Изм.

Лист

№ докум.

Подпись

Дата

Лист

У 3

ДП Т.995088.401 ПЗ

Разраб.

*Мазго М.А.*

Провер.

Басалыга Л.В.

Т. Контр.

*Багласова Т.Г.*

Н. Контр.

Басалыга Л.В.

Утверд.

*Багласова Т.Г.*

Создание программы для организации процесса дистанционного обучения учащихся

Лит.

Листов

74

КБП

СОДЕРЖАНИЕ

[Введение 4](#_Toc137716025)

[1 Описание задачи 6](#_Toc137716026)

[1.1 Анализ предметной области 6](#_Toc137716027)

[1.2 Постановка задачи 6](#_Toc137716028)

[2 Проектирование системы 8](#_Toc137716029)

[2.1 Требования к приложению 8](#_Toc137716030)

[2.2 Проектирование модели 8](#_Toc137716031)

[2.3 Проектирование структуры базы данных 10](#_Toc137716032)

[2.4 Концептуальный прототип 12](#_Toc137716033)

[3 Описание реализации программного средства 18](#_Toc137716034)

[3.1 Инструменты разработки и применяемые технологии 18](#_Toc137716035)

[3.2 Порядок авторизации пользователей 19](#_Toc137716036)

[3.3 Организация данных 19](#_Toc137716037)

[3.4 Функции: логическая и физическая организация 26](#_Toc137716038)

[3.5 Входные и выходные данные 29](#_Toc137716039)

[3.6 Функциональное тестирование 30](#_Toc137716040)

[3.7 Описание справочной системы 32](#_Toc137716041)

[4 Применение 36](#_Toc137716042)

[4.1 Назначение программного средства 36](#_Toc137716043)

[4.2 Условия применения 36](#_Toc137716044)

[5 Охрана труда и охрана окружающей среды 37](#_Toc137716045)

[5.1 Правовые, нормативные, социально-экономические и организационные вопросы охраны труда. 37](#_Toc137716046)

[5.2 Разработка мер по нормализации и защите от электромагнитного излучения при эксплуатации ПЭВМ 39](#_Toc137716047)

[5.3 Пожарная безопасность 43](#_Toc137716048)

[5.4 Охрана окружающей среды 46](#_Toc137716049)

[6 Экономический раздел 48](#_Toc137716050)

[6.1 Технико-экономическое обоснование разработки программного средства 48](#_Toc137716051)

[6.2 Составление плана по разработке программного средства 48](#_Toc137716052)

[6.3 Определение цены программного средства 49](#_Toc137716053)

[6.4 Экономическая эффективность разработки программного продукта 54](#_Toc137716054)

[6.5 Экономическая эффективность у пользователя программного продукта 55](#_Toc137716055)

[Заключение 57](#_Toc137716056)

[Список информационных источников 58](#_Toc137716057)

[ПРИЛОЖЕНИЕ А 60](#_Toc137716058)

[ПРИЛОЖЕНИЕ Б 73](#_Toc137716059)

# Введение

В настоящее время автоматизация процессов имеет большое значение для современного мира. Она помогает снизить затраты, увеличить эффективность и производительность, а также снизить риски потери данных, уменьшить возможное влияние человеческого фактора. Автоматизация упрощает не только повседневную работу, а также и профессиональную деятельность. Ключевой особенностью возможности автоматизации является то, что с каждым днем это становиться актуально все больше и больше.

Целью дипломного проекта на тему «Создание программы для организации процесса дистанционного обучения учащихся» является разработка программного средства «UnilifeClassesRooms» состоящее из 2 частей «UnilifeClassesRoomsDiplomDesktop» и «UnilifeClassesRoomsDiplomServer». В программном средстве будет организовано ведение базы данных пользователей, должностей, подразделений аккаунтов, заданий, файлов, работ, ссылок, ролей. Будет организовано отображение описанных ранее объектов, а также поиск, фильтрация и сортировка объектов по различным категориям в зависимости от специфики пользователей, реализован учет работ учащихся, оценивание работ, хранение файлов.

Пояснительная записка к дипломному проекту состоит из шести разделов, которые содержат решение поставленных задач, а также необходимую информацию по использованию программного средства «UnilifeClassesRooms».

В первом разделе «Описание задачи» описывается предметная область и круг задач, которые должны быть решены. Приводится характеристика периодичности использования программы. Описываются существующие аналоги.

Во втором разделе «Проектирование системы» перечисляются требования к приложению. Проводится моделирование всей программы и структуры базы данных. Предоставляется модель внешнего пользовательского интерфейса.

В третьем разделе «Описание реализации программного средства» описываются инструменты разработки, а также используемые технологии. Описывается инфраструктура базы данных, логическая и физическая организация данных реализованных функций. Описывается испытание разработанного программного средства с описанием организации входных и выходных данных. Так же описывается раздел справочной системы.

В четвертом разделе «Применение» описываются сведенья о назначении программного средства и области его применения.

В пятом разделе «Охрана труда и окружающей среды» описываются вопросы охраны труда и охраны окружающей среды согласно действующему законодательству.

В шестом разделе «Экономический раздел» содержит составление плана разработки программного средства, расчет цены и экономическую эффективность разработки программного продукта.

В заключении будет проанализировано созданное программное средство, определена степень соответствия поставленной задачи и выполненной работы. И описаны причины несоответствия, если таковые имеются.

ПРИЛОЖЕНИЕ А будет содержать текст программы.

ПРИЛОЖЕНИЕ Б будет содержать изображения с результатами тестирования программного средства и изображения справки.

В графической части будут представлены диаграммы деятельности, последовательности, вариантов использования, компонентов.

# 1 Описание задачи

## 1.1 Анализ предметной области

Тема дипломного проекта «Создание программы для организации процесса дистанционного обучения учащихся». Предметной областью является учебный класс «Средняя школа №133».

В качестве инструментов разработки выступят WPF (система для построения клиентских приложений Windows) с предоставляемым ею инструментарием, а так же WCF(платформа для создания приложений, ориентированных на службы). Данные инструменты необходимы для создания клиентно-серверного приложения. Где в качестве клиента выступает десктопное приложение, сделанное по технологии WPF, а серверная часть с применением служб WCF.

В программе должно быть реализовано:

* разграничение прав доступа;
* создание и сопровождение классов администратором;
* сопровождение пользователей администратором;
* выставление отметок за выполненные задания;
* автоматическая рассылка сообщений на электронную почту о новых заданиях в классах и отметках за задания;
* формирование данных по успеваемости учащихся в классе;
* добавление файлов работ и отправки работы на проверку;
* поиск и фильтрация данных по различным критериям;
* наполнение классов заданиями, преподавателем.

Программа будет нуждаться в интернете по причине удалённости сервера от пользователя.

Сервер – это компьютер, подключенный по локальной или внешней сети, к которому периодически обращаются другие компьютеры, именуемые клиентами.

Серверное программное обеспечение (Серверное ПО) – специальные приложения, предназначенные для предоставления определенных услуг и обеспечивающие их выполнение.

Клиент – это аппаратный или программный компонент вычислительной системы, посылающий запросы серверу.

Потенциальная аудитория данного приложения — это учащиеся /учителя учреждений образования, которым необходимо сдавать/принимать работы удалённо.

## 1.2 Постановка задачи

Исходя из анализа предметной области можно выделить следующие задачи подлежащие автоматизации:

* ведение базы данных пользователей, классов, работ;
* разграничение прав доступа;
* выставление отметок за работы учащихся;
* оповещать учащихся о выставление отметки им по работе;
* оповещать учащихся о добавленном задания в классе;
* учет сданных работ;
* возможность перехода по прикреплённым ссылкам к заданиям;
* просмотр файлов заданий работ;
* поиск, фильтрация и сортировка данных по различным критериям;
* формирование отчёт по аккаунтам приложения и закреплённых за ними людей;
* создание справочной системы приложения.

Аналогом разрабатываемого программного средства является сайт для удалённого обучения «Google Сlassroom». Он предоставляет возможности такие как добавление курсов, наполнения их, учёт работ учащихся.

Разрабатываемая программа упростит сдачу работ учащихся в учебных заведениях, преподаватель сможет не только наполнять курсы, а также отслеживать количество сданных работ, проводить поиск и фильтрацию тех или иных работ, заданий, курсов используя при этом одно программное средство.

# 2 Проектирование системы

## 2.1 Требования к приложению

[Требования к интерфейсу программного средства](#_heading=h.3as4poj):

* дизайн приложения должен быть лаконичным и в то же время выглядеть стильно, современно;
* предпочтительные цвета приложения: тёмные, для комфортной работы ночью, а также снижению энергопотребления матриц.

[Требования к шрифтовому оформлению:](#_heading=h.1pxezwc)

* [размер (кегль) шрифтов должен обеспечивать удобство восприятия текста.](#_heading=h.1pxezwc)
* [предпочтителен шрифт без засечек.](#_heading=h.1pxezwc)

Система управления контентом приложения должна обеспечить пользователю приложения возможность выполнения следующих действий:

* реализовать наглядное отображение всех объектов базы данных;
* осуществить возможность добавления записей в объекты базы данных;
* реализовать просмотр записей;
* организовать сохранение добавленных записей в базу данных;
* обеспечить поиск, фильтрацию и сортировку объектов по различным критериям;
* реализовать ограничения на таблицы в которых видеться работа с числовыми полями, вводимые значения должны быть больше нуля.

Требования к аппаратным и операционным ресурсам:

* среда разработки Visual Studio 2022;
* клавиатура и мышь;
* процессор Pentium E5300;
* оперативная память 4,00 ГБ;
* видеокарта NVIDIA GeForce GT 240;
* пакет программ Microsoft Office;
* ядро реляционной база данных MSSQL.

## 2.2 Проектирование модели

Диаграмма вариантов использования описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Она является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы.

В данной проектируемой системе в качестве актера выступает пользователь (учитель), который служит источником воздействия на моделируемую систему.

К основным функциям разрабатываемой программы относятся следующие функции:

* добавление заданий в классы;
* проверка работ учащихся и выставление отметок;
* настройка аккаунта;
* просмотр справочной системы;
* формирование отчета.

Функция добавление заданий для учащимся содержит в себе вспомогательную функцию добавление файлов и ссылок.

Диаграмма вариантов использования для учителя представлена на листе 1 графической части.

**Диаграмма последовательности.** Эта визуально привлекательная диаграмма, популярная не только в сообществе разработчиков, хорошо показывает все типы бизнес-процессов. Она просто раскрывает структуру системы, показывая последовательность сообщений и взаимодействий между операторами и объектами в хронологическом порядке. Диаграммы последовательности отображают простую итерацию и ветвление. Это имеет преимущества для многозадачности. Диаграмма последовательности для проверки работы и выставления отметки преподавателем, представлена на листе 2 графической части.

Диаграммы деятельности – частный случай диаграмм состояний. Основная цель использования таких диаграмм – визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности для администратора приложения, представлена на листе 3 графической части.

**Диаграмма компонентов.** Также известна как блок-схема компонентов, она показывает логические группы элементов и их взаимосвязи. Другими словами, она дает упрощенное представление о сложной системе, разбивая ее на более мелкие компоненты. Каждый из элементов показан в прямоугольной рамке с названием, написанным внутри. Соединители определяют отношения/ зависимости между различными компонентами. Диаграмма компонентов представлена на листе 4 графической части.

## 2.3 Проектирование структуры базы данных

Цель моделирования данных состоит в обеспечении разработчика информационной системы концептуальной схемой базы данных в форме одной или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему без данных.

Наиболее распространенным средством моделирования данных является диаграмма «Сущность-связь» (ERD). С их помощью определяется важные для предметной области объекты (сущности), их свойства (атрибуты) и отношения друг с другом (связи). ERD непосредственно используются для проектирования реляционных баз данных.

Диаграмма «Сущность-связь» представлена на рисунке 2.1.

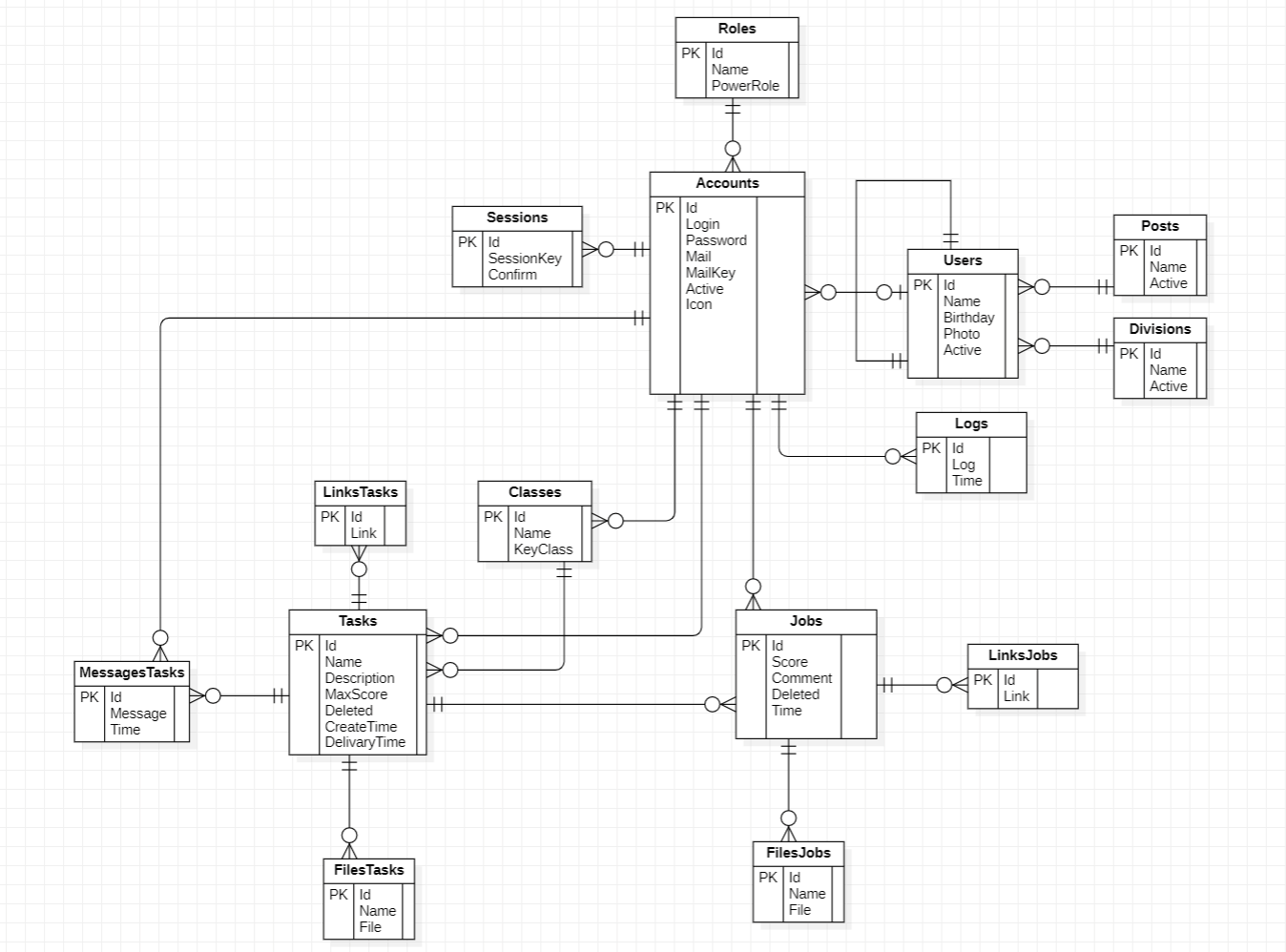


Рисунок 2.1

Исходя из анализа предметной области можно выделить следующие сущности разработки: «Roles», «Classes», «MessagesTasks», «FilesTasks», «Tasks», «LinksTasks», «Users», «Accounts», «LinksJobs», «Jobs», «Sessions», «FilesJobs», «Logs», «Posts», «Divisions».

Для сущности «Roles» атрибутами будут являться:

* name;
* powerRole.

Для сущности «Classes» атрибутами будут являться:

* name;
* keyClass.

Для сущности «MessagesTasks» атрибутами будут являться:

* message;
* time.

Для сущности «FilesTasks» атрибутами будут являться:

* name;
* file.

Для сущности «Tasks» атрибутами будут являться:

* name;
* description;
* maxScore;
* deleted;
* createTime;
* delivaryTime.

Для «LinksTasks» атрибутами будут являться:

* link.

Для сущности «Users» атрибутами будут являться:

* name;
* birthday;
* photo;
* active.
* код.

Для сущности «Accounts» атрибутами будут являться:

* login;
* password;
* nail;
* mailKey;
* active;
* icon.

Для сущности «LinksJobs» атрибутами будут являться:

* link.

Для сущности «Jobs» атрибутами будут являться:

* score;
* comment;
* deleted;
* time.

Для сущности «Sessions» атрибутами будут являться:

* sessionKey;
* confirm.

Для сущности «FilesJobs» атрибутами будут являться:

* name;
* file.

Для сущности «Logs» атрибутами будут являться:

* log;
* time.

Для сущности «Posts» атрибутами будут являться:

* name;
* active.

Для сущности «Divisions» атрибутами будут являться:

* name;
* active.

## 2.4 Концептуальный прототип

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – систему кнопок и форм.

При открытии программы мы видим форму, представленную на рисунке 2.2. Данная форма содержит поля необходимые для входа.

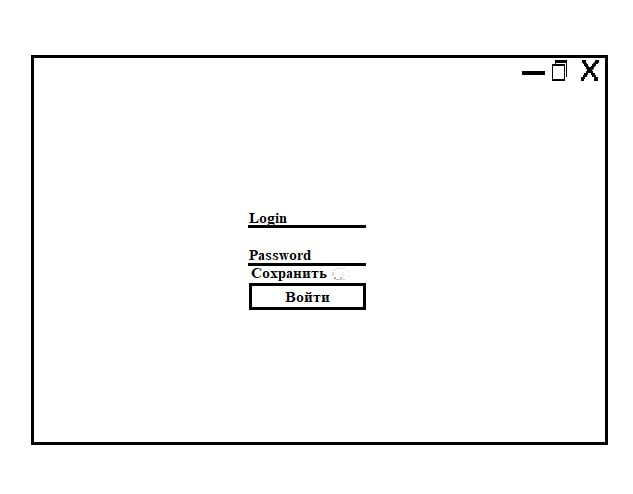


Рисунок 2.2

После нажатия на кнопку «Войти», если все данные для входа подходят, переходим на форму подтверждения входа, представленную на рисунке 2.3. Данная форма содержит поле для ввода кода и кнопку для подтверждения.

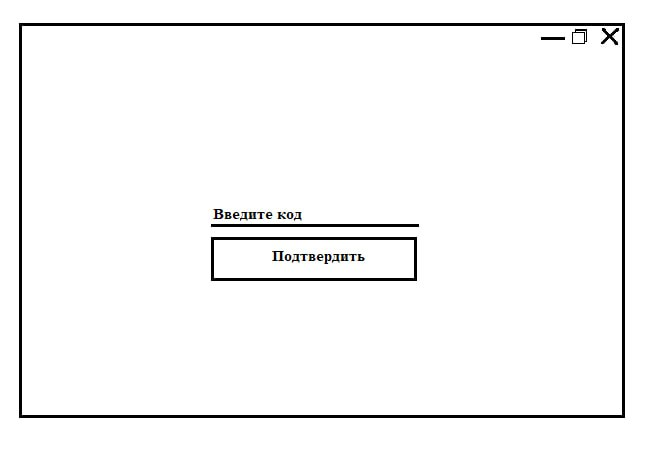


Рисунок 2.3

После нажатия на кнопку подтверждения кода, если он верен, переходим на главную форму, представленную на рисунке 2.4. Слева находиться колонка, отвечающая за переход между страницами приложения.

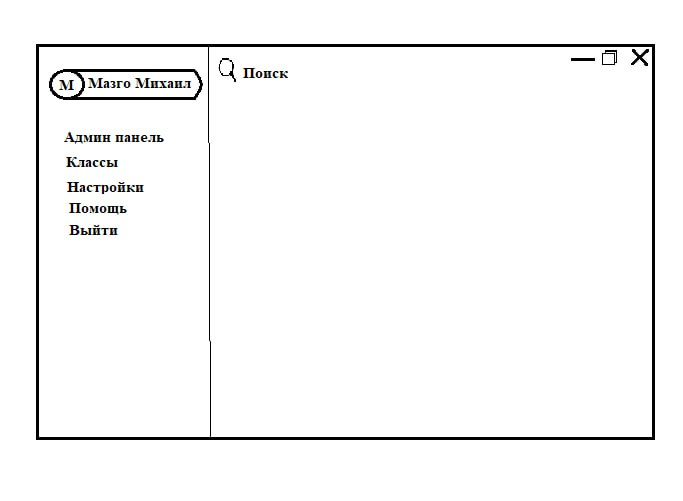


Рисунок 2.4

При нажатии на кнопку «Настройки», переходим на форму настроек аккаунта, представленную на рисунке 2.5. Данная форма предназначена для изменения настроек аккаунта.

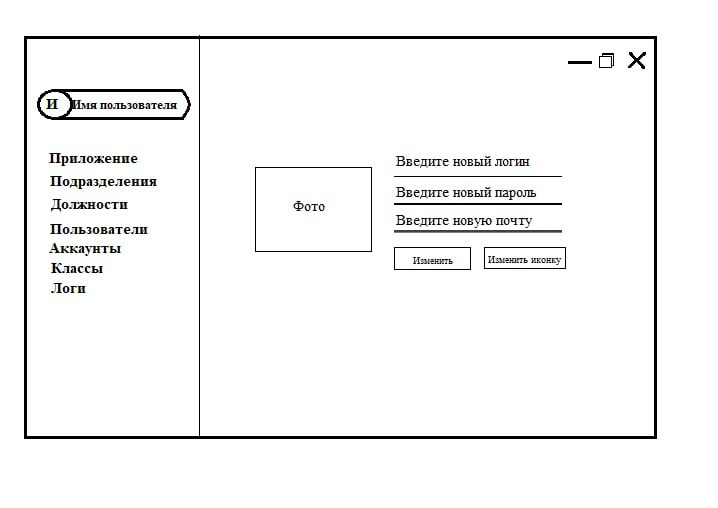


Рисунок 2.5

При нажатии на форме меню кнопки «Помощь» откроется файл, содержащий справку по приложению.

При нажатии на форме меню кнопки «Классы» переходим на форму, представленную на рисунке 2.6. Данная форма содержит список классов, к которым относиться данный пользователь.

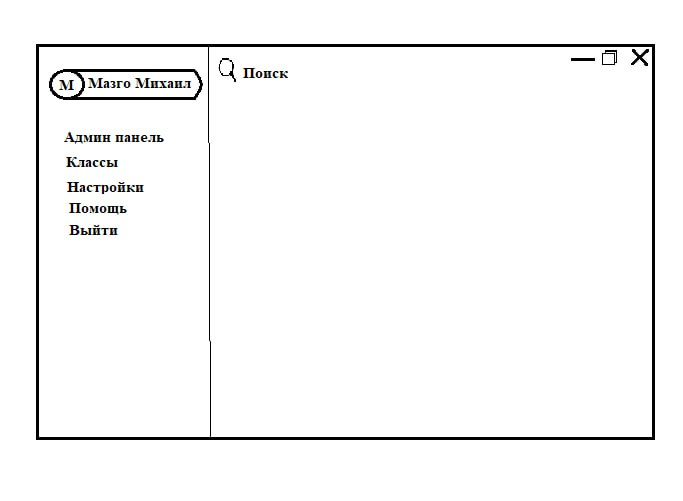


Рисунок 2.6

При нажатии на форме меню кнопки «Выйти» переходим на форму входа, которая представлена на рисунке 2.2.

При нажатии на форме меню кнопки «Админ панель», которая видна, только если пользователь имеет доступ к функциям администрирования, переходим на форму, представленную на рисунке 2.7. Данная форма содержит меню с функциями администрирования приложения.

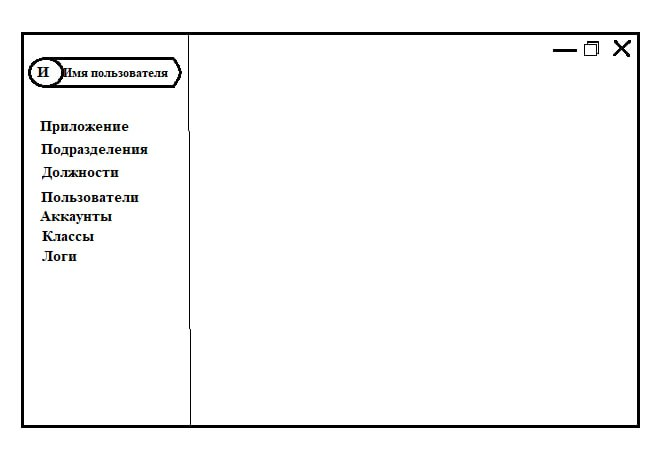


Рисунок 2.7

При нажатии на форме панели администратора кнопки «Подразделения» переходим на форму, представленную на рисунке 2.8. Данная форма содержит список с информацией о Подразделениях.

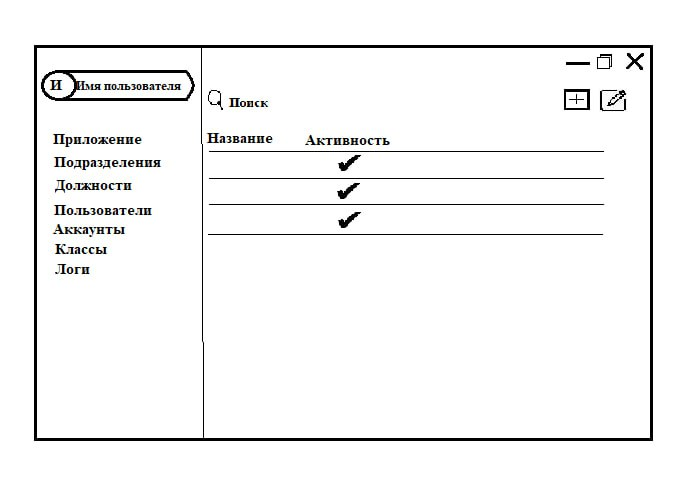


Рисунок 2.8

При нажатии на форме панели администратора, кнопки «Должности», переходим на форму, представленную на рисунке 2.9. Данная форма содержит список с информацией о должностях.

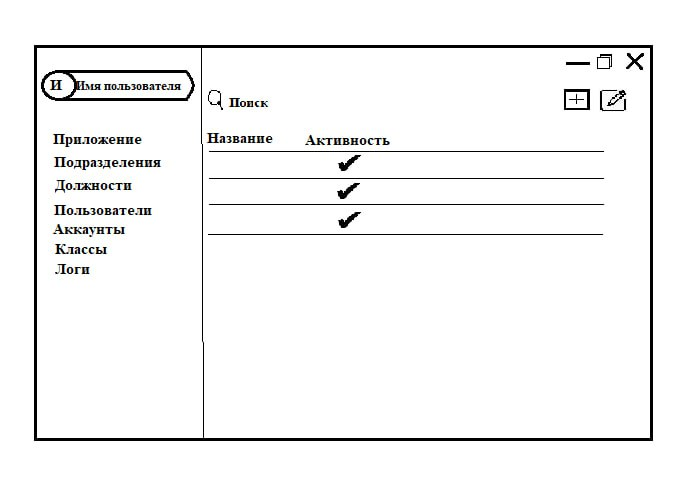


Рисунок 2.9

При нажатии на форме панели администратора кнопки «Пользователи» переходим на форму, представленную на рисунке 2.10. Данная форма содержит список с информацией о пользователях.

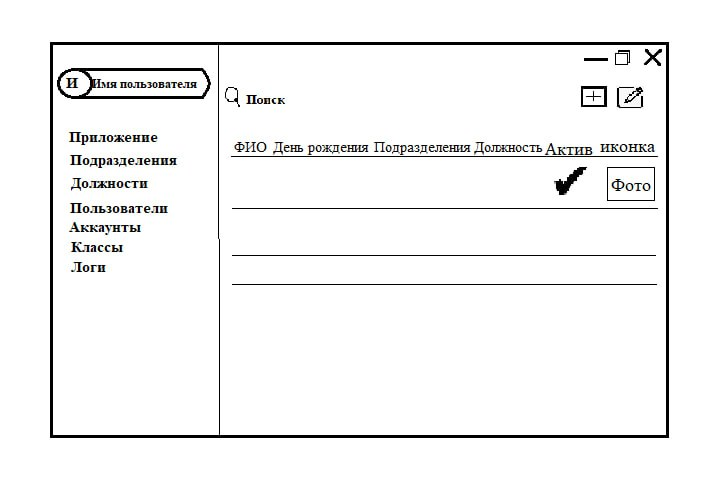


Рисунок 2.10

При нажатии на форме панели администратора кнопки «Аккаунты» переходим на форму, представленную на рисунке 2.11. Данная форма содержит список с информацией об аккаунтах.

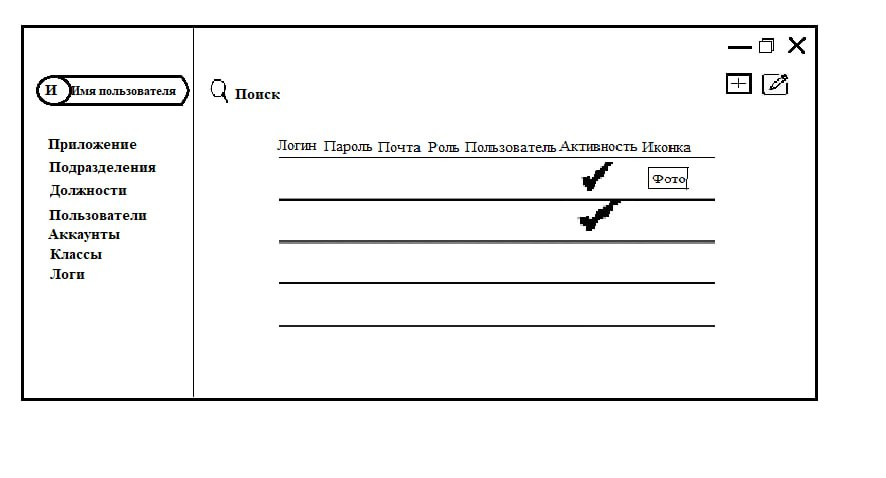


Рисунок 2.11

При нажатии на форме панели администратора кнопки «Логи», переходим на форму, представленную на рисунке 2.11. Данная форма содержит список с информацией о логах.

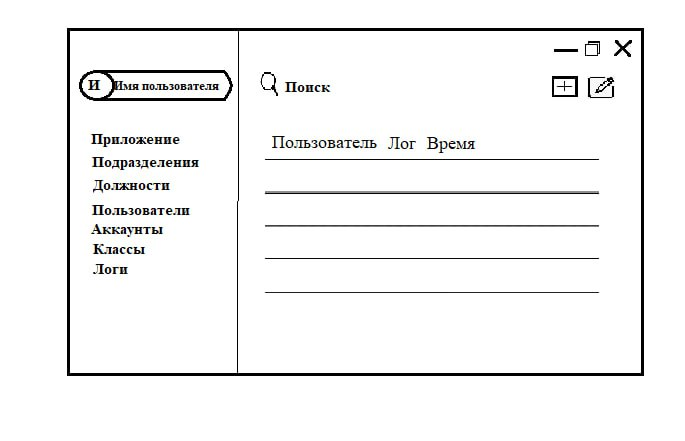


Рисунок 2.12

# 3 Описание реализации программного средства

## 3.1 Инструменты разработки и применяемые технологии

Программное обеспечение, используемое для разработки программного средства:

* среда разработки Visual Studio 2022 (версия 17.3.6);
* операционная система Windows 10;
* пакет программ Microsoft Office;
* программа для создания файлов справки Dr.Explain;
* программа для разработки диаграмм draw.io;
* структурированный язык запросов SQL;
* среда для управления SQL Server Management Studio;
* ядро реляционной база данных MSSQL 2019 Express (версия 15.0.2000.5);
* язык программирование С#.

Microsoft Visual Studio 2022 – это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio [13].

Windows 10 – операционная система семейства Windows NT производства корпорации Microsoft. Предназначена для рабочих станций, персональных компьютеров и портативных устройств; версия, предназначенная для решения серверных задач [8].

Microsoft Office – это совокупность программных средств автоматизации офисной деятельности. В состав пакета входит множество приложений, каждое из которых предназначено для выполнения определенных функций и может быть использовано автономно и независимо от остальных [12].

Dr.Explain – это приложение для быстрого создания файлов справки, справочных систем, онлайн-руководств пользователя, пособий и технической документации к программному обеспечению и техническим системам [11].

Программа «draw.io» – это интернет-ресурс, позволяющий разрабатывать различные диаграммы. Имеет огромный функционал и все необходимые инструменты, необходимые для разработки диаграмм [10].

SQL – это структурированный язык запросов, созданный для того, чтобы получать из базы данных необходимую информацию. Если описать схему работы SQL простыми словами, то специалист формирует запрос и направляет его в базу. Та в свою очередь обрабатывает эту информацию, «понимает», что именно нужно специалисту, и отправляет ответ [2].

SQL Server Management Studio (SSMS) – это интегрированная среда для управления любой инфраструктурой SQL. Используйте SSMS для доступа, настройки, администрирования и разработки всех компонентов SQL Server [2].

SQL Server Express – это бесплатная база данных начального уровня, идеально подходящая для обучения и создания классических и небольших серверных приложений, управляемых данными. Этот выпуск – лучший выбор для независимых поставщиков программного обеспечения, непрофессиональных разработчиков и любителей, создающих клиентские приложения [2].

C# – объектно-ориентированный язык программирования. Разработан в 1998-2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. В данном программном средстве не предусмотрена возможность сетевой поддержки [9].

## 3.2 Порядок авторизации пользователей

Авторизация — это процесс опознавания пользователя при попытке входа в систему. В программе 2 этапа входа, первый заключается в воде логина и пароля, далее отправляется код на электронную почту.

Авторизация пользователей происходит следующим образом:

* запуск программы;
* открытие окна авторизации;
* заполнение полей «Логин» и «Пароль»;
* получение одноразового кода для входа в систему;
* нажатие кнопки «Войти»;
* открытие окна для ввода данных в поле «Код»;
* ввод кода;
* нажатие кнопки «Подтвердить».

## 3.3 Организация данных

Реляционная база данных — это пространство, в котором связанную информацию хранят в нескольких таблицах. При этом есть возможность запрашивать информацию в нескольких таблицах одновременно.

Хранилище данных — один из основных компонентов, определяющих производительность и доступность больших и малых экземпляров SQL Server. В условиях, возросших вычислительных возможностей физических и виртуальных серверов и поддержки объемной памяти хранилища данных и подсистема ввода-вывода могут оказаться узкими местами, снижающими общую пропускную способность.

Для правильного выполнения дипломного проекта. Была создана база данных «UnilifeClassesRooms». Структура данных и их краткое описание приводятся в таблицах 3.1-3.16.

Таблица «Divisions» хранит информацию о должностях, имеющихся в учебном заведении, структура которой приведена в таблице 3.1.

Таблица 3.1 – Структура таблицы «Divisions»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «Divisions» |
| Name | nvarchar | 50 | Название подразделения |
| Active | bit | 1 | Существует ли подразделение сейчас |

Таблица «Roles» хранит информацию о ролях в приложении, структура которой приведена в таблице 3.2.

Таблица 3.2 – Структура таблицы «Roles»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «Roles» |
| Name | nvarchar | 50 | Название роли |
| PowerRole | int | 4 | Сила роли |

Таблица «ClassAccount» хранит информацию о связях аккаунтов с классами в приложении, структура которой приведена в таблице 3.3.

Таблица 3.3 – Структура таблицы «ClassAccount»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «ClassAccount» |
| AccountId | int | 4 | Название роли |
| Teacher | bit | 1 | Сила роли |
| ClassId | int | 4 | Внешний ключ к «Classes» |

Таблица «Classes» хранит информацию о классах в приложении, структура которой приведена в таблице 3.4.

Таблица 3.4 – Структура таблицы «Classes»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «Classes» |
| Name | nvarchar | 50 | Название класса |
| KeyClass | nchar | 6 | Ключ для подключения к классу |

Таблица «Users» хранит информацию о пользователях в приложении, структура которой приведена в таблице 3.5.

Таблица 3.5 – Структура таблицы «Users»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «Пользователи» |
| Name | nvarchar | 50 | ФИО Пользователя |
| Birthday | date | 3 | Дата рождения пользователя |
| PostId | int | 4 | Внешний ключ к таблице «Posts» |
| DivisionId | int | 4 | Внешний ключ к таблице «Divisions» |
| Photo | varbinary | MAX | Фото пользователя |
| BossId | int | 4 | Внешний ключ к «Users» |
| Active | bit | 1 | Активен ли пользователь в системе |

Таблица «Posts» хранит информацию о должностях, имеющихся в приложении организации, структура которой приведена в таблице 3.6.

Таблица 3.6 – Структура таблицы «Posts»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «Posts» |
| Name | nvarchar | 50 | Название должности |
| Active | bit | 1 | Существует ли подразделение сейчас |

Таблица «Accounts» хранит информацию об аккаунтах в приложении, структура которой приведена в таблице 3.7.

Таблица 3.7 – Структура таблицы «Accounts»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «Accounts» |
| Login | nvarchar | 50 | Логин для входа в аккаунт |
| Password | nvarchar | 50 | Пароль для входа в аккаунт |
| Mail | nvarchar | 50 | Почта привязанная к аккаунту |
| UserId | int | 4 | Внешний ключ к «User» |
| MailKey | nvarchar | 24 | Код подтверждения входа |
| RoleId | int | 4 | Внешний ключ к «Roles» |
| Active | bit | 1 | Активен ли аккаунт |
| Icon | varbinary | MAX | Иконка аккаунта |

Таблица «Tasks» хранит информацию о заданиях, имеющихся в классах, структура которой приведена в таблице 3.8.

Таблица 3.8 – Структура таблицы «Tasks»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «Tasks» |
| ClassId | Int | 4 | Внешний ключ к «Classes» |
| Name | nvarchar | 50 | Название задания |
| Description | nvarchar | 50 | Описание задания |
| MaxScore | int | 4 | Максимальный бал за задание |
| Deleted | bit | 1 | Удалено ли задание |
| AccountId | int | 4 | Внешний ключ к «Accounts» |
| CreateTime | datetime | 8 | Дата создания задания |
| DelivaryTime | datetime | 8 | Срок сдачи задания |

Таблица «Sessions» хранит информацию о ключах сессии у аккаунтов, структура которой приведена в таблице 3.9.

Таблица 3.9 – Структура таблицы «Sessions»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «Sessions» |
| SessionKey | nvarchar | 100 | Ключ сессии |
| AccountId | int | 4 | Внешний ключ к «Accounts» |
| Confirm | bit | 1 | Прошёл ли вариацию через почту |

Таблица «LinksTasks» хранит информацию о ссылках, прикреплённых к заданиям, структура которой приведена в таблице 3.10.

Таблица 3.10 – Структура таблицы «LinksTasks»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «LinksTasks» |
| Link | nvarchar | 200 | Ссылка на сайт |
| TaskId | int | 4 | Внешний ключ к «Tasks» |

Таблица «FilesTasks» хранит информацию о файлах, прикреплённых к заданиям, структура которой приведена в таблице 3.11.

Таблица 3.11 – Структура таблицы «FilesTasks»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «FilesTasks» |
| TaskId | int | 4 | Внешний ключ к «Tasks» |
| Name | nvarchar | 50 | Название файла |
| File | varbinary | MAX | Файл в байтовом виде |

Таблица «FilesJobs» хранит информацию о файлах, прикреплённых к работам учащихся, структура которой приведена в таблице 3.12.

Таблица 3.12 – Структура таблицы «FilesJobs»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «FilesJobs» |
| JobId | int | 4 | Внешний ключ к «Работы» |
| Name | nvarchar | 50 | Название файла |
| File | varbinary | MAX | Файл в байтовом виде |

Таблица «Jobs» хранит информацию о работах учащихся, структура которой приведена в таблице 3.13.

Таблица 3.13 – Структура таблицы «Jobs»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «Jobs» |
| AccountId | int | 4 | Внешний ключ к «Accounts» |
| Score | int | 4 | Оценка за работу |
| Comment | nvarchar | 50 | Комментарий к работе |
| TaskId | int | 4 | Внешний ключ к «Tasks» |
| Deleted | bit | 1 | Статус работы |
| Time | datetime | 8 | Время сдачи |

Таблица «Logs» хранит информацию о событиях что происходили в приложении, структура которой приведена в таблице 3.14.

Таблица 3.14 – Структура таблицы «Logs»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «Logs» |
| Log | nvarchar | 2000 | Лог приложения |
| AccountId | int | 4 | Внешний ключ к «Accounts» |
| Time | datetime | 8 | Время события |

Таблица «MessagesTasks» хранит информацию о сообщениях в заданиях, структура которой приведена в таблице 3.15.

Таблица 3.15 – Структура таблицы «MessagesTasks»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «MessagesTasks» |
| AccountId | int | 4 | Внешний ключ к «Accounts» |
| Message | nvarchar | 2000 | Сообщение |
| Time | datetime | 8 | Время отправки сообщения |
| TaskId | int | 4 | Внешний ключ к «Tasks» |

Таблица «LinksJobs» хранит информацию о сссылкаприкреплённых к работам учащихся, структура которой приведена в таблице 3.16.

Таблица 3.16 – Структура таблицы «LinksJobs»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип поля | Размер поля, байт | Описание поля |
| Id | int | 4 | Первичный ключ в таблице «LinksJobs» |
| Link | nvarchar(200) | 200 | Ссылка |
| JodId | int | 4 | Внешний ключ к «Jobs» |

Все описанные таблицы выше созданы в реляционной базе данных SQL. Диаграмма базы данных представлена на рисунке 3.1.

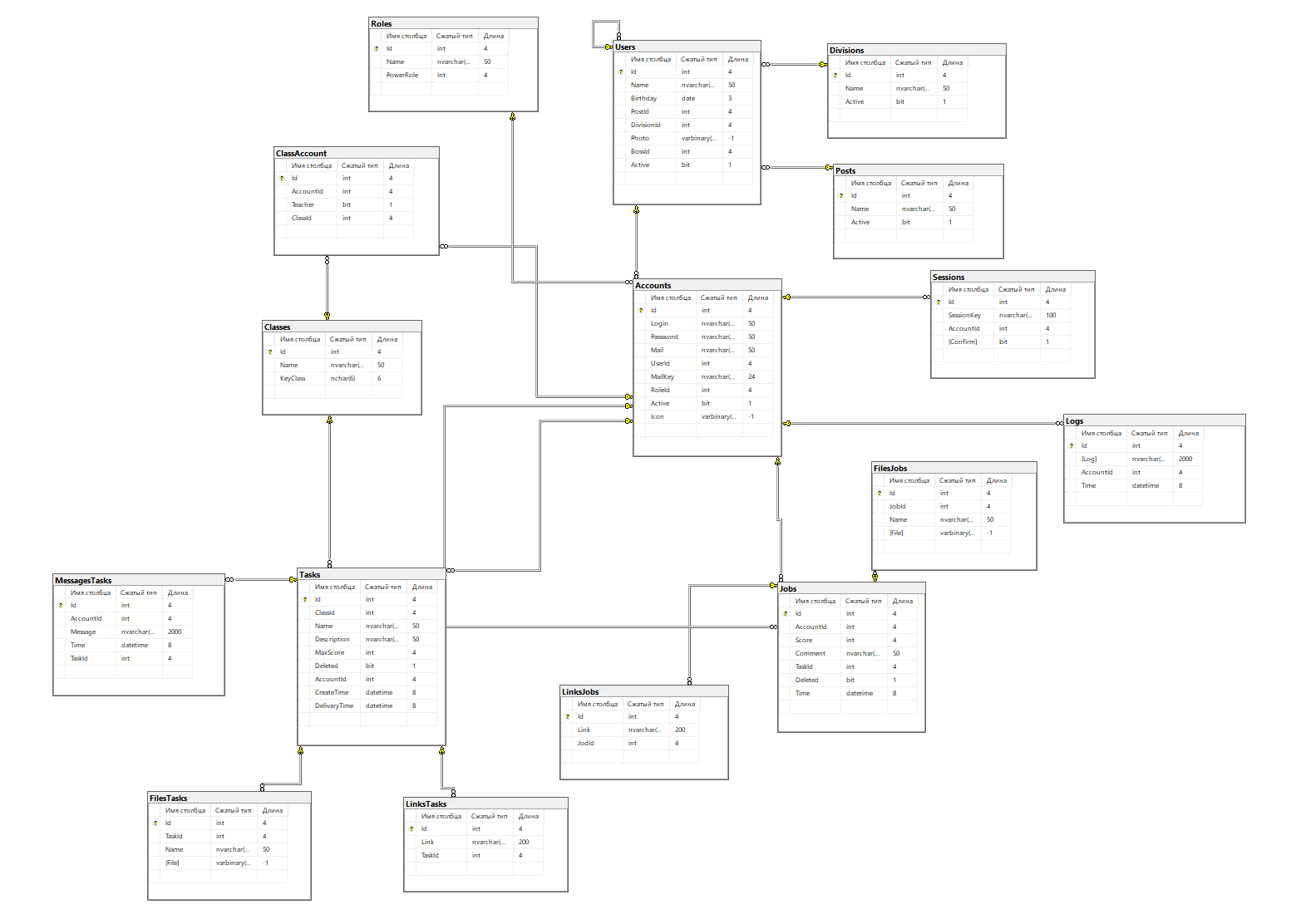


Рисунок 3.1

## 3.4 Функции: логическая и физическая организация

При реализации программного средства «UnilifeClassesRoomsDesktop.exe» был создан метод для добавления файлов и преобразования их в байтовый массив для дальнейшего хранения. Код метода представлен ниже.

public RelayCommand AddFile

{

get

{

return \_addFile ??

(\_addFile = new RelayCommand(obj =>

{

Microsoft.Win32.OpenFileDialog dlg = new Microsoft.Win32.OpenFileDialog();

dlg.FileName = ""; // Default file name

dlg.DefaultExt = ""; // Default file extension

dlg.Filter = "All files (\*.\*)|\*.\*"; // Filter files by extension

// Show open file dialog box

Nullable<bool> result = dlg.ShowDialog();

// Process open file dialog box results

if (result == true)

{

FilesJob file = new FilesJob();

file.Name = dlg.FileName.Substring(dlg.FileName.LastIndexOf('\\') + 1);

using (System.IO.FileStream fs = new System.IO.FileStream(dlg.FileName, FileMode.Open))

{

file.File = new byte[fs.Length];

fs.Read(file.File, 0, file.File.Length);

}

FilesJobs.Add(file);

}

}));

}

}

Создан метод для преобразования массива байтов в изображение для вывода на формы. Код метода представлен ниже.

public BitmapImage ToImage(byte[] array)

{

using (var ms = new System.IO.MemoryStream(array))

{

var image = new BitmapImage();

image.BeginInit();

image.CacheOption = BitmapCacheOption.OnLoad; // here

image.StreamSource = ms;

image.EndInit();

return image;

}

}

Создан метод для конвертирования изображений в массив байтов. Код метода представлен ниже.

if (user.PhotoImage != null)

{

byte[] data;

JpegBitmapEncoder encoder = new JpegBitmapEncoder();

encoder.Frames.Add(BitmapFrame.Create(divisionWindow.User.PhotoImage));

using (MemoryStream ms = new MemoryStream())

{

encoder.Save(ms);

data = ms.ToArray();

}

user.Photo = data;

}

Создано свойство, которое при выборе ссылки автоматически открывает её в браузере по умолчанию. Код метода представлен ниже.

public LinksJob SelectedLinkJob

{

get

{

return \_selectedLinkJob;

}

set

{

\_selectedLinkJob = value;

System.Diagnostics.Process.Start(\_selectedLinkJob.Link);

}

}

Создано свойство для открытия файла при выборе его в списке. Код метода представлен ниже.

public FilesJob SelectedFilesJob

{

get

{

return \_selectedFileJob;

}

set

{

\_selectedFileJob = value;

string fileName = String.Format($"../../Files/{\_selectedFileJob.Name}");

using (File.Create(fileName)) ;

File.WriteAllBytes(fileName, \_selectedFileJob.File);

string fullFileName = System.IO.Path.GetFullPath(fileName);

System.Diagnostics.Process.Start(fullFileName);

}

}

Создан класс для создания команд соответствующих патерну MVVM. Код метода представлен ниже.

public class RelayCommand : ICommand

{

internal Action<object> \_execute;

internal Func<object, bool> \_canExecute;

public event EventHandler CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)

{

this.\_execute = execute;

this.\_canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return this.\_canExecute == null || this.\_canExecute(parameter);

}

public void Execute(object parameter)

{

this.\_execute(parameter);

}

}

Полный код программы представлен в приложении А.

## 3.5 Входные и выходные данные

Входные данные – это данные, которые передаются в программное средство или функцию для обработки. Эти данные могут быть представлены в различных форматах, таких как числа, строки, булевы значения и т.д. Входные данные могут быть получены от пользователя, из файла, базы данных или других источников.

К входным данным относится информация, которая добавляется на формах: классы, аккаунты, подразделения, должности, пользователи, задания, работы и иные. Описание входных данных представлено в таблице 3.17.

Таблица 3.17 – Описание входных данных

|  |  |
| --- | --- |
| Наименование таблицы | Наименование полей для заполнения |
| Classes | Название, код доступа к классу. |
| Accounts | Пользователь, логин, пароль, mail, активен, роль. |
| Divisions | Название, активен. |
| Posts | Название, активен. |
| Users | ФИО, должность, дата рождения, подразделение, руководитель. |
| Tasks | Название, максимальный бал, описание, срок сдачи, фалы, ссылки |
| Jobs | Файлы, комментарий. |

Выходные данные – это результат обработки входных данных программным средством или функцией. Эти данные также могут быть представлены в различных форматах, в зависимости от того, что программное средство или функция должна вернуть. Например, это может быть число, строка, булево значение, массив или объект.

К выходным данным относятся документы, которые формируются в результате работы программы. К выходным данным относятся: отчеты, результаты поиска и фильтрации. Результат формирования отчёта по аккаунтам в приложении Б на рисунке Б.1. Результат получения выходных данных в результате поиска и фильтрации представлен в приложении Б на рисунке Б.2.

## 3.6 Функциональное тестирование

В процессе написания приложения необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тестирование приложения будет оформлено в виде тест-кейсов. Все действия в тест-кейсах 003-006 начинаются с главной формы приложения. Главная форма представлена в приложении Б на рисунке Б.3.

В таблице 3.18 приводятся тест-кейс для проверки функционирования приложения.

Таблица 3.18 – Тест-кейсы для проверки функционирования приложения

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/Функция | Шаги воспроизведения | Результат |
| 1 | 2 | 3 | 4 |
| 001 | Вход в аккаунт. | 1. Открыть приложение. 2. Ввод данных в поля.   «Логин» – Server1  «Пароль» – 321   1. Нажать на кнопку «Войти». | Ожидаемый результат: переход на форму подтверждения входа. |
| Фактический результат: успешный вход в аккаунт переход на форму, представленную в приложении Б на рисунке Б.4 |
| 002 | Проверка рассылки почты для подтверждения входа. | 1. Открыть приложение. 2. Ввод данных в поля.   «Логин» – Server1  «Пароль» – 321   1. Нажать на кнопку «Войти». | Ожидаемый результат: письмо на почту с кодом отправлено. |
| Фактический результат: письмо на почту с кодом пришло на почту. Результат представлен в приложении Б, рисунок Б.5 |

Продолжение таблицы 3.18

|  |  |  |  |
| --- | --- | --- | --- |
| 003 | Проверка фильтрации аккаунтов по логину. | 1. Нажать на кнопку «Админ панель». 2. Нажать на кнопку «Аккаунты». 3. В поле «Поиск» ввести «Server».   Нажать на кнопку «Поиск». | Ожидаемый результат: в таблице отобразилась информация по аккаунтам содержащих в логине «Server» |
| Фактический результат: в таблице содержащих в логине «Server». Результат представлен в приложении Б, рисунок Б.2 |
| 004 | Создание отчёта по аккаунтам. | 1. Нажать на кнопку «Админ панель» 2. Нажать на кнопку «Аккаунты» 3. Нажать на кнопку «Отчёт». | Ожидаемый результат: создан отчёт об аккаунтах. |
| Фактический результат: создан отчёт об аккаунтах в приложении. Результат представлен в приложении Б, рисунок Б.1 |
| 005 | Выход из аккаунта | 1. Нажать на кнопку «Выйти» | Ожидаемый результат: выход из аккаунта, переход на форму входа. |
| Фактический результат: выход из аккаунта, переход на форму входа. Результат представлен в приложении Б, рисунок Б.5 |
| 006 | Изменение настроек аккаунта. | 1. Нажать на кнопку «Настройки» 2. Ввод данных в поле   «Пароль» – 321   1. Нажимаем на кнопку «Изменить» | Ожидаемый результат: Пароль аккаунта изменён. |
| Фактический результат: Пароль аккаунта изменён. Результат представлен в приложении Б, рисунок Б.6 |

## 3.7 Описание справочной системы

Справочная система состоит из 4 разделов.

В первом разделе «Введение» описываются цели и применение программного средства.

Во втором разделе описывается процесс установки и настройки приложения.

В третьем раздел содержится информация об использовании стандартных функций приложения для рядовых пользователей.

В четвёртом разделе содержится информация об использовании функций администратора в приложении.

Справочная система представленна на рисунках 3.2-3.5.

