

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

К защите допустить:

Заведующая кафедрой ПОИТ

_____ Н. В. Лапицкая

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

**МОБИЛЬНОГО ПРИЛОЖЕНИЯ «ЭЛЕКТРОННЫЙ ДНЕВНИК»
ДЛЯ ОБРАЗОВАТЕЛЬНЫХ УЧРЕЖДЕНИЙ С ИСПОЛЬЗОВАНИЕМ
ТЕХНОЛОГИИ .NET MAUI**

БГУИР ДП 1-40 01 01 01 036 ПЗ

Студент

Д.А. Дубовский

Руководитель

Е.Г. Мелких

Консультанты:
*от кафедры ПОИТ
по экономической части*

Е.Г. Мелких
А.А. Горюшкин

Нормоконтролер

А.А. Грибович

Рецензент

Минск 2025

РЕФЕРАТ

Мобильного приложения «Электронный дневник» для образовательных учреждений с использованием технологии .NET MAUI: дипломный проект / Д.А. Дубовский. — Минск: БГУИР, 2025.

Дипломный проект выполнен на 7 листах формата А1 с пояснительной запиской на 101 странице, включающей одно приложение с исходным кодом программного средства. Пояснительная записка содержит 7 глав, 17 рисунков, 7 таблиц, 4 формулы и 16 литературных источников.

Цель проекта — разработка мобильного приложения «Электронный дневник», предназначенного для замены бумажных журналов и дневников в образовательных учреждениях. Приложение обеспечивает удобный доступ к учебным данным, сокращает временные затраты на их ведение и повышает мотивацию учащихся за счёт прозрачности и оперативности отображения результатов.

Основные задачи:

- 1) Анализ существующих решений и формирование требований к функциональности приложения.
- 2) Проектирование архитектуры системы с поддержкой работы нескольких школ в рамках единой платформы.
- 3) Реализация мобильного приложения с интуитивным интерфейсом для отображения расписания, оценок, домашних заданий и уведомлений.
- 4) Обеспечение безопасности хранения и передачи данных.
- 5) Тестирование приложения и оценка его соответствия функциональным требованиям.

Используемые технологии:

- Язык программирования: C#.
- Фреймворк: .NET MAUI для кроссплатформенной разработки (Android, iOS).
- База данных: Oracle.
- Дополнительные инструменты: Visual Studio, PL/SQL Developer, .NET MAUI Community Toolkit.

Результаты работы:

- Разработано мобильное приложение с ролевым доступом (ученик, родитель, учитель, администратор).
- Реализованы функции: управление расписанием, выставление оценок, отслеживание посещаемости, публикация новостей, чат для коммуникации.
- Спроектирована архитектура с клиент-серверной моделью и реляционной базой данных.
- Проведено тестирование, подтвердившее работоспособность и соответствие требованиям.

Преимущества решения:

- Кроссплатформенность (поддержка Android, iOS, Windows, macOS).

- Централизованное управление данными для множества школ.
- Адаптивный интерфейс и высокая безопасность (шифрование данных, аутентификация).
- Сокращение времени на административные задачи для педагогов.
- Проект является завершённым, поставленные цели достигнуты.

Планируется дальнейшее развитие функциональности, включая интеграцию с государственными образовательными платформами и внедрение модулей аналитики на основе нейронных сетей.

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра _____ Программное обеспечение информационных технологий _____
(наименование кафедры)

УТВЕРЖДАЮ
Заведующий кафедрой
_____ Лапицкая Н. В.
(подпись) (фамилия, инициалы)
_____ 2025 г.

ЗАДАНИЕ
на дипломный проект (дипломную работу)

Обучающемуся _____ Дубовскому Алексею Владимировичу _____
(фамилия, собственное имя, отчество (если таковое имеется))

Курс _____ 4 _____ Учебная группа _____ 151004 _____ Специальность _____ 1-40 01 01 _____

1 Тема дипломного проекта (дипломной работы):

_____ Мобильное приложение «Электронный дневник» для образовательных учреждений с
использованием технологии .NET MAUI _____

Утверждена приказом ректора от _____ 26 _____ 02 _____ 2025 г. № _____ 520-с _____

2 Исходные данные к дипломному проекту (дипломной работе):

_____ Операционная система – Android 5. Язык программирования – C#.

_____ Перечень выполняемых функций: добавление аккаунтов школ, создание профилей с разными
ролями для каждой школы, введение учёта успеваемости по предметам, подсчёт среднего балла
для учеников.

_____ Назначение разработки: создание электронного дневника, поддерживающего все базовые
функции традиционных дневников в школах.

3 Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов):

_____ Введение

_____ 1 Анализ прототипов, литературных источников и формирование требований к проектируемому
программному средству

_____ 2 Моделирование предметной области

_____ 3 Проектирование программного средства

_____ 4 Создание программного средства

_____ 5 Тестирование, проверка работоспособности и анализ полученных результатов

_____ 6 Руководство по установке и использованию

_____ 7 Техничко-экономическое обоснование разработки и внедрения ПС

_____ Заключение

_____ Список использованных источников

_____ Приложение А (обязательное) Исходный код программного средства

5 Перечень графического материала (с точным указанием обязательных чертежей и графиков):

- Главный алгоритм программного средства. Схема алгоритма – формат А1, лист 1.
- Выбор элемента из списка. Схема алгоритма – формат А1, лист 1.
- Общение с сервером. Схема алгоритма – формат А1, лист 1.
- Use case диаграмма. Плакат – формат А1, лист 1.
- Физическая модель базы данных. Плакат – формат А1, лист 1.
- Deployment диаграмма. Плакат – формат А1, лист 1.
- Диаграмма классов наследования интерфейса «Список учеников». Плакат – формат А1, лист 1.

6 Консультанты по дипломному проекту (дипломной работе) (с указанием разделов, по которым они консультируют):

Консультант от кафедры – Мелких Е.Г.,
консультант по экономической части – Горюшкин А.А.,
нормоконтролёр – Грибович А.А.

7 Примерный календарный график выполнения дипломного проекта (дипломной работы):

- Анализ предметной области, разработка технического задания – 25.03–01.04
- Разработка функциональных требований, проектирование архитектуры ПС – 02.04–13.04
- Разработка схемы программы, алгоритмов, данных: 14.04–21.04
- Разработка программного средства: 22.04–05.05
- Тестирование и отладка: 06.05–13.05
- Оформление пояснительной записки и графического материала: 14.05–31.05

Дата выдачи задания 10 февраля 2025.

Срок сдачи законченного дипломного проекта (дипломной работы) 31 мая 2025 г.

Руководитель дипломного проекта
(дипломной работы)

(подпись)

(фамилия, инициалы)

Подпись обучающегося _____

Дата 10 февраля 2025 г.

СОДЕРЖАНИЕ

Определения и сокращения.....	7
Введение	8
1 Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству	10
1.1 Анализ прототипов.....	10
1.2 Сравнение прототипов.....	16
1.3 Формирование требований к проектируемому программному средству.....	20
2 Моделирование предметной области	24
2.1 Моделирование программного обеспечения	24
2.2 Инфологическая модель базы данных	27
2.3 Спецификация функциональных требований.....	32
3 Проектирования программного средства	34
3.1 Выбор архитектуры для разработки	34
3.2 Логическая модель баз данных.....	35
3.3 Физическая модель базы данных.....	42
3.4 Разработка алгоритмов программного средства и отдельных модулей	43
4 Создание программного средства.....	50
4.1 Выбор инструментов разработки.....	50
4.2 Разработка контроллеров.....	52
4.3 Разработка пользовательского интерфейса	53
5 Тестирование, проверка работоспособности и анализ результатов	59
6 Руководство по установке и использованию.....	63
6.1 Развёртывание серверной части	63
6.2 Установка клиентской части.....	64
6.3 Руководство по использованию.....	65
7 Техничко-экономическое обоснование разработки веб-приложения «электронный дневник» для учёта успеваемости в образовательных учреждениях с использованием технологии spring	69
7.1 Описание функций, назначения и потенциальных пользователей программного средства.....	69
7.2 Расчёт затрат на разработку программного средства	70
7.3 Оценка результата от использования программного сервиса.....	73
Заключение	75
Список использованных источников	76
Приложение А.....	77

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяются следующие определения и сокращения.

ПС – программное средство.

СУБД – система управления базами данных.

БД – база данных.

ВВЕДЕНИЕ

В последние годы наблюдается активное внедрение цифровых технологий в образовательный процесс. Традиционные бумажные журналы и дневники постепенно заменяются современными электронными системами, которые упрощают взаимодействие между участниками образовательного процесса — учащимися, родителями, педагогами и администрацией. Это позволяет повысить прозрачность, оперативность и эффективность обучения.

Актуальность темы заключается в необходимости создания централизованной и масштабируемой информационной системы, способной обслуживать множество образовательных учреждений с возможностью индивидуальной настройки каждой школы. На практике часто отсутствуют решения, которые позволяют реализовать такую гибкость в рамках одной платформы. Предлагаемая система предоставляет единое хранилище данных, при этом каждая школа может работать независимо: со своими классами, расписанием, преподавателями и учебными планами. Это особенно актуально для внедрения решения на уровне района, города или всей страны.

Разрабатываемый программный продукт представляет собой электронный дневник — информационную систему, предназначенную для хранения и предоставления учебных данных в удобной цифровой форме. Он обеспечивает доступ к расписанию, оценкам, домашним заданиям и другой информации, связанной с учебным процессом. Пользователи системы — учащиеся, родители и преподаватели — получают доступ к актуальным данным в любое время.

Целью данной работы является разработка мобильного приложения в составе системы электронного дневника, обеспечивающего быстрый и удобный доступ к информации для учащихся и их родителей.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) Проанализировать существующие решения и определить требования к функциональности электронного дневника.
- 2) Разработать архитектуру системы с поддержкой работы нескольких школ в рамках единой платформы.
- 3) Реализовать мобильное приложение с интуитивно понятным интерфейсом для отображения расписания, оценок, домашних заданий и уведомлений.
- 4) Обеспечить безопасность хранения и передачи данных.
- 5) Провести тестирование приложения и оценить его соответствие функциональным требованиям.

Таким образом, результатом работы станет современное мобильное приложение, интегрированное в масштабируемую систему электронного дневника, способное повысить качество взаимодействия участников образовательного процесса и быть внедрённым в любые учебные заведения страны.

1 АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ

1.1 Анализ прототипов

Анализ существующих аналогичных решений позволяет не только выявить тенденции на рынке образовательных приложений, но и формирует ясные требования для разработки мобильного приложения для электронного дневника, ориентированного на улучшение пользовательского опыта и эффективное управление учебным процессом.

1.1.1 School.by

School.by — это белорусская платформа, предназначенная для цифровизации образовательного процесса в школах и других учебных заведениях. Она обеспечивает эффективное управление успеваемостью учащихся, улучшает коммуникацию между учителями, учениками и родителями, а также способствует организации учебного процесса в электронном формате. [1][2]

Основные особенности:

- Электронный журнал и дневник: Учителя имеют возможность вводить и обновлять оценки, задания и другую учебную информацию, доступную для учеников и их родителей.

- Аналитика успеваемости: Платформа предоставляет графики и таблицы успеваемости для каждого класса и ученика, что позволяет отслеживать динамику обучения.

- Встроенный чат для коммуникации: Ученики и родители могут общаться с учителями и школьной администрацией через встроенный чат, что способствует оперативному решению возникающих вопросов. □

- SMS-уведомления о посещаемости: Родители получают сообщения о пропусках учеников, что помогает своевременно реагировать на проблемы с посещаемостью.

- Настраиваемый дизайн школьного портала: Возможность редактирования дизайна страниц школы, включая цветовое оформление и графическую структуру, позволяет адаптировать платформу под индивидуальные потребности учебного заведения.

Преимущества:

- Удобство и простота использования: Интуитивно понятный интерфейс облегчает работу для учителей, учеников и родителей.
- Повышенная вовлеченность участников: Оперативная обратная связь и доступ к информации способствуют активному участию всех участников образовательного процесса.
- Интеграция с онлайн-ресурсами: Платформа поддерживает объединение с различными сервисами для онлайн-школ, расширяя функциональные возможности.
- Мобильный доступ: Доступность платформы через мобильные устройства позволяет использовать ее в любое время и в любом месте.

Недостатки:

- Ограниченные аналитические функции: Платформа предоставляет базовые отчеты, но не поддерживает глубокий анализ успеваемости и динамики учащихся.
- Ограниченная настройка отчетности: Система отчетности может не удовлетворять специфические потребности некоторых учебных заведений.
- Ограниченная интеграция с внешними системами: Несмотря на возможность интеграции, некоторые внешние сервисы могут быть несовместимы или требовать дополнительной настройки.
- Ограниченная гибкость настройки интерфейса: Ограниченные возможности настройки интерфейса и персонализации могут не соответствовать ожиданиям всех пользователей.

School.by эффективная платформа для управления образовательным процессом, которая способствует улучшению коммуникации между учителями, учениками и родителями. Несмотря на некоторые ограничения в аналитике и настройках, она предоставляет широкий спектр инструментов для автоматизации учебного процесса и повышения его качества.

1.1.2 MyClassroom

MyClassroom — это образовательная платформа, разработанная для улучшения взаимодействия между учителями и учениками в условиях дистанционного, гибридного и смешанного обучения. Она предоставляет ряд уникальных инструментов, направленных на повышение эффективности учебного процесса.[3][4]

Основные особенности:

- Универсальность и доступность: MyClassroom работает на любом устройстве через веб-браузер, позволяя преподавать до 60 студентам одновременно.
- Интерактивное управление классом: Платформа предлагает инструменты для упрощения управления классом, включая учет посещаемости, совместную работу и удержание внимания учащихся.
- Групповая работа: Функция групп позволяет учащимся общаться в чате, взаимодействовать с доской своей группы и представлять свою работу классу, что способствует развитию сотрудничества и коммуникации.
- Интеграция с Google Classroom: MyClassroom легко интегрируется с Google Classroom, что обеспечивает эффективное управление занятиями и совместимость с другими образовательными сервисами.
- Интерактивные учебные материалы: Платформа предоставляет доступ к библиотеке готовых интерактивных материалов, планов уроков и обучающего контента, что облегчает подготовку и проведение занятий.
- Образовательный ассистент Companion: MyClassroom включает виртуального помощника Companion, который помогает преподавателям и учащимся в организации учебного процесса, предоставляя полезные рекомендации и поддержку.

Преимущества:

- Упрощение управления классом: Инструменты платформы облегчают организацию учебного процесса, включая управление посещаемостью и стимулирование участия учащихся.
- Стимулирование групповой работы: Функции групповой работы способствуют развитию сотрудничества и коммуникации среди учащихся.
- Интеграция с популярными сервисами: Совместимость с Google Classroom и другими сервисами обеспечивает гибкость и расширенные возможности для преподавателей и учащихся.
- Доступ к интерактивным ресурсам: Библиотека учебных материалов и планов уроков помогает преподавателям эффективно готовить и проводить занятия.
- Поддержка виртуального помощника: Виртуальный ассистент Companion предоставляет рекомендации и поддержку, улучшая организацию учебного процесса.

Недостатки:

- Ограничения по количеству студентов: Платформа поддерживает до 60 студентов одновременно, что может быть недостаточно для крупных учебных заведений.
- Необходимость в обучении персонала: Для эффективного использования всех функций платформы может потребоваться

дополнительное обучение преподавателей и технического персонала.

MyClassroom — это мощная и гибкая платформа, предназначенная для улучшения взаимодействия между учителями и учениками в различных форматах обучения. Благодаря широкому спектру инструментов и интеграций, она способствует повышению эффективности учебного процесса. Однако перед внедрением рекомендуется учитывать возможные ограничения и подготовить персонал к работе с новой системой.

1.1.3 Edmodo

Edmodo — это образовательная социальная сеть, предназначенная для взаимодействия между учителями, учениками и их родителями. Платформа предоставляет инструменты для обмена учебным контентом, общения в режиме реального времени и управления учебным процессом.[5][6]

Основные особенности:

- Обмен учебным контентом: Учителя могут публиковать задания, тесты и учебные материалы, доступные для учеников и их родителей.
- Коммуникация в реальном времени: Платформа поддерживает обмен сообщениями, что облегчает оперативное решение учебных вопросов.
- Управление классами: Edmodo позволяет организовывать учебный процесс, отслеживать успеваемость и взаимодействовать с учениками и их родителями.

Преимущества:

- Безопасность: Платформа обеспечивает защищенную среду для общения и обмена информацией между участниками образовательного процесса.
- Удобство использования: Интуитивно понятный интерфейс облегчает доступ к учебным материалам и коммуникацию между пользователями.
- Интеграция с другими сервисами: Edmodo поддерживает совместную работу с различными образовательными инструментами, расширяя возможности обучения.

Недостатки:

- Ограничения функционала: Некоторые пользователи отмечают недостаток расширенных функций для анализа успеваемости и настройки интерфейса.

Edmodo продолжает оставаться популярным инструментом в сфере образования, предлагая широкий спектр функций для эффективного управления учебным процессом. Несмотря на некоторые ограничения, платформа предоставляет ценную поддержку для учителей, учеников и родителей, способствуя улучшению образовательного опыта.

1.1.4 Google Classroom

Google Classroom — это бесплатная образовательная платформа от Google, предназначенная для упрощения процесса создания, распространения и оценки заданий без использования бумаги. [7][8]

Основные особенности:

- Интеграция с продуктами Google: Classroom объединяет приложения, такие как Google Диск, Документы, Таблицы, Презентации, Формы и Почта, обеспечивая удобство работы в одном пространстве.
- Управление заданиями: Учителя могут создавать задания с различными параметрами доступа, устанавливать сроки выполнения и предоставлять обратную связь, что способствует эффективному обучению.
- Коммуникация: Платформа поддерживает обмен сообщениями между учителями и учениками, а также возможность публиковать объявления и вести обсуждения в классе.
- Мобильный доступ: Доступность мобильных приложений для iOS и Android позволяет использовать Classroom на различных устройствах, обеспечивая гибкость в обучении.

Преимущества:

- Удобство использования: Интуитивно понятный интерфейс и интеграция с другими сервисами Google делают платформу доступной для пользователей с разным уровнем подготовки.
- Организация учебного процесса: Classroom помогает систематизировать задания, отслеживать успеваемость и упрощает взаимодействие между всеми участниками образовательного процесса.

Недостатки:

- Ограниченная поддержка стандартов электронного обучения: Отсутствие поддержки SCORM, Tin Can (xAPI) и cm5 может затруднить использование некоторых интерактивных курсов.
- Отсутствие встроенной вебинарной комнаты: Для проведения онлайн-занятий требуется использование дополнительных сервисов, таких как YouTube или Google Hangouts.
- Ограничения для бесплатных пользователей: В бесплатной версии

сервиса количество участников курса ограничено 200 человек, что может быть недостаточно для крупных образовательных учреждений.

Google Classroom является мощным инструментом для организации учебного процесса, предлагая интеграцию с продуктами Google, эффективное управление заданиями и удобные коммуникационные возможности. Однако перед его использованием следует учитывать ограничения, связанные с поддержкой стандартов и функциональными возможностями, чтобы определить его соответствие специфическим потребностям образовательного учреждения.

1.1.5 ClassDojo

ClassDojo — это бесплатная образовательная платформа, предназначенная для содействия сотрудничеству между учителями, учениками и их семьями. Сервис поддерживает социально-эмоциональное обучение, предоставляя инструменты для обратной связи, создания портфолио и коммуникации.[9][10]

Основные особенности:

- Обратная связь через баллы: Учителя могут присуждать ученикам баллы за различные достижения и поведение, что способствует развитию положительных навыков и привычек.

- Портфолио учеников: ClassDojo предоставляет возможность ученикам создавать персональные портфолио, где они могут демонстрировать свои работы и достижения, а также получать обратную связь от учителей и родителей.

- Коммуникация с родителями: Платформа упрощает общение между учителями и родителями, позволяя отправлять уведомления о достижениях и проблемах учеников, а также получать обратную связь.

- Настройка ценностей сообщества: ClassDojo позволяет учителям устанавливать и отслеживать определённые ценности и ожидания в классе, способствуя формированию сплочённой и позитивной учебной среды.

Преимущества:

- Удобство использования: Сервис обладает интуитивно понятным интерфейсом и доступен на нескольких языках, включая русский, что облегчает его использование учителями, учениками и родителями.

- Гибкость и адаптивность: ClassDojo можно настроить в соответствии с потребностями конкретного класса или школы, позволяя учителям индивидуализировать подход к обучению и воспитанию.

- Доступность на мобильных устройствах: Платформа имеет мобильное

приложение, что позволяет учителям и родителям оставаться на связи и отслеживать прогресс учеников в любое время и в любом месте.

Недостатки:

– Зависимость от интернет-соединения: Для использования ClassDojo требуется стабильное интернет-соединение, что может быть проблемой в районах с ограниченным доступом к интернету.

– Ограничения функциональности: Некоторые пользователи отмечают, что функциональность мобильного приложения ClassDojo может быть ограничена по сравнению с веб-версией, что может затруднить использование платформы на мобильных устройствах.

ClassDojo является эффективным инструментом для поддержки социально-эмоционального обучения и улучшения коммуникации между учителями, учениками и их семьями. Несмотря на некоторые ограничения, такие как зависимость от интернет-соединения и функциональные ограничения мобильного приложения, ClassDojo предоставляет множество возможностей для создания позитивной и продуктивной учебной среды

1.2 Сравнение прототипов

Таблица 1 – Сравнительная характеристика прототипов

Характеристика	School.by	MyClassroom	Edmodo	Google Classroom	ClassDojo
Целевая аудитория	Школы и колледжи Беларус	Учителя и ученики по всему миру	Учителя, ученики и родители	Учебные заведения всех уровней	Учителя, ученики и их семьи
Тип платформы	Веб-сервис и мобильные приложения	Веб-сервис и мобильные приложения	Веб-сервис и мобильные приложения	Веб-сервис и мобильные приложения	Веб-сервис и мобильные приложения

Продолжение таблицы 1 – Сравнительная характеристика прототипов

Основной функционал	Электронный дневник, расписание, домашние задания, оценки, уведомления	Виртуальные классы, расписание, тесты, обмен сообщениями	Электронный дневник, задания, тесты, форумы	Расписание, задания, тесты, аналитика	Баллы, портфолио, связь с родителями
Интеграция с другими сервисами	Ограниченная интеграция с внешними сервисами	Интеграция с Google, Microsoft, Zoom	Интеграция с Google, Microsoft, внешними сервисами	Полная интеграция с Google (Docs, Drive и т.д.)	Интеграция с Google и другими сервисами
Поддержка мобильных платформ	iOS, Android	iOS, Android	iOS, Android	iOS, Android	iOS, Android
Возможность офлайн-доступа	Нет	Частично (ограниченный функционал)	Нет	Частично (ограниченный функционал)	Частично (ограниченный функционал)
Поддержка мультиязычности	Русский, белорусский	Многоязычная поддержка	Многоязычная поддержка	Многоязычная поддержка	Многоязычная поддержка
Аналитика и отчётность	Базовая аналитика, ограниченные отчёты	Расширенная аналитика, визуализация данных	Ограниченная аналитика, базовые отчёты	Расширенная аналитика, интеграция с Google Sheets	Базовая аналитика, отчёты по поведению

Продолжение таблицы 1 – Сравнительная характеристика прототипов

Безопасность данных	Соответствие национальным стандартам	Шифрование данных, соответствие международным стандартам	Шифрование данных, соответствие международным стандартам	Шифрование данных, соответствие международным стандартам	Шифрование данных, соответствие международным стандартам
Поддержка пользователей	Электронная почта, телефон	Онлайн-чат, база знаний	Форумы, база знаний	Форумы, база знаний	Онлайн-чат, база знаний
Уровень персонализации интерфейса	Ограниченная настройка	Расширенная настройка	Ограниченная настройка	Ограниченная настройка	Расширенная настройка
Поддержка стандартов электронного обучения (SCORM, xAPI)	Нет	Частично	Нет	Нет	Нет
Наличие встроенного видеочата	Нет	Да	Нет	Нет	Нет
Поддержка социально-эмоционального обучения	Нет	Нет	Нет	Нет	Да

Продолжение таблицы 1 – Сравнительная характеристика прототипов

Наличие виртуального ассистента	Нет	Да (Companion)	Нет	Нет	Нет
Поддержка геймификации	Нет	Частично	Нет	Нет	Да
Наличие API для разработчиков	Нет	Да	Нет	Да	Нет
Регулярность обновлений	Нерегулярные	Регулярные	Регулярные	Регулярные	Регулярные
Поддержка обратной связи от пользователей	Да	Да	Да	Да	Да
Уровень адаптации под различные образовательные системы	Высокий для Беларуси	Средний	Средний	Высокий	Средний
Возможность настройки уведомлений	Да	Да	Да	Да	Да
Возможность экспорта данных	Ограниченная	Да	Ограниченная		Ограниченная

Продолжение таблицы 1 – Сравнительная характеристика прототипов

Поддержка форумов и обсуждений	Нет	Да	Да	Да	Нет
Возможность настройки прав доступа	Ограниченная	Да	Ограниченная	Да	Ограниченная
Поддержка уведомлений в реальном времени	Да	Да	Да	Да	Да
Возможность настройки интерфейса под бренд школы	Ограниченная	Да	Ограниченная	Ограниченная	Да
Поддержка интеграции с социальными сетями	Нет	Да	Да	Да	Да

1.3 Формирование требований к проектируемому программному средству

1.3.1 Назначение разработки

Проектируемое программное средство — это система «Электронный дневник», предназначенная для автоматизации учебного процесса в образовательных учреждениях. Основной задачей системы является упрощение взаимодействия между учениками, родителями, учителями и администрацией школы. Система предоставляет доступ к информации о

расписании, успеваемости, домашних заданиях, а также позволяет управлять учебными планами и школьными данными.

Платформа ориентирована на использование в масштабе всей страны и позволяет добавлять в систему множество школ с индивидуальной настройкой параметров каждой. Это делает систему универсальной, гибкой и пригодной для централизованного управления образовательным процессом на уровне регионов и государства. Она будет способствовать улучшению контроля за успеваемостью учащихся, повышению прозрачности учебного процесса и упрощению коммуникации между всеми участниками.

1.3.2 Состав выполняемых функций

Состав выполняемых функций включает:

- Создание журналов для учителей с возможностью внесения оценок, отметок о пропусках и другой важной информации по каждому учебному предмету.
- Формирование электронных дневников для учеников и родителей с доступом к расписанию, учебным предметам, оценкам и домашним заданиям.
- Возможность добавления школ с уникальными объектами конкретной школы, такими как расписание и преподаватели.
- Назначение администратора для каждой школы с правами редактирования и управления данными школы.
- Автоматический расчет среднего балла ученика за учебную четверть.
- Реализация профилей пользователей с различным доступом в зависимости от их роли (ученик, родитель, учитель, администратор).
- Привязка родителей к ученикам для доступа к электронному дневнику.
- Создание групп и подгрупп для раздельного проведения уроков.
- Отображение общей информации о школах, в том числе новостей и объявлений.
- Возможность публикации школьных новостей, мероприятий и объявлений.
- Реализация функции комментирования новостей для взаимодействия между пользователями.
- Организация личных сообщений между участниками системы для обмена информацией.
- Добавление школ в систему.

1.3.3 Входные данные

Система будет принимать следующие данные:

- Информация о пользователях: имя, роль, школа, классы, преподаваемые предметы (для учителей), контактные данные, аватар профиля.
- Данные о расписании уроков: дата, время, аудитория, преподаватель.
- Оценки учеников: баллы, дата выставления, ID преподавателя.
- Отметки о пропусках: дата.

- Домашние задания: текст задания, ID класса.
- Новости и объявления: заголовок, текст, дата публикации, ID автора.
- Личные сообщения: текст, ID отправителя и получателя, метка времени.
- Комментарии: текст, ID автора, ID новости/профиля, дата публикации.
- Системные данные: логи аудита (изменения оценок, правки расписания), пароли, логины.
- Школы: название, контактная информация, адрес, объекты школы.

1.3.4 Выходные данные

Система будет генерировать:

- Электронные дневники: оценки, домашние задания, уведомления о событиях (для учеников и родителей).
- Журналы учителей: таблицы с оценками, отметки о пропусках, возможность редактирования данных.
- Отчеты об успеваемости: средний балл ученика, динамика оценок, статистика посещаемости.
- Личные сообщения: история переписок, уведомления о новых сообщениях.
- Комментарии: текстовые сообщения под новостями и профилями пользователей.
- Новости и объявления: заголовок, текст, дата публикации, автор.
- Сводная информация: актуальное расписание, изменения в учебном процессе, активность пользователей.

1.3.5 Требования к составу и параметрам технических и программных средств

Для корректной работы системы потребуется:

- Мобильные приложения для платформ Android и iOS с адаптивным интерфейсом, а также возможность запуска на Windows и macOS.
- Базы данных для хранения информации о пользователях, школьных данных, расписаниях и оценках.
- Операционные системы: Linux/Windows для серверной части, Android/iOS для мобильных устройств, Windows/macOS для возможного запуска на ПК.
- Подключение к интернету для обеспечения синхронизации данных между различными устройствами и пользователями.

1.3.6 Требования к информационной и программной совместимости

Система должна быть совместима с современными операционными системами и веб-браузерами, включая:

- Операционные системы: Android, iOS, Windows, macOS.
- Обеспечение безопасности данных с использованием хеширования паролей с солью, разграничения доступа по ролям и защиты передаваемой информации.

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1 Моделирование программного обеспечения

2.1.1 Табличное представление

Таблица 2 – Описание ролей

Роль	Описание	Уровень доступа
Оператор министерства	Обеспечивает техническую поддержку и настройку системы на уровне министерства	Полный доступ к системным настройкам
Администрация министерства	Управляет образовательными учреждениями, анализирует статистику	Доступ к агрегированным данным
Оператор школы	Настраивает и поддерживает работу системы в рамках конкретной школы	Доступ к настройкам школы
Администрация школы	Контролирует учебный процесс, формирует отчеты	Доступ к данным школы
Ученик	Просматривает расписание, оценки, домашние задания, взаимодействует с учителями	Доступ к личному кабинету
Родитель	Отслеживает успеваемость ребенка, получает уведомления, общается с учителями	Доступ к личному кабинету и данным ребенка
Учитель	Выставляет оценки, публикует задания, ведет электронный журнал, общается с учениками и родителями	Доступ к личному профилю, своему классу/предмету

Продолжение таблицы 2 – Описание ролей

Гость	Имеет доступ исключительно к авторизации	Доступ к странице авторизации
-------	--	----------------------------------

2.1.2 Схематичное представление

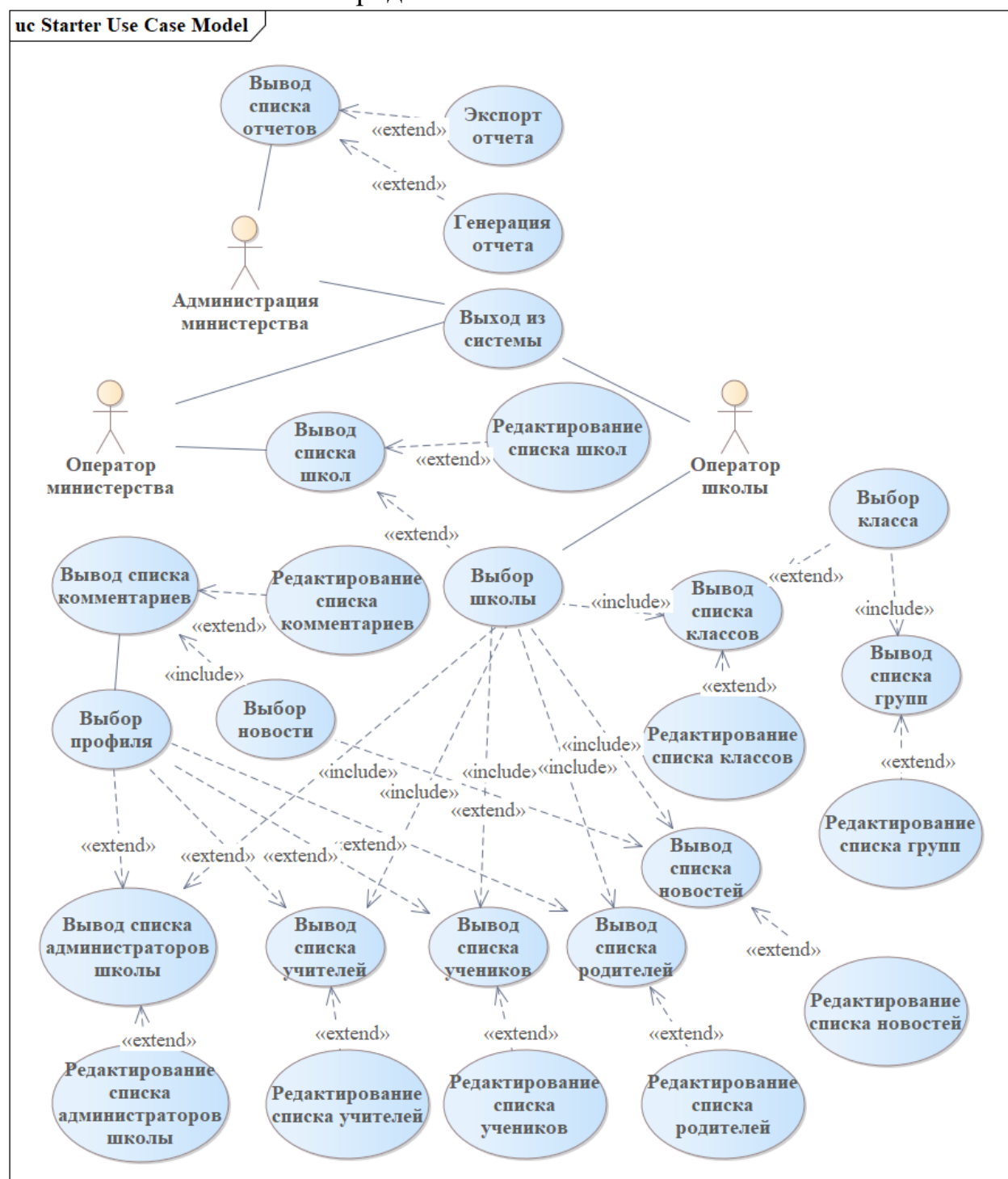


Рисунок 1 — UML диаграмма административного уровня (без коммуникации)

[illegible]

26

На рисунке 2 представлена структура взаимодействия участников образовательного процесса. Здесь реализованы функции, связанные с повседневной деятельностью учебного заведения: публикация комментариев, обмен сообщениями, управление расписанием, ведение электронных журналов и дневников. В отличие от административного уровня, система здесь выступает платформой для активного взаимодействия между всеми участниками, обеспечивая выполнение образовательных задач и коммуникацию.

2.2 Инфологическая модель базы данных

2.2.1 Табличное представление

Таблица 3 – Сущности и связи

Сущность	Атрибуты	Связанные сущности
Базовые данные пользователей (ELD_USERS)	Логин, Хеш, Соль	
Типы учебных заведений (ELD_EDUCATIONAL_INSTITUTIONS_TYPES)	Название	
Регионы (ELD_REGIONS)	Название	
Населенные пункты (ELD_SETTLEMENTS)	Название	Регионы (ELD_REGIONS)
Учебные заведения (ELD_EDUCATIONAL_INSTITUTIONS)	Адрес, email, название, путь к изображению, телефон	Типы учебных заведений (ELD_EDUCATIONAL_INSTITUTIONS_TYPES), Населенные пункты (ELD_SETTLEMENTS)
Администраторы (ELD_ADMINISTRATORS)	email, имя, фамилия, путь к изображению, отчество, телефон	Учебные заведения (ELD_EDUCATIONAL_INSTITUTIONS), Базовые данные пользователей (ELD_USERS)

Продолжение таблицы 3 – Сущности и связи

Учителя (ELD_TEACHERS)	email, имя, фамилия, путь к изображению, отчество, телефон	Учебные заведения (ELD_EDUCATIONAL_IN STITUTIONS), Базовые данные пользователей (ELD_USERS)
Классы (ELD_CLASSES)	Название	Учителя (ELD_TEACHERS)
Школьные предметы (ELD_SCHOOL_SUB JECTS)	Название	
Назначения учителей (ELD_TEACHER_AS SIGNMENTS)		Школьные предметы (ELD_SCHOOL_SUBJECT S), Учителя (ELD_TEACHERS), Группы (ELD_GROUPS)
Группы (ELD_GROUPS)	Название	Классы (ELD_CLASSES)
Дни журнала (ELD_GRADEBOOK _DAYS)	Дата, тема урока, домашнее задание	Уроки в расписании (ELD_SCHEDULE_LESSON S)
Ученики (ELD_SCHOOL_STU DENTS)	email, имя, фамилия, путь к изображению, отчество, телефон	Классы (ELD_CLASSES), Учебные заведения (ELD_EDUCATIONAL_IN STITUTIONS), Базовые данные пользователей (ELD_USERS)
Посещаемость (ELD_GRADEBOOK _ATTENDANCES)	Наличие записи при отсутствии	Дни журнала (ELD_GRADEBOOK_DAY S), Ученики (ELD_SCHOOL_STUDENT S)
Оценки (ELD_GRADEBOOK _SCORES)	Оценка	Дни журнала (ELD_GRADEBOOK_DAY S), Ученики (ELD_SCHOOL_STUDENT S)

Продолжение таблицы 3 – Сущности и связи

Участники групп (ELD_GROUP_MEMBERS)		Группы (ELD_GROUPS), Ученики (ELD_SCHOOL_STUDENTS)
Изображения (ELD_IMAGES)	Путь изображения школы	Учебные заведения (ELD_EDUCATIONAL_INSTITUTIONS)
Сообщения (ELD_MESSAGES)	Текст, время отправки	Пользователь адресат (ELD_USERS), Пользователь отправитель (ELD_USERS)
Новости (ELD_NEWS)	Заголовок, содержание, дата публикации	Пользователь владелец (ELD_USERS), Учебные заведения (ELD_EDUCATIONAL_INSTITUTIONS)
Комментарии к новостям (ELD_NEW_COMMENTS)	Текст, время	Новости (ELD_NEWS), Пользователи (ELD_USERS)
Родители (ELD_PARENTS)	email, имя, фамилия, путь к изображению, отчество, телефон	Учебные заведения (ELD_EDUCATIONAL_INSTITUTIONS), Базовые данные пользователей (ELD_USERS)
Типы родственных связей (ELD_PARENTS_TYPES)	Название (отец, мать и т.д.)	
Связь детей и родителей (ELD_SCHOOL_STUDENTS_AND_PARENTS)		Типы родственных связей (ELD_PARENTS_TYPES), Родители (ELD_PARENTS), Ученики (ELD_SCHOOL_STUDENTS)
Информация о четверти (ELD_QUARTER_INFO)	Дата начала, дата конца, номер четверти	

Продолжение таблицы 3 – Сущности и связи

Оценки за четверть (ELD_QUARTER_SCORES)	Номер четверти, оценка	Предметы (ELD_SCHOOL_SUBJECTS), Ученики (ELD_SCHOOL_STUDENTS), Информация о четверти (ELD_QUARTER_INFO)
Связи учеников и родителей (ELD_SCHOOL_STUDENTS_AND_PARENTS)		Ученики (ELD_SCHOOL_STUDENTS), Родители (ELD_PARENTS), Типы родственных связей (ELD_PARENTS_TYPES)
Уроки в расписании (ELD_SCHEDULE_LESSONS)	День недели, номер урока, предмет урока, кабинет	Группы (ELD_GROUPS), Информация о четверти (ELD_QUARTER_INFO), Назначения учителей (ELD_TEACHER_ASSIGNMENTS)
Комментарии пользователей (ELD_USER_COMMENTS)	Текст, время	Пользователь отправитель (ELD_USERS), Пользователь-получатель (ELD_USERS)
Типы пользователей (ELD_USERS_TYPES)	Название (админ, учитель и т.д.)	
Уведомления пользователей (ELD_NOTIFICATION)	Название, описание, время, ссылка	Пользователи (ELD_USERS),

2.2.2 Схематичное представление

Концептуальная инфологическая модель базы данных отражает основные сущности системы и связи между ними. В модели выделены ключевые объекты, такие как пользователи, учебные заведения, журналы успеваемости и учебные дисциплины. Каждая сущность обладает уникальными атрибутами, определяющими ее свойства и взаимоотношения в системе.

Инфологическая модель базы данных представлена на рисунке 3.

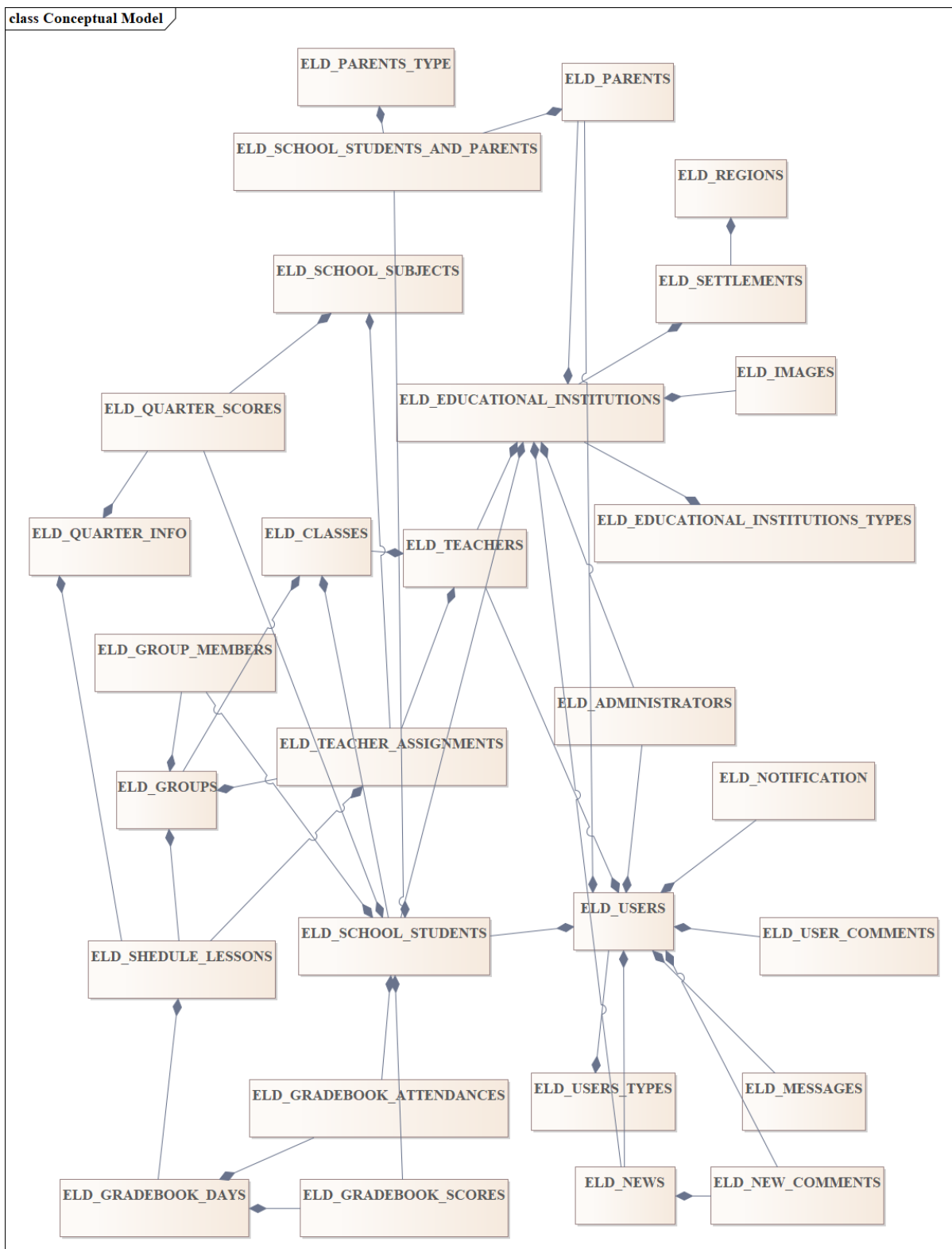


Рисунок 3 — Инфологическая модель базы данных

2.3 Спецификация функциональных требований

2.3.1 Авторизация и ролевой доступ

Спецификация:

- Пользователь входит в систему по логину и паролю.
- Пароль хранится в зашифрованном виде (алгоритм SHA-256 с солью).
- Администратор может сбросить пароль пользователя через панель управления.
- Ролевая модель доступа.

Спецификация ролевой модели доступа:

- Общий функционал: доступ к сообщениям\новостям\комментариям.
- Ученик: просмотр оценок, расписания, домашних заданий.
- Родитель: доступ к данным ребенка.
- Учитель: внесение оценок, управление домашними заданиями.
- Локальный администратор: доступ к определенной школе, управление пользователями, аудит действий.
- Глобальный админ: расширение локального админа с доступом ко всем школам.

2.3.2 Просмотр электронного журнала учителем

Спецификация:

- Учитель должен иметь доступ к журналу с возможностью просмотра всех учеников своего класса.
- Журнал содержит столбцы с датами, выставленными оценками, темами урока, домашними заданиями.
- Ученик и родитель могут просматривать только оценки, выставленные ученику.
- Журнал автоматически рассчитывает средний балл ученика за четверть.
- Данные обновляются в реальном времени при наличии интернет-соединения.

2.3.3 Просмотр электронного журнала

Спецификация:

- Дневник содержит расписание учебных занятий по дням недели.
- В информацию об уроке входят: номер, название предмета, ФИО учителя, кабинет.

2.3.4 Управление пользователями

Спецификация:

- Администратор добавляет пользователей через панель администратора, задавая информацию о пользователе, логин и пароль.

- Администратор изменяет\удаляет пользователей.
- Родительский аккаунт не может существовать без аккаунта ученика.
- При удалении удаляются все связанные сущности.

2.3.5 Управление школами

Спецификация:

- Главный администратор осуществляет добавление\изменение\удаление.
- При удалении удаляются все связанные сущности.
- Администраторы добавляют пользователей с привязкой к школе.

2.3.6 Управление новостями\сообщениями\комментариями

Спецификация:

- Администраторы\учителя создают\редактируют\удаляют новости.
- Любой авторизованный пользователь создаёт комментарий\сообщение.
- Администратор редактирует удаляет сообщения\комментарии.

3 ПРОЕКТИРОВАНИЯ ПРОГРАММНОГО СРЕДСТВА

3.1 Выбор архитектуры для разработки

Архитектура системы «Электронный дневник» основана на многоуровневой клиент-серверной модели, обеспечивающей разделение ответственности между компонентами и поддержку кроссплатформенности.

Клиентский уровень:

- Мобильные приложения: интерфейсы для учеников, родителей, учителей и администраторов.
- Функционал: отображение расписания, оценок, домашних заданий; взаимодействие через чаты и уведомления.

Серверный уровень:

- Использование серверной части стороннего разработчика

Уровень данных:

- База данных: реляционная СУБД для хранения структурированной информации (пользователи, школы, оценки, расписания).
- Медиа сервер: получение\хранение\удаление файлов.
- Резервное копирование: обеспечение сохранности данных.

Выбор клиент-серверной архитектуры обоснован ключевыми требованиями проекта:

- Кроссплатформенность — необходимость поддержки мобильных устройств (Android/iOS).
- Централизованное управление данными — единый источник истины для учебных заведений, учеников и родителей.
- Безопасность и масштабируемость — защита персональных данных и возможность роста числа пользователей без деградации производительности.
- Разделение ответственности. Клиент отвечает за взаимодействие с пользователем и конвертацию его действий в единый формат. Сервер отвечает за централизованную обработку данных единого формата независимо от платформы пользователя.

Выбор реляционной СУБД обоснован следующими факторами:

- Целостность данных: Внешние ключи, транзакции и нормализация предотвращают дублирование и противоречия в данных.
- Безопасность: Встроенные механизмы шифрования и разграничения прав доступа соответствуют требованиям GDPR.

3.2 Логическая модель баз данных

3.2.1 Табличное представление

Таблица 4 – Сущности и связи

Сущности	Атрибуты	Ключи
ELD_USERS	HASH(BYTE[]), ID(INTEGER), LOGIN(String), SALT(BYTE[]), U_UT_ID(INTEGER),	ID первичный ключ, U_UT_ID внешний ключ к таблице ELD_USER_TYPES
ELD_REGIONS	ID(INTEGER), NAME(String)	ID первичный ключ
ELD_SETTLEMENTS	ID(INTEGER), NAME(String), S_R_ID(INTEGER)	ID первичный ключ, S_R_ID внешний ключ к таблице ELD_REGIONS
ELD_EDUCATIONAL_INSTITUTIONS_TYPES	ID(INTEGER), NAME(String)	ID первичный ключ
ELD_EDUCATIONAL_INSTITUTIONS	ID(INTEGER), ADDRESS(String), EMAIL(String), NAME(String), PATH_IMAGE(String) , PHONE_NUMBER(String), EI_EIT_ID(INTEGER), EI_S_ID(INTEGER)	ID первичный ключ, EI_EIT_ID внешний ключ к таблице ELD_EDUCATIONAL_INSTITUTIONS_TYPES, EI_S_ID внешний ключ к таблице ELD_SETTLEMENTS
ELD_CLASSES	ID(INTEGER), NAME(String), C_T_ID(INTEGER)	ID первичный ключ, C_T_ID внешний ключ к таблице ELD_TEACHERS
ELD_SCHOOL_SUBJECTS	ID(INTEGER), NAME(String)	ID первичный ключ

Продолжение таблицы 4 – Сущности и связи

ELD_ADMINISTRATORS	ID(INTEGER), EMAIL(String), FIRST_NAME(String), LAST_NAME(String), PATH_IMAGE(String), PATRONYMIC(String), PHONE_NUMBER(String), A_EI_ID(INTEGER), A_U_ID(INTEGER)	ID первичный ключ, T_EI_ID внешний ключ к таблице ELD_EDUCATIONAL_INSTITUTIONS, T_U_ID внешний ключ к таблице ELD_USERS
ELD_TEACHERS	ID(INTEGER), EMAIL(String), FIRST_NAME(String), LAST_NAME(String), PATH_IMAGE(String), PATRONYMIC(String), PHONE_NUMBER(String), T_EI_ID(INTEGER), T_U_ID(INTEGER)	ID первичный ключ, T_EI_ID внешний ключ к таблице ELD_EDUCATIONAL_INSTITUTIONS, T_U_ID внешний ключ к таблице ELD_USERS
ELD_TEACHER_ASSIGNMENTS	ID(INTEGER), TA_G_ID(INTEGER), TA_SS_ID(INTEGER), TA_T_ID(INTEGER)	ID первичный ключ, TA_G_ID внешний ключ к таблице ELD_GROUPS, TA_SS_ID внешний ключ к таблице ELD_SCHOOL_SUBJECTS, TA_T_ID внешний ключ к таблице ELD_TEACHERS
ELD_GROUPS	ID(INTEGER), GROUP_NAME(String) G_TA_ID(INTEGER)	ID первичный ключ, G_TA_ID внешний ключ к таблице ELD_TEACHER_ASSIGNMENTS

Продолжение таблицы 4 – Сущности и связи

ELD_GRADEBOOK_DAYS	ID(INTEGER), DATE_TIME(DATETIME), HOMEWORK(STRING), TOPIC(STRING) GD_SL_ID(INTEGER),	ID первичный ключ, GD_SL_ID внешний ключ к таблице ELD_SCHEDULE_LESS ONS
ELD_SCHOOL_STUDENTS	ID(INTEGER), EMAIL(STRING), FIRST_NAME(STRING), LAST_NAME(STRING), PATH_IMAGE(BYTE[]), PATRONYMIC(STRING), PHONE_NUMBER(STRING), SST_C_ID(INTEGER), SST_EI_ID(INTEGER), SST_U_ID(INTEGER)	ID первичный ключ, SST_C_ID внешний ключ к таблице ELD_CLASSES, SST_EI_ID внешний ключ к таблице ELD_EDUCATIONAL_ INSTITUTIONS, SST_U_ID внешний ключ к таблице ELD_USERS
ELD_GRADEBOOK_ATTENDANCES	ID(INTEGER), GA_GD_ID(INTEGER), GA_SST_ID(INTEGER),	ID первичный ключ, GA_GD_ID внешний ключ к таблице ELD_GRADEBOOK_DAYS, GA_SST_ID внешний ключ к таблице, ELD_SCHOOL_STUDENTS
ELD_GRADEBOOK_SCORES	ID(INTEGER), SCORE(INTEGER) GS_GD_ID(INTEGER), GS_SST_ID(INTEGER),	ID первичный ключ, GS_GD_ID внешний ключ к таблице ELD_GRADEBOOK_DAYS, GS_SST_ID внешний ключ к таблице ELD_SCHOOL_STUDENTS

Продолжение таблицы 4 – Сущности и связи

ELD_GROUP_MEMBERS	ID(INTEGER), GM_G_ID(INTEGER), GM_SST_ID(INTEGER)	ID первичный ключ, GM_G_ID внешний ключ к таблице ELD_GROUPS, GM_SST_ID внешний ключ к таблице ELD_SCHOOL_STUDENTS
ELD_IMAGES	ID(INTEGER), PATH_IMAGE(String) , I_EI_ID(INTEGER)	ID первичный ключ, I_EI_ID внешний ключ к таблице ELD_EDUCATIONAL_INSTITUTIONS
ELD_MESSAGES	ID(INTEGER), MESSAGE(String), DATE(DATETIME) M_U_GETTER_ID(INTEGER), M_U_SENDER_ID(INTEGER),	ID первичный ключ, M_U_GETTER_ID внешний ключ к таблице ELD_USERS M_U_SENDER_ID, внешний ключ к таблице ELD_USERS
ELD_NEWS	ID(INTEGER), TITLE(String), CONTENT(String), DATE(DATE) N_U_OWNER_ID(INTEGER),	ID первичный ключ, N_U_OWNER_ID внешний ключ к таблице ELD_USERS
ELD_NEW_COMMENTS	ID(INTEGER), TEXT(String), DATE(DATETIME) NC_N_GETTER_ID(INTEGER), NC_U_GETTER_ID(INTEGER),	ID первичный ключ, NC_N_GETTER_ID внешний ключ к таблице ELD_NEWS, NC_U_SENDER_ID внешний ключ к таблице ELD_USERS
ELD_PARENTS_TYPES	ID(INTEGER), NAME(String)	ID первичный ключ

Продолжение таблицы 4 – Сущности и связи

ELD_PARENTS	ID(INTEGER), EMAIL(String), FIRST_NAME(String), LAST_NAME(String), PATH_IMAGE(String) , PATRONYMIC(String) , PHONE_NUMBER(String), P_EI_ID(INTEGER), P_U_ID(INTEGER)	ID первичный ключ, P_EI_ID внешний ключ к таблице ELD_EDUCATIONAL_ INSTITUTIONS, P_U_ID внешний ключ к таблице ELD_USERS
ELD_QUARTER_SCORES	ID(INTEGER), SCORE(INTEGER), QS_SS_ID(INTEGER), QS_SST_ID(INTEGER), QS_QI_ID(INTEGER),	ID первичный ключ, QS_SS_ID внешний ключ к таблице ELD_SCHOOL_SUBJECTS, QS_SST_ID внешний ключ к таблице ELD_SCHOOL_STUDENTS, QS_QI_ID внешний ключ к таблице ELD_QUARTER_INFO,
ELD_QUARTER_INFO	ID(INTEGER), QUARTER_NUMBER(INTEGER), DATA_END_TIME(DATETIME), DATA_START_TIME(DATETIME)	ID первичный ключ

Продолжение таблицы 4 – Сущности и связи

ELD_SCHOOL_STUDENTS_AND_PARENTS	ID(INTEGER), SSTAP_P_ID(INTEGER), SSTAP_PT_ID(INTEGER), SSTAP_SST_ID(INTEGER)	ID первичный ключ, SSTAP_P_ID внешний ключ к таблице ELD_PARENTS, SSTAP_PT_ID внешний ключ к таблице ELD_PARENT_TYPES, SSTAP_SST_ID внешний ключ к таблице ELD_SCHOOL_STUDENTS
ELD_SCHEDULE_LESSONS	ID(INTEGER), DAY_NUMBER(INTEGER), LESSON_NUMBER(INTEGER) SL_G_ID(INTEGER), SL_QI_ID(INTEGER), SL_TA_ID(INTEGER),	ID первичный ключ, SL_QI_ID внешний ключ к таблице ELD_QUARTER_INFO, SL_TA_ID внешний ключ к таблице ELD_TEACHER_ASSIGNMENTS
ELD_USER_COMMENTS	ID(INTEGER), TEXT(STRING), DATE(DATETIME) UC_U_GETTER_ID(INTEGER), UC_U_SENDER_ID(INTEGER),	ID первичный ключ, UC_U_GETTER_ID внешний ключ к таблице ELD_USERS, UC_U_SENDER_ID внешний ключ к таблице ELD_USERS
ELD_USERS_TYPES	ID(INTEGER), NAME(STRING)	ID первичный ключ
ELD_NOTIFICATION	ID(INTEGER), CONTENT(STRING), LINK(STRING), DATA_TIME(DATETIME), TITLE(STRING), NO_U_ID(INTEGER)	ID первичный ключ, NO_U_ID внешний ключ к таблице ELD_USERS

3.2.2 Схематичное представление

Логическая модель базы данных представлена на рисунке 4.

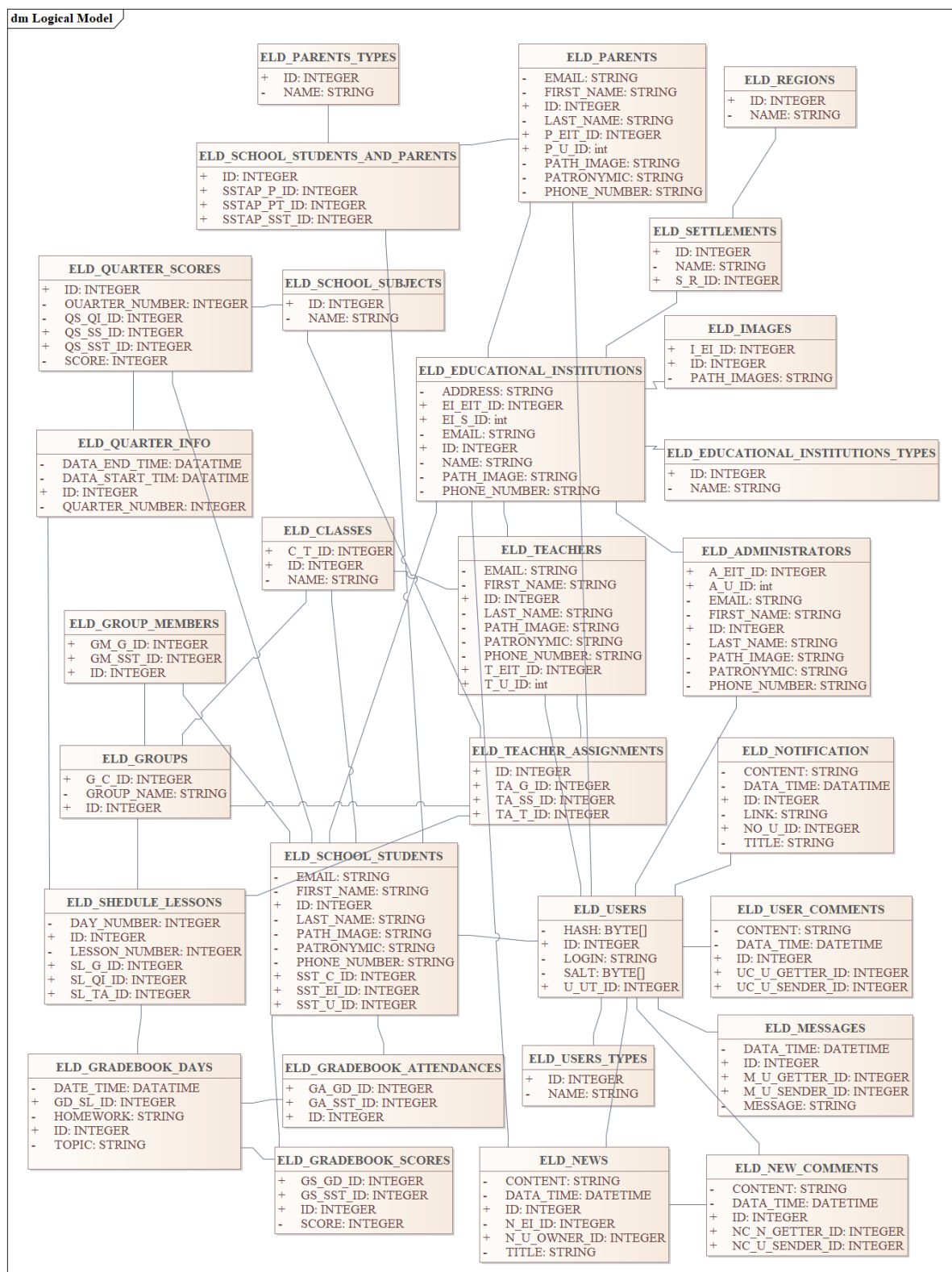


Рисунок 4 — Логическая модель базы данных

Физическая модель базы данных представлена на рисунке 5.



3.4 Разработка алгоритмов программного средства и отдельных модулей

3.3.1 Алгоритм ПС

Приложение начинает работу с чтения данных из файлов настроек и профиля пользователя. После загрузки данных выполняется авторизация. При успешной проверке открывается главное окно, интерфейс которого адаптируется под роль пользователя. Пользователь взаимодействует с программой через элементы управления. Каждое действие обрабатывается последовательно — валидируются входные параметры, выполняются операции с базой данных, а результаты выводятся на экран. Основная логика сосредоточена в обработке событий.

Алгоритм ПС представлен на рисунках 6, 7.

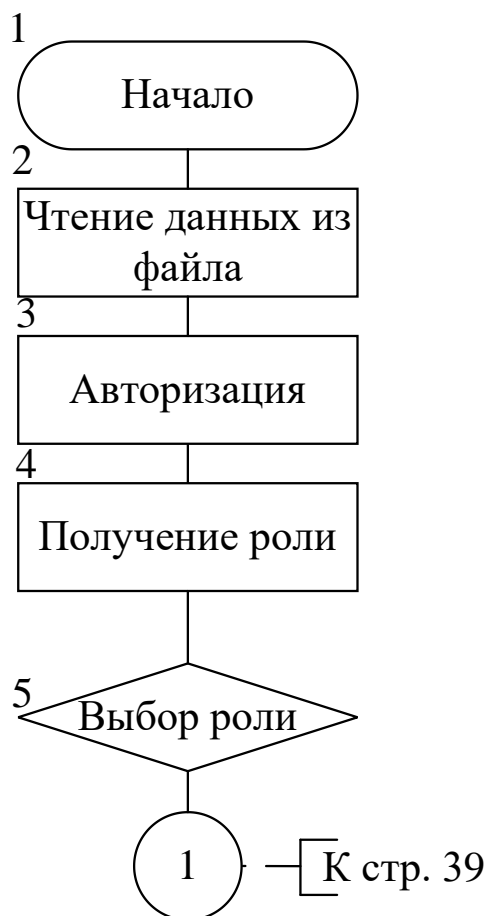


Рисунок 6 — Алгоритм ПС (Часть 1)



Рисунок 7 — Алгоритм ПС (Часть 2)

1.3.2 Алгоритм выбора элементов в списке

Алгоритм выбора элемента в списке начинается с загрузки доступных действий для выбранного объекта, после чего пользователю отображаются опции: редактирование (переход к форме с текущими данными), описание (просмотр в режиме только чтения), удаление (с подтверждением через диалоговое окно) или переход к связанным объектам. При удалении система запрашивает подтверждение, а при переходе — выводит новый список объектов, повторяя алгоритм для выбранного элемента. Все операции валидируются на соответствие правам доступа, обеспечивая безопасное взаимодействие с данными. Алгоритм завершается после выполнения действия или возврата к предыдущему шагу при отмене.

Алгоритм выбора элемента в списке представлен на рисунке 8.

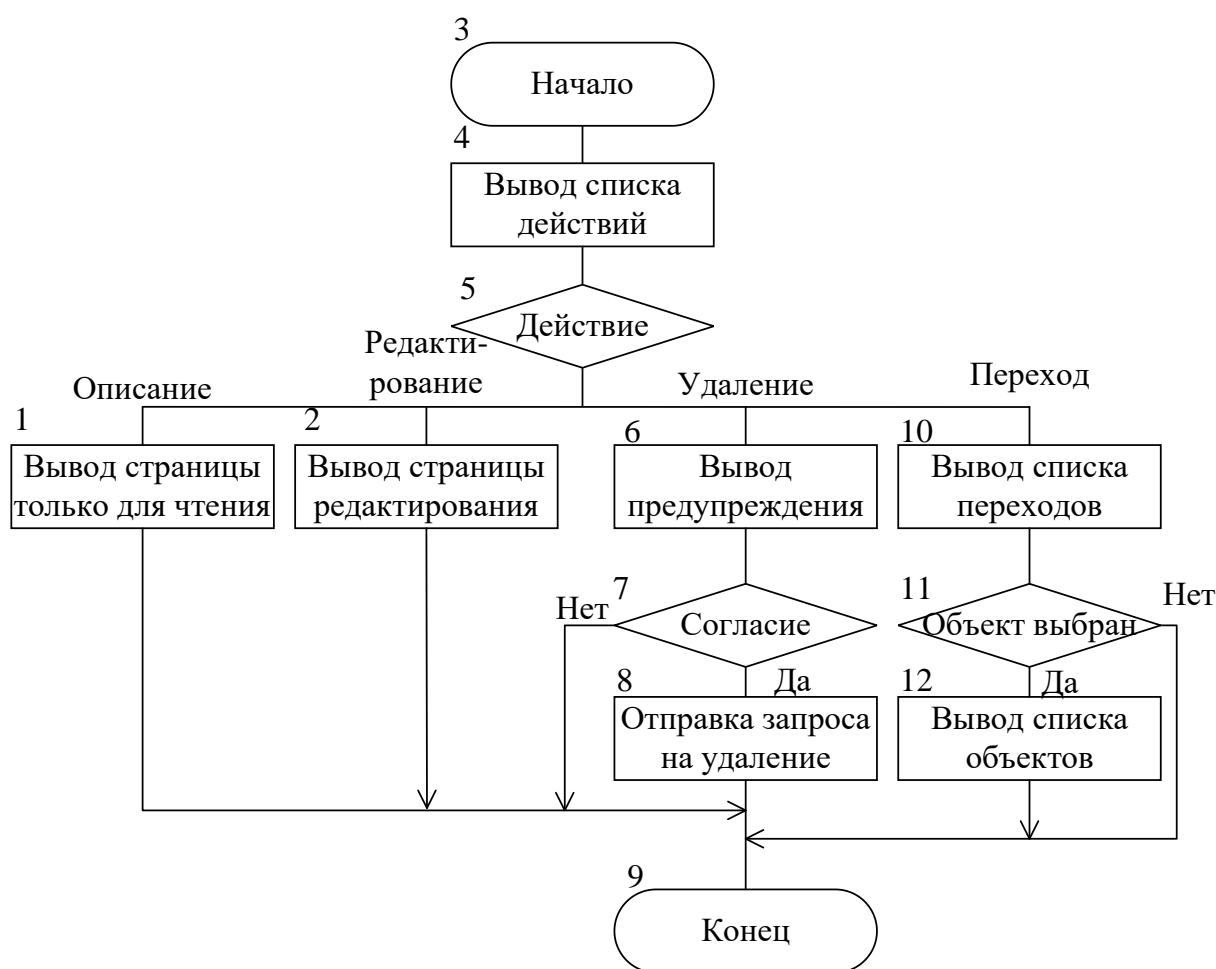


Рисунок 8 — Алгоритм выбора элемента в списке

1.3.3 Алгоритм общения с сервером

Алгоритм предназначен для стандартизации взаимодействия с сервером, обеспечивая единый подход к обработке HTTP-запросов и ошибок. Процесс начинается с определения типа запроса, после чего отправляется

соответствующий запрос к серверу. При успешном ответе (статус 200-299) данные возвращаются для дальнейшей обработки. Если возникает ошибка, система анализирует её тип. Метод завершается либо возвратом данных, либо уведомлением о проблеме.

Алгоритм общения с сервером представлен на рисунке 9.

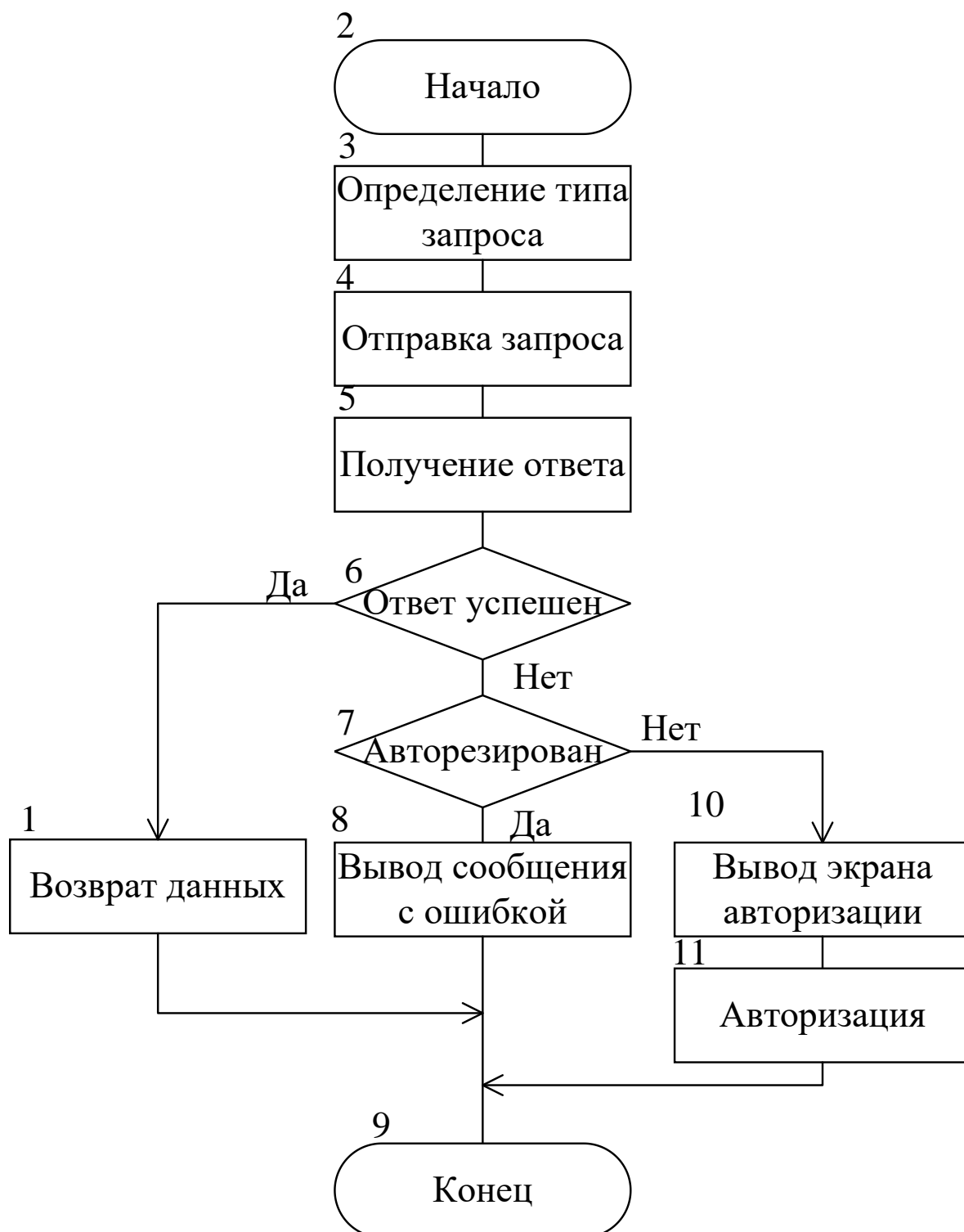


Рисунок 9 — Алгоритм общения с сервером

1.3.4 Алгоритм адаптивной вёрстки



Рисунок 10 — Алгоритм адаптивной вёрстки

Алгоритм предназначен для динамической адаптации интерфейса при изменении размеров окна или других условий отображения. Он обеспечивает единый подход к перераспределению и отображению столбцов на экране.

Процесс начинается с регистрации события изменения окна. Затем система инициирует перерасчет — определяется максимально возможное количество столбцов и соответствующая им ширина с учетом текущих параметров окна. Если количество имеющихся столбцов превышает допустимое — лишние элементы временно скрываются. После этого формируется набор столбцов для отображения. На финальном этапе производится визуализация — добавленные столбцы отображаются в интерфейсе с заданной шириной. Таким образом, достигается адаптивное отображение данных без перегрузки пользовательского интерфейса.

Алгоритм адаптивной вёрстки представлен на рисунке 10.

1.3.5 Алгоритм расчёта размера столбцов

Алгоритм предназначен для вычисления оптимального количества отображаемых столбцов и их ширины на основе плотности пикселей и текущей ширины окна приложения.

На первом этапе осуществляется получение плотности пикселей экрана устройства. Затем определяется текущая ширина окна приложения.

Если полученное количество столбцов превышает верхний предел (3) — используется максимальное значение. Если результат меньше допустимого минимума (1) — устанавливается минимальное значение.

После определения числа столбцов происходит расчет ширины экрана в пикселях, и на его основе вычисляется ширина одного столбца с применением дополнительного коэффициента.

На выходе алгоритм возвращает рассчитанное количество столбцов и соответствующую им ширину, обеспечивая адаптивное отображение данных в зависимости от устройства и текущих условий.

Алгоритм расчета размера столбцов представлен на рисунке 11.

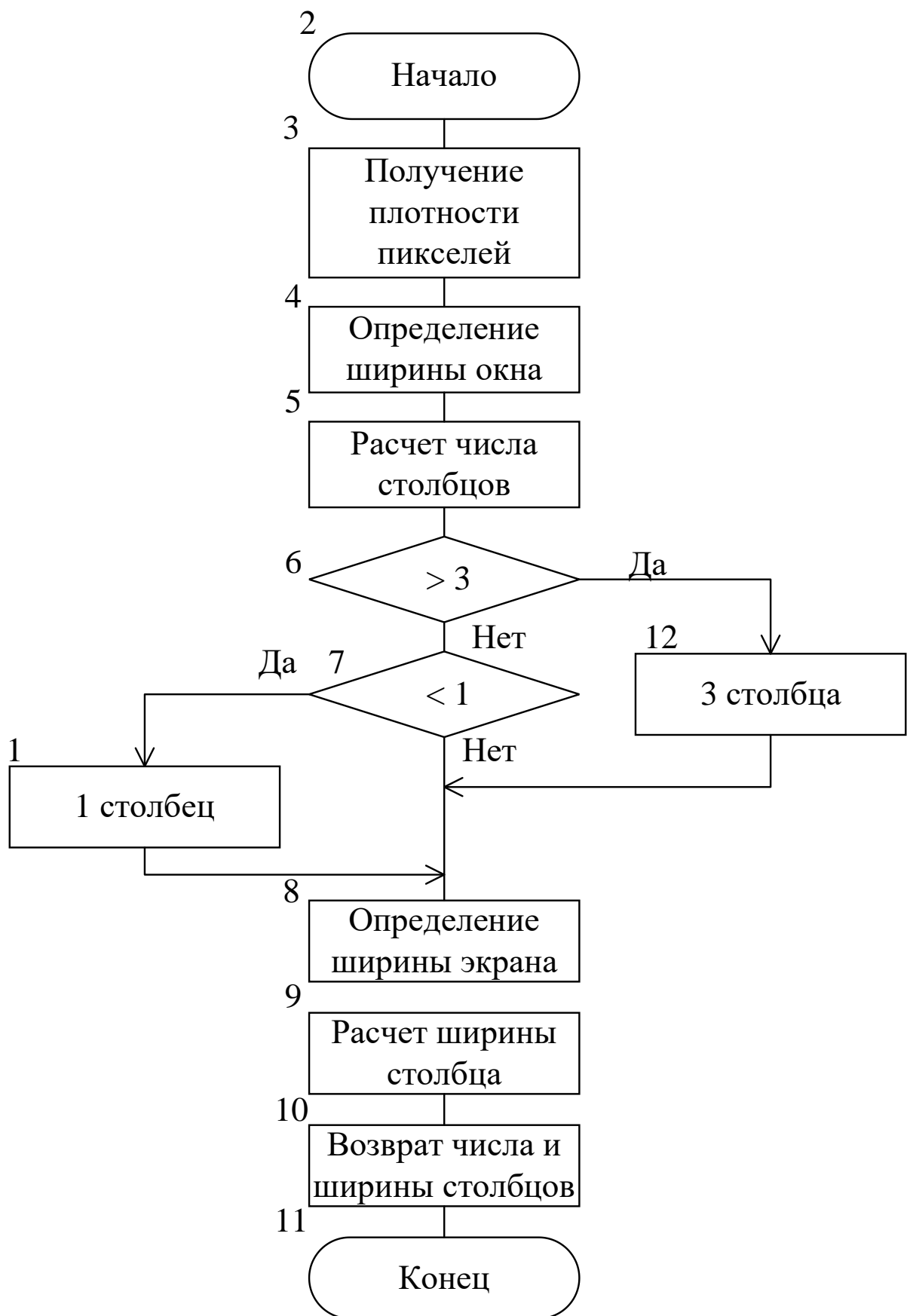


Рисунок 11 — Алгоритм расчёта размера столбцов

4 СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА

4.1 Выбор инструментов разработки

4.1.1 Язык программирования C#

C# — это объектно-ориентированный язык программирования, разработанный корпорацией Microsoft в рамках платформы .NET. Он был создан в начале 2000-х годов и с тех пор постоянно развивается, становясь одним из самых популярных языков для создания программного обеспечения. C# используется для разработки веб-приложений, десктопных приложений, мобильных решений, а также игр.[11]

Особенности C#:

- Объектно-ориентированность — поддержка классов, интерфейсов, наследования и полиморфизма.
- Безопасность типов — предотвращение ошибок, связанных с некорректными операциями с памятью.
- Совместимость с .NET — возможность работы с библиотеками и фреймворками .NET.
- Автоматическое управление памятью — система сборки мусора освобождает память автоматически.
- Асинхронность — поддержка асинхронного программирования с использованием `async` и `await`.
- C# активно используется в корпоративной среде благодаря своей надежности, безопасности и удобству работы.

4.1.2 Фреймворк .NET MAUI

.NET Multi-platform App UI (.NET MAUI) — это кроссплатформенный фреймворк от Microsoft, предназначенный для создания мобильных и десктопных приложений с использованием единой кодовой базы. Он является развитием Xamarin.Forms и позволяет разрабатывать приложения для Windows, macOS, iOS и Android.[12]

Преимущества .NET MAUI:

- Единая кодовая база — один проект для всех платформ.
- Гибкость интерфейса — поддержка MVU (Model-View-Update) и традиционного MVVM (Model-View-ViewModel).
- Поддержка C# и XAML — разработка UI с использованием знакомых инструментов.
- Кроссплатформенные API — возможность работы с файлами, сетью, сенсорами и другими функциями устройств.
- Оптимизированная производительность — улучшенная работа с памятью и графическими интерфейсами.

4.1.3 Community Toolkit для .NET MAUI

.NET MAUI Community Toolkit — это библиотека, созданная для расширения возможностей .NET MAUI.[13]

Она включает в себя:

- Дополнительные элементы управления (Expander, Popup и другие).
- Конвертеры данных (например, преобразование bool в Visibility).
- Расширенные анимации и стилизация UI.
- Использование Community Toolkit значительно упрощает разработку, помогая создавать удобные и функциональные интерфейсы.

4.1.4 Интегрированная среда разработки Visual Studio

Visual Studio — это мощная интегрированная среда разработки (IDE) от Microsoft. Она поддерживает множество языков программирования, включая C#, и предоставляет удобные инструменты для написания, отладки и тестирования кода.[14]

Ключевые возможности Visual Studio:

- Редактор кода с подсветкой синтаксиса и автодополнением.
- Инструменты отладки — возможность установки точек останова, просмотра переменных в реальном времени.
- Интеграция с Git и GitHub — удобное управление версиями.
- Поддержка контейнеров и облачных сервисов — развертывание приложений в Azure.
- Благодаря Visual Studio разработка становится более удобной и эффективной.

4.1.5 Oracle Database

Oracle Database — это одна из самых мощных и надежных систем управления базами данных (СУБД). Она широко используется в корпоративных решениях благодаря своей высокой производительности, безопасности и поддержке работы с большими объемами данных.[15]

Преимущества Oracle Database:

- Высокая масштабируемость — поддержка кластерных решений.
- Надежность и безопасность — расширенные механизмы резервного копирования и восстановления данных.
- Поддержка SQL и PL/SQL — удобный язык программирования для работы с БД.
- Интеграция с облачными сервисами — возможность работы с Oracle Cloud.
- Oracle Database является основой для множества корпоративных приложений, требующих высокой надежности.

4.1.6 PL/SQL Developer

PL/SQL Developer — это специализированная среда разработки для работы с языком программирования PL/SQL в Oracle Database. Этот инструмент позволяет писать, отлаживать и тестировать SQL-скрипты и хранимые процедуры.[16]

Ключевые возможности PL/SQL Developer:

- Редактор кода с автодополнением.
- Инструменты отладки и профилирования.
- Поддержка работы с объектами базы данных.
- Гибкая система отчётов и анализа данных.
- Использование PL/SQL Developer значительно облегчает работу разработчиков баз данных, ускоряя процесс написания кода и его тестирования.

4.2 Разработка контроллеров

Для взаимодействия с серверной частью системы, используется интерфейс IController. Этот интерфейс предоставляет методы для выполнения основных операций с данными, таких как получение, добавление, редактирование и удаление объектов, а также для работы с изображениями.

Ключевые методы:

– GetAll(long id = -1) — Получение всех объектов или объектов, связанных с определенным id. Этот метод используется для формирования списков в интерфейсе, например, для отображения списка образовательных учреждений или пользователей.

– GetById(long id) — Получение объекта по его идентификатору. Это необходимо для отображения детализированной информации о выбранном объекте, такой как описание образовательного учреждения, информация о пользователе и т.д.

– Add(object request) — Метод добавления нового объекта. Используется для создания нового элемента в базе данных, например, добавление нового образовательного учреждения или пользователя через соответствующий интерфейс.

– Edit(object request) — Метод редактирования существующего объекта. После того как пользователь выбрал элемент для редактирования, данные отправляются на сервер для обновления информации.

– Delete(long id) — Удаление объекта по идентификатору. Этот метод позволяет удалять элементы, такие как пользователи или образовательные учреждения, через интерфейс с подтверждением.

– AddImage(long id, FileResult image) — Метод добавления изображения для объекта. Этот метод может быть использован для загрузки изображений, связанных с объектами, например, для добавления логотипов образовательных

учреждений или фотографий пользователей.

4.3 Разработка пользовательского интерфейса

Основой архитектуры интерфейса является использование обобщений и наследования, что позволяет создать гибкую структуру для отображения различных типов данных. Такой подход обеспечивает возможность легко адаптировать интерфейс для работы с разнообразными объектами, минимизируя дублирование кода и улучшая поддержку различных типов данных.

В интерфейсе используется `HorizontalStackLayout`, который служит контейнером для элементов, выстраиваясь по горизонтали. В зависимости от ширины экрана, количество элементов (`ScrollView`) в стеке меняется так, чтобы поместить максимальное количество элементов в доступном пространстве. Таким образом, экран всегда будет заполняться оптимально, показывая последние N элементов, где N — это количество элементов, которое помещается на текущем экране.

4.3.1 `IBaseViewListCreator`

Для работы с разнообразными списками объектов был реализован интерфейс `IBaseViewListCreator`, который обеспечивает единый стандарт для создания и отображения списков на экране. Все классы, реализующие данный интерфейс, обязаны иметь метод `Create`, который возвращает объект типа `ScrollView`. Этот метод гарантирует создание интерфейса в соответствии с установленными стандартами и позволяет гибко управлять контентом.

```
public interface IBaseViewListCreator
{
    ScrollView Create(HorizontalStackLayout mainStack,
                    List<ScrollView> viewList,
                    long objetParentId = -1,
                    bool readOnly = false,
                    long objetPreParentId = -1);
}
```

4.3.2 `BaseViewListCreator`

Для обработки и отображения списков объектов была разработана абстрактная реализация `BaseViewListCreator`. Этот класс обрабатывает создание и отображение объектов в интерфейсе, обеспечивая гибкость и масштабируемость при работе с различными типами данных.

Ключевые методы:

– `Create` — главный метод, который создает пользовательский интерфейс для отображения списка объектов. Он принимает параметры для настройки контейнера элементов, списка отображаемых объектов, а также опций для работы с родительскими объектами и режимом только для чтения. Этот метод строит интерфейс с фильтрами, кнопками и элементами списка.

