Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

У Т В Е Р Ж Д А Ю

**Зав. кафедрой ВММБ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Ю. Столбов

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

***ЗАДАНИЕ***

***НА ПОРТФОЛИО***

***по дисциплине***

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Никулина Дарья Илларионовна, Лимонова Мария Владимировна ЦТУ-20-2Б

(фамилия, имя, отчество студента; группа)

1. **Тема портфолио**

Модуль корпоративной информационной системы сети онлайн-школ «Exam Masters»

2. **Срок сдачи студентом отчета:** « » \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

3. **Содержание отчета:**

В данном портфолио представлены отчеты по всем этапам разработки “Модуля корпоративной информационной системы сети онлайн-школ «Exam Masters»”, среди которых: техническое задание на разработку системы со сроками выполнения, проектирование хранилища данных на MySQL, проектирование интерфейса КИС на C# Windows Forms, разработка диаграммы классов, программная реализация и тестирование с выделением задач на развитие и доработку проекта.

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

**ПОРТФОЛИО**

по дисциплине

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Тема: **«Модуль корпоративной информационной системы сети аптек»**

Выполнили:

студенты гр. ЦТУ-20-2Б

Никулина Дарья Илларионовна

Лимонова Мария Владимировна

(Ф.И.О.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(подпись)*

Принял:

ст. преподаватель, Банников Р.Ю.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(должность, ФИО руководителя)*

*\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Пермь 2024**

# Содержание

[1. ТЗ на разработку КИС 4](#_Toc163217665)

[1.1. Введение 4](#_Toc163217666)

[1.2. Основания для разработки 4](#_Toc163217667)

[1.3. Функциональные требования к программе 4](#_Toc163217668)

[1.3.1. Авторизованный доступ 4](#_Toc163217669)

[1.3.2. Требования к функциональным характеристикам 5](#_Toc163217670)

[1.3.3. Возможность редактирования данных ИС 11](#_Toc163217671)

[1.3.4. Возможность генерации отчётов 11](#_Toc163217672)

[1.3.5. Реакция системы на исключительные ситуации 12](#_Toc163217673)

[1.4. Этапы и их сроки выполнения: 12](#_Toc163217674)

[1.5. Список литературы 13](#_Toc163217675)

[2. Проектирование хранилища данных КИС 14](#_Toc163217676)

[2.1. Постановка задачи 14](#_Toc163217677)

[2.1.1. Описание назначения КИС, описание основного функционала согласно ТЗ 14](#_Toc163217678)

[2.1.2. Цель работы и задачи, которые нужно выполнить для достижения цели 14](#_Toc163217679)

[2.2. Предметная область проектируемой БД 15](#_Toc163217680)

[2.2.3. Описание основных сущностей и связей между ними 15](#_Toc163217681)

[2.3. Логическая модель 17](#_Toc163217682)

[2.4. Физическая модель 17](#_Toc163217683)

[2.5. Содержимое таблиц БД 21](#_Toc163217684)

[2.6. Выводы 24](#_Toc163217685)

[3. Проектирование интерфейса КИС 25](#_Toc163217686)

[3.1. Постановка задачи 25](#_Toc163217687)

[3.1.1. Описание назначения КИС и основного функционала согласно ТЗ 25](#_Toc163217688)

[3.1.2. Цель работы и задачи, которые нужно выполнить для достижения цели 25](#_Toc163217689)

[3.2. Интерфейс 26](#_Toc163217690)

[3.2.1. Средства и среда проектирования интерфейса 26](#_Toc163217691)

[3.2.2. Элементы интерфейса 26](#_Toc163217692)

[3.3.3. Отчеты 36](#_Toc163217693)

[3.3.4. Реакция ИС на ошибочный ввод данных 37](#_Toc163217694)

[3.3. Выводы 39](#_Toc163217695)

[4. Разработка структур классов 40](#_Toc163217696)

[4.1. Постановка задачи 40](#_Toc163217697)

[4.1.1. Описание КИС и основного функционала 40](#_Toc163217698)

[4.1.2. Цель работы и задачи 40](#_Toc163217699)

[4.2. Диаграмма классов 41](#_Toc163217700)

[4.2.1. Средство разработки 41](#_Toc163217701)

[4.2.2. Описание диаграммы классов 42](#_Toc163217702)

[4.3. Выводы 47](#_Toc163217703)

[5. Программная реализация КИС 48](#_Toc163217704)

[5.1. Постановка задачи 48](#_Toc163217705)

[5.1.1. Описание КИС и основного функционала 48](#_Toc163217706)

[5.1.2. Цель работы и задачи 48](#_Toc163217707)

[5.2. Реализация 49](#_Toc163217708)

[5.2.1. Средства и среда разработки 49](#_Toc163217709)

[5.2.2. Работа основного функционала 51](#_Toc163217710)

[5.2.3. Отчеты 61](#_Toc163217711)

[5.2.4. Реакция ИС на ошибочный ввод данных 66](#_Toc163217712)

[5.3. Вывод 71](#_Toc163217713)

[6. Тестирование КИС 72](#_Toc163217714)

[6.1. Постановка задачи 72](#_Toc163217715)

[6.1.1. Описание КИС и основного функционала 72](#_Toc163217716)

[6.1.2. Цель работы и задачи 72](#_Toc163217717)

[6.2. Реализация тестирования 73](#_Toc163217718)

[6.2.1. Вид и параметры тестирования 73](#_Toc163217719)

[6.2.2. Тестирование основного функционала 74](#_Toc163217720)

[6.3. Задачи на доработку 89](#_Toc163217721)

[6.4. Выводы 90](#_Toc163217722)

[Приложение 91](#_Toc163217723)

[Классы форм 91](#_Toc163217724)

[Autorization 91](#_Toc163217725)

[Admin\_Main 96](#_Toc163217726)

[Admin\_AddData 97](#_Toc163217727)

[Admin\_DeleteData 102](#_Toc163217728)

[Admin\_Statistics 106](#_Toc163217729)

[Teacher\_Main 114](#_Toc163217730)

[Teacher\_Info 115](#_Toc163217731)

[Teacher\_Schedule 117](#_Toc163217732)

[Teacher\_Statistics 120](#_Toc163217733)

[Student\_Main 126](#_Toc163217734)

[Student\_Debts 126](#_Toc163217735)

[Student\_Schedule 128](#_Toc163217736)

[Объекты системы 130](#_Toc163217737)

[StudentClass 130](#_Toc163217738)

[TeacherClass 131](#_Toc163217739)

[Course 133](#_Toc163217740)

[Group 133](#_Toc163217741)

[Lesson 134](#_Toc163217742)

[Test 135](#_Toc163217743)

[Классы для работы с данными и БД 135](#_Toc163217744)

[Message 135](#_Toc163217745)

[DBConnection 136](#_Toc163217746)

[DB\_Do 136](#_Toc163217747)

# ТЗ на разработку КИС

## Введение

В качестве корпоративной информационной системы планируется разработать информационную систему *«Exam Masters»* – образовательный онлайн-модуль для осуществления учебного процесса и взаимодействия между руководителями, преподавателями и учениками.

## Основания для разработки

Современные онлайн-модули также оборудованы вспомогательными методами, которые помогают преподавателям вести грамотную индивидуальную работу с каждым учеником [1].

Наиболее известным и функциональным образовательным онлайн-модулем по мнению автора является *«Maximum Education»* [2]. Данная информационная система отлично подходит для взаимодействия учеников и преподавателей. Модуль содержит в себе все, что нужно для успешной подготовки к экзамену: точки контроля, практики, теоретические листы, видео-уроки и записи онлайн-занятий. Ученика также может привлечь большое разнообразие курсов и форматов обучения.

Более того, информационная система крайне удобна для преподавателя. Функционал содержит в себе расписание занятий, причем выставляется местное время занятия для всех к нему причастных людей.

Наиболее важной частью модуля является возможность проведения аналитики по ученикам. Система содержит в себе несколько сводных отчетов, в которых преподаватель может отследить динамику выполнения ДЗ и процент правильных ответов, посещаемость занятий, а также личные данные своих учеников, по которым он может с ними связаться.

Помимо представленной выше платформы, существуют достаточно известные веб-приложения такие, как *«Skillbox»*, *«Skyeng»*, *«Умскул»* и др. [3–5]. Все они имеют свои значимые особенности и ценности. А основой осуществления качественного дистанционного образования выступает использование современных информационных технологий.

Таким образом, резюмируя основные достоинства рассмотренных выше информационных систем для подготовки к экзаменам, было принято решение разработать корпоративную информационную систему для онлайн-школы *«Exam Masters».* Более подробно функционал системы будет рассмотрен в следующих разделах.

Система принята на разработку в рамках учебной дисциплины «Корпоративные информационные системы» по заказу Банникова Руслана Юрьевича.

## Функциональные требования к программе

### Авторизованный доступ

В программе должен быть авторизованный доступ к учетной записи каждого пользователя. Ко всему прочему, существует разграничение по типам пользователя. Ниже в Табл. 1 описано разграничение пользователей, а также указано краткое описание и основной функционал каждого типа пользователя:

Табл. 1. Разграничение типов пользователей в системе

|  |  |  |
| --- | --- | --- |
| **Роль пользователя** | **Описание** | **Функционал** |
| Администратор | Решает все технические вопросы, связанные с взаимодействием других типов пользователя и КИС. | Добавление новых учетных записей для пользователей;  Исправление ошибок, связанных с данными пользователей. |
| Ученик | Использует КИС для подготовки к экзаменам. Может просматривать только материал курса, к которому подключен. | Просмотр видеоуроков;  Просмотр теории по теме;  Решение тестов;  Редактирование учетной записи. |
| Преподаватель | Использует КИС для контроля и аналитики процесса обучения. Может просматривать информацию по каждому ученику, а также доступные ученику модули курса. | Просмотр видеоуроков;  Просмотр теории по теме;  Решение тестов;  Редактирование учетной записи;  Просмотр информации по ученикам (процент ДЗ, долги, посещаемость и т.д.);  Вывод отчетности по ученикам. |
| Руководитель | Использует КИС для контроля и анализа обучения и качества преподавания. Может просматривать полный функционал преподавателя и ученика. | Просмотр видеоуроков;  Просмотр теории по теме;  Решение тестов;  Редактирование учетной записи;  Просмотр информации по ученикам (процент ДЗ, долги, посещаемость и т.д.);  Вывод отчетности по ученикам;  Вывод отчетности по преподавателям. |

Из Табл. 1, представленной выше можно видеть, что функционал типов пользователей имеет большие различия. Поэтому в следующем подразделе 3.2. более подробно расписаны основные функции, доступные каждому пользователю.

### Требования к функциональным характеристикам

Как говорилось ранее, в системе «Exam Masters» существует разграничение пользователей по трем основным ролям: ученик, преподаватель, руководитель. В настоящем разделе ТЗ на разработку более детально расписан функционал для каждой роли.

**Роль «Ученик»** использует КИС для подготовки к экзаменам (ОГЭ, ЕГЭ, ВПР). Может просматривать только материал курса, к которому подключен. Ученик может быть подключен сразу к нескольким курсам. Основные активности ученика – это выполнение домашней практики, просмотр видеоуроков, решение домашней практики, а также проверочных, контрольных работ и пробных вариантов экзамена.

Ниже на Рис. 1представлена диаграмма прецедентов для роли «Ученик»:

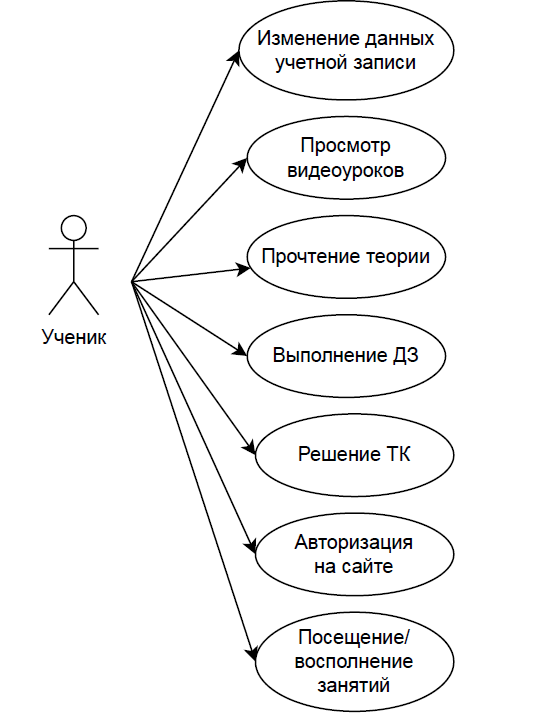


Рис. 1. Диаграмма прецедентов для роли «Ученик»

Помимо этого, следует более детально расписать бизнес-процессы, доступные для роли «Ученик». Ниже в Табл. 2 описан функционал системы для данного типа пользователя:

Табл. 2. Описание функционала для роли «Ученик»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование бизнес-процесса | Описание операции | Информация на входе | Информация на выходе |
| Авторизация на сайте | Ученик вводит логин и пароль на начальной странице в поле авторизации. | Пользователь зарегистрирован в системе. | Вход выполнен успешно при корректном вводе данных. |
| Изменение данных учетной записи | Ученик может изменить пароль, почту или номер телефона на актуальный. При вводе нового пароля требуется подтвердить операцию при помощи ввода старого пароля. | Пользователь зарегистрирован и авторизован в системе. | Успешно обновленные данные (пароль, электронная почта, номер телефона) |
| Просмотр видеоуроков | Ученик переходит на страницу какой-либо темы. Чтобы просмотреть видеоурок, пользователь должен перейти на вкладку видео. | – | Открывается вкладка видео, которое просматривается на странице сайта. |
| Прочтение теории | Ученик переходит на страницу какой-либо темы. Чтобы просмотреть видеоурок, пользователь должен перейти на вкладку теории. | – | Открывается вкладка теория на странице сайта. |
| Выполнение ДЗ | Ученик переходит на страницу какой-либо темы. Чтобы выполнить ДЗ, пользователь должен перейти на вкладку практики. Далее ученик приступает к решению, вводя ответы в указанные поля. | – | Открывается вкладка практика на странице сайта. Можно приступать к выполнению ДЗ. |
| Решение ТК | Ученик переходит на страницу, которая отображает все точки контроля. Далее он выбирает ту работу, которую планирует решить, после чего нажимает на нее. | – | Открывается вкладка с точкой контроля. |

**Роль «Преподаватель»** может выполнять те же действия, что и ученик, используя свою учетную запись, однако его функционал является более обширным. Учитель составляет общую картину по каждому ученику на курсе. Преподаватель должен осуществлять контроль как над каждым учеником, так и над всей группой одновременно. Функционал системы для преподавателя представляет собой просмотр данных об учениках: фамилия и имя, номер телефона, e-mail, процент выполнения домашнего задания, результаты контрольных точек, посещаемость и т.д.

Ниже на Рис. 2 представлена диаграмма прецедентов для роли «Преподаватель»:

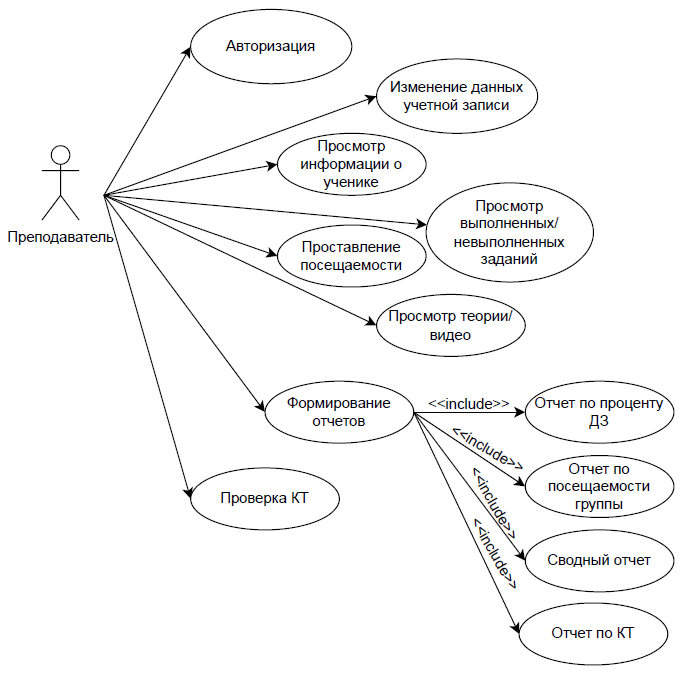


Рис. 2. Диаграмма прецедентов для роли «Преподаватель»

Далее следует пояснить каждый бизнес-процесс, который указан на Рис. 2. Более подробно функции преподавателя расписаны ниже в Табл. 3:

Табл. 3. Описание функционала роли «Преподаватель»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование бизнес-процесса | Описание операции | Информация на входе | Информация на выходе |
| Авторизация на сайте | Преподаватель вводит логин и пароль на начальной странице в поле авторизации. | Пользователь зарегистрирован в системе. | Вход выполнен успешно при корректном вводе данных. |
| Изменение данных учетной записи | Преподаватель может изменить пароль, почту или номер телефона на актуальный. При вводе нового пароля требуется подтвердить операцию при помощи ввода старого пароля. | Пользователь зарегистрирован и авторизован в системе. | Успешно обновленные данные (пароль, электронная почта, номер телефона) |
| Просмотр информации по ученику | Преподаватель видит всю информацию по ученику: почта, номер телефона, выполняемость ДЗ, просмотр видео и теории, выполнение КТ. | – | – |
| Просмотр выполненных и невыполненных заданий | Перейдя на вкладку группы, преподаватель видит информацию о выполненных и невыполненных заданиях домашней практики. | – | – |
| Проставление посещаемости | Перейдя на вкладку группы в раздел посещаемости преподаватель может проставлять статус занятия: посещено / пропущено / восполнено. | – | Введенные данные о посещаемости. |
| Формирование отчетов по группе | Преподаватель может составить автоматический отчет по показателям группы. На выбор даются следующие варианты отчетности: отчет по проценту ДЗ каждого ученика, отчет по посещаемости занятий, сводный отчет (отчет по всем показателям), отчет по контрольным точкам. | – | Сформированный отчет в формате *.xls* |
| Проверка контрольных точек | Преподаватель может посмотреть результаты контрольной работы, перейдя на страницу группы во вкладку «Контрольные точки» и выбрав нужную контрольную точку. | – | – |

**Роль «Руководитель»**. Всем руководителям в подразделение распределяется определенное количество преподавателей. Его задача осуществлять контроль над качеством преподавания и следить за показателями каждого ученика на курсе. Таким образом, руководитель может выводить статистику как по каждому ученику, так и по каждому преподавателю.

На Рис. 3 представлена диаграмма прецедентов для роли «Руководитель»:

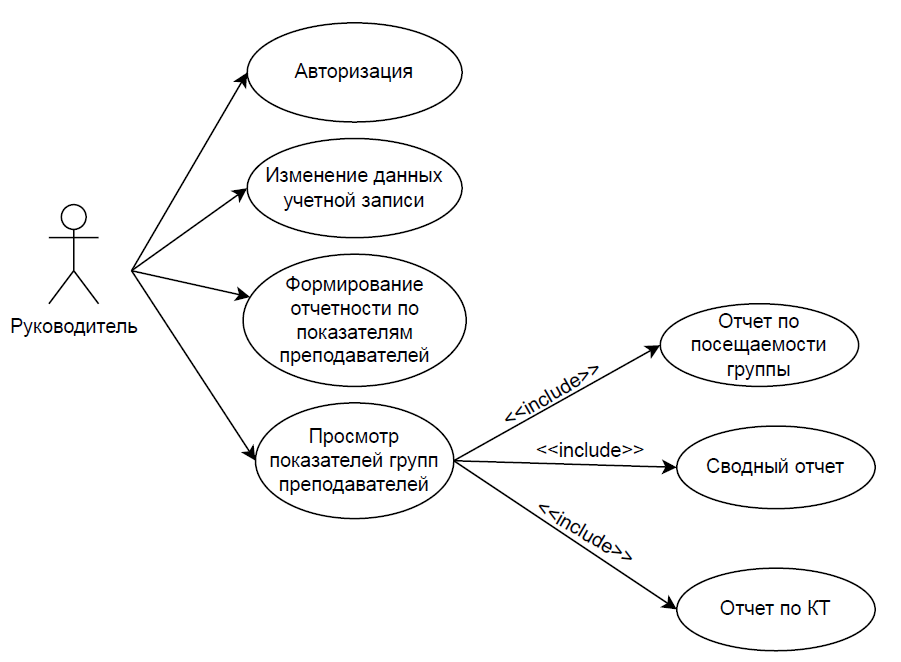


Рис. 3. Диаграмма прецедентов для роли «Руководитель»

Как и было сказано выше, роль «Руководитель» имеет все доступные роли «Преподаватель» функции. Однако, для него также доступен дополнительный функционал, который расписан более детально ниже в Табл. 4:

Табл. 4. Описание функционала для роли «Руководитель»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование бизнес-процесса | Описание операции | Информация на входе | Информация на выходе |
| Авторизация на сайте | Руководитель вводит логин и пароль на начальной странице в поле авторизации. | Пользователь зарегистрирован в системе. | Вход выполнен успешно при корректном вводе данных. |
| Изменение данных учетной записи | Руководитель может изменить пароль, почту или номер телефона на актуальный. При вводе нового пароля требуется подтвердить операцию при помощи ввода старого пароля. | Пользователь зарегистрирован и авторизован в системе. | Успешно обновленные данные (пароль, электронная почта, номер телефона) |
| Просмотр групп всех преподавателей | Руководитель может просматривать показатели группы преподавателя (посещаемость, ДЗ, КТ). Таким образом, руководитель имеет представление о каждом ученике. | – | – |
| Формирование отчета | Руководитель может сформировать автоматический сводный отчет по показателям преподавателей (процент ДЗ и посещаемость по группе). Файл выгружается на ПК в формате *.xls.* | – | Сформированный отчет в формате *.xls.* |

### Возможность редактирования данных ИС

В системе также должна поддерживаться возможность редактирования данных. Однако, должен существовать перечень данных для изменения и разграничение пользователей, которые могут эти данные изменять. Ниже в Табл. 5 приведены основные функции, которые предполагают собой изменение данных:

Табл. 5. Разграничение редактирования данных по ролям пользователей

|  |  |  |
| --- | --- | --- |
| **Данные для редактирования** | **Вариация ввода данных** | **Роль пользователя, который может редактировать данные** |
| Изменение пользовательских данных | Строковый тип данных | Ученик, преподаватель, руководитель |
| Выполнение домашнего задания, КТ | Строковый / числовой тип данных | Ученик |
| Проставление посещаемости | Строковый тип данных  (Выбор из вариантов: посещено, пропущено, восполнено) | Преподаватель |
| Проверка КТ | Целочисленный тип данных (оценка) от 0 до 100 | Преподаватель |

### Возможность генерации отчётов

В системе присутствует возможность выгружать автоматизированные отчеты в формате *.xls.* Отчеты формируются на основе данных, обработанных и полученных из хранилища системы. Ниже в Табл. 6 описаны основные формы генерации отчетности, предусмотренные в разрабатываемой КИС:

Табл. 6. Типы генерации отчетности

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование отчета | Описание | Роль пользователя | Формат файла |
| Отчетность процента ДЗ по группе | Выводит отчет процентного соотношения ДЗ по каждому занятию и ученику точечно, а также средний процент по группе. | Преподаватель | *.xls* |
| Отчетность посещаемости занятий по группе | Выводит отчет посещаемости по каждому занятию и ученику точечно, а также средний процент по группе. | Преподаватель | *.xls* |
| Сводная отчетность по группе | Выводит отчет по общим показателям ДЗ и посещаемости каждого ученика, а также общий процент по группе. | Преподаватель | *.xls* |
| Отчет по КТ | Выводит отчет по решенным и нерешенным заданиям каждого ученика контрольной точки. В последней строке файла выводится процент выполняемости задания по группе. | Преподаватель | *.xls* |
| Отчет рейтинга учеников в группе | Выводится отчет рейтинга учеников по убыванию по бальной системе (посещение занятий, выполнение ДЗ, написание КТ) | Преподаватель |  |
| Сводный отчет по группам | Выводятся наименование группы, имя преподавателя и проценты решения ДЗ и посещаемости на данный момент | Руководитель | *.xls* |
| Отчетность для просмотра долгов | Выводятся темы и количество заданий, которые не были решены в этих темах. | Ученик | *.xls* |

### Реакция системы на исключительные ситуации

В данном случае, в качестве исключительной ситуации принято считать любой некорректный вывод данных в системе. Чтобы предотвратить непредвиденную поломку приложения и его хранилища следует на этапе разработки включить в проект обработку исключений. Если в систему ввелись некорректные значения, к примеру, не тот тип данных, следует вывести пользователю сообщение о том, что был произведен неправильный ввод данных, и автоматически очистить используемое поле для повторного ввода.

## Этапы и их сроки выполнения:

После описания основного функционала системы были рассмотрены планы на дальнейшие проектирование и разработку. Ниже в Табл. 7 представлены основные этапы разработки КИС *«Exam Masters»,* а также сроки их проведения и конечная форма отчетности, которая будет представляться по окончанию проведения каждой стадии проекта.

Табл. 7. Постановка этапов и их сроков начала и завершения

|  |  |  |
| --- | --- | --- |
| **Наименование этапа** | **Сроки проведения** | **Форма отчетности** |
| Проектирование хранилища | 06.02.2024 – 13.02.2024 | Отчет |
| Проектирование интерфейса | 13.02.2024 – 20.02.2024 | Отчет |
| Разработка диаграммы классов | 20.02.2024 – 27.02.2024 | Отчет |
| Программная реализация | 27.02.2024 – 19.03.2024 | Отчет |
| Тестирование | 19.03.2024 – 26.03.2024 | Отчет |

## Список литературы

1. Закутасова Н.Д., Белов В.А., Проблемы и преимущества онлайн-образования // Современное образование: содержание, технологии, качество. – 2020. – №1. – С.346-349.
2. Веб-приложение «Maximum Education» [Электронный ресурс]. – Режим доступа: https://maximumtest.ru/ (Дата обращения: 04.02.2024).
3. Веб-приложение «Skillbox» [Электронный ресурс]. – Режим доступа: <https://skillbox.ru/> (Дата обращения 04.02.2024).
4. Веб-приложение «Skyeng» [Электронный ресурс]. – Режим доступа: <https://skyeng.ru/> (Дата обращения 04.02.2024).
5. Веб-приложение «Умскул» [Электронный ресурс]. – Режим доступа: <https://umschool>

# Проектирование хранилища данных КИС

## Постановка задачи

### Описание назначения КИС, описание основного функционала согласно ТЗ

Наименование системы: «Модуль корпоративной информационной системы функционирования филиалов онлайн-школы “Exam Masters”».

Модуль корпоративной информационной системы онлайн-школы предназначен для обеспечения пользователям удобного учебного процесса как со стороны учеников, так и со стороны преподавателей и руководства.

Внутри системы пользователи смогут:

1. Добавлять данные в разделы в соответствии с ТЗ;
2. Просматривать статистики;
3. Выгружать отчеты, описанные в ТЗ.

В системе существует разграничение по правам пользователей. Преподавателям доступны следующие функции:

1. Просмотр групп, у которых он ведет занятия;
2. Просмотр каждого отдельного ученика в группе, анализ его статистики;
3. Выставление оценок, корректирование данных в соответствии с ТЗ;
4. Выгрузка отчетов по успеваемости, посещаемости и т.д.;
5. Функции по просмотру статистики, описанной в ТЗ;
6. Функции по выгрузке отчетов из разделов, описанных в ТЗ.

Ученикам доступны следующие функции:

1. Добавление данных по разделам, описанным в ТЗ;
2. Удаление данных по разделам, описанным в ТЗ;
3. Просмотр статистики по персональной успеваемости, посещаемости и т.д.;
4. Выгрузка отчетов по персональной успеваемости.

### Цель работы и задачи, которые нужно выполнить для достижения цели

Цель работы: разработать базу данных для реализации модуля корпоративно-информационной системы сети онлайн-школ для подготовки к экзаменам (ВПР, ЕГЭ, ОГЭ).

Задачи:

1. Выделить основные сущности системы;
2. Определить связи между сущностями;
3. Описать предметную область;
4. Построить логическую модель и описать её;
5. Построить физическую модель и описать её;
6. Составить тестовые данные для БД;
7. Сделать выводы о проделанной работе.

## Предметная область проектируемой БД

### 2.2.3. Описание основных сущностей и связей между ними

Согласно ТЗ, для функционирования КИС на начальном этапе следует выделить основные сущности системы, на основе которых и будет осуществляться работа с данными (добавление, удаление, редактирование и т.д.). Помимо этого, следует также прописать взаимосвязи между основными сущностями, а также указать их неотъемлемые атрибуты.

В хранилище данных поддерживаются следующие основные сущности системы: школы, преподаватели, ученики, курсы, уроки и КТ. Более подробно Сущности описаны ниже в Табл. 8:

Табл. 8. Описание основных сущностей и их атрибутов хранилища данных КИС

|  |  |  |
| --- | --- | --- |
| **Название сущности** | **Описание сущности** | **Атрибуты** |
| Филиал | Онлайн-школа имеет филиалы в разных городах РФ. Чтобы информация о разных филиалах была в доступе, было принято решение создать таблицу «Филиалы». По этой таблице можно посмотреть, в каких городах находятся филиалы, а также их контактная информация. | * Код филиала * Город * Адрес * Телефон администратора * Почта филиала |
| Преподаватель | Каждый преподаватель прикрепляется к одному или к нескольким филиалам. В таблице «Преподаватели» должна быть собрана основная информация о каждом из числящихся преподавателей. Так как в системе существует четкое разделение по ролям, разграничение в хранилище данных так же необходимо. | * Код преподавателя * ФИО * Почта * Телефон преподавателя * Пароль учетной записи |
| Группа | В онлайн-школе поддерживается только групповой формат, поэтому для удобного отображения данных принято решение добавить отдельную таблицу «Группа». Преподаватель сможет видеть основную информацию о каждой группе, в которой преподает, а также отдельно следить за каждым учеником. | * Код группы * Код преподавателя * Наименование группы * Дата старта занятий * Дата окончания занятий |
| Ученик | Группа состоит из учеников, отдельная таблица по ученикам позволяет преподавателю осуществлять контроль над успеваемостью и показателями каждого ученика на курсе.  Ученик – это вторая роль, предусмотренная в системе. Поэтому в хранилище данных также должна быть вся нужная информация по ученику. | * Код ученика * Код группы * ФИО * Пароль * Почта ученика * Номер телефона ученика |
| Курс | Так как курсы унифицированы по предметам для каждой группы, для каждой группы можно подключать один и тот же предметный курс. В курсе можно найти уроки, контрольные работы и все остальные активности, которые выполняются учениками и преподавателем. | * Код курса * Название курса * Описание курса * Стоимость курса * Количество уроков |
| Занятие | В одном уроке много данных, из-за чего было принято решение объединить его в отдельную сущность, а в атрибутах указать основные аспекты, которые есть у каждого урока. | * Код занятия * Код курса * Код преподавателя * Название урока * Видео по курсу * Домашнее задание |
| КТ | Контрольные точки, которые выполняет ученик в своем личном кабинете. Характер контрольных точек может меняться (контрольная работа, проверочная работа, опрос по теории, симуляция экзамена и т.д.). Тесты выполняет ученик выполняет в своем личном кабинете. Преподаватель же может проверить результаты и вывести отчетность по ним. | * Код КТ * Код курса * Наименование КТ * Тип КТ * Время выполнения КТ |

В роли ученика, пользователь может просматривать свою статистику, посещать занятия, выполнять контрольные и домашние работы. Помимо этого, ученик сможет просматривать расписание и список доступных ему курсов.

Преподаватель может просматривать как всю группу, так и каждого ученика по отдельности. Может менять данные по посещаемости, выставлять оценки за КТ и т.д. Помимо этого, преподаватель может выгружать отчетную статистику по каждому ученику в отдельности и по всей группе.

## Логическая модель

Перед тем, как было создано хранилище данных для корпоративной информационной системы «Exam Masters», была составлена логическая модель будущей БД. Она представлена ниже на Рис. 4:

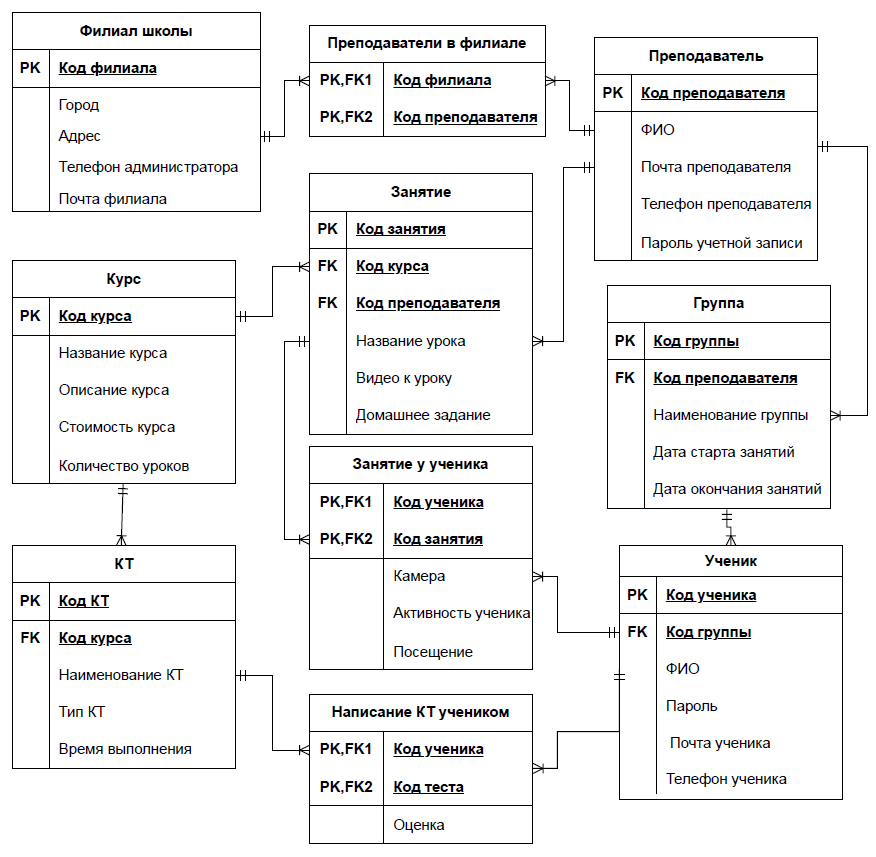


Рис. 4. Логическая модель БД

## Физическая модель

Далее на основе логической модели была разработана физическая модель хранилища данных для корпоративной информационной системы «Exam Masters». Ниже на Рис. 5 представлена физическая модель, которая была разработана:

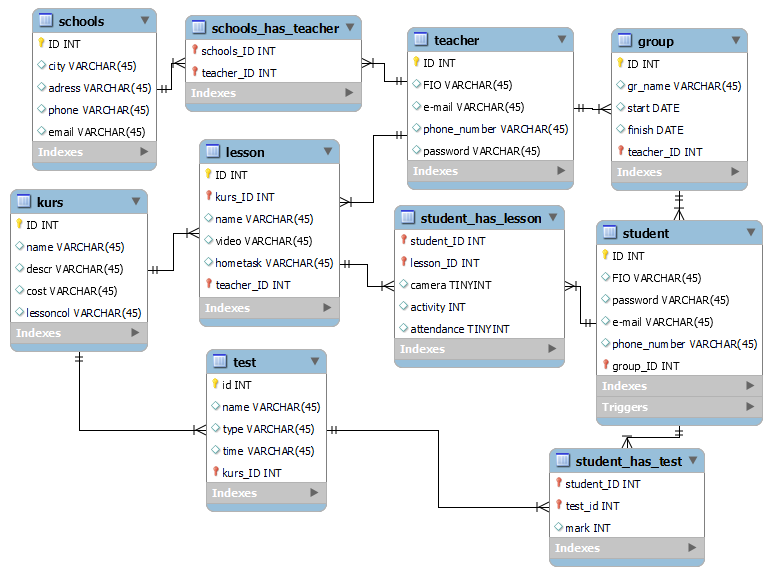


Рис. 5. Физическая модель БД

***Таблица «schools»*** *–* таблица для хранения информации о филиалах школы.

* + - 1. ID – идентификатор филиала;
      2. City – город, в котором находится филиал;
      3. Address – адрес филиала;
      4. Phone – номер рабочего телефона филиала;
      5. Email – электронная почта филиала.

***Таблица «schools\_has\_teacher» –*** таблица для хранения данных о преподавателях разных филиалов. Дает возможность реализовать связь «многие-ко-многим».

1. Schools\_ID – идентификатор филиала (внешний ключ);
2. Teacher\_ID – идентификатор преподавателя (внешний ключ);

***Таблица «teacher»*** - таблица для хранения информации о преподавателях.

1. ID – идентификатор преподавателя;
2. FIO – ФИО преподавателя;
3. Email– электронная почта преподавателя;
4. Phone – номер телефона преподавателя;
5. Password– пароль для авторизации преподавателя в системе.

***Таблица «group»*** – таблица, в которой хранится основная информация о группе.

1. ID – идентификатор группы;
2. Gr\_name – наименование группы;
3. Start – дата начала занятий;
4. Finish – дата окончания занятий;
5. Teacher\_ID – идентификатор преподавателя (внешний ключ).

***Таблица «student» –*** таблица, в которой хранится информация о каждом отдельном ученике.

1. ID – идентификатор ученика;
2. FIO – ФИО ученика;
3. Email– электронная почта ученика;
4. Phone – номер телефона ученика;
5. Password– пароль для авторизации ученика в системе;
6. Group\_ID – идентификатор группы (внешний ключ).

***Таблица «lesson»*** – таблица для хранения информации о уроках.

1. ID – идентификатор занятия;
2. Name – наименование урока или тема;
3. Video – видеоурок;
4. Hometask – домашнее задание к уроку;
5. Kurs\_ID – идентификатор курса (внешний ключ);
6. Teacher\_ID – идентификатор группы (внешний ключ).

***Таблица «student\_has\_lesson» –***таблица для хранения данных о успеваемости учеников на уроке. Помогает реализовать связь «многие-ко-многим».

1. Camera – Наличие включенной камеры на занятии;
2. Activity – оценка за работу на уроке, уровень активности;
3. Attendance – посещаемость занятия;
4. Student\_ID – идентификатор ученика (внешний ключ);
5. Lesson\_ID – идентификатор урока (внешний ключ).

***Таблица «kurs»*** – таблица, в которой хранятся данные о каждом курсе.

1. ID – идентификатор курса;
2. Name – наименование курса;
3. Descr – описание курса;
4. Cost – стоимость курса;
5. LessonCol – количество уроков в курсе.

***Таблица «test»*** – таблица, в которой хранятся данные о контрольных точках курса.

1. ID – идентификатор контрольной точки;
2. Name – наименование контрольной точки;
3. Type – тип контрольной точки;
4. Time – время написания контрольной точки;
5. Kurs\_ID – идентификатор курса (внешний ключ).

***Таблица «student\_has\_test»*** – таблица, в которой хранятся данные о прохождении контрольных точек курса каждым учеником. Позволяет реализовать связь «многие-ко-многим».

1. Mark – оценка за контрольную точку;
2. Student\_ID – идентификатор ученика (внешний ключ);
3. Test\_ID – идентификатор контрольной точки (внешний ключ).

Помимо этого, следует указать, какие основные типы используются в хранилище данных, а также указать, почему именно такие были предусмотрены. Более подробно этот вопрос рассмотрен в Табл. 9 ниже:

Табл. 9. Описание типов данных, присвоенных атрибутам таблицы

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип** | **Таблица** | **Атрибуты** | **Описание** |
| **INT** | *Schools* | *ID* | В данном случае ***для идентификаторов*** было принято решение взять INT, так как этот тип данных позволяет наиболее просто создавать уникальный ключ бля каждого кортежа таблицы.  Помимо этого, для ***переменной Mark*** также было принято решение взять тип данных INT, так как оценка – это целое положительное число (от 0 до 100). |
| *Schools\_has\_teacher* | *School\_ID*  *Teacher\_ID* |
| *Teacher* | *ID* |
| *Group* | *ID*  *Teacher\_ID* |
| *Student* | *ID*  *Student\_ID* |
| *Student\_has\_lesson* | *Student\_ID*  *Lesson\_ID*  *Activity* |
| *Lesson* | *ID*  *kurs \_ID*  *teacher\_ID* |
| *Kurs* | *ID*  *Lessoncol* |
| *Test* | *ID*  *Kurs\_ID* |
| *Student\_has\_test* | *Student\_ID*  *Test\_ID*  *Mark* |
| **VARCHAR** | *Schools* | *City*  *Address*  *Phone*  *Email* | Для работы с переменными, представленными слева, было принято решение взять строковый тип данных, так как каждая из записей должна содержать в себе цифры, буквы и иные символы (номера телефонов, адреса электронных почт и т.д.) |
| *Teacher* | *FIO*  *Email*  *Phone*  *Password* |
| *Group* | *Gr\_name* |
| *Student* | *FIO*  *Email*  *Phone*  *Password* |
| *Lesson* | *Name*  *Video*  *Hometask* |
| *Kurs* | *Name*  *Descr*  *Cost* |
| *Test* | *Name*  *Type*  *Time* |
| **DATE** | *Group* | *Start*  *Finish* | Тип данных DATE используется для тех переменных, которые из себя представляют дату (начало и окончание занятий) |
| **TINYINT** | *Student\_has\_lesson* | *Attendance*  *Camera* | Является аналогом «булевой» переменной. Иными словами, описывает вариацию двух ответов на вопрос: да/нет. Применимо для посещаемости, к примеру. |

## Содержимое таблиц БД

Помимо проектирования хранилища данных для корпоративной информационной системы функционирования филиалов онлайн-школы «Exam Masters» следует разработать прототип информационной системы. Для тестового модуля следует также внедрить в хранилище тестовые данные, которые помогут более детально проверить функционирование системы.

Для начала были расписаны тестовые данные для таблицы «Филиалы». Данные представлены на Рис. 6:

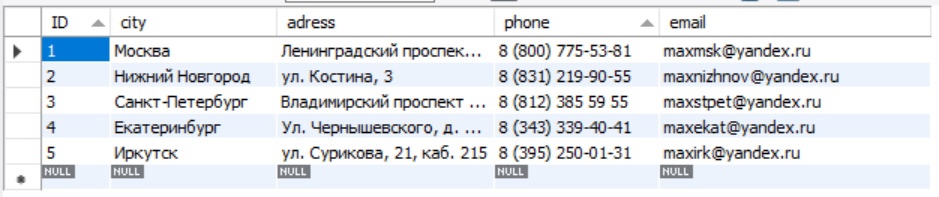


Рис. 6. Тестовые данные для таблицы «Филиалы»

Далее были добавлены данные для таблицы «Преподаватели». Тестовые записи таблицы представлены на Рис. 7:

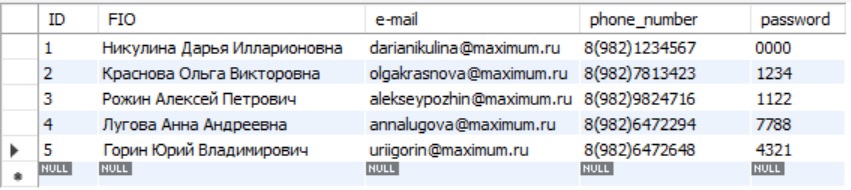


Рис. 7. Тестовые данные для таблицы «Преподаватели»

Далее были добавлены данные для вспомогательной таблицы «Преподаватели в филиалах», которая помогает реализовывать в системе связь «многие-ко-многим». Записи этой таблицы представлены ниже на Рис. 8:

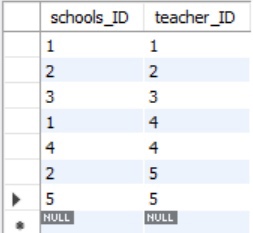


Рис. 8. Тестовые данные для таблицы «Преподаватели в филиалах»

Далее были прописаны тестовые данные для таблицы «Группы». Ниже на Рис. 9 представлены данные для этой таблицы:

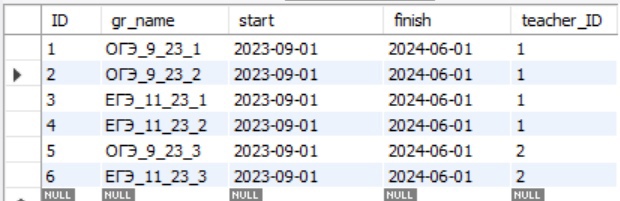


Рис. 9. Тестовые данные для таблицы «Группы»

После этого была заполнена таблица «Ученики». Ниже на Рис. 10 представлены тестовые данные для этой таблицы:

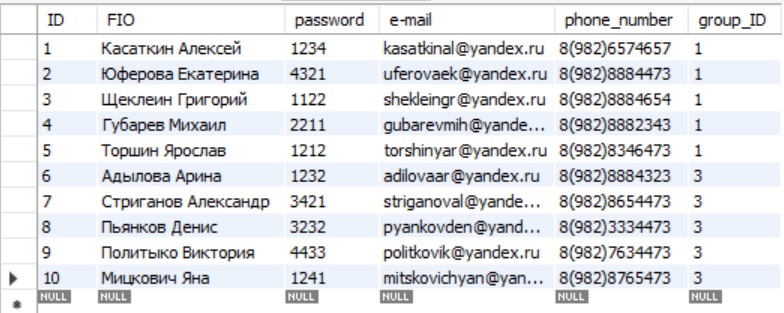


Рис. 10. Тестовые данные для таблицы «Ученики»

Далее были составлены тестовые записи таблицы «Занятия». Ниже на Рис. 11 представлены записи этой таблицы:

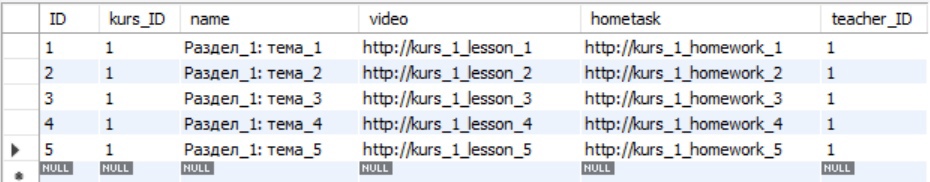


Рис. 11. Тестовые данные для таблицы «Занятия»

Так как между таблицами «Ученики» и «Занятия» реализована связь «Многие-ко-многим». Была автоматически создана дополнительная таблица «Занятие для ученика». Ниже на Рис. 12 представлены тестовые данные, которые были внесены в эту таблицу:

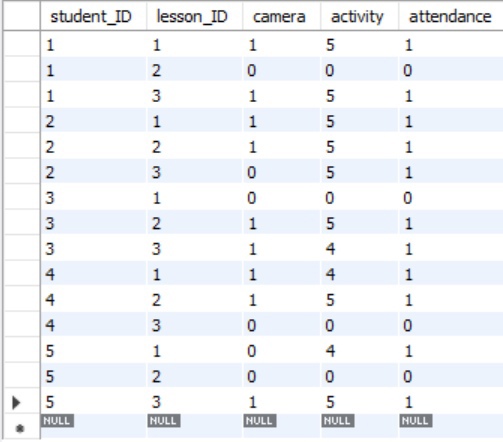


Рис. 12. Тестовые данные для таблицы «Занятие для ученика»

Далее были составлены тестовые данные для таблицы «Курсы». Тестовые записи этой таблицы представлены на Рис. 13:

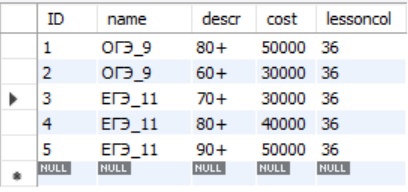


Рис. 13. Тестовые данные для таблицы «Курсы»

После этого была заполнена таблица «Контрольные точки». Тестовые данные для этой таблицы представлены на Рис. 14:

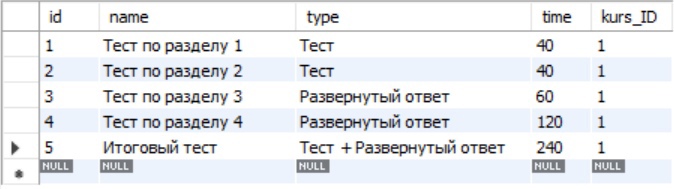


Рис. 14. Тестовые данные для таблицы «Контрольные точки»

Последним действием является заполнение таблицы «Контрольные точки у ученика». Тестовые записи для нее представлены на Рис. 15:

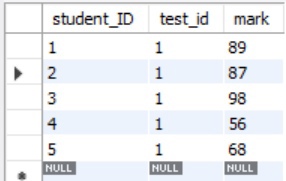


Рис. 15. Тестовые данные для таблицы «Контрольные точки ученика»

## Выводы

В ходе лабораторной работы были выполнены все основные задачи:

1. Выделены основные сущности системы;
2. Описана взаимосвязь основных сущностей;
3. Построена логическая и физическая модель БД;
4. Собрана БД в СУБД MySQL;
5. Прописаны и загружены тестовые данные.

Таким образом, все поставленные задачи на настоящую лабораторную работу были выполнены успешно. Следующим шагом разработки системы является проектирование интерфейса, что будет рассматриваться в следующем отчете.

# Проектирование интерфейса КИС

## Постановка задачи

### Описание назначения КИС и основного функционала согласно ТЗ

Наименование системы: «Модуль корпоративной информационной системы функционирования филиалов онлайн-школы “Exam Masters”».

Модуль корпоративной информационной системы онлайн-школы предназначен для обеспечения пользователям удобного учебного процесса как со стороны учеников, так и со стороны преподавателей и руководства.

Внутри системы пользователи смогут:

1. Добавлять данные в разделы в соответствии с ТЗ;
2. Просматривать статистики;
3. Выгружать отчеты, описанные в ТЗ.

В системе существует разграничение по правам пользователей. Преподавателям доступны следующие функции:

1. Просмотр групп, у которых он ведет занятия;
2. Просмотр каждого отдельного ученика в группе, анализ его статистики;
3. Выставление оценок, корректирование данных в соответствии с ТЗ;
4. Выгрузка отчетов по успеваемости, посещаемости и т.д.;
5. Функции по просмотру статистики, описанной в ТЗ;
6. Функции по выгрузке отчетов из разделов, описанных в ТЗ.

Ученикам доступны следующие функции:

1. Добавление данных по разделам, описанным в ТЗ;
2. Удаление данных по разделам, описанным в ТЗ;
3. Просмотр статистики по персональной успеваемости, посещаемости и т.д.;
4. Выгрузка отчетов по персональной успеваемости.

### Цель работы и задачи, которые нужно выполнить для достижения цели

Цель работы: спроектировать интерфейс для реализации модуля корпоративно-информационной системы онлайн-школы, а также реализовать вывод отчетности и реакции информационной системы на ошибочный ввод данных.

Задачи:

1. Описать средство реализации интерфейса;
2. Реализовать интерфейс приложения согласно ТЗ;
3. Составить шаблоны отчетов;
4. Реализовать механизм реакции на ошибочный ввод данных;

## Интерфейс

### Средства и среда проектирования интерфейса

WinForms в C# – это фреймворк для создания приложений под Windows, который предоставляет широкий набор элементов управления для создания пользовательского интерфейса, основан на событийно-ориентированном программировании, и обеспечивает обширные возможности, подходящие для различных задач. Приложения, разработанные с помощью WinForms, могут легко запускаться на Windows без дополнительной установки компонентов.

Графический интерфейс пользователя представляет собой набор элементов управления таких, как кнопки, поля, списки и т.д., что значительно упрощает разработку пользовательского интерфейса. Помимо того, с помощью интегрированного визуального конструктора в среде разработки *Visual Studio* достаточно просто разместить и настроить требуемые в системе компоненты.

Рассматриваемый подход к созданию пользовательских интерфейсов основан на идее событийно-ориентированного программирования. Иными словами, любое действие пользователя, к примеру, нажатие кнопки или заполнение текстового поля, программой воспринимается как событие. Таким образом, при разработке приложения с пользовательским интерфейсом гораздо проще настроить связь между пользователем и программой, так как можно связать элементы управления и события между собой.

Одним из весомых преимуществ *Windows Forms* является легкость разработки. Дело в том, что визуальный конструктор и панель элементов представляют собой интуитивно-понятный инструмент разработки, что делает ее более быстрой. Плюсом ко всему, разнообразие инструментов позволяет сильно расширить функционал разрабатываемого приложения.

Помимо этого, приложения, написанные с использованием *WinForms,* легко запускаются на операционных системах Windows без необходимости установки сторонних компонентов или фреймворков. Именно поэтому *Windows Forms* является одним из самых актуальных и часто используемых инструментов для разработки настольных приложений для операционных систем Windows.

### Элементы интерфейса

* + - 1. Форма авторизации.

Следующим шагом разработки КИС «Exam Masters» было проектирование элементов интерфейса системы. Для начала была разработана форма авторизации.

Так как в системе (первостепенно в разработанном хранилище данных) предусмотрено разграничение пользователей, было принято решение разграничить форму авторизации для разного типа пользователей.

Таким образом, прежде чем войти в личный кабинет пользователь должен выбрать страницу «Ученик» или «Преподаватель». Помимо этого, если пользователь входит в систему первый раз, следует пройти упрощенную регистрацию, выбрав страницу «Первый раз системе»».

На Рис. 16 представлена форма авторизации для роли пользователя «Ученик»:

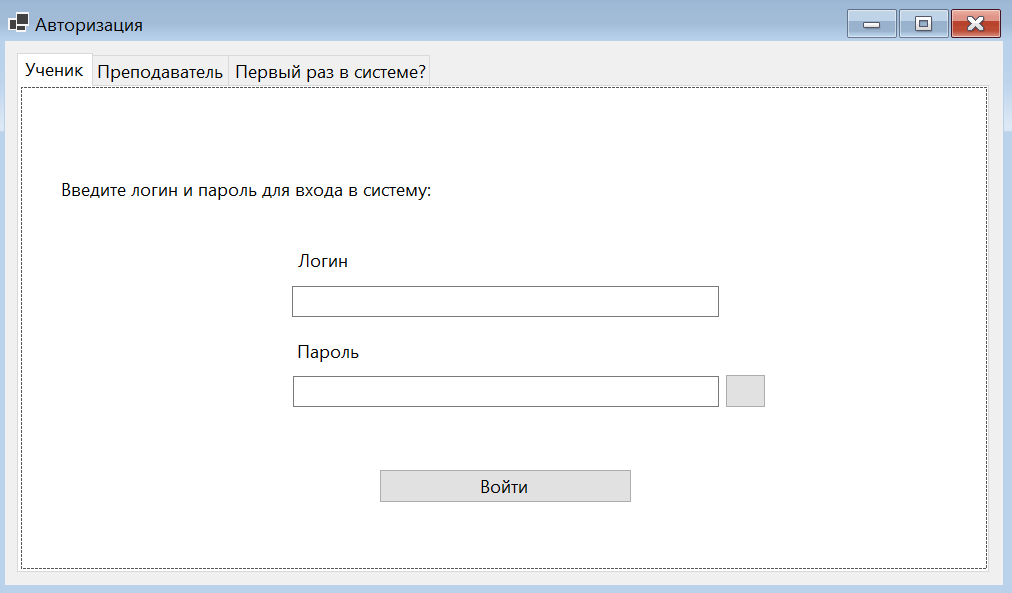


Рис. 16. Форма авторизации пользователя для роли «Ученик»

Далее была спроектирована часть формы авторизации для пользователя роли «Преподаватель». Она представлена ниже на Рис. 17:

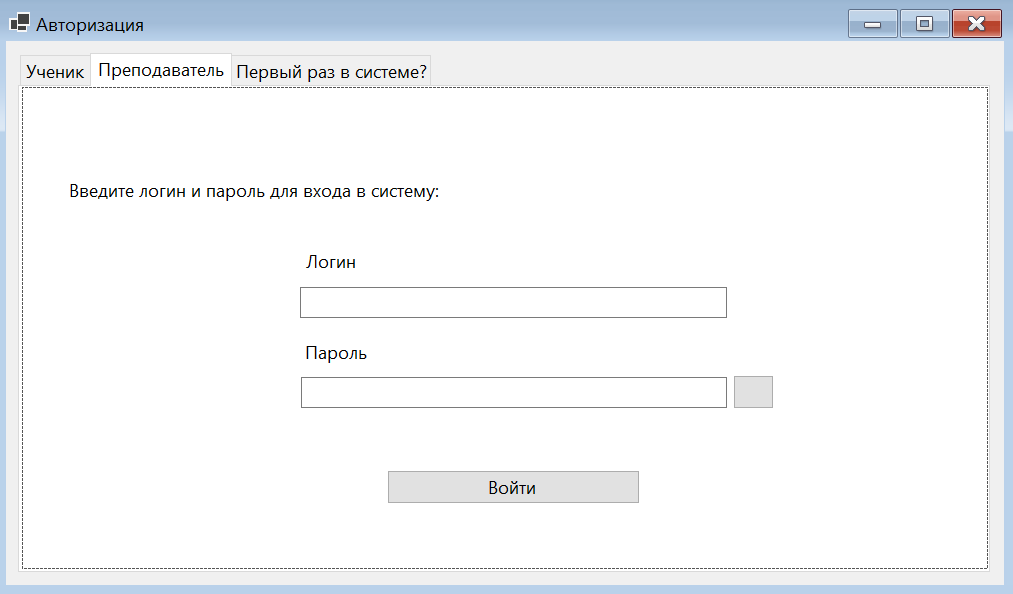


Рис. 17. Форма авторизации пользователя для роли «Преподаватель»

Как упоминалось выше, если пользователь не зарегистрирован в системе, то для входа в личный кабинет он должен пройти упрощенную регистрацию. Как описано в ТЗ, пользователем, который добавляет все данные в систему, является администратор. Администратор вписывает в систему нового пользователя (преподавателя или ученика), таким образом формируя для него свой уникальный логин. Однако, администратор не генерирует пароль для пользователя. Поэтому чтобы авторизоваться, новый пользователь должен придумать для себя пароль, ввести его в странице «Первый раз в системе?», а также его подтвердить повторным вводом.

Данная форма представлена ниже на Рис. 18:

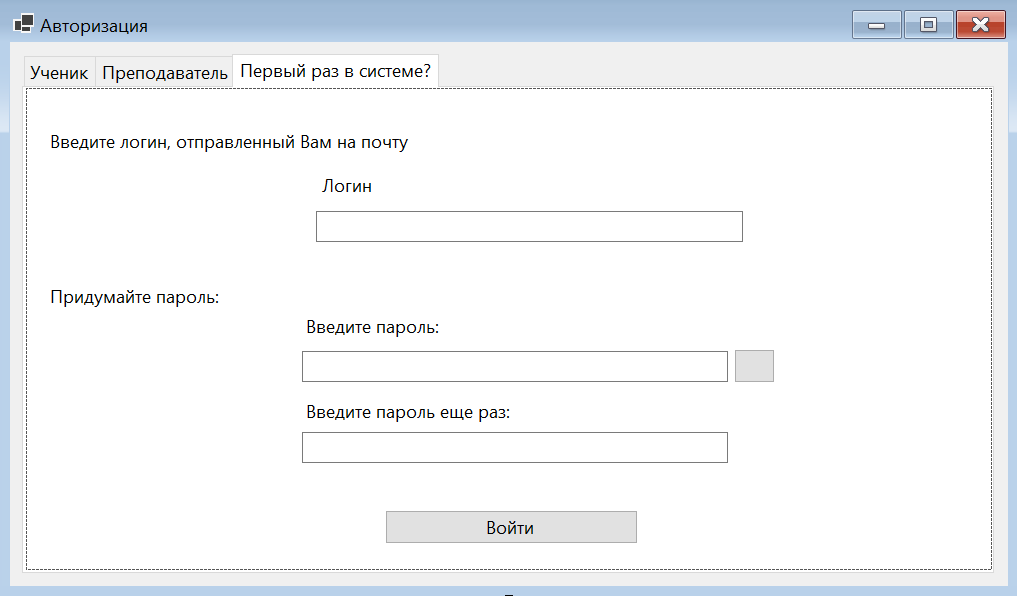


Рис. 18. Форма упрощенной регистрации в системе

Помимо этого, следует отметить, что на всех трех формах присутствует дополнительная кнопка около текстового поля «Ввод пароля», которая может скрыть вводимые данные.

* + - 1. Вид приложения под разными пользователями.

Далее были созданы визуальные элементы взаимодействия для всех типов пользователей. Для начала были созданы формы для пользователя роли «Администратор».

***Администратор***

Для начала была создана начальная страница личного кабинета администратора. Так как администратор работает с основными данными системы, его личный кабинет является достаточно нагруженным в плане функционала. Поэтому было принято решение структурировать функционал и разнести разные блоки работы с данными на отдельные формы. На главной странице администратор может выбрать дальнейшие действия для работы с данными системы. Главная страница личного кабинета администратора представлена ниже на Рис. 19:

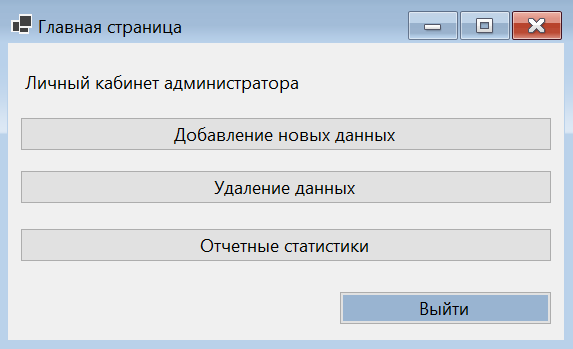


Рис. 19. Главная страница личного кабинета администратора

1. *«Добавление новых данных»:* нажимая на эту кнопку, пользователь переходит на форму, на которой он может добавлять новые данные в систему.
2. *«Удаление данных»:* нажимая на эту кнопку, пользователь переходит на форму, на которой он может удалять определенные данные.
3. *«Отчетные статистики»:* нажимая на эту кнопку, пользователь переходит на форму, на которой он может формировать отчетные статистики по преподавателям, ученикам и группам.

Далее была спроектирована форма для добавления новых данных. Для большей структурированности данные разграничены по двум разделам: «Пользователи» и «Курсы». В разделе «Пользователи» администратор может добавлять новых преподавателей, учеников и группы. Этот раздел представлен ниже на Рис. 20:

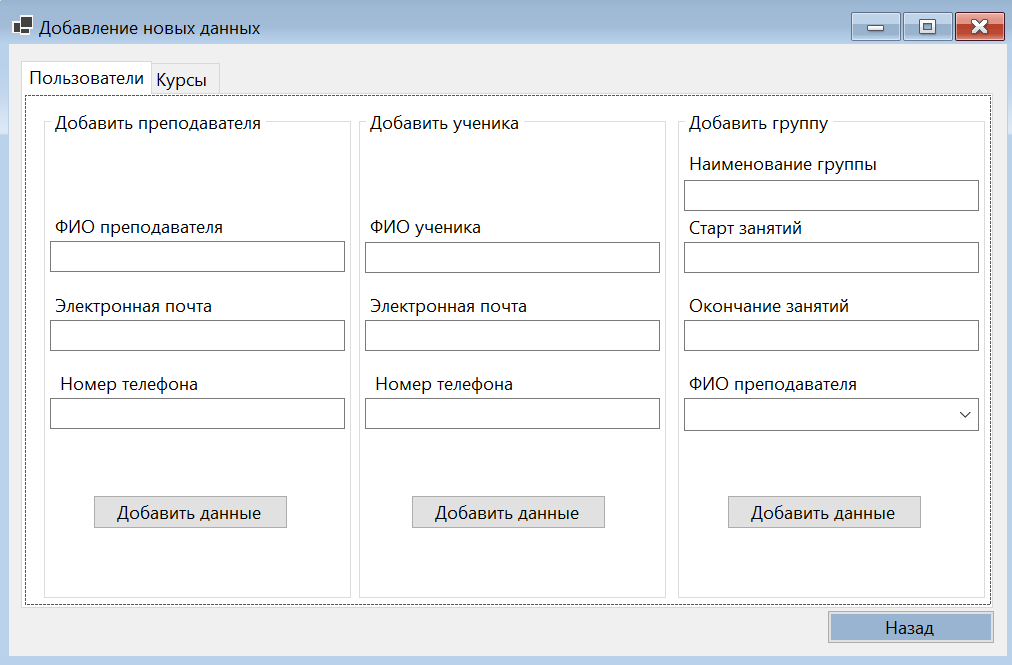


Рис. 20. Добавление новых данных о преподавателях, учениках и группах

В разделе «Пользователи» администратор может добавлять новых преподавателей, учеников и группы. Этот раздел представлен ниже на Рис. 21:

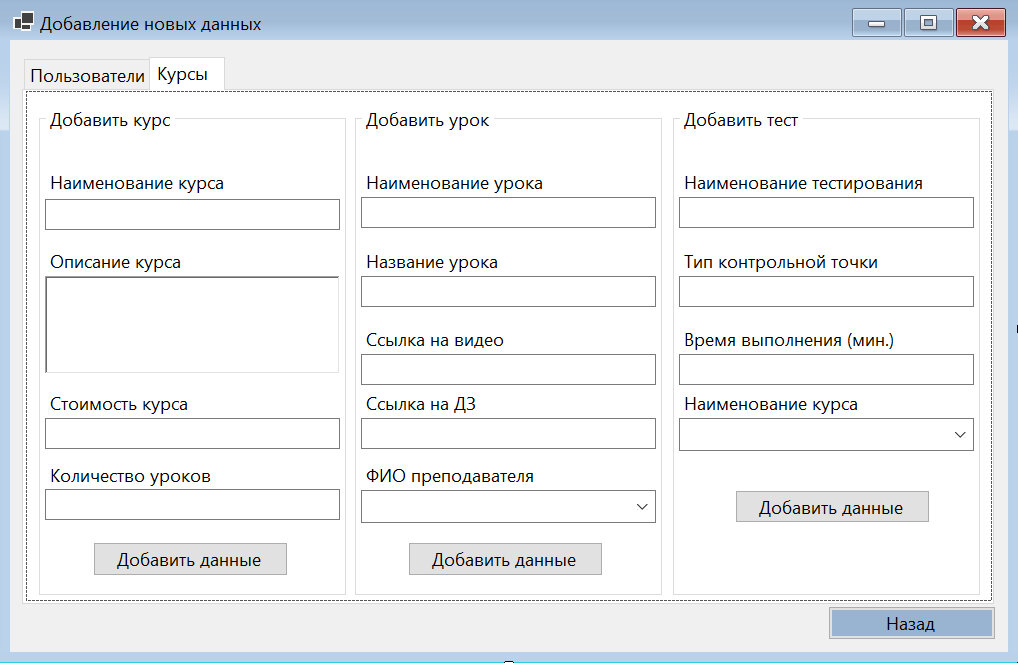


Рис. 21. Добавление новых данных о курсах, занятиях и тестах

Далее была реализована форма «Удаление данных». В данном случае для того, чтобы удалить какие-либо данные, пользователь должен выбрать из выпадающего списка те данные, которые можно удалить из системы. Эта форма представлена на Рис. 22:

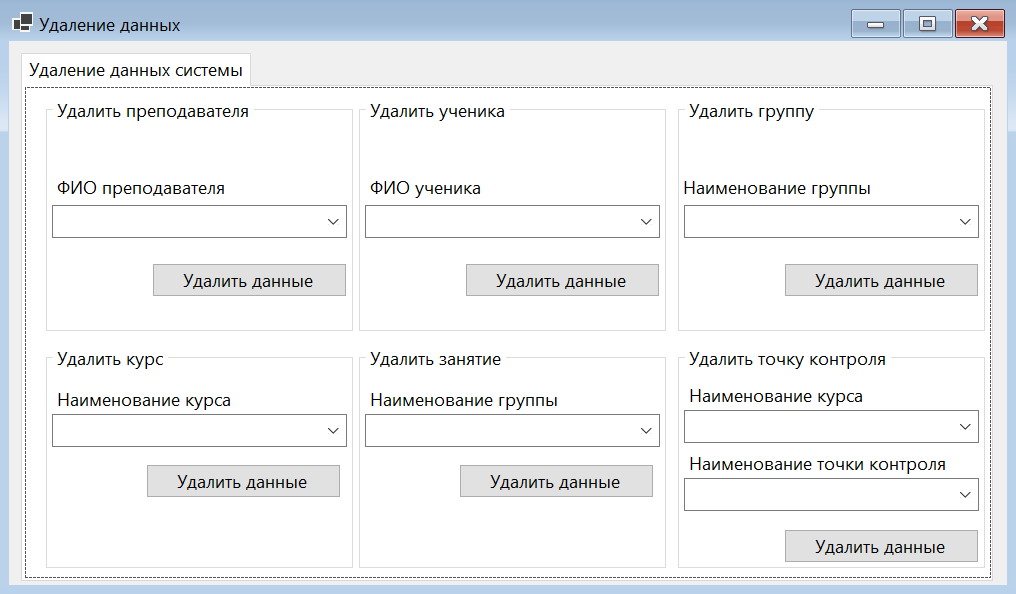


Рис. 22. Удаление данных

Последним шагом была реализована форма отчетных статистик по показателям онлайн-школы. В данном случае, отчетность должна формироваться по нескольким пунктам: общие статистики преподавателей, учеников и групп. Для удобства представления данных данные показатели были вынесены в разные разделы.

В отчетных статистиках по преподавателям выгружается ФИО преподавателя, его номер телефона и почта, а также средние показатели посещаемости и ДЗ по всем группам. Кроме того, можно сортировать вывод данных по процентам посещаемости и ДЗ. В данном случае, можно вывести статистику и экспортировать ее в файл Excel. Нажатие на кнопку «На главную» возвращает на первую страницу личного кабинета администратора. Реализация этого раздела представлена ниже на Рис. 23:

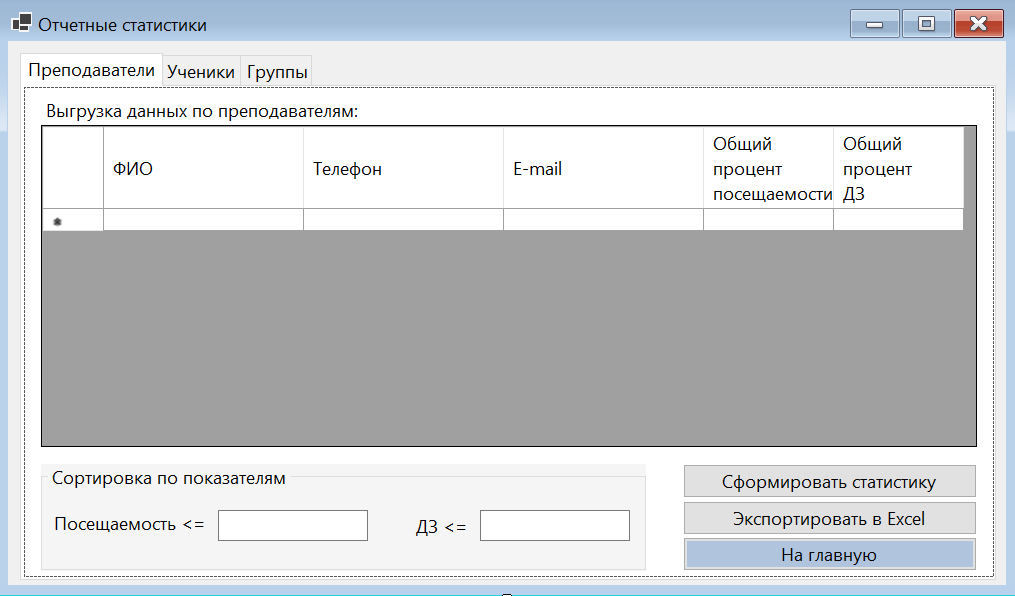


Рис. 23. Раздел отчетных статистик по преподавателям

В разделе «Отчетные статистики по ученикам» можно сформировать отчет по ученикам, сортируя учеников по процентам посещаемости и ДЗ или по группам и преподавателям. Этот раздел представлен ниже на Рис. 24:

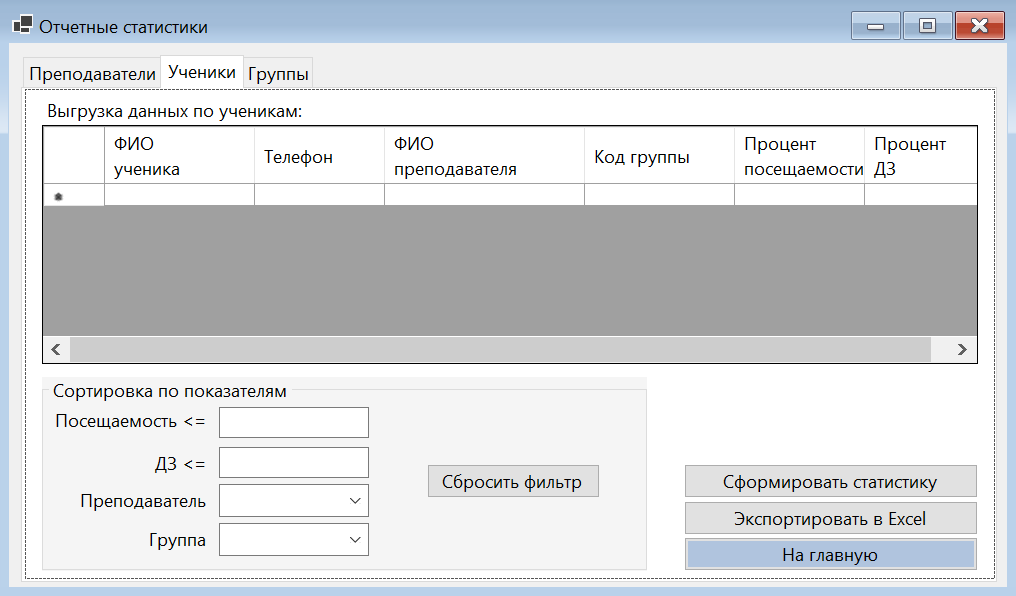


Рис. 24. Раздел отчетных статистик по ученикам

В разделе «Отчетные статистики по группам» реализован почти идентичный статистикам по ученикам функционал. Единственное отличие заключается в том, что в данном случае проценты ДЗ и посещаемости высчитывается средним арифметическим между учениками группы. Эта форма представлена ниже на Рис. 25:

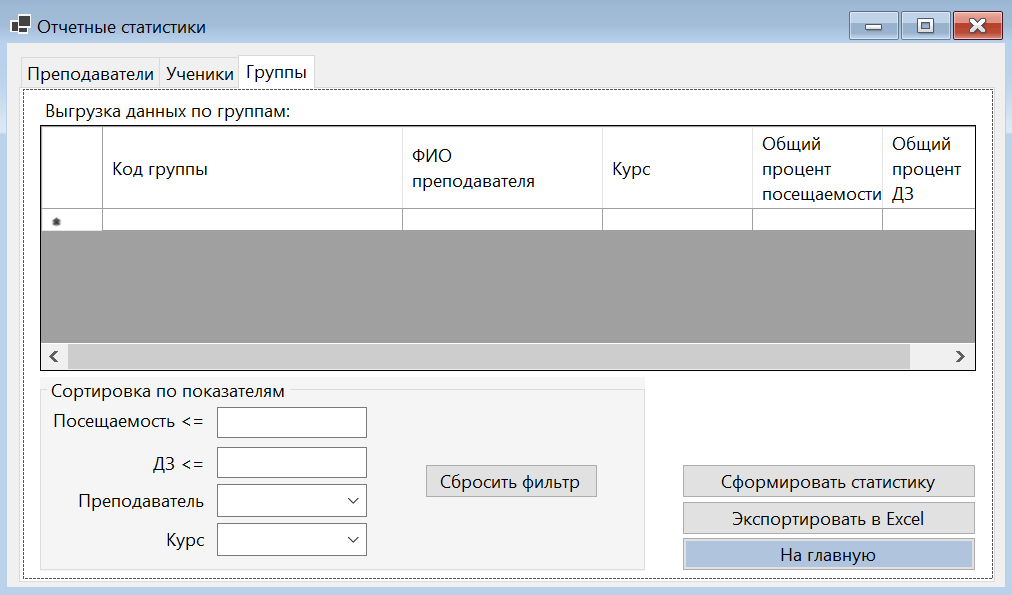


Рис. 25. Раздел отчетных статистик по группам

***Ученик***

При проектировании визуальной составляющей личного кабинета ученика первым шагом была создана начальная страница его личного кабинета. Личный кабинет ученика разделен на два модуля: просмотр расписания, в котором он может просматривать расписание, а также выполнять ДЗ, и просмотр долгов по учебе. Форма личного кабинета ученика представлена на Рис. 26:

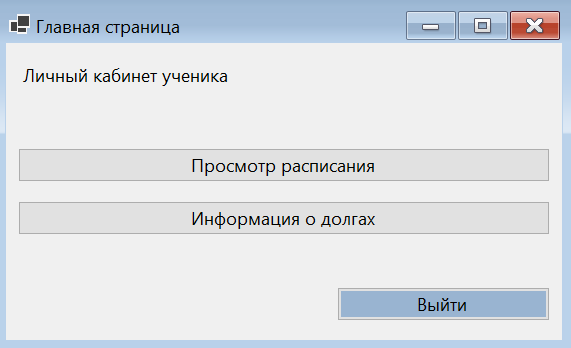


Рис. 26. Главная страница личного кабинета ученика

Далее была реализована форма «Расписание занятий». В этой форме ученик может просматривать свое расписание с сортировкой по периоду. Помимо этого, пользователь может выбирать сортировку расписания по доступным ему курсам. Помимо этого, пользователь при необходимости может сбросить сортировку, а также экспортировать в Excel. Форма вывода расписания занятий представлена на Рис. 27:

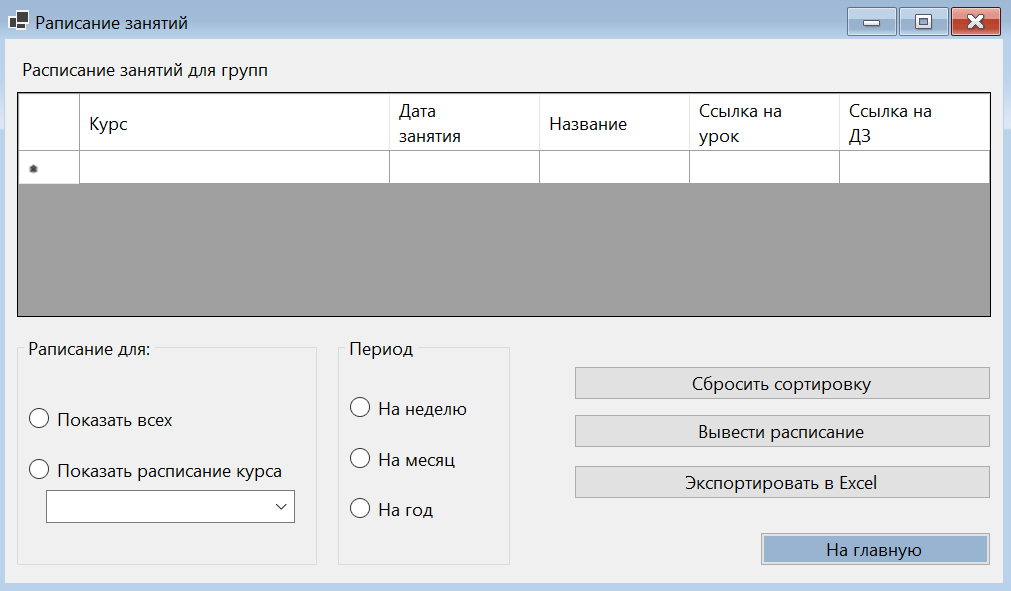


Рис. 27. Форма вывода расписания занятий для ученика

Далее была реализована форма, на которой ученик может просматривать свои долги по учебе. В данном случае выводятся данные о занятиях, которые он либо пропустил, либо не выполнил ДЗ. Форма представлена ниже на Рис. 28:

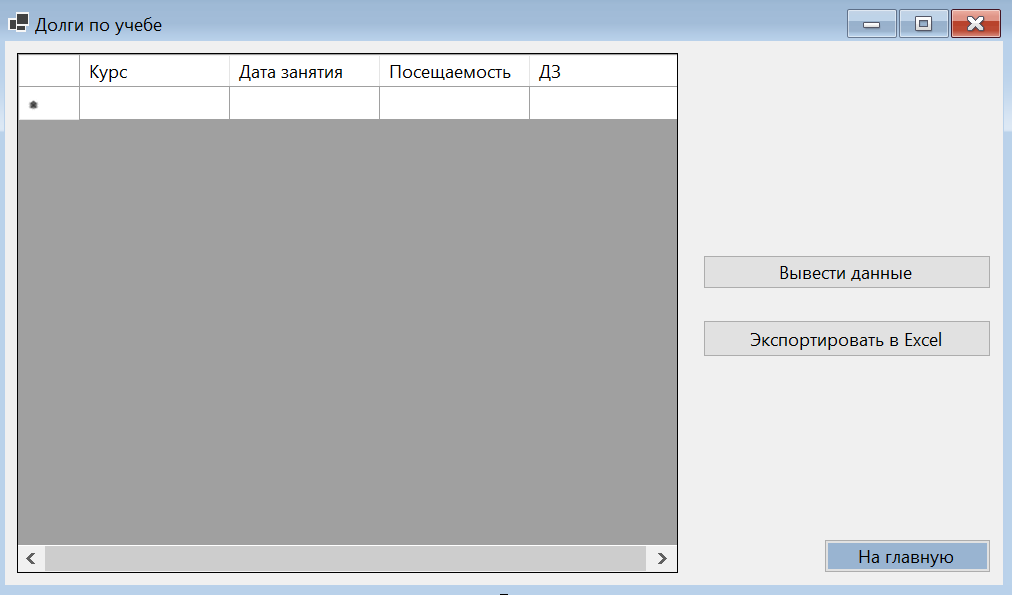


Рис. 28. Форма вывода долгов по учебе

***Преподаватель***

Последняя роль, предусмотренная в системе, – преподаватель. В данном случае, было принято решение так же, как и для других ролей реализовать главную страницу личного кабинета. В своем личном кабинете преподаватель может просматривать свое расписание занятий по разным группам, выводить информацию об учениках, а также формировать сводные отчеты. Форма личного кабинета преподавателя представлена ниже на Рис. 29:

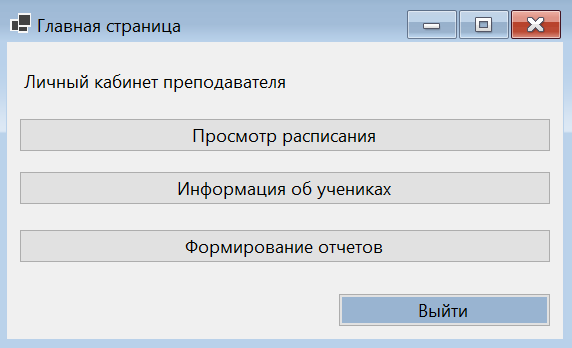


Рис. 29. Личный кабинет преподавателя

Далее была реализована форма вывода расписания. В данном случае, преподаватель может сортировать расписание по группам и по периодам. Помимо этого, существует возможность экспорта расписания в Excel. Форма вывода расписания представлена ниже на Рис. 30:

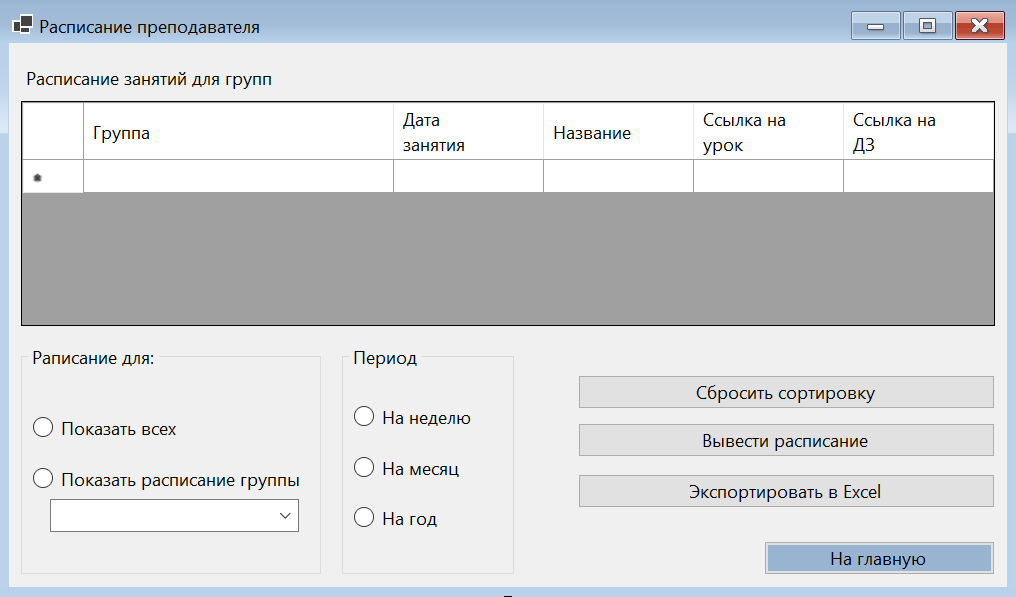


Рис. 30. Форма вывода расписания

Далее была создана форма вывода контактной информации по ученикам. В данном случае, преподаватель может вывести информацию по группе, сортируя учеников по проценту посещаемости или проценту ДЗ. Эти данные также можно экспортировать в Excel при необходимости. Данная форма представлена ниже на Рис. 31:

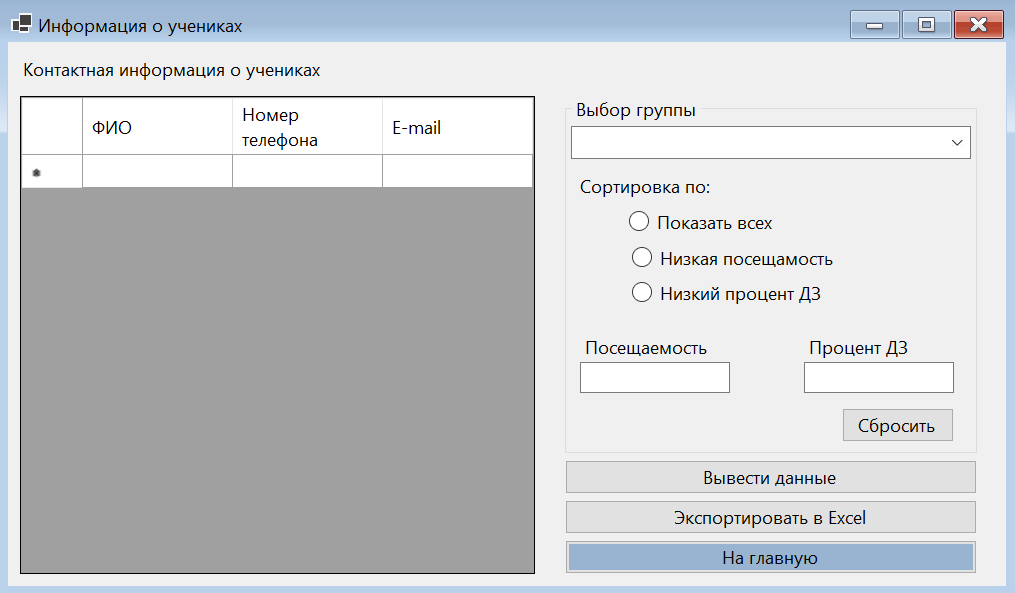


Рис. 31. Форма вывода контактной информации о учениках

Помимо этого, преподаватель также может формировать отчетные статистики по группам и ученикам. Для удобства отдельные функции вывода отчетности разграничены по разным разделам. В первом разделе преподаватель может сформировать статистику по ученикам на основе их средних показателей: ДЗ и посещаемости. Данная форма представлена ниже на Рис. 32:

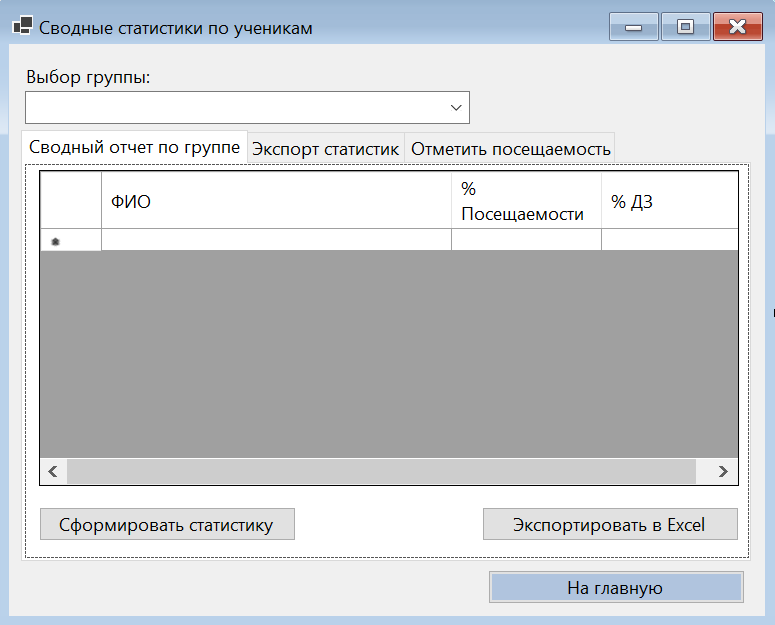


Рис. 32. Формирование сводного отчета по группе

В форме сводных статистик по ученикам также можно экспортировать данные по посещаемости и проценту ДЗ по ученикам. Для этого следует выбрать группу, по которой планируется создать статистику и нажать кнопку «Экспорт в Excel».

На Рис. 33 представлена реализация раздела «Экспорт статистик»:

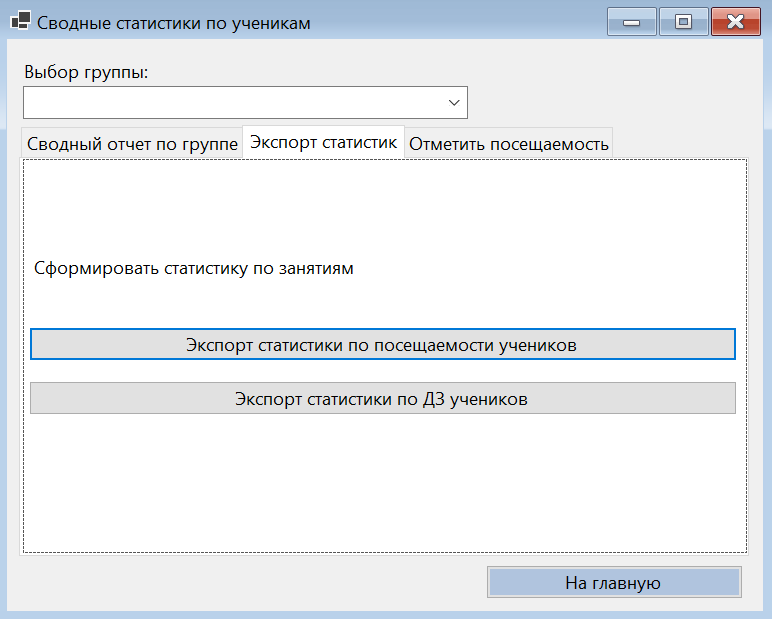


Рис. 33. Раздел «Экспорт статистик»

Помимо этого, преподаватель может в своем личном кабинете проставлять посещаемость, поэтому для этого был создан раздел «Отметить посещаемость». Реализация этого раздела представлена ниже на Рис. 34:

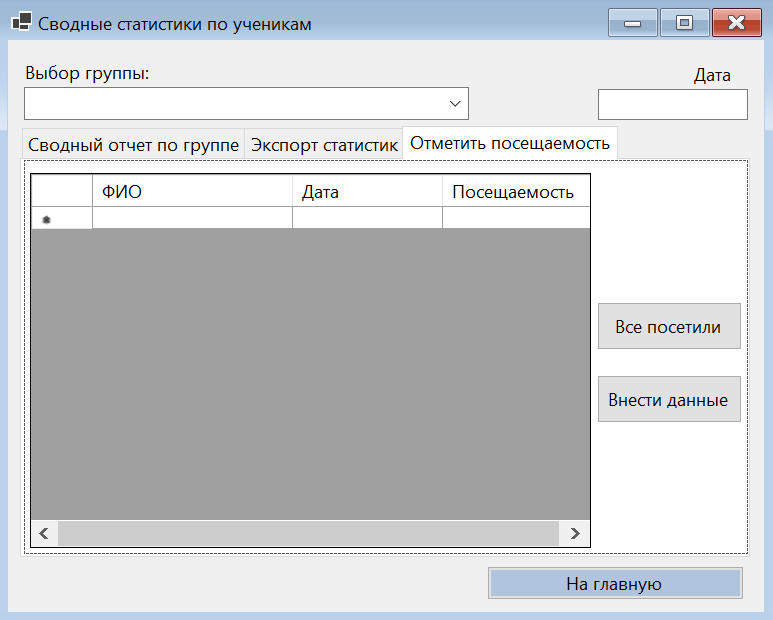


Рис. 34. Раздел «Отметить посещаемость»

* + - 1. Редактирование информационной составляющей БД

Редактирование данных реализовано согласно ТЗ. В зависимости от типа пользователя. Ниже описано варианты редактирования данных информационной системы с разграничением типов пользователей:

1. ***Администратор.*** Пользователь может добавлять и удалять данные в разделах «Добавление данных» и «Удаление данных». *Добавление данных* можно осуществлять в следующих разделах: добавление пользователей (преподаватели, ученики, курсы) и добавление курсов (курсы, занятия, тесты). *Удаление данных* доступно в разделе «Удаление данных». Здесь можно удалять следующие строки таблиц БД: преподаватели, ученики, группы, курсы, занятия, тесты.
2. ***Преподаватель.*** Пользователь может редактировать данные о посещаемости учеников в разделе «Сводные статистики по ученикам».
3. ***Ученик.*** Пользователь может редактировать данные о показателях ДЗ в разделах «Просмотр расписания» и «Просмотр долгов».

### 3.3.3. Отчеты

***Администратор.*** Администратор имеет возможность экспортировать статистики по преподавателям, ученикам и группам. Примеры отчетов в Excel представлены на Рис. 35 – Рис. 37:

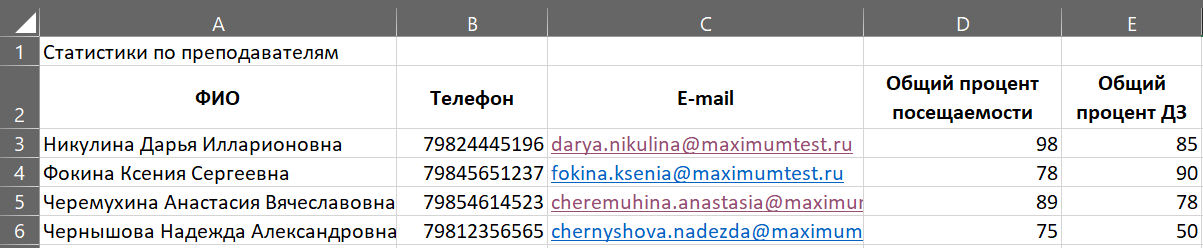


Рис. 35. Экспорт статистики по преподавателям

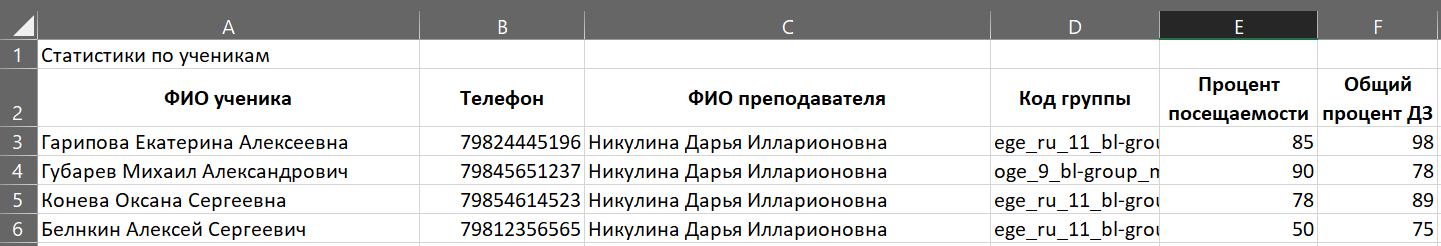


Рис. 36. Экспорт статистики по ученикам

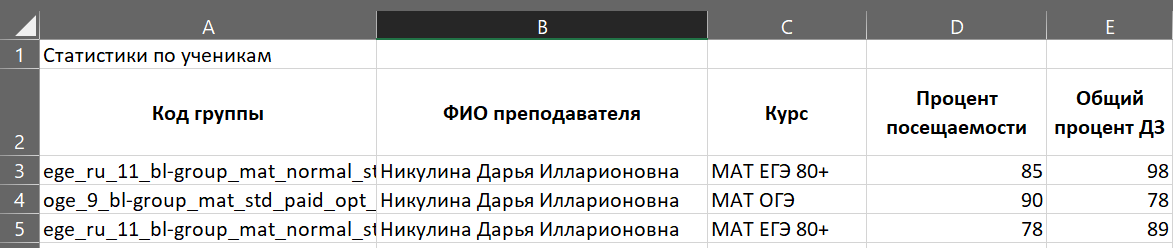


Рис. 37. Экспорт статистик по группам

***Ученик.*** Ученик может выгружать статистики по своим долгам, а также расписание занятий. Примеры отчетов в Excel представлены на Рис. 38, Рис. 39:

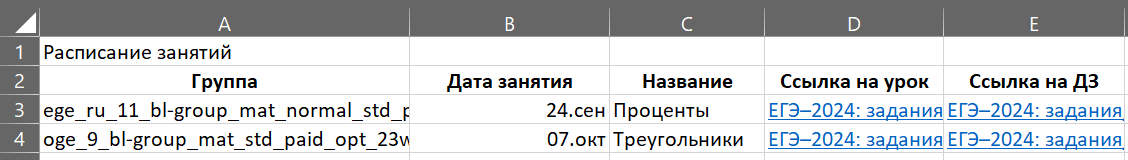


Рис. 38. Экспорт расписания занятий

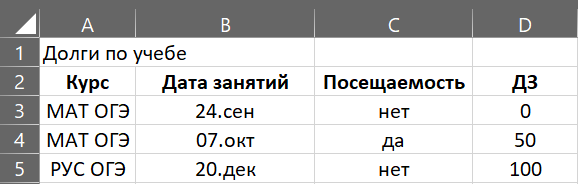


Рис. 39. Экспорт долгов по занятиям

***Преподаватель.*** Преподаватель может выгружать статистики по ученикам и группам, в которых он преподает. Примеры отчетов в Excel представлены на Рис. 40 – Рис. 44:

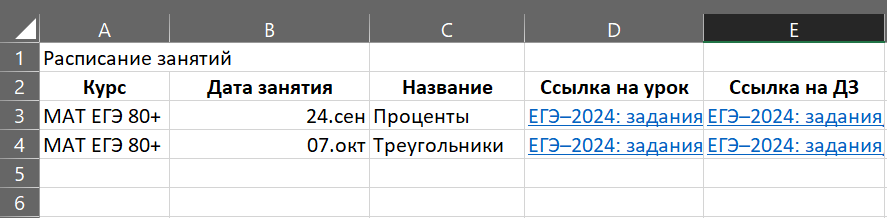


Рис. 40. Экспорт расписания занятий

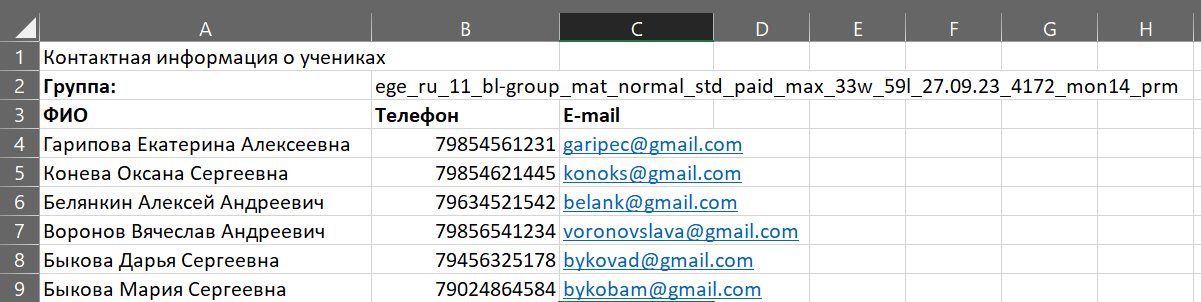


Рис. 41. Экспорт контактной информации об учениках

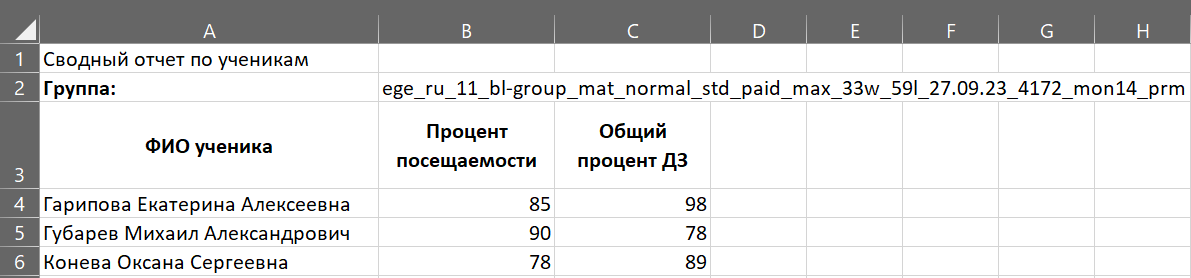


Рис. 42. Экспорт сводного отчета по ученикам

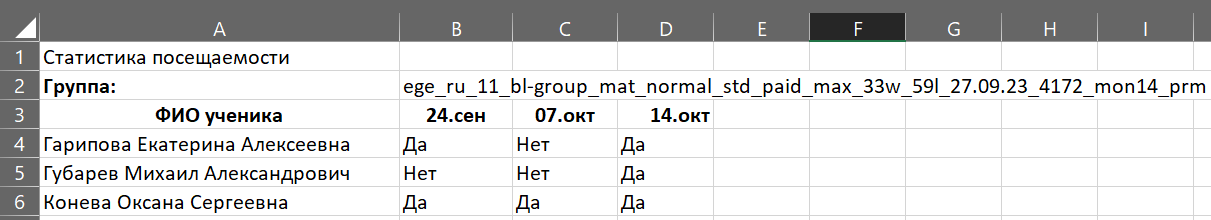


Рис. 43. Экспорт отчетности по посещаемости занятий

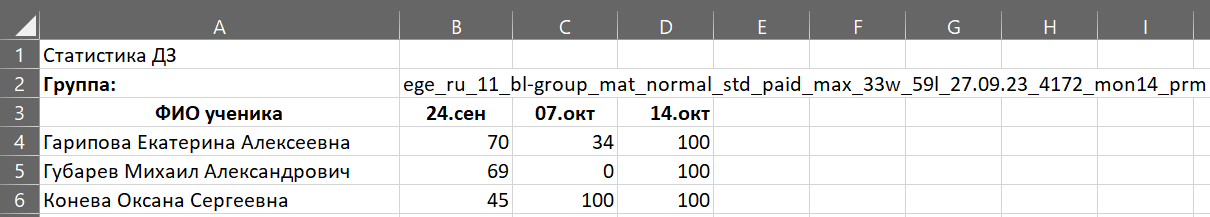


Рис. 44. Экспорт отчетности по процентам ДЗ

### 3.3.4. Реакция ИС на ошибочный ввод данных

На форме «Авторизация» необходимо, чтобы пользователь ввел свои логин и пароль, однако если он этого не сделает, следует выводить сообщение о ошибочном вводе данных. В данном случае, открывается диалоговое окно, которое оповещает пользователя, что следует заполнить необходимые поля. Диалоговое окно ошибки ввода данных представлено ниже на Рис. 45:

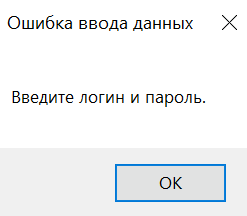


Рис. 45. Реакция системы на ошибку ввода данных в поле «Авторизация»

Далее во всех полях для ввода данных в формах для преподавателя и администратора в случае неполного ввода данных выводится диалоговое окно, показанное ниже на Рис. 46:

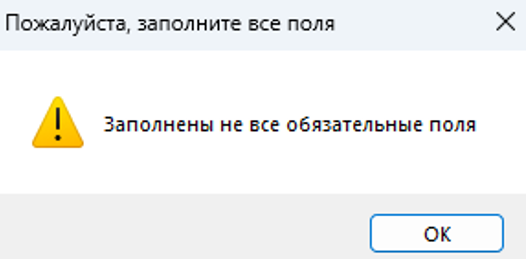


Рис. 46. Реакция системы на добавление и удаление данных с незаполненными полями

## Выводы

Входе настоящей работы по проектированию КИС *«Exam Masters»* был разработан ее интерфейс. Помимо этого, для тестирования интерфейса были составлены тестовые данные, которые помогают провести тестирование готового интерфейса.

Согласно ТЗ, следующим этапом разработки является диаграмма классов, которая будет описана в следующем отчете.

# Разработка структур классов

## Постановка задачи

### Описание КИС и основного функционала

Наименование системы: «Модуль корпоративной информационной системы функционирования филиалов онлайн-школы “Exam Masters”».

Модуль корпоративной информационной системы онлайн-школы предназначен для обеспечения пользователям удобного учебного процесса как со стороны учеников, так и со стороны преподавателей и руководства.

Внутри системы пользователи смогут:

1. Добавлять данные в разделы в соответствии с ТЗ;
2. Просматривать статистики;
3. Выгружать отчеты, описанные в ТЗ.

В системе существует разграничение по правам пользователей. Преподавателям доступны следующие функции:

1. Просмотр групп, у которых он ведет занятия;
2. Просмотр каждого отдельного ученика в группе, анализ его статистики;
3. Выставление оценок, корректирование данных в соответствии с ТЗ;
4. Выгрузка отчетов по успеваемости, посещаемости и т.д.;
5. Функции по просмотру статистики, описанной в ТЗ;
6. Функции по выгрузке отчетов из разделов, описанных в ТЗ.

Ученикам доступны следующие функции:

1. Добавление данных по разделам, описанным в ТЗ;
2. Удаление данных по разделам, описанным в ТЗ;
3. Просмотр статистики по персональной успеваемости, посещаемости и т.д.;
4. Выгрузка отчетов по персональной успеваемости.

### Цель работы и задачи

**Цель работы:** спроектировать диаграмму классов, помогающей реализовать модуль корпоративно-информационной системы онлайн-школы *«Exam Masters».*

**Задачи:**

* + - 1. Выбрать средство реализации диаграммы классов;
      2. Согласно ТЗ определить необходимые в системе классы;
      3. Определить связи между классами и компонентами;
      4. Разработать диаграмму классов;
      5. Составить описание диаграммы классов.

## Диаграмма классов

### Средство разработки

Для разработки диаграммы классов было принято использовать бесплатный онлайн-сервис *«Draw.io»*. Данный онлайн-сервис предназначен для создания диаграмм, блок-схем, схем бизнес-процессов, *UML*, схем *ER*-моделей и т.д.

Достоинство сервиса заключается в его обширном функционале. Иными словами, пользователь может работать с большим набором форм, стрелок и других элементов, необходимым для создания диаграмм. Помимо того, в сервисе также предусмотрены функции экспорта и импорта изображений в различных форматах: *PDF*, *XML*, *JPG*, *PNG* и т.д.

Общий алгоритм работы с сервисом представлен ниже:

Создание новой диаграммы. На данном этапе можно выбрать как пустую диаграмму, так и определенный тип диаграммы, заложенный в систему.

Добавление элементов. После создания диаграммы можно добавлять элементы на полотно, перетаскивая их из панели инструментов.

Сохранение и экспорт. После завершения проектирования диаграммы можно сохранить схему либо в облачном хранилище, либо на компьютере в любом удобном формате.

Совместная работа. Сервис позволяет совместно проектировать схемы и диаграммы. Пользователь может пригласить других пользователей к проекту для просмотра и редактирования.

## 

### Описание диаграммы классов

На Рис. 47 ниже показана диаграмма классов, спроектированная в сервисе *«Draw.io»*:

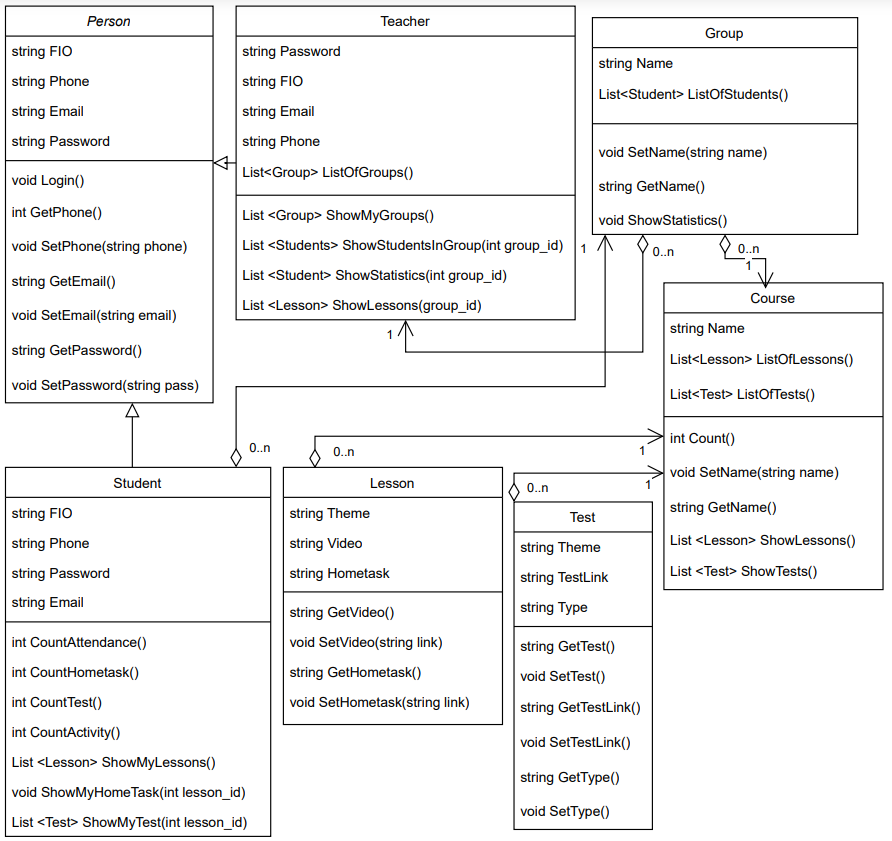


Рис. 47. Диаграмма классов для реализации КИС «Exam Masters»

Таким образом, в системе будут реализованы 7 классов. Далее следует описать классы и их компоненты, которые будут предусмотрены в системе при дальнейшей разработке.

**Класс «*Person».*** Класс используется в самом начале в окне авторизации. Ниже в Табл. 10 представлены поля класса «Person»:

Табл. 10. Поля класса «Person»

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип данных** | **Описание** |
| FIO | String | ФИО пользователя |
| Phone | String | Номер телефона пользователя |
| Email | String | Электронная почта пользователя (в системе также подразумевается, как логин при входе в систему) |
| Password | String | Пароль от учетной записи пользователя |

Далее были прописаны методы класса «Person». Более подробное описание представлено ниже на Табл. 11:

Табл. 11. Методы класса «Person»

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Тип данных** | **Описание** |
| Login | Void | Метод предназначен для авторизации пользователя в системе. |
| GetPhone | String | Метод возвращает исходное значение номера телефона пользователя. |
| SetPhone | String | Метод **на вход** принимает строку.  Метод присваивает новое значение полю Phone класса Person. |
| GetEmail | String | Метод возвращает исходное значение электронной почты пользователя. |
| SetEmail | String | Метод **на вход** принимает строку.  Метод присваивает новое значение полю Email класса Person. |
| GetPassword | String | Метод возвращает исходное значение пароля пользователя для входа в систему. |
| SetPassword | String | Метод **на вход** принимает строку.  Метод присваивает новое значение полю Password класса Person. |

**Класс «*Teacher».*** Класс наследует поля и методы класса «Person». Ниже в Табл. 12 представлены поля класса «Teacher»:

Табл. 12. Поля класса «Teacher»

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип данных** | **Описание** |
| FIO | String | ФИО преподавателя |
| Phone | String | Номер телефона преподавателя |
| Email | String | Электронная почта преподавателя |
| Password | String | Пароль от учетной записи преподавателя |
| ListOfGroups | List <Group> | Список групп, которые ведет преподаватель.  *В данном случае, используется тип данных «Group», который является отдельным классом в системе. Он будет описан ниже.* |

Так как класс «Teacher» является дочерним от класса «Person», он наследует и методы материнского класса. Помимо этого, дочерний класс выполняет специфические методы, которые описаны ниже на Табл. 13:

Табл. 13. Методы класса «Teacher»

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Тип данных** | **Описание** |
| ShowMyGroups | List <Group> | Метод возвращает список групп, в которых преподаватель ведет занятия. |
| ShowStudentsInGroup | List <Student> | **На** **вход** метод получает идентификатор группы.  Возвращает список студентов, которые принадлежат группе, в которой преподаватель ведет занятия. |
| ShowStatistics | List <Group> | **На** **вход** метод получает идентификатор группы.  Метод выводит статистику по ученикам группы, в которой преподаватель ведет занятия. |
| ShowLessons | List <Lesson> | **На** **вход** метод получает идентификатор группы.  Метод выводит занятия группы, в которой преподаватель ведет занятия. |

**Класс «*Student».*** Класс наследует поля и методы класса «Person». Ниже в Табл. 14 представлены поля класса «Student»:

Табл. 14. Поля класса «Student»

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип данных** | **Описание** |
| FIO | String | ФИО ученика |
| Phone | String | Номер телефона ученика |
| Email | String | Электронная почта ученика |
| Password | String | Пароль от учетной записи ученика |

Так как класс «Student» является дочерним от класса «Person», он наследует и методы материнского класса. Помимо этого, дочерний класс выполняет специфические методы, которые описаны ниже на Табл. 15:

Табл. 15. Методы класса «Student»

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Тип данных** | **Описание** |
| CountAttendance | Int | Метод возвращает процент посещаемости занятий по предмету. |
| CountHometask | Int | Метод возвращает процент домашнего задания по предмету. |
| CountTest | Int | Метод возвращает среднюю оценку за контрольные точки по предмету. |
| CountActivity | Int | Метод возвращает среднюю оценку за активность на занятиях по предмету. |
| ShowMyLessons | List <Lesson> | Метод возвращает список уроков на курсе. |
| ShowMyHomeTask | Void | Метод возвращает список домашнего занятия на курсе. |
| ShowMyTest | List <Test> | Метод возвращает список тестов на курсе. |

**Класс «*Group».*** Класс является объектом-агрегатом класса «Teacher». Таким образом, была реализована связь «один-ко-многим». В свою очередь, между классом «Group» и «Student» также реализована эта связь, так как в группе может быть несколько учеников. Ниже в Табл. 16 представлены поля класса «Group»:

Табл. 16. Поля класса «Group»

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип данных** | **Описание** |
| Name | String | Наименование группы |
| ListOfStudents | List <Student> | Список учеников, которые обучаются в одной группе |

Далее были прописаны методы, которые присущи только классу «Group». Эти методы описаны ниже на Табл. 17:

Табл. 17. Методы класса «Group»

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Тип данных** | **Описание** |
| SetName | Void | Метод **на вход** принимает строку и присваивает новое значение полю «Name». |
| GetName | String | Метод возвращает значение поля «Name». |
| ShowStatistics | String | Метод возвращает статистику по группе. |

**Класс «*Course».*** Между этим классом и классом «Croup» также реализована связь «один-ко-многим», так как несколько разных групп могут заниматься по программе одного курса. Ниже в Табл. 18 представлены поля класса «Course»:

Табл. 18. Поля класса «Course»

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип данных** | **Описание** |
| Name | String | Наименование курса |
| ListOfLessons | List <Lesson> | Список уроков на курсе |
| ListOfTests | List <Test> | Список тестов на курсе |

Далее были прописаны методы, которые присущи только классу «Course». Эти методы описаны ниже на Табл. 19:

Табл. 19. Методы класса «Course»

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Тип данных** | **Описание** |
| Count | Int | Метод подсчитывает количество уроков на курсе. |
| SetName | Void | Метод присваивает новое значение наименования курса. |
| GetName | String | Метод возвращает наименование курса. |
| ShowLessons | List <Lesson> | Метод возвращает список уроков курса. |
| ShowTests | List <Test> | Метод возвращает список тестов курса. |

**Класс «*Lesson».*** Класс является объектом-агрегатом класса «Group». Таким образом, была реализована связь «один-ко-многим», так как в одном курсе может быть много занятий. Ниже в Табл. 20 представлены поля класса «Lesson»:

Табл. 20. Поля класса «Lesson»

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип данных** | **Описание** |
| Theme | String | Тема занятия |
| Video | String | Ссылка для подключения занятию |
| Hometask | String | Ссылка домашнего задания к занятию |

Далее были прописаны методы, принадлежащие классу «Lesson». Эти методы описаны ниже на Табл. 21:

Табл. 21. Методы класса «Lesson»

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Тип данных** | **Описание** |
| SetVideo | Void | Метод принимает **на вход** ссылку и присваивает новое значение ссылки для подключения к занятию. |
| GetVideo | String | Метод возвращает ссылку для подключения к занятию. |
| SetHometask | Void | Метод принимает **на вход** ссылку и присваивает новое значение ссылки для выполнения домашнего задания к занятию. |
| GetHometask | String | Метод возвращает ссылку на домашнее задание. |

**Класс «*Test».*** Класс является объектом-агрегатом класса «Group». Таким образом, была реализована связь «один-ко-многим», так как в одном курсе может быть несколько тестов. Ниже в Табл. 22 представлены поля класса «Test»:

Табл. 22. Поля класса «Test»

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип данных** | **Описание** |
| Theme | String | Тема контрольной точки |
| TestLink | String | Ссылка для прохождения контрольной точки |
| Type | String | Тип контрольной точки |

Далее были прописаны методы, принадлежащие классу «Test». Эти методы описаны ниже на Табл. 23:

Табл. 23. Методы класса «Test»

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Тип данных** | **Описание** |
| GetTheme | string | Метод возвращает название контрольной точки. |
| SetTheme | Void | Метод принимает **на вход** строку и присваивает новое значение темы теста. |
| GetTestLink | string | Метод возвращает ссылку для прохождения контрольной точки. |
| SetTestLink | Void | Метод принимает **на вход** строку и присваивает новое значение ссылки для прохождения контрольной точки. |
| GetType | string | Метод возвращает тип контрольной точки. |
| SetType | Void | Метод принимает **на вход** строку и присваивает новое значение типа контрольной точки. |

## Выводы

В ходе настоящей лабораторной работы была спроектирована диаграмма классов для корпоративной информационной системы онлайн-школы *«Exam Masters».* Были реализованы необходимы в системе классы, а также продуманы из поля и методы. Помимо этого, были реализованы наиболее рациональные связи между классами согласно ТЗ и хранилищу данных.

В результате выполнения лабораторной работы была спроектирована диаграмма классов, которая полностью покрывает все требования ТЗ и обеспечивает наиболее удобную дельнейшую реализацию системы. В структуре классов были учтены основные составляющие, задействованные также в хранилище данных. Помимо того, методы классов были подобраны таким образом, чтобы можно было реализовать все функции, указанные в ТЗ.

Согласно ТЗ следующим этапом разработки информационной системы является реализация программного кода.

# Программная реализация КИС

## Постановка задачи

### Описание КИС и основного функционала

Наименование системы: «Модуль корпоративной информационной системы функционирования филиалов онлайн-школы “Exam Masters”».

Модуль корпоративной информационной системы онлайн-школы предназначен для обеспечения пользователям удобного учебного процесса как со стороны учеников, так и со стороны преподавателей и руководства.

Внутри системы пользователи смогут:

1. Добавлять данные в разделы в соответствии с ТЗ;
2. Просматривать статистики;
3. Выгружать отчеты, описанные в ТЗ.

В системе существует разграничение по правам пользователей. Преподавателям доступны следующие функции:

1. Просмотр групп, у которых он ведет занятия;
2. Просмотр каждого отдельного ученика в группе, анализ его статистики;
3. Выставление оценок, корректирование данных в соответствии с ТЗ;
4. Выгрузка отчетов по успеваемости, посещаемости и т.д.;
5. Функции по просмотру статистики, описанной в ТЗ;
6. Функции по выгрузке отчетов из разделов, описанных в ТЗ.

Ученикам доступны следующие функции:

1. Добавление данных по разделам, описанным в ТЗ;
2. Удаление данных по разделам, описанным в ТЗ;
3. Просмотр статистики по персональной успеваемости, посещаемости и т.д.;
4. Выгрузка отчетов по персональной успеваемости.

### Цель работы и задачи

**Цель работы:** спроектировать диаграмму классов, помогающей реализовать модуль корпоративно-информационной системы онлайн-школы *«Exam Masters».*

**Задачи:**

* + - 1. Выбрать средство реализации диаграммы классов;
      2. Согласно ТЗ определить необходимые в системе классы;
      3. Определить связи между классами и компонентами;
      4. Разработать диаграмму классов;
      5. Составить описание диаграммы классов.

## Реализация

### Средства и среда разработки

Для дальнейшей реализации корпоративной информационной системы *«Exam Masters»* были использованы инструменты языка *C#* и инструменты *Windows Forms*. В качестве среды разработки была выбрана *Visual Studio.*

**C#** – это объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он используется для создания различных приложений, игр, веб-сервисов и многого другого. Ниже представлены основные достоинства языка:

1. ***Простота.*** Имеет простой и понятный синтаксис, что делает его доступным для изучения даже начинающим разработчикам.
2. ***Универсальность.*** Используется для создания приложений для Windows, macOS, Linux и других платформ. Это делает его универсальным инструментом для разработки.
3. ***Безопасность.*** Обеспечивает высокий уровень безопасности благодаря встроенным механизмам защиты от ошибок и уязвимостей.
4. ***Интеграция с .NET.*** Тесно интегрирован с платформой .NET, что позволяет разработчикам использовать множество готовых библиотек и инструментов для создания приложений.
5. ***Масштабируемость.*** Позволяет создавать как небольшие приложения, так и крупные корпоративные системы.
6. ***Поддержка сообщества.*** Имеет большое и активное сообщество разработчиков, которые делятся опытом, знаниями и инструментами.
7. ***Регулярные обновления.*** Microsoft регулярно выпускает обновления и новые версии C#, что позволяет разработчикам использовать новые функции и возможности.

**Windows Forms** – это технология, разработанная корпорацией Microