Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский экономический университет имени Г.В. Плеханова»

Московский приборостроительный техникум

Выпускная квалификационная работа

(Дипломная работа)

|  |  |
| --- | --- |
| На тему: | Разработка информационной системы электронного документооборота по организации производственных практик (на примере ФГБОУ ВО "Российский экономический университет им Г.В. Плеханова") |

АЛПАТОВА НИКИТЫ СЕРГЕЕВИЧА

Студент (-ка) 4 курса группы П-2-17

по специальности 09.02.03 «Программирование в компьютерных системах»

для присвоения квалификации: техник – программист

Форма обучения: очная

Руководитель: /Дубовик Алексей Сергеевич/

(подпись)

« » 2021 г.

Консультант: / Ермашенко Екатерина Антоновна /

(подпись)

« » 2021 г.

Студент (-ка): / Алпатов Никита Сергеевич /

(подпись)

« » 2021 г.

Допущен к защите

Распоряжение от « » 2021 г. №

2021

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc73654994)

[1. ОБЩАЯ ЧАСТЬ 5](#_Toc73654995)

[1.1 Цель разработки 5](#_Toc73654996)

[1.2 Средства разработки 5](#_Toc73654997)

[2. СПЕЦИАЛЬНАЯ ЧАСТЬ 7](#_Toc73654998)

[2.1 Постановка задачи 7](#_Toc73654999)

[2.1.1 Входные данные и выходные данные 7](#_Toc73655000)

[2.1.2 Подробные требования к проекту 7](#_Toc73655001)

[2.2 Внешняя спецификация 8](#_Toc73655002)

[2.2.1 Описание задачи 8](#_Toc73655003)

[2.2.2 Описание входных и выходных данных 14](#_Toc73655004)

[2.2.3 Методы 21](#_Toc73655005)

[2.2.4 Тесты 22](#_Toc73655006)

[2.2.5 Контроль целостности данных 24](#_Toc73655007)

[2.3 Проектирование 24](#_Toc73655008)

[2.3.1 Функциональная схема 25](#_Toc73655009)

[2.3.2 Структурная схема 26](#_Toc73655010)

[2.3.3 Схема архитектуры 28](#_Toc73655011)

[2.3.4 Схема данных 29](#_Toc73655012)

[2.3.5 Диаграмма классов 31](#_Toc73655013)

[2.3.6 Схема интерфейса 32](#_Toc73655014)

[2.4 Результат работы программы 33](#_Toc73655015)

[3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 37](#_Toc73655016)

[3.1 Инструментальные средства 37](#_Toc73655017)

[3.2 Отладка программы 38](#_Toc73655018)

[3.3 Защитное программирование 40](#_Toc73655019)

[3.4 Характеристика программы 42](#_Toc73655020)

[ЗАКЛЮЧЕНИЕ 44](#_Toc73655021)

[СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ 45](#_Toc73655022)

[ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ 1](#_Toc73655023)

[ПРИЛОЖЕНИЕ Б. СЦЕНАРИЙ И РЕЗУЛЬТАТЫ ТЕСТОВЫХ ИСПЫТАНИЙ 1](#_Toc73655024)

[ПРИЛОЖЕНИЕ В. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 1](#_Toc73655025)

[ПРИЛОЖЕНИЕ Г. СКРИПТ БАЗЫ ДАННЫХ 1](#_Toc73655026)

[ПРИЛОЖЕНИЕ Д. СХЕМА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА 1](#_Toc73655027)

[ПРИЛОЖЕНИЕ Е. СЛОВАРЬ ДАННЫХ 1](#_Toc73655028)

ВВЕДЕНИЕ

Московский приборостроительный техникум является структурным подразделением Российского экономического университета имени Г.В. Плеханова, реализующее программы среднего профессионального образования по следующим группам специальностей: 09.00.00 Информатика и вычислительная техника и 10.00.00 Информационная безопасность. Помимо технических специальностей, традиционных для техникума, с 2012 года началась реализация образовательной программы, входящей в следующую группу специальностей: 40.00.00. Юриспруденция и 40.02.01 Право и организация социального обеспечения. Московский приборостроительный техникум принимает в свои стены абитуриентов, которые ищут техникум (колледж) после окончания 9-ти или 11-ти классов школы.

Помимо этого, техникум сотрудничает с другими компаниями и распределяет в них студентов для прохождения производственных практик.

В данный момент Московский приборостроительный техникум нуждается в специализированном программном обеспечении, работа персонала в основном производится вручную, что является трудозатратным способом достижения цели, а именно распределение студента на производственную практику.

Для упрощения и автоматизирования процесса распределения студентов на производственную практику будет создана информационная система электронного документооборота по организации производственных практик.

В программе будет присутствовать две роли «Сотрудник» и «Администратор», при регистрации в программе потребуется код, который и будет присваивать роли каждому пользователю, который проходит регистрацию. Данные коды будет возможно сменить пользователю с ролью «Администратор», помимо этого программа будет иметь понятный и простой интерфейс и позволит пользователю сохранять настройки под себя.

Также программа позволит автоматически создавать файлы для распределения студентов на производственную практику, директорию сохранения данных файлов можно будет выбрать в настройках. Файлы, созданные в программе, автоматически сортируются по папкам, в зависимости от группы студента, это позволит упростить поиск этих файлов на компьютере пользователя. В программе будет реализована защита от некорректных действий пользователя, которая обеспечит максимальную безопасность вводимых в программу данных и обеспечит их целостность.

1. ОБЩАЯ ЧАСТЬ

1.1 Цель разработки

Целью разработки информационной системы «WKR» является упрощение и автоматизирование процесса распределения студентов на производственные практики.

1.2 Средства разработки

Для разработки и проектирования системы, а также написания технической документации, использовались программные средства, представленные в Таблице 1.

Таблица 1 − Программное обеспечение

|  |  |  |  |
| --- | --- | --- | --- |
|  | Назначение программного средства | Наименование | Описание |
| № | 1 | 2 | 3 |
| 1 | Базовое средство для разработки | Microsoft SQL Server Management Studio 18 | Создание и управление базой данных |
| 2 | Средство проектирования | Draw.io | Информационный ресурс позволяющий разрабатывать схемы для проектирования приложения |
| 3 | Текстовый редактор | Microsoft Word 2016 | Разработка документации, формирование отчётных документов по шаблонам |
| 4 | Инструментальное средство разработки программных решений | Visual Studio 2019 Community 16.8.2 | Разработка клиент-серверного приложения и базы данных. |

В качестве средств вычислительной техники использовались средства, перечисленные в Таблице 2.

Таблица 2 − Аппаратные средства

|  | Наименование | Описание |
| --- | --- | --- |
| № | 1 | 2 |
| 1 | Размер экрана: | 43” |
| 2 | Разрешение экрана: | 3840x1200 |
| 3 | Линейка процессора: | AMD Ryzen |
| 4 | Количество ядер процессора: | 6/12 |
| 5 | Оперативная память: | 16 ГБ |
| 6 | Тип видеокарты: | дискретная |
| 7 | Видеокарта: | Nvidia Gtx 1080 |
| 8 | Конфигурация накопителей: | SSD + HDD |
| 9 | Общий объем всех накопителей: | 1200 Гб |

2. СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1 Постановка задачи

Разработать информационную систему электронного документооборота по организации производственных практик (на примере ФГБОУ ВО «Российский экономический университет им. Г.В. Плеханова»)

2.1.1 Входные данные и выходные данные

Входными данными являются данные о студентах, компаниях, группах, пользователе, датах практик.

Выходными данными являются сформированные документы в формате (.doc) и данные о распределенных студентах.

2.1.2 Подробные требования к проекту

* В программе должна быть реализована авторизация и регистрация пользователя с определенной ролью;
* В программе должны быть реализованы две роли: «Сотрудник» и «Администратор»;
* Роль администратора должна обеспечивать возможность пользователю изменять код для получения ролей в программе;
* Роль сотрудника должна обеспечивать возможность пользователю добавлять, удалять, изменять данные о компаниях, студентах, датах практик и группах;
* Должны быть реализованы настройки для роли «Сотрудник». В настройках пользователю с ролью «Сотрудник» должна быть возможность выбора места хранения сформированных документов и шаблонов этих документов;
* В программе должна присутствовать возможность формирования документов для распределения студентов на производственные практики в компании, каждому документу должен присваиваться свой идентификационный номер;
* Пользователь с ролью «Сотрудник» должен иметь возможность сменить пароль;
* Программа должна обеспечивать целостность данных.

2.2 Внешняя спецификация

2.2.1 Описание задачи

Необходимо разработать информационную систему электронного документооборота, которая позволит автоматизировать и упростить процесс распределения студентов на производственные практики.

На Рисунке 1 представлена диаграмма прецедентов, она позволяет описать все возможности каждой из ролей программы.

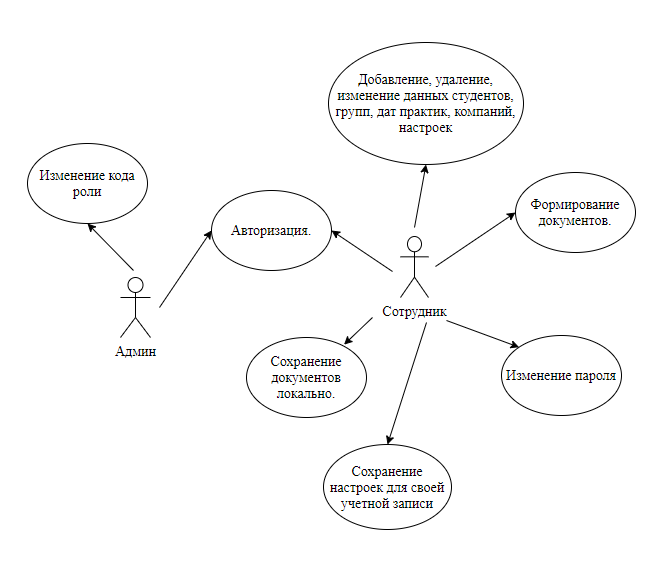


Рисунок 1 – Диаграмма прецедентов

На диаграмме представлены два актера: «Сотрудник» и «Админ».

Пользователь с ролью «Админ» обладает следующими функциональными возможностями:

* Изменение кода роли;
* Авторизация.

Пользователь с ролью «Сотрудник» обладает следующими функциональными возможностями:

* Добавление, удаление, изменение данных студентов, групп, дат практики, компаний, настроек;
* Авторизация;
* Формирование документов;
* Сохранение документов локально;
* Сохранение настроек для своей учетной записи;
* Изменение пароля.

На Рисунках 2-3 продемонстрирована диаграмма бизнес-процессов (IDEF0), а именно уровни A-0 и A0, на котором графически изображены механизмы, алгоритмы или инструкции для управления процессами, а также входные и выходные данные.

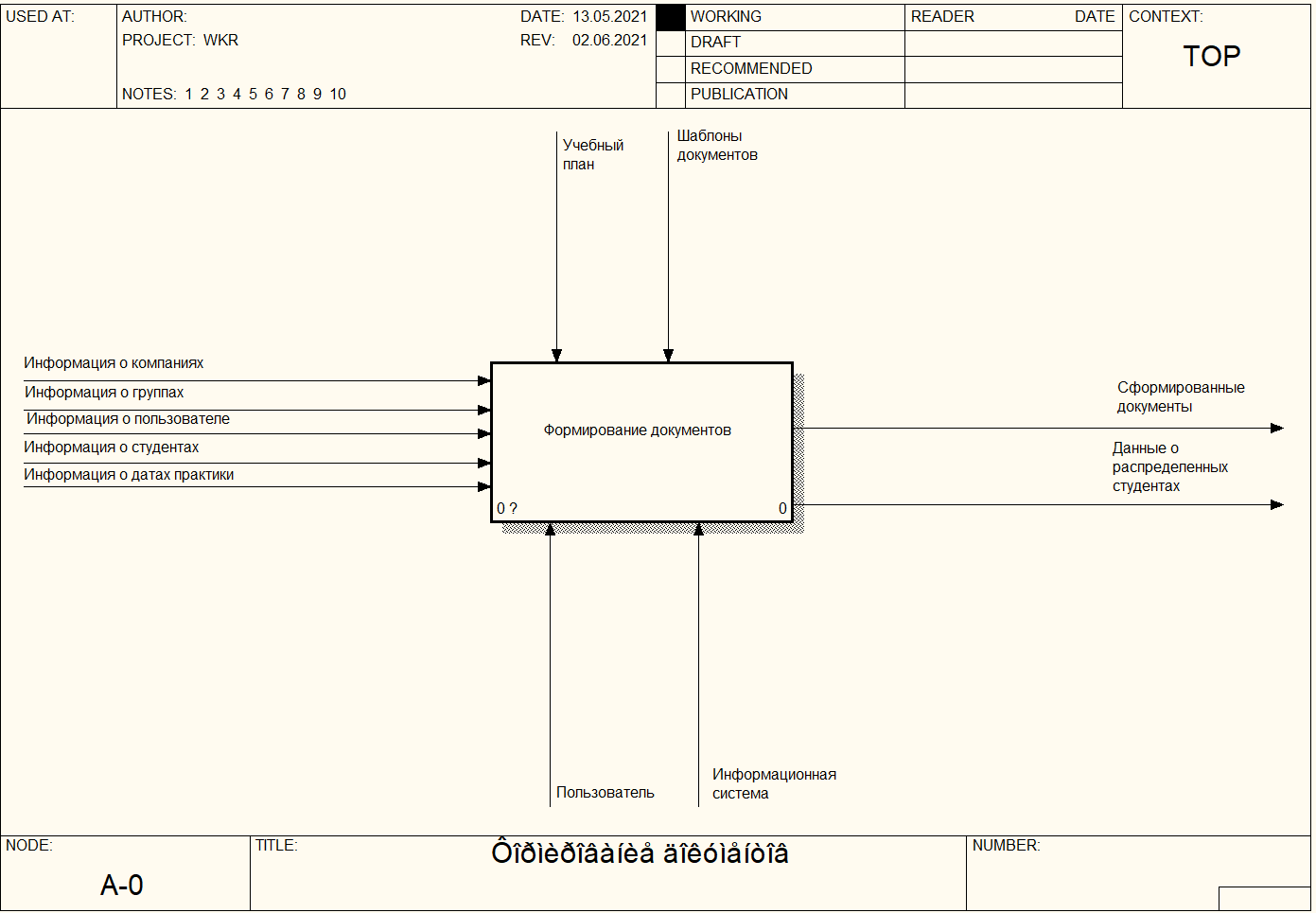


Рисунок 2 − Контекстная диаграмма

Таблица 3 − Подробное описание диаграммы бизнес-процесса уровня А0

|  |  |
| --- | --- |
| Название стрелки | Описание |
| 1 | 2 |
| Входные данные | |
| Информация о компаниях | Данные о компаниях, которые вводит пользователь программы например: название компании, почта компании, ФИО руководителя и т.д. |
| Информация о группах | Данные о группах, которые вводит пользователь программы например: номер группы и специальность. |
| Информация о пользователе | Данные о пользователе, которые вводит пользователь программы например: ФИО пользователя, почта, логин, пароль, роль и т.д. |
| Информация о студентах | Данные о студентах, которые вводит пользователь программы например: ФИО студента, почта, компания по практике, дата рождения и т.д. |
| Информация о датах практики | Данные о датах практики, которые вводит пользователь программы например: даты практик, название модуля, название практики, производственная или преддипломная практика и т.д. |
| Управление | |
| Учебный план | Учебный план, на основании которого происходит направление студентов на производственную практику. |
| Шаблоны документов | Шаблоны документов на основании которого формируются документы |
| Выходные данные | |
| Данные о распределенных студентах | Информация, которая была введена п ходе работы пользователя с программой. Информация о группах, датах практики, компаниях, студентах. |
| Сформированные документы | Документы, которые были сформированы приложением. |
| Механизм | |
| Пользователь | Пользователь программы, который пользуется ее функционалом. |
| Информационная система | Программа, которая содержит в себе некоторый функционал. |

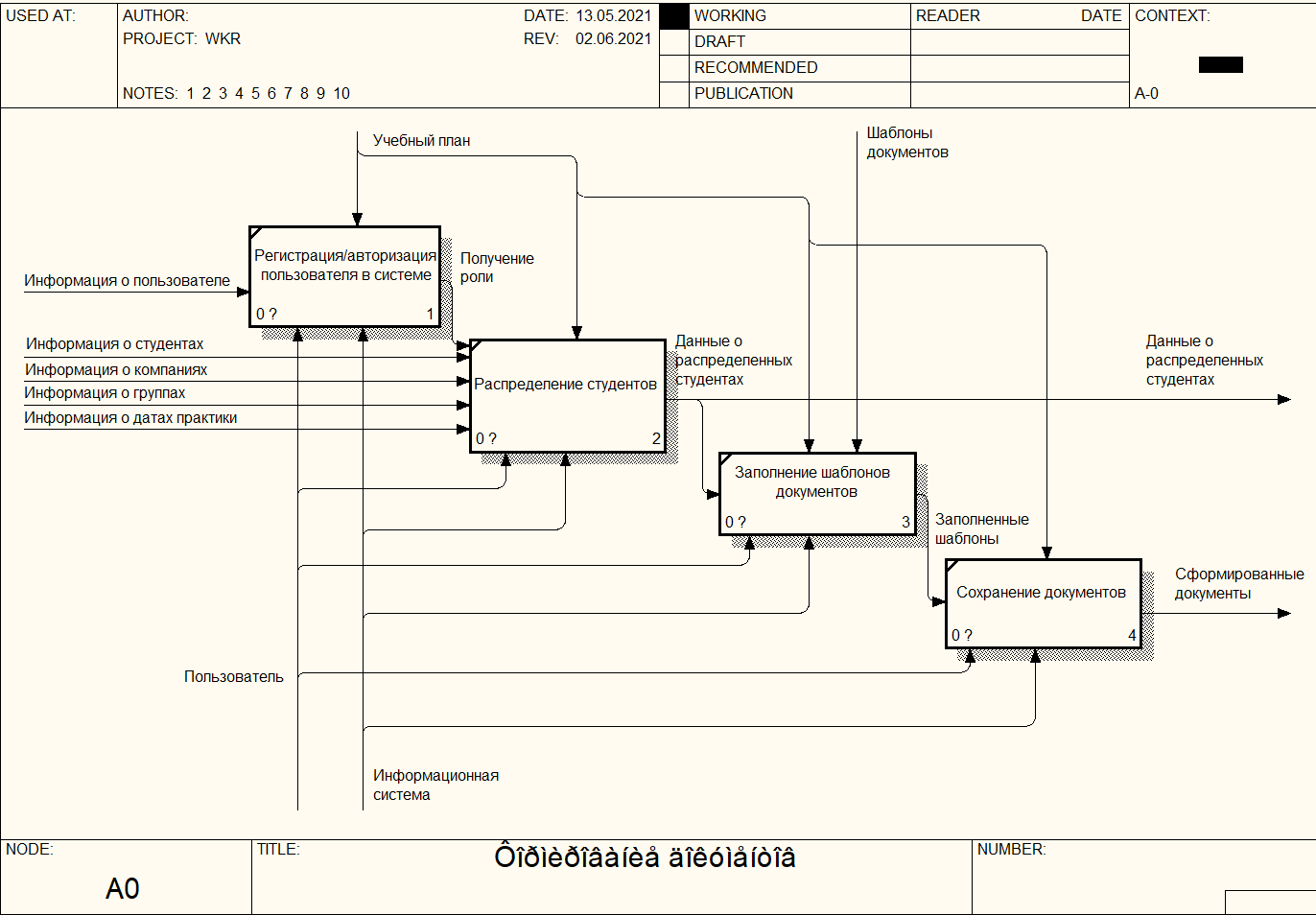


Рисунок 3 − Диаграмма бизнес-процессов, уровень A0

Таблица 4 − Подробное описание диаграммы бизнес-процесса уровня А0

| Название стрелки | Описание |
| --- | --- |
| 1 | 2 |
| Вход | |
| Информация о компаниях | Данные о компаниях, которые вводит пользователь программы например: название компании, почта компании, ФИО руководителя и т. д |
| Информация о группах | Данные о группах, которые вводит пользователь программы например: номер группы и специальность. |
| Информация о пользователе | Данные о пользователе, которые вводит пользователь программы например: ФИО пользователя, почта, логин, пароль, роль и т. д |
| Информация о студентах | Данные о студентах, которые вводит пользователь программы например: ФИО студента, почта, компания по практике, дата рождения и т. д |
| Информация о датах практики | Данные о датах практики, которые вводит пользователь программы например: |
| Управление | |
| Учебный план | Учебный план, на основании которого происходит направление студентов на производственную практику. |
| Шаблоны документов | Шаблоны документов на основании которого формируются документы |
| Выход | |
| Данные о распределенных студентах | Информация, которая была введена п ходе работы пользователя с программой. Информация о группах, датах практики, компаниях, студентах. |
| Сформированные документы | Документы, которые были сформированы приложением. |
| Заполненные шаблоны | Шаблоны документов, которые содержат введенную пользователем информацию. |
| Получение роли | Данные о роле пользователя. |
| Механизм | |
| Пользователь | Пользователь программы, который пользуется ее функционалом. |
| Информационная система | Программа, которая содержит в себе некоторый функционал. |

2.2.2 Описание входных и выходных данных

Ниже представлена Таблица 5 с описанием входных данных.

Таблица 5 − Входные данные модулей

| № | Наименование | Ограничения | Тип | Описание |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Форма «Администратор» | | | | |
| 1 | Password | [А-Я, A-Z, 0-9]  {1-50} | string | Поле кода роли |
| Форма «Авторизация» | | | | |
| 1 | Login | [А-Я, A-Z, 0-9]  {1-50} | string | Поле логина |
| 2 | Password | [А-Я, A-Z, 0-9]  {1-50} | string | Поле пароля |
| Форма «Компании» | | | | |
| 1 | Name | [А-Я, A-Z, 0-9]  {1-50} | string | Поле названия компании |
| 2 | Manager\_Surname | [А-Я, A-Z, 0-9]  {1-50} | string | Поле фамилии руководителя |
| 3 | Manager\_Name | [А-Я, A-Z, 0-9]  {1-50} | string | Поле имени руководителя |
| 4 | Manager\_Patronymic | [А-Я, A-Z, 0-9]  {1-50} | string | Поле отчества руководителя |
| 5 | Manager\_Position | [А-Я, A-Z, 0-9]  {1-50} | string | Поля должности руководителя |
| 6 | Manager\_surname\_in\_the\_dative\_case | [А-Я, A-Z, 0-9]  {1-50} | string | Поле фамилии руководителя в дательном падеже |
| 7 | Manager\_name\_in\_the\_dative\_case | [А-Я, A-Z, 0-9]  {1-50} | string | Поле имени руководителя в дательном падеже |
| 8 | Manager\_patronymic\_in\_the\_dative\_case | [А-Я, A-Z, 0-9]  {1-50} | string | Поле отчества руководителя в дательном падеже |
| 9 | Email | [А-Я, A-Z, 0-9]  {1-50} | string | Поле почты |
| Форма «Группы» | | | | |
| 1 | Group\_name | [А-Я, A-Z, 0-9]  {1-50} | string | Поле названия группы |
| 2 | Specialization | [А-Я, A-Z, 0-9]  {1-50} | string | Поле специальности группы |
| Форма «Даты практик» | | | | |
| 1 | Professional\_module | [А-Я, A-Z, 0-9]  {1-50} | string | Поле названия профессионального  модуля |
| 2 | Name\_of\_the\_practice | [А-Я, A-Z, 0-9]  {1-50} | string | Поле названия практики |
| 3 | Pre\_graduate\_practice | [А-Я, A-Z, 0-9]  {1-50} | string | Поле преддипломной практики |
| 4 | Group\_name | [А-Я, A-Z, 0-9]  {1-50} | string | Поле названия группы |
| 5 | Pactice\_start\_date | {1-9}.{1-9}.{1-9 | date | Поле даты начала практики |
| 6 | Pactice\_termination\_date | {1-9}.{1-9}.{1-9 | date | Поле даты приостановки практики |
| 7 | Date\_of\_continuation\_of\_the\_practice | {1-9}.{1-9}.{1-9 | date | Поле даты продолжения практики |
| 8 | Pactice\_termination\_date\_2 | {1-9}.{1-9}.{1-9 | date | Поле даты приостановления практики |
| 9 | Date\_of\_continuation\_of\_the\_practice\_2 | {1-9}.{1-9}.{1-9 | date | Поле даты продолжения практики |
| 10 | Practice\_end\_date | {1-9}.{1-9}.{1-9 | date | Поле окончания практики |
| Форма «Регистрация» | | | | |
| 1 | Surname | [А-Я, A-Z, 0-9]  {1-50} | string | Поле фамилии пользователя |
| 2 | Name | [А-Я, A-Z, 0-9]  {1-50} | string | Поле имени пользователя |
| 3 | Patronymic | [А-Я, A-Z, 0-9]  {1-50} | string | Поле отчества пользователя |
| 4 | Email | [А-Я, A-Z, 0-9]  {1-50} | string | Поле почты пользователя |
| 5 | Login | [А-Я, A-Z, 0-9]  {1-50} | string | Поле логина пользователя |
| 6 | Password | [А-Я, A-Z, 0-9]  {1-50} | string | Поле пароля пользователя |
| 7 | Birthday | {1-9}.{1-9}.{1-9 | date | Поле даты рождения |
| 8 | Role | [А-Я, A-Z, 0-9]  {1-50} | string | Поле кода роли |
| Форма «Настройки» | | | | |
| 1 | Document\_storage\_location | [А-Я, A-Z, 0-9]  {1-50} | string | Поле ввода места хранения сформированных документов |
| 2 | Document\_template\_storage\_location | [А-Я, A-Z, 0-9]  {1-50} | string | Поле ввода места хранения первого шаблона |
| 3 | Document2\_template\_storage\_location | [А-Я, A-Z, 0-9]  {1-50} | string | Поле ввода места хранения второго шаблона |
| Форма «Студент» | | | | |
| 1 | Surname | [А-Я, A-Z, 0-9]  {1-50} | string | Поле фамилии студента |
| 2 | Name | [А-Я, A-Z, 0-9]  {1-50} | string | Поле имени студента |
| 3 | Patronymic | [А-Я, A-Z, 0-9]  {1-50} | string | Поле отчества студента |
| 4 | Group | [А-Я, A-Z, 0-9]  {1-50} | string | Поле группы студента |
| 5 | Birthday | {1-9}.{1-9}.{1-9 | date | Поле даты рождения студента |
| 6 | Email | [А-Я, A-Z, 0-9]  {1-50} | string | Поле почты студента |
| 7 | Surname\_in\_the\_accusative\_case | [А-Я, A-Z, 0-9]  {1-50} | string | Поле фамилии студента в винительном падеже |
| 8 | First\_name\_in\_the\_accusative\_case | [А-Я, A-Z, 0-9]  {1-50} | string | Поле имени студента в винительном падеже |
| 9 | Patronymic\_in\_the\_accusative\_case | [А-Я, A-Z, 0-9]  {1-50} | string | Поле отчества студента в винительном падеже |
| Форма «Пользователь» | | | | |
| 1 | Password | [А-Я, A-Z, 0-9]  {1-50} | string | Поле старого пароля |
| 2 | New\_Password | [А-Я, A-Z, 0-9]  {1-50} | string | Поле нового пароля |

Ниже представлена Таблица 6 с описанием выходных данных.

Таблица 6 − Выходные данные

| № | Данные | Тип данные | Формат вывода | Описание |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Форма «Администратор» | | | | |
| 1 | Notification | Объект | Сообщение | Сообщение о неверно введенных данных |
| 2 | Notification | Объект | Сообщение | Сообщение, что введенный пароль уже используется другой ролью |
| Форма «Авторизация» | | | | |
| 1 | Notification | Объект | Сообщение | Сообщение о неверно введенном логине или пароле |
| 2 | Notification | Объект | Сообщение | Сообщение о неверно введенных данных |
| Форма «Компании» | | | | |
| 1 | Notification | Объект | Сообщение | Сообщение, что одно из полей не заполнено |
| 2 | Notification | Объект | Сообщение | Сообщение, что компанию невозможно удалить |
| 3 | Notification | Объект | Сообщение | Сообщение о неверно введенных данных |
| 4 | Notification | Объект | Сообщение | Сообщение о том, что одно из полей не заполнено |
| Форма «Документооборот» | | | | |
| 1 | Notification | Объект | Сообщение | Сообщение о том, что папка существует |
| 2 | Notification | Объект | Сообщение | Сообщение о том, что папки группы не существует |
| 3 | Data | Объект | Документ в формате «doc» | Сформированный программой документ в формате .doc |
| 4 | Notification | Объект | Сообщение | Сообщение о том, что папка не указана в настройках |
| Форма «Группы» | | | | |
| 1 | Notification | Объект | Сообщение | Сообщение о неверно введенных данных |
| 2 | Notification | Объект | Сообщение | Сообщение о том, что одно из полей не заполнено |
| 3 | Notification | Объект | Сообщение | Сообщение о том, что такая группа уже существует |
| Форма «Даты практик» | | | | |
| 1 | Notification | Объект | Сообщение | Сообщение о том, что такой группы не существует |
| 2 | Notification | Объект | Сообщение | Сообщение о том, что у группы может быть только одна преддипломная практика |
| Форма «Регистрация» | | | | |
| 1 | Notification | Объект | Сообщение | Сообщение о том, что пароль или логин меньше 8 и 4 символов соответственно |
| 2 | Notification | Объект | Сообщение | Сообщение о том, что одно из полей не заполнено |
| 3 | Notification | Объект | Сообщение | Сообщение о том, что ФИО введено неверно |
| 4 | Notification | Объект | Сообщение | Сообщение о том, что такой логин уже существует |
| Форма «Регистрация» | | | | |
| 1 | Notification | Объект | Сообщение | Сообщение о том, что данные сохранены |
| Форма «Студент» | | | | |
| 1 | Notification | Объект | Сообщение | Сообщение о том, что папка существует |
| 2 | Notification | Объект | Сообщение | Сообщение о том, что одно из полей не заполнено |
| 3 | Notification | Объект | Сообщение | Сообщение о том, что для удаления необходимо выбрать строку со студентом |
| Форма «Пользователь» | | | | |
| 1 | Notification | Объект | Сообщение | Сообщение о том, что новый пароль должен отличаться от старого |
| 2 | Notification | Объект | Сообщение | Сообщение о том, что старый пароль неправильный |

2.2.3 Методы

При разработке информационной системы «WKR» была использована методология объектно-ориентированного программирования, основана на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования. Основные принципы структурирования в случае ООП связаны с различными аспектами базового понимания предметной задачи, которое требуется для оптимального управления соответствующей моделью.

Была использована инкапсуляция для быстрой и безопасной организации собственно иерархической управляемости: чтобы было достаточно простой команды «что делать», без одновременного уточнения как именно делать, так как это уже другой уровень управления.

На Рисунке 4 представлен фрагмент кода с примером инкапсуляции. Настройки приватности переменных.



Рисунок 4 – Инкапсуляция

2.2.4 Тесты

Для тестирования информационной системы была составлена схема тестирования, представленная на Рисунке 5.

Результаты тестовых испытаний продемонстрированы в Приложении Б. СЦЕНАРИЙ И РЕЗУЛЬТАТЫ ТЕСТОВЫХ ИСПЫТАНИЙ.

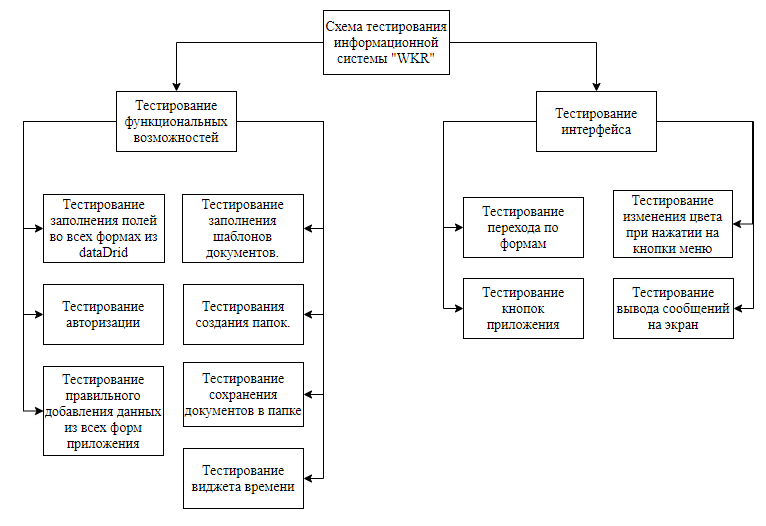


Рисунок 5 – Схема тестирования

2.2.5 Контроль целостности данных

Ниже представлена Таблица 7, в которой описан контроль целостности данных.