- CreateTitile — метод, который создает заголовок для интерфейса. Он формирует строку с названием для отображения в сетке.
- CreateFilterUI — создает элементы интерфейса для фильтрации списка объектов. Здесь можно добавить дополнительные фильтры, если они требуются для отображения данных.
- CreateGetButton — создает кнопку "Найти", которая запускает процесс получения данных для списка. При нажатии на кнопку будет выполнена асинхронная операция загрузки списка.
- CreateAddButton — создает кнопку "Добавить", которая позволяет добавить новый элемент в список. Этот метод создает новый объект и обновляет список отображаемых элементов.
- CreateColumnName — метод для создания заголовков столбцов, если это необходимо для отображаемых данных.
- GetButtonClicked — обработчик события для кнопки "Найти", который инициирует загрузку данных с сервера и обновление интерфейса.
- AddButtonClicked — обработчик события для кнопки "Добавить", который добавляет новый объект в список и перерисовывает интерфейс.
- CreateListUI — метод для создания интерфейса списка. Он вызывает загрузку данных и формирует список с использованием элементов, созданных соответствующими классами.
- ChageListHandler — обработчик события изменения списка, который запускает перерисовку интерфейса с обновленными данными.
- GetList — асинхронный метод для получения списка объектов с помощью контроллера. Он получает данные с сервера и фильтрует их по нужным критериям.
- FilterList — метод, предназначенный для фильтрации полученных данных перед их отображением (пока не используется).
- ChangeList — метод, вызывающий событие обновления списка.

Обобщения:

- TResponse: тип данных для ответа от сервера (например, объект с данными для отображения).
- TRequest: тип данных для запроса, который отправляется на сервер.
- TController: контроллер, отвечающий за обработку запросов и получение данных.
- TViewElemCreator: создатель элементов интерфейса для отображения элементов списка.
- TViewObjectCreator: создатель объекта для отображения подробной информации.

```
public class BaseViewListCreator<TResponse, TRequest,
TController, TViewElemCreator, TViewObjectCreator> :
IBaseViewListCreator
    where TResponse : BaseResponse, new()
    where TRequest : BaseRequest, new()
```

```

where TController : IController, new()
where TViewElemCreator :
    BaseViewElemCreator<TResponse, TRequest,
        TController, TViewObjectCreator>, new()
where TViewObjectCreator :
    BaseViewObjectCreator<TResponse, TRequest,
        TController>, new()

```

Этот пример показывает, как можно использовать BaseViewListCreator для создания интерфейса для списка образовательных учреждений.

```

public sealed class
    EducationalInstitutionViewListCreator :
        BaseViewListCreator<EducationalInstitutionResponse,
            EducationalInstitutionRequest, EducationalInstitutionController,
            EducationalInstitutionViewElemCreator, EducationalInstitutionViewObjectCreator>

```

4.3.3 BaseViewElemCreator

Класс BaseViewElemCreator является абстрактной основой для создания элементов интерфейса для отображения объектов в списках, управляемых классом BaseViewListCreator. Он использует обобщенные типы, что позволяет работать с различными типами данных и адаптировать поведение элементов для разных объектов, таких как пользователи, классы, учреждения и другие сущности.

Ключевые методы:

- Create — основной метод для создания элемента интерфейса. Он настраивает и отображает элементы для списка объектов, такие как кнопки, текстовые поля, изображения и другие элементы управления.
- CreateUI — абстрактный метод, который должен быть переопределен в наследующих классах для создания конкретного UI (пользовательского интерфейса) для элементов. Этот метод вызывается в процессе создания каждого элемента, чтобы определить его визуальное представление.
- GestureTapped — метод для обработки событий при нажатии на элемент. Например, при нажатии на элемент интерфейса может быть выполнена операция редактирования, удаления или перехода к другому объекту.
- ShowInfo — метод, отображающий информацию о выбранном объекте. В этом методе создается новый элемент интерфейса с детализированной информацией о выбранном объекте и добавляется в список.
- Edit — метод для редактирования выбранного объекта. Он создает элемент интерфейса с полями для редактирования данных объекта и заменяет текущий элемент интерфейса.
- Delete — метод для удаления объекта. Перед удалением отображается окно подтверждения, и если пользователь соглашается, объект удаляется.
- ChangeList — метод для обновления списка, когда объект был удален или изменен. Этот метод инициирует перерисовку интерфейса с обновленными данными.

Обобщения:

- TResponse: тип данных для ответа от сервера (например, объект с данными для отображения).
- TRequest: тип данных для запроса, который отправляется на сервер.
- TController: контроллер, отвечающий за обработку запросов и получение данных.
- TViewObjectCreator: создатель объекта для отображения подробной информации.

```
public class BaseViewElemCreator<TResponse, TRequest,  
TController, TViewObjectCreator>  
    where TResponse : BaseResponse, new()  
    where TRequest : BaseRequest, new()  
    where TController : IController, new()  
    where TViewObjectCreator :  
        BaseViewObjectCreator<TResponse, TRequest,  
TController>, new()
```

Этот пример показывает, как можно использовать BaseViewElemCreator для создания элементов списка образовательных учреждений.

```
public class EducationalInstitutionViewElemCreator :  
BaseViewElemCreator<EducationalInstitutionResponse,  
EducationalInstitutionRequest, EducationalInstitutionController,  
EducationalInstitutionViewObjectCreator>
```

4.3.4 BaseViewObjectCreator

Класс BaseViewObjectCreator представляет собой основу для создания и отображения объектов, связанных с управлением конкретными сущностями в интерфейсе. Это важная часть архитектуры, которая позволяет гибко работать с различными типами данных и объектов, такими как образовательные учреждения, пользователи и другие сущности.

Ключевые методы:

- Create — основной метод, который создает и инициализирует элемент интерфейса для отображения объекта. В нем устанавливаются параметры состояния компонента (новый, редактируемый или только для чтения), и создается интерфейс с использованием различных элементов.

- CreateUI — метод, который используется для создания и настройки интерфейса для отображаемого объекта. Этот метод может быть переопределен в наследуемых классах для добавления специфичных элементов интерфейса для конкретных объектов.

- SaveButtonClicked — метод для обработки нажатия на кнопку "Сохранить". В нем осуществляется проверка состояния компонента (новый объект или редактирование) и выполнение соответствующего запроса на сервер для добавления или редактирования объекта.

Обобщения:

- TResponse: тип данных для ответа от сервера (например, объект с данными для отображения).
- TRequest: тип данных для запроса, который отправляется на сервер.
- TController: контроллер, отвечающий за обработку запросов и получение данных.

```
public class BaseViewObjectCreator<TResponse, TRequest, TController>  
    where TResponse : BaseResponse, new()  
    where TRequest : BaseRequest, new()  
    where TController : IController, new()
```

Этот пример показывает, как можно использовать BaseViewObjectCreator для создания элементов списка образовательных учреждений.

```
public class EducationalInstitutionViewObjectCreator :  
    BaseViewObjectCreator<EducationalInstitutionResponse,  
        EducationalInstitutionRequest, EducationalInstitutionController>
```

4.3.5 Представление элементов в пользовательском интерфейсе

На рисунке 12 представлен пользовательский интерфейс с использованием приведённых в пункте 4.3 классов.

В левом столбце интерфейса отображаются списки объектов, для чего используется класс наследник BaseViewListCreator. Например, для работы с образовательными учреждениями создается класс EducationalInstitutionViewListCreator, который наследует BaseViewListCreator. Этот наследник реализует все необходимые методы для создания и отображения списка образовательных учреждений. Элементы этого списка управляются с помощью BaseViewElemCreator и его конкретных реализаций, таких как EducationalInstitutionViewElemCreator, которые отвечают за создание и настройку каждого элемента в списке.

В правом столбце отображается подробная информация о выбранном объекте. Здесь используется класс BaseViewObjectCreator и его специализированные наследники, такие как EducationalInstitutionViewObjectCreator для образовательных учреждений. Этот наследник класса BaseViewObjectCreator отвечает за создание и отображение интерфейса для детализированной информации об объекте. Он включает логику для отображения, редактирования и сохранения данных. В частности, каждый объект, например, образовательное учреждение, управляется через наследников BaseViewObjectCreator, предоставляя гибкость в работе с разными типами данных.

Список учебных заведений

Область

Минская область

Населённый пункт

г.Солигорск

Название

ГУО ...

Найти

Добавить

Название

Школа Алексея

Регион

Минская область

Населённый пункт

г. Слуцк

Название


Школа Короля Без Алексеев

Регион

Минская область

Населённый пункт

г. Солигорск



Название

Школа Алексея

Регион

Минская область

Населённый пункт

г. Слуцк

Адресс

23

Email

23

Телефон

23

Рисунок 12 — Интерфейс пользователя

5 ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ РЕЗУЛЬТАТОВ

При разработке приложения для создания ПС тестирование проводилось в соответствии с тестовыми сценариями, представленными в таблице 5.

Таблица 5 – Набор тестовых сценариев позитивного тестирования

Тест	Шаги теста	Ожидаемый результат	Результат
Авторизация пользователя с корректными данными	1. Открытие страницы авторизации. 2. Ввод корректных данных пользователя. 3. Попытка авторизации.	Перенаправление на соответствующую страницу пользователя.	Тест пройден успешно
Создание пользователя администратором с корректными данными	1. Открытие страницы добавления пользователя. 2. Ввод данных (имя, номер, логин пароль, ...). 3. Сохранение данных.	Пользователь добавлен с правильными данными и доступом.	Тест пройден успешно
Проверка отображения расписания на мобильном устройстве	1. Открытие раздела расписания. 2. Проверка отображения всех уроков и их времени.	Расписание отображается корректно на мобильном устройстве.	Тест пройден успешно

Продолжение таблицы 5 – Набор тестовых сценариев позитивного тестирования

Создание новости администратором	1. Открытие страницы создания новости. 2. Ввод данных (заголовок, текст). 3. Публикация новости.	Новость публикуется и отображается на главной странице.	Тест пройден успешно
Проверка безопасности паролей при авторизации	1. Ввод правильных данных для входа. 2. Ввод неверного пароля и проверка реакции системы.	Система блокирует неудачные попытки входа и сообщает об ошибке.	Тест пройден успешно
Редактирование данных пользователя	1. Вход под учетной записью администратора. 2. Редактирование имени пользователя.	Данные пользователя обновляются корректно и отображаются.	Тест пройден успешно
Выход пользователя из учетной записи	1. Нажатие на кнопку выхода. 2. Проверка, что пользователь выходит из аккаунта.	Пользователь успешно выходит из системы и перенаправляется на экран авторизации.	Тест пройден успешно
Добавление новой школы в систему	1. Открытие страницы добавления школы. 2. Ввод данных (название, адрес, преподаватели, расписание). 3. Сохранение школы.	Школа добавлена в систему и отображается в списке школ.	Тест пройден успешно

Продолжение таблицы 5 – Набор тестовых сценариев позитивного тестирования

Проверка автозаполнения данных при входе	1. Запуск	Отображение страницы пользователя	Тест пройден успешно
Проверка создания групп и подгрупп для уроков	1. Открытие страницы создания групп. 2. Создание новой группы. 3. Проверка отображения группы.	Группа успешно создается и отображается.	Тест пройден успешно
Проверка правильности отображения новостей в системе	1. Открытие страницы новостей. 2. Проверка отображения всех опубликованных новостей.	Все новости отображаются в хронологическом порядке.	Тест пройден успешно
Проверка удаления школы	1. Открытие страницы списка школ. 2. Удаление школы.	Школа и все её объекты удаляется из системы и больше не отображается.	Тест пройден успешно
Проверка отображения информации о школьных объектах	1. Открытие страницы школы. 2. Проверка отображения всех объектов школы (кабинеты, спортивные залы и т.д.).	Все объекты школы отображаются корректно на странице.	Тест пройден успешно

Продолжение таблицы 5 – Набор тестовых сценариев позитивного тестирования

Проверка работы с фотографиями в профиле	1. Вход в профиль. 2. Загрузка и сохранение новой фотографии профиля. 3. Проверка, что фото отображается.	Фотография успешно загружается и отображается в профиле пользователя.	Тест пройден успешно
Проверка работы с типами пользователей (администратор, учитель, ученик)	1. Вход под различными ролями. 2. Проверка доступности функционала для каждой роли.	Доступ к функционалу ограничен в зависимости от роли пользователя.	Тест пройден успешно
Проверка корректности отображения домашнего задания у ученика	1. Создание домашнего задания. 2. Проверка отображения задания в личном кабинете ученика.	Домашнее задание отображается корректно в профиле ученика.	Тест пройден успешно

6 РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ

6.1 Развёртывание серверной части

6.1.1 Установка зависимостей в Ubuntu

Обновление системы.

```
sudo apt update && sudo apt upgrade -y
```

Установка докера.

```
sudo apt install docker.io docker-compose -y  
sudo systemctl enable --now docker
```

Установка Java.

```
sudo apt install openjdk-17-jdk -y
```

Установка Ngix;

```
sudo apt install nginx -y
```

6.1.2 Запуск Oracle Database в Docker

Проверка наличия образа.

```
docker images | grep gvenzl/oracle-free
```

При отсутствии образа выполнить загрузку.

```
docker pull gvenzl/oracle-free:23-slim
```

Создание контейнера.

```
docker run -d \  
  --name oracle \  
  -p 1521:1521 \  
  -e ORACLE_PASSWORD=<ваш_пароль> \  
  gvenzl/oracle-free:23-slim
```

Замените <ваш_пароль> на надежный пароль для администратора БД.

Подключитесь к контейнеру и выполните SQL-скрипты для создания пользователя и схемы.

```
docker exec -it oracle sqlplus sys/<ваш_пароль>@//localhost:1521/FREEPDB1 as  
sysdba
```

6.1.3 Запуск и настройка Ngix

Создание файла конфигурации.

```
sudo nano /etc/nginx/sites-available/meeting-rooms.conf
```

Поместить в файл.

```
server {  
    listen 80;
```

```

server_name <ip>;

# Включение MIME-типов
include /etc/nginx/mime.types;
default_type application/octet-stream;

# Одна директория для загрузки и скачивания
location /files/ {
    alias /var/www/files/;

    # Разрешение PUT-запросов (загрузка)
    dav_methods PUT;
    create_full_put_path on;
    dav_access user:rw group:rw all:r;
    client_max_body_size 100M;

    # Включение листинга (если нужно)
    autoindex on;
    autoindex_exact_size off;
    autoindex_localtime on;

    # Оптимизация кеширования
    expires 30d;
    add_header Cache-Control "public, no-transform";

    # Запрет выполнения PHP
    location ~ /\.php$ {
        deny all;
    }
}

```

Заменить <ip> на ip адрес сервера в глобальной сети.

Активация конфигурации.

```

sudo ln -s /etc/nginx/sites-available/meeting-rooms.conf /etc/nginx/sites-enabled/
sudo nginx -t && sudo systemctl reload nginx

```

6.1.4 Запуск Java сервер приложения

Разместить MeetingRooms-0.0.1-SNAPSHOT.jar в директории на сервере.

Запуск сервера. Сервер автоматически создаст таблицы в базе данных если были выполнены предыдущие шаги.

```

nohup java -jar MeetingRooms-0.0.1-SNAPSHOT.jar \
  --spring.datasource.url=jdbc:oracle:thin:@localhost:1521/FREEDB1 \
  --spring.datasource.username=meeting_rooms \
  --spring.datasource.password=<пароль_пользователя_БД> \
  > app.log 2>&1 &

```

6.2 Установка клиентской части

Скачать установочный файл для выбранной платформы.

Установить.

Принять разрешения.

6.3 Руководство по использованию

После авторизации в качестве оператора пользователю открывается интерфейс панели операторов. Если оператор от министерства будет доступен список школ, иначе школа, к которой причислен оператор. Панель позволяет просматривать, добавлять, редактировать, удалять объекты.

Страница представлена на рисунке 13.

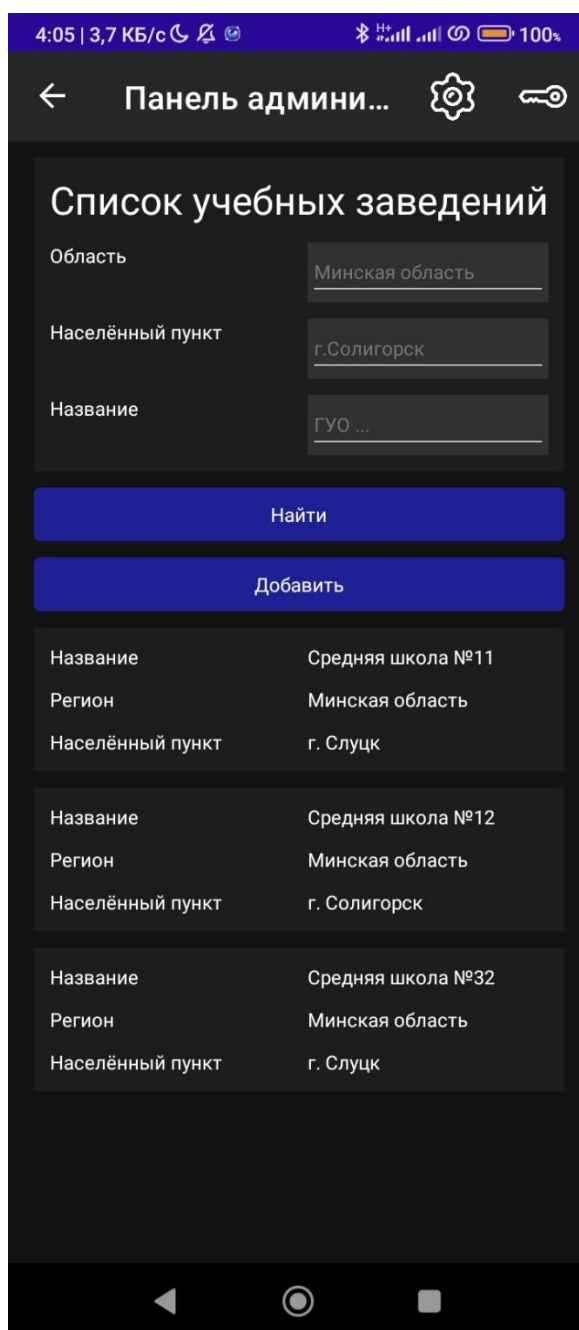


Рисунок 13 — Страница панели оператора

Рисунок 14, 15 показывает навигацию в рамках панели. Всплывающие окна появляются при нажатии на объект из списка на рисунке 13.

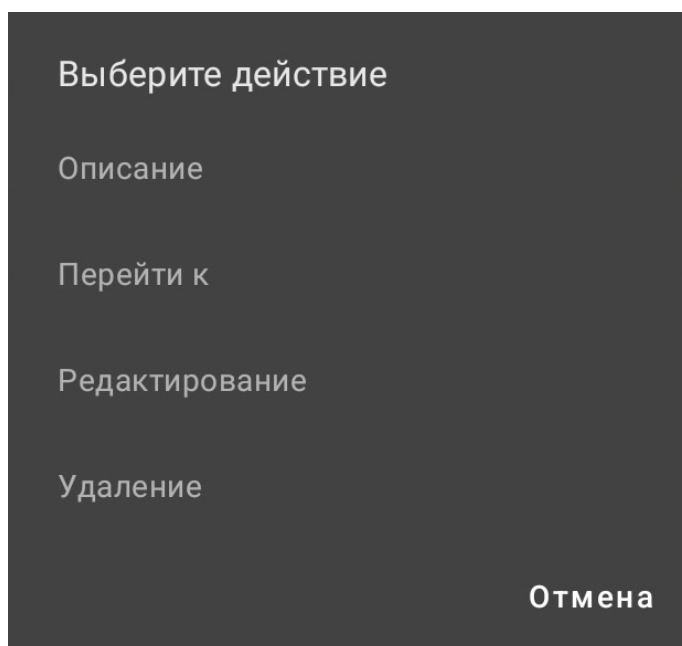


Рисунок 14 — Панель действий с объектами

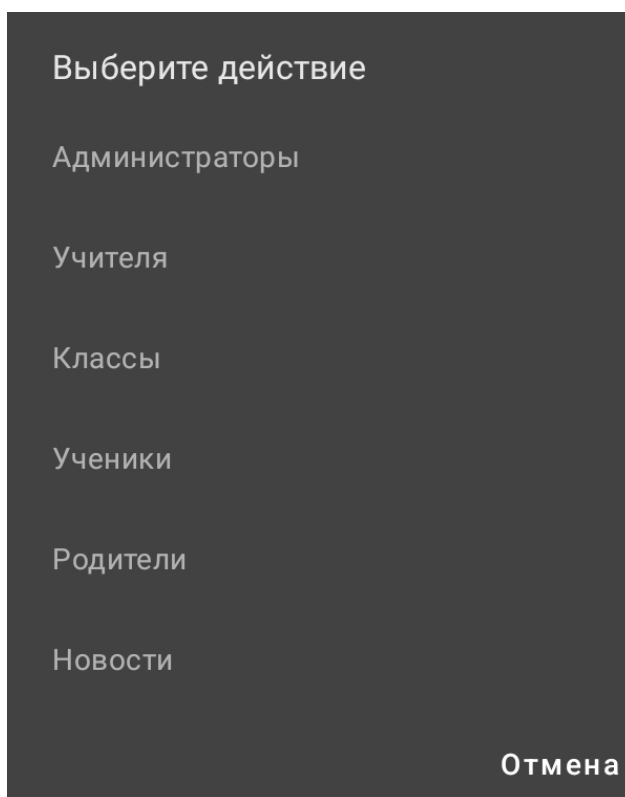


Рисунок 15 — Панель выбора объекта

По нажатию на значок шестерёнки на рисунке 13 открывается экран настроек. Экран позволяет выбрать цветовое оформление и масштаб интерфейса. Экран представлен на рисунке 16.