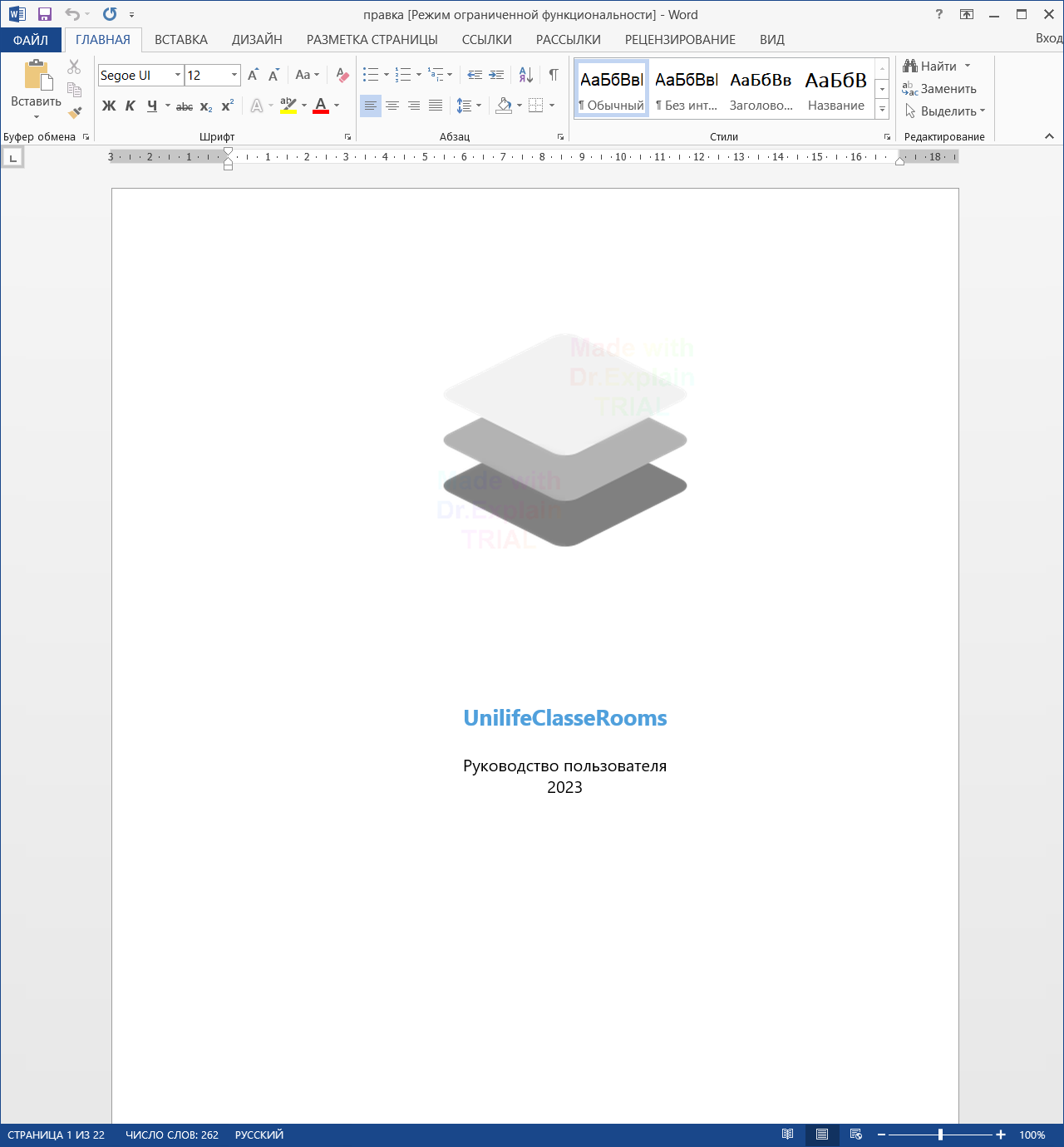


Рисунок 3.2

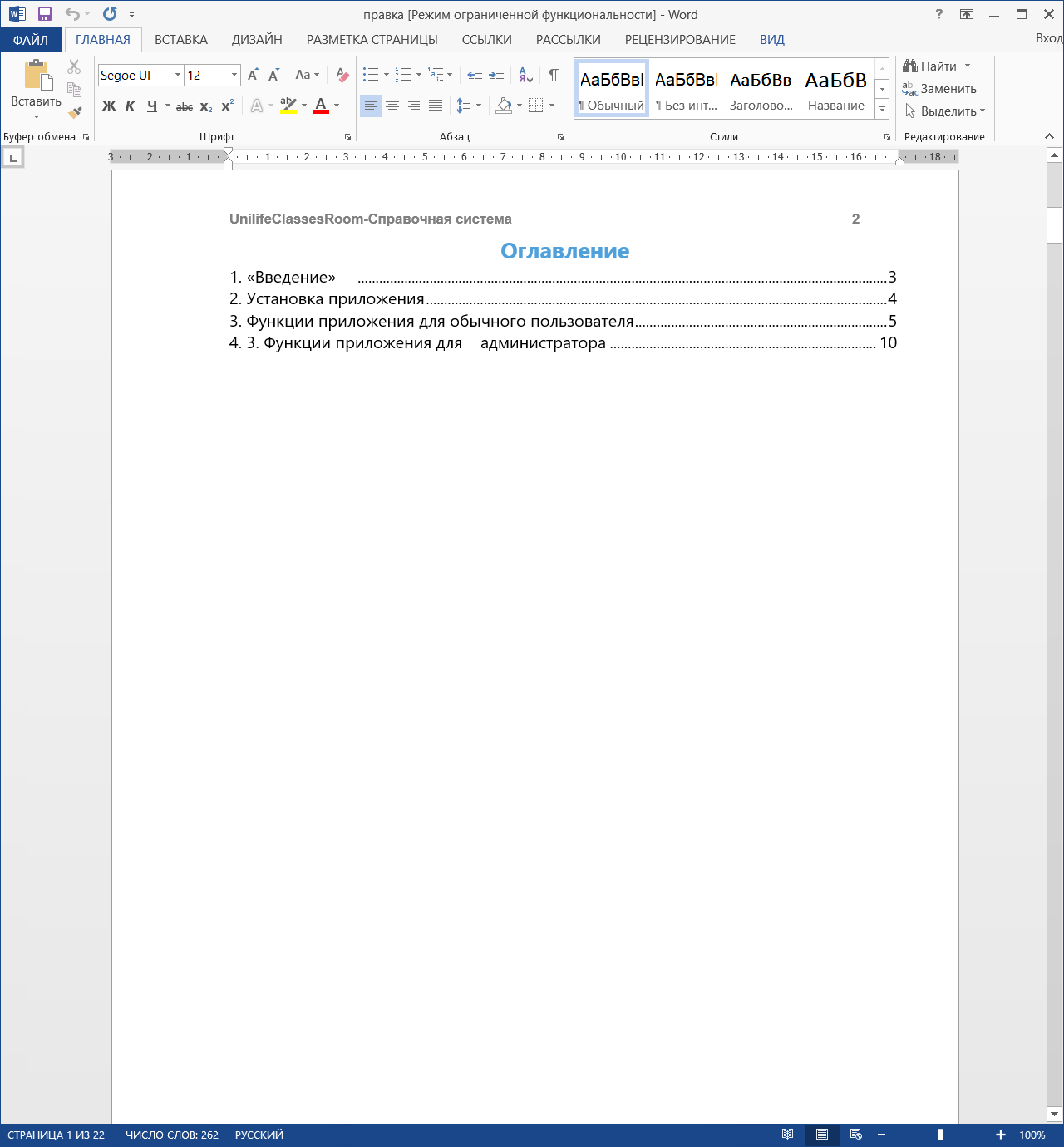


Рисунок 3.3

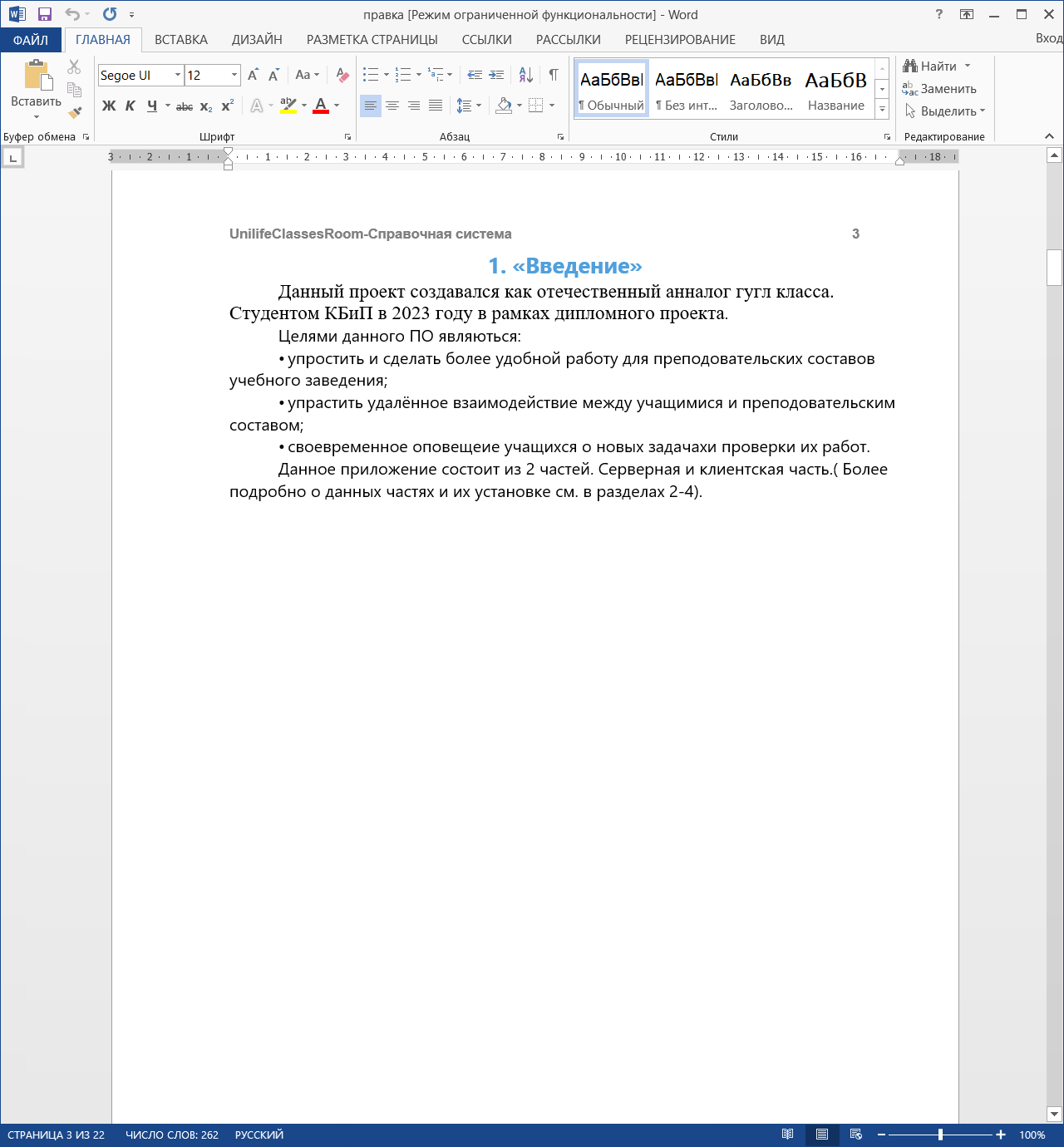


Рисунок 3.4

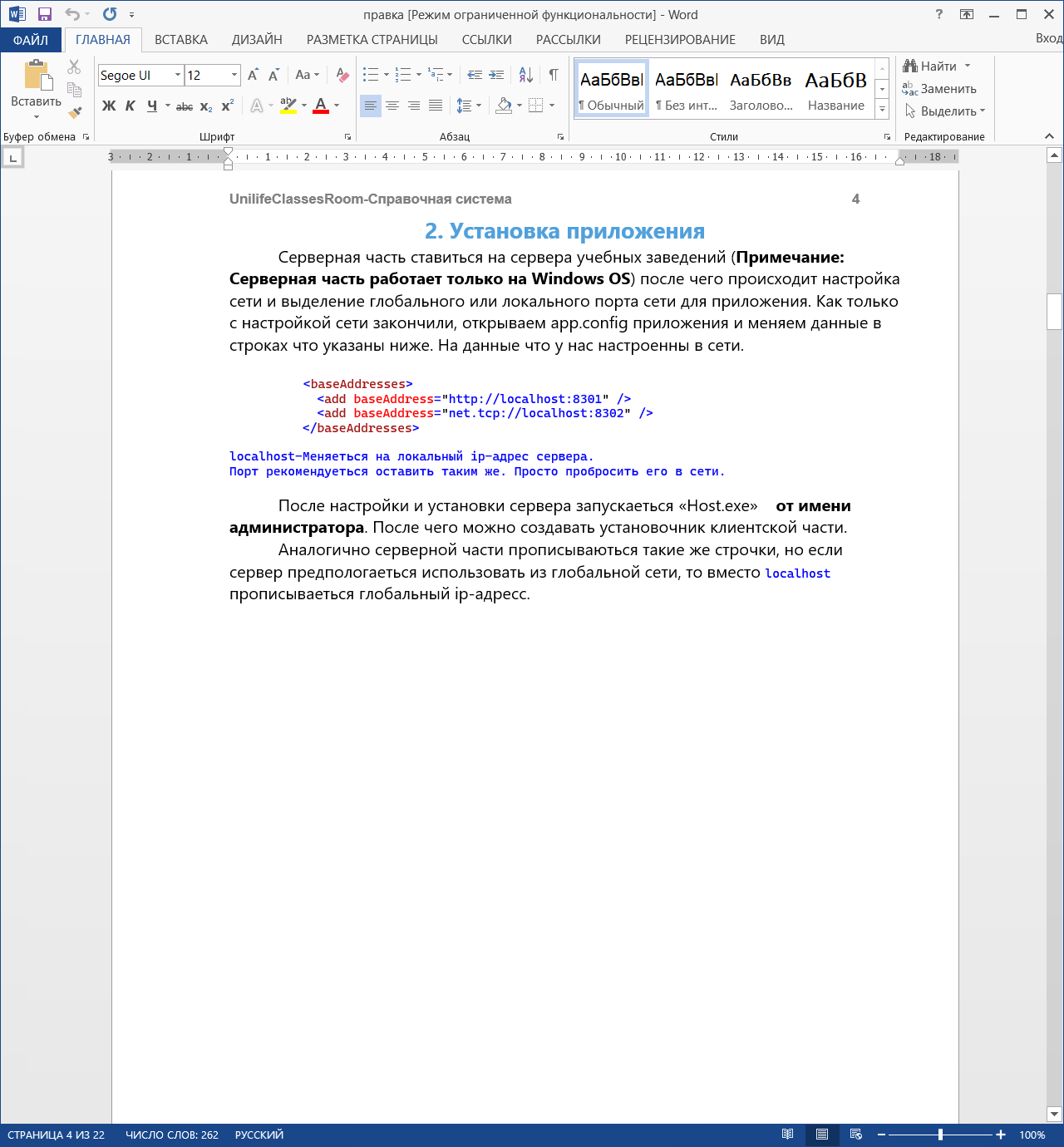


Рисунок 3.5

# 4 Применение

## 4.1 Назначение программного средства

Программное средство «UnilifeClassesRoom.exe» включает в себя ведение базы данных классов, аккаунтов, пользователей, подразделений, заданий, работ. С помощью данного программного средства упрощается работа учителей, облегчается поиск нужного класса, задания, работ по учащимся, а также поиск и фильтрация по интересующей сотрудника информации. В программном средстве предусмотрено отслеживание действий пользователей.

Данное программное средство применимо в учебных заведениях.

## 4.2 Условия применения

Для применения данного программного средства необходимо получить установочные файлы от системного администратора организации где программа применяется. Установить «UnilifeClassesRoomsDiplomDesktop» на компьютер с доступом в интернет. Для экспорта отчетов необходимо иметь установленные на компьютере программные средства Microsoft Word.

# 5 Охрана труда и охрана окружающей среды

**5.1 Правовые, нормативные, социально-экономические и организационные вопросы охраны труда.**

В ГУО «Средняя школа №133» разработано и функционирует Система управления охраной труда (СУОТ). СУОТ разработан в соответствии с требованиями нормативных правовых актов Республики Беларусь в области охраны труда и СТБ 18001-2009 «Системы управления охраной труда. Требования» и направлено на создание на предприятии условий, обеспечивающих сохранение жизни и здоровья работников в процессе трудовой деятельности, соблюдение законодательства в области охраны труда.

Для организации работы и осуществление контроля по охране труда в учреждении назначено директором лицо, отвечающее за организацию охраны труда.

Основными задачами ответственного лица:

* координация деятельности структурных подразделений организации по обеспечению здоровых и безопасных условий труда;
* совершенствование СУОТ;
* внедрение передового опыта и научных разработок по безопасности и гигиене труда, пропаганда охраны труда;
* информирование и консультирование работников организации, в том числе ее руководителя, по вопросам охраны труда;
* осуществление контроля за соблюдением требований актов законодательства, технических нормативных правовых актов и локальных нормативных правовых актов, содержащих требования охраны труда.

Ответственное лицо в учереждении организует выполнение мероприятий по устранению недостатков и нарушений по охране труда, выявленных в ходе проведения ежемесячного контроля. О нарушениях, которые не могут быть устранены работниками структурного подразделения, сообщается директору учереждения.

Организация, оперативное руководство и координация работы по охране труда, общее руководство управлением охраной труда осуществляется директором ГУО «Средняя школа №133». Ответственным за реализацию функций СУОТ является лицо, назначенное директором

Ответственный по охране труда осуществляет организационно-методическую работу по управлению охраной труда, подготовку управленческих решений, контроль за их реализацией под руководством директора.

Функции ответственного лица по охране труда:

* организация обучения, повышения квалификации и проверки знаний по вопросам охраны труда руководителей и специалистов учреждения.
* подготовка государственной статистической отчетности по охране труда;
* пропаганда и распространение передового опыта в области охраны труда в организациях, расположенных на подведомственной территории, в том числе через средства массовой информации, организацию выставок, смотров-конкурсов, семинаров, совещаний и других мероприятий;
* участие в установленном законодательством порядке в расследовании несчастных случаев на производстве.

В соответствии с требованиями ст.17 Закона от 23.06.2008 №356-З «Об охране труда» и ст.226 Трудового кодекса работодатель обязан осуществлять обучение, стажировку, инструктаж и проверку знаний, работающих по вопросам охраны труда.

Обучение по охране труда работников учреждения производится в соответствии с действующими Государственными стандартами, межотраслевыми положениями, типовыми инструкциями и программами.

Обучение руководителей и специалистов осуществляется по учебным планам и программам, составленным на основании типового перечня вопросов для обучения и проверки знаний по вопросам охраны труда руководителей и специалистов в соответствующих учреждениях.

Руководитель организации утверждает перечень должностей специалистов, которые должны проводить стажировку и определяет ее продолжительность.

Не позднее месяца со дня назначения на должность и периодически в соответствии с требованиями нормативных правовых актов, но не реже одного раза в три года, руководители и специалисты проходят проверку знаний по вопросам охраны труда и комиссиях для проверки знаний по вопросам охраны труда.

Ответственность за организацию своевременного и качественного проведения обучения, инструктажа, стажировки и проверки знаний работников по вопросам охраны труда в учреждении возлагается на директора.

Вводный инструктаж с вновь принятыми работниками всех специальностей проводит ответственное лицо по охране труда.

Первичный, повторный, целевой и внеплановый инструктажи проводятся лицом назначенным директором.

Расследование и учет несчастных случаев, профессиональных заболеваний в ГУО «Средняя школа №133» осуществляется в соответствии с «Правилами расследования и учета несчастных случаев на производстве и профессиональных заболеваний», утвержденными Постановлением Совета Министров Республики Беларусь от 15.01.2004 г. №30.

Расследование несчастного случая, профессионального заболевания, на производстве проводится с целью выявления вызвавших их причин и принятия соответствующих мер для предотвращения повторения подобных ситуаций, а именно:

* установления причин несчастного случая и определение мер по экстренному устранению причин травм.
* установления должностных и иных лиц, виновных в нарушениях, которые привели к несчастному случаю; какие требования норм, правил были нарушены;
* привлечения к ответственности виновных лиц на основании законодательства и иных нормативных правовых актов.

Основными видами контроля на предприятии являются:

* контроль за соблюдением законодательства об охране труда, осуществляемый руководителями и специалистами предприятия в соответствии с их должностными обязанностями;
* контроль по охране труда, осуществляемый службой охраны труда предприятия в соответствии с Типовым положением о службе охраны труда организации, утвержденным постановлением Министерства труда и социальной защиты Республики Беларусь от 24 мая 2002 г. N82 (Национальный реестр правовых актов Республики Беларусь, 2002 г., N89, 8/8286);
* периодический контроль за соблюдением законодательства об охране труда, осуществляемый представителями нанимателя с участием общественных инспекторов профсоюзов по охране труда.

В ГУО «Средняя школа №133» периодический контроль осуществляется представителями нанимателя с участием общественных инспекторов профсоюзов по охране труда, проводится:

* ежедневно;
* ежемесячно;
* ежеквартально.

Ежедневно контроль за состоянием охраны труда осуществляется их непосредственным руководителем.

Ежемесячно контроль проводится в каждом структурном подразделении организации, руководителем этих подразделений.

Ежеквартально контроль осуществляется в учреждении в целом директором учреждения с участием руководителей структурных подразделений, общественных инспекторов по охране труда [14].

**5.2 Разработка мер по нормализации и защите от электромагнитного излучения при эксплуатации ПЭВМ**

Инструментальный контроль электромагнитной обстановки, создаваемой видеодисплейными терминалами (далее – ВДТ), электронно-вычислительными машинами (далее – ЭВМ) и персональными электронно-вычислительными машинами (далее – ПЭВМ), производится:

* при аттестации рабочих мест по условиям труда;
* при комплексной гигиенической оценке условий труда работников; при вводе ВДТ, ЭВМ и ПЭВМ в эксплуатацию, а также организации
* новых и переоборудованию существующих рабочих мест;
* после проведения организационно-технических мероприятий, направленных на нормализацию электромагнитной обстановки;
* в рамках государственной санитарно-гигиенической экспертизы; при проведении производственного контроля.

Инструментальный контроль осуществляется органами и учреждениями, осуществляющими государственный санитарный надзор, и (или) испытательными лабораториями (центрами), аккредитованными в установленном порядке.

Инструментальный контроль уровней электромагнитных полей (далее – ЭМП) должен осуществляться приборами с допускаемой основной относительной погрешностью измерений, не превышающей ±25 %, включенными в Государственный реестр средств измерений Республики Беларусь и имеющими действующие свидетельства о поверке.

Следует отдавать предпочтение измерителям с изотропными антеннами-преобразователями.

Перед проведением измерений переменных электрических и магнитных полей (далее – ЭП и МП), статических электрических полей от ВДТ, ЭВМ и ПЭВМ должна быть обеспечена подготовка ВДТ, ЭВМ или ПЭВМ и средств измерения в соответствии с их эксплуатационной документацией.

При проведении измерений переменных ЭП и МП, статических электрических полей на рабочем месте, оборудованном ВДТ, ЭВМ и ПЭВМ, должно быть включено все другое используемое для работы электрооборудование, размещенное в данном помещении, в том числе устройства общего и местного освещения.

При проведении измерений переменных ЭП и МП, статических электрических полей от ВДТ, ЭВМ и ПЭВМ в рамках государственной санитарно-гигиенической экспертизы должно быть выключено все другое используемое для работы электрооборудование, размещенное в данном помещении, в том числе устройства общего и местного освещения.

Измерения проводятся в нормальных климатических условиях, если другое не установлено технической и эксплуатационной документацией на ВДТ, ЭВМ и ПЭВМ.

Измерения проводятся не ранее чем через 20 минут после включения питания.

Для проведения измерений переменных ЭП и МП полей на рабочих местах необходимо установить на экране ВДТ, ЭВМ и ПЭВМ типичное для данного вида работы изображение (текст, графики и другое).

Для проведения измерений переменных ЭП и МП в рамках государственной санитарно-гигиенической экспертизы вся рабочая зона экрана (темный фон) заполняется светлой прописной буквой «Н», имеющей высоту 3 мм, которая соответствует расстоянию наблюдения 500 мм и оптимальному угловому размеру знака 20 градусов.

Измерение уровней переменных ЭП и МП, статических электрических полей на рабочем месте, оборудованном ВДТ, ЭВМ и ПЭВМ, производится:

* для дисплеев на электронно-лучевой трубке (далее – ЭЛТ) и плоских дискретных экранов (жидкокристаллические, плазменные и другое) – в точках, расположенных на расстоянии 0,5 м от экрана дисплея на высоте 0,5 м, 1,0 м и 1,5 м от поверхности пола;
* для дисплеев портативных компьютеров – в точках, расположенных на расстоянии 0,4 м от центра клавиатуры портативного компьютера на высоте 0,5 м, 1,0 м и 1,5 м от поверхности пола.

