомлений в Telegram при изменениях в расписании.

СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

1. WPF в .NET. Работа с XAML и визуальным интерфейсом [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/overview/, дата обращения: 22.04.2025.

2. Microsoft SQL Server. Документация по СУБД [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/sql/sql-server/, дата обращения: 24.04.2025.

3. ASP.NET Core. Руководство для разработчиков [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/, дата обращения: 29.04.2025.

4. Telegram Bot API. Документация Telegram [Электронный ресурс]. – Режим доступа: https://core.telegram.org/bots/api, дата обращения: 30.04.2025.

5. Aiogram 3.0 Documentation [Электронный ресурс]. – Режим доступа: https://docs.aiogram.dev, дата обращения: 25.04.2025.

6. Работа с Excel-файлами через библиотеку ClosedXML [Электронный ресурс]. – Режим доступа: https://github.com/ClosedXML/ClosedXML, дата обращения: 27.04.2025.

7. BeautifulSoup: библиотека для парсинга HTML [Электронный ресурс]. – Режим доступа: https://www.crummy.com/software/BeautifulSoup/bs4/doc/, дата обращения: 03.05.2025.

8. Entity Framework Core. Руководство от Microsoft [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/ef/core/, дата обращения: 28.04.2025.

9. Stack Overflow — международный форум разработчиков [Электронный ресурс]. – Режим доступа: https://stackoverflow.com, дата обращения: 21.04.2025.

10. Telegram-бот на aiogram (канал Vladislav Markeev) [Электронный ресурс]. – Режим доступа: https://www.youtube.com/playlist?list=PLQJx1kckE_AiGzkfrY6FkBr7kLh-FVxtn, дата обращения: 01.05.2025.

11. Создание WPF-интерфейсов с Material Design [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=0MkRpn5wX9g, дата обращения: 05.05.2025.

12. Основы SQL Server Management Studio [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=c5V9a8ThCm0, дата обращения: 10.05.2025.

13. JetBrains Academy: Курсы по C# и WPF [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/academy/, дата обращения: 11.05.2025.

14. C# 9.0 и .NET 5. Профессиональное программирование / Э. Троелсен, Ф. Джепикс [Печатный ресурс]. – СПб.: Питер, 2021. – 928 с.

15. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 / Д. Рихтер [Печатный ресурс]. – СПб.: Питер, 2013. – 896 с.

16. Telegram-боты. Создание и продвижение / С. Ульянов [Печатный ресурс]. – СПб.: БХВ-Петербург, 2021. – 312 с.

17. Основы тестирования программного обеспечения / Г. Майерс [Печатный ресурс]. – М.: Вильямс, 2020. – 256 с.