Таблица 7 − Контроль целостности данных

|  |  |  |  |
| --- | --- | --- | --- |
| Ситуация | Аномалия | Реакция программы | Примечание |
| 1 | 2 | 3 | 4 |
| Удаление, изменение и добавление данных из таблицы. | Таблица является связанной с другой. | Сообщение от программы. | Вывод сообщения о том, что для редактирования данных в таблице необходимо отредактировать другую таблицу. |
| Регистрация нового пользователя. | Неправильное заполнение полей в базе данных. | Сообщение от программы. | Предупреждение пользователя о том, что поля заполнены некорректно. |

2.3 Проектирование

В данном разделе представлена внутренняя структура программы, данных и пользовательского интерфейса.

А конкретно следующие схемы:

1) Функциональная схема.

2) Структурная схема.

3) Схема архитектуры.

4) Схема данных.

5) Диаграмма классов.

6) Схема интерфейса.

2.3.1 Функциональная схема

На Рисунке 6 представлена функциональная схема, она описывает весь функционал информационной системы.

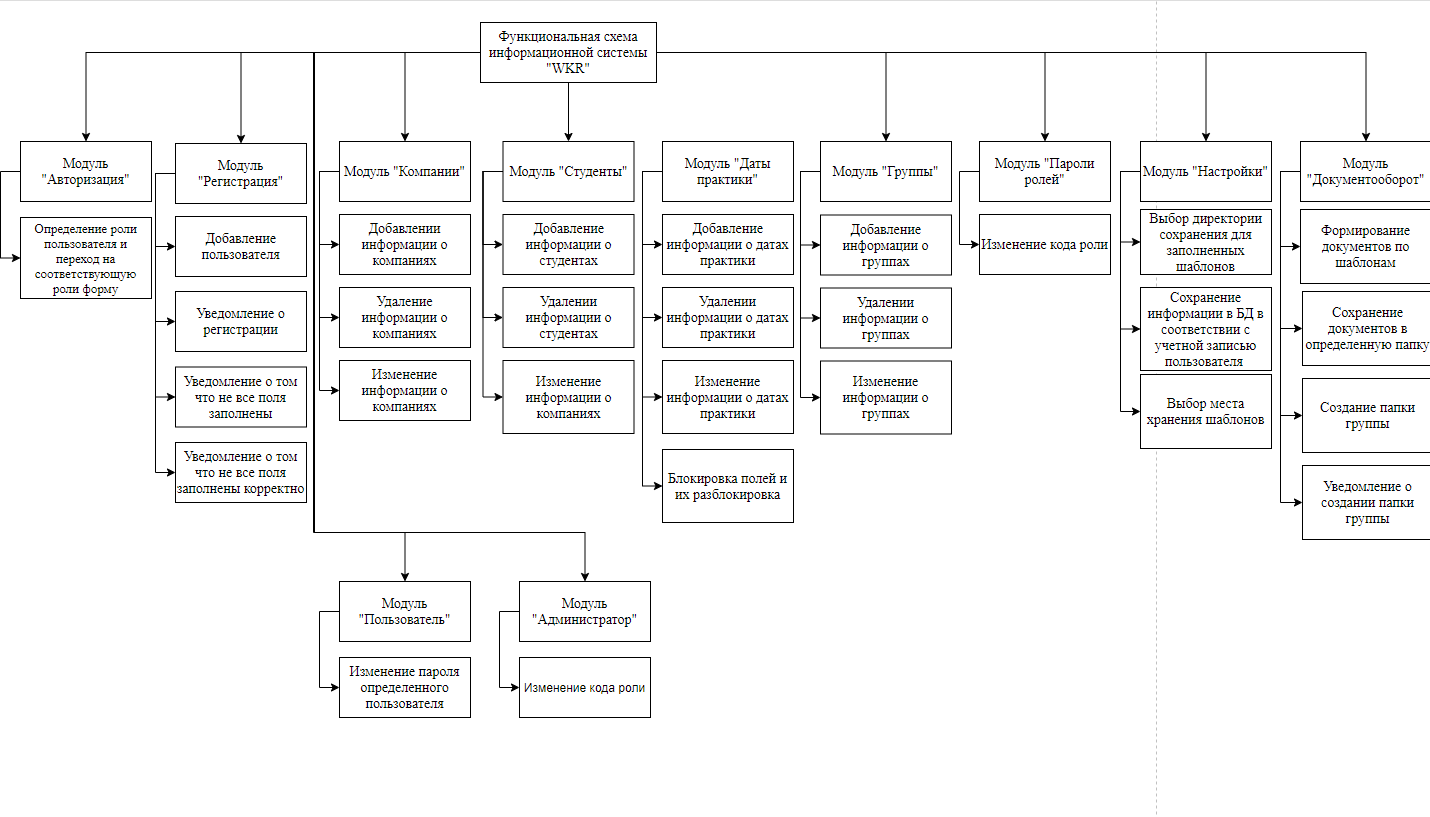


Рисунок 6 − Функциональная схема

2.3.2 Структурная схема

На Рисунке 7 представлена структурная схема информационной системы. Ниже приведена Таблица 8 с описанием структурной схемы.

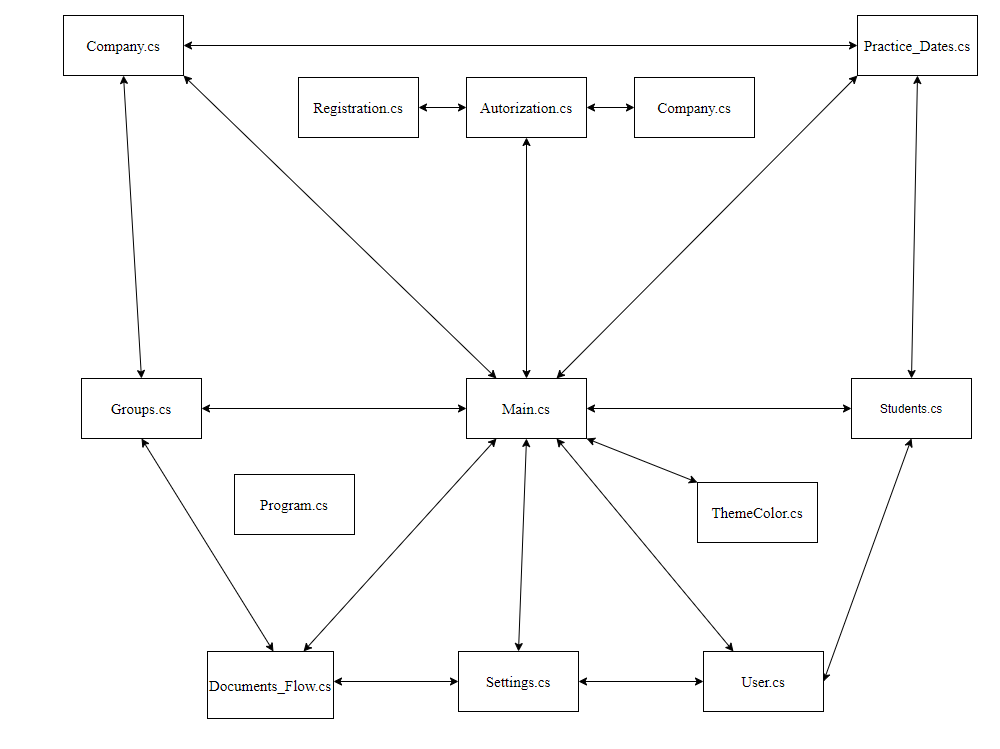


Рисунок 7 − Структурная схема

Таблица 8 − Подробное описание структурной схемы

|  |  |
| --- | --- |
| Название | Описание |
| 1 | 2 |
| Autorization.cs | Форма авторизации |
| Company.cs | Форма для работы с компаниями |
| Document\_Flow.cs | Форма для работы с документооборотом |
| Groups.cs | Форма для работы с группами |
| Practice\_Dates.cs | Форма для работы с датами практик |
| Registration.cs | Форма регистрации |
| User.cs | Форма пользователя |
| Settings.cs | Форма настроек пользователя |
| Students.cs | Форма работы со студентами |
| Main.cs | Основная форма для перехода по всем остальным |
| Program.cs | Класс программы, хранящий настройки проекта |
| ThemeColor.cs | Класс программы ответственный за дизайн |
| Administrator.cs | Форма для изменения пароля ролей |

2.3.3 Схема архитектуры

На Рисунке 8 представлена схема архитектуры.

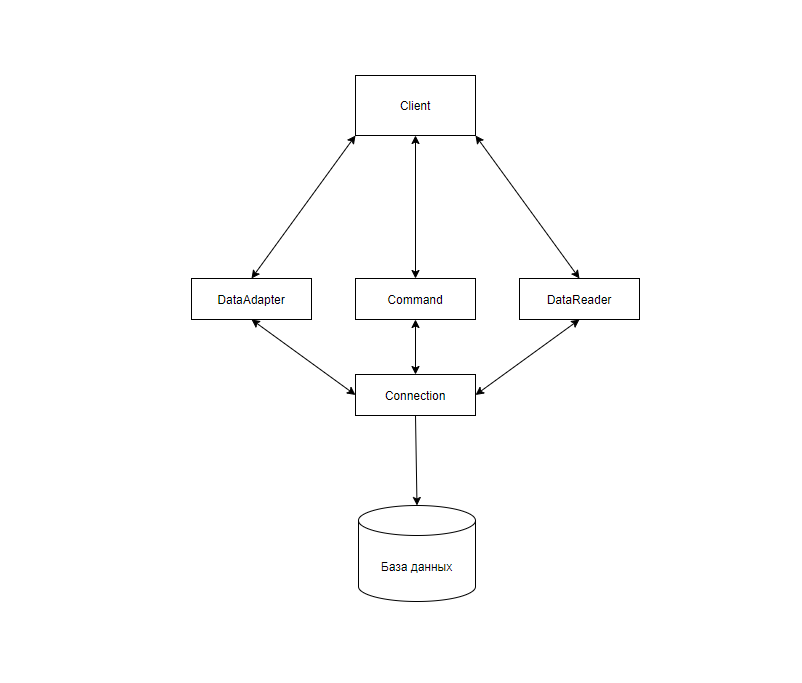


Рисунок 8 – Схема архитектуры

Связь информационной системы и базы данных осуществляется посредством запросов, последовательного однонаправленного чтения данных, для этого требуется компонент, хранящий строку подключения к базе данных, которая в свою очередь является последним звеном в цепочке клиент - база данных.

2.3.4 Схема данных

На Рисунках 9 представлена логическая схема базы данных. Логическая схема содержит в себе универсальное описание структуры данных, независимое от конечной реализации базы данных.

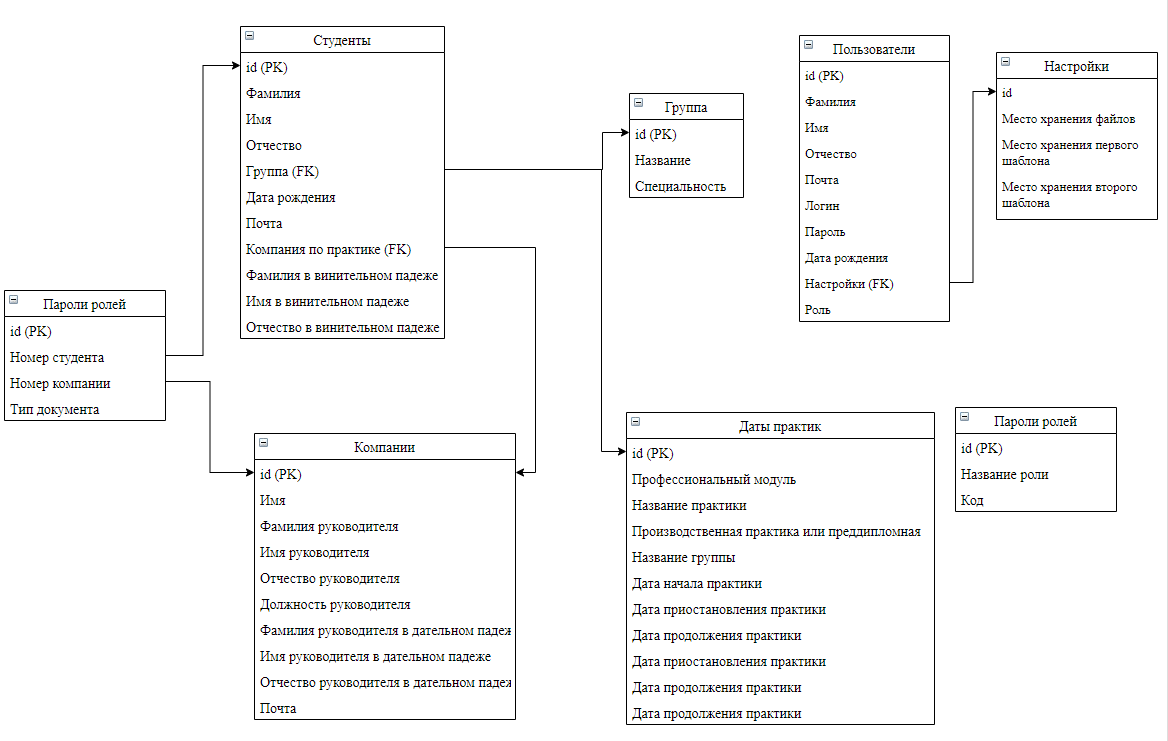


Рисунок 9 − Логическая схема базы данных

На Рисунке 10 представлена физическая схема базы данных, она определяет каким образом представляются данные, и содержит в себе все детали, необходимые системе управления базами данных для создания базы данных.

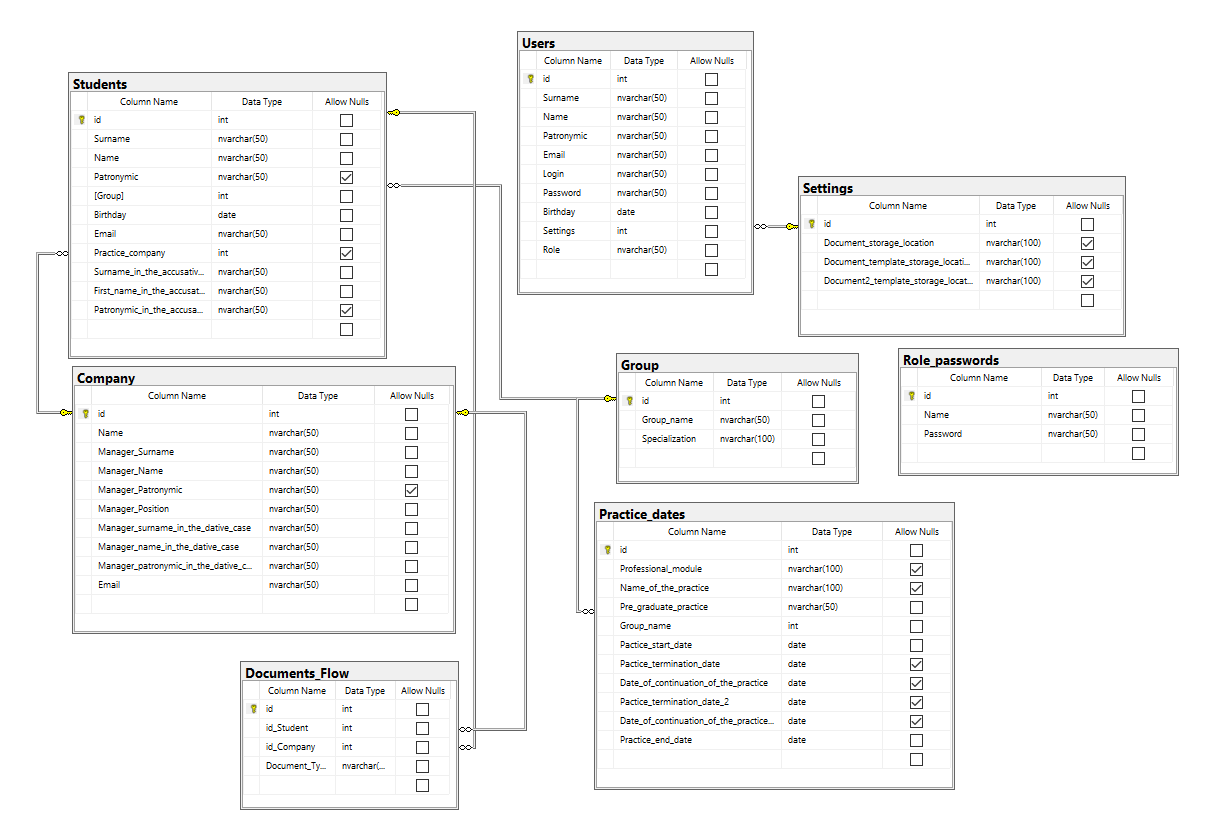


Рисунок 10 – Физическая схема базы данных

Словарь данных представлен в Приложении Е. СЛОВАРЬ ДАННЫХ.

2.3.5 Диаграмма классов

Ниже на Рисунках 10-11 представлена диаграмма классов.

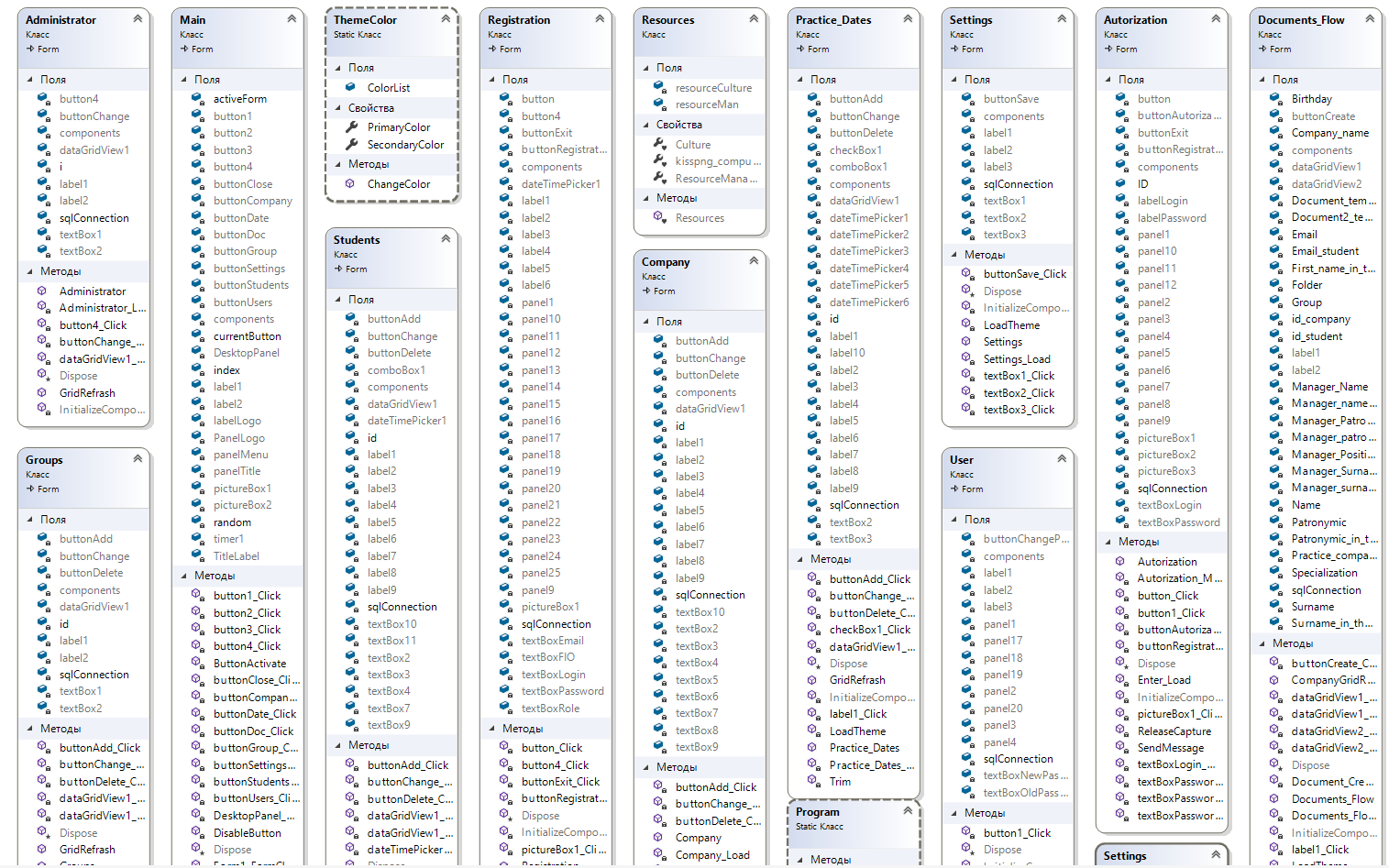


Рисунок 11 – Диаграмма классов

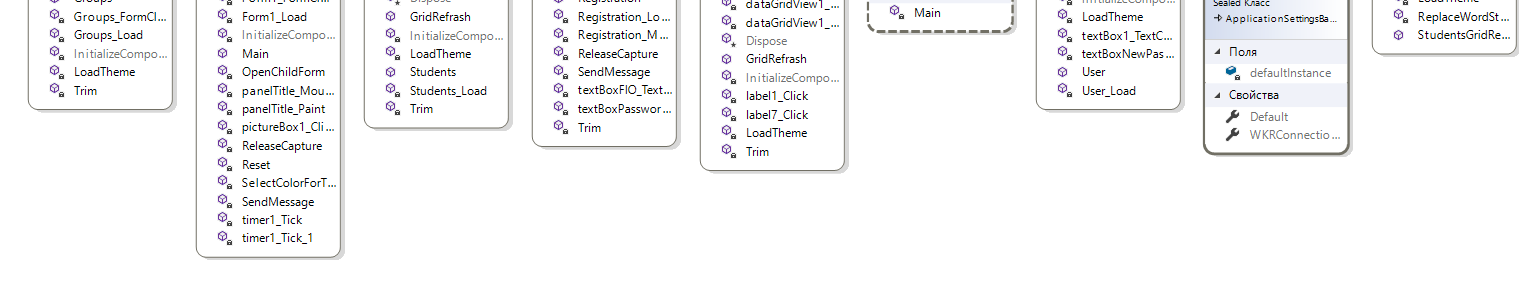


Рисунок 12 − Диаграмма классов

* + 1. Схема интерфейса

На Рисунке 12 представлена схема пользовательского интерфейса.

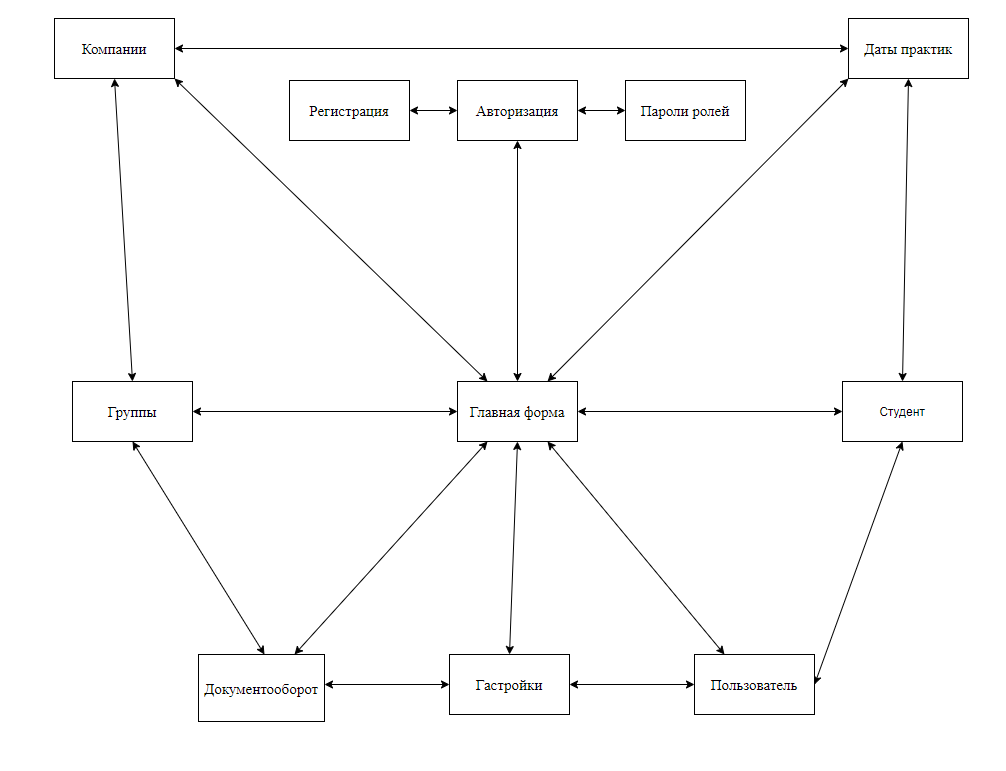


Рисунок 13 − Схема пользовательского интерфейса

2.4 Результат работы программы

На Рисунках 14-17 представлен результат работы программы. Подробнее описано в Приложении В. Руководство пользователя.

Результатом разработки информационного системы является готовое программное обеспечение.

На Рисунке 14 представлено окно программы, в котором происходит формирование документов по шаблонам.

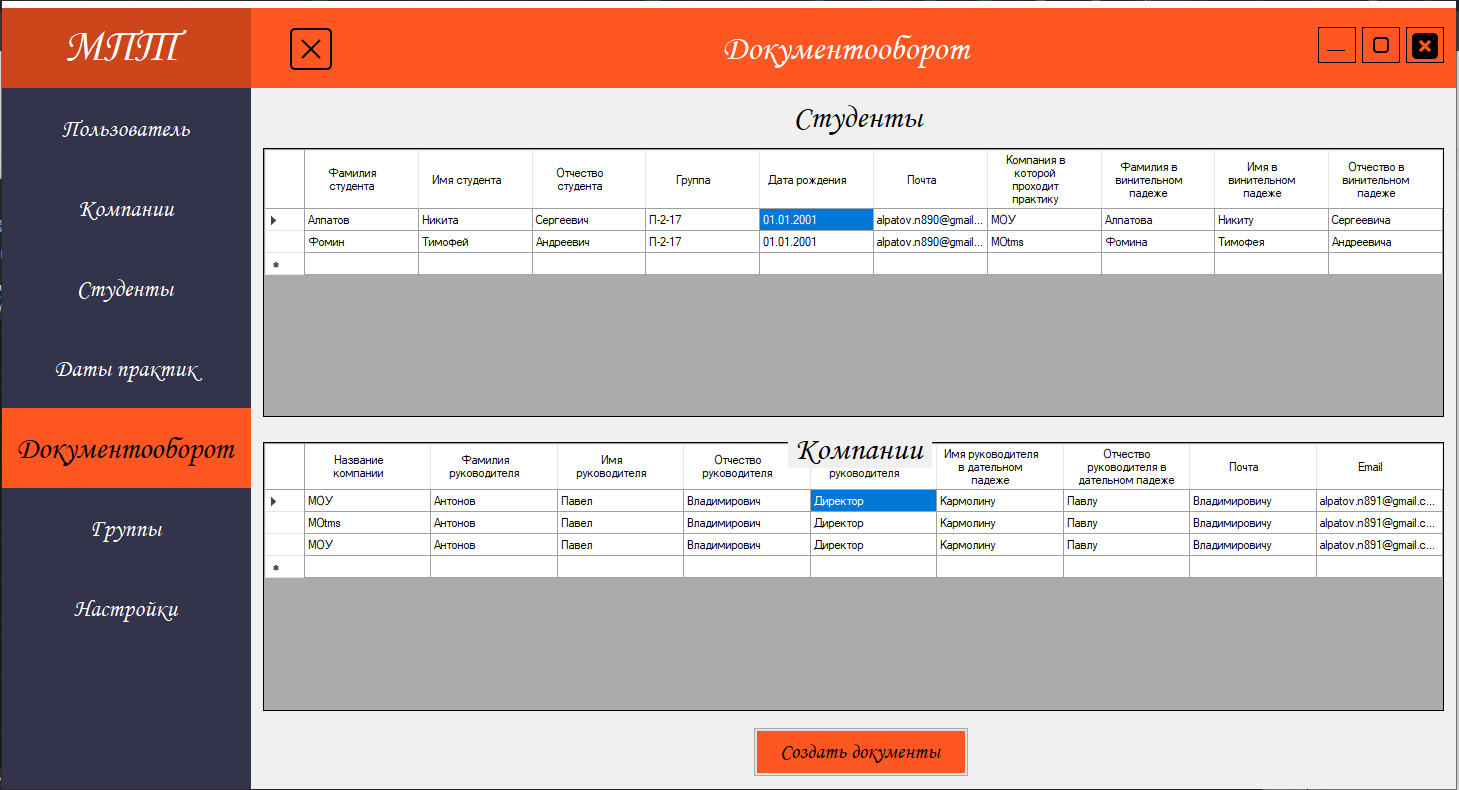


Рисунок 14 – Форма «Документооборот»

На Рисунке 15 представлено результат работы информационной системы, она автоматически распределяет документы по папкам в зависимости от группы студента чьи документы были сформированы.

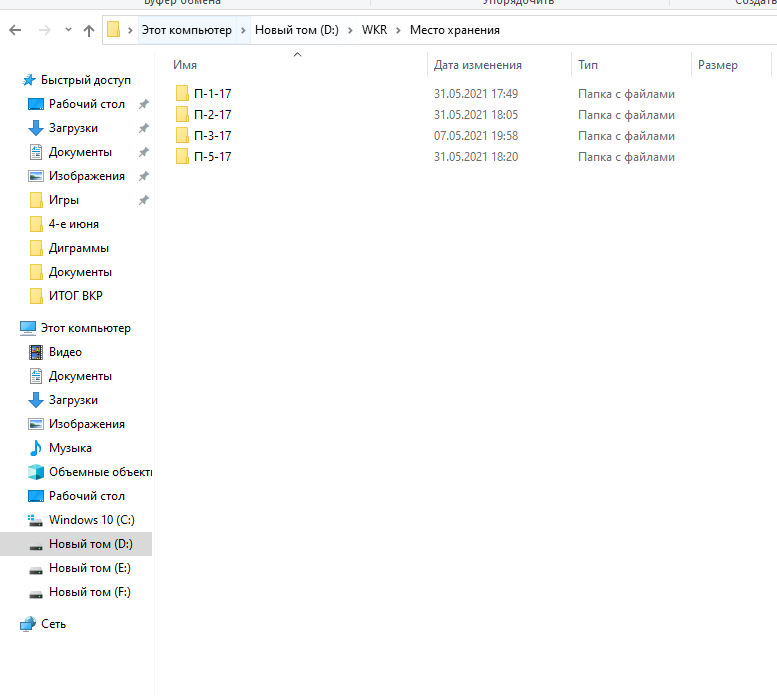


Рисунок 15 – Место хранения папок с файлами

На Рисунке 16 представлена открытая папка с документами группы «П-1-17».

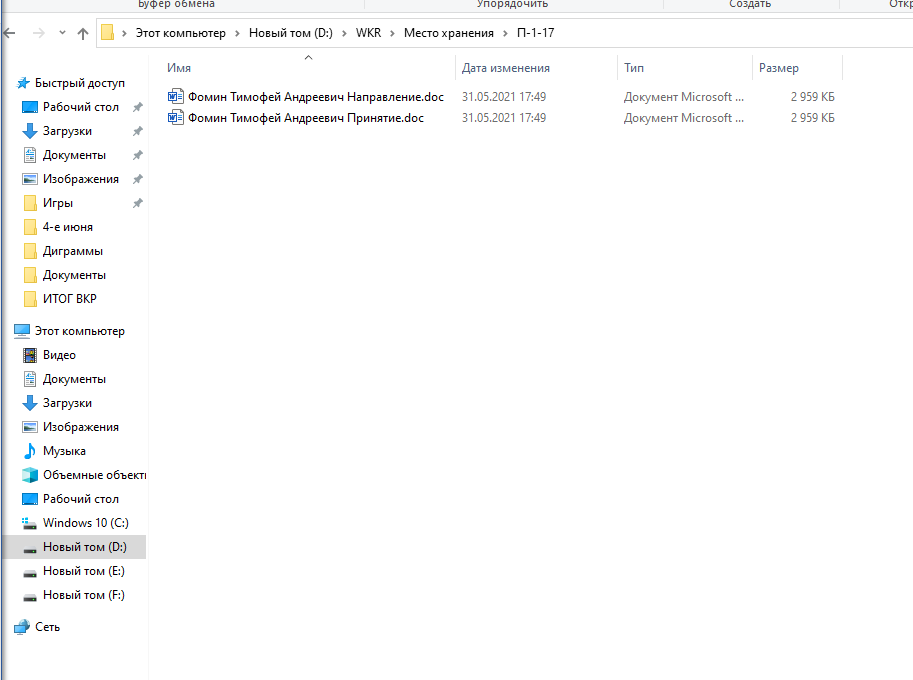


Рисунок 16 – Основное окно

На Рисунке 17 представлен сформированный по шаблону документ.

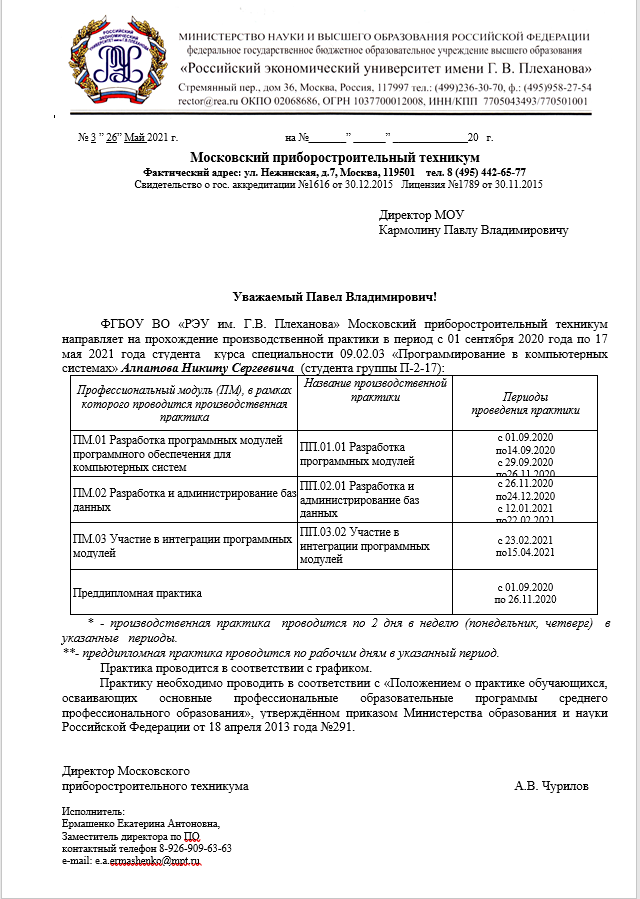


Рисунок 17 – Пример созданного файла

3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1 Инструментальные средства

Для разработки информационной системы «WKR» был использован язык программирования C# и библиотека WinForms.

Windows Forms – интерфейс программирования приложений (API), который отвечает за графический интерфейс пользователя и является частью Microsoft .Net Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде.

В качестве редактора кода использовалась среда разработки Visual Studio 19, так как он имеет удобный интерфейс, возможность написания кода на различных языках, а именно: c#, c++, python, javascript/typescript и т.д.

Проект хранится на локальном сервере Microsoft SQL Server Management Studio 18. SSMS – это интегрированная среда для управления любой инфраструктурой SQL, от SQL Server до баз данных SQL Azure. SSMS предоставляет средства для настройки, наблюдения и администрирования экземпляров SQL Server и баз данных. С помощью SSMS можно развертывать, отслеживать и обновлять компоненты уровня данных, используемые вашими приложениями, а также создавать запросы и скрипты.

3.2 Отладка программы

Для отладки программы был использован встроенный в среду разработки отладчик.

1. На Рисунке 18 представлен пример ошибки, когда в таблицу добавляются пустые данные.

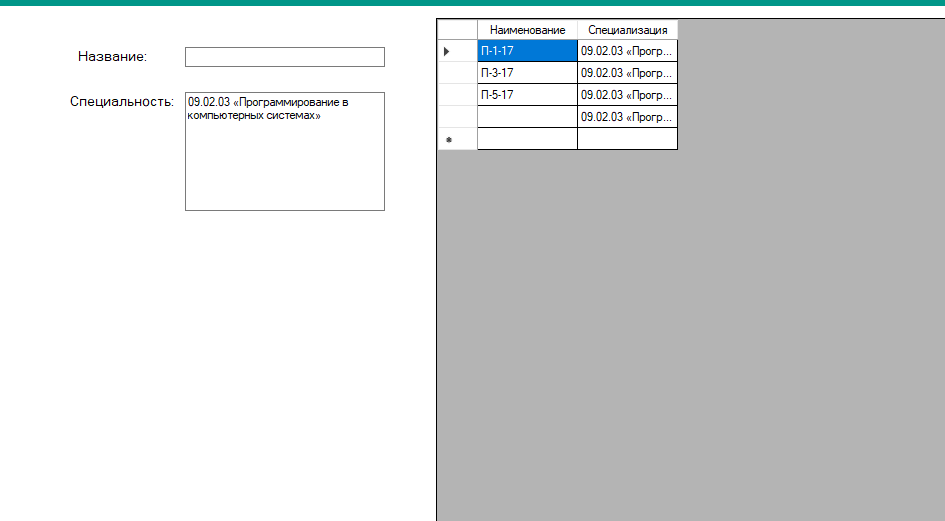


Рисунок 18 – Заполнение несуществующей таблицы

Для решения данной ошибки код был переписан и были добавлены проверки на корректность введенных данных, новый код представлен на Рисунке 19.



Рисунок 19 – Изменённый код

1. На рисунке 20 представлен пример ошибки, неправильного запроса на выбор места хранения документов.

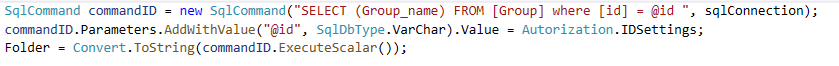


Рисунок 20 – Сравнение двух типов данных

Для решения данной ошибки был переписан запрос к базе данных, новый код представлен на Рисунке 21.

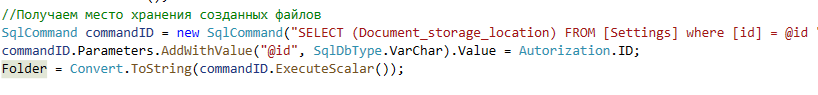


Рисунок 21 – Изменённый запрос

3.3 Защитное программирование

Защитное программирование в информационной системе «WKR» присутствует и его примеры приведены на Рисунках 22-24.

1. Проверка базы данных на предмет существует ли группа при добавлении некоторых записей представлена на Рисунке 22.

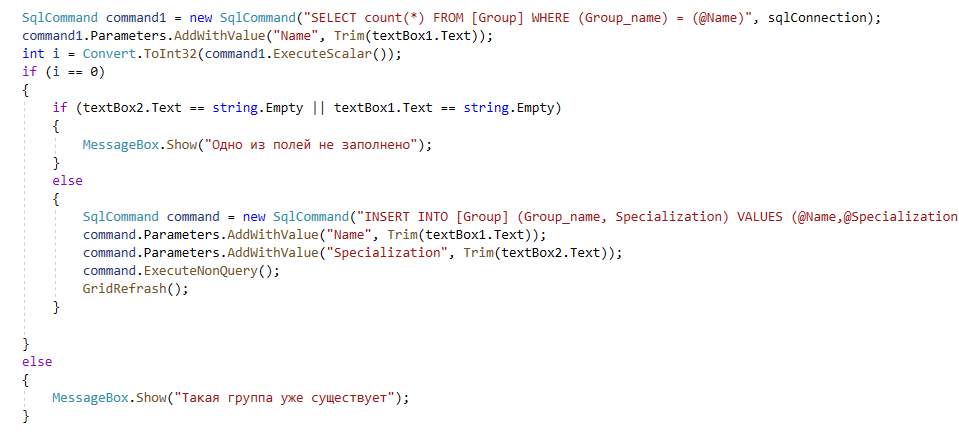


Рисунок 22 – Код проверки наличия группы в базе данных

1. Проверка полей на заполнение, проверка полей на допустимую символьную длину и проверка на существующий логин в базе данных представлена на Рисунке 23.



Рисунок 23 – Код проверки на заполнение, существующий логин и длину

1. Проверка папки на наличие представлена на Рисунке 24.

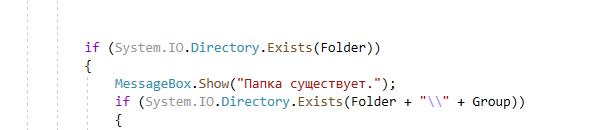


Рисунок 24 – Проверка существует ли папка по указанному пути

3.4 Характеристика программы

В программе имеются 13 авторских модулей. Ниже в Таблице 9 представлены характеристики программы.

Таблица 9 – Характеристика программы

| № | Наименование модуля | Описание | Количество строк | Размер модуля (Кб) |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Autorization.cs | Форма авторизации | 161 | 4 |
| 2 | Company.cs | Форма для работы с компаниями | 210 | 5 |
| 3 | Documet\_Flow.cs | Форма для работы с документооборотом | 484 | 12 |
| 4 | Groups.cs | Форма для работы с группами | 203 | 3 |
| 5 | Practice\_Dates.cs | Форма для работы с датами практик | 387 | 6 |
| 6 | Registration.cs | Форма регистрации | 176 | 13 |
| 7 | User.cs | Форма пользователя | 110 | 2 |
| 8 | Settings.cs | Форма настроек пользователя | 102 | 2 |
| 9 | Students.cs | Форма работы со студентами | 227 | 3 |
| 10 | Main.cs | Основная форма для перехода по всем остальным | 243 | 5 |
| 11 | Program.cs | Класс программы, хранящий настройки проекта | 24 | 1 |
| 12 | ThemeColor.cs | Класс программы ответственный за дизайн | 45 | 1 |
| 13 | Administrator.cs | Форма для изменения пароля ролей | 101 | 2 |

При разработке использовались следующие встроенные и дополнительные (импортированные библиотеки):

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

using System.Globalization;

using System.IO;

using Microsoft.Office.Interop.Word;

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы, было разработано программное обеспечения для упрощения процесса определения студента на производственную практику.

Программное обеспечение было разработано в среде разработки Visual Studio 2019. На этапе анализа требований к проекту были выделены цели и задачи проекта, а также было сформировано и подписано задание для выполнения дипломного проекта. На этапе проектирования были разработаны функциональная схема, структурная схема, схема интерфейса, схема тестирования, логическая модель базы данных, диаграмма классов. На этапе тестирования продукты были выявлены и решены ошибки.

До внедрения программы в процесс направления студента на производственную практику, вся работа по формированию и заполнению документов выполнялась вручную, что зачастую вело к появлению опечаток и неточностей в документах и снижению эффективности труда, при внедрении программы все ошибки и неточности свелись к минимуму, так как в программе была реализована защита от некорректных действий пользователя, что повлияло на сохранность данных и их целостность.

Благодаря тому, что программное обеспечение имеет удобный и понятный интерфейс, оно обеспечит наиболее быстрое направление студента на производственную практику.

СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ

1. ГОСТ 19.404-79 ЕСПД. Пояснительная записка. ПЕРЕИЗДАНИЕ Январь 2010 г. Дата обращения: 04.05.2021
2. ГОСТ Р 7.0.5 2008. Система стандартов по информации, библиотечному и издательскому делу. Дата обращения: 06.05.2021.
3. ГОСТ 34.602-89. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы, 1990
4. Ишкова, Э. А. Самоучитель С#. Начала программирования / Э.А. Ишкова. - М.: Наука и техника, 2013. - 496 c.
5. Эндрю Троелсен. Язык программирования C# 2010 и платформа .NET 4.0. – М.,Вильямс, 2010 г. .– 1392 с.
6. Андрей Шшковский. C# и платформы .NET. Библиотека программиста – СПБ.: Питер, 2005 – 796 с.: ил
7. Подбельский, В. В. Язык С#. Базовый курс / В.В. Подбельский. - М.: Финансы и статистика, Инфра-М, 2011. - 384 c
8. Добро пожаловать в Visual Studio – Режим доступа: <https://msdn.microsoft.com/ru-ru/library/dd831853.aspx> (дата обращения: 05.05.2021)
9. Библиотеки C# (Руководство по программированию на C#) – Режим доступа: <https://msdn.microsoft.com/ru-ru/library/ms228390(v=vs.90).aspx> (дата обращения: 22.05.2021)
10. Справочник C# – Режим доступа: <https://msdn.microsoft.com/ruru/library/618ayhy6.aspx> (дата обращения: 17.05.2021)
11. Библиотеки DLL [.NET, примеры на C#] – Режим доступа: <http://vbbook.ru/visual-c.net/biblioteki-dll--net-primeru-na-c/> (дата обращения: 25.04.2021)
12. Библиотеки DLL [.NET, примеры на C#] – Режим доступа: <http://vbbook.ru/visual-c.net/biblioteki-dll--net-primeru-na-c/> (дата обращения: 25.04.2021)
13. Метод Main () и другие методы (C# и Java) – Режим доступа: <https://msdn.microsoft.com/ru-ru/library/ms228506%28v=vs.90%29.aspx> (дата обращения: 14.05.2021)
14. Википедия — свободная энциклопедия – Режим доступа: <https://ru.wikipedia.org> (дата обращения: 24.04.2021).
15. Библиотека классов .NET Framework – Режим доступа: <https://msdn.microsoft.com/library/ms229335(v=vs.90).aspx> (дата обращения: 04.05.2021)
16. Пространство имен System – Режим доступа: <https://msdn.microsoft.com/ruru/library/system(v=vs.110).aspx> (дата обращения: 10.05.2021)
17. METANIT.COM - Сайт о программировании – Режим доступа: <https://metanit.com> (дата обращения: 29.04.2021)
18. Руководство по языку C# | Microsoft Docs – Режим доступа: <https://docs.microsoft.com> (дата обращения: 06.05.2021)
19. Форум для программистов «Киберфорум» – Режим доступа: [www.cyberforum.ru](http://www.cyberforum.ru) (дата обращения: 15.05.2021)
20. Stack Overflow – Режим доступа: <https://ru.stackoverflow.com/> (дата обращения: 28.04.2021)

ПРИЛОЖЕНИЕ А. ТЕКСТ ПРОГРАММЫ

АННОТАЦИЯ

Данное приложение программной документации описывает существующие модуль программного продукта, а также его текст и вносит ясность в механизмы его работы.

Приложение состоит их разделов: «Модули программы» и «Текст программы».

1. Модули программы

Ниже представлена Таблица 1, в которой описаны все существующие модули программы. Программный продукт содержит в себе 13 авторских модулей.

Таблица 1 − Модули

|  |  |  |  |
| --- | --- | --- | --- |
| Номер модуля | Название модуля | Описание | Количество строк |
| 1 | 2 | 3 | 4 |
| 1 | Autorization.cs | Авторизация | 161 |
| 2 | Company.cs | Форма для работы с компаниями | 210 |
| 3 | Documet\_Flow.cs | Форма для работы с документооборотом | 484 |
| 4 | Groups.cs | Форма для работы с группами | 203 |
| 5 | Practice\_Dates.cs | Форма для работы с датами практик | 387 |
| 6 | Registration.cs | Регистрация | 176 |
| 7 | User.cs | Форма пользователя | 110 |
| 8 | Settings.cs | Форма настроек пользователя | 102 |
| 9 | Students.cs | Форма работы со студентами | 227 |
| 10 | Main.cs | Основная форма для перехода по всем остальным | 243 |
| 11 | Program.cs | Класс программы, хранящий настройки проекта | 24 |
| 12 | ThemeColor.cs | Класс программы ответственный за дизайн | 45 |
| 13 | Administrator.cs | Форма для изменения пароля ролей | 101 |

2. Текст программы

Ниже представлен код каждого из модулей.

Autorization.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Runtime.InteropServices;

using System.Data.SqlClient;

using System.Configuration;

namespace WKR.Forms

{

public partial class Autorization : Form

{

public static int ID;

private SqlConnection sqlConnection = null;

public Autorization()

{

InitializeComponent();

}

[DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]

private extern static void ReleaseCapture();

[DllImport("user32.DLL", EntryPoint = "SendMessage")]

private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int wParam, int lParam);

private void Enter\_Load(object sender, EventArgs e)

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString);

sqlConnection.Open();

if (sqlConnection.State == ConnectionState.Open)

{

}

this.buttonAutorization.FlatAppearance.BorderSize = 0;

this.buttonAutorization.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.buttonRegistration.FlatAppearance.BorderSize = 0;

this.buttonRegistration.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

textBoxLogin.Text="Логин";

textBoxPassword.Text = "Пароль";

}

private void buttonAutorization\_Click(object sender, EventArgs e)

{

//Запись id в переменную.

if (textBoxLogin.Text == "Логин" || textBoxPassword.Text == "Пароль")

{

MessageBox.Show("Пароль или логин введены некорректно.");

}

else

{

string Role = null;

SqlDataAdapter adapter = new SqlDataAdapter();

DataTable table = new DataTable();

SqlCommand command = new SqlCommand("select \* from [Users] where [Login] = @uL and [Password] = @uP ", sqlConnection);

command.Parameters.AddWithValue("@uL", SqlDbType.VarChar).Value = textBoxLogin.Text;

command.Parameters.AddWithValue("@uP", SqlDbType.VarChar).Value = textBoxPassword.Text;

adapter.SelectCommand = command;

adapter.Fill(table);

SqlCommand returnRole = new SqlCommand

{

CommandText = $"select [Role] from [Users] where [Login] = '{textBoxLogin.Text}' AND [Password] = '{textBoxPassword.Text}'",

Connection = sqlConnection

};

if (returnRole.ExecuteScalar() != null)

{

Role = returnRole.ExecuteScalar().ToString();

}

if (table.Rows.Count > 0)

{

if (Role == "Администратор")

{

Administrator administrator = new Administrator();

this.Visible = false;

administrator.Visible = true;

}

if (Role == "Сотрудник")

{

SqlCommand commandID = new SqlCommand("SELECT (id) FROM [Users] where [Login] = @uL and [Password] = @uP", sqlConnection);

commandID.Parameters.AddWithValue("@uL", SqlDbType.VarChar).Value = textBoxLogin.Text;

commandID.Parameters.AddWithValue("@uP", SqlDbType.VarChar).Value = textBoxPassword.Text;

commandID.Parameters.AddWithValue("Login", textBoxLogin.Text);

ID = Convert.ToInt32(commandID.ExecuteScalar());

Main form1 = new Main();

this.Visible = false;

form1.Visible = true;

}

}

else

{

MessageBox.Show("Введены неправильные данные");

}

}

}

private void buttonRegistration\_Click(object sender, EventArgs e)

{

Forms.Registration registration = new Forms.Registration();

this.Visible = false;

registration.Visible = true;

}

private void textBoxLogin\_MouseClick(object sender, MouseEventArgs e)

{

if (textBoxLogin.Text == "Логин")

{

textBoxLogin.Text = "";

}

}

private void textBoxPassword\_MouseClick(object sender, MouseEventArgs e)

{

if (textBoxPassword.Text == "Пароль")

{

textBoxPassword.Text = "";

}

}

private void button1\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button\_Click(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Minimized;

}

private void textBoxPassword\_Click(object sender, EventArgs e)

{

if(textBoxLogin.Text == "")

{

textBoxLogin.Text = "Логин";

}

}

private void textBoxPassword\_TextChanged(object sender, EventArgs e)

{

textBoxPassword.PasswordChar = '\*';

}

private void Autorization\_MouseDown(object sender, MouseEventArgs e)

{

ReleaseCapture();

SendMessage(this.Handle, 0x112, 0xf012, 0);

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

}

}

Company.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

namespace WKR.Forms

{

public partial class Company : Form

{

private string id;

private SqlConnection sqlConnection = null;

public Company()

{

InitializeComponent();

GridRefrash();

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void label7\_Click(object sender, EventArgs e)

{

}

public void GridRefrash()

{

using (SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString))

{

string a = "Select\* from [Company]";

SqlDataAdapter da = new SqlDataAdapter(a, conn);

DataSet ds = new DataSet();

da.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns["ID"].Visible = false;

dataGridView1.DefaultCellStyle.ForeColor = System.Drawing.Color.Black;

dataGridView1.Columns[1].HeaderText = "Название компании";

dataGridView1.Columns[2].HeaderText = "Фамилия руководителя";

dataGridView1.Columns[3].HeaderText = "Имя руководителя";

dataGridView1.Columns[4].HeaderText = "Отчество руководителя";

dataGridView1.Columns[5].HeaderText = "Должность руководителя";

dataGridView1.Columns[6].HeaderText = "Фамилия руководителя в дательном падеже";

dataGridView1.Columns[7].HeaderText = "Имя руководителя в дательном падеже";

dataGridView1.Columns[8].HeaderText = "Отчество руководителя в дательном падеже";

dataGridView1.Columns[9].HeaderText = "Почта";

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.HeaderCell.Style.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

}

}

private void Company\_Load(object sender, EventArgs e)

{

LoadTheme();

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString);

sqlConnection.Open();

if (sqlConnection.State == ConnectionState.Open)

{

}

}

private void LoadTheme()

{

foreach (Control buttons in this.Controls)

{

if (buttons.GetType() == typeof(Button))

{

Button btn = (Button)buttons;

btn.BackColor = ThemeColor.PrimaryColor;

btn.ForeColor = Color.Black;

btn.FlatAppearance.BorderColor = ThemeColor.SecondaryColor;

}

}

}

private void buttonDelete\_Click(object sender, EventArgs e)

{

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Students] WHERE (Practice\_company) = (@id)", sqlConnection);

command1.Parameters.AddWithValue("id", id);

int i = Convert.ToInt32(command1.ExecuteScalar());

if (i == 0)

{

if (id == null)

{

MessageBox.Show("Одно из полей не заполнено");

}

else

{

SqlCommand command = new SqlCommand($"Delete from [Company] where [id]={id}", sqlConnection);

command.ExecuteNonQuery();

GridRefrash();

}

}

else

{

MessageBox.Show("Компанию невозможно удалить, так как в ней проходят практику студенты.");

}

}

private void buttonAdd\_Click(object sender, EventArgs e)

{

if (Trim(textBox2.Text) == string.Empty || Trim(textBox3.Text) ==string.Empty|| Trim(textBox4.Text) == string.Empty|| Trim(textBox5.Text) == string.Empty || Trim(textBox6.Text) == string.Empty || Trim(textBox7.Text) == string.Empty || Trim(textBox8.Text) == string.Empty || Trim(textBox9.Text) == string.Empty || Trim(textBox10.Text) == string.Empty)

{

MessageBox.Show("Данные введены не верно.");

}

else

{

if (textBox2.Text == string.Empty || textBox3.Text == string.Empty || textBox4.Text == string.Empty || textBox6.Text == string.Empty || textBox7.Text == string.Empty || textBox8.Text == string.Empty || textBox10.Text == string.Empty)

{

MessageBox.Show("Одно из полей не заполнено");

}

else

{

SqlCommand command = new SqlCommand("INSERT INTO [Company] (Name,[Manager\_Surname],[Manager\_Name],[Manager\_Patronymic],[Manager\_Position],[Manager\_surname\_in\_the\_dative\_case],[Manager\_name\_in\_the\_dative\_case],[Manager\_patronymic\_in\_the\_dative\_case],Email) " +

"VALUES (@Name,@Manager\_s\_Surname,@Manager\_s\_Name,@Manager\_s\_Patronymic,@Manager\_s\_Position,@Manager\_s\_surname\_in\_the\_dative\_case,@Manager\_s\_name\_in\_the\_dative\_case,@Manager\_s\_patronymic\_in\_the\_dative\_case,@Email)", sqlConnection);

command.Parameters.AddWithValue("Name", Trim(textBox2.Text));

command.Parameters.AddWithValue("Manager\_s\_Surname", Trim(textBox3.Text));

command.Parameters.AddWithValue("Manager\_s\_Name", Trim(textBox4.Text));

command.Parameters.AddWithValue("Manager\_s\_Patronymic", Trim(textBox5.Text));

command.Parameters.AddWithValue("Manager\_s\_Position", Trim(textBox6.Text));

command.Parameters.AddWithValue("Manager\_s\_surname\_in\_the\_dative\_case", Trim(textBox7.Text));

command.Parameters.AddWithValue("Manager\_s\_name\_in\_the\_dative\_case", Trim(textBox8.Text));

command.Parameters.AddWithValue("Manager\_s\_patronymic\_in\_the\_dative\_case", Trim(textBox9.Text));

command.Parameters.AddWithValue("Email", Trim(textBox10.Text));

command.ExecuteNonQuery();

GridRefrash();

}

}

}

string Trim(string s)

{

s = s.Trim();

return s;

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridView1\_MouseClick(object sender, MouseEventArgs e)

{

int r = 0;

r = dataGridView1.CurrentRow.Index;

id = dataGridView1.Rows[r].Cells[0].Value.ToString();

textBox2.Text = dataGridView1.Rows[r].Cells[1].Value.ToString();

textBox3.Text = dataGridView1.Rows[r].Cells[2].Value.ToString();

textBox4.Text = dataGridView1.Rows[r].Cells[3].Value.ToString();

textBox5.Text = dataGridView1.Rows[r].Cells[4].Value.ToString();

textBox6.Text = dataGridView1.Rows[r].Cells[5].Value.ToString();

textBox7.Text = dataGridView1.Rows[r].Cells[6].Value.ToString();

textBox8.Text = dataGridView1.Rows[r].Cells[7].Value.ToString();

textBox9.Text = dataGridView1.Rows[r].Cells[8].Value.ToString();

textBox10.Text = dataGridView1.Rows[r].Cells[9].Value.ToString();

}

private void buttonChange\_Click(object sender, EventArgs e)

{

if (Trim(textBox2.Text) == string.Empty || Trim(textBox3.Text) == string.Empty || Trim(textBox4.Text) == string.Empty || Trim(textBox5.Text) == string.Empty || Trim(textBox6.Text) == string.Empty || Trim(textBox7.Text) == string.Empty || Trim(textBox8.Text) == string.Empty || Trim(textBox9.Text) == string.Empty || Trim(textBox10.Text) == string.Empty)

{

MessageBox.Show("Данные введены не верно.");

}

else

{

if (textBox2.Text == string.Empty || textBox3.Text == string.Empty || textBox4.Text == string.Empty || textBox6.Text == string.Empty || textBox7.Text == string.Empty || textBox8.Text == string.Empty || textBox10.Text == string.Empty)

{

MessageBox.Show("Одно из полей не заполнено");

}

else

{

SqlCommand command = new SqlCommand($"UPDATE [Company] set [Name]=@Name, [Manager\_Surname]=@Manager\_s\_Surname, [Manager\_Name]=@Manager\_s\_Name, [Manager\_Patronymic]=@Manager\_s\_Patronymic," +

$" [Manager\_Position]=@Manager\_s\_Position ,[Manager\_surname\_in\_the\_dative\_case]=@Manager\_s\_surname\_in\_the\_dative\_case,[Manager\_name\_in\_the\_dative\_case]=@Manager\_s\_name\_in\_the\_dative\_case ,[Manager\_patronymic\_in\_the\_dative\_case]=@Manager\_s\_patronymic\_in\_the\_dative\_case ,[Email]=@Email WHERE [id]=@id", sqlConnection);

command.Parameters.AddWithValue("Name", Trim(textBox2.Text));

command.Parameters.AddWithValue("Manager\_s\_Surname", Trim(textBox3.Text));

command.Parameters.AddWithValue("Manager\_s\_Name", Trim(textBox4.Text));

command.Parameters.AddWithValue("Manager\_s\_Patronymic", Trim(textBox5.Text));

command.Parameters.AddWithValue("Manager\_s\_Position", Trim(textBox6.Text));

command.Parameters.AddWithValue("Manager\_s\_surname\_in\_the\_dative\_case", Trim(textBox7.Text));

command.Parameters.AddWithValue("Manager\_s\_name\_in\_the\_dative\_case", Trim(textBox8.Text));

command.Parameters.AddWithValue("Manager\_s\_patronymic\_in\_the\_dative\_case", Trim(textBox9.Text));

command.Parameters.AddWithValue("Email", Trim(textBox10.Text));

command.Parameters.AddWithValue("id", id);

command.ExecuteNonQuery();

GridRefrash();

}

}

}

}

}

Documents\_Flow.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

using System.Globalization;

using System.IO;

using Microsoft.Office.Interop.Word;

using Word = Microsoft.Office.Interop.Word;

using GemBox.Document;

namespace WKR.Forms

{

public partial class Documents\_Flow : Form

{

// Переменные Студентов:

string id\_student;

string Surname;

string Name;

string Patronymic;

string Group;

string Birthday;

string Email\_student;

string Practice\_company;

string Surname\_in\_the\_accusative\_case;

string First\_name\_in\_the\_accusative\_case;

string Patronymic\_in\_the\_accusative\_case;

string Specialization;

// Переменные Компаний:

string id\_company;

string Company\_name;

string Manager\_Surname;

string Manager\_Name;

string Manager\_Patronymic;

string Manager\_Position;

string Manager\_surname\_in\_the\_dative\_case;

string Manager\_name\_in\_the\_dative\_case;

string Manager\_patronymic\_in\_the\_dative\_case;

string Email;

// Переменные Настроек:

string Folder;

string Document\_template\_storage\_location;

string Document2\_template\_storage\_location;

private SqlConnection sqlConnection = null;

public Documents\_Flow()

{

InitializeComponent();

StudentsGridRefrash();

CompanyGridRefrash();

}

private void label1\_Click(object sender, EventArgs e)

{

}

public void StudentsGridRefrash()

{

using (SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString))

{

string a = "Select \* from [Students\_Company\_Group]";

SqlDataAdapter da = new SqlDataAdapter(a, conn);

DataSet ds = new DataSet();

da.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns["ID"].Visible = false;

dataGridView1.Columns["Specialization"].Visible = false;

dataGridView1.DefaultCellStyle.ForeColor = System.Drawing.Color.Black;

dataGridView1.Columns[1].HeaderText = "Фамилия студента";

dataGridView1.Columns[2].HeaderText = "Имя студента";

dataGridView1.Columns[3].HeaderText = "Отчество студента";

dataGridView1.Columns[4].HeaderText = "Группа";

dataGridView1.Columns[5].HeaderText = "Дата рождения";

dataGridView1.Columns[6].HeaderText = "Почта";

dataGridView1.Columns[7].HeaderText = "Компания в которой проходит практику";

dataGridView1.Columns[8].HeaderText = "Фамилия в винительном падеже";

dataGridView1.Columns[9].HeaderText = "Имя в винительном падеже";

dataGridView1.Columns[10].HeaderText = "Отчество в винительном падеже";

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.HeaderCell.Style.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

}

}

private void LoadTheme()

{

foreach (System.Windows.Forms.Control buttons in this.Controls)

{

if (buttons.GetType() == typeof(System.Windows.Forms.Button))

{

System.Windows.Forms.Button btn = (System.Windows.Forms.Button)buttons;

btn.BackColor = ThemeColor.PrimaryColor;

btn.ForeColor = System.Drawing.Color.Black;

btn.FlatAppearance.BorderColor = ThemeColor.SecondaryColor;

}

}

}

public void CompanyGridRefrash()

{

using (SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString))

{

string a = "Select\* from [Company]";

SqlDataAdapter da = new SqlDataAdapter(a, conn);

DataSet ds = new DataSet();

da.Fill(ds);

dataGridView2.DataSource = ds.Tables[0];

dataGridView2.Columns["ID"].Visible = false;

dataGridView2.DefaultCellStyle.ForeColor = System.Drawing.Color.Black;

dataGridView2.Columns[1].HeaderText = "Название компании";

dataGridView2.Columns[2].HeaderText = "Фамилия руководителя";

dataGridView2.Columns[3].HeaderText = "Имя руководителя";

dataGridView2.Columns[4].HeaderText = "Отчество руководителя";

dataGridView2.Columns[5].HeaderText = "Должность руководителя";

dataGridView2.Columns[6].HeaderText = "Имя руководителя в дательном падеже";

dataGridView2.Columns[7].HeaderText = "Отчество руководителя в дательном падеже";

dataGridView2.Columns[8].HeaderText = "Почта";

foreach (DataGridViewColumn column in dataGridView2.Columns)

{

column.HeaderCell.Style.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

}

}

private void dataGridView2\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void Documents\_Flow\_Load(object sender, EventArgs e)

{

LoadTheme();

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString);

sqlConnection.Open();

if (sqlConnection.State == ConnectionState.Open)

{

}

}

private void dataGridView1\_SizeChanged(object sender, EventArgs e)

{

}

private void buttonCreate\_Click(object sender, EventArgs e)