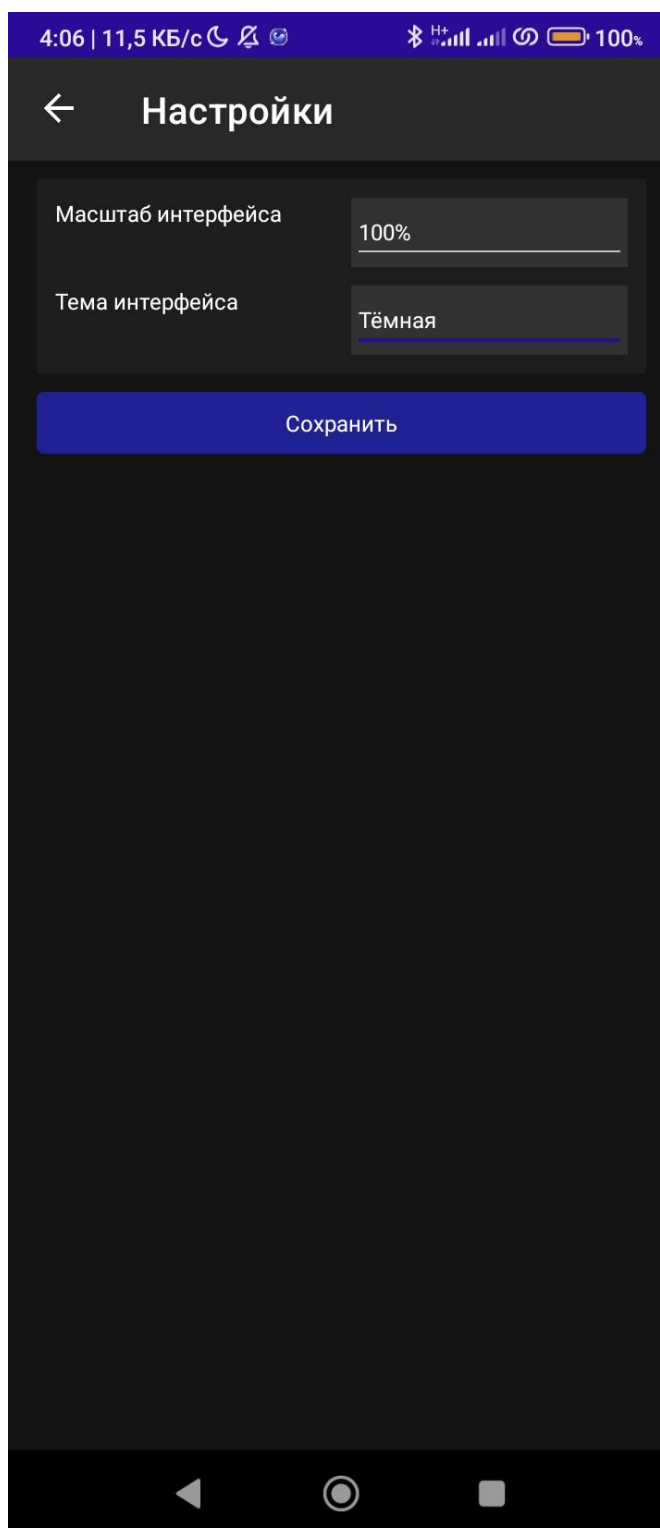


Рисунок 16 — Экран настроек

При авторизации в качестве не оператора открывается профиль пользователя. В нижней части экрана расположены кнопки навигации по экранам. На рисунке 17 представлены кнопки навигации: профиль, чат, новости, расписание, дневник.

Профиль представлен на рисунке 17.

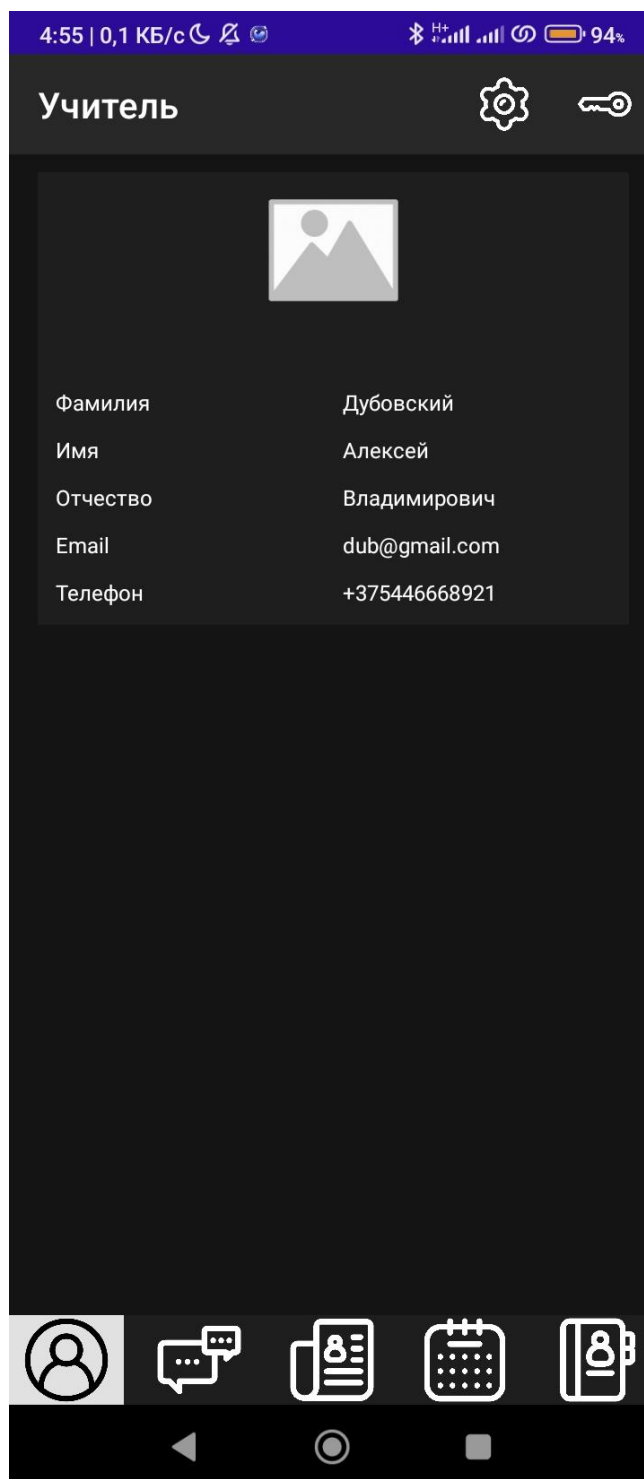


Рисунок 17 — Профиль пользователя

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЯ «ЭЛЕКТРОННЫЙ ДНЕВНИК» ДЛЯ УЧЁТА УСПЕВАЕМОСТИ В ОБРАЗОВАТЕЛЬНЫХ УЧРЕЖДЕНИЯХ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ SPRING

7.1 Описание функций, назначения и потенциальных пользователей программного средства

Целью разработки данного программного средства является создание комплексного решения для автоматизации ведения электронного дневника в образовательных учреждениях. Программное средство направлено на упрощение документооборота, повышение прозрачности учебного процесса, оперативный контроль успеваемости учащихся и улучшение взаимодействия между всеми участниками образовательного процесса. Внедрение данного решения позволит минимизировать ошибки при ведении учебной документации, снизить нагрузку на преподавателей и администрацию, а также повысить удобство доступа к актуальной информации для учеников и их родителей.

Программное средство разрабатывается в рамках инициативы Министерства образования Республики Беларусь и предназначено для внутреннего использования в образовательных учреждениях.

Программное средство обладает широким спектром функций, направленных на оптимизацию учебного процесса:

- Электронное расписание – отображение расписания занятий для учащихся и преподавателей.
- Контроль успеваемости – ведение электронного журнала с возможностью выставления оценок.
- Контроль посещаемости – ведение электронного журнала с возможностью выставления посещаемости.
- Домашнее задание – добавление и редактирование домашнего задания преподавателем.
- Интерактивное взаимодействие – встроенная система обмена сообщениями между учениками, преподавателями и администрацией.
- Права доступа и безопасность – разграничение уровней доступа к данным в зависимости от роли пользователя.
- Персонализированные настройки – настройка интерфейса и функционала в соответствии с потребностями конкретного образовательного учреждения.

Сегмент программных решений для ведения электронного дневника в образовательных учреждениях остается недостаточно развитым, что создает возможности для разработки специализированных инструментов,

соответствующих современным требованиям образовательного процесса. Разрабатываемое программное средство направлено на улучшение управления учебным процессом и взаимодействия между участниками образовательного процесса. Внедрение системы повысит уровень цифровизации в образовательных учреждениях, сократит затраты времени на ведение документации и обеспечит удобный доступ к актуальной информации для преподавателей, учащихся и родителей.

Преимущества по сравнению с аналогами:

- Кроссплатформенность (поддержка iOS, Android, Windows, macOS).
- Мобильный интерфейс.
- Поддержка широкого спектра учебных заведений. Обычно такие системы разрабатываются строго под конкретное учебное заведение.

7.2 Расчёт затрат на разработку программного средства

Расчет затрат на разработку программного средства состоит из следующих статей:

- затраты на основную заработную плату разработчиков;
- затраты на дополнительную заработную плату разработчиков;
- отчисления на социальные нужды;
- прочие затраты (амортизационные отчисления, расходы на электроэнергию, командировочные расходы, арендная плата за офисные помещения и оборудование, расходы на управление и реализацию и т.п.).

7.2.1 Затраты на основную заработную плату команды разработчиков

Расчёт основной заработной платы участников команды осуществляется по формуле 1:

$$Z_o = K_{\text{пр}} \sum_{i=1}^n Z_{\text{ч}i} \cdot t_i, \quad (1)$$

где n – количество исполнителей, занятых разработкой конкретного ПО;

$K_{\text{пр}}$ – коэффициент, учитывающий процент премий ($K_{\text{пр}} = 1,3$);

$Z_{\text{ч}i}$ – часовая заработная плата i -го исполнителя, р.;

t_i – трудоёмкость работ, выполняемых i -м исполнителем, ч.

Расчёт затрат на основную заработную плату осуществляется в табличной форме (таблица 6).

Месячная заработная плата определяется исходя по фактическим данным организации, на которой проходила преддипломная практика.

Часовая заработная плата определяется путём деления месячной заработной платы (оклад плюс надбавки) на количество рабочих часов в месяце (принимается равным 168 ч).

Трудоёмкость определяется исходя из сложности разработки программного обеспечения и объёма выполняемых им функций. Размер премии выбран равным 30%.

Таблица 6 – Расчет зарплат на основную заработную плату разработчиков

Наименование должности разработчика	Вид выполняемой работы	Месячная заработная плата, р.	Часовая заработная плата,	Трудоёмкость работ, ч	Сумма, р.
1	2	3	4	5	6
1. Бизнес-аналитик	Анализ рынка и продукта	3000	17,86	90	1607,41
2. Системный архитектор	Проектирование программного средства и управление разработкой	2800	16,67	220	3667,41
3. Инженер-программист	Разработка программного средства	2200	13,10	320	4192,01
4. Специалист по тестированию программного обеспечения	Обеспечение качества программного обеспечения	1500	8,93	150	1339,51
5. Дизайнер	Проектирование интерфейса программного средства	1800	10,71	100	1071,01
Итого					10537,84
Премия (30%)					3161,35
Всего основная заработная плата					13699,19

7.2.2 Затраты на дополнительную заработную плату разработчиков

Дополнительная заработная плата включает выплаты, предусмотренные законодательством о труде, и определяется по формуле 2:

$$З_д = \frac{З_о \cdot Н_д}{100}, \quad (2)$$

где Z_o – затраты на основную заработную плату, р.;
 H_d – норматив дополнительной заработной платы ($H_d = 15\%$).
Дополнительная заработная плата разработчиков составит:

$$Z_d = \frac{13699,19 \cdot 15}{100} = 2054,88 \text{ р.}$$

7.2.3 Отчисления на социальные нужды

Отчисления на социальные нужды определяются в соответствии с действующими законодательными актами по формуле 3:

$$P_{\text{соц}} = \frac{(Z_o + Z_d) \cdot H_{\text{соц}}}{100}, \quad (3)$$

где $H_{\text{соц}}$ – норматив отчислений от фонда оплаты труда ($H_{\text{соц}} = 35\%$).
Отчисления на социальные нужды составят:

$$P_{\text{соц}} = \frac{(13699,19 + 2054,88) \cdot 35}{100} = 5513,92 \text{ р.}$$

7.2.4 Прочие затраты

Прочие затраты включают затраты, как напрямую связанные с разработкой конкретного программного продукта в соответствии с планируемой суммой затрат на эти мероприятия, так и затраты, связанные с функционированием организации-разработчика в целом.

Данные затраты рассчитываются по формуле 4:

$$P_{\text{пз}} = \frac{Z_o \cdot H_{\text{пз}}}{100}, \quad (4)$$

где $H_{\text{пз}}$ – норматив прочих затрат ($H_{\text{пз}} = 40\%$).
Прочие затраты составят:

$$P_{\text{пз}} = \frac{13699,19 \cdot 40}{100} = 5479,68 \text{ р.}$$

Полученные значения затрат на разработку программного средства представлены в таблице 7.

Таблица 7 – Затраты на разработку программного средства

Наименование статьи затрат	Значение, р.
Основная заработная плата разработчиков	13699,19
Дополнительная заработная плата разработчиков	2054,88
Отчисления на социальные нужды	5513,92
Прочие затраты	5479,68
Общая сумма инвестиций в разработку	26747,67

7.3 Оценка результата от использования программного сервиса

Программное средство разрабатывается в рамках инициативы Министерства образования Республики Беларусь и предназначено для использования в образовательных учреждениях.

Разработка направлена на повышение удобства ведения учебного процесса, улучшение контроля за успеваемостью и вовлечённостью учащихся, а также автоматизацию взаимодействия между участниками образовательного процесса. Проект носит социально значимый характер и не преследует экономической выгоды.

7.3.1 Оценка неэкономического эффекта

Эффективная организация образовательного процесса требует современных инструментов для автоматизации ведения учебной документации, контроля успеваемости и взаимодействия между участниками образовательного процесса. Ведение бумажных журналов и ручной учёт оценок и посещаемости отнимает значительное время у педагогов, повышает риск ошибок и затрудняет оперативное получение информации учащимися и их родителями.

Разрабатываемое программное средство – электронный дневник – призвано упростить администрирование учебного процесса, автоматизировав ключевые аспекты работы образовательного учреждения. Основными задачами системы являются централизованное ведение расписания, контроль посещаемости, автоматизированный учёт оценок, управление домашними заданиями и обеспечение удобного взаимодействия между учащимися, преподавателями и родителями.

Программное средство разработано с учётом специфики образовательных учреждений Республики Беларусь и ориентировано на повышение прозрачности учебного процесса. Благодаря автоматизированному внесению данных преподаватели смогут сосредоточиться на образовательной деятельности, минимизируя рутинные административные задачи. Кроме того, исключается вероятность потери информации, так как все данные хранятся в цифровом формате.

Внедрение электронного дневника позволит достичь следующих значительных неэкономических эффектов:

Сокращение времени на ведение учета: Электронный дневник автоматизирует ввод данных, расчеты и подготовку отчетов, сокращая время, затрачиваемое преподавателями на эти задачи.

Сокращение времени на получение информации: Родители и учащиеся получают актуальную информацию о успеваемости и расписании в реальном времени, без необходимости ожидать отчеты.

Сокращение времени на административные процессы: Автоматизация рутинных задач сокращает время, затрачиваемое административным персоналом на обработку документов и составление отчетов.

Минимизация ошибок при учёте данных: Автоматизированный ввод данных снижает вероятность ошибок, связанных с человеческим фактором.

Упрощение взаимодействия: Электронный дневник облегчает обмен информацией между преподавателями, учащимися и родителями.

Повышение вовлечённости учащихся: Доступность информации о прогрессе в обучении способствует более осознанному подходу учащихся к учебе.

Оптимизация контроля за посещаемостью: Система позволяет администраторам быстро отслеживать пропуски и нарушения дисциплины.

Внедрение электронного дневника не приводит к прямым экономическим выгодам для образовательного учреждения, так как заработная плата педагогов определяется на основе количества проведенных уроков, а не времени, затрачиваемого на выполнение учебных задач.

Общая сумма инвестиций в разработку составит 26747,67р.

Таким образом, внедрение электронного дневника значительно повысит эффективность образовательного процесса, обеспечит удобные инструменты для администрирования учебного процесса, снизит временные затраты на ведение документации и улучшит взаимодействие всех участников образовательной деятельности.

ЗАКЛЮЧЕНИЕ

Разработанное мобильное приложение «Электронный дневник» для образовательных учреждений на базе технологии .NET MAUI представляет собой комплексное решение, направленное на цифровизацию учебного процесса. Проект успешно решает ключевые задачи, поставленные в рамках дипломной работы: автоматизацию учета успеваемости, оптимизацию взаимодействия между участниками образовательного процесса и обеспечение безопасности данных.

Проведенный анализ существующих платформ (School.by, MyClassroom, Google Classroom и др.) позволил выделить их слабые стороны и сформировать требования к новой системе. Это обеспечило разработку продукта с расширенным функционалом, включающим гибкую настройку ролевого доступа, интеграцию с внешними сервисами, поддержку многоплатформенности и адаптивный интерфейс.

Архитектура системы, основанная на клиент-серверной модели, обеспечивает масштабируемость и надежность. Использование Oracle Database гарантирует целостность и безопасность данных, а выбор .NET MAUI и C# позволил реализовать кроссплатформенность с единой кодовой базой. Логическая и физическая модели базы данных, а также алгоритмы работы приложения (авторизация, синхронизация, адаптивная верстка) были тщательно проработаны, что подтверждается результатами тестирования.

Технико-экономическое обоснование продемонстрировало, что внедрение системы сократит временные затраты педагогов на административные задачи, минимизирует ошибки ручного ввода данных и повысит прозрачность учебного процесса. Социальный эффект проекта выражен в улучшении коммуникации между школами, учениками и родителями, а также в повышении вовлеченности учащихся.

Перспективы развития системы включают внедрение модулей аналитики на основе ИИ для прогнозирования успеваемости, интеграцию с государственными образовательными порталами и расширение функционала для дистанционного обучения. Разработанное решение соответствует современным тенденциям цифровизации образования и обладает высоким потенциалом для внедрения в учебные заведения Республики Беларусь и других стран.

Таким образом, проект подтвердил свою актуальность, практическую значимость и готовность к реальному использованию, что открывает новые возможности для повышения качества образовательного процесса через инновационные технологии.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] schools.by [Электронный ресурс]. – Режим доступа: <https://schools.by>.
- [2] Обзор schools.by [Электронный ресурс]. – Режим доступа: <https://picktech.ru/product/schools-by>.
- [3] myClassroom: Class Tools [Электронный ресурс]. – Режим доступа: <https://minga.io/solutions/classroom-management-tools-teachers>.
- [4] Обзор myClassroom [Электронный ресурс]. – Режим доступа: <https://webcatalog.io/ru/apps/myclassroom>.
- [5] Edmodo [Электронный ресурс]. – Режим доступа: <https://bea-edmodo-instruction-rus.tilda.ws>.
- [6] Обзор Edmodo [Электронный ресурс]. – Режим доступа: <https://soware.ru/products/edmodo>.
- [7] Google Classroom [Электронный ресурс]. – Режим доступа: <https://edu.google.co.ug/workspace-for-education/products/classroom>.
- [8] Минобрнауки Татарстана. Методические рекомендации по использованию платформы Google classroom в процессе обучения [Электронный ресурс]. – Режим доступа: https://mon.tatarstan.ru/rus/file/pub/pub_2277963.pdf.
- [9] ClassDojo [Электронный ресурс]. – Режим доступа: <https://www.classdojo.com/ru-ru/points>.
- [10] Гродненский областной институт развития образования. Знакомство с ClassDojo [Электронный ресурс]. – Режим доступа: <https://groiro.by/об-институте/сервисы/новости/p-68273.html>.
- [11] Документация C# [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp>.
- [12] Документация MAUI [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/maui/?view=net-maui-9.0>.
- [13] Документация Community Toolkit для .NET MAUI [Электронный ресурс]. – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/communitytoolkit/maui>.
- [14] Документация Visual Studio 2022 [Электронный ресурс]. – Режим доступа: <https://visualstudio.microsoft.com/ru>.
- [15] Документация Oracle DataBase [Электронный ресурс]. – Режим доступа: <https://www.oracle.com/database>.
- [16] Документация PL/SQL Developer [Электронный ресурс]. – Режим доступа: <https://www.allroundautomations.com/products/pl-sql-developer>.

ПРИЛОЖЕНИЕ А
(обязательное)
Код программы

```
////////////////////////////////////  
// FILE:  
D:\WPS\ElectronicDiary\ElectronicDiary\Pages\AdminPageComponents\BaseView\BaseViewElemC  
reator.cs  
////////////////////////////////////  
  
using ElectronicDiary.Pages.AdminPageComponents.ClassView;  
using ElectronicDiary.Pages.AdminPageComponents.General;  
using ElectronicDiary.Pages.AdminPageComponents.NewsView;  
using ElectronicDiary.Pages.AdminPageComponents.ParentView;  
using ElectronicDiary.Pages.AdminPageComponents.SchoolStudentView;  
using ElectronicDiary.Pages.AdminPageComponents.UserView;  
using ElectronicDiary.Pages.Components.Elems;  
using ElectronicDiary.Web.Api.Other;  
using ElectronicDiary.Web.Api.Users;  
using ElectronicDiary.Web.DTO.Requests.Educations.Other;  
using ElectronicDiary.Web.DTO.Requests.Users;  
using ElectronicDiary.Web.DTO.Responses.Other;  
using ElectronicDiary.Web.DTO.Responses.Users;  
  
namespace ElectronicDiary.Pages.AdminPageComponents.BaseView  
{  
    public class BaseViewElemCreator<TResponse, TRequest, TController,  
TViewObjectCreator>  
    {  
        where TResponse : BaseResponse, new()  
        where TRequest : BaseRequest, new()  
        where TController : IController, new()  
        where TViewObjectCreator : BaseViewObjectCreator<TResponse, TRequest,  
TController>, new()  
        {  
            protected TResponse _baseResponse = new();  
            protected TRequest _baseRequest = new();  
            protected TController _controller = new();  
  
            protected HorizontalStackLayout _mainStack = [];  
            protected List<ScrollView> _viewList = [];  
            protected event Action ChageListAction = delegate { };  
  
            protected int _maxCountViews;  
            protected long _objectParentId;  
            protected bool _readOnly = false;  
  
            protected Grid _grid = [];  
            public Grid Create(HorizontalStackLayout mainStack,  
                                List<ScrollView> viewList,  
                                Action chageListAction,  
                                BaseResponse? baseResponse,  
                                int maxCountViews,  
                                long objetParentId,  
                                bool readOnly = false)  
            {  
                _mainStack = mainStack;  
                _viewList = viewList;  
                ChageListAction = chageListAction;  
                _baseResponse = baseResponse ?? new();  
                _objectParentId = objetParentId;  
            }  
        }  
    }  
}
```

```

        _maxCountViews = maxCountViews;
        _readOnly = readOnly;

        var tapGesture = new TapGestureRecognizer();
        tapGesture.Tapped += GestureTapped;
        _grid = BaseElemsCreator.CreateGrid();
        _grid.GestureRecognizers.Add(tapGesture);

        var rowIndex = 0;
        CreateUI(ref rowIndex);

        return _grid;
    }

    // Пусто
    protected virtual void CreateUI(ref int rowIndex)
    {

    }

    protected bool _moveTo = false;
    protected string _moveToName = "Перейти к ";
    protected virtual async void GestureTapped(object? sender, EventArgs e)
    {
        if (!_readOnly)
        {
            if (_baseResponse.Id > -1)
            {
                var action = string.Empty;
                if (_moveTo)
                {
                    action = await BaseElemsCreator.CreateActionSheet(
                        [
                            "Описание",
                            _moveToName,
                            "Редактирование",
                            "Удаление"]);
                }
                else
                {
                    action = await BaseElemsCreator.CreateActionSheet(
                        [
                            "Описание",
                            "Редактирование",
                            "Удаление"]);
                }

                switch (action)
                {
                    case "Описание":
                        ShowInfo(_baseResponse.Id);
                        break;
                    case "Редактирование":
                        Edit(_baseResponse.Id);
                        break;
                    case "Удаление":
                        Delete(_baseResponse.Id);
                        break;
                    default:

```

```

        if (action == _moveToName)
        {
            MoveTo(_baseResponse.Id);
        }
        return;
    }
}
else
{
    ShowInfo(_baseResponse.Id);
}
}

protected virtual void ShowInfo(long id)
{
    var baseViewObjectCreator = new TViewObjectCreator();
    var scrollView = baseViewObjectCreator.Create(_mainStack, _viewList,
ChagelistAction, _baseResponse, _objectParentId);
    AdminPageStatic.DeleteLastView(_mainStack, _viewList,
_maxCountViews);
    _viewList.Add(scrollView);
    AdminPageStatic.RepaintPage(_mainStack, _viewList);
}

protected virtual async void MoveTo(long id)
{
    string action = string.Empty;
    action = await BaseElemsCreator.CreateActionSheet(
        [
            "Администраторы",
            "Учителя",
            "Классы",
            "Ученики",
            "Родители",
            "Новости" ]);

    IBaseViewListCreator? viewCreator = null;
    switch (action)
    {
        case "Администраторы":
            viewCreator = new UserViewListCreator<UserResponse,
UserRequest, AdministratorController,
AdministratorController,
AdministratorController>>,
            UserViewObjectCreator<UserResponse, UserRequest,
AdministratorController>>,
            UserViewObjectCreator<UserResponse, UserRequest,
AdministratorController>>());
            break;
        case "Учителя":
            viewCreator = new UserViewListCreator<UserResponse,
UserRequest, TeacherController,
TeacherController,
TeacherController>>,
            UserViewObjectCreator<UserResponse, UserRequest,
TeacherController>>());
    }
}

```

```

        break;
        case "Классы":
            viewCreator = new ClassViewListCreator();
            break;
        case "Ученики":
            viewCreator = new UserViewListCreator<SchoolStudentResponse,
SchoolStudentRequest, SchoolStudentController,
            UserViewElemCreator<SchoolStudentResponse,
SchoolStudentRequest, SchoolStudentController,
            SchoolStudentViewObjectCreator>,
            SchoolStudentViewObjectCreator>();
            break;
        case "Родители":
            viewCreator = new UserViewListCreator<UserResponse,
ParentRequest, ParentController,
            UserViewElemCreator<UserResponse, ParentRequest,
ParentController,
            ParentViewObjectCreator>,
            ParentViewObjectCreator>();
            break;
        case "Новости":
            viewCreator = new NewsViewListCreator();
            break;
        default:
            return;
    }

    if (viewCreator != null)
    {
        AdminPageStatic.DeleteLastView(_mainStack, _viewList,
_maxCountViews);
        var scrollView = viewCreator.Create(_mainStack, _viewList,
_baseResponse.Id);

        _viewList.Add(scrollView);
        AdminPageStatic.RepaintPage(_mainStack, _viewList);
    }

    protected virtual void Edit(long id)
    {
        var baseViewObjectCreator = new TViewObjectCreator();
        var scrollView = baseViewObjectCreator.Create(_mainStack, _viewList,
ChageListAction, _baseResponse, _objectParentId, true);
        AdminPageStatic.DeleteLastView(_mainStack, _viewList,
_maxCountViews);
        _viewList.Add(scrollView);
        AdminPageStatic.RepaintPage(_mainStack, _viewList);
    }

    protected virtual async void Delete(long id)
    {
        var page = Application.Current?.Windows[0].Page;
        if (page != null)
        {
            var accept = await page.DisplayAlert("Подтверждение", "Удаление
объекта", "Да", "Нет");

            if (accept && _controller != null)
            {

```



```

        await _controller.Delete(id);
        Changelist();
    }
}

protected virtual void Changelist()
{
    ChageListAction.Invoke();
}
}

////////////////////////////////////
// FILE:
D:\WPS\ElectronicDiary\ElectronicDiary\Pages\AdminPageComponents\BaseView\BaseViewListC
reator.cs
////////////////////////////////////

using System.Text.Json;

using ElectronicDiary.Pages.AdminPageComponents.General;
using ElectronicDiary.Pages.Components.Elems;
using ElectronicDiary.Pages.Components.Other;
using ElectronicDiary.SaveData.Static;
using ElectronicDiary.Web.Api.Other;
using ElectronicDiary.Web.DTO.Requests.Educations.Other;
using ElectronicDiary.Web.DTO.Responses.Other;

namespace ElectronicDiary.Pages.AdminPageComponents.BaseView
{
    public interface IBaseViewListCreator
    {
        ScrollView Create(HorizontalStackLayout mainStack,
                        List<ScrollView> viewList,
                        long objetParentId = -1,
                        bool readOnly = false,
                        long objetPreParentId = -1);
    }

    public class BaseViewListCreator<TResponse, TRequest, TController,
TViewElemCreator, TViewObjectCreator> : IBaseViewListCreator
        where TResponse : BaseResponse, new()
        where TRequest : BaseRequest, new()
        where TController : IController, new()
        where TViewElemCreator : BaseViewElemCreator<TResponse, TRequest,
TController, TViewObjectCreator>, new()
        where TViewObjectCreator : BaseViewObjectCreator<TResponse, TRequest,
TController>, new()
    {
        protected TController _controller = new();

        protected HorizontalStackLayout _mainStack = [];
        protected List<ScrollView> _viewList = [];
        protected event Action ChageListAction;

        protected int _maxCountViews;
        protected VerticalStackLayout _listVerticalStack = new()
        {

```

```

        Spacing = UserData.Settings.Sizes.SPACING_ALL_PAGES
    };
    protected long _objectParentId;
    protected long _objectPreParentId;

    public BaseViewListCreator() : base()
    {
        _maxCountViews = 0;
        ChageListAction += ChageListHandler;
    }

    protected Grid _grid = [];
    protected bool _readOnly = false;
    public ScrollView Create(HorizontalStackLayout mainStack,
                            List<ScrollView> viewList,
                            long objetParentId = -1,
                            bool readOnly = false,
                            long objetPreParentId = -1)
    {
        _mainStack = mainStack;
        _viewList = viewList;
        _objectParentId = objetParentId;
        _readOnly = readOnly;
        _objectPreParentId = objetPreParentId;

        _ = CreateListUI();

        var vStack = new VerticalStackLayout
        {
            // Положение
            Padding = UserData.Settings.Sizes.PADDING_ALL_PAGES,
            Spacing = UserData.Settings.Sizes.SPACING_ALL_PAGES,
        };

        var scrollView = new ScrollView()
        {
            Content = vStack
        };

        _grid = BaseElemsCreator.CreateGrid();

        var rowIndex = 0;
        CreateFilterUI(ref rowIndex);
        vStack.Add(_grid);

        CreateGetButton(vStack);
        if (!_readOnly)
        {
            CreateAddButton(vStack);
        }
        CreateColumnTitle(vStack);

        vStack.Add(_listVerticalStack);

        return new ScrollView() { Content = vStack };
    }

    protected string _titleView = string.Empty;

    protected virtual void CreateTitile(ref int rowIndex)

```

```

    {
        LineElemsCreator.AddLineElems(
            grid: _grid,
            rowIndex: rowIndex++,
            objectList: [
                new LineElemsCreator.Data
                {
                    Elem = BaseElemsCreator.CreateTitleLabel(_titleView),
                    CountJoinColumns = 2
                },
            ]
        );
    }

    protected virtual void CreateFilterUI(ref int rowIndex)
    {
        CreateTitile(ref rowIndex);
    }

    protected virtual void CreateGetButton(VerticalStackLayout vStack)
    {
        var getButton = BaseElemsCreator.CreateButton("Найти",
GetButtonClicked);
        vStack.Add(getButton);
    }

    protected virtual void CreateAddButton(VerticalStackLayout vStack)
    {
        var addButton = BaseElemsCreator.CreateButton("Добавить",
AddButtonClicked);
        vStack.Add(addButton);
    }

    protected virtual void CreateColumnTitle(VerticalStackLayout vStack)
    {
    }

    protected virtual async void GetButtonClicked(object? sender, EventArgs
e)
    {
        await CreateListUI();
    }

    protected virtual void AddButtonClicked(object? sender, EventArgs e)
    {
        var viewObjectCreator = new TViewObjectCreator();
        var scrollView = viewObjectCreator.Create(_mainStack, _viewList,
ChageListAction, null, _objectParentId);
        AdminPageStatic.DeleteLastView(_mainStack, _viewList,
_maxCountViews);
        _viewList.Add(scrollView);
        AdminPageStatic.RepaintPage(_mainStack, _viewList);
    }

    // Получение списка объектов
    protected TResponse[] _objectsArr = [];
    protected virtual async Task CreateListUI()
    {

```

```

        await GetList();

        Application.Current?.Dispatcher.Dispatch(() =>
        {
            _listVerticalStack.Clear();

            for (var i = 0; i < _objectsArr.Length; i++)
            {
                var baseViewElemCreator = new TViewElemCreator();
                var grid = baseViewElemCreator.Create(_mainStack, _viewList,
ChageListAction, _objectsArr[i], _maxCountViews, _objectParentId, _readOnly);
                _listVerticalStack.Add(grid);
            }
        });
    }

    protected virtual void ChageListHandler()
    {
        _ = CreateListUI();
    }

    protected virtual async Task GetList()
    {
        var response = await _controller.GetAll(_objectParentId);
        if (!string.IsNullOrEmpty(response)) _objectsArr =
JsonSerializer.Deserialize<TResponse[]>(response, PageConstants.JsonSerializerOptions)
?? [];

        FilterList();
    }

    // Пусто
    protected virtual void FilterList()
    {
    }

    protected virtual void ChangeList()
    {
        ChageListAction.Invoke();
    }
}

```

```

////////////////////////////////////
// FILE:
D:\WPS\ElectronicDiary\ElectronicDiary\Pages\AdminPageComponents\BaseView\BaseViewObject
tCreator.cs
////////////////////////////////////

```

```

using ElectronicDiary.Pages.AdminPageComponents.General;
using ElectronicDiary.Pages.Components.Elems;
using ElectronicDiary.SaveData.Static;
using ElectronicDiary.Web.Api.Other;
using ElectronicDiary.Web.DTO.Requests.Educations.Other;
using ElectronicDiary.Web.DTO.Responses.Other;

namespace ElectronicDiary.Pages.AdminPageComponents.BaseView
{
    public class BaseViewObjectCreator<TResponse, TRequest, TController>

```

```

where TResponse : BaseResponse, new()
where TRequest : BaseRequest, new()
where TController : IController, new()
{
    protected TResponse _baseResponse = new();
    protected TRequest _baseRequest = new();
    protected TController _controller = new();

    protected HorizontalStackLayout _mainStack = [];
    protected List<ScrollView> _viewList = [];
    protected event Action ChageListAction = delegate { };

    protected long _objectParentId;

    protected AdminPageStatic.ComponentState _componentState;

    // Вид объекта
    protected VerticalStackLayout _infoStack = [];
    protected Grid _baseInfoGrid = [];
    protected int _baseInfoGridRowIndex = 0;
    protected VerticalStackLayout _vStack = [];

    public ScrollView Create(HorizontalStackLayout? mainStack,
                            List<ScrollView>? viewList,
                            Action? chageListAction,
                            BaseResponse? baseResponse,
                            long objetParentId,
                            bool edit = false)
    {
        _mainStack = mainStack ?? [];
        _viewList = viewList ?? [];
        ChageListAction = chageListAction ?? delegate { };
        _baseResponse = (baseResponse as TResponse) ?? new();
        _objectParentId = objetParentId;

        if (edit)
        {
            _componentState = AdminPageStatic.ComponentState.Edit;
        }
        else
        {
            if (_baseResponse.Id > -1)
            {
                _componentState = AdminPageStatic.ComponentState.Read;
            }
            else
            {
                _componentState = AdminPageStatic.ComponentState.New;
            }
        }

        _vStack = new VerticalStackLayout
        {
            // Положение
            Padding = UserData.Settings.Sizes.PADDING_ALL_PAGES,
            Spacing = UserData.Settings.Sizes.SPACING_ALL_PAGES,
        };
    }
}

```

```

        _infoStack = new VerticalStackLayout
        {
            // Положение
            Spacing = UserData.Settings.Sizes.SPACING_ALL_PAGES,
        };
        _vStack.Add(_infoStack);

        _baseInfoGrid = BaseElemsCreator.CreateGrid();
        _infoStack.Add(_baseInfoGrid);

        CreateUI();

        if (_componentState != AdminPageStatic.ComponentState.Read)
        {
            var saveButton = BaseElemsCreator.CreateButton(text: "Сохранить",
SaveButtonClicked);
            _vStack.Add(saveButton);
        }

        return new ScrollView() { Content = _vStack };
    }

    // Пусто
    protected virtual void CreateUI()
    {
        _baseInfoGridRowIndex = 0;

        if (_componentState == AdminPageStatic.ComponentState.Edit)
        {
            _baseRequest = new();
            _baseRequest.Id = _baseResponse.Id;
        }
    }

    protected virtual async void SaveButtonClicked(object? sender, EventArgs
e)
    {
        var response = _componentState == AdminPageStatic.ComponentState.New
?
            await _controller.Add(_baseRequest) :
            await _controller.Edit(_baseRequest);
        if (response != null)
        {
            ChageListAction.Invoke();
            AdminPageStatic.OnBackButtonPressed(_mainStack, _viewList);
        }
    }
}

```

```

////////////////////////////////////
// FILE:
D:\WPS\ElectronicDiary\ElectronicDiary\Pages\AdminPageComponents\ClassView\ClassViewEle
mCreator.cs
////////////////////////////////////

```

```

using ElectronicDiary.Pages.AdminPageComponents.BaseView;
using ElectronicDiary.Pages.AdminPageComponents.General;
using ElectronicDiary.Pages.AdminPageComponents.GroupView;

```

```

using ElectronicDiary.Pages.Components.Elems;
using ElectronicDiary.Web.Api.Educations;
using ElectronicDiary.Web.DTO.Requests.Educations;
using ElectronicDiary.Web.DTO.Responses.Educations;

namespace ElectronicDiary.Pages.AdminPageComponents.ClassView
{
    public class ClassViewElemCreator : BaseViewElemCreator<ClassResponse,
ClassRequest, ClassController, ClassViewObjectCreator>
    {
        public ClassViewElemCreator()
        {
            _moveTo = true;
            _moveToName = "Переход к группам";
        }

        protected override void CreateUI(ref int rowIndex)
        {
            base.CreateUI(ref rowIndex);

            LineElemsCreator.AddLineElems(
                grid: _grid,
                rowIndex: rowIndex++,
                objectList: [
                    new LineElemsCreator.Data
                    {
                        Elem = BaseElemsCreator.CreateLabel("Название")
                    },
                    new LineElemsCreator.Data
                    {
                        Elem = BaseElemsCreator.CreateLabel(_baseResponse.Name)
                    },
                ]
            );

            LineElemsCreator.AddLineElems(
                grid: _grid,
                rowIndex: rowIndex++,
                objectList: [
                    new LineElemsCreator.Data
                    {
                        Elem = BaseElemsCreator.CreateLabel("Классный
руководитель")
                    },
                    new LineElemsCreator.Data
                    {
                        Elem =
BaseElemsCreator.CreateLabel($"{_baseResponse.Teacher?.LastName}
{_baseResponse.Teacher?.FirstName} {_baseResponse.Teacher?.Patronymic}")
                    },
                ]
            );
        }

        protected override void MoveTo(long id)
        {
            var viewCreator = new GroupViewListCreator();

            AdminPageStatic.DeleteLastView(_mainStack, _viewList,
_maxCountViews);
        }
    }
}

```

```

        var scrollView = viewCreator.Create(_mainStack, _viewList,
_baseResponse.Id);

        _viewList.Add(scrollView);
        AdminPageStatic.RepaintPage(_mainStack, _viewList);
    }
}

// FILE:
D:\WPS\ElectronicDiary\ElectronicDiary\Pages\AdminPageComponents\ClassView\ClassViewListCreator.cs
using ElectronicDiary.Pages.AdminPageComponents.BaseView;
using ElectronicDiary.Pages.Components.Elems;
using ElectronicDiary.Web.Api.Educations;
using ElectronicDiary.Web.DTO.Requests.Educations;
using ElectronicDiary.Web.DTO.Responses.Educations;

namespace ElectronicDiary.Pages.AdminPageComponents.ClassView
{
    public class ClassViewListCreator : BaseViewListCreator<ClassResponse,
ClassRequest, ClassController, ClassViewElemCreator, ClassViewObjectCreator>
    {
        public ClassViewListCreator() : base()
        {
            _maxCountViews = 3;
            _titleView = "Список классов";
        }

        private string _nameFilter = string.Empty;

        protected override void CreateFilterUI(ref int rowIndex)
        {
            base.CreateFilterUI(ref rowIndex);

            LineElemsCreator.AddLineElems(
                grid: _grid,
                rowIndex: rowIndex++,
                objectList: [
                    new LineElemsCreator.Data
                    {
                        Elem = BaseElemsCreator.CreateLabel("Название")
                    },
                    new LineElemsCreator.Data
                    {
                        Elem = BaseElemsCreator.CreateEditor(newText =>
_nameFilter = newText, "7A")
                    },
                ]
            );
        }

        // Получение списка объектов
        protected override void FilterList()
        {
            bool nameFilter = string.IsNullOrEmpty(_nameFilter);

```



```

        _objectsArr = [... _objectsArr
            .Where(e =>
                (!nameFilter || (e.Name ??
string.Empty).Contains(_nameFilter!, StringComparison.OrdinalIgnoreCase)))]];
    }
}

////////////////////////////////////
// FILE:
D:\WPS\ElectronicDiary\ElectronicDiary\Pages\AdminPageComponents\ClassView\ClassViewObjectCreator.cs
////////////////////////////////////

using System.Text.Json;

using ElectronicDiary.Pages.AdminPageComponents.BaseView;
using ElectronicDiary.Pages.AdminPageComponents.General;
using ElectronicDiary.Pages.Components.Elems;
using ElectronicDiary.Pages.Components.Other;
using ElectronicDiary.Web.Api.Educations;
using ElectronicDiary.Web.Api.Users;
using ElectronicDiary.Web.DTO.Requests.Educations;
using ElectronicDiary.Web.DTO.Responses.Educations;
using ElectronicDiary.Web.DTO.Responses.Other;
using ElectronicDiary.Web.DTO.Responses.Users;

namespace ElectronicDiary.Pages.AdminPageComponents.ClassView
{
    public class ClassViewObjectCreator : BaseViewObjectCreator<ClassResponse,
ClassRequest, ClassController>
    {
        protected override void CreateUI()
        {
            base.CreateUI();

            if (_componentState == AdminPageStatic.ComponentState.Edit)
            {
                _baseRequest.Name = _baseResponse.Name;
                _baseRequest.TeacherId = _baseResponse.Teacher?.Id ?? -1;
            }

            LineElemsCreator.AddLineElems(
                grid: _baseInfoGrid,
                rowIndex: _baseInfoGridRowIndex++,
                objectList: [
                    new LineElemsCreator.Data
                    {
                        Elem = BaseElemsCreator.CreateLabel( "Название")
                    },
                    _componentState == AdminPageStatic.ComponentState.Read ?
                    new LineElemsCreator.Data
                    {
                        Elem = BaseElemsCreator.CreateLabel(
_baseResponse.Name)
                    }
                ]
                :
                new LineElemsCreator.Data

```

```

        {
            Elem = BaseElemsCreator.CreateEditor( newText =>
_baseRequest.Name = newText, "7A", _baseResponse.Name )
        }
    ]
);

LineElemsCreator.AddLineElems(
    grid: _baseInfoGrid,
    rowIndex: _baseInfoGridRowIndex++,
    objectList: [
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateLabel("Классный
руководитель")
        },
        _componentState == AdminPageStatic.ComponentState.Read ?
        new LineElemsCreator.Data
        {
            Elem =
BaseElemsCreator.CreateLabel($"{_baseResponse.Teacher?.LastName}
{_baseResponse.Teacher?.FirstName} {_baseResponse.Teacher?.Patronymic}")
        }
        :
        new LineElemsCreator.Data
        {
            Elem =
BaseElemsCreator.CreateSearchPopupAsLabel(GetTeachers(), selectedIndex =>
_baseRequest.TeacherId = selectedIndex)
        }
    ]
);

private List<TypeResponse> GetTeachers()
{
    var list = new List<TypeResponse>();

    Task.Run(async () =>
    {
        UserResponse[]? arr = null;
        var controller = new TeacherController();
        var response = await controller.GetAll(_objectParentId);
        if (!string.IsNullOrEmpty(response)) arr =
JsonSerializer.Deserialize<UserResponse[]>(response,
PageConstants.JsonSerializerOptions) ?? [];

        foreach (var elem in arr ?? [])
        {
            list.Add(new TypeResponse(elem.Id, $"{elem?.LastName}
{elem?.FirstName} {elem?.Patronymic}"));
        }
    });

    return list;
}
}
}

```

```

////////////////////////////////////
// FILE:
D:\WPS\ElectronicDiary\ElectronicDiary\Pages\AdminPageComponents\EducationalInstitution
View\EducationalInstitutionViewElemCreator.cs
////////////////////////////////////

using ElectronicDiary.Pages.AdminPageComponents.BaseView;
using ElectronicDiary.Pages.Components.Elems;
using ElectronicDiary.Web.Api.Educations;
using ElectronicDiary.Web.DTO.Requests.Educations;
using ElectronicDiary.Web.DTO.Responses.Educations;

namespace ElectronicDiary.Pages.AdminPageComponents.EducationalInstitutionView
{
    public class EducationalInstitutionViewElemCreator :
BaseViewElemCreator<EducationalInstitutionResponse, EducationalInstitutionRequest,
EducationalInstitutionController, EducationalInstitutionViewObjectCreator>
    {
        public EducationalInstitutionViewElemCreator()
        {
            _moveTo = true;
        }

        protected override void CreateUI(ref int rowIndex)
        {
            base.CreateUI(ref rowIndex);

            LineElemsCreator.AddLineElems(
                grid: _grid,
                rowIndex: rowIndex++,
                objectList: [
                    new LineElemsCreator.Data
                    {
                        Elem = BaseElemsCreator.CreateLabel("Название")
                    },
                    new LineElemsCreator.Data
                    {
                        Elem = BaseElemsCreator.CreateLabel(_baseResponse.Name)
                    },
                ]
            );

            LineElemsCreator.AddLineElems(
                grid: _grid,
                rowIndex: rowIndex++,
                objectList: [
                    new LineElemsCreator.Data
                    {
                        Elem = BaseElemsCreator.CreateLabel("Регион")
                    },
                    new LineElemsCreator.Data
                    {
                        Elem =
BaseElemsCreator.CreateLabel(_baseResponse.Settlement?.Region?.Name)
                    },
                ]
            );

            LineElemsCreator.AddLineElems(
                grid: _grid,

```

```

        rowIndex: rowIndex++,
        objectList: [
            new LineElemsCreator.Data
            {
                Elem = BaseElemsCreator.CreateLabel("Населённый пункт")
            },
            new LineElemsCreator.Data
            {
                Elem =
BaseElemsCreator.CreateLabel(_baseResponse.Settlement?.Name)
            },
        ]
    };
}
}
}

////////////////////////////////////
// FILE:
D:\WPS\ElectronicDiary\ElectronicDiary\Pages\AdminPageComponents\EducationalInstitution
View\EducationalInstitutionViewListCreator.cs
////////////////////////////////////

using ElectronicDiary.Pages.AdminPageComponents.BaseView;
using ElectronicDiary.Pages.Components.Elems;
using ElectronicDiary.Web.Api.Educations;
using ElectronicDiary.Web.DTO.Requests.Educations;
using ElectronicDiary.Web.DTO.Responses.Educations;

namespace ElectronicDiary.Pages.AdminPageComponents.EducationalInstitutionView
{
    public sealed class EducationalInstitutionViewListCreator :
BaseViewListCreator<EducationalInstitutionResponse, EducationalInstitutionRequest,
EducationalInstitutionController, EducationalInstitutionViewElemCreator,
EducationalInstitutionViewObjectCreator>
    {
        public EducationalInstitutionViewListCreator() : base()
        {
            _maxCountViews = 2;
            _titleView = "Список учебных заведений";
        }

        private string _regionFilter = string.Empty;
        private string _settlementFilter = string.Empty;
        private string _nameFilter = string.Empty;

        protected override void CreateFilterUI(ref int rowIndex)
        {
            base.CreateFilterUI(ref rowIndex);

            LineElemsCreator.AddLineElems(
                grid: _grid,
                rowIndex: rowIndex++,
                objectList: [
                    new LineElemsCreator.Data
                    {
                        Elem = BaseElemsCreator.CreateLabel("Область")
                    },
                    new LineElemsCreator.Data

```

```

        {
            Elem = BaseElemsCreator.CreateEditor(newText =>
            _regionFilter = newText, "Минская область")
        },
    ]
);

LineElemsCreator.AddLineElems(
    grid: _grid,
    rowIndex: rowIndex++,
    objectList: [
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateLabel("Населённый пункт")
        },
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateEditor(newText =>
            _settlementFilter = newText, "г.Солигорск")
        },
    ]
);

LineElemsCreator.AddLineElems(
    grid: _grid,
    rowIndex: rowIndex++,
    objectList: [
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateLabel("Название"),
        },
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateEditor(newText =>
            _nameFilter = newText, "ГУО ...")
        },
    ]
);
}

// Получение списка объектов
protected override void FilterList()
{
    if (_objectParentId == -1)
    {
        bool regionFilter = string.IsNullOrEmpty(_regionFilter);
        bool settlementFilter = string.IsNullOrEmpty(_settlementFilter);
        bool nameFilter = string.IsNullOrEmpty(_nameFilter);

        _objectsArr = [.. _objectsArr
            .Where(e =>
                (!regionFilter || (e.Settlement?.Region?.Name ??
                string.Empty).Contains(_regionFilter!, StringComparison.OrdinalIgnoreCase)) &&
                (!settlementFilter || (e.Settlement?.Name ??
                string.Empty).Contains(_settlementFilter!, StringComparison.OrdinalIgnoreCase)) &&
                (!nameFilter || (e.Name ??
                string.Empty).Contains(_nameFilter!, StringComparison.OrdinalIgnoreCase)))]];
    }
    else
    {

```

```

        _objectsArr = [.. _objectsArr.Where(e => e.Id ==
_objectParentId)];
    }

    }

}

////////////////////////////////////
// FILE:
D:\WPS\ElectronicDiary\ElectronicDiary\Pages\AdminPageComponents\EducationalInstitution
View\EducationalInstitutionViewObjectCreator.cs
////////////////////////////////////

using System.Text.Json;

using ElectronicDiary.Pages.AdminPageComponents.BaseView;
using ElectronicDiary.Pages.AdminPageComponents.General;
using ElectronicDiary.Pages.Components.Elems;
using ElectronicDiary.Pages.Components.Other;
using ElectronicDiary.Web.Api.Educations;
using ElectronicDiary.Web.DTO.Requests.Educations;
using ElectronicDiary.Web.DTO.Responses.Educations;
using ElectronicDiary.Web.DTO.Responses.Other;

namespace ElectronicDiary.Pages.AdminPageComponents.EducationalInstitutionView
{
    public class EducationalInstitutionViewObjectCreator :
BaseViewObjectCreator<EducationalInstitutionResponse, EducationalInstitutionRequest,
EducationalInstitutionController>
    {
        private VerticalStackLayout? _imageStack = null;
        protected override void CreateUI()
        {
            base.CreateUI();

            if (_componentState == AdminPageStatic.ComponentState.Edit)
            {
                _baseRequest.Name = _baseResponse.Name;
                _baseRequest.Address = _baseResponse.Address;
                _baseRequest.Email = _baseResponse.Email;
                _baseRequest.PhoneNumber = _baseResponse.PhoneNumber;
                _baseRequest.RegionId = _baseResponse.Settlement?.Region?.Id ?? -
1;
                _baseRequest.SettlementId = _baseResponse.Settlement?.Id ?? -1;
            }

            _imageStack =
BaseElemsCreator.CreateImageFromUrl(_baseResponse.PathImage,
_componentState == AdminPageStatic.ComponentState.Edit ?
AddImageTapped : null);
            LineElemsCreator.AddLineElems(
                grid: _baseInfoGrid,
                rowIndex: _baseInfoGridRowIndex++,
                objectList: [
                    new LineElemsCreator.Data
                    {
                        CountJoinColumns = 2,
                        Elem = _imageStack
                    }
                ]
            );
        }
    }
}

```

```

    }
    });

    LineElemsCreator.AddLineElems(
        grid: _baseInfoGrid,
        rowIndex: _baseInfoGridRowIndex++,
        objectList: [
            new LineElemsCreator.Data
            {
                Elem = BaseElemsCreator.CreateLabel( "Название")
            },
            _componentState == AdminPageStatic.ComponentState.Read ?
            new LineElemsCreator.Data
            {
                Elem = BaseElemsCreator.CreateLabel(
_baseResponse.Name)
            }
            :
            new LineElemsCreator.Data
            {
                Elem = BaseElemsCreator.CreateEditor( newText =>
_baseRequest.Name = newText, "ГЮ0 ...",_baseResponse.Name )
            }
        ]
    );

    List<TypeResponse> settlementList = [];
    LineElemsCreator.AddLineElems(
        grid: _baseInfoGrid,
        rowIndex: _baseInfoGridRowIndex++,
        objectList: [
            new LineElemsCreator.Data
            {
                Elem = BaseElemsCreator.CreateLabel("Регион")
            },
            _componentState == AdminPageStatic.ComponentState.Read ?
            new LineElemsCreator.Data
            {
                Elem =
BaseElemsCreator.CreateLabel(_baseResponse.Settlement?.Region?.Name)
            }
            :
            new LineElemsCreator.Data
            {
                Elem =
BaseElemsCreator.CreateSearchPopupAsLabel(GetRegions(),
selectedIndex =>
            {
                _baseRequest.RegionId = selectedIndex;
                if(_baseRequest.RegionId > -1)
                {
                    GetSettlements(settlementList,
_baseRequest.RegionId);
                }
            }
            )
        ]
    );

```

```

LineElemsCreator.AddLineElems(
    grid: _baseInfoGrid,
    rowIndex: _baseInfoGridRowIndex++,
    objectList: [
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateLabel("Населённый пункт")
        },
        _componentState == AdminPageStatic.ComponentState.Read ?
        new LineElemsCreator.Data
        {
            Elem =
BaseElemsCreator.CreateLabel(_baseResponse.Settlement ?.Name)
        }
        :
        new LineElemsCreator.Data
        {
            Elem =
BaseElemsCreator.CreateSearchPopupAsLabel(settlementList, selectedIndex =>
_baseRequest.SettlementId = selectedIndex)
        }
    ]
);

LineElemsCreator.AddLineElems(
    grid: _baseInfoGrid,
    rowIndex: _baseInfoGridRowIndex++,
    objectList: [
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateLabel("Адресс")
        },
        _componentState == AdminPageStatic.ComponentState.Read ?
        new LineElemsCreator.Data
        {
            Elem =
BaseElemsCreator.CreateLabel(_baseResponse.Address)
        }
        :
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateEditor(newText =>
_baseRequest.Address = newText, "ул. Ленина, 12", _baseResponse.Address )
        }
    ]
);

LineElemsCreator.AddLineElems(
    grid: _baseInfoGrid,
    rowIndex: _baseInfoGridRowIndex++,
    objectList: [
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateLabel("Email")
        },
        _componentState == AdminPageStatic.ComponentState.Read ?
        new LineElemsCreator.Data
        {
            Elem =
BaseElemsCreator.CreateLabel(_baseResponse.Email)

```



```

        }
        :
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateEditor( newText =>
_baseRequest.Email = newText, "sh4@edus.by", _baseResponse.Email)
        }
    ]
);

LineElemsCreator.AddLineElems(
    grid: _baseInfoGrid,
    rowIndex: _baseInfoGridRowIndex++,
    objectList: [
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateLabel("Телефон")
        },
        _componentState == AdminPageStatic.ComponentState.Read ?
        new LineElemsCreator.Data
        {
            Elem =
BaseElemsCreator.CreateLabel(_baseResponse.PhoneNumber)
        }
        :
        new LineElemsCreator.Data
        {
            Elem = BaseElemsCreator.CreateEditor(newText =>
_baseRequest.PhoneNumber = newText, "+375 17 433-09-02", _baseResponse.PhoneNumber)
        }
    ]
);
}

private Stream? _image_stream = null;
private FileResult? _imageFile = null;
private async void AddImageTapped(object? sender, EventArgs e)
{
    var result = await FilePicker.Default.PickAsync(new PickOptions
    {
        PickerTitle = "Выберите изображение",
        FileTypes = FilePickerFileType.Images
    });

    if (result == null) return;
    _imageFile = result;

    if (_imageStack != null)
    {
        _image_stream?.Dispose();
        _image_stream = await result.OpenReadAsync();
        Image image = (Image)_imageStack[^1];
        image.Source = ImageSource.FromStream(() => _image_stream);
    }
}

protected override void SaveButtonClicked(object? sender, EventArgs e)
{
    base.SaveButtonClicked(sender, e);
}

```

```

        if (_imageFile != null)
        {
            _controller.AddImage(_baseResponse.Id, _imageFile);
        }
    }

    private static List<TypeResponse> GetRegions()
    {
        var list = new List<TypeResponse>();

        Task.Run(async () =>
        {
            Web.DTO.Responses.Other.TypeResponse[]? arr = null;
            var response = await AddressController.GetRegions();
            if (!string.IsNullOrEmpty(response)) arr =
JsonSerializer.Deserialize<Web.DTO.Responses.Other.TypeResponse[]>(response,
PageConstants.JsonSerializerOptions) ?? [];

            foreach (var elem in arr ?? [])
            {
                list.Add(new TypeResponse(elem.Id, elem.Name));
            }
        });

        return list;
    }

    private static void GetSettlements(List<TypeResponse> list, long
regionId)
    {
        list.Clear();
        Task.Run(async () =>
        {
            Web.DTO.Responses.Other.TypeResponse[]? arr = null;
            var response = await AddressController.GetSettlements(regionId);
            if (!string.IsNullOrEmpty(response)) arr =
JsonSerializer.Deserialize<Web.DTO.Responses.Other.TypeResponse[]>(response,
PageConstants.JsonSerializerOptions) ?? [];

            foreach (var elem in arr ?? [])
            {
                list.Add(new TypeResponse(elem.Id, elem.Name));
            }
        });
    }
}

```

```

////////////////////////////////////
// FILE:
D:\WPS\ElectronicDiary\ElectronicDiary\Pages\AdminPageComponents\General\AdminPage.cs
////////////////////////////////////

```

```

using ElectronicDiary.Pages.AdminPageComponents.EducationalInstitutionView;
using ElectronicDiary.Pages.Components.Elems;
using ElectronicDiary.Pages.Components.Navigation;
using ElectronicDiary.SaveData.Static;

```

```

namespace ElectronicDiary.Pages.AdminPageComponents.General

```

```

    {
        public partial class AdminPage : ContentPage
        {
            private readonly HorizontalStackLayout _mainStack =
BaseElementsCreator.CreateHorizontalStackLayout();
            private readonly List<ScrollView> _viewList = [];

            public AdminPage()
            {
                Title = "Панель администратора";
                ToolbarItemsAdder.AddSettings(ToolbarItems);
                ToolbarItemsAdder.AddLogOut(ToolbarItems);
                BackgroundColor = UserData.Settings.Theme.BackgroundColor;

                var view = new EducationalInstitutionViewListCreator();

                var scrollView = view.Create(_mainStack, _viewList);

                _viewList.Add(scrollView);

                AdminPageStatic.RepaintPage(_mainStack, _viewList);
                Content = _mainStack;
                SizeChanged += WindowSizeChanged;
            }

            private void WindowSizeChanged(object? sender, EventArgs e)
            {
                AdminPageStatic.RepaintPage(_mainStack, _viewList);
            }

            // Переопределение возврата
            protected override bool OnBackButtonPressed()
            {
                return AdminPageStatic.OnBackButtonPressed(_mainStack, _viewList);
            }
        }
    }

////////////////////////////////////
// FILE:
D:\WPS\ElectronicDiary\ElectronicDiary\Pages\AdminPageComponents\General\AdminPageStati
c.cs
////////////////////////////////////

namespace ElectronicDiary.Pages.AdminPageComponents.General
{
    public static class AdminPageStatic
    {
        // Обновление видов
        public static void RepaintPage(HorizontalStackLayout mainStack,
List<ScrollView> viewList)
        {
            Application.Current?.Dispatcher.Dispatch(() =>
            {
                mainStack.Clear();

                CalcViewWidth(out var width, out var countColumn);
            });
        }
    }
}

```

```

        for (var i = int.Max(viewList.Count - countColumn, 0); i <
viewList.Count; i++)
        {
            viewList[i].MinimumWidthRequest = width;
            viewList[i].MaximumWidthRequest = width;
            mainStack.Add(viewList[i]);
        }
    });
}

public static void CalcViewWidth(out double width, out int countColumn)
{
    double dpi = DeviceDisplay.MainDisplayInfo.Density * 640;
    var widthWindow = 0d;
    if (Application.Current?.Windows.Count > 0)
    {
        widthWindow = Application.Current?.Windows[0].Width ?? 0d;
    }

    #if WINDOWS
        const double coeffFixAndroidWidth = 1;
    #else
        const double coeffFixAndroidWidth = 3.3;
    #endif

    countColumn = int.Max((int)(widthWindow * coeffFixAndroidWidth /
dpi), 1);

    width = 0.9 * dpi / (coeffFixAndroidWidth);
}

public static bool OnBackButtonPressed(HorizontalStackLayout mainStack,
List<ScrollView> viewList)
{
    if (viewList.Count > 1)
    {
        viewList.RemoveAt(viewList.Count - 1);
        RepaintPage(mainStack, viewList);
    }

    return true;
}

public static void DeleteLastView(HorizontalStackLayout mainStack,
List<ScrollView> viewList, int maxCountViews, int indexDel = 1)
{
    while (viewList.Count >= maxCountViews)
    {
        viewList.RemoveAt(viewList.Count - indexDel);
    }
}

public enum ComponentState
{
    Read, New, Edit
}

}
}

```

Обозначение					Наименование		Дополнительные сведения		
					Текстовые документы				
БГУИР ДП 1–40 01 01 01 036 ПЗ					Пояснительная записка		101 с.		
					Отзыв руководителя				
					Рецензия				
					Акт о внедрении				
					Графические документы				
ГУИР.151004-01 СА					Схема алгоритма		Формат А1		
					Главный алгоритм программного средства				
ГУИР.151004-02 СА					Схема алгоритма				
					Выбор элемента из списка		Формат А1		
ГУИР.151004-03 СА					Схема алгоритма				
					Общения с сервером		Формат А1		
ГУИР.151004-01 ПЛ					Плакат				
					Use case диаграмма		Формат А1		
ГУИР.151004-02 ПЛ					Плакат				
					Физическая модель базы данных		Формат А1		
ГУИР.151004-03 ПЛ					Плакат				
					Deployment диаграмма		Формат А1		
ГУИР.151004-04 ПЛ					Плакат		Формат А1		
					Диаграмма классов наследования				
					интерфейса «Список учеников»				