Измерение уровней переменных ЭП и МП, статических электрических полей от ВДТ, ЭВМ и ПЭВМ в рамках государственной санитарно-гигиенической экспертизы производится:

* для дисплеев на ЭЛТ и плоских дискретных экранов (жидкокристаллические, плазменные) – в 1 точке, расположенной по нормали к центру экрана на расстоянии 0,5 м от экрана дисплея (рисунок); во 2 точке, расположенной по нормали к центру левой боковой поверхности дисплея на расстоянии 0,5 м; в 3 точке, расположенной по нормали к центру правой боковой поверхности дисплея на расстоянии 0,5 м; в 4 точке, расположенной по нормали к центру тыльной поверхности дисплея на расстоянии 0,5 м;23
* для дисплеев портативных компьютеров – в 1 точке, расположенной по нормали к центру экрана на расстоянии 0,4 м от центра клавиатуры портативного компьютера (рисунок 5.1); во 2 точке, расположенной по нормали к центру левой боковой поверхности дисплея на расстоянии 0,5 м; в 3 точке, расположенной по нормали к центру правой боковой поверхности дисплея на расстоянии 0,5 м; в 4 точке, расположенной по нормали к центру тыльной поверхности дисплея на расстоянии 0,5 м.

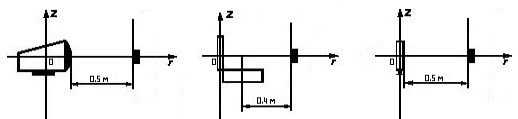


Рисунок 5.1 Расположение измерительной антенны относительно экранов дисплеев ВДТ, ЭВМ или ПЭВМ

В каждой точке необходимо проводить не менее трех измерений ЭМП. Для гигиенической оценки выбираются максимальные из измеренных на различных высотах средних значений.

Измерения уровней ЭМП от оборудования (клавиатура, системный блок, манипулятор «мышь», беспроводные системы передачи информации и иные периферийные устройства), использование которого связано с возможным его приближением к телу человека на расстояние менее чем 0,1±0,01 м, а также контактирующих с телом человека или отдельными его частями, проводятся на расстоянии 0,05±0,01 м от корпуса такого оборудования.

Измерения уровней ЭМП от оборудования (клавиатура, системный блок, манипулятор «мышь», беспроводные системы передачи информации и иные периферийные устройства), использование которого связано с возможным его приближением к человеку на расстояние более чем 0,1±0,01 м, проводятся на расстоянии 0,5±0,02 м от корпуса такого оборудования.

Гигиеническая оценка воздействия ЭМП различных частот должна производиться на соответствие установленным предельно-допустимым уровням (далее – ПДУ) для соответствующего диапазона частот.

Беспроводные периферийные устройства (клавиатуры, манипуляторы «мышь», беспроводные системы передачи информации и другое) оцениваются в соответствии с частотой ЭМП, указанной на самих устройствах или в технико-эксплуатационной документации. Проводные периферийные устройства (клавиатуры, манипуляторы «мышь», принтеры, сканеры и иные), а также системные блоки оцениваются на частоте 0,3-300 кГц.

Гигиеническая оценка уровней электромагнитных полей должна производиться с учетом погрешности средства измерения. При этом с нормативным значением сравнивается результат измерения, к которому прибавлена абсолютная погрешность средства измерения.

При проведении инструментального контроля ЭМП от экранов ВДТ, ЭВМ или ПЭВМ в помещениях с высоким уровнем фонового ЭМП промышленной частоты 50 Гц (Уровни ЭМП промышленной частоты 50 Гц в помещении при выключенных ВДТ, ЭВМ или ПЭВМ с напряженностью ЭП более 8 В/м и индукции МП более 80 нТл.), в которых уровни ЭМП в диапазоне 5 Гц-2 кГц превышают значения установленных нормативов, измерения в этом диапазоне рекомендуется проводить следующим образом:

* необходимо использовать средство измерения, обеспечивающее возможность раздельного измерения ЭП и МП в полосе частот 45-55 Гц и в диапазоне частот 5 Гц-2 кГц с вырезанной полосой частот 45-55 Гц;
* уровни ЭП и МП на рабочих местах пользователей ВДТ, ЭВМ или ПЭВМ следует считать предельно-допустимыми, если в полосе частот 45-55 Гц они не превышают предельно-допустимых для населения (напряженности ЭП 500 В/м и индукции МП 5 мкТл), а в оставшейся части диапазона частот 5 Гц-2 кГц не превышают ПДУ (напряженность ЭП 25 В/м и индукция МП 0,25 мкТл), предусмотренные Гигиеническим нормативом «Предельно-допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами», утвержденным настоящим постановлением (далее – Гигиенический норматив);
* допускается для раздельной оценки соответствующих частотных составляющих использовать два отдельных прибора, один из которых измеряет ЭМП во всем диапазоне частот 5 Гц-2 кГц, а другой – на промышленной частоте 50 Гц в полосе шириной ±5 Гц[(Измерения уровней ЭМП промышленной частоты 50 Гц проводятся при выключенных ВДТ, ЭВМ или ПЭВМ.).](#_page_49_0) В качестве уровней ЭП и МП, создаваемых ВДТ, ЭВМ или ПЭВМ, следует брать абсолютную разницу в показаниях этих приборов с учетом их абсолютных погрешностей. Она не должна превышать ПДУ, предусмотренные Гигиеническим нормативом. ЭМП промышленной частоты не должны превышать предельно-допустимых уровней для населения (относительная погрешность каждого из приборов не должна превышать ±25 %).

**5.3 Пожарная безопасность**

В ГУО «Средняя школа №133» действуют следующие пожарные правила.

Начнем со школьной территории. Школьный двор должен быть не только ухоженным и красивым, но и пожаробезопасным. Для этого необходимо поддерживать его в чистоте. Прежде всего он не должен быть захламлен мусором, особенно сгораемым: листвой, бумагой, старой мебелью и т. п. Достаточно непогашенной спички для их воспламенения, а до здания рукой подать. Ветром искры от огня могут быть занесены и в открытое окно школы, и на кровлю здания. Поэтому Правилами пожарной безопасности запрещено складировать на территории школ горючие материалы. Они должны регулярно убираться и вывозиться за пределы школы. Этими же правилами категорически запрещено разведение костров и сжигание мусора на школьных дворах.

Содержать в чистоте территорию школы надо не только для того, чтобы она выглядела красиво. Загромождение подъездов к зданию школы не дает возможности пожарным машинам подъехать к ней в случае необходимости. И вы можете догадаться, чем может обернуться потерянное при пожаре время.

В здании школы должно быть не менее двух эвакуационных выходов. Если один из них отрезан огнем, то для спасения необходимо использовать другой. Двери выходов должны открываться наружу и быть оборудованными запорами, которые легко открываются изнутри. Над дверями этих выходов должна быть надпись: "Запасный выход".

Расстановка мебели и оборудования в классах, мастерских и других помещениях не должна препятствовать эвакуации людей, а также их подходу к средствам пожаротушения. Эвакуационные проходы, лестницы нельзя загромождать какими-либо предметами и оборудованием.

Кому-то эти требования могут показаться мелочными и даже ненужными. Но так можно рас­суждать, когда все спокойно и нет никакой опасности. Если же в школе сложилась чрезвычайная ситуация и большое количество учащихся одновременно стараются покинуть здание, то эти "мелочи" могут привести к человеческим жертвам.

В школе есть помещения и кабинеты, являющиеся пожароопасными. К ним относятся химические лаборатории, кабинеты химии и физики, учебные мастерские. В них используются легковоспламеняющиеся жидкости, горючие вещества, имеются электрические приборы. Работа с ними должна проводиться только в присутствии педагогов или лаборантов. Перед началом работы необходимо убедиться в исправности электроприборов и проверить наличие соответствующих этикеток на банках с реактивами. Работать с огнеопасными жидкостями нужно только в вытяжных шкафах при включенной вентиляции.

Закончив работу, промасленные ветошь или тряпки нужно убрать в металлический ящик с крышкой. После занятий ящик должен быть очищен. Следует убрать мусор и отходы, не забыть выключить электроприборы.

Вы также должны бережно относиться к первичным средствам пожаротушения, имеющимся в школе. Возможно, в чрезвычайной ситуации они спасут вам или кому-то жизнь.

Действия при пожаре в школе. Пожары всегда опасны для людей. Но пожар в школе или другом общественном здании более опасен из-за большого скопления людей. Опасность состоит в том, что они обычно плохо знают пути спасения. Печальный опыт свидетельствует, что большинство людей в экстремальной ситуации пытается спастись знакомым путем, поэтому они часто бегут буквально в огонь. Чтобы этого не произошло, находясь в любом помещении, вы должны узнать, где находится аварийный или эвакуационный выход. Нельзя забывать об этом и в салоне автобуса, троллейбуса или авиалайнера.

Каждый учащийся должен правильно и без паники действовать при пожаре в школе. Ученик, заметивший загорание, обязан немедленно сообщить об этом учителю или дежурному МЧС по телефону 101. Чем быстрее прибудут спасатели, тем легче и успешнее будет ликвидирован пожар и оказана помощь людям. Не надо тратить время на выяснение причин и поиск места возгорания, потому что пожар тем временем может принять большие размеры. Этому способствует отделка учебных помещений деревом, пластмассами и другими горючими материалами. Не надо забывать и об особой опасности мастерских, а также физических и химических лабораторий.

После сообщения о пожаре все учащиеся и взрослые должны покинуть помещение через основные или запасные пути эвакуации. Эвакуация должна проводиться немедленно и из всего здания независимо от размеров пожара. В первую очередь эвакуируются учащиеся из кабинетов, где создалась наибольшая угроза для жизни.

Каждое школьное здание имеет не менее двух эвакуационных выходов. В случае невозможности использовать один выход, отрезанный огнем, используется другой. В каждой школе также имеется план эвакуации на случай пожара или другого чрезвычайного происшествия. Обязанность всех учащихся — ознакомиться с планом эвакуации, его содержанием, определить на нем место расположения своего класса и пути движения при эвакуации. План вывешивается на хорошо видимых местах, примыкающих к путям эвакуации.

Эвакуация (при пожаре) — процесс движения людей из помещения, здания по эвакуационным путям с целью предотвращения возможного воздействия на них опасных факторов пожара.

Эвакуационный путь — это путь от возможного места пребывания человека по свободным проходам до выхода из здания.

При неорганизованной эвакуации может возникнуть паника, что часто заканчивается трагически. Например, в классе все ученики одновременно встают со своих мест и направляются к выходу. Вследствие ограниченной пропускной способности эвакуационных путей и выходов создается толчея. Она уменьшает скорость движения учащихся. Это приводит к противоречию: чем быстрее люди стремятся покинуть помещение, тем больше времени они вынуждены затратить на это. Чтобы такого не произошло, крайне важными при эвакуации являются дисциплинированность и четкое выполнение распоряжений учителя.

Передвижение должно осуществляться организованно и без паники. Не толкайте друг друга. При необходимости оказывайте помощь другим учащимся, особенно младших классов.

Эвакуируясь с верхних этажей, используйте лестничные клетки. При их задымлении раскройте или разбейте окна на лестничных площадках или в их пределах. Не забудьте закрыть двери помещений, откуда проникает дым. Если помещение задымлено, двигайтесь, пригнувшись к полу. Не забывайте защищать органы дыхания известными вам средствами.

В случае невозможности эвакуации обычным путем можно попытаться вылезти через окна, если класс находится на первом этаже. Если и это не­возможно, необходимо поступать, как в ситуациях при пожаре в жилище. Важно только не паниковать! С верхних этажей людей спасают, используя стационарные и приставные лестницы.

Если загорание небольшое и пожар не принял угрожающих размеров, нужно до прибытия пожарных потушить огонь самим. Для этого используются первичные средства пожаротушения: огнетушители, вода (в том числе из пожарных кранов), а также плотная ткань.

После вызова спасателей не забывайте об их встрече. Встречающий должен проинформировать пожарное подразделение о сложившейся обстановке, есть ли угроза людям, где они находятся.

Для того, чтобы действия учащихся при возможном пожаре были слаженными, в каждой школе ежегодно проводится "День гражданской обороны" или "День защиты детей". На этих мероприятиях отрабатываются действия на случай пожара в школе. Отнеситесь к этому серьезно и ответственно.

**5.4 Охрана окружающей среды**

В организации ГУО «Средняя школа №133» за охрану окружающей среды отвечает лицо, назначенное директором учреждения.

Основные меры по охране окружающей среды предпринимаемыми в организации являются:

* обезвреживание реактивов после уроков химии, приведение их в нейтральное состояние;
* снижение потребления электроэнергии;
* отправка мусора и сточных вод на переработку;
* сбор макулатуры.

После приведения реактивов в нейтральное состояние они отправляться в сточные воды, после чего перерабатываться на станции очистки и поступают в водопровод.

Процесс очистки сточных вод делиться на 4 этапа:

* механический;
* биологический;
* физико-химический;
* дезинфекция сточных вод.

На механическом этапе производится предварительная очистка поступающих на очистные сооружения сточных вод с целью подготовки их к биологической очистке. На механическом этапе происходит задержание грубых и тонкодисперсных примесей.

Сооружения для механической отчистки:

* решётки (или УФС — устройство фильтрующее самоочищающееся) и сита;
* песколовки;
* первичные отстойники;
* фильтры;
* септики.

На биологическом этапе происходит очистка от растворённой части загрязнений сточных вод специальным биоценозом, который называется активным илом или биоплёнкой.

На физико-химическом используют для очистки от растворённых примесей, а в некоторых случаях и от взвешенных веществ. Многие методы физико-химической очистки требуют предварительного глубокого выделения из сточной воды взвешенных веществ, для чего широко используют процесс коагуляции.

Дезинфекция сточных вод происходит путём обеззараживания с помощью ультрафиолетового излучения. Для обеззараживания биологически очищенных сточных вод, наряду с ультрафиолетовым облучением, которое используется, как правило, на очистных сооружениях крупных городов, применяется также обработка хлором в течение 30 минут.

Снижение электропотребления достигается путём использования энергоэффективных устройств. Отключение электричества в целых помещения что позволяет минимизировать потери электроэнергии при передаче через провода.

Отправка мусора на переработку происходит после сбора мусора в кабинетах где были уроки. После чего их выносят в специальные мусорные баки, которые затем вывозиться на специальный полигон где проходят сортировку и дальнейшую переработку.

Сбор макулатуры осуществляется сотрудниками учреждения, а также учащимися. После окончания сборов взвешивается макулатура, и её грузят в самосвалы, которые увозят её на завод по переработки бумаги.

# 6 Экономический раздел

## 6.1 Технико-экономическое обоснование разработки программного средства

Цель экономического раздела дипломного проекта по теме «Создание программы для организации процесса дистанционного обучения учащихся» ­­– рассчитать затраты на разработку программного продукта и определить экономическую эффективность от его внедрения.

Технико-экономическое обоснование программы удалённого обучения является необходимым шагом для обеспечения эффективного и рационального обучения учащихся данной организации. Оно позволяет определить целесообразность и выгодность внедрения программы, оценить ее экономическую эффективность и сравнить ее с затратами на реализацию проекта.

Процесс обучения долгий и кропотливый труд работников учреждений образования. В связи с ростом заболеваемости и течением времени необходимо внедрять новые технологии прежде всего сделать возможным удалённое обучение учащихся учреждений образования. Оптимизации учёта работ и отметок по ним.

Основными преимуществами данной программы являются:

* возможность удалённо отправлять работы от учащихся;
* возможность удалённо выдавать задания учащимся определённых групп/классов;
* повышение эффективности взаимодействия между учащимся и учительским составом;
* увеличение производительности труда сотрудников;
* повышение репутации учебного заведения.

## 6.2 Составление плана по разработке программного средства

План разработки программного средства представлен в таблице 6.1.

Таблица 6.1 – План разработки программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов и видов работ | Исполнитель (должность, квалификация) | Количество исполнителей | Трудоемкость, человеко-дни |
| 1 | 2 | 3 | 4 |
| Инициация | Техник-программист | 1 | 5 |
| Планирование | Техник-программист | 1 | 7 |
| Разработка | Техник-программист | 1 | 10 |

Продолжение таблицы 6.1 – План разработки программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Реализация и тестирование | Техник-программист | 1 | 15 |
| Мониторинг и завершение проекта | Техник-программист | 1 | 3 |

На рисунке 6.1 отображена трудоёмкость, затраченная на разработку дипломного проекта на разных этапах.

Рисунок 6.1 – Гистограмма распределения времени работы при разработке программного продукта

## 6.3 Определение цены программного средства

Стоимостная оценка программного продукта и определение экономического эффекта у разработчика предполагает расчет затрат, которые включают следующие элементы:

* материальные затраты;
* затраты на оплату труда;
* отчисления на социальные нужды;
* амортизация основных средств и нематериальных активов;
* прочие затраты.

В материальных затратах отражаются затраты на материалы и принадлежности, необходимые для проведения научно-исследовательских работ, магнитные носители, бумагу, красящие ленты и другие материалы, необходимые для разработки программного продукта. Затраты определяются по действующим отпускным ценам. Здесь так же отражаются затраты на электроэнергию.

Расчет осуществляется по формуле (6.1).

, (6.1)

где – коэффициент, учитывающий транспортно-заготовительные расходы Ктзр ≈ от 1,05 до 1,10;

Hpi – норма расхода i-го вида материалов на макет или опытный образец;

Цi – действующая отпускная цена за единицу i-го вида материала, руб.;

Odi – возвратные отходы i-го вида материала (кг, м, и так далее);

Цdi – цена за единицу возвращенных отходов i-го вида материала, руб.;

n – количество применяемых видов материалов.

Расчет целесообразно представить в табличной форме. Результат представлен в таблице 6.2.

Таблица 6.2 – Расчет материальных затрат

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование материалов | Единица измерения | Коли-чество | Цена приобретения руб. | Сумма, руб. |
| Флэш-накопитель | штука | 1 | 13,87 | 13,87 |
| Картридж | штука | 1 | 22,05 | 22,05 |
| Бумага | пачка | 1 | 14,45 | 14,45 |
| Всего расходов | | | | 50,37 |
| Всего с транспортно-заготовительными расходами | | | | 55,41 |

Затраты на электроэнергию находятся исходя из продолжительности периода разработки программного обеспечения, количества кВт/ч, затраченных на его проектирование и тарифа за 1 кВт/ч по следующей формуле (6.2).

, (6.2)

где– стоимость одного кВт/ч, руб.;

– количество кВт/ч.

Тарифы на электроэнергию применяются согласно приложению к Декларации «Об уровне тарифов на электроэнергию, отпускаемую РУП Электроэнергетики ГПО «Белэнерго» для юридических лиц и ИП» на соответствующий период времени, когда разрабатывается программный продукт. Время реализации проекта 40 дней, среднее потребление энергии в месяц составляет 75 кВт/ч, то есть необходимое потребление энергии в дипломном проекте принимается 143 кВт/ч.

Рэ= руб.

Величина затрат исчисляется исходя из численности различных категорий исполнителей и трудоемкости выполнения отдельных видов работ, премиальных систем оплаты труда исполнителей по формуле (6.3).

(6.3)

где – тарифная ставка за день (месячный оклад) i-й категорий работников;

– время фактической работы работника i-й категории по теме, дн. или мес.;

– коэффициент премий по премиальным системам,  от 1,10 до 1,30.

Расчет данных прямых расходов также целесообразно представить в табличной форме. Результат представлен в таблице 8.3. Для этого необходимо рассчитать заработную плату исполнителя, исходя из тарифной ставки 1 разряда, которая в организации составляет 260 руб. Разряд техника программиста 7, тарифный коэффициент – 2,03.