{

if (id\_student == null || id\_company == null)

{

MessageBox.Show("Выберите студента или компанию.");

}

else

{

SqlCommand commandID1 = new SqlCommand("SELECT (Document\_template\_storage\_location) FROM [Settings] where [id] = @id ", sqlConnection);

commandID1.Parameters.AddWithValue("@id", SqlDbType.VarChar).Value = Autorization.ID;

Document\_template\_storage\_location = Convert.ToString(commandID1.ExecuteScalar());

SqlCommand commandID2 = new SqlCommand("SELECT (Document2\_template\_storage\_location) FROM [Settings] where [id] = @id ", sqlConnection);

commandID2.Parameters.AddWithValue("@id", SqlDbType.VarChar).Value = Autorization.ID;

Document2\_template\_storage\_location = Convert.ToString(commandID2.ExecuteScalar());

//Проверка на существующие данные в бд

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Documents\_Flow] WHERE (id\_Student) = (@id\_Student) AND (id\_Company) = (@id\_Company) AND (Document\_Type) = (@Document\_Type)", sqlConnection);

command1.Parameters.AddWithValue("id\_Student", id\_student);

command1.Parameters.AddWithValue("id\_Company", id\_company);

command1.Parameters.AddWithValue("Document\_Type", "Направление");

int i = Convert.ToInt32(command1.ExecuteScalar());

SqlCommand command2 = new SqlCommand("SELECT count(\*) FROM [Documents\_Flow] WHERE (id\_Student) = (@id\_Student) AND (id\_Company) = (@id\_Company) AND (Document\_Type) = (@Document\_Type)", sqlConnection);

command2.Parameters.AddWithValue("id\_Student", id\_student);

command2.Parameters.AddWithValue("id\_Company", id\_company);

command2.Parameters.AddWithValue("Document\_Type", "Принятие");

int j = Convert.ToInt32(command2.ExecuteScalar());

if (i == 0)

{

SqlCommand command = new SqlCommand("INSERT INTO [Documents\_Flow] (id\_Student, id\_Company , Document\_Type) VALUES (@id\_Student,@id\_Company,@Document\_Type)", sqlConnection);

command.Parameters.AddWithValue("id\_Student", id\_student);

command.Parameters.AddWithValue("id\_Company", id\_company);

command.Parameters.AddWithValue("Document\_Type", "Направление");

command.ExecuteNonQuery();

Document\_Create(Document\_template\_storage\_location, "Направление");

}

if (j == 0)

{

SqlCommand command = new SqlCommand("INSERT INTO [Documents\_Flow] (id\_Student, id\_Company , Document\_Type) VALUES (@id\_Student,@id\_Company,@Document\_Type)", sqlConnection);

command.Parameters.AddWithValue("id\_Student", id\_student);

command.Parameters.AddWithValue("id\_Company", id\_company);

command.Parameters.AddWithValue("Document\_Type", "Принятие");

command.ExecuteNonQuery();

Document\_Create(Document2\_template\_storage\_location, "Принятие");

}

else

{

Document\_Create(Document\_template\_storage\_location, "Направление");

Document\_Create(Document2\_template\_storage\_location, "Принятие");

}

}

}

private void Document\_Create(string file, string Type)

{

//Получение текущей даты

DateTime curDate = DateTime.Now;

var Chislo = curDate.Day.ToString();

var currentYear = DateTime.Now.Year;

var Year = curDate.Year.ToString();

string mymonth = DateTime.Now.ToString("MMMM", CultureInfo.CurrentCulture);

//Записываем студенту его компанию

SqlCommand command = new SqlCommand($"UPDATE [Students] set [Practice\_company]=@id\_company WHERE [id]=@id\_student", sqlConnection);

int r = 0;

r = dataGridView1.CurrentRow.Index;

command.Parameters.AddWithValue("id\_company", id\_company);

command.Parameters.AddWithValue("id\_student", id\_student);

command.ExecuteNonQuery();

StudentsGridRefrash();

//Получаем место хранения созданных файлов

SqlCommand commandID = new SqlCommand("SELECT (Document\_storage\_location) FROM [Settings] where [id] = @id ", sqlConnection);

commandID.Parameters.AddWithValue("@id", SqlDbType.VarChar).Value = Autorization.ID;

Folder = Convert.ToString(commandID.ExecuteScalar());

//Получаем id группы

SqlCommand commandID\_GROUP = new SqlCommand("SELECT (id) FROM [Group] WHERE (Group\_name) = (@Group)", sqlConnection);

commandID\_GROUP.Parameters.AddWithValue("Group", Group);

int Group\_id = Convert.ToInt32(commandID\_GROUP.ExecuteScalar());

//Считаем колличество записей в бд

SqlCommand commandGroup\_count = new SqlCommand("SELECT count(\*) FROM [Practice\_dates] WHERE (Group\_name) = (@Group)", sqlConnection);

commandGroup\_count.Parameters.AddWithValue("Group", Group\_id);

int Group\_count = Convert.ToInt32(commandGroup\_count.ExecuteScalar());

//Считываем количество записей в бд если название "Производсвенная практика"

SqlCommand commandGroup\_count\_pp = new SqlCommand("SELECT count(\*) FROM [Practice\_dates] WHERE (Group\_name) = (@Group) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

commandGroup\_count\_pp.Parameters.AddWithValue("Group", Group\_id);

commandGroup\_count\_pp.Parameters.AddWithValue("Pre\_graduate\_practice", "Производственная практика");

int Group\_count\_pp = Convert.ToInt32(commandGroup\_count\_pp.ExecuteScalar());

//Считываем количество записей в бд если название "Преддипломная практика"

SqlCommand commandGroup\_count\_pd = new SqlCommand("SELECT count(\*) FROM [Practice\_dates] WHERE (Group\_name) = (@Group) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

commandGroup\_count\_pd.Parameters.AddWithValue("Group", Group\_id);

commandGroup\_count\_pd.Parameters.AddWithValue("Pre\_graduate\_practice", "Преддипломная практика");

int Group\_count\_pd = Convert.ToInt32(commandGroup\_count\_pd.ExecuteScalar());

string a = "Select \* from [Practice\_dates]";

SqlDataAdapter da = new SqlDataAdapter(a, new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString));

System.Data.DataTable dt = new System.Data.DataTable();

da.Fill(dt);

int[] array\_PP = new int[Group\_count\_pp];

int[] array\_PD = new int[Group\_count\_pd];

string[,] arr = new string[Group\_count, 6];

if (System.IO.Directory.Exists(Folder))

{

MessageBox.Show("Папка существует.");

if (System.IO.Directory.Exists(Folder + "\\" + Group))

{

MessageBox.Show("Папка группы существует.");

var wordApp = new Word.Application();

wordApp.Visible = false;

var wordDocument = wordApp.Documents.Open(file);

Word.Table table = wordDocument.Tables[2];

var SimpleDt = new System.Data.DataTable();

int pp = 0;

int pd = 0;

//Запись в массив всех id определенной группы для каждого вида практик.

foreach (DataRow dr in dt.Rows)

{

if ((int)dr["Group\_name"] == Group\_id & dr["Pre\_graduate\_practice"].ToString() == "Преддипломная практика")

{

array\_PD[pd] = (int)dr["id"];

pd++;

}

}

foreach (DataRow dr in dt.Rows)

{

if ((int)dr["Group\_name"] == Group\_id & dr["Pre\_graduate\_practice"].ToString() == "Производственная практика")

{

array\_PP[pp] = (int)dr["id"];

pp++;

}

}

for (int i = 0; i < Group\_count - array\_PD.Length; i++)

{

//Получение данных для таблицы

//1

SqlCommand Professional\_module = new SqlCommand("SELECT (Professional\_module) FROM [Practice\_dates] WHERE (id) = (@id) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

Professional\_module.Parameters.AddWithValue("id", array\_PP[i]);

Professional\_module.Parameters.AddWithValue("Pre\_graduate\_practice", "Производственная практика");

string Professional\_modul = Convert.ToString(Professional\_module.ExecuteScalar());

//2

SqlCommand Name\_of\_the\_practice = new SqlCommand("SELECT (Name\_of\_the\_practice) FROM [Practice\_dates] WHERE (id) = (@id) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

Name\_of\_the\_practice.Parameters.AddWithValue("id", array\_PP[i]);

Name\_of\_the\_practice.Parameters.AddWithValue("Pre\_graduate\_practice", "Производственная практика");

string Name\_of\_the\_practic = Convert.ToString(Name\_of\_the\_practice.ExecuteScalar());

//3

SqlCommand Pactice\_start\_date = new SqlCommand("SELECT (Pactice\_start\_date) FROM [Practice\_dates] WHERE (id) = (@id) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

Pactice\_start\_date.Parameters.AddWithValue("id", array\_PP[i]);

Pactice\_start\_date.Parameters.AddWithValue("Pre\_graduate\_practice", "Производственная практика");

string Pactice\_start\_dat = Convert.ToString(Pactice\_start\_date.ExecuteScalar());

Pactice\_start\_dat = DateTime.Parse(Pactice\_start\_dat).ToShortDateString();

//4

SqlCommand Pactice\_termination\_date = new SqlCommand("SELECT (Pactice\_termination\_date) FROM [Practice\_dates] WHERE (id) = (@id) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

Pactice\_termination\_date.Parameters.AddWithValue("id", array\_PP[i]);

Pactice\_termination\_date.Parameters.AddWithValue("Pre\_graduate\_practice", "Производственная практика");

string Pactice\_termination\_dat = Convert.ToString(Pactice\_termination\_date.ExecuteScalar());

Pactice\_termination\_dat = DateTime.Parse(Pactice\_termination\_dat).ToShortDateString();

//5

SqlCommand Date\_of\_continuation\_of\_the\_practice = new SqlCommand("SELECT (Date\_of\_continuation\_of\_the\_practice) FROM [Practice\_dates] WHERE (id) = (@id) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

Date\_of\_continuation\_of\_the\_practice.Parameters.AddWithValue("id", array\_PP[i]);

Date\_of\_continuation\_of\_the\_practice.Parameters.AddWithValue("Pre\_graduate\_practice", "Производственная практика");

string Date\_of\_continuation\_of\_the\_practic = Convert.ToString(Date\_of\_continuation\_of\_the\_practice.ExecuteScalar());

Date\_of\_continuation\_of\_the\_practic = DateTime.Parse(Date\_of\_continuation\_of\_the\_practic).ToShortDateString();

//6

SqlCommand Pactice\_termination\_date\_2 = new SqlCommand("SELECT (Pactice\_termination\_date\_2) FROM [Practice\_dates] WHERE (id) = (@id) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

Pactice\_termination\_date\_2.Parameters.AddWithValue("id", array\_PP[i]);

Pactice\_termination\_date\_2.Parameters.AddWithValue("Pre\_graduate\_practice", "Производственная практика");

string Pactice\_termination\_dat\_2 = Convert.ToString(Pactice\_termination\_date\_2.ExecuteScalar());

Pactice\_termination\_dat\_2 = DateTime.Parse(Pactice\_termination\_dat\_2).ToShortDateString();

//7

SqlCommand Date\_of\_continuation\_of\_the\_practice\_2 = new SqlCommand("SELECT (Date\_of\_continuation\_of\_the\_practice\_2) FROM [Practice\_dates] WHERE (id) = (@id) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

Date\_of\_continuation\_of\_the\_practice\_2.Parameters.AddWithValue("id", array\_PP[i]);

Date\_of\_continuation\_of\_the\_practice\_2.Parameters.AddWithValue("Pre\_graduate\_practice", "Производственная практика");

string Date\_of\_continuation\_of\_the\_practic\_2 = Convert.ToString(Date\_of\_continuation\_of\_the\_practice\_2.ExecuteScalar());

Date\_of\_continuation\_of\_the\_practic\_2 = DateTime.Parse(Date\_of\_continuation\_of\_the\_practic\_2).ToShortDateString();

//8

SqlCommand Practice\_end\_date = new SqlCommand("SELECT (Practice\_end\_date) FROM [Practice\_dates] WHERE (id) = (@id) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

Practice\_end\_date.Parameters.AddWithValue("id", array\_PP[i]);

Practice\_end\_date.Parameters.AddWithValue("Pre\_graduate\_practice", "Производственная практика");

string Practice\_end\_dat = Convert.ToString(Practice\_end\_date.ExecuteScalar());

Practice\_end\_dat = DateTime.Parse(Practice\_end\_dat).ToShortDateString();

//Создание таблицы

Word.Row row = table.Rows.Add();

row.Cells[1].Range.Text = Professional\_modul;

row.Cells[2].Range.Text = Name\_of\_the\_practic;

if (Pactice\_termination\_dat == "1900-01-01" & Date\_of\_continuation\_of\_the\_practic == "1900-01-01" & Pactice\_termination\_dat\_2 == "1900-01-01" & Date\_of\_continuation\_of\_the\_practic\_2 == "1900-01-01")

{

row.Cells[3].Range.Text = "c " + Pactice\_start\_dat + "\r\n" + "по" + Practice\_end\_dat;

}

else if (Pactice\_termination\_dat\_2 == "1900-01-01" & Date\_of\_continuation\_of\_the\_practic\_2 == "1900-01-01")

{

row.Cells[3].Range.Text = "c " + Pactice\_start\_dat + "\r\n" + "по" + Pactice\_termination\_dat + "\r\n" + "c " + Date\_of\_continuation\_of\_the\_practic + "\r\n" + "по" + Practice\_end\_dat;

}

else

{

row.Cells[3].Range.Text = "c " + Pactice\_start\_dat + "\r\n" + "по" + Pactice\_termination\_dat + "\r\n" + "c " + Date\_of\_continuation\_of\_the\_practic + "\r\n" + "по" + Pactice\_termination\_dat\_2 + "\r\n" + "c " + Date\_of\_continuation\_of\_the\_practic\_2 + "\r\n" + "по" + Practice\_end\_dat;

}

}

for (int i = 0; i < Group\_count - array\_PP.Length; i++)

{

SqlCommand Pactice\_start\_date = new SqlCommand("SELECT (Pactice\_start\_date) FROM [Practice\_dates] WHERE (id) = (@id) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

Pactice\_start\_date.Parameters.AddWithValue("id", array\_PP[i]);

Pactice\_start\_date.Parameters.AddWithValue("Pre\_graduate\_practice", "Производственная практика");

string Pactice\_start\_dat = Convert.ToString(Pactice\_start\_date.ExecuteScalar());

Pactice\_start\_dat = DateTime.Parse(Pactice\_start\_dat).ToShortDateString();

SqlCommand Practice\_end\_date = new SqlCommand("SELECT (Practice\_end\_date) FROM [Practice\_dates] WHERE (id) = (@id) AND (Pre\_graduate\_practice) = (@Pre\_graduate\_practice)", sqlConnection);

Practice\_end\_date.Parameters.AddWithValue("id", array\_PP[i]);

Practice\_end\_date.Parameters.AddWithValue("Pre\_graduate\_practice", "Производственная практика");

string Practice\_end\_dat = Convert.ToString(Practice\_end\_date.ExecuteScalar());

Practice\_end\_dat = DateTime.Parse(Practice\_end\_dat).ToShortDateString();

Word.Row row = table.Rows.Add();

row.Cells[1].Range.Text = "Преддипломная практика";

row.Cells[3].Range.Text = "c " + Pactice\_start\_dat + "\r\n" + "по " + Practice\_end\_dat;

table.Rows[array\_PD.Length + array\_PP.Length + 2].Cells[1].Merge(table.Rows[array\_PD.Length + array\_PP.Length + 2].Cells[2]);

}

table.Cell(2, 3).Delete();

table.Cell(2, 2).Delete();

table.Cell(2, 1).Delete();

SqlCommand command2 = new SqlCommand("SELECT (id) FROM [Documents\_Flow] WHERE (id\_Student) = (@id\_Student) AND (id\_Company) = (@id\_Company) AND (Document\_Type) = (@Document\_Type)", sqlConnection);

command2.Parameters.AddWithValue("id\_Student", id\_student);

command2.Parameters.AddWithValue("id\_Company", id\_company);

command2.Parameters.AddWithValue("Document\_Type", Type);

string Number = Convert.ToString(command2.ExecuteScalar());

//Работа с шаблоном

ReplaceWordStub("{Number}",Number, wordDocument);

ReplaceWordStub("{Chislo}",Chislo, wordDocument);

ReplaceWordStub("{Month}",mymonth, wordDocument);

ReplaceWordStub("{Year}",Year, wordDocument);

ReplaceWordStub("{Dolz}", Manager\_Position, wordDocument);

ReplaceWordStub("{Company}", Company\_name, wordDocument);

ReplaceWordStub("{FioInVin}", Manager\_surname\_in\_the\_dative\_case + " " + Manager\_name\_in\_the\_dative\_case + " " + Manager\_patronymic\_in\_the\_dative\_case, wordDocument);

ReplaceWordStub("{FioWithoutSurname}", Manager\_Name + " " + Manager\_Patronymic, wordDocument);

ReplaceWordStub("{Specialization}", Specialization, wordDocument);

ReplaceWordStub("{NameInRod}", Surname\_in\_the\_accusative\_case + " " + First\_name\_in\_the\_accusative\_case + " " + Patronymic\_in\_the\_accusative\_case, wordDocument);

ReplaceWordStub("{year}", 2020.ToString(), wordDocument);

ReplaceWordStub("{year+1}", 2021.ToString(), wordDocument);

ReplaceWordStub("{Group}", Group, wordDocument);

//Сохранение документа по шаблону

string FIO = $"{Surname + " " + Name + " " + Patronymic + " " + Type}.doc";

wordDocument.SaveAs2(Folder + "\\" + Group + "\\" + FIO);

wordApp.Visible = true;

wordDocument.Application.Quit();

wordApp.Application.Quit();

}

else

{

MessageBox.Show("Папка группы не существует, папка была создана.");

Directory.CreateDirectory(Folder + "\\" + Group);

Document\_Create(file,Type);

}

}

else

{

MessageBox.Show("Папки не существует или она не указана в настройках.");

}

}

private void ReplaceWordStub(string stubToReplace, string text, Word.Document wordDocument)

{

var range = wordDocument.Content;

range.Find.ClearFormatting();

range.Find.Execute(FindText: stubToReplace, ReplaceWith: text);

}

private void dataGridView1\_MouseClick(object sender, MouseEventArgs e)

{

int r = 0;

r = dataGridView1.CurrentRow.Index;

id\_student = dataGridView1.Rows[r].Cells[0].Value.ToString();

Surname = dataGridView1.Rows[r].Cells[1].Value.ToString();

Name = dataGridView1.Rows[r].Cells[2].Value.ToString();

Patronymic = dataGridView1.Rows[r].Cells[3].Value.ToString();

Group = dataGridView1.Rows[r].Cells[4].Value.ToString();

Birthday = dataGridView1.Rows[r].Cells[5].Value.ToString();

Email\_student = dataGridView1.Rows[r].Cells[6].Value.ToString();

Practice\_company = dataGridView1.Rows[r].Cells[7].Value.ToString();

Surname\_in\_the\_accusative\_case = dataGridView1.Rows[r].Cells[8].Value.ToString();

First\_name\_in\_the\_accusative\_case = dataGridView1.Rows[r].Cells[9].Value.ToString();

Patronymic\_in\_the\_accusative\_case = dataGridView1.Rows[r].Cells[10].Value.ToString();

Specialization = dataGridView1.Rows[r].Cells[11].Value.ToString();

}

private void dataGridView2\_MouseClick(object sender, MouseEventArgs e)

{

int r = 0;

r = dataGridView2.CurrentRow.Index;

id\_company = dataGridView2.Rows[r].Cells[0].Value.ToString();

Company\_name = dataGridView2.Rows[r].Cells[1].Value.ToString();

Manager\_Surname = dataGridView2.Rows[r].Cells[2].Value.ToString();

Manager\_Name = dataGridView2.Rows[r].Cells[3].Value.ToString();

Manager\_Patronymic = dataGridView2.Rows[r].Cells[4].Value.ToString();

Manager\_Position = dataGridView2.Rows[r].Cells[5].Value.ToString();

Manager\_surname\_in\_the\_dative\_case = dataGridView2.Rows[r].Cells[6].Value.ToString();

Manager\_name\_in\_the\_dative\_case = dataGridView2.Rows[r].Cells[7].Value.ToString();

Manager\_patronymic\_in\_the\_dative\_case = dataGridView2.Rows[r].Cells[8].Value.ToString();

Email = dataGridView2.Rows[r].Cells[9].Value.ToString();

}

}

}

Groups.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

namespace WKR.Forms

{

public partial class Groups : Form

{

private string id;

private SqlConnection sqlConnection = null;

public Groups()

{

InitializeComponent();

GridRefrash();

}

public void GridRefrash()

{

using (SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString))

{

string a = "Select\* from [Group]";

SqlDataAdapter da = new SqlDataAdapter(a, conn);

DataSet ds = new DataSet();

da.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns["ID"].Visible = false;

dataGridView1.DefaultCellStyle.ForeColor = System.Drawing.Color.Black;

dataGridView1.Columns[1].HeaderText = "Наименование";

dataGridView1.Columns[2].HeaderText = "Специализация";

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.HeaderCell.Style.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

}

}

private void Groups\_Load(object sender, EventArgs e)

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString);

sqlConnection.Open();

if (sqlConnection.State == ConnectionState.Open)

{

}

LoadTheme();

}

private void LoadTheme()

{

foreach (Control buttons in this.Controls)

{

if (buttons.GetType() == typeof(Button))

{

Button btn = (Button)buttons;

btn.BackColor = ThemeColor.PrimaryColor;

btn.ForeColor = Color.Black;

btn.FlatAppearance.BorderColor = ThemeColor.SecondaryColor;

}

}

}

private void Groups\_FormClosing(object sender, FormClosingEventArgs e)

{

}

private void buttonAdd\_Click(object sender, EventArgs e)

{

if (Trim(textBox2.Text) == string.Empty || Trim(textBox1.Text) == string.Empty)

{

MessageBox.Show("Данные введены не верно.");

}

else

{

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Group] WHERE (Group\_name) = (@Name)", sqlConnection);

command1.Parameters.AddWithValue("Name", Trim(textBox1.Text));

int i = Convert.ToInt32(command1.ExecuteScalar());

if (i == 0)

{

if (textBox2.Text == string.Empty || textBox1.Text == string.Empty)

{

MessageBox.Show("Одно из полей не заполнено");

}

else

{

SqlCommand command = new SqlCommand("INSERT INTO [Group] (Group\_name, Specialization) VALUES (@Name,@Specialization)", sqlConnection);

command.Parameters.AddWithValue("Name", Trim(textBox1.Text));

command.Parameters.AddWithValue("Specialization", Trim(textBox2.Text));

command.ExecuteNonQuery();

GridRefrash();

}

}

else

{

MessageBox.Show("Такая группа уже существует");

}

}

}

string Trim(string s)

{

s = s.Trim();

return s;

}

private void buttonDelete\_Click(object sender, EventArgs e)

{

if (id == null)

{

MessageBox.Show("Выберите то что ходите удалить");

}

else

{

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Students] WHERE [Group] = (@id)", sqlConnection);

command1.Parameters.AddWithValue("id", id);

int i = Convert.ToInt32(command1.ExecuteScalar());

if (i == 0)

{

if (textBox2.Text == string.Empty || textBox1.Text == string.Empty)

{

MessageBox.Show("Одно из полей не заполнено");

}

else

{

SqlCommand command = new SqlCommand($"Delete from [Group] where [id]={id}", sqlConnection);

SqlCommand command2 = new SqlCommand($"Delete from [Practice\_dates] where [Group\_name]={id}", sqlConnection);

command2.ExecuteNonQuery();

command.ExecuteNonQuery();

GridRefrash();

}

}

else

{

MessageBox.Show("Группу невозможно удалить, так как в ней состоят студенты.");

}

}

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridView1\_MouseClick(object sender, MouseEventArgs e)

{

int r = 0;

r = dataGridView1.CurrentRow.Index;

textBox1.Text = dataGridView1.Rows[r].Cells[1].Value.ToString();

textBox2.Text = dataGridView1.Rows[r].Cells[2].Value.ToString();

id = dataGridView1.Rows[r].Cells[0].Value.ToString();

}

private void buttonChange\_Click(object sender, EventArgs e)

{

if (Trim(textBox2.Text) == string.Empty || Trim(textBox1.Text) == string.Empty)

{

MessageBox.Show("Данные введены не верно.");

}

else

{

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Group] WHERE (Group\_name) = (@Name)", sqlConnection);

command1.Parameters.AddWithValue("Name", Trim(textBox1.Text));

int i = Convert.ToInt32(command1.ExecuteScalar());

if (i == 0)

{

if (textBox2.Text == string.Empty || textBox1.Text == string.Empty)

{

MessageBox.Show("Одно из полей не заполнено");

}

else

{

SqlCommand command = new SqlCommand($"UPDATE [Group] set [Group\_name]=@Name, [Specialization]=@Specialization WHERE [id]=@id", sqlConnection);

command.Parameters.AddWithValue("Name", Trim(textBox1.Text));

command.Parameters.AddWithValue("Specialization", Trim(textBox2.Text));

command.Parameters.AddWithValue("id", id);

command.ExecuteNonQuery();

GridRefrash();

}

}

else

{

MessageBox.Show("Такая группа уже существует");

}

}

}

}

}

Practice\_dates.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

using System.Globalization;

namespace WKR.Forms

{

public partial class Practice\_Dates : Form

{

private string id;

private SqlConnection sqlConnection = null;

public Practice\_Dates()

{

InitializeComponent();

GridRefrash();

}

private void Practice\_Dates\_Load(object sender, EventArgs e)

{

LoadTheme();

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString);

sqlConnection.Open();

if (sqlConnection.State == ConnectionState.Open)

{

}

GridRefrash();

}

private void LoadTheme()

{

foreach (Control buttons in this.Controls)

{

if (buttons.GetType() == typeof(Button))

{

Button btn = (Button)buttons;

btn.BackColor = ThemeColor.PrimaryColor;

btn.ForeColor = Color.Black;

btn.FlatAppearance.BorderColor = ThemeColor.SecondaryColor;

}

}

}

public void GridRefrash()

{

using (SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString))

{

string a = "Select \* from [Practice\_dates\_Group]";

SqlDataAdapter da = new SqlDataAdapter(a, conn);

DataSet ds = new DataSet();

da.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns["id"].Visible = false;

dataGridView1.DefaultCellStyle.ForeColor = System.Drawing.Color.Black;

dataGridView1.Columns[1].HeaderText = "Название профессионального модуля";

dataGridView1.Columns[2].HeaderText = "Название практики";

dataGridView1.Columns[3].HeaderText = "Преддипломнаая ли практика?";

dataGridView1.Columns[4].HeaderText = "Группа";

dataGridView1.Columns[5].HeaderText = "Начало практики";

dataGridView1.Columns[6].HeaderText = "Остановка практики";

dataGridView1.Columns[7].HeaderText = "Продолжение практики";

dataGridView1.Columns[8].HeaderText = "Остановка практики";

dataGridView1.Columns[9].HeaderText = "Продолжение практики";

dataGridView1.Columns[10].HeaderText = "Окончание практики";

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.HeaderCell.Style.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

for (int i = 0; i < dataGridView1.RowCount; i++)

{

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

if (Convert.ToString(dataGridView1.Rows[i].Cells[j].Value) == "01.01.1900 0:00:00")

{

dataGridView1.Rows[i].Cells[j].Value = "";

}

}

}

SqlDataAdapter daa = new SqlDataAdapter(" Select(Group\_name) FROM[Group]", conn);

DataSet dss = new DataSet();

daa.Fill(dss);

comboBox1.DataSource = dss.Tables[0];

comboBox1.DisplayMember = "Group\_name";

}

}

private void buttonAdd\_Click(object sender, EventArgs e)

{

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Group] WHERE (Group\_name) = (@Name)", sqlConnection);

command1.Parameters.AddWithValue("Name", Trim(comboBox1.Text));

int i = Convert.ToInt32(command1.ExecuteScalar());

if (i == 0)

{

MessageBox.Show("Такой группы не существует");

}

else

{

SqlCommand command = new SqlCommand($"INSERT INTO [Practice\_dates] (Professional\_module,Name\_of\_the\_practice,Pre\_graduate\_practice,Group\_name,Pactice\_start\_date,Pactice\_termination\_date,Date\_of\_continuation\_of\_the\_practice,Pactice\_termination\_date\_2,Date\_of\_continuation\_of\_the\_practice\_2,Practice\_end\_date) " +

"VALUES (@Professional\_module,@Name\_of\_the\_practice,@Pre\_graduate\_practice,@Group\_name,@Pactice\_start\_date,@Pactice\_termination\_date,@Date\_of\_continuation\_of\_the\_practice,@Pactice\_termination\_date\_2,@Date\_of\_continuation\_of\_the\_practice\_2,@Practice\_end\_date)", sqlConnection);

SqlCommand Group\_id = new SqlCommand($"SELECT [id] FROM [Group] WHERE [Group\_name] = N'{Trim(comboBox1.Text)}'", sqlConnection);

var x = Group\_id.ExecuteScalar();

command.Parameters.AddWithValue("Group\_name", x);

command.Parameters.AddWithValue("Pactice\_start\_date", dateTimePicker1.Value);

command.Parameters.AddWithValue("Practice\_end\_date", dateTimePicker6.Value);

if (checkBox1.Checked)

{

// проверка на наличие преддипломной практики для этой группы

SqlCommand command4 = new SqlCommand("SELECT (id) FROM [Group] WHERE (Group\_name) = (@Name)", sqlConnection);

command4.Parameters.AddWithValue("Name", Trim(comboBox1.Text));

int ii = Convert.ToInt32(command4.ExecuteScalar());

SqlCommand command3 = new SqlCommand("SELECT count(\*) FROM [Practice\_dates] WHERE (Pre\_graduate\_practice) = (@Pre\_graduate\_practice) AND (Group\_name) = (@Group\_name)", sqlConnection);

command3.Parameters.AddWithValue("Pre\_graduate\_practice", "Преддипломная практика");

command3.Parameters.AddWithValue("Group\_name", ii);

int s = Convert.ToInt32(command3.ExecuteScalar());

if (s == 0)

{

command.Parameters.AddWithValue("Pactice\_termination\_date", "");

command.Parameters.AddWithValue("Date\_of\_continuation\_of\_the\_practice", "");

command.Parameters.AddWithValue("Pactice\_termination\_date\_2", "");

command.Parameters.AddWithValue("Date\_of\_continuation\_of\_the\_practice\_2", "");

var a = "Преддипломная практика";

command.Parameters.AddWithValue("Professional\_module", "");

command.Parameters.AddWithValue("Name\_of\_the\_practice", "");

command.Parameters.AddWithValue("Pre\_graduate\_practice", a);

command.ExecuteNonQuery();

}

else

{

MessageBox.Show("У группы может быть только одна преддипломная практика.");

}

}

else

{

command.Parameters.AddWithValue("Pactice\_termination\_date", dateTimePicker2.Value);

command.Parameters.AddWithValue("Date\_of\_continuation\_of\_the\_practice", dateTimePicker3.Value);

command.Parameters.AddWithValue("Pactice\_termination\_date\_2", dateTimePicker4.Value);

command.Parameters.AddWithValue("Date\_of\_continuation\_of\_the\_practice\_2", dateTimePicker5.Value);

if (Trim(textBox2.Text) == "")

{

MessageBox.Show("Нет названия профессионального модуля");

}

else if (Trim(textBox3.Text) == "")

{

MessageBox.Show("Нет названия практики");

}

else

{

var b = "Производственная практика";

command.Parameters.AddWithValue("Professional\_module", Trim(textBox2.Text));

command.Parameters.AddWithValue("Name\_of\_the\_practice", Trim(textBox3.Text));

command.Parameters.AddWithValue("Pre\_graduate\_practice", b);

command.ExecuteNonQuery();

}

}

GridRefrash();

}

}

private void buttonChange\_Click(object sender, EventArgs e)

{

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Group] WHERE (Group\_name) = (@Name)", sqlConnection);

command1.Parameters.AddWithValue("Name", Trim(comboBox1.Text));

int i = Convert.ToInt32(command1.ExecuteScalar());

if (i == 0)

{

MessageBox.Show("Такой группы не существует");

}

else

{

SqlCommand command = new SqlCommand($"INSERT INTO [Practice\_dates] (Professional\_module,Name\_of\_the\_practice,Pre\_graduate\_practice,Group\_name,Pactice\_start\_date,Pactice\_termination\_date,Date\_of\_continuation\_of\_the\_practice,Pactice\_termination\_date\_2,Date\_of\_continuation\_of\_the\_practice\_2,Practice\_end\_date) " +

"VALUES (@Professional\_module,@Name\_of\_the\_practice,@Pre\_graduate\_practice,@Group\_name,@Pactice\_start\_date,@Pactice\_termination\_date,@Date\_of\_continuation\_of\_the\_practice,@Pactice\_termination\_date\_2,@Date\_of\_continuation\_of\_the\_practice\_2,@Practice\_end\_date)", sqlConnection);

SqlCommand Group\_id = new SqlCommand($"SELECT [id] FROM [Group] WHERE [Group\_name] = N'{Trim(comboBox1.Text)}'", sqlConnection);

var x = Group\_id.ExecuteScalar();

command.Parameters.AddWithValue("Group\_name", x);

command.Parameters.AddWithValue("Pactice\_start\_date", dateTimePicker1.Value);

command.Parameters.AddWithValue("Practice\_end\_date", dateTimePicker6.Value);

if (checkBox1.Checked)

{

// проверка на наличие преддипломной практики для этой группы

SqlCommand command4 = new SqlCommand("SELECT (id) FROM [Group] WHERE (Group\_name) = (@Name)", sqlConnection);

command4.Parameters.AddWithValue("Name", Trim(comboBox1.Text));

int ii = Convert.ToInt32(command4.ExecuteScalar());

SqlCommand command3 = new SqlCommand("SELECT count(\*) FROM [Practice\_dates] WHERE (Pre\_graduate\_practice) = (@Pre\_graduate\_practice) AND (Group\_name) = (@Group\_name)", sqlConnection);

command3.Parameters.AddWithValue("Pre\_graduate\_practice", "Преддипломная практика");

command3.Parameters.AddWithValue("Group\_name", ii);

int s = Convert.ToInt32(command3.ExecuteScalar());

if (s == 0)

{

command.Parameters.AddWithValue("Pactice\_termination\_date", "");

command.Parameters.AddWithValue("Date\_of\_continuation\_of\_the\_practice", "");

command.Parameters.AddWithValue("Pactice\_termination\_date\_2", "");

command.Parameters.AddWithValue("Date\_of\_continuation\_of\_the\_practice\_2", "");

var a = "Преддипломная практика";

command.Parameters.AddWithValue("Professional\_module", "");

command.Parameters.AddWithValue("Name\_of\_the\_practice", "");

command.Parameters.AddWithValue("Pre\_graduate\_practice", a);

command.ExecuteNonQuery();

}

else

{

MessageBox.Show("У группы может быть только одна преддипломная практика.");

}

}

else

{

command.Parameters.AddWithValue("Pactice\_termination\_date", dateTimePicker2.Value);

command.Parameters.AddWithValue("Date\_of\_continuation\_of\_the\_practice", dateTimePicker3.Value);

command.Parameters.AddWithValue("Pactice\_termination\_date\_2", dateTimePicker4.Value);

command.Parameters.AddWithValue("Date\_of\_continuation\_of\_the\_practice\_2", dateTimePicker5.Value);

if (Trim(textBox2.Text) == "")

{

MessageBox.Show("Нет названия профессионального модуля");

}

else if (Trim(textBox3.Text) == "")

{

MessageBox.Show("Нет названия практики");

}

else

{

var b = "Производственная практика";

command.Parameters.AddWithValue("Professional\_module", Trim(textBox2.Text));

command.Parameters.AddWithValue("Name\_of\_the\_practice", Trim(textBox3.Text));

command.Parameters.AddWithValue("Pre\_graduate\_practice", b);

command.ExecuteNonQuery();

}

}

GridRefrash();

}

}

private void buttonDelete\_Click(object sender, EventArgs e)

{

if (id == null )

{

MessageBox.Show("Выберите что хотите удалить.");

}

else

{

SqlCommand command = new SqlCommand($"Delete from [Practice\_dates] where [id]={id}", sqlConnection);

command.ExecuteNonQuery();

GridRefrash();

}

}

string Trim(string s)

{

s = s.Trim();

return s;

}

private void dataGridView1\_MouseClick(object sender, MouseEventArgs e)

{

int r = 0;

r = dataGridView1.CurrentRow.Index;

id = dataGridView1.Rows[r].Cells[0].Value.ToString();

textBox2.Text = dataGridView1.Rows[r].Cells[1].Value.ToString();

textBox3.Text = dataGridView1.Rows[r].Cells[2].Value.ToString();

string a = dataGridView1.Rows[r].Cells[3].Value.ToString();

if (a == "Преддипломная практика")

{

checkBox1.Checked = true;

}

else if(a == "Производственная практика")

{

checkBox1.Checked = false;

}

dateTimePicker1.Value = (DateTime)new DateTimeConverter().ConvertFrom(dataGridView1.Rows[r].Cells[5].Value.ToString());

dateTimePicker6.Value = (DateTime)new DateTimeConverter().ConvertFrom(dataGridView1.Rows[r].Cells[10].Value.ToString());

comboBox1.Text = dataGridView1.Rows[r].Cells[4].Value.ToString();

if (Convert.ToString(dataGridView1.Rows[r].Cells[6].Value) == "01.01.0001 0:00:00")

{

dateTimePicker2.Value = (DateTime)new DateTimeConverter().ConvertFrom("01.01.1900");

}

else

{

dateTimePicker2.Value = (DateTime)new DateTimeConverter().ConvertFrom(dataGridView1.Rows[r].Cells[6].Value.ToString());

}

if (Convert.ToString(dataGridView1.Rows[r].Cells[7].Value) == "01.01.0001 0:00:00")

{

dateTimePicker3.Value = (DateTime)new DateTimeConverter().ConvertFrom("01.01.1900");

}

else

{

dateTimePicker3.Value = (DateTime)new DateTimeConverter().ConvertFrom(dataGridView1.Rows[r].Cells[7].Value.ToString());

}

if (Convert.ToString(dataGridView1.Rows[r].Cells[8].Value) == "01.01.0001 0:00:00")

{

dateTimePicker4.Value = (DateTime)new DateTimeConverter().ConvertFrom("01.01.1900");

}

else

{

dateTimePicker4.Value = (DateTime)new DateTimeConverter().ConvertFrom(dataGridView1.Rows[r].Cells[8].Value.ToString());

}

if (Convert.ToString(dataGridView1.Rows[r].Cells[9].Value) == "01.01.0001 0:00:00")

{

dateTimePicker5.Value = (DateTime)new DateTimeConverter().ConvertFrom("01.01.1900");

}

else

{

dateTimePicker5.Value = (DateTime)new DateTimeConverter().ConvertFrom(dataGridView1.Rows[r].Cells[9].Value.ToString());

}

if (checkBox1.Checked)

{

dateTimePicker2.Enabled = false;

dateTimePicker3.Enabled = false;

dateTimePicker4.Enabled = false;

dateTimePicker5.Enabled = false;

textBox2.Enabled = false;

textBox3.Enabled = false;

}

else

{

dateTimePicker2.Enabled = true;

dateTimePicker3.Enabled = true;

dateTimePicker4.Enabled = true;

dateTimePicker5.Enabled = true;

textBox2.Enabled = true;

textBox3.Enabled = true;

}

}

private void checkBox1\_Click(object sender, EventArgs e)

{

if (checkBox1.Checked)

{

dateTimePicker2.Enabled = false;

dateTimePicker3.Enabled = false;

dateTimePicker4.Enabled = false;

dateTimePicker5.Enabled = false;

textBox2.Enabled = false;

textBox3.Enabled = false;

}

else

{

dateTimePicker2.Enabled = true;

dateTimePicker3.Enabled = true;

dateTimePicker4.Enabled = true;

dateTimePicker5.Enabled = true;

textBox2.Enabled = true;

textBox3.Enabled = true;

}

}

private void label1\_Click(object sender, EventArgs e)

{

}

}

}

Registration.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Runtime.InteropServices;

using System.Data.SqlClient;

using System.Configuration;

namespace WKR.Forms

{

public partial class Registration : Form

{

private SqlConnection sqlConnection = null;

public Registration()

{

InitializeComponent();

}

[DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]

private extern static void ReleaseCapture();

[DllImport("user32.DLL", EntryPoint = "SendMessage")]

private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int wParam, int lParam);

private void buttonExit\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button\_Click(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Minimized;

}

string Trim(string s)

{

s = s.Trim();

return s;

}

private void Registration\_Load(object sender, EventArgs e)

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString);

sqlConnection.Open();

if (sqlConnection.State == ConnectionState.Open)

{

}

}

private void buttonRegistration\_Click(object sender, EventArgs e)

{

if (Trim(textBoxFIO.Text) == "" || Trim(textBoxLogin.Text) == "" || Trim(textBoxPassword.Text) == "" || Trim(textBoxEmail.Text) == "" || Trim(textBoxRole.Text) == "")

{

MessageBox.Show("Не все поля заполнены");

}

else

{

if (Trim(textBoxLogin.Text).Length <= 4 || Trim(textBoxPassword.Text).Length < 8)

{

MessageBox.Show("Пароль или логин меньше 8 и 4 символов соответственно");

}

else

{

String[] FIO = Trim(textBoxFIO.Text).Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

if (FIO.Length > 3 || FIO.Length == 1)

{

MessageBox.Show("Фио введено неправильно");

}

else

{

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Users] WHERE (Login) = (@Login)", sqlConnection);

command1.Parameters.AddWithValue("Login", Trim(textBoxLogin.Text));

int i = Convert.ToInt32(command1.ExecuteScalar());

if (i == 0)

{

SqlCommand command2 = new SqlCommand("INSERT INTO [Settings] (Document\_storage\_location,Document\_template\_storage\_location) VALUES (@Document\_storage\_location,@Document\_template\_storage\_location)", sqlConnection);

command2.Parameters.AddWithValue("Document\_storage\_location", "");

command2.Parameters.AddWithValue("Document\_template\_storage\_location", "");

SqlCommand command = new SqlCommand("INSERT INTO [Users] (Surname,Name,Patronymic,Email,Login,Password,Birthday,Settings,Role) VALUES (@Surname,@Name,@Partonymic,@Email,@Login,@Password,@Birthday,@Settings,@Role)", sqlConnection);

command.Parameters.AddWithValue("Surname", Trim(FIO[0]));

command.Parameters.AddWithValue("Name", Trim(FIO[1]));

if(FIO.Length == 3)

{

command.Parameters.AddWithValue("Partonymic", Trim(FIO[2]));

}

command.Parameters.AddWithValue("Email", Trim(textBoxEmail.Text));

command.Parameters.AddWithValue("Login", Trim(textBoxLogin.Text));

command.Parameters.AddWithValue("Password", Trim(textBoxPassword.Text));

command.Parameters.AddWithValue("Birthday", dateTimePicker1.Value);

SqlCommand command3 = new SqlCommand("SELECT count(\*) FROM [Role\_passwords] WHERE (Password) = (@Password)", sqlConnection);

command3.Parameters.AddWithValue("Password", Trim(textBoxRole.Text));

int role = Convert.ToInt32(command3.ExecuteScalar());

if (role == 0)

{

MessageBox.Show("Неправильный код роли");

}

else

{

SqlCommand command4 = new SqlCommand("SELECT (Name) FROM [Role\_passwords] WHERE (Password) = (@Password)", sqlConnection);

command4.Parameters.AddWithValue("Password", Trim(textBoxRole.Text));

string name = Convert.ToString(command4.ExecuteScalar());

if (name == "Администратор")

{

command2.ExecuteNonQuery();

SqlCommand command5 = new SqlCommand("SELECT MAX(id) FROM [Settings]", sqlConnection);

int settings = Convert.ToInt32(command5.ExecuteScalar());

command.Parameters.AddWithValue("Settings", settings);

command.Parameters.AddWithValue("Role", "Администратор");

command.ExecuteNonQuery();

MessageBox.Show("Вы зарегистрировались");

Forms.Autorization Autorization = new Forms.Autorization();

this.Visible = false;

Autorization.Visible = true;

}

else if (name == "Сотрудник")

{

command2.ExecuteNonQuery();

SqlCommand command5 = new SqlCommand("SELECT MAX(id) FROM [Settings]", sqlConnection);

int settings = Convert.ToInt32(command5.ExecuteScalar());

command.Parameters.AddWithValue("Settings", settings);

command.Parameters.AddWithValue("Role", "Сотрудник");

command.ExecuteNonQuery();

MessageBox.Show("Вы зарегистрировались");

Forms.Autorization Autorization = new Forms.Autorization();

this.Visible = false;

Autorization.Visible = true;

}

}

}

else

{

MessageBox.Show("Такой логин уже существует");

}

}

}

}

}

private void textBoxPassword\_TextChanged(object sender, EventArgs e)

{

textBoxPassword.PasswordChar = '\*';

}

private void Registration\_MouseDown(object sender, MouseEventArgs e)

{

ReleaseCapture();

SendMessage(this.Handle, 0x112, 0xf012, 0);

}

private void textBoxFIO\_TextChanged(object sender, EventArgs e)

{

}

private void button4\_Click(object sender, EventArgs e)

{

Forms.Autorization autorization = new Forms.Autorization();

this.Visible = false;

autorization.Visible = true;

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

}

}

User.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Runtime.InteropServices;

using System.Data.SqlClient;

using System.Configuration;

namespace WKR.Forms

{

public partial class User : Form

{

private SqlConnection sqlConnection = null;

public User()

{

InitializeComponent();

}

private void User\_Load(object sender, EventArgs e)

{

LoadTheme();

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString);

sqlConnection.Open();

if (sqlConnection.State == ConnectionState.Open)

{

}

}

private void LoadTheme()

{

foreach (Control buttons in this.Controls)

{

if (buttons.GetType() == typeof(Button))

{

Button btn = (Button)buttons;

btn.BackColor = ThemeColor.PrimaryColor;

btn.ForeColor = Color.Black;

btn.FlatAppearance.BorderColor = ThemeColor.SecondaryColor;

}

}

label1.ForeColor = ThemeColor.SecondaryColor;

}

private void button1\_Click(object sender, EventArgs e)

{

if(textBoxOldPassword.Text == "")

{

MessageBox.Show("Введите текущий пароль");

}

else

{

SqlCommand command1 = new SqlCommand("SELECT (Password) FROM [Users] WHERE (id) = (@id)", sqlConnection);

command1.Parameters.AddWithValue("id", Autorization.ID);

string i = Convert.ToString(command1.ExecuteScalar());

if (i == textBoxOldPassword.Text)

{

if (textBoxNewPassword.Text == "")

{

MessageBox.Show("Введите новый пароль");

}

else

{

if (textBoxNewPassword.Text.Length < 8)

{

MessageBox.Show("Новый пароль должен быть больше 8 символов");

}

else

{

if (textBoxOldPassword.Text == textBoxNewPassword.Text)

{

MessageBox.Show("Новый пароль должен отличаться от старого");

}

else

{

SqlCommand command = new SqlCommand($"UPDATE [Users] set [Password]=@Password WHERE [id]=@id", sqlConnection);

command.Parameters.AddWithValue("Password", textBoxNewPassword.Text);

command.Parameters.AddWithValue("id", Autorization.ID);

command.ExecuteNonQuery();

MessageBox.Show("Вы успешно поменяли пароль");

}

}

}

}

else

{

MessageBox.Show("Вы ввели неправильный пароль");

}

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

textBoxOldPassword.PasswordChar = '\*';

}

private void textBoxNewPassword\_TextChanged(object sender, EventArgs e)

{

textBoxNewPassword.PasswordChar = '\*';

}

}

}

Students.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

using System.Globalization;

namespace WKR.Forms

{

public partial class Students : Form

{

private string id;

private SqlConnection sqlConnection = null;

public Students()

{

InitializeComponent();

GridRefrash();

}

public void GridRefrash()

{

using (SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString))

{

string a = "Select \* from [Students\_Company\_Group]";

SqlDataAdapter da = new SqlDataAdapter(a, conn);

DataSet ds = new DataSet();

da.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns["ID"].Visible = false;

dataGridView1.Columns["Specialization"].Visible = false;

dataGridView1.DefaultCellStyle.ForeColor = System.Drawing.Color.Black;

dataGridView1.Columns[1].HeaderText = "Фамилия студента";

dataGridView1.Columns[2].HeaderText = "Имя студента";

dataGridView1.Columns[3].HeaderText = "Отчество студента";

dataGridView1.Columns[4].HeaderText = "Группа";

dataGridView1.Columns[5].HeaderText = "Дата рождения";

dataGridView1.Columns[6].HeaderText = "Почта";

dataGridView1.Columns[7].HeaderText = "Компания в которой проходит практику";

dataGridView1.Columns[8].HeaderText = "Фамилия в винительном падеже";

dataGridView1.Columns[9].HeaderText = "Имя в винительном падеже";

dataGridView1.Columns[10].HeaderText = "Отчество в винительном падеже";

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.HeaderCell.Style.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

SqlDataAdapter daa = new SqlDataAdapter(" Select(Group\_name) FROM[Group]", conn);

DataSet dss = new DataSet();

daa.Fill(dss);

comboBox1.DataSource = dss.Tables[0];

comboBox1.DisplayMember = "Group\_name";

}

}

private void buttonAdd\_Click(object sender, EventArgs e)

{

if (Trim(textBox2.Text) == string.Empty || Trim(textBox3.Text) == string.Empty || Trim(comboBox1.Text) == string.Empty || Trim(dateTimePicker1.Text) == string.Empty || Trim(textBox7.Text) == string.Empty || Trim(textBox9.Text) == string.Empty || Trim(textBox10.Text) == string.Empty )

{

MessageBox.Show("Одно из полей не заполнено");

}

else

{

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Group] WHERE (Group\_name) = (@Name)", sqlConnection);

command1.Parameters.AddWithValue("Name", Trim(comboBox1.Text));

int i = Convert.ToInt32(command1.ExecuteScalar());

if (i == 0)

{

MessageBox.Show("Такой группы не существует");

}

else

{

SqlCommand command = new SqlCommand($"INSERT INTO [Students] (Surname,Name,Patronymic,[Group],Birthday,Email,Surname\_in\_the\_accusative\_case,First\_name\_in\_the\_accusative\_case,Patronymic\_in\_the\_accusative\_case) " +

"VALUES (@Surname,@Name,@Patronymic,@Group,@Birthday,@Email,@Surname\_in\_the\_accusative\_case,@First\_name\_in\_the\_accusative\_case,@Patronymic\_in\_the\_accusative\_case)", sqlConnection);

SqlCommand Group\_id = new SqlCommand($"SELECT [id] FROM [Group] WHERE [Group\_name] = N'{Trim(comboBox1.Text)}'", sqlConnection);

var x = Group\_id.ExecuteScalar();

command.Parameters.AddWithValue("Surname", Trim(textBox2.Text));

command.Parameters.AddWithValue("Name", Trim(textBox3.Text));

command.Parameters.AddWithValue("Patronymic", Trim(textBox4.Text));

command.Parameters.AddWithValue("Group", x);

command.Parameters.AddWithValue("Birthday", dateTimePicker1.Value);

command.Parameters.AddWithValue("Email", Trim(textBox7.Text));

command.Parameters.AddWithValue("Surname\_in\_the\_accusative\_case", Trim(textBox9.Text));

command.Parameters.AddWithValue("First\_name\_in\_the\_accusative\_case", Trim(textBox10.Text));

command.Parameters.AddWithValue("Patronymic\_in\_the\_accusative\_case", Trim(textBox11.Text));

command.ExecuteNonQuery();

GridRefrash();

}

}

}

private void Students\_Load(object sender, EventArgs e)

{

LoadTheme();

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString);

sqlConnection.Open();

if (sqlConnection.State == ConnectionState.Open)

{

}

}

private void LoadTheme()

{

foreach (Control buttons in this.Controls)

{

if (buttons.GetType() == typeof(Button))

{

Button btn = (Button)buttons;

btn.BackColor = ThemeColor.PrimaryColor;

btn.ForeColor = Color.Black;

btn.FlatAppearance.BorderColor = ThemeColor.SecondaryColor;

}

}

}

private void dataGridView1\_MouseClick(object sender, MouseEventArgs e)

{

int r = 0;

r = dataGridView1.CurrentRow.Index;

id = dataGridView1.Rows[r].Cells[0].Value.ToString();

textBox2.Text = dataGridView1.Rows[r].Cells[1].Value.ToString();

textBox3.Text = dataGridView1.Rows[r].Cells[2].Value.ToString();

textBox4.Text = dataGridView1.Rows[r].Cells[3].Value.ToString();

comboBox1.Text = dataGridView1.Rows[r].Cells[4].Value.ToString();

dateTimePicker1.Text = dataGridView1.Rows[r].Cells[5].Value.ToString();

textBox7.Text = dataGridView1.Rows[r].Cells[6].Value.ToString();

textBox9.Text = dataGridView1.Rows[r].Cells[8].Value.ToString();

textBox10.Text = dataGridView1.Rows[r].Cells[9].Value.ToString();

textBox11.Text = dataGridView1.Rows[r].Cells[10].Value.ToString();

}

private void buttonDelete\_Click(object sender, EventArgs e)

{

if (id == null)

{

MessageBox.Show("Выберите то что хотите удалить.");

}

else

{

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Students] WHERE [Group] = (@id)", sqlConnection);

command1.Parameters.AddWithValue("id", id);

int i = Convert.ToInt32(command1.ExecuteScalar());

if (i == 0)

{

SqlCommand command2 = new SqlCommand($"Delete from [Documents\_Flow] where [id\_Student] = {id}", sqlConnection);

SqlCommand command = new SqlCommand($"Delete from [Students] where [id] = {id}", sqlConnection);

command2.ExecuteNonQuery();

command.ExecuteNonQuery();

GridRefrash();

}

else

{

MessageBox.Show("Группу невозможно удалить, так как в ней состоят студенты.");

}

}

}

string Trim(string s)

{

s = s.Trim();

return s;

}

private void buttonChange\_Click(object sender, EventArgs e)

{

if (Trim(textBox2.Text) == string.Empty || Trim(textBox3.Text) == string.Empty || Trim(textBox4.Text) == string.Empty || Trim(dateTimePicker1.Text) == string.Empty || Trim(textBox7.Text) == string.Empty || Trim(textBox10.Text) == string.Empty || Trim(textBox11.Text) == string.Empty || Trim(textBox9.Text) == string.Empty)

{

MessageBox.Show("Одно из полей не заполнено");

}

else

{

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Group] WHERE (Group\_name) = (@Name)", sqlConnection);

command1.Parameters.AddWithValue("Name", Trim(comboBox1.Text));

int i = Convert.ToInt32(command1.ExecuteScalar());

if (i == 0)

{

MessageBox.Show("Такой группы не сущетсвует");

}

else

{

SqlCommand command = new SqlCommand($"UPDATE [Students] set [Surname]=@Surname, [Name]=@Name, [Patronymic]=@Patronymic, [Group]=@Group," +

$" [Birthday]=@Birthday ,[Email]=@Email ,[Surname\_in\_the\_accusative\_case]=@Surname\_in\_the\_accusative\_case ,[First\_name\_in\_the\_accusative\_case]=@First\_name\_in\_the\_accusative\_case ,[Patronymic\_in\_the\_accusative\_case]=@Patronymic\_in\_the\_accusative\_case WHERE [id]=@id", sqlConnection);

SqlCommand Group\_id = new SqlCommand($"SELECT [id] FROM [Group] WHERE [Group\_name] = N'{Trim(comboBox1.Text)}'", sqlConnection);

var x = Group\_id.ExecuteScalar();

command.Parameters.AddWithValue("id", id);

command.Parameters.AddWithValue("Surname", Trim(textBox2.Text));

command.Parameters.AddWithValue("Name", Trim(textBox3.Text));

command.Parameters.AddWithValue("Patronymic", Trim(textBox4.Text));

command.Parameters.AddWithValue("Group", x);

command.Parameters.AddWithValue("Birthday", dateTimePicker1.Value);

command.Parameters.AddWithValue("Email", Trim(textBox7.Text));

command.Parameters.AddWithValue("Surname\_in\_the\_accusative\_case", Trim(textBox9.Text));

command.Parameters.AddWithValue("First\_name\_in\_the\_accusative\_case", Trim(textBox10.Text));

command.Parameters.AddWithValue("Patronymic\_in\_the\_accusative\_case", Trim(textBox11.Text));

command.ExecuteNonQuery();

GridRefrash();

}

}

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

}

}

}

Main.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Runtime.InteropServices;

using System.Data.SqlClient;

using System.Configuration;

namespace WKR

{

public partial class Main : Form

{

private Button currentButton;

private Random random;

private int index;

private Form activeForm;

public Main()

{

InitializeComponent();

random = new Random();

buttonClose.Visible = false;

this.Text = string.Empty;

this.ControlBox = false;

timer1.Start();

this.MaximizedBounds = Screen.FromHandle(this.Handle).WorkingArea;

}

[DllImport("user32.DLL", EntryPoint = "ReleaseCapture")]

private extern static void ReleaseCapture();

[DllImport("user32.DLL", EntryPoint = "SendMessage")]

private extern static void SendMessage(System.IntPtr hWnd, int wMsg, int wParam, int lParam);

private Color SelectColorForTheme()

{

int Index = random.Next(ThemeColor.ColorList.Count);

while (index == Index)

{

Index = random.Next(ThemeColor.ColorList.Count);

}

index = Index;

string color = ThemeColor.ColorList[index];

return ColorTranslator.FromHtml(color);

}

private void ButtonActivate(object btnSender)

{

if(btnSender!=null)

{

if(currentButton != (Button)btnSender)

{

DisableButton();

Color color = SelectColorForTheme();

currentButton = (Button)btnSender;

currentButton.BackColor = color;

currentButton.ForeColor = Color.Black;

currentButton.Font = new System.Drawing.Font("Monotype Corsiva", 23F, System.Drawing.FontStyle.Italic);

panelTitle.BackColor = color;

PanelLogo.BackColor = ThemeColor.ChangeColor(color, -0.2);

ThemeColor.PrimaryColor = color;

ThemeColor.SecondaryColor = ThemeColor.ChangeColor(color, -0.2);

buttonClose.Visible = true;

}

}

}

private void DisableButton()

{

foreach (Control previousBtn in panelMenu.Controls)

{

if (previousBtn.GetType() == typeof(Button))

{

previousBtn.BackColor = Color.FromArgb(51,51,76) ;

previousBtn.ForeColor = Color.White;

previousBtn.Font = new System.Drawing.Font("Monotype Corsiva", 18F, System.Drawing.FontStyle.Italic);

}

}

}

private void OpenChildForm(Form childForm, object btnSender)

{

if (activeForm != null)

activeForm.Close();

ButtonActivate(btnSender);

activeForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Fill;

this.DesktopPanel.Controls.Add(childForm);

this.DesktopPanel.Tag = childForm;

childForm.BringToFront();

childForm.Show();

TitleLabel.Text = childForm.Text;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void buttonUsers\_Click(object sender, EventArgs e)

{

OpenChildForm(new Forms.User(), sender);

}

private void buttonCompany\_Click(object sender, EventArgs e)

{

OpenChildForm(new Forms.Company(), sender);

}

private void buttonStudents\_Click(object sender, EventArgs e)

{

OpenChildForm(new Forms.Students(), sender);

}

private void buttonDate\_Click(object sender, EventArgs e)

{

OpenChildForm(new Forms.Practice\_Dates(), sender);

}

private void buttonSettings\_Click(object sender, EventArgs e)

{

OpenChildForm(new Forms.Settings(), sender);

}

private void buttonDoc\_Click(object sender, EventArgs e)

{

OpenChildForm(new Forms.Documents\_Flow(), sender);

}

private void buttonGroup\_Click(object sender, EventArgs e)

{

//ButtonActivate(sender);

OpenChildForm(new Forms.Groups(), sender);

//Forms.Groups groups = new Forms.Groups();

//groups.Visible = true

}

private void buttonClose\_Click(object sender, EventArgs e)

{

if (activeForm != null)

{

activeForm.Close();

Reset();

}

}

private void Reset()

{

DisableButton();

TitleLabel.Text = "Домашняя страница";

panelTitle.BackColor = Color.FromArgb(79, 40, 169);

PanelLogo.BackColor = Color.FromArgb(39,39,58);

currentButton = null;

buttonClose.Visible = false;

}

private void timer1\_Tick(object sender, EventArgs e)

{

/\*timer1.Stop();

Form1 formForm1 = new Form1();

this.Visible = false;

formForm1.Visible = true;\*/

}

private void panelTitle\_MouseDown(object sender, MouseEventArgs e)

{

ReleaseCapture();

SendMessage(this.Handle, 0x112, 0xf012, 0);

}

private void button1\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button2\_Click(object sender, EventArgs e)

{

if (WindowState == FormWindowState.Normal)

{

this.WindowState = FormWindowState.Maximized;

}

else

{

this.WindowState = FormWindowState.Normal;

}

}

private void button3\_Click(object sender, EventArgs e)

{

this.WindowState = FormWindowState.Minimized;

}

private void timer1\_Tick\_1(object sender, EventArgs e)

{

label1.Text = DateTime.Now.ToLongTimeString();

label2.Text = DateTime.Now.ToLongDateString();

}

private void DesktopPanel\_Paint(object sender, PaintEventArgs e)

{

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

}

private void panelTitle\_Paint(object sender, PaintEventArgs e)

{

}

private void button4\_Click(object sender, EventArgs e)

{

Forms.Autorization autorization= new Forms.Autorization();

this.Visible = false;

autorization.Visible = true;

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WKR

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Forms.Autorization());

//Application.Run(new Form1());

//Application.Run(new Forms.DataBaseConnect());

}

}

}

ThemeColor.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WKR

{

public static class ThemeColor

{

public static Color PrimaryColor { get; set; }

public static Color SecondaryColor { get; set; }

public static List<string> ColorList = new List<string>() { "#3F51B5","#009688","#FF5722","#607D8B","#FF9800","#9C27B0","#2196F3","#EA676C","#E41A4A","#5978BB","#018790","#0E3441","#00B0AD","#721D47","#EA4833","#EF937E","#F37521","#A12059",

"#126881","#8BC240","#364D5B","#C7DC5B","#0094BC","#E4126B","#43B76E","#7BCFE9","#B71C46"};

public static Color ChangeColor(Color color, double correctionFactor)

{

double red = color.R;

double green = color.G;

double blue = color.B;

if (correctionFactor < 0)

{

correctionFactor = 1 + correctionFactor;

red \*= correctionFactor;

green \*= correctionFactor;

blue \*= correctionFactor;

}

else

{

red = (255 - red) \* correctionFactor + red;

green = (255 - green) \* correctionFactor + green;

blue = (255 - blue) \* correctionFactor + blue;

}

return Color.FromArgb(color.A, (byte)red, (byte)green, (byte)blue);

}

}

}

Administrator.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

namespace WKR.Forms

{

public partial class Administrator : Form

{

private SqlConnection sqlConnection = null;

string i;

public Administrator()

{

InitializeComponent();

GridRefrash();

}

private void Administrator\_Load(object sender, EventArgs e)

{

sqlConnection = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString);

sqlConnection.Open();

if (sqlConnection.State == ConnectionState.Open)

{

}

}

public void GridRefrash()

{

using (SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["WKR"].ConnectionString))

{

string a = "Select\* from [Role\_passwords]";

SqlDataAdapter da = new SqlDataAdapter(a, conn);

DataSet ds = new DataSet();

da.Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns["ID"].Visible = false;

dataGridView1.DefaultCellStyle.ForeColor = System.Drawing.Color.Black;

dataGridView1.Columns[1].HeaderText = "Название роли";

dataGridView1.Columns[2].HeaderText = "Код роли";

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.HeaderCell.Style.Alignment = DataGridViewContentAlignment.MiddleCenter;

}

}

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

int r = 0;

r = dataGridView1.CurrentRow.Index;

i = dataGridView1.Rows[r].Cells[0].Value.ToString();

textBox1.Text = dataGridView1.Rows[r].Cells[2].Value.ToString();

textBox2.Text = dataGridView1.Rows[r].Cells[1].Value.ToString();

}

private void buttonChange\_Click(object sender, EventArgs e)

{

textBox1.Text = textBox1.Text.Trim();

if (textBox1.Text == string.Empty)

{

MessageBox.Show("Данные введены не верно");

}

else

{

SqlCommand command1 = new SqlCommand("SELECT count(\*) FROM [Role\_passwords] WHERE (Password) = (@Password)", sqlConnection);

command1.Parameters.AddWithValue("Password", textBox1.Text);

int b = Convert.ToInt32(command1.ExecuteScalar());

if (b == 0)

{

SqlCommand command = new SqlCommand($"UPDATE [Role\_passwords] set [Password]=@Password WHERE [id]=@id", sqlConnection);

command.Parameters.AddWithValue("id", i);

command.Parameters.AddWithValue("Password", textBox1.Text);

command.ExecuteNonQuery();

GridRefrash();

}

else

{

MessageBox.Show("Введенный пароль уже используется для другой роли.");

}

}

}

private void button4\_Click(object sender, EventArgs e)

{

Forms.Autorization autorization = new Forms.Autorization();

this.Visible = false;

autorization.Visible = true;

}

}

}

ПРИЛОЖЕНИЕ Б. СЦЕНАРИЙ И РЕЗУЛЬТАТЫ ТЕСТОВЫХ ИСПЫТАНИЙ

АННОТАЦИЯ

В данном программном документе приведены сценарии и результаты тестовых испытаний. Данное приложение включает раздел: «Описание» и «Тестирование».