Таблица 6.3 – Расчет затрат на основную заработную плату персонала

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование категорий работников и должностей | Количество штатных единиц, чел. | Заработная плата за 1 месяц, руб. | Трудо-затраты, дн. или мес. | Сумма, руб. |
| Техник-программист | 1 | 527,80 | 2 | 1 055,60 |
| Всего с коэффициентом премий | | | | 1 055,60 |

Определяется дополнительная заработная плата исполнителей, включающая разнообразные предусмотренные трудовым законодательством выплаты, по формуле (6.4).

, (6.4)

где Ндз – норматив дополнительной заработной платы, Ндз ≈ от 10 до 20 %.

Рдз= = 116,12 руб.

Отражается сумма обязательных страховых взносов, взносов на профессиональное пенсионное страхование в бюджет государственного внебюджетного фонда социальной защиты населения Республики Беларусь.

Рассчитываются отчисления органам социальной защиты по формуле (6.5).

, (6.5)

где Нос – норма отчислений на социальную защиту, Нос = 34 %.

Рос= = 434,27 руб.

Также рассчитываются отчисления на страхование от несчастных случаев на производстве и профессиональных заболеваний (Pстр) по ставке действующего законодательства. Для расчетов среднее значение Нбгс принимается равным 0,6 %, отчисления на страхование от несчастных случаев вычисляется по формуле (6.6).

(6.6)

Рстр= = 7,66 руб.

Амортизация основных средств и нематериальных активов рассчитываются в установленном законодательством Республики Беларусь порядке, по одному из методов начисления амортизации (линейный, нелинейный, производительный), исходя из целесообразности его применения.

По статье «Амортизация основных средств и нематериальных активов» рассчитываются амортизационные отчисления (АО), исходя из стоимости основных средств (ОС), используемых в процессе разработки программного обеспечения, сроков эксплуатации оборудования (Тс) и годовой нормы амортизации (На).

Для определения затрат по данному элементу будет использоваться линейный способ начисления амортизации. Нормативные сроки службы оборудования составляет пять лет.

Норма амортизации для линейного способа начисления вычисляется по формуле (6.7).

, (6.7)

где Тс – срок службы оборудования, лет.

.

Для линейного способа начисления амортизационные отчисления равномерно распределены на весь период службы оборудования и вычисляются на один год. Так как разработка программного продукта длилась 40 дней, то сумма амортизационных отчислений (АО) за этот период составит, линейный способ начисления амортизационные вычисляется по формуле (6.8).

, (6.8)

где – стоимость основных средств.

руб.

Прочие затраты на конкретное программное обеспечение включают затраты на: арендную плату; вознаграждения за рационализаторские предложения и выплата авторских гонораров; затраты на гарантийный ремонт и обслуживание изделий; начисленные налоги, сборы (пошлины), платежи, включаемые в затраты на производство продукции (работ, услуг); связь; на оплату услуг рекламы и маркетинга и пр.

В расчетах принимаем условно размер прочих затрат равных от 10 до 30 % от суммы всех остальных затрат на разработку, вычисляется по формуле (6.9).

(6.9)

где ≈ 10 %.

На основании полученных данных по отдельным статьям затрат рассчитывается общая сумма затрат на разработку программного продукта.

Общая сумма затрат по элементам на разработку программного продукта рассчитывается по формуле (6.10).

(6.10)

З=55,41+ 62,79 + 1 161,16 + 116,12 + 434,27 + 7,66+44,73 + 116,12 =

=1 998,26 руб.

Результаты расчетов представлены в таблице 6.4.

Таблица 6.4 – Расчет затрат на разработку программного продукта

|  |  |
| --- | --- |
| Элемент затрат | Затраты, руб. |
| Материальные затраты (Рм) | 55,41 |
| Электроэнергия (Рэ) | 62,79 |
| Затраты на оплату труда (Роз) | 1 161,16 |
| Дополнительная заработная плата (Рдз ) | 116,12 |
| Отчисления органам социальной защиты (Рос) | 434,27 |
| Отчисления на страхования от несчастных случаев (Рстр) | 7,66 |
| Амортизация (РАо) | 44,73 |
| Прочие затраты (Рпр) | 116,12 |
| Общая сумма затрат (З) | 1 998,26 |

Таблицу 6.4 удобно представить в виде диаграммы, отображающей элементы затрат программного продукта. Диаграмма представлена на рисунке 6.2.

Рисунок 6.2 – Структура затрат

Исходя из расчётов затрат на разработку программного продукта, можно сделать вывод, что самыми затратными являются выплаты зарплаты сотруднику и отчисление органам социальной защиты. Также не мало средств уходит на прочие затраты и выплату дополнительной заработной платы сотруднику. Незначительное количество средств уходит на электроэнергию, материальные расходы и амортизацию.

## 6.4 Экономическая эффективность разработки программного продукта

Заказчик оплачивает разработчику всю сумму расходов по проекту, включая прибыль. После уплаты налогов из прибыли в распоряжении заказчика остается чистая прибыль от проекта. Ввиду того, что программное обеспечение разрабатывается для одного объекта, чистую прибыль можно считать в качестве экономического эффекта разработчика от реализованного программного продукта.

В дипломном проекте отпускная цена программного продукта, представляет собой не цену за единицу продукции, а цену проекта, за которую его можно продать и получить определённую выгоду.

Отпускная цена продукции формируется исходя из плановой себестоимости производства продукции, всех видов установленных налогов и прибыли, а также качества, потребительских свойств продукции и конъюнктуры рынка.

С учетом действующих в республике нормативных документов отпускная цена на продукцию рассчитывается по формуле (6.11).

(6.11)

где ОЦ– отпускная цена разработчика, руб.;

З– затраты на разработку, руб.;

П – прибыль, руб.

ОЦ = 1 998,26+ =2203,38 руб.

Прибыль рассчитывается по следующей формуле (6.12).

, (6.12)

гдеR–уровень рентабельности (от 10 до 30 %).

= руб.

Стоимость проекта с учётом НДС, представляет собой сумму отпускной цены и налога на добавленную стоимость, рассчитывается по формуле (6.13).

, (6.13)

где СтавкаНДС – 20 %.

руб.

Отпускная цена с учетом НДС, рассчитывается по формуле (6.14).

(6.14)

2643,00 руб.

Таким образом, разработчик программного продукта может продать заказчику программный продукт, что покроет затраты и обеспечит прибыль за разработку проекта.

## 6.5 Экономическая эффективность у пользователя программного продукта

Экономический эффект у пользователя программного продукта выражается в виде экономии трудовых, материальных и финансовых ресурсов, получаемой от таких показателей как:

* повышения уровня сервиса и репутации учебного заведения.
* улучшения показателей основной деятельности предприятия в результате использования программного продукта и т.д.

Исходя из полученных результатов, можно сделать вывод, что данный проект позволит значительно упростить работу преподавателям. Отпускная цена готового программного средства составила 2643,00 руб., следовательно, проект оказался, хоть и затратным, но весьма полезным и перспективным, повышая репутацию учебного заведения.

**Заключение**

В рамках дипломного проекта на тему «Создание программы для организации процесса дистанционного обучения учащихся» было разработано программное средство «UnilifeClassesRoomsDiplomDesktop.exe».

Для достижения цели дипломного проекта в программе реализовано:

* разграничение прав доступа;
* создание и сопровождение классов администратором;
* сопровождение пользователей администратором;
* выставление отметок за выполненные задания;
* автоматическая рассылка сообщений на электронную почту о новых заданиях в классах и отметках за задания;
* формирование данных по успеваемости учащихся в классе;
* добавление файлов работ и отправки работы на проверку;
* поиск и фильтрация данных по различным критериям;
* наполнение классов заданиями, преподавателем.

Разработка имеет интуитивно понятный графический интерфейс, позволяющий даже с минимальным знанием использования персонального компьютера достичь максимум производительности в работе.

Программное средство реализовано в полном объеме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.

Программное средство готово к практическому использованию и может быть дополнено и модернизировано.

В процессе разработки программного средства использовался в большом объеме материал по базам данных и системами управления базами данных, что способствовало закреплению наработанных навыков и умений в этих областях знаний. При разработке приложения наибольшее внимание уделялось максимальному созданию лаконичного интерфейса.

Для разработки программного средства был использован язык программирования C#, ядро реляционной базы данных SQL Server Express было использовано для создания и разворачивания базы данных «UnilifeClassesRoomsDiplomDesktop».

Список информационных источников

1. Багласова, Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г. Багласова. – Минск : КБП, 2022. – 30 c.
2. Бондарь, А.Г. Microsoft SQL Server 2012 / А.Г. Бондарь. – СПб. : БХВ-Петербург, 2013. – 608 с.
3. Мацяшек, Лешек А. Анализ и проектирование информационных систем с помощью UML 2.0 / Лешек А. Мацяшек. – 3-е изд. – М. : ООО «И.Д. Вильямс», 2008. – 816 с.
4. Митин, А.И. Работа с базами данных Microsoft SQL Server: сценарии практических занятий / А. И. Митин. – М. : Директ-Медиа, 2020. – 142 с
5. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
6. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.
7. Текст программы. Требования к содержанию, оформлению и контролю качества: ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.
8. Обзор обновлений и новых функций Windows 10 [Электронный ресурс]. – Microsoft, 2022 – Режим доступа: https://support.microsoft.com/ru-ru/windows#WindowsVersion=Windows\_10. – Дата доступа: 25.04.2023.
9. Обзор C# [Электронный ресурс]. – Microsoft, 2023 – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/. – Дата доступа: 25.04.2023
10. Draw.io – бесплатное средство для создания блок-схем, инфографики, прототипов [Электронный ресурс]. – EL-BLOG.RU, 2023 – Режим доступа: https://el-blog.ru/draw-io/. – Дата доступа: 25.04.2023.
11. Dr.Explain – приложение для быстрого создания файлов справки [Электронный ресурс]. – Режим доступа: https://help-spravka.ru/blog/13-hat-programs/49-dr-explain-otechestvennaya-programma-dlya-razrabotki-spravki – Дата доступа 25.04.2023.
12. Microsoft Office [Электронный ресурс]. – Wikipedia, 2023 – Режим доступа : https://ru.wikipedia.org/wiki/Microsoft\_Office. – Дата доступа : 13.04.2023.
13. Visual Studio 2022 [Электронный ресурс]. – Microsoft, 2023. – Режим доступа: https://visualstudio.microsoft.com/ru/vs. – Дата доступа 25.04.2023.
14. Периодический контроль [Электронный ресурс] – сop. Охрана труда.by 2019 – Режим доступа: http://www.ohrana-truda.by/topic/2515-periodicheskij-kontrol-za-soblyudeniem-zakonod. – Дата доступа 25.04.2023.

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Текст программных модулей**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

using UnilifeClassesRoomsDiplomServerDLL.ModelsDB;

namespace UnilifeClassesRoomsDiplomServerDLL

{

// ПРИМЕЧАНИЕ. Можно использовать команду "Переименовать" в меню "Рефакторинг", чтобы изменить имя интерфейса "IUnilifeClassesRoomsDiplomServerDDL" в коде и файле конфигурации.

[ServiceContract]

public interface IUnilifeClassesRoomsDiplomServerDDL//Интерфейс для службы WCF(через него осуществляеться взаимодействие клиента с сервером

{

[OperationContract]

string LogInAccount(string login, string password);//OperationContract проверки данных для входа и отправки кода подтверждения

[OperationContract]

bool ConfirmLogInAccount(string hashKey, string mailkey);//OperationContract проверки кода подтверждения

[OperationContract]

bool CheakHashKey(string hashKey);//OperationContract проверке хеш-ключа для доступа к приложению

[OperationContract]

Account GetAccountUser(string hashKey);//OperationContract получения аккаунта пользователя

[OperationContract]

User GetUser(int? id);//OperationContract получения пользователя по id

[OperationContract]

List<Account> GetAccounts();//OperationContract получения списка аккаунтов

[OperationContract]

List<User> GetUsers();//OperationContract получения списка пользователей

[OperationContract]

List<Role> GetRoles();//OperationContract получения списка ролей

[OperationContract]

bool LogOut(string hashKey);//OperationContract для выхода из приложения и удаления ключа сессии

[OperationContract]

void UpdAccount(Account acc);//OperationContract для обновление данных аккаунта хранящтихся в БД

[OperationContract]

int GetPowerRole(string hashKey);//OperationContract для получения силы роли аккаунта

[OperationContract]

Role GetRole(int? id);//OperationContract для получения роли по id

[OperationContract]

void AddAccount(Account acc);//OperationContract для добавления аккаунта в БД

[OperationContract]

List<Class> GetClassesUser(string hashKey);//OperationContract для получения списка классов,пользователя

[OperationContract]

List<ModelsDB.Task> GetTasksClassFalse(int classId, string hashKey);//OperationContract получение заданий не выполненных пользователем, срок которых не истёк

[OperationContract]

List<ModelsDB.Task> GetTasksClass(int classId);//OperationContract получения списка заданий соддержащегося в классе

[OperationContract]

bool СheckingTeacher(int classId, string hashKey);//OperationContract проверки на учителя

[OperationContract]

List<Account> GetAccountsClass(int classId);//OperationContract получения списка аккаунтов привязанных к классу

[OperationContract]

void AddTask(ModelsDB.Task \_task, string hashKey);//OperationContract добавления задания

[OperationContract]

Job GetJobUser(int userId, int taskId);//OperationContract получения данных о работе пользователя для формирования отчета

[OperationContract]

List<Division> GetDivisions();//OperationContract получения списка подразделений

[OperationContract]

void AddDivision(Division division);//OperationContract добавления подразделения

[OperationContract]

void UpdDivision(Division division); //OperationContract изменения подразделения

[OperationContract]

int AddMessage(MessagesTask messaage, string hashKey);//OperationContract добавления сообщения в задании

[OperationContract]

List<MessagesTask> GetMessages(int taskId);//OperationContract получения списка сообщений в задании

[OperationContract]

List<User> GetUsersClass(int classId);//OperationContract получения списка пользователей класса

[OperationContract]

Account GetAccount(int accountId);//OperationContract получения аккаунта по id

[OperationContract]

List<FilesTask> GetFileTask(int taskId); //OperationContract получения файлов задания

[OperationContract]

List<LinksTask> GetLinksTask(int taskId); //OperationContract получение ссылок задания

[OperationContract]

List<FilesJob> GetFilesJob(int jobId);//OperationContract получения файлов работы

[OperationContract]

List<Post> GetPosts(); //OperationContract получения списка должностей

[OperationContract]

List<Session> GetSessionsAccount(int accountId);//OperationContract получение списка сессий для аккаунта

[OperationContract]

void AddPost(Post post); //OperationContract добавления должности

[OperationContract]

void UpdPost(Post post);//OperationContract изменения должности

[OperationContract]

void DelSession(Session session);//OperationContract удаления сессии

[OperationContract]

void UpdJob(Job job);//OperationContract изменение данных работы учащегося

[OperationContract]

void AddUser(User user);//OperationContract добавления пользователя

[OperationContract]

void UpdUser(User user);//OperationContract изменения пользователя

[OperationContract]

bool AddUserToClass(string hashKey, string key);//OperationContract добавления пользователся в класс по ключу класса

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

using UnilifeClassesRoomsDiplomServerDLL.ModelsDB;

namespace UnilifeClassesRoomsDiplomServerDLL

{

// ПРИМЕЧАНИЕ. Можно использовать команду "Переименовать" в меню "Рефакторинг", чтобы изменить имя интерфейса "IUnilifeClassesRoomsDiplomServerDDL" в коде и файле конфигурации.

[ServiceContract]

public interface IUnilifeClassesRoomsDiplomServerDDL//Интерфейс для службы WCF(через него осуществляеться взаимодействие клиента с сервером

{

[OperationContract]

string LogInAccount(string login, string password);//OperationContract проверки данных для входа и отправки кода подтверждения

[OperationContract]

bool ConfirmLogInAccount(string hashKey, string mailkey);//OperationContract проверки кода подтверждения

[OperationContract]

bool CheakHashKey(string hashKey);//OperationContract проверке хеш-ключа для доступа к приложению

[OperationContract]

Account GetAccountUser(string hashKey);//OperationContract получения аккаунта пользователя

[OperationContract]

User GetUser(int? id);//OperationContract получения пользователя по id

[OperationContract]

List<Account> GetAccounts();//OperationContract получения списка аккаунтов

[OperationContract]

List<User> GetUsers();//OperationContract получения списка пользователей

[OperationContract]

List<Role> GetRoles();//OperationContract получения списка ролей

[OperationContract]

bool LogOut(string hashKey);//OperationContract для выхода из приложения и удаления ключа сессии

[OperationContract]

void UpdAccount(Account acc);//OperationContract для обновление данных аккаунта хранящтихся в БД

[OperationContract]

int GetPowerRole(string hashKey);//OperationContract для получения силы роли аккаунта

[OperationContract]

Role GetRole(int? id);//OperationContract для получения роли по id

[OperationContract]

void AddAccount(Account acc);//OperationContract для добавления аккаунта в БД

[OperationContract]

List<Class> GetClassesUser(string hashKey);//OperationContract для получения списка классов,пользователя

[OperationContract]

List<ModelsDB.Task> GetTasksClassFalse(int classId, string hashKey);//OperationContract получение заданий не выполненных пользователем, срок которых не истёк

[OperationContract]

List<ModelsDB.Task> GetTasksClass(int classId);//OperationContract получения списка заданий соддержащегося в классе

[OperationContract]

bool СheckingTeacher(int classId, string hashKey);//OperationContract проверки на учителя

[OperationContract]

List<Account> GetAccountsClass(int classId);//OperationContract получения списка аккаунтов привязанных к классу

[OperationContract]

void AddTask(ModelsDB.Task \_task, string hashKey);//OperationContract добавления задания

[OperationContract]

Job GetJobUser(int userId, int taskId);//OperationContract получения данных о работе пользователя для формирования отчета

[OperationContract]

List<Division> GetDivisions();//OperationContract получения списка подразделений

[OperationContract]

void AddDivision(Division division);//OperationContract добавления подразделения

[OperationContract]

void UpdDivision(Division division); //OperationContract изменения подразделения

[OperationContract]

int AddMessage(MessagesTask messaage, string hashKey);//OperationContract добавления сообщения в задании

[OperationContract]

List<MessagesTask> GetMessages(int taskId);//OperationContract получения списка сообщений в задании

[OperationContract]

List<User> GetUsersClass(int classId);//OperationContract получения списка пользователей класса

[OperationContract]

Account GetAccount(int accountId);//OperationContract получения аккаунта по id

[OperationContract]

List<FilesTask> GetFileTask(int taskId); //OperationContract получения файлов задания

[OperationContract]

List<LinksTask> GetLinksTask(int taskId); //OperationContract получение ссылок задания

[OperationContract]

List<FilesJob> GetFilesJob(int jobId);//OperationContract получения файлов работы

[OperationContract]

List<Post> GetPosts(); //OperationContract получения списка должностей

[OperationContract]

List<Session> GetSessionsAccount(int accountId);//OperationContract получение списка сессий для аккаунта

[OperationContract]

void AddPost(Post post); //OperationContract добавления должности

[OperationContract]

void UpdPost(Post post);//OperationContract изменения должности

[OperationContract]

void DelSession(Session session);//OperationContract удаления сессии

[OperationContract]

void UpdJob(Job job);//OperationContract изменение данных работы учащегося

[OperationContract]

void AddUser(User user);//OperationContract добавления пользователя

[OperationContract]

void UpdUser(User user);//OperationContract изменения пользователя

[OperationContract]

bool AddUserToClass(string hashKey, string key);//OperationContract добавления пользователся в класс по ключу класса

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Mail;

using System.Net;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

using System.Data.SqlClient;

using System.Runtime.Remoting.Contexts;

using UnilifeClassesRoomsDiplomServerDLL.ModelsDB;

using System.Threading.Tasks;

using System.Runtime;

using System.Security.Cryptography;

using System.Security.Principal;

namespace UnilifeClassesRoomsDiplomServerDLL

{

// ПРИМЕЧАНИЕ. Команду "Переименовать" в меню "Рефакторинг" можно использовать для одновременного изменения имени класса "UnilifeClassesRoomsDiplomServerDDL" в коде и файле конфигурации.

public class UnilifeClassesRoomsDiplomServerDDL : IUnilifeClassesRoomsDiplomServerDDL//Реализация OperationContract-ов

{

UnilifeDB \_db = new UnilifeDB();

public string LogInAccount(string \_login, string \_password)

{

var res = (from a in \_db.Accounts

where a.Password == \_password && a.Login == \_login && a.Active == true

select a).FirstOrDefault();

if (res != null)

{

res.MailKey = Generator().ToString();

// отправитель - устанавливаем адрес и отображаемое в письме имя

MailAddress from = new MailAddress("unilife2023@yandex.ru", "Unilife\_ClassesRoom");

// кому отправляем

MailAddress to = new MailAddress(res.Mail.ToString());//это адрес на который отправляем

// создаем объект сообщения

MailMessage m = new MailMessage(from, to);

// тема письма

m.Subject = "Код подтверждения";

// текст письма

m.Body = "Код подтверждения для входа в Unilife\_ClassesRoom " + res.MailKey;

// письмо представляет код html

m.IsBodyHtml = true;

// адрес smtp-сервера и порт, с которого будем отправлять письмо

SmtpClient smtp = new SmtpClient("smtp.yandex.ru", 587);

// логин и пароль

smtp.Credentials = new NetworkCredential("unilife2023@yandex.ru", "ynunavzspenewerw");

smtp.EnableSsl = true;

smtp.Send(m);

Session s = new Session();

s.SessionKey = Generator().ToString();

s.AccountId = res.Id;

\_db.Sessions.Add(s);

\_db.SaveChanges();

return s.SessionKey.ToString();

}

else

{

return "0";

}

}

public bool ConfirmLogInAccount(string hashKey, string mailkey)

{

var res1 = (from a in \_db.Accounts

join s in \_db.Sessions on a.Id equals s.AccountId

where a.MailKey == mailkey && s.SessionKey == hashKey

select s).FirstOrDefault();//Поиск кода подтверждения

if (res1 != null)//Если код найден статус сессии меняеться на пройденная валидацию.

{

res1.Confirm = true;

\_db.Entry(res1).State = System.Data.Entity.EntityState.Modified;

\_db.SaveChanges();

return true;

}

else// возвращаеться false. Пользователь остаёться на форме подтверждения ключа.

{

return false;

}

}

public List<Account> GetAccounts()

{

\_db.Configuration.ProxyCreationEnabled = false;// отключение проксирования сущностей EF.(Без отключения не работает сервер по причине тайм-аутов из-за подгрузки связанных данных и проблемм с сереализацией виртуальных объектов)

return \_db.Accounts.ToList();// возвращение списка аккаунтов

}

public List<User> GetUsers()

{

\_db.Configuration.ProxyCreationEnabled = false;

return \_db.Users.ToList();// возвращение списка пользователей

}

public List<Role> GetRoles()

{

\_db.Configuration.ProxyCreationEnabled = false;

return \_db.Roles.ToList();// возвращение списка ролей

}

public bool CheakHashKey(string hashKey)

{

var res = (from s in \_db.Sessions

where s.SessionKey == hashKey && s.Confirm == true

select s).FirstOrDefault();//поиск хеш-ключа в базе

if (res != null)//Если найден отправляем true.

{

return true;

}

else

{

return false;

}

}

public Account GetAccountUser(string hashKey)

{

\_db.Configuration.ProxyCreationEnabled = false;

Account acc = (from a in \_db.Accounts

join s in \_db.Sessions on a.Id equals s.AccountId

where s.SessionKey == hashKey

select a).FirstOrDefault();//получение списка аккаунтов связанных с классом

return acc;

}

public bool LogOut(string hashKey)

{

var res = (from s in \_db.Sessions

where s.SessionKey == hashKey

select s).FirstOrDefault();//Перепроверка существования хэш-ключа и получение сессии

if (res != null)

{

\_db.Sessions.Remove(res);//Удаление сессии

return true;

}

return false;

}

public User GetUser(int? id)

{

\_db.Configuration.ProxyCreationEnabled = false;

User res = (from s in \_db.Users

where s.Id == id

select s).FirstOrDefault();//Получение пользователя по id

return res;

}

public void UpdAccount(Account acc)

{

\_db.Entry(acc).State = System.Data.Entity.EntityState.Modified;//изменение данных аккаунта

\_db.SaveChanges();//сохранение изменений в бд

}

public int GetPowerRole(string hashKey)

{

int power = (from r in \_db.Roles

join a in \_db.Accounts on r.Id equals a.RoleId

join s in \_db.Sessions on a.Id equals s.AccountId

where s.SessionKey == hashKey

select r.PowerRole).FirstOrDefault();// получение силы роли пользователя

return power;

}

public Role GetRole(int? id)

{

\_db.Configuration.ProxyCreationEnabled = false;

Role res = (from s in \_db.Roles

where s.Id == id

select s).FirstOrDefault();//Получение роли по id

return res;

}

public void AddAccount(Account acc)

{

\_db.Accounts.Add(acc);//Добавление аккаунта в БД

\_db.SaveChanges();

}

public List<Class> GetClassesUser(string hashKey)

{

\_db.Configuration.ProxyCreationEnabled = false;

List<Class> res = (from c in \_db.Classes

join ca in \_db.ClassAccounts on c.Id equals ca.ClassId

join a in \_db.Accounts on ca.AccountId equals a.Id

join s in \_db.Sessions on a.Id equals s.AccountId

where s.SessionKey == hashKey

select c).ToList();//Получение списка классов пользователя

return res;

}

public List<ModelsDB.Task> GetTasksClassFalse(int classId, string hashKey)

{

\_db.Configuration.ProxyCreationEnabled = false;

List<ModelsDB.Task> res = (from t in \_db.Tasks

join c in \_db.Classes on t.ClassId equals c.Id

join j in \_db.Jobs on t.Id equals j.TaskId

join a in \_db.Accounts on j.AccountId equals a.Id

join s in \_db.Sessions on a.Id equals s.AccountId

where c.Id == classId && j.Deleted == false && t.DelivaryTime > DateTime.Now && s.SessionKey == hashKey

select t).ToList();// Получение списказаданий где выполненны или сданны на проверку работы. Данным пользователем.(дата сдачи которых не истекла)

List<ModelsDB.Task> res2 = (from t in \_db.Tasks

join c in \_db.Classes on t.ClassId equals c.Id

where c.Id == classId && t.DelivaryTime > DateTime.Now

select t).ToList();//Получение списка заданий класса(дата сдачи которых не истекла)

return res2.Except(res).ToList(); //Возвращение списка не сданных/выполненных работ пользователя(дата сдачи которых не истекла)

}

public List<ModelsDB.Task> GetTasksClass(int classId)

{

\_db.Configuration.ProxyCreationEnabled = false;

List<ModelsDB.Task> res = (from t in \_db.Tasks

join c in \_db.Classes on t.ClassId equals c.Id

where c.Id == classId

select t).ToList();// Получение списка заданий в классе

return res;

}

public bool СheckingTeacher(int classId, string hashKey)

{

\_db.Configuration.ProxyCreationEnabled = false;

var res = (from a in \_db.Accounts

join ca in \_db.ClassAccounts on a.Id equals ca.AccountId

join s in \_db.Sessions on a.Id equals s.AccountId

where s.SessionKey == hashKey && ca.ClassId == classId

select ca.Teacher).FirstOrDefault();//проверка являеться ли пользователь учителем в классе

return res;

}

public List<Account> GetAccountsClass(int classId)

{

\_db.Configuration.ProxyCreationEnabled = false;

var res1 = from a in \_db.Accounts

join j in \_db.ClassAccounts on a.Id equals j.AccountId

where j.ClassId == classId

select a;// Получения списка аккаунтов связанных с классом

return res1.ToList();

}

public void AddTask(ModelsDB.Task \_task, string hashKey)

{

using (UnilifeDB db = new UnilifeDB())

{

\_task.AccountId = (from a in db.Accounts

join s in db.Sessions on a.Id equals s.AccountId

where s.SessionKey == hashKey

select a.Id).FirstOrDefault();

db.Tasks.Add(\_task);

db.SaveChanges();

var listMail = from a in db.Accounts

join ca in db.ClassAccounts on a.Id equals ca.AccountId

where ca.ClassId == \_task.ClassId && ca.Teacher == false

select a.Mail;

string className = (from c in db.Classes

where c.Id == \_task.ClassId

select c.Name).FirstOrDefault();

// отправитель - устанавливаем адрес и отображаемое в письме имя

MailAddress from = new MailAddress("unilife2023@yandex.ru", "Unilife\_ClassRoom");

// кому отправляем

foreach (var l in listMail)

{

MailAddress to = new MailAddress(l.ToString());//это адрес на который отправляем

MailMessage m = new MailMessage(from, to);

// тема письма

m.Subject = "Новое задание";

// текст письма

m.Body = $"Новое задание в классе {className}. {\_task.Name}";

// письмо представляет код html

m.IsBodyHtml = true;

// адрес smtp-сервера и порт, с которого будем отправлять письмо

SmtpClient smtp = new SmtpClient("smtp.yandex.ru", 587);

// логин и пароль

smtp.Credentials = new NetworkCredential("unilife2023@yandex.ru", "ynunavzspenewerw");

smtp.EnableSsl = true; // создаем объект сообщения

smtp.Send(m);

}

}

}

public Job GetJobUser(int userId, int taskId)

{

\_db.Configuration.ProxyCreationEnabled = false;

Job job = (from j in \_db.Jobs

where j.TaskId == taskId && j.Deleted == false && j.Account.User.Id == userId

select j).FirstOrDefault();//Получение работы пользователя

return job;

}

public List<Division> GetDivisions()

{

\_db.Configuration.ProxyCreationEnabled = false;

return \_db.Divisions.ToList();//Возвращение списка подразделений

}

public void AddDivision(Division division)

{

\_db.Divisions.Add(division);//Добавление подразделения в БД

\_db.SaveChanges();

}

public void UpdDivision(Division division)

{

\_db.Entry(division).State = System.Data.Entity.EntityState.Modified;//Изменения подразделения в БД

\_db.SaveChanges();

}

public int AddMessage(MessagesTask messaage, string hashKey)

{

\_db.Configuration.ProxyCreationEnabled = false;

messaage.AccountId = (from a in \_db.Accounts

join s in \_db.Sessions on a.Id equals s.AccountId

where s.SessionKey == hashKey

select a.Id).FirstOrDefault();

\_db.MessagesTasks.Add(messaage);//Добавление сообщения в БД

\_db.SaveChanges();

return messaage.AccountId;

}

public List<MessagesTask> GetMessages(int taskId)

{

\_db.Configuration.ProxyCreationEnabled = false;

var res = (from m in \_db.MessagesTasks

where m.TaskId == taskId

select m).ToList();//Получения списка сообщений, задания

return res;

}

public List<User> GetUsersClass(int classId)

{

\_db.Configuration.ProxyCreationEnabled = false;

List<User> Users = new List<User>((from u in \_db.Users

join a in \_db.Accounts on u.Id equals a.UserId

join ca in \_db.ClassAccounts on a.Id equals ca.AccountId

join c in \_db.Classes on ca.ClassId equals c.Id

where ca.Teacher != true && c.Id == classId

select u).ToList());//Получение списка пользователей класса

return Users;

}

public Account GetAccount(int accountId)

{

\_db.Configuration.ProxyCreationEnabled = false;

Account acc = (from a in \_db.Accounts

where a.Id == accountId

select a).FirstOrDefault();//Получение аккаунта по id

return acc;

}

public List<FilesTask> GetFileTask(int taskId)

{

\_db.Configuration.ProxyCreationEnabled = false;

var res = from f in \_db.FilesTasks

where f.TaskId == taskId

select f;//Получение списка файлов задания по id задания.

return res.ToList();

}

public List<LinksTask> GetLinksTask(int taskId)

{

\_db.Configuration.ProxyCreationEnabled = false;

var res = from f in \_db.LinksTasks

where f.TaskId == taskId

select f;//Получение списка ссылок задания по id задания.

return res.ToList();

}

public List<FilesJob> GetFilesJob(int jobId)

{

\_db.Configuration.ProxyCreationEnabled = false;

var res = from f in \_db.FilesJobs

where f.JobId == jobId

select f;//Получение списка файлов работ по id раблоты.

return res.ToList();

}

public void UpdJob(Job job)

{

\_db.Entry(job).State = System.Data.Entity.EntityState.Modified;

\_db.SaveChanges();

ModelsDB.Task task = (from j in \_db.Jobs

join t in \_db.Tasks on j.TaskId equals t.Id

where j.Id == job.Id

select t).FirstOrDefault();

Account acc = (from j in \_db.Jobs

join a in \_db.Accounts on j.AccountId equals a.Id

where j.Id == job.Id

select a).FirstOrDefault();

MailAddress from = new MailAddress("unilife2023@yandex.ru", "Unilife\_ClassRoom");

MailAddress to = new MailAddress(acc.Mail.ToString());//это адрес на который отправляем

MailMessage m = new MailMessage(from, to);

// тема письма

m.Subject = $"Отметка за работу. Задание {task.Name}";

// текст письма

m.Body = $"Вам {job.Score} за работу по заданию {task.Name}";

// письмо представляет код html

m.IsBodyHtml = true;

// адрес smtp-сервера и порт, с которого будем отправлять письмо

SmtpClient smtp = new SmtpClient("smtp.yandex.ru", 587);

// логин и пароль

smtp.Credentials = new NetworkCredential("unilife2023@yandex.ru", "ynunavzspenewerw");

smtp.EnableSsl = true; // создаем объект сообщения

smtp.Send(m);

}

public List<Post> GetPosts()

{

\_db.Configuration.ProxyCreationEnabled = false;

return \_db.Posts.ToList();//Получение списка должностей.

}

public void AddPost(Post post)

{

\_db.Posts.Add(post);//Добавление должности

\_db.SaveChanges();

}

public void UpdPost(Post post)

{

\_db.Entry(post).State = System.Data.Entity.EntityState.Modified;//Изменение должности

\_db.SaveChanges();

}

public List<Session> GetSessionsAccount(int accountId)

{

\_db.Configuration.ProxyCreationEnabled = false;

var res = from s in \_db.Sessions

where s.AccountId == accountId

select s;//Получение списка сессий для аккаунта

return res.ToList();

}

public void DelSession(Session session)

{

Session ses = (from s in \_db.Sessions

where s.SessionKey == session.SessionKey

select s).FirstOrDefault();

\_db.Sessions.Remove(ses);// удаление сессии

\_db.SaveChanges();

}

public void AddUser(User user)

{

\_db.Users.Add(user);//Добавление пользователя

\_db.SaveChanges();

}

public void UpdUser(User user)

{

\_db.Entry(user).State = System.Data.Entity.EntityState.Modified;//изменение пользователя

\_db.SaveChanges();

}

public bool AddUserToClass(string hashKey, string key)

{

var resClass = (from c in \_db.Classes

where c.KeyClass == key

select c).FirstOrDefault();//Получение класса

if (resClass != null)

{

Account resAccount = (from a in \_db.Accounts

join s in \_db.Sessions on a.Id equals s.AccountId

where s.SessionKey == hashKey

select a).Single();//Получение аккаунта

var check = from c in \_db.ClassAccounts

where c.Account.Id == resAccount.Id && c.ClassId == resClass.Id

select c;//Проверка есть ли данный аккаунт в данном классе

if (check.Count() == 0)//если нет, то добавляеться связь между аккаунтом и классом

{

ClassAccount classAccount = new ClassAccount();

classAccount.Class = resClass;

classAccount.Account = resAccount;

classAccount.Teacher = false;

\_db.ClassAccounts.Add(classAccount);

\_db.SaveChanges();

return true;

}

}

return false;

}

StringBuilder Generator()// Генератор кодов

{

StringBuilder \_key = new StringBuilder();

Random random = new Random();

var rand = new Random();

int y;

for (int i = 0; i < 24; i++)

{

y = rand.Next(1, 3);

if (y == 1)

{

\_key.Append(Convert.ToChar(rand.Next(97, 124)));

}

else

{

\_key.Append(rand.Next(1, 10));

}

}

return \_key;

}

}

}

**ПРИЛОЖЕНИЕ Б**

**(справочное)**

**Результаты работы приложения**

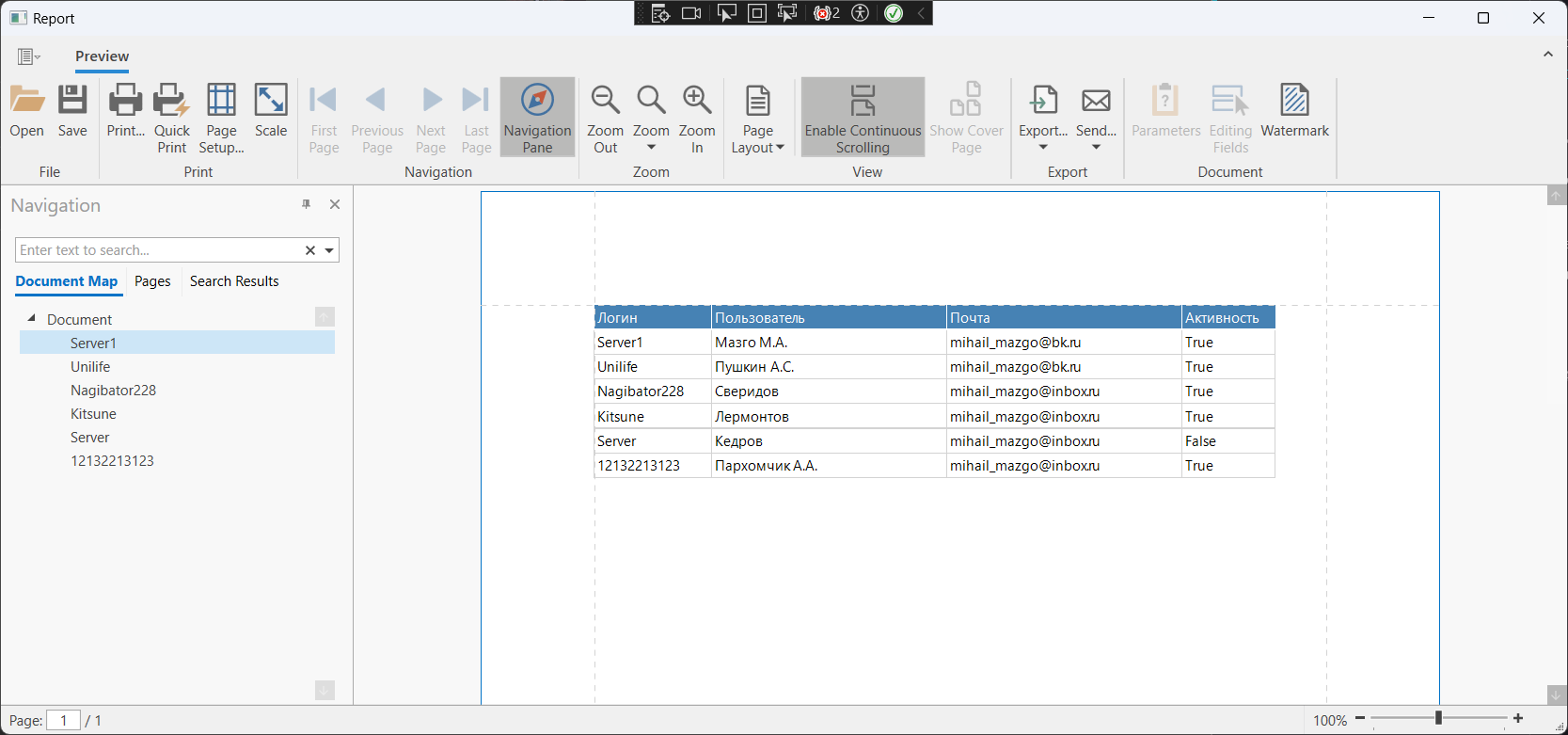


Рисунок Б.1

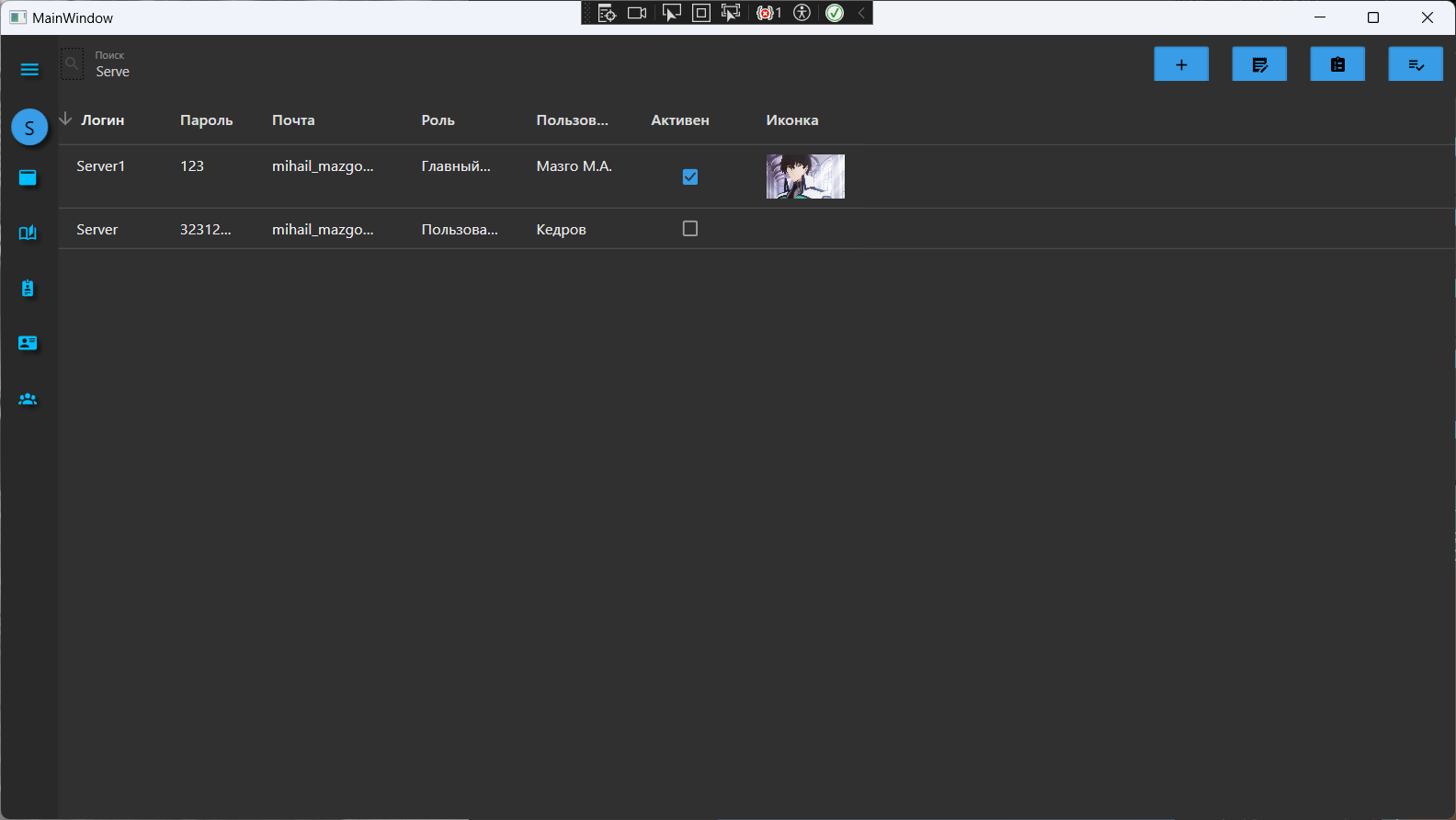


Рисунок Б.2

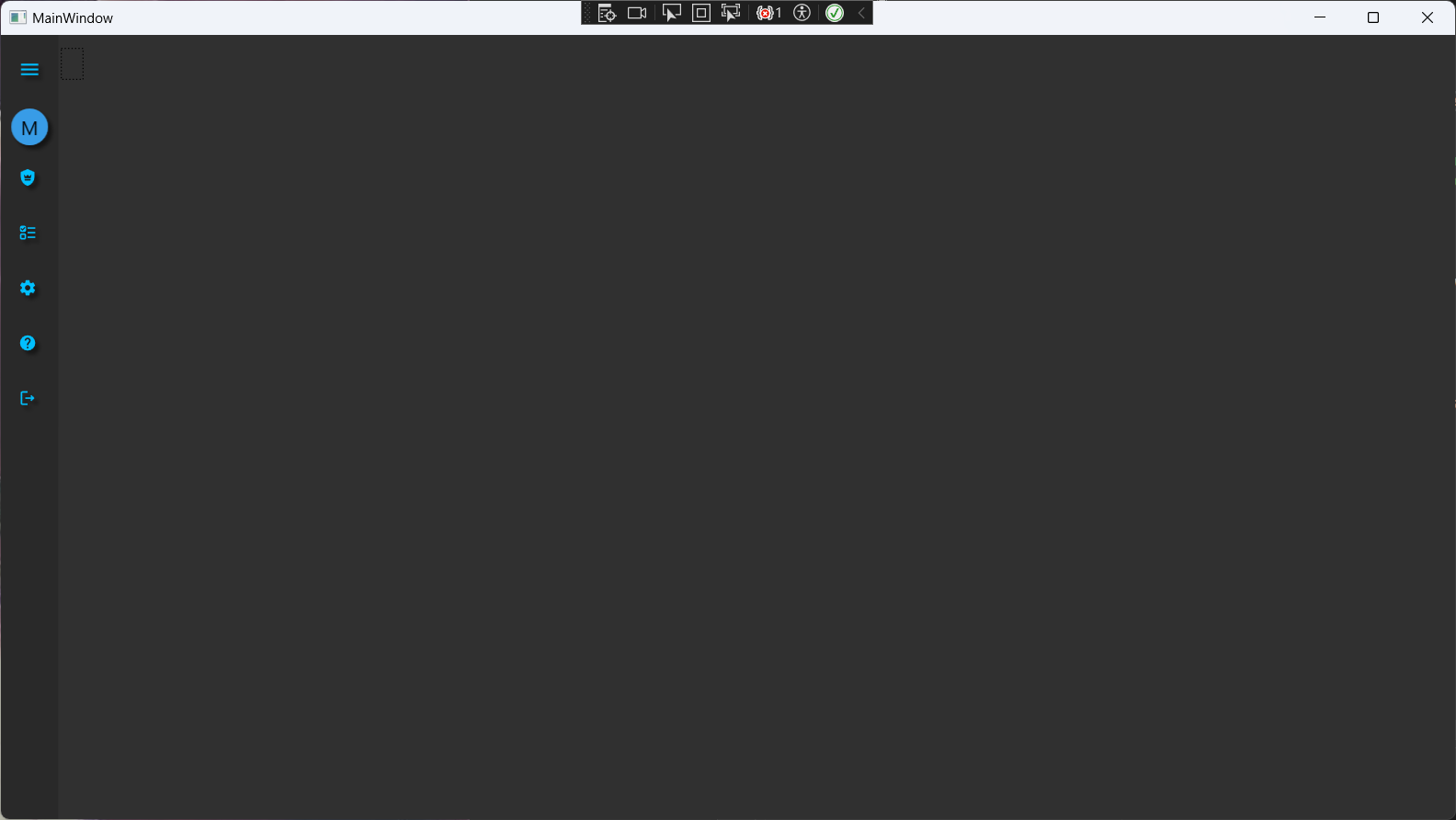


Рисунок Б.3

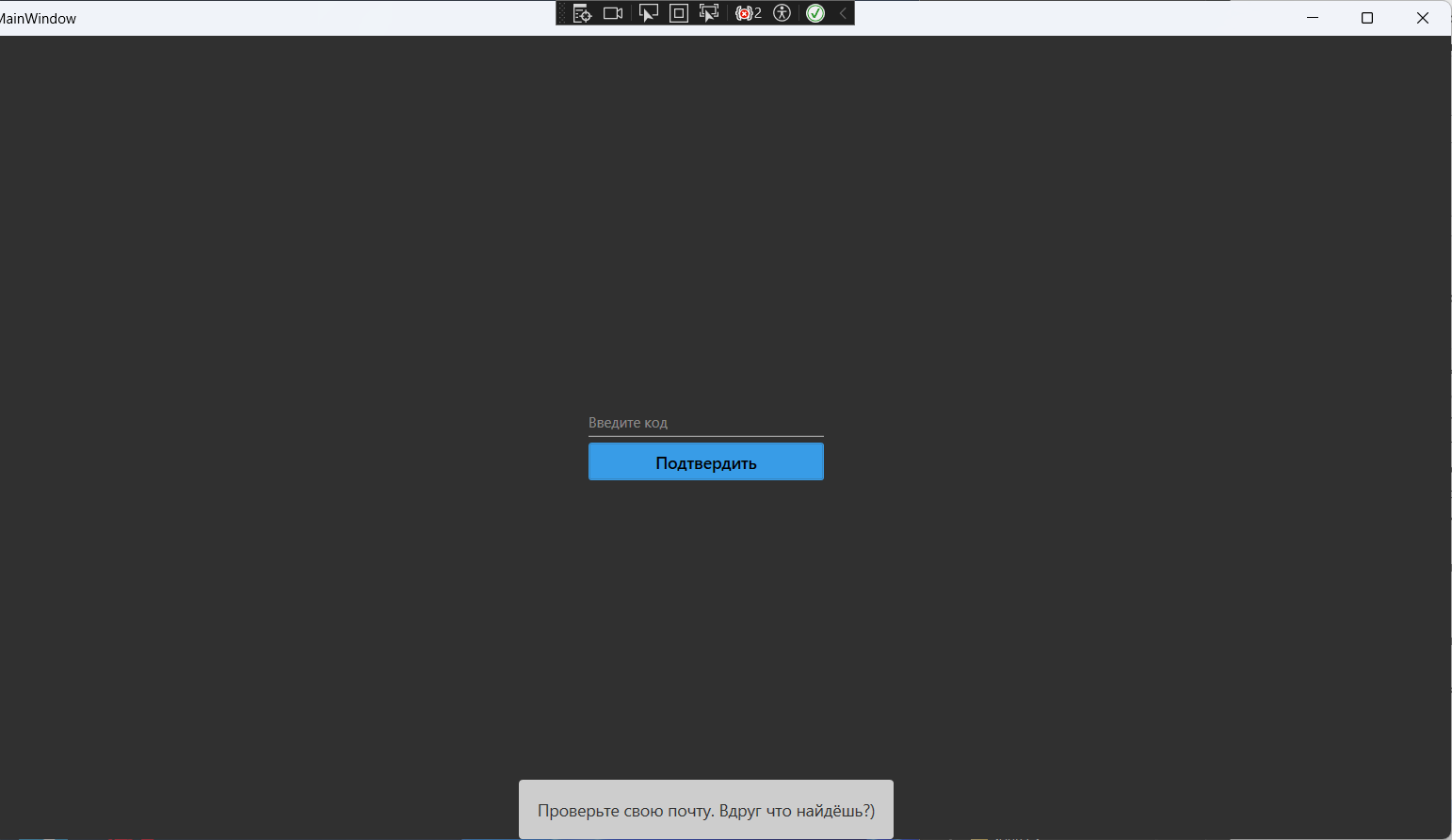


Рисунок Б.4

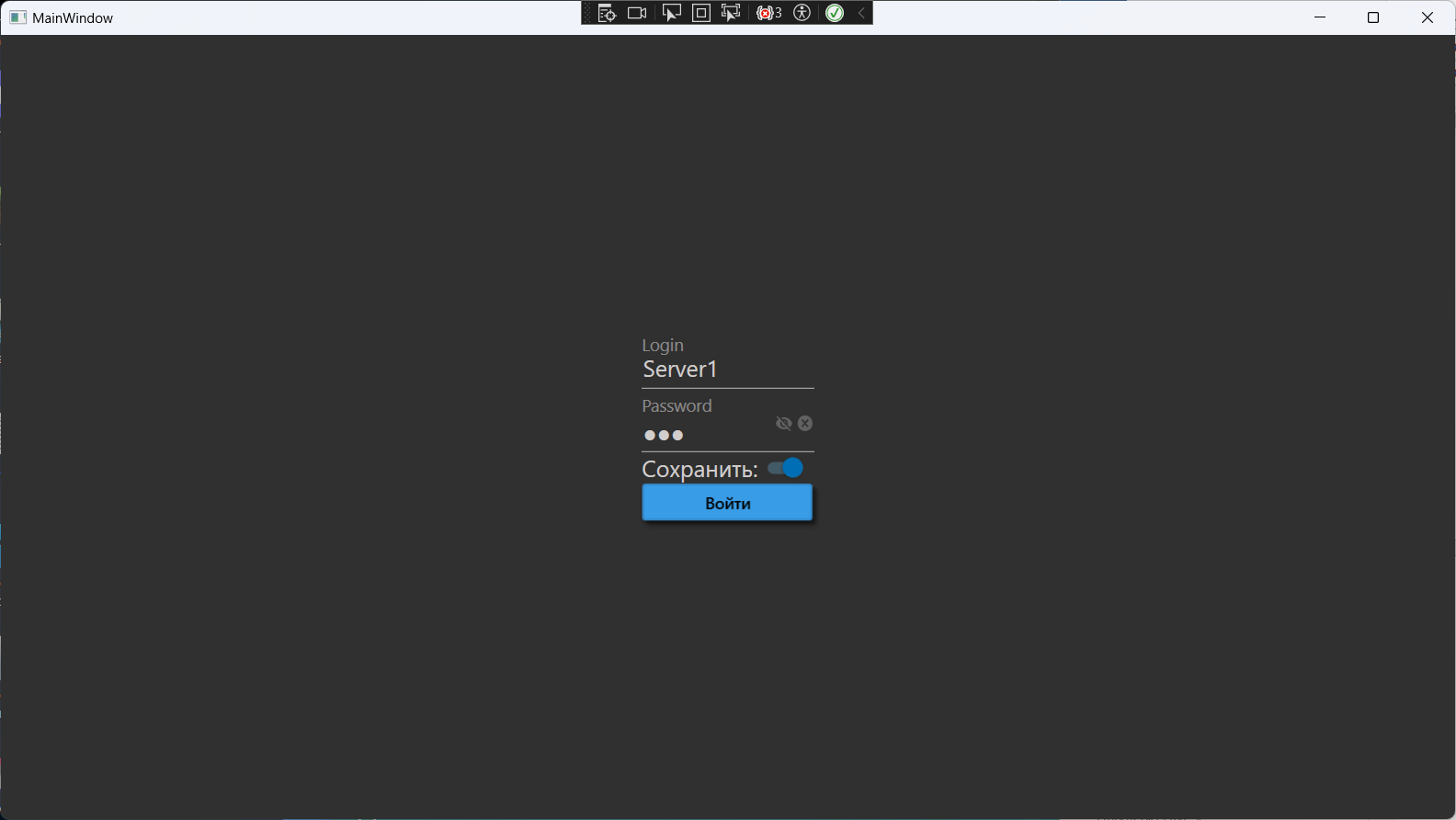


Рисунок Б.5

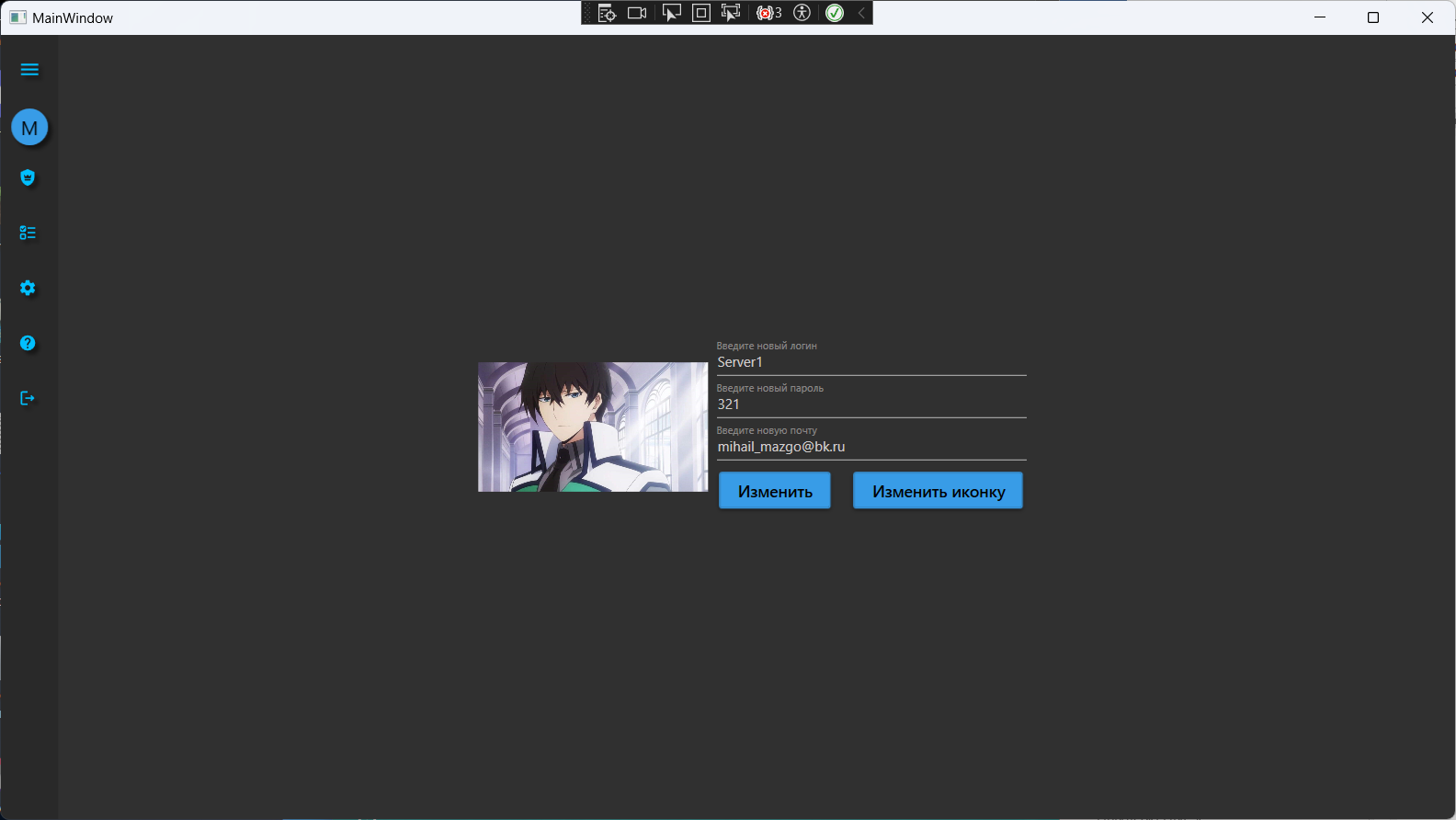


Рисунок Б.6