1. Описание

В Таблице 1 предоставлены данные специалиста по тестированию.

Таблица 1 – Данные специалиста по тестированию

|  |  |  |
| --- | --- | --- |
| № | Данные | Описание |
| 1 | 2 | 3 |
| 1 | Название проекта | «WKR» |
| 2 | Номер версии | 1.0 |
| 3 | Имя тестера | Алпатов Никита |
| 4 | Даты тестирования | 01.05.2021 – 14.05.2021 |

1. Тестирование

Таблица 2 – Тестирование авторизации

| Test Case #1 | Тестирование авторизации |
| --- | --- |
| 1 | 2 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий. |
| Название тестирования/Имя | Авторизация. |
| Резюме испытания | Проверка добавления пользователя |
| Шаги тестирования | |  |  | | --- | --- | | № шага | Действие | | 1 | Заполнение всех актуальными данными пользователя. | | 2 | Нажатие на кнопку «Вход» | |
| Данные тестирования | Autorization.cs |
| Ожидаемый результат | Пользователь перенаправляется на форму выбора |
| Фактический результат | Пользователь перенаправляется на форму выбора |
| Предпосылки | Подключение к базе данных |
| Постусловия | - |
| Статус (Pass/Fail) | Pass. |

Таблица 3 – Тестирование регистрации нового пользователя

| Test Case #2 | Тестирование добавления пользователя |
| --- | --- |
| 1 | 2 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий. |
| Название тестирования/Имя | Регистрация нового пользователя. |
| Резюме испытания | Проверка добавления пользователя |
| Шаги тестирования | |  |  | | --- | --- | | № шага | Действие | | 1 | Заполнение всех полей данными пользователя. | | 2 | Нажатие на кнопку «Зарегистрироваться» | |
| Данные тестирования | Registration.cs |
| Ожидаемый результат | Запрос на регистрацию для выдачи роли |
| Фактический результат | Запрос на регистрацию для выдачи роли |
| Предпосылки | Подключение к базе данных |
| Постусловия | - |
| Статус (Pass/Fail) | Pass. |

Таблица 4 - Тестирование заполнение полей

| Test Case #3 | Тестирование заполнение полей |
| --- | --- |
| 1 | 2 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий. |
| Название тестирования/Имя | Заполнение всех полей при нажатии на datagrid соответствующей формы |
| Резюме испытания | Проверка отправки |
| Шаги тестирования | |  |  | | --- | --- | | № шага | Действие | | 1 | Заполнить все поля. | | 2 | Нажатие на кнопку отправить | |
| Данные тестирования | Company.cs, Documents\_Flow.cs, Groups.cs, Practice\_Dates.cs, Students.cs |
| Ожидаемый результат | Заполнение всех полей при нажатии на datagrid соответствующей формы |
| Фактический результат | Заполнение всех полей при нажатии на datagrid соответствующей формы |
| Предпосылки | Подключение к базе данных |
| Постусловия | - |
| Статус (Pass/Fail) | Pass. |

Таблица 5 - Тестирование сохранения документов локально

| Test Case #4 | Тестирование сохранения документов локально |
| --- | --- |
| 1 | 2 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий. |
| Название тестирования/Имя | Возможность сохранить документы локально |
| Резюме испытания | Проверка сохранение электронный пропуск на пк |
| Шаги тестирования | |  |  | | --- | --- | | № шага | Действие | | 1 | Нажатие на кнопку «Создать документ» | |
| Данные тестирования | Documents\_Flow.cs |
| Ожидаемый результат | Сохранение документов локально |
| Фактический результат | Сохранение документов локально |
| Предпосылки | Подключение к базе данных |
| Постусловия | - |
| Статус (Pass/Fail) | Pass. |

Таблица 6 - Тестирование работы с данными в бд

| Test Case #5 | Тестирование работы с данными в бд |
| --- | --- |
| 1 | 2 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий. |
| Название тестирования/Имя | Crud данных. |
| Резюме испытания | Проверка правильной заполнения бд |
| Шаги тестирования | |  |  | | --- | --- | | № шага | Действие | | 1 | Нажатие на кнопку «Добавить» | | 2 | Нажатие на кнопку «Изменить» | | 3 | Нажатие на кнопку «Удалить» | | 4 | Нажатие на кнопку «Авторизация» | | 5 | Нажатие на кнопку «Сохранить» | | 6 | Нажатие на кнопку «Зарегистрироваться» | | 7 | Нажате на кнопку «Создать документы» | |
| Данные тестирования | Company.cs, Documents\_Flow.cs, Groups.cs, Practice\_Dates.cs, Students.cs, Autorization.cs, Registration.cs, Settings.cs |
| Ожидаемый результат | Правильное заполнение бд |
| Фактический результат | Правильное заполнение бд |
| Предпосылки | - |
| Постусловия | - |
| Статус (Pass/Fail) | Pass. |

Таблица 7 - Тестирование заполнения полей при выборе папки

| Test Case #6 | Тестирование заполнения полей при выборе папки |
| --- | --- |
| 1 | 2 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий. |
| Название тестирования/Имя |  |
| Резюме испытания | Проверка заполнения полей при выборе папки |
| Шаги тестирования | |  |  | | --- | --- | | № шага | Действие | | 1 | Нажать на поле выбора пути | | 2 | Выбор нужного пути | | 3 | Нажатие на кнопку «Открыть» | |
| Данные тестирования | Settings.cs |
| Ожидаемый результат | Заполнение поля птем к папке |
| Фактический результат | Заполнение поля путем к папке |
| Предпосылки | - |
| Постусловия | - |
| Статус (Pass/Fail) | Pass. |

Таблица 8 - Тестирование заполнения полей при выборе файла

| Test Case #7 | Тестирование заполнения полей при выборе файла |
| --- | --- |
| 1 | 2 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий. |
| Название тестирования/Имя | Возможность получить заявку на почту. |
| Резюме испытания | Проверка заполнения полей при выборе файла |
| Шаги тестирования | |  |  | | --- | --- | | № шага | Действие | | 1 | Нажать на поле выбора пути | | 2 | Выбор нужного пути к файлу | | 3 | Нажатие на кнопку «Открыть» | |
| Данные тестирования | Settings.cs |
| Ожидаемый результат | Заполнение поля птем к файлу |
| Фактический результат | Заполнение поля путем к файлу |
| Предпосылки | - |
| Постусловия | - |
| Статус (Pass/Fail) | Pass. |

Таблица 9 - Тестирование входа в систему с определенной ролью

| Test Case #8 | Тестирование входа в систему с определенной ролью |
| --- | --- |
| 1 | 2 |
| Приоритет тестирования (Малый/Средний/высокий) | Высокий. |
| Название тестирования/Имя | Возможность войти с определенной ролью |
| Резюме испытания | Проверка выдачи роли |
| Шаги тестирования | |  |  | | --- | --- | | № шага | Действие | | 1 | Заполнение полей актуальными данными пользователя. | | 2 | Нажатие на кнопку «Авторизация» | |
| Данные тестирования | Autorization.cs |
| Ожидаемый результат | Переход на форму в соответствии с ролью |
| Фактический результат | Переход на форму в соответствии с ролью |
| Предпосылки | Подключение к базе данных. |
| Постусловия | - |
| Статус (Pass/Fail) | Pass. |

ПРИЛОЖЕНИЕ В. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

АННОТАЦИЯ

В данном программном документе приведена установка и основная настройка приложения «WKR».

Приложение состоит из разделов:

1. Установка приложения.
2. Использование.
3. Деинсталляция.
4. Сообщения.
   1. Перейти по ссылке <https://drive.google.com/drive/u/1/folders/1kDkvUpMbGLvT0y5jwD2KFiBA03VNMbRY> .
   2. Скачать архив.

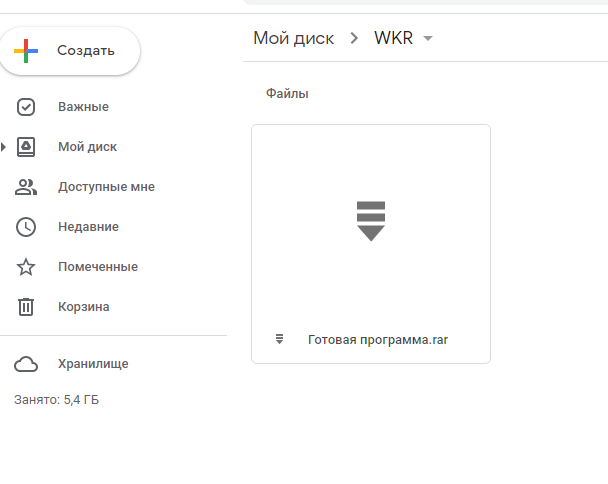


Рисунок 1 – Архив

* 1. Распаковать архив.
  2. Файл WKR.sql открыть в программе Microsoft SQL Server Management Studio 18.

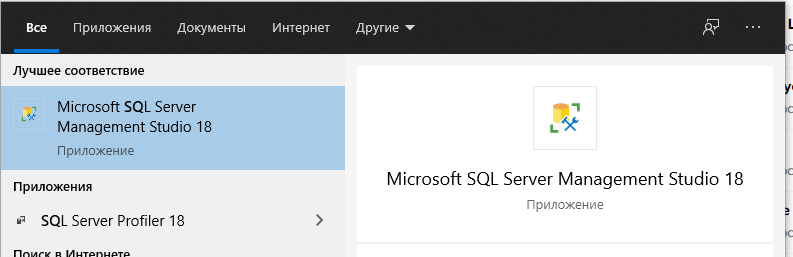


Рисунок 2 – Программа

* 1. Нажать на кнопку «Execute».

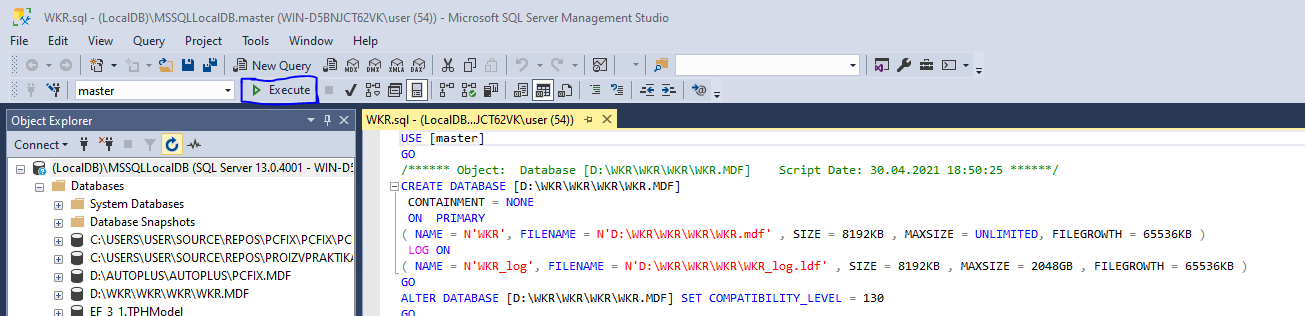


Рисунок 3 – База

* 1. Запустить файл Setup.exe .



Рисунок 4 – Запуск файла

* 1. После успешной установки появится основное окно программы. Нажимаем на кнопку «Регистрация» для перехода на окно регистрации нового пользователя.

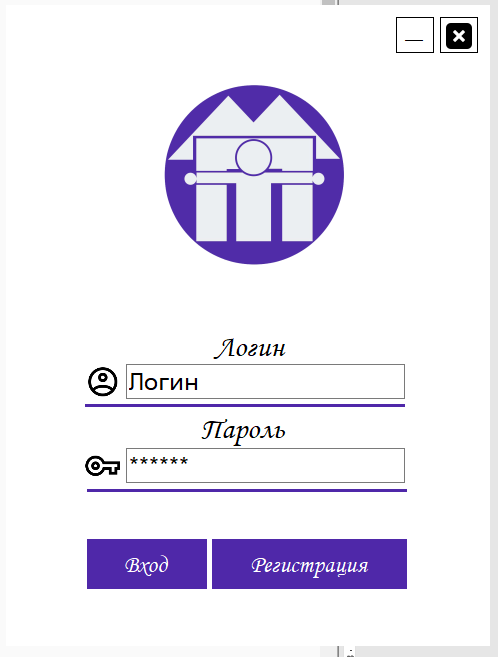


Рисунок 5 – Авторизация

* 1. В окне регистрации вводим данные и код для выдачи роли.



Рисунок 6 – Регистрация

* 1. При успешной регистрации появится уведомление о успешной регистрации.



Рисунок 7 - Успешная регистрация

* 1. Далее вводим данные, указанные при регистрации и, если все указано правильно попадаем в следующее окно. Это окно – основное окно программы из него можно перемещаться по всем остальным окнам.

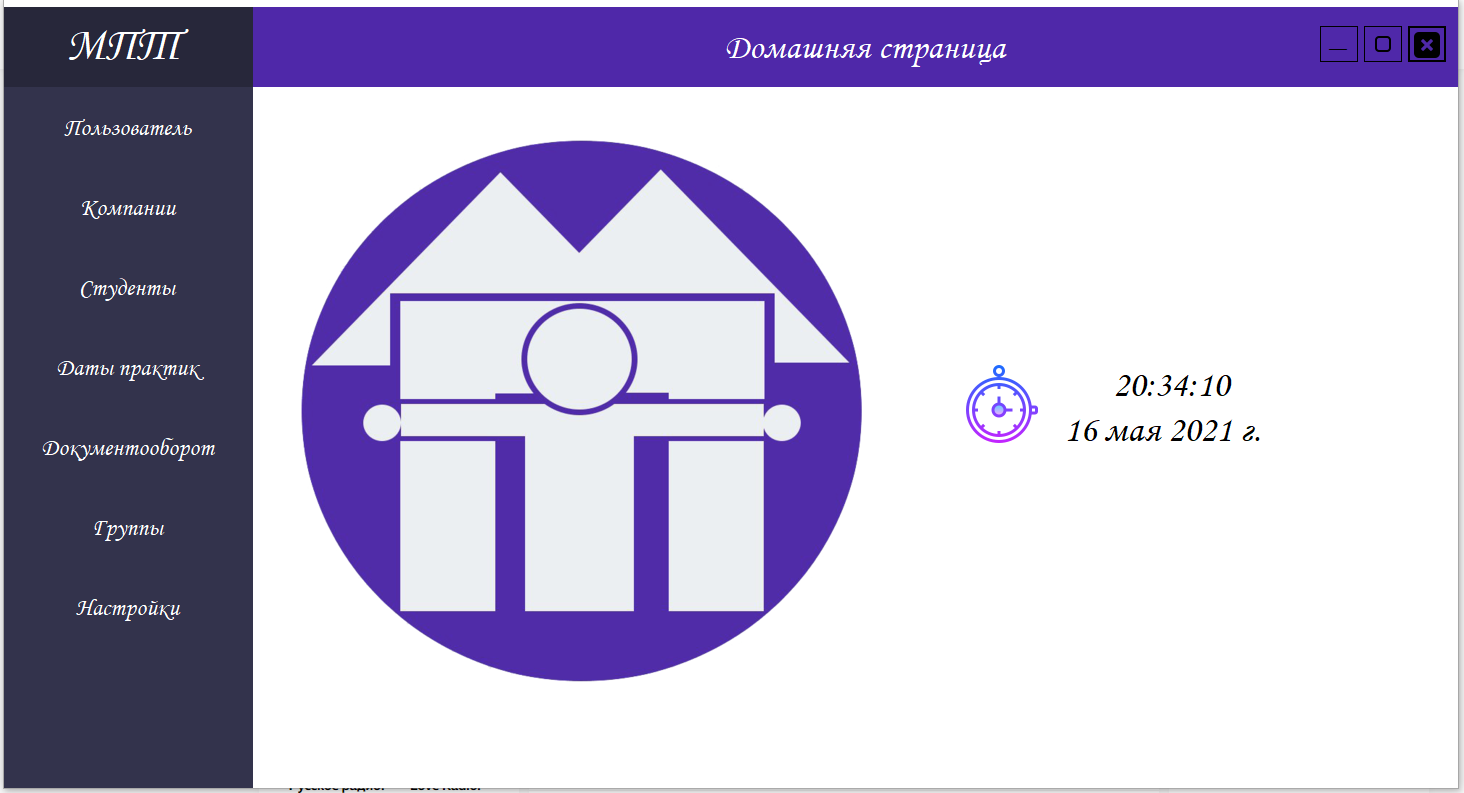


Рисунок 8 – Основное окно

* 1. Во вкладках: Компании, Студенты, Даты практик, Группы можно работать с данными из базы. Рассмотрим этот функционал на примере вкладки «Компании».

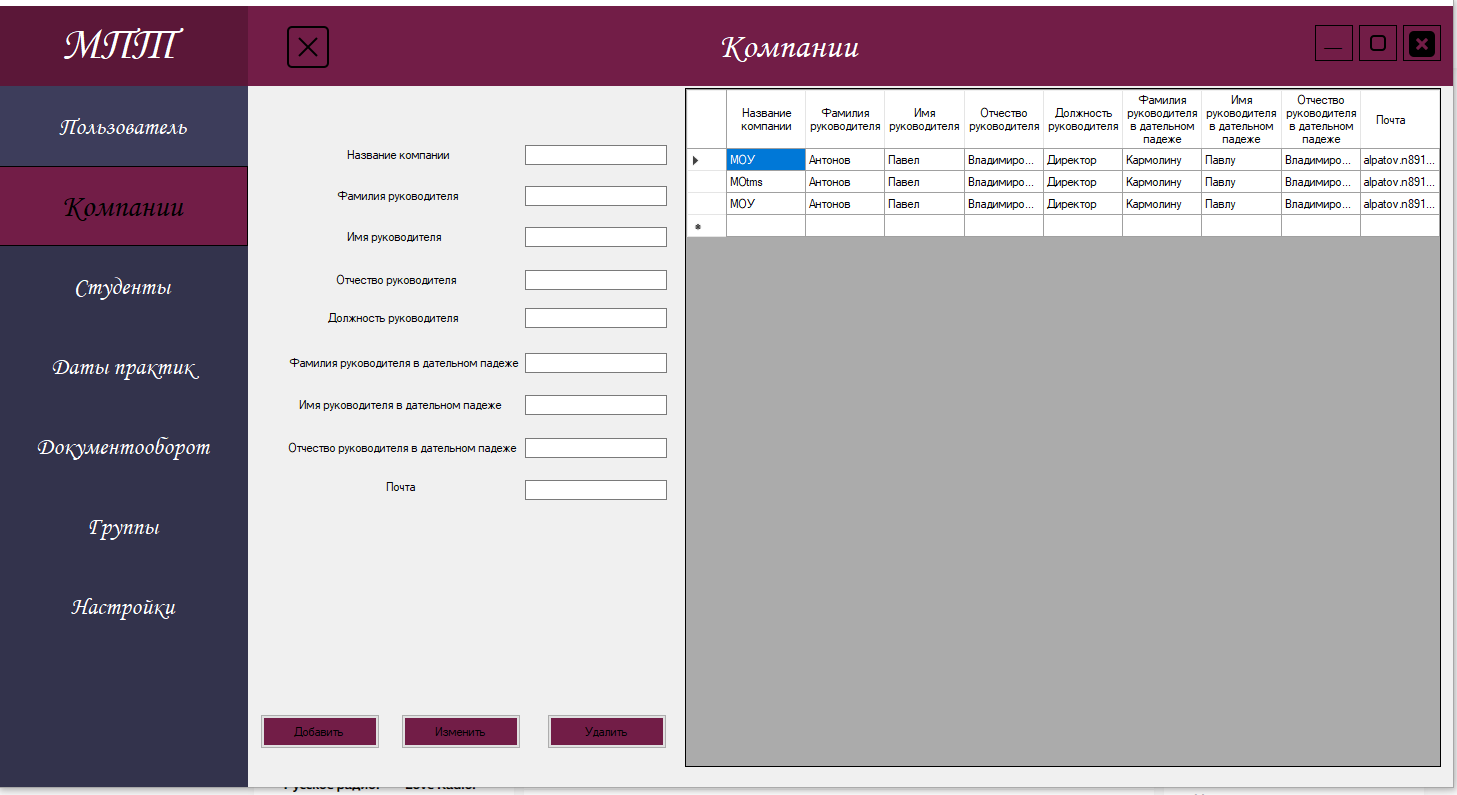


Рисунок 9 - Компании

* 1. Для начала работы с данными необходимо нажать на нужное поле в таблице. Если же в таблице ничего нет, то необходимо заполнить все поля соответственно данным, которые указаны левее каждого поля.

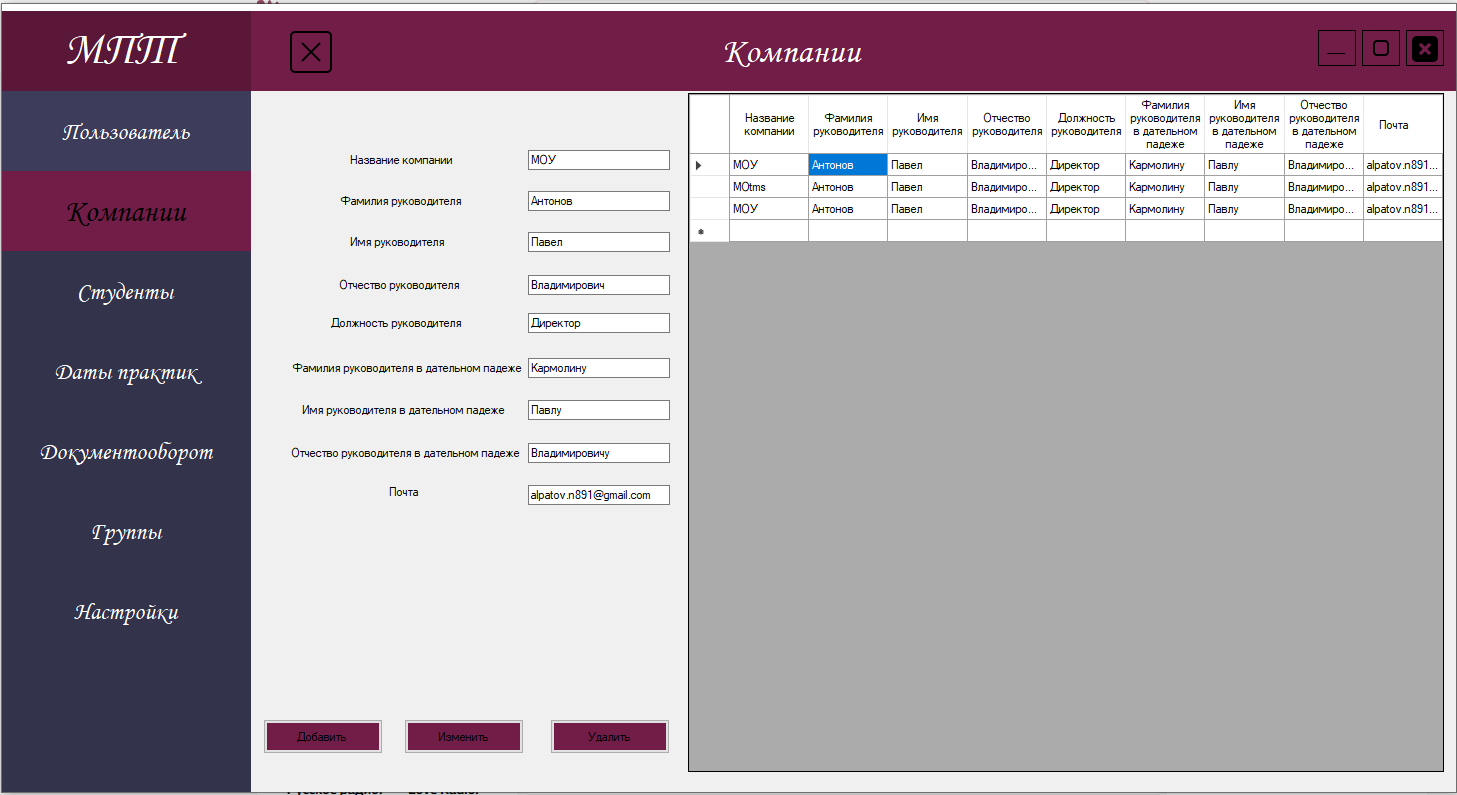


Рисунок 10 – Заполненные данные

* 1. Далее, когда все данные в полях имеются, необходимо нажать на одну из трех кнопок, которые находят во вкладке. Названия кнопок означают то, что произойдет с данными.
  2. На вкладке «Настройки» нужно указать место хранения итоговых документов и то место, где хранятся шаблоны документов. Для их указания необходимо нажать на соответствующее поле.

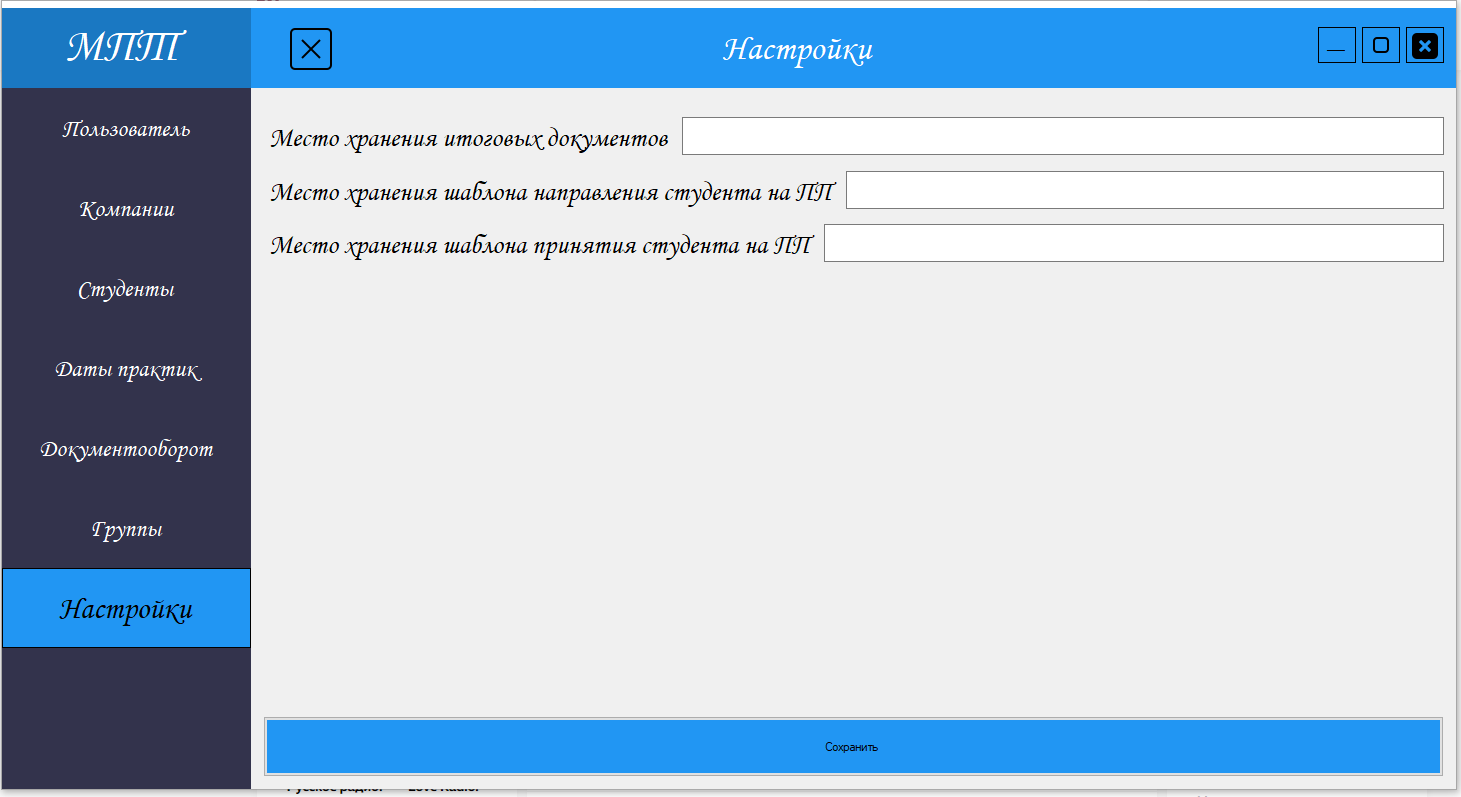


Рисунок 11 – Настройки

* 1. После заполнения всех полей необходимо нажать на кнопку «Сохранить».
  2. Далее переходим к вкладке «Документооборот». В данной вкладке нужно выбрать студента и компанию, в которой он будет проходить практику. После их выбора необходимо нажать на кнопку «Создать документы» для создания документов с просьбой о принятии студента на производственную практику и направление студента на производственную практику.

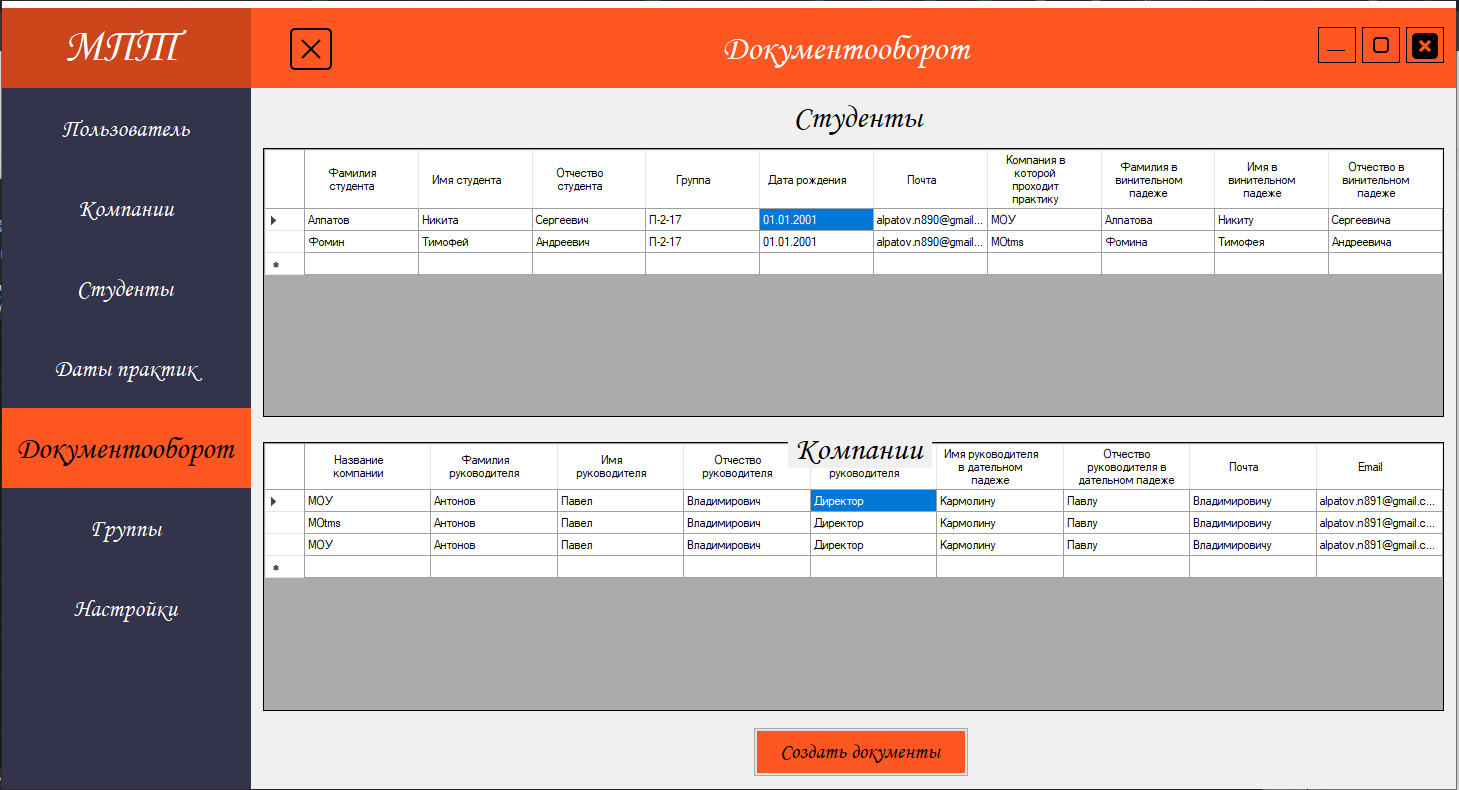


Рисунок 12 – Документооборот

* 1. В итоге в том месте, которое было указано в качестве места хранения документов создастся папка группы, в которой находится данный студент. Если папка уже была создана раннее она не будет создана заново. В папке будут находится два сформированных файла.

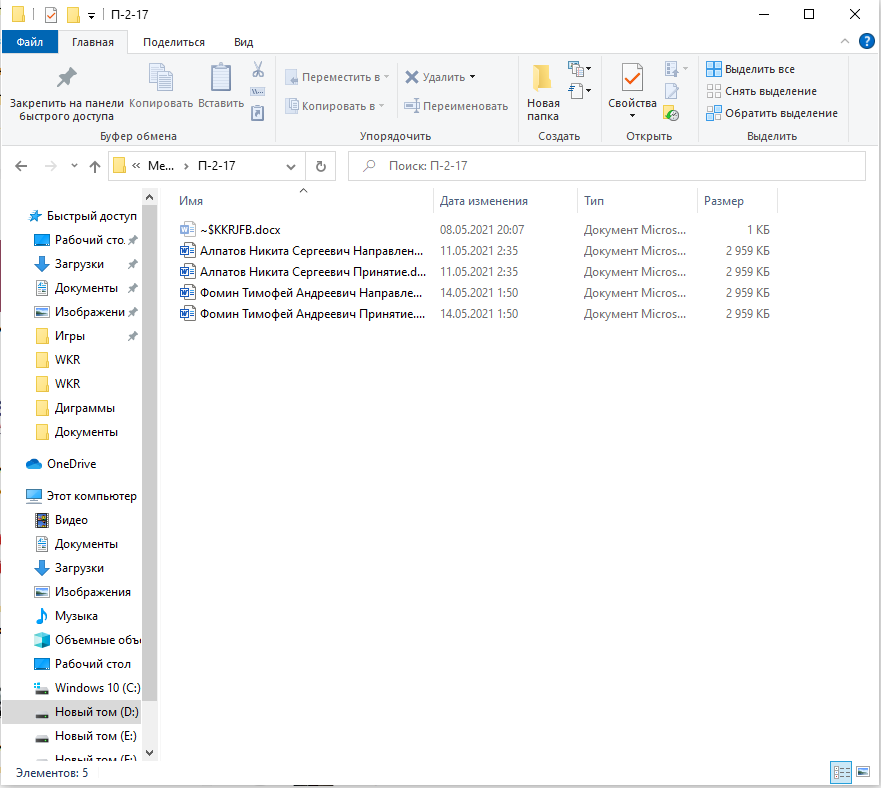


Рисунок 13 – Созданные файлы

3.1 Для удаления программы необходимо удалить данные, получившиеся в ходе установки и в программе Microsoft SQL Server Management Studio 18 удалить базу нажав на нее правой кнопкой и выбрав пункт «Delete».

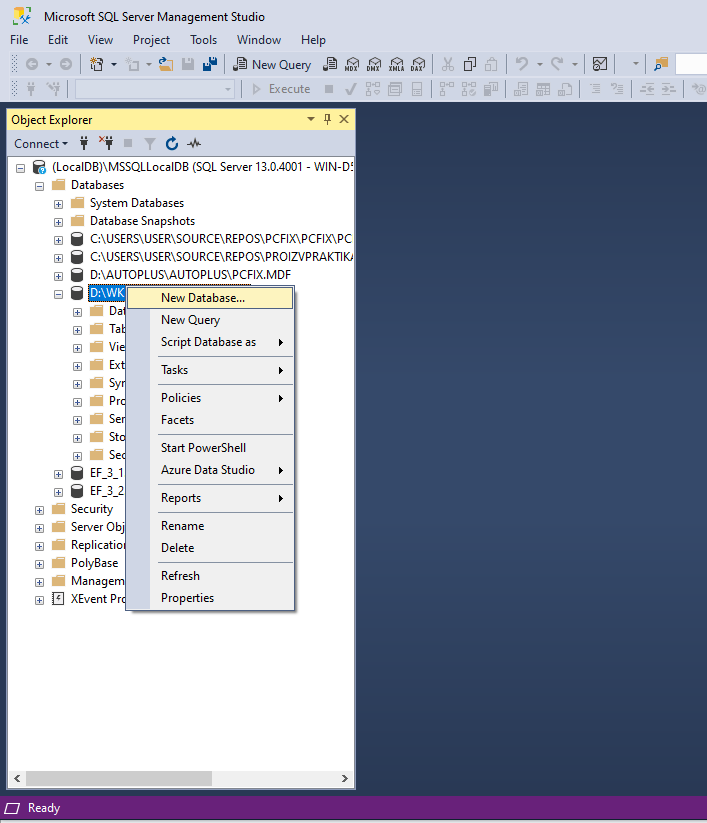


Рисунок 14 – Удаление БД

4.1 В ходе работы программы будут появляться сообщения в виде дополнительно окна.

4.2 Первое окно будет появляться при неудачно регистрации. Если данные были введены неправильно, и оно оповестит пользователя об этом.



Рисунок 15 – Оповещение

4.3 Следующее окно появится у пользователя при авторизации. Оно оповестит пользователя о том, что данные были введены неверно или данных нет в базе.

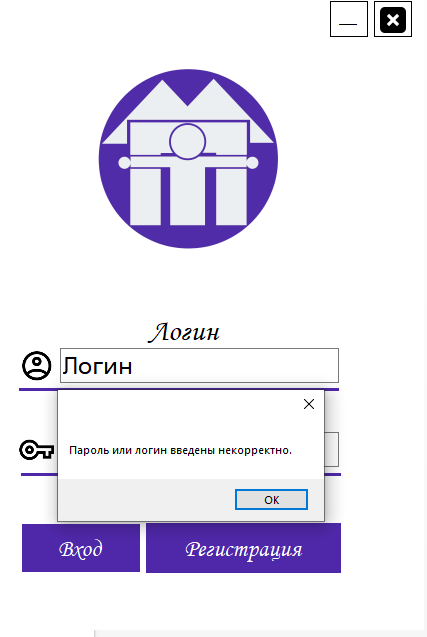


Рисунок 16 - Оповещение при регистрации

4.4 Следующее окно будет возникать у пользователя во вкладках: Студенты, Группы, Даты практик. Окно будет появляться в том случае если пользователь будет вводить в поле группы группу которой нет в бд.

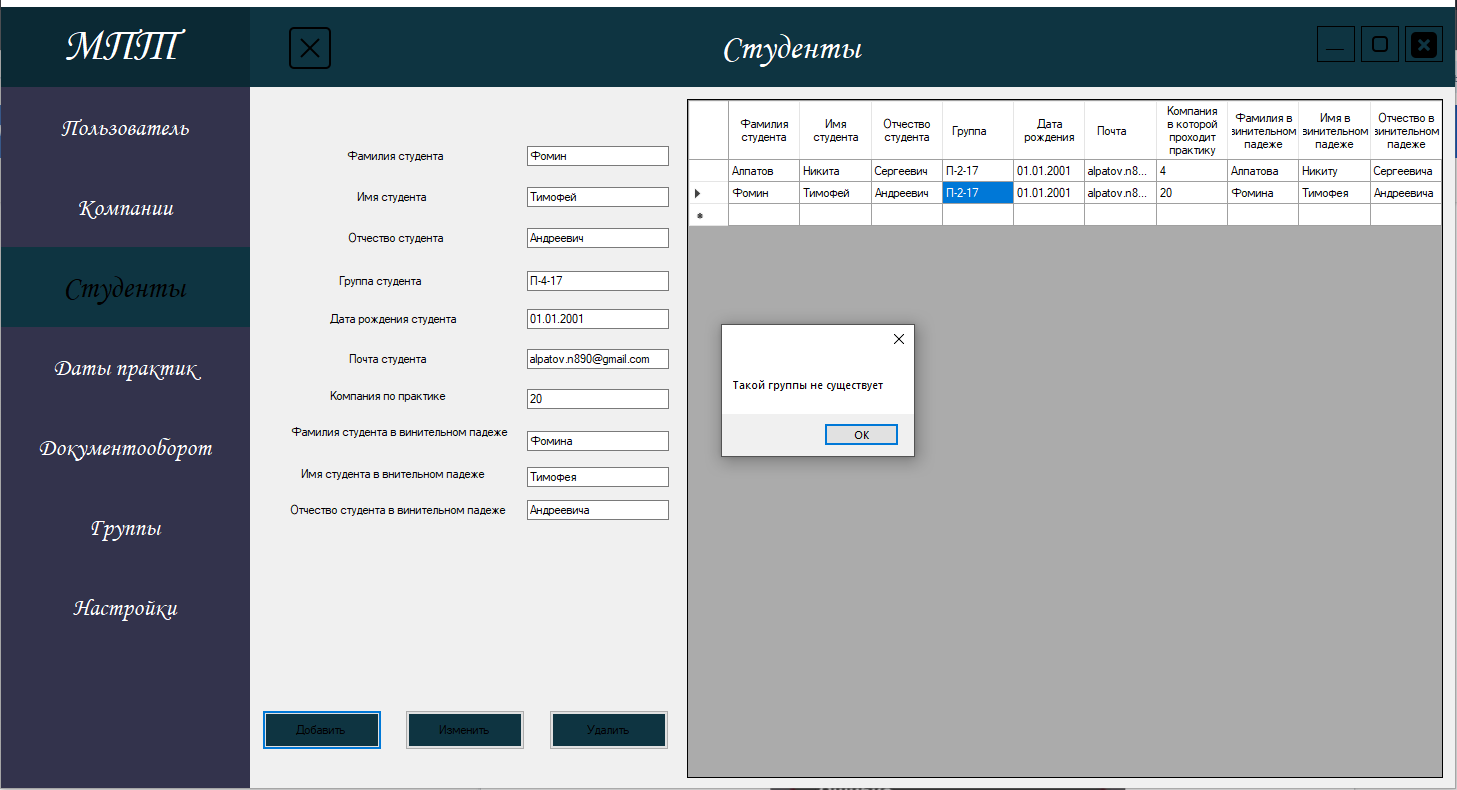


Рисунок 17 – Оповещении о несуществующей папке

4.5 Следующее оповещение будет возникать при формировании документов. Оно возникает если по пути создании документов нет папки с группой и автоматически создаст ее.

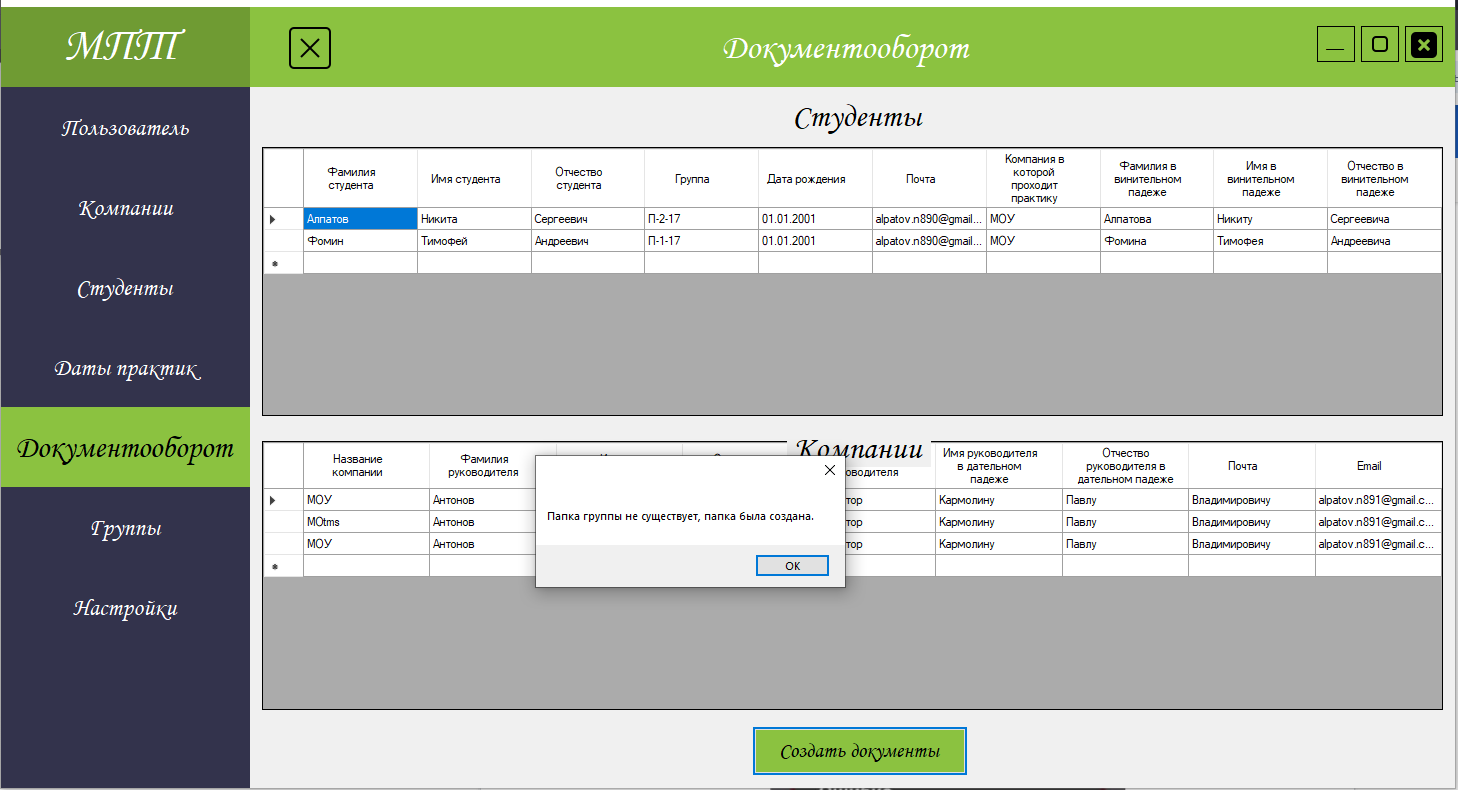


Рисунок 18 – Оповещение о папке группы которой не существовало

.

ПРИЛОЖЕНИЕ Г. СКРИПТ БАЗЫ ДАННЫХ

АННОТАЦИЯ

В данном программном документе приведен скрипт базы данных.

Приложение состоит из разделов:

1. Скрипт базы данных.
2. Скрипт базы данных.

USE [master]

GO

/\*\*\*\*\*\* Object: Database [G:\WKR\WKR\WKR.MDF] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

CREATE DATABASE [G:\WKR\WKR\WKR.MDF]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'WKR', FILENAME = N'G:\WKR\WKR\WKR.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

LOG ON

( NAME = N'WKR\_log', FILENAME = N'G:\WKR\WKR\WKR\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET COMPATIBILITY\_LEVEL = 130

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [G:\WKR\WKR\WKR.MDF].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET ARITHABORT OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET AUTO\_CLOSE ON

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET AUTO\_SHRINK ON

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET DISABLE\_BROKER

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET RECOVERY SIMPLE

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET MULTI\_USER

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET DB\_CHAINING OFF

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET TARGET\_RECOVERY\_TIME = 60 SECONDS

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET QUERY\_STORE = OFF

GO

USE [G:\WKR\WKR\WKR.MDF]

GO

ALTER DATABASE SCOPED CONFIGURATION SET LEGACY\_CARDINALITY\_ESTIMATION = OFF;

GO

ALTER DATABASE SCOPED CONFIGURATION SET MAXDOP = 0;

GO

ALTER DATABASE SCOPED CONFIGURATION SET PARAMETER\_SNIFFING = ON;

GO

ALTER DATABASE SCOPED CONFIGURATION SET QUERY\_OPTIMIZER\_HOTFIXES = OFF;

GO

USE [G:\WKR\WKR\WKR.MDF]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Company] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Company](

[id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Manager\_Surname] [nvarchar](50) NOT NULL,

[Manager\_Name] [nvarchar](50) NOT NULL,

[Manager\_Patronymic] [nvarchar](50) NULL,

[Manager\_Position] [nvarchar](50) NOT NULL,

[Manager\_surname\_in\_the\_dative\_case] [nvarchar](50) NOT NULL,

[Manager\_name\_in\_the\_dative\_case] [nvarchar](50) NOT NULL,

[Manager\_patronymic\_in\_the\_dative\_case] [nvarchar](50) NOT NULL,

[Email] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Company] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Group] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Group](

[id] [int] IDENTITY(1,1) NOT NULL,

[Group\_name] [nvarchar](50) NOT NULL,

[Specialization] [nvarchar](100) NOT NULL,

CONSTRAINT [PK\_Group] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Students] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Students](

[id] [int] IDENTITY(1,1) NOT NULL,

[Surname] [nvarchar](50) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Patronymic] [nvarchar](50) NULL,

[Group] [int] NOT NULL,

[Birthday] [date] NOT NULL,

[Email] [nvarchar](50) NOT NULL,

[Practice\_company] [int] NULL,

[Surname\_in\_the\_accusative\_case] [nvarchar](50) NOT NULL,

[First\_name\_in\_the\_accusative\_case] [nvarchar](50) NOT NULL,

[Patronymic\_in\_the\_accusative\_case] [nvarchar](50) NULL,

CONSTRAINT [PK\_Students] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[Students\_Company\_Group] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[Students\_Company\_Group]

AS

SELECT dbo.Students.id, dbo.Students.Surname, dbo.Students.Name, dbo.Students.Patronymic, dbo.[Group].Group\_name, dbo.Students.Birthday, dbo.Students.Email, dbo.Company.Name AS Company\_name,

dbo.Students.Surname\_in\_the\_accusative\_case, dbo.Students.First\_name\_in\_the\_accusative\_case, dbo.Students.Patronymic\_in\_the\_accusative\_case, dbo.[Group].Specialization

FROM dbo.Students INNER JOIN

dbo.[Group] ON dbo.Students.[Group] = dbo.[Group].id LEFT OUTER JOIN

dbo.Company ON dbo.Students.Practice\_company = dbo.Company.id

GO

/\*\*\*\*\*\* Object: Table [dbo].[Practice\_dates] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Practice\_dates](

[id] [int] IDENTITY(1,1) NOT NULL,

[Professional\_module] [nvarchar](100) NULL,

[Name\_of\_the\_practice] [nvarchar](100) NULL,

[Pre\_graduate\_practice] [nvarchar](50) NOT NULL,

[Group\_name] [int] NOT NULL,

[Pactice\_start\_date] [date] NOT NULL,

[Pactice\_termination\_date] [date] NULL,

[Date\_of\_continuation\_of\_the\_practice] [date] NULL,

[Pactice\_termination\_date\_2] [date] NULL,

[Date\_of\_continuation\_of\_the\_practice\_2] [date] NULL,

[Practice\_end\_date] [date] NOT NULL,

CONSTRAINT [PK\_Practice\_dates] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: View [dbo].[Practice\_dates\_Group] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[Practice\_dates\_Group]

AS

SELECT dbo.Practice\_dates.id, dbo.Practice\_dates.Professional\_module, dbo.Practice\_dates.Name\_of\_the\_practice, dbo.Practice\_dates.Pre\_graduate\_practice, dbo.[Group].Group\_name, dbo.Practice\_dates.Pactice\_start\_date,

dbo.Practice\_dates.Pactice\_termination\_date, dbo.Practice\_dates.Date\_of\_continuation\_of\_the\_practice, dbo.Practice\_dates.Pactice\_termination\_date\_2, dbo.Practice\_dates.Date\_of\_continuation\_of\_the\_practice\_2,

dbo.Practice\_dates.Practice\_end\_date

FROM dbo.Practice\_dates INNER JOIN

dbo.[Group] ON dbo.Practice\_dates.Group\_name = dbo.[Group].id

GO

/\*\*\*\*\*\* Object: View [dbo].[Students\_Group] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE VIEW [dbo].[Students\_Group]

AS

SELECT dbo.Students.id, dbo.Students.Surname, dbo.Students.Name, dbo.Students.Patronymic, dbo.[Group].Group\_name, dbo.Students.Birthday, dbo.Students.Email, dbo.Students.Practice\_company,

dbo.Students.Surname\_in\_the\_accusative\_case, dbo.Students.First\_name\_in\_the\_accusative\_case, dbo.Students.Patronymic\_in\_the\_accusative\_case

FROM dbo.Students INNER JOIN

dbo.[Group] ON dbo.Students.[Group] = dbo.[Group].id

GO

/\*\*\*\*\*\* Object: Table [dbo].[Documents\_Flow] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Documents\_Flow](

[id] [int] IDENTITY(1,1) NOT NULL,

[id\_Student] [int] NOT NULL,

[id\_Company] [int] NOT NULL,

[Document\_Type] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Documents\_Flow] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Role\_passwords] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Role\_passwords](

[id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Password] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Role\_passwords] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Settings] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Settings](

[id] [int] IDENTITY(1,1) NOT NULL,

[Document\_storage\_location] [nvarchar](100) NULL,

[Document\_template\_storage\_location] [nvarchar](100) NULL,

[Document2\_template\_storage\_location] [nvarchar](100) NULL,

CONSTRAINT [PK\_Settings] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Users] Script Date: 30.05.2021 15:09:53 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Users](

[id] [int] IDENTITY(1,1) NOT NULL,

[Surname] [nvarchar](50) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Patronymic] [nvarchar](50) NOT NULL,

[Email] [nvarchar](50) NOT NULL,

[Login] [nvarchar](50) NOT NULL,

[Password] [nvarchar](50) NOT NULL,

[Birthday] [date] NOT NULL,

[Settings] [int] NOT NULL,

[Role] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Users] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Documents\_Flow] WITH CHECK ADD CONSTRAINT [FK\_Documents\_Flow\_Company] FOREIGN KEY([id\_Company])

REFERENCES [dbo].[Company] ([id])

GO

ALTER TABLE [dbo].[Documents\_Flow] CHECK CONSTRAINT [FK\_Documents\_Flow\_Company]

GO

ALTER TABLE [dbo].[Documents\_Flow] WITH CHECK ADD CONSTRAINT [FK\_Documents\_Flow\_Students] FOREIGN KEY([id\_Student])

REFERENCES [dbo].[Students] ([id])

GO

ALTER TABLE [dbo].[Documents\_Flow] CHECK CONSTRAINT [FK\_Documents\_Flow\_Students]

GO

ALTER TABLE [dbo].[Practice\_dates] WITH CHECK ADD CONSTRAINT [FK\_Practice\_dates\_Group] FOREIGN KEY([Group\_name])

REFERENCES [dbo].[Group] ([id])

GO

ALTER TABLE [dbo].[Practice\_dates] CHECK CONSTRAINT [FK\_Practice\_dates\_Group]

GO

ALTER TABLE [dbo].[Students] WITH CHECK ADD CONSTRAINT [FK\_Students\_Company] FOREIGN KEY([Practice\_company])

REFERENCES [dbo].[Company] ([id])

GO

ALTER TABLE [dbo].[Students] CHECK CONSTRAINT [FK\_Students\_Company]

GO

ALTER TABLE [dbo].[Students] WITH CHECK ADD CONSTRAINT [FK\_Students\_Group] FOREIGN KEY([Group])

REFERENCES [dbo].[Group] ([id])

GO

ALTER TABLE [dbo].[Students] CHECK CONSTRAINT [FK\_Students\_Group]

GO

ALTER TABLE [dbo].[Users] WITH CHECK ADD CONSTRAINT [FK\_Users\_Settings] FOREIGN KEY([Settings])

REFERENCES [dbo].[Settings] ([id])

GO

ALTER TABLE [dbo].[Users] CHECK CONSTRAINT [FK\_Users\_Settings]

GO

EXEC sys.sp\_addextendedproperty @name=N'MS\_DiagramPane1', @value=N'[0E232FF0-B466-11cf-A24F-00AA00A3EFFF, 1.00]

Begin DesignProperties =

Begin PaneConfigurations =

Begin PaneConfiguration = 0

NumPanes = 4

Configuration = "(H (1[40] 4[20] 2[20] 3) )"

End

Begin PaneConfiguration = 1

NumPanes = 3

Configuration = "(H (1 [50] 4 [25] 3))"

End

Begin PaneConfiguration = 2

NumPanes = 3

Configuration = "(H (1 [50] 2 [25] 3))"

End

Begin PaneConfiguration = 3

NumPanes = 3

Configuration = "(H (4 [30] 2 [40] 3))"

End

Begin PaneConfiguration = 4

NumPanes = 2

Configuration = "(H (1 [56] 3))"

End

Begin PaneConfiguration = 5

NumPanes = 2

Configuration = "(H (2 [66] 3))"

End

Begin PaneConfiguration = 6

NumPanes = 2

Configuration = "(H (4 [50] 3))"

End

Begin PaneConfiguration = 7

NumPanes = 1

Configuration = "(V (3))"

End

Begin PaneConfiguration = 8

NumPanes = 3

Configuration = "(H (1[56] 4[18] 2) )"

End

Begin PaneConfiguration = 9

NumPanes = 2

Configuration = "(H (1 [75] 4))"

End

Begin PaneConfiguration = 10

NumPanes = 2

Configuration = "(H (1[66] 2) )"

End

Begin PaneConfiguration = 11

NumPanes = 2

Configuration = "(H (4 [60] 2))"

End

Begin PaneConfiguration = 12

NumPanes = 1

Configuration = "(H (1) )"

End

Begin PaneConfiguration = 13

NumPanes = 1

Configuration = "(V (4))"

End

Begin PaneConfiguration = 14

NumPanes = 1

Configuration = "(V (2))"

End

ActivePaneConfig = 0

End

Begin DiagramPane =

Begin Origin =

Top = 0

Left = 0

End

Begin Tables =

Begin Table = "Practice\_dates"

Begin Extent =

Top = 6

Left = 38

Bottom = 285

Right = 336

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Group"

Begin Extent =

Top = 6

Left = 374

Bottom = 159

Right = 544

End

DisplayFlags = 280

TopColumn = 0

End

End

End

Begin SQLPane =

End

Begin DataPane =

Begin ParameterDefaults = ""

End

Begin ColumnWidths = 12

Width = 284

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

End

End

Begin CriteriaPane =

Begin ColumnWidths = 11

Column = 1440

Alias = 900

Table = 1170

Output = 720

Append = 1400

NewValue = 1170

SortType = 1350

SortOrder = 1410

GroupBy = 1350

Filter = 1350

Or = 1350

Or = 1350

Or = 1350

End

End

End

' , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'Practice\_dates\_Group'

GO

EXEC sys.sp\_addextendedproperty @name=N'MS\_DiagramPaneCount', @value=1 , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'Practice\_dates\_Group'

GO

EXEC sys.sp\_addextendedproperty @name=N'MS\_DiagramPane1', @value=N'[0E232FF0-B466-11cf-A24F-00AA00A3EFFF, 1.00]

Begin DesignProperties =

Begin PaneConfigurations =

Begin PaneConfiguration = 0

NumPanes = 4

Configuration = "(H (1[40] 4[20] 2[20] 3) )"

End

Begin PaneConfiguration = 1

NumPanes = 3

Configuration = "(H (1 [50] 4 [25] 3))"

End

Begin PaneConfiguration = 2

NumPanes = 3

Configuration = "(H (1 [50] 2 [25] 3))"

End

Begin PaneConfiguration = 3

NumPanes = 3

Configuration = "(H (4 [30] 2 [40] 3))"

End

Begin PaneConfiguration = 4

NumPanes = 2

Configuration = "(H (1 [56] 3))"

End

Begin PaneConfiguration = 5

NumPanes = 2

Configuration = "(H (2 [66] 3))"

End

Begin PaneConfiguration = 6

NumPanes = 2

Configuration = "(H (4 [50] 3))"

End

Begin PaneConfiguration = 7

NumPanes = 1

Configuration = "(V (3))"

End

Begin PaneConfiguration = 8

NumPanes = 3

Configuration = "(H (1[56] 4[18] 2) )"

End

Begin PaneConfiguration = 9

NumPanes = 2

Configuration = "(H (1 [75] 4))"

End

Begin PaneConfiguration = 10

NumPanes = 2

Configuration = "(H (1[66] 2) )"

End

Begin PaneConfiguration = 11

NumPanes = 2

Configuration = "(H (4 [60] 2))"

End

Begin PaneConfiguration = 12

NumPanes = 1

Configuration = "(H (1) )"

End

Begin PaneConfiguration = 13

NumPanes = 1

Configuration = "(V (4))"

End

Begin PaneConfiguration = 14

NumPanes = 1

Configuration = "(V (2))"

End

ActivePaneConfig = 0

End

Begin DiagramPane =

Begin Origin =

Top = 0

Left = 0

End

Begin Tables =

Begin Table = "Students"

Begin Extent =

Top = 52

Left = 336

Bottom = 327

Right = 611

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Group"

Begin Extent =

Top = 6

Left = 38

Bottom = 119

Right = 208

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Company"

Begin Extent =

Top = 58

Left = 750

Bottom = 318

Right = 1054

End

DisplayFlags = 280

TopColumn = 0

End

End

End

Begin SQLPane =

End

Begin DataPane =

Begin ParameterDefaults = ""

End

Begin ColumnWidths = 13

Width = 284

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

End

End

Begin CriteriaPane =

Begin ColumnWidths = 11

Column = 1440

Alias = 900

Table = 1170

Output = 720

Append = 1400

NewValue = 1170

SortType = 1350

SortOrder = 1410

GroupBy = 1350

Filter = 1350

Or = 1350

Or = 1350

Or = 1350

End

End

End

' , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'Students\_Company\_Group'

GO

EXEC sys.sp\_addextendedproperty @name=N'MS\_DiagramPaneCount', @value=1 , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'Students\_Company\_Group'

GO

EXEC sys.sp\_addextendedproperty @name=N'MS\_DiagramPane1', @value=N'[0E232FF0-B466-11cf-A24F-00AA00A3EFFF, 1.00]

Begin DesignProperties =

Begin PaneConfigurations =

Begin PaneConfiguration = 0

NumPanes = 4

Configuration = "(H (1[44] 4[18] 2[20] 3) )"

End

Begin PaneConfiguration = 1

NumPanes = 3

Configuration = "(H (1 [50] 4 [25] 3))"

End

Begin PaneConfiguration = 2

NumPanes = 3

Configuration = "(H (1 [50] 2 [25] 3))"

End

Begin PaneConfiguration = 3

NumPanes = 3

Configuration = "(H (4 [30] 2 [40] 3))"

End

Begin PaneConfiguration = 4

NumPanes = 2

Configuration = "(H (1 [56] 3))"

End

Begin PaneConfiguration = 5

NumPanes = 2

Configuration = "(H (2 [66] 3))"

End

Begin PaneConfiguration = 6

NumPanes = 2

Configuration = "(H (4 [50] 3))"

End

Begin PaneConfiguration = 7

NumPanes = 1

Configuration = "(V (3))"

End

Begin PaneConfiguration = 8

NumPanes = 3

Configuration = "(H (1[56] 4[18] 2) )"

End

Begin PaneConfiguration = 9

NumPanes = 2

Configuration = "(H (1 [75] 4))"

End

Begin PaneConfiguration = 10

NumPanes = 2

Configuration = "(H (1[66] 2) )"

End

Begin PaneConfiguration = 11

NumPanes = 2

Configuration = "(H (4 [60] 2))"

End

Begin PaneConfiguration = 12

NumPanes = 1

Configuration = "(H (1) )"

End

Begin PaneConfiguration = 13

NumPanes = 1

Configuration = "(V (4))"

End

Begin PaneConfiguration = 14

NumPanes = 1

Configuration = "(V (2))"

End

ActivePaneConfig = 0

End

Begin DiagramPane =

Begin Origin =

Top = 0

Left = -52

End

Begin Tables =

Begin Table = "Students"

Begin Extent =

Top = 41

Left = 553

Bottom = 381

Right = 828

End

DisplayFlags = 280

TopColumn = 0

End

Begin Table = "Group"

Begin Extent =

Top = 6

Left = 90

Bottom = 102

Right = 260

End

DisplayFlags = 280

TopColumn = 0

End

End

End

Begin SQLPane =

End

Begin DataPane =

Begin ParameterDefaults = ""

End

Begin ColumnWidths = 12

Width = 284

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

Width = 1500

End

End

Begin CriteriaPane =

Begin ColumnWidths = 11

Column = 1440

Alias = 900

Table = 1170

Output = 720

Append = 1400

NewValue = 1170

SortType = 1350

SortOrder = 1410

GroupBy = 1350

Filter = 1350

Or = 1350

Or = 1350

Or = 1350

End

End

End

' , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'Students\_Group'

GO

EXEC sys.sp\_addextendedproperty @name=N'MS\_DiagramPaneCount', @value=1 , @level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'VIEW',@level1name=N'Students\_Group'

GO

USE [master]

GO

ALTER DATABASE [G:\WKR\WKR\WKR.MDF] SET READ\_WRITE

GO

# ПРИЛОЖЕНИЕ Д. СХЕМА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

АННОТАЦИЯ

В данном программном документе приведены тестовые испытания для информационной системы «WKR».

В разделе «Эскизы» представлены схемы пользовательского интерфейса.

1. Эскизы
   1. Схема интерфейса формы «Авторизация»

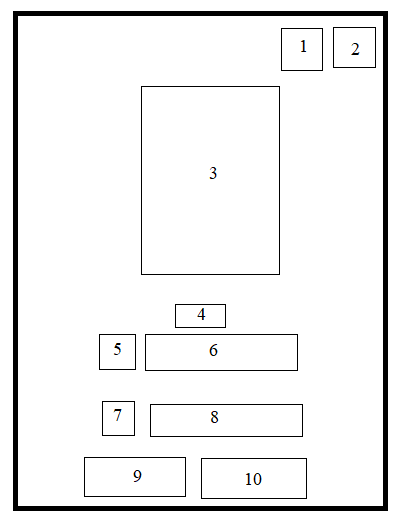


Рисунок 1 – Схема интерфейса формы «Авторизация»

Таблица 1 – Описание схемы интерфейса «Авторизации»

| № | Описание |
| --- | --- |
| 1 | 2 |
| 1 | Кнопка «Свернуть» |
| 2 | Кнопка «Выход» |
| 3 | Поле картинки |
| 4 | Надпись «Логин» |
| 5 | Поле картинки |
| 6 | Поле ввода логина |
| 7 | Поле картинки |
| 8 | Поле ввода пароля |
| 9 | Кнопка «Вход» |
| 10 | Кнопка «Регистрация» |

* 1. Схема интерфейса формы «Администратор»

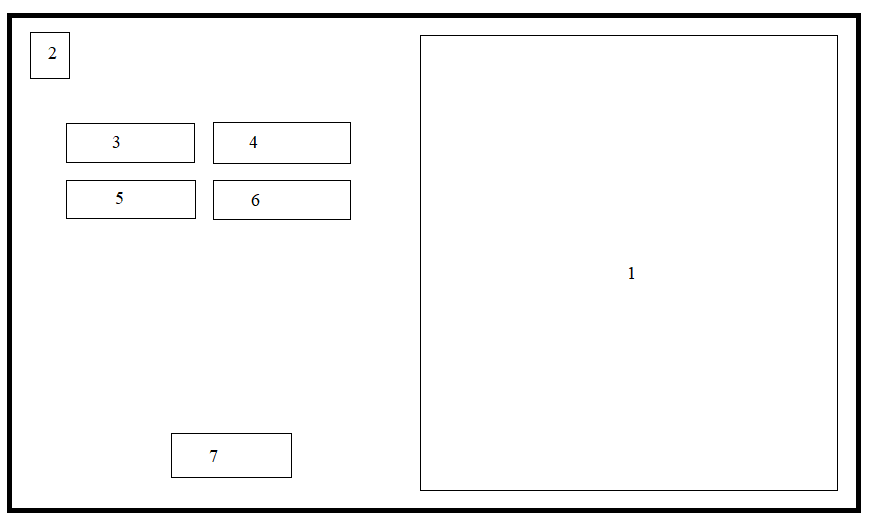


Рисунок 2 – Схема интерфейса формы «Администратор»

Таблица 2 – Описание схемы интерфейса «Администратор»

| № | Описание |
| --- | --- |
| 1 | 2 |
| 1 | Таблица «Администратор» |
| 2 | Кнопка «Выход» |
| 3 | Надпись «Название роли» |
| 4 | Поле ввода названия роли |
| 5 | Надпись «Код роли» |
| 6 | Поле ввода кода роли |
| 7 | Кнопка «Выход» |

* 1. Схема интерфейса формы «Регистрация»

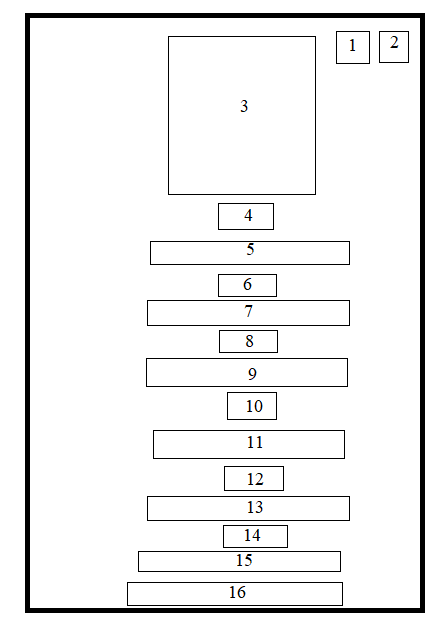


Рисунок 3 – Схема интерфейса формы «Регистрация»

Таблица 3 – Описание схемы интерфейса «Регистрации»

| № | Описание |
| --- | --- |
| 1 | 2 |
| 1 | Кнопка «Свернуть» |
| 2 | Кнопка «Выход» |
| 3 | Поле картинки |
| 4 | Надпись «ФИО» |
| 5 | Поле ввода ФИО |
| 6 | Надпись «Почта» |
| 7 | Поле ввода почты |
| 8 | Надпись «Логин» |
| 9 | Поле ввода логина |
| 10 | Надпись «Пароль» |
| 11 | Поле ввода Пароля |
| 12 | Надпись «Дата рождения» |
| 13 | Поле ввода даты рождения |
| 14 | Надпись «Код пароля» |
| 15 | Поле ввода кода пароля |
| 16 | Кнопка «Зарегистрироваться» |

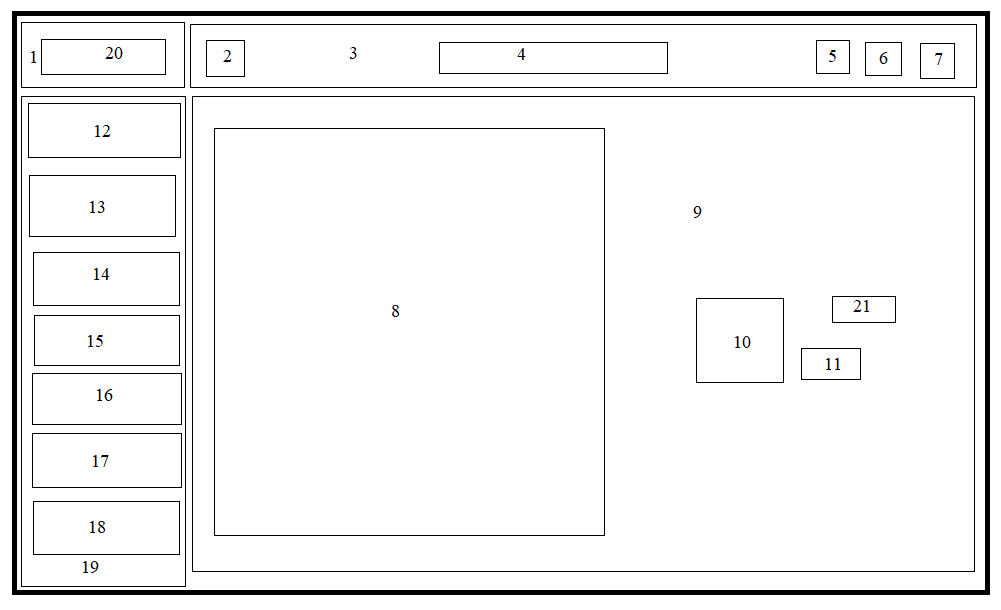
* 1. Схема интерфейса формы «Меню» 

Рисунок 4 – Схема интерфейса формы «Меню»

Таблица 4 – Описание схемы интерфейса «Меню»

|  |  |
| --- | --- |
| № | Описание |
| 1 | 2 |
| 1 | Панель |
| 2 | Кнопка переход в форму «Оформить заявку на пропуск» |
| 3 | Панель |
| 4 | Надпись «Домашняя страница» |
| 5 | Кнопка сворачивания программы |
| 6 | Кнопка развёртки программы на полный экран |
| 7 | Кнопка выходы |
| 8 | Поле картинки |
| 9 | Панель |
| 10 | Поле картинки |
| 11 | Надпись «Date» |
| 12 | Кнопка переход в форму «Пользователь» |
| 13 | Кнопка переход в форму «Компании» |
| 14 | Кнопка переход в форму «Студенты» |
| 15 | Кнопка переход в форму «Даты практик» |
| 16 | Кнопка переход в форму «Документооборот» |
| 17 | Кнопка переход в форму «Группы» |
| 18 | Кнопка переход в форму «Настройки» |
| 19 | Панель |
| 20 | Надпись «МПТ» |
| 21 | Надпись «Time» |

* 1. Схема интерфейса формы «Компании»

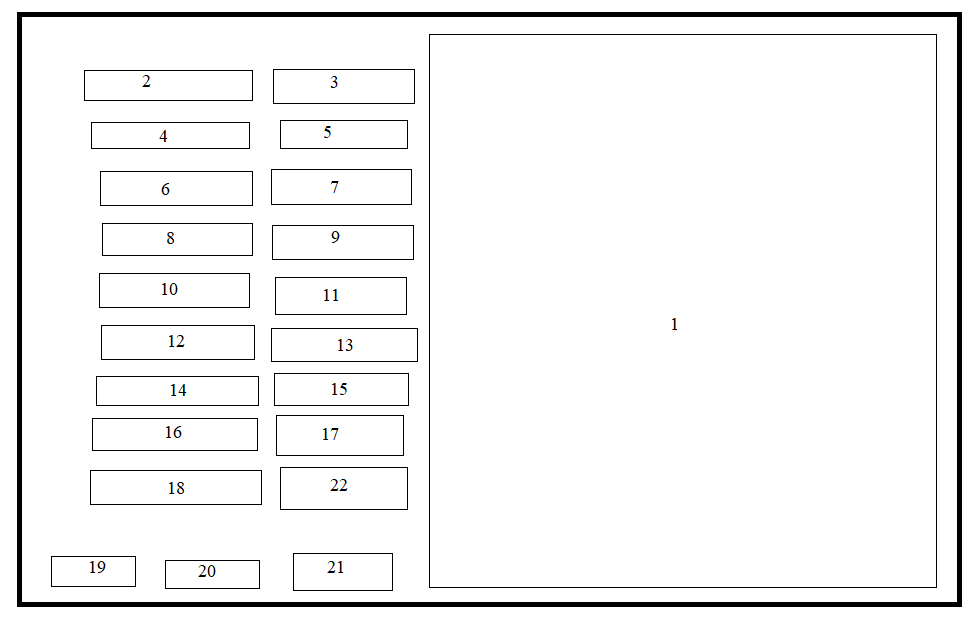


Рисунок 5 – Схема интерфейса формы «Компании»

Таблица 5 – Описание схемы интерфейса «Компаний»

| № | Описание |
| --- | --- |
| 1 | 2 |
| 1 | Таблица «Компании» |
| 2 | Надпись «Название компании» |
| 3 | Поле ввода название компании |
| 4 | Надпись «Фамилия руководителя» |
| 5 | Поле ввода фамилии руководителя |
| 6 | Надпись «Имя руководителя» |
| 7 | Поле ввода имени руководителя |
| 8 | Надпись «Отчество руководителя» |
| 9 | Поле ввода отчество руководителя |
| 10 | Надпись «Должность руководителя» |
| 11 | Поле ввода должности руководителя |
| 12 | Надпись «Фамилия руководителя в дательном падеже» |
| 13 | Поле ввода фамилии руководителя в дательном падеже |
| 14 | Надпись «Имя руководителя в дательном падеже» |
| 15 | Поле ввода имени руководителя в дательном падеже |
| 16 | Надпись «Отчество руководителя в дательном падеже» |
| 17 | Поле ввода отчества руководителя в дательном падеже |
| 18 | Надпись «Почта» |
| 19 | Кнопка «Добавить» |
| 20 | Кнопка «Изменить» |
| 21 | Кнопка «Удалить» |
| 22 | Поле ввода почты |

* 1. Схема интерфейса формы «Студенты»

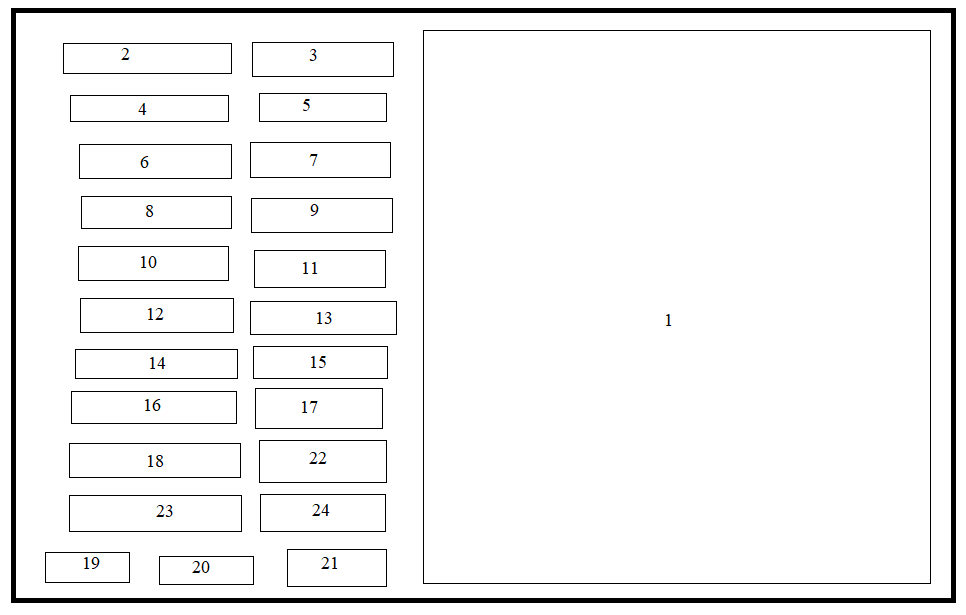


Рисунок 6 – Схема интерфейса формы «Студенты»

Таблица 6 – Описание схемы интерфейса «Студентов»

| № | Описание |
| --- | --- |
| 1 | 2 |
| 1 | Таблица «Студенты» |
| 2 | Надпись «Фамилия студента» |
| 3 | Поле ввода фамилии студента |
| 4 | Надпись «Имя студента» |
| 5 | Поле ввода имени студента |
| 6 | Надпись «Отчество студента» |
| 7 | Поле ввода отчества студента |
| 8 | Надпись «Группа студента» |
| 9 | Поле ввода группы студента |
| 10 | Надпись «Дата рождения» |
| 11 | Поле ввода даты рождения |
| 12 | Надпись «Почта студента» |
| 13 | Поле ввода почты студента |
| 14 | Надпись «Компания по практике» |
| 15 | Поле ввода компании по практике |
| 16 | Надпись «Фамилия студента в винительном падеже» |
| 17 | Поле ввода фамилии студента в винительном падеже |
| 18 | Надпись «Имя студента винительном падеже» |
| 19 | Кнопка «Добавить» |
| 20 | Кнопка «Изменить» |
| 21 | Кнопка «Удалить» |
| 22 | Поле ввода имени студента винительном падеже |
| 23 | Надпись «Отчество студента в винительном падеже» |
| 24 | Поле ввода отчества студента в винительном падеже |

* 1. Схема интерфейса формы «Даты практик»

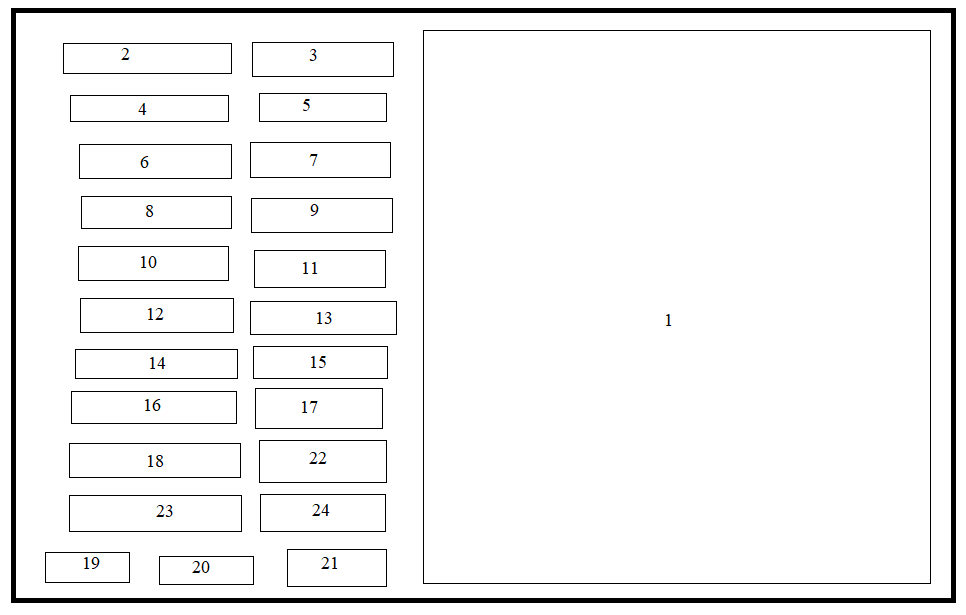


Рисунок 7 – Схема интерфейса формы «Даты практик»

Таблица 7 – Описание схемы интерфейса «Дат практик»

| № | Описание |
| --- | --- |
| 1 | 2 |
| 1 | Таблица «Даты практик» |
| 2 | Надпись «Профессиональный модуль» |
| 3 | Поле ввода профессионального модуля |
| 4 | Надпись «Название практики» |
| 5 | Поле ввода названия практики |
| 6 | Надпись «Преддипломная практика» |
| 7 | Поле выбора преддипломная практика или нет |
| 8 | Надпись «Номер группы» |
| 9 | Поле ввода номера группы |
| 10 | Надпись «Дата начала практики» |
| 11 | Поле ввода даты начала практики |
| 12 | Надпись «Дата остановки практики» |
| 13 | Поле ввода даты остановки |
| 14 | Надпись «Дата продолжения практики» |
| 15 | Поле ввода даты продолжения практики |
| 16 | Надпись «Дата остановки практики» |
| 17 | Поле ввода даты остановки практики |
| 18 | Надпись «Дата продолжения практики» |
| 19 | Кнопка «Добавить» |
| 20 | Кнопка «Изменить» |
| 21 | Кнопка «Удалить» |
| 22 | Поле ввода имени студента винительном падеже |
| 23 | Надпись «Дата окончания практики» |
| 24 | Поле ввода даты окончания практики |

* 1. Схема интерфейса формы «Документооборот»

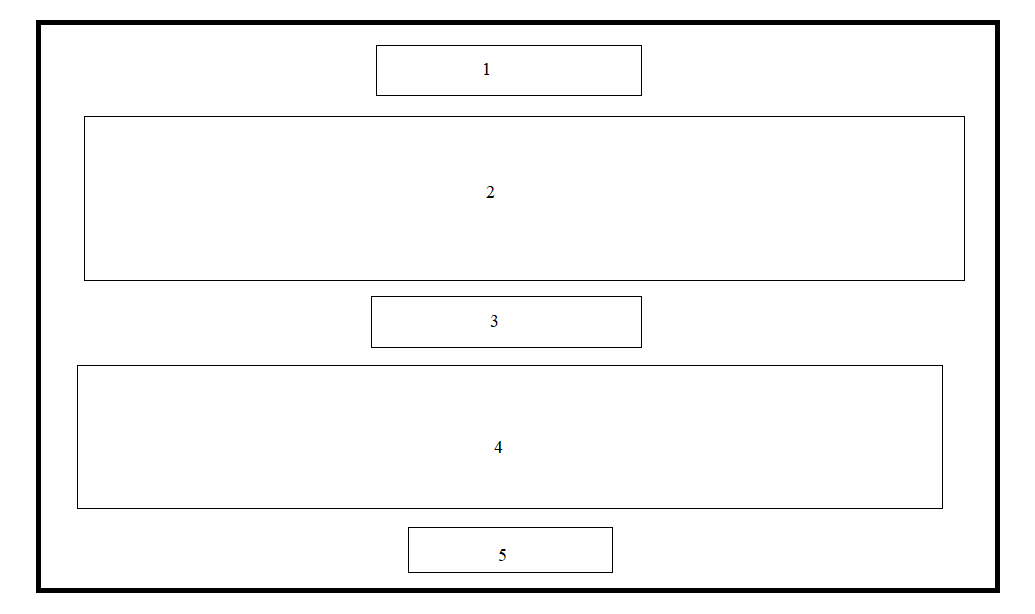


Рисунок 8 – Схема интерфейса формы «Документооборот»

Таблица 8 – Описание схемы интерфейса «Документооборот»

|  |  |
| --- | --- |
| № | Описание |
| 1 | 2 |
| 1 | Надпись «Студенты» |
| 2 | Таблица «Студент» |
| 3 | Надпись «Компании» |
| 4 | Таблица «Компании» |
| 5 | Кнопка создания документов |

* 1. Схема интерфейса формы «Группы»

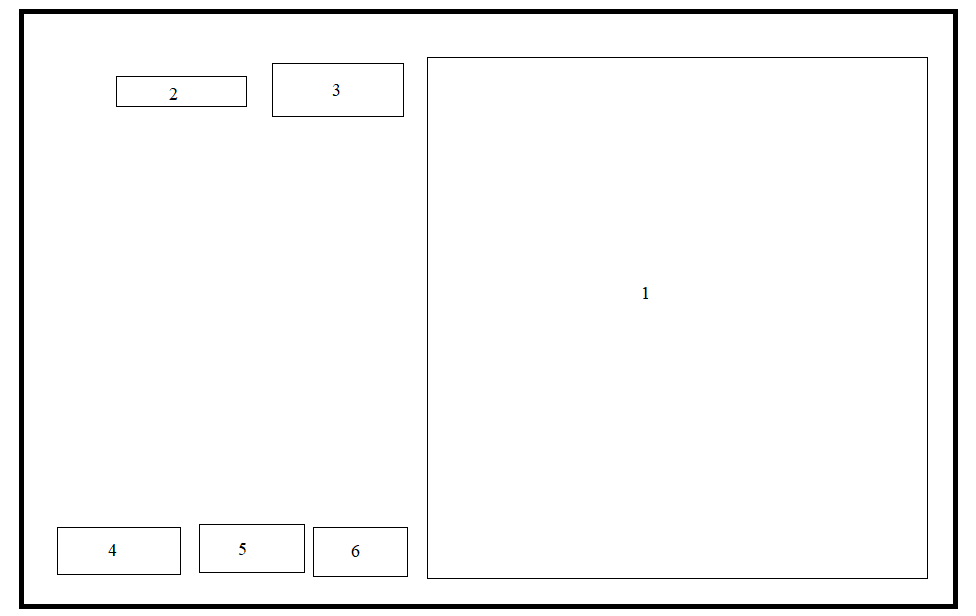


Рисунок 9 – Схема интерфейса формы «Группы»

Таблица 9 – Описание схемы интерфейса «Группы»

|  |  |
| --- | --- |
| № | Описание |
| 1 | 2 |
| 1 | Таблица «Группы» |
| 2 | Надпись «Название» |
| 3 | Поле ввода названия |
| 4 | Кнопка «Добавить» |
| 5 | Кнопка «Изменить» |
| 6 | Кнопка «Удалить» |

* 1. Схема интерфейса формы «Настройки»

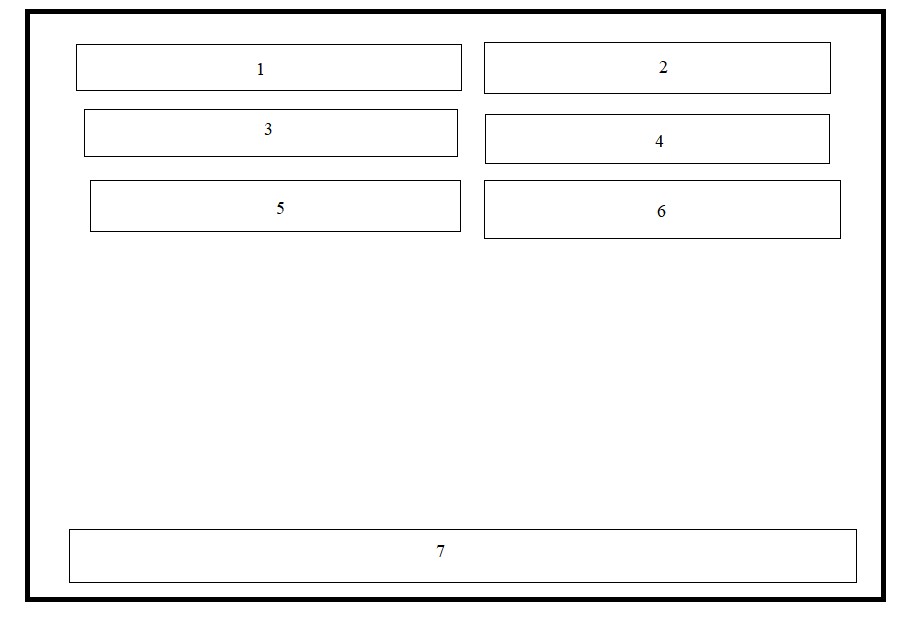


Рисунок 10 – Схема интерфейса формы «Настройки»

Таблица 10 – Описание схемы интерфейса «Настройки»

|  |  |
| --- | --- |
| № | Описание |
| 1 | 2 |
| 1 | Надпись «Место хранения итоговых документов» |
| 2 | Поле ввода места итогового хранения документов |
| 3 | Надпись «Место хранения шаблона направления студента на ПП» |
| 4 | Поле ввода места хранения шаблона направления студента на ПП |
| 5 | Надпись «Место хранения шаблона принятия студента на ПП» |
| 6 | Поле ввода места хранения шаблона принятия студента на ПП |
| 7 | Поле роль |

* 1. Схема интерфейса формы «Пользователь»

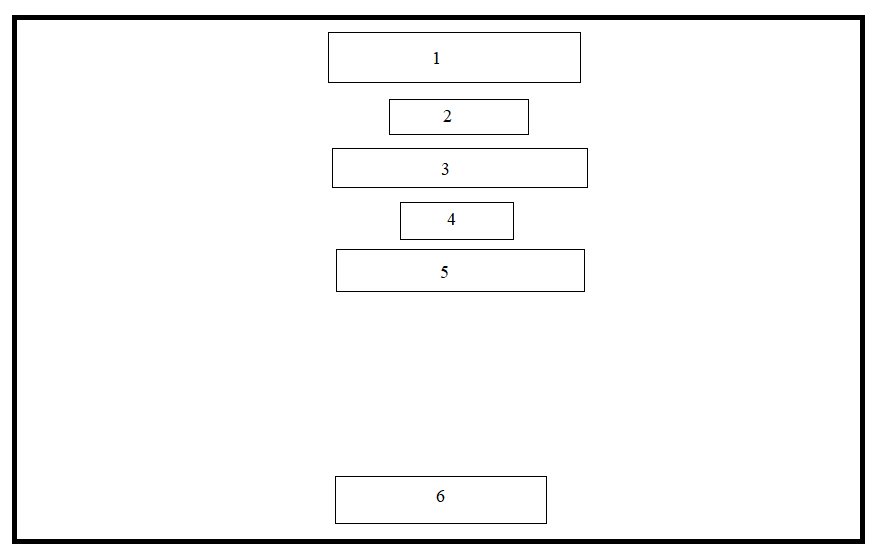


Рисунок 11 – Схема интерфейса формы «Пользователь»

Таблица 11 – Описание схемы интерфейса «Настройки»

|  |  |
| --- | --- |
| № | Описание |
| 1 | 2 |
| 1 | Надпись «Изменение пароля» |
| 2 | Надпись «Текущий пароль» |
| 3 | Поле ввода текущего пароля |
| 4 | Надпись «Новый пароль» |
| 5 | Поле ввода нового пароля |
| 6 | Кнопка «Изменить» |

# ПРИЛОЖЕНИЕ Е. СЛОВАРЬ ДАННЫХ

АННОТАЦИЯ

В данном программном документе приведён словарь данных информационной системы «WKR».

В разделе «Таблицы базы данных» предоставлена информация о структуре базе данных.

1. Таблица базы данных

В Таблице 1 представлена информация о базе данных.

Таблица 1 – Список таблиц

| № | Поле | Тип данных | Ограничения | Ключ |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Таблица Users | | | | |
| 1 | id | int | Identity | PK |
| 2 | Surname | int | Not null |  |
| 3 | Name | Varchar | Not null |  |
| 4 | Patronymic | Varchar | Not null |  |
| 5 | Email | Varchar | Not null |  |
| 6 | Login | Varchar | Not null |  |
| 7 | Password | Varchar | Not null |  |
| 8 | Birthday | date | Not null |  |
| 9 | Settings | int | Not null | FK |
| 10 | Role | Varchar | Not null |  |
| Таблица Settings | | | | |
| 1 | Id | int | Not null | PK |
| 2 | Document\_storage\_location | Varchar | Not null |  |
| 3 | Document\_template\_storage\_location | Varchar | Not null |  |
| 4 | Document2\_template\_storage\_location | Varchar | Not null |  |
| Таблица Students | | | | |
| 1 | Id | int | Not null | PK |
| 2 | Surname | Varchar | Not null |  |
| 3 | Name | Varchar | Not null |  |
| 4 | Patronymic | Varchar | Not null |  |
| 5 | [Group] | int | Not null | FK |
| 6 | Birthday | date | Not null |  |
| 7 | Email | Varchar | Not null |  |
| 8 | Practice\_company | int | Null | FK |
| 9 | Surname\_in\_the\_accusative\_case | Varchar | Not null |  |
| 10 | First\_name\_in\_the\_accusative\_case | Varchar | Not null |  |
| 11 | Patronymic\_in\_the\_accusative\_case | Varchar | Not null |  |
| Таблица Company | | | | |
| 1 | Id | int | Not null | PK |
| 2 | Name | Varchar | Not null |  |
| 3 | Manager\_Surname | Varchar | Not null |  |
| 4 | Manager\_Name | Varchar | Not null |  |
| 5 | Manager\_Patronymic | Varchar | Not null |  |
| 6 | Manager\_Position | Varchar | Not null |  |
| 7 | Manager\_surname\_in\_the\_dative\_case | Varchar | Not null |  |
| 8 | Manager\_name\_in\_the\_dative\_case | Varchar | Not null |  |
| 9 | Manager\_patronymic\_in\_the\_dative\_case | Varchar | Not null |  |
| 10 | Email | Varchar | Not null |  |
| Таблица Group | | | | |
| 1 | Id | int | Not null | PK |
| 2 | Group\_name | Varchar | Not null |  |
| 3 | Specialization | Varchar | Not null |  |
| Таблица Practice\_dates | | | | |
| 1 | Id | int | Not null | PK |
| 2 | Professional\_module | Varchar | Null |  |
| 3 | Name\_of\_the\_practice | Varchar | Null |  |
| 4 | Pre\_graduate\_practice | v | Not null |  |
| 5 | Group\_name | int | Not null | FK |
| 6 | Pactice\_start\_date | Date | Not null |  |
| 7 | Pactice\_termination\_date | Date | Null |  |
| 8 | Date\_of\_continuation\_of\_the\_practice | Date | Null |  |
| 9 | Pactice\_termination\_date\_2 | Date | Null |  |
| 10 | Date\_of\_continuation\_of\_the\_practice\_2 | Date | Null |  |
| 11 | Practice\_end\_date | Date | Not null |  |
| Таблица Role\_passwords | | | | |
| 1 | id | int | Not null | PK |
| 2 | Name | Varchar | Not null |  |
| 3 | Password | Varchar | Not null |  |
| Таблица Documents\_Flow | | | | |
| 1 | id | Int | Not null | PK |
| 2 | id\_Student | Int | Not null | FK |
| 3 | id\_Company | Int | Not null | FK |
| 4 | Document\_Type | Varchar | Not null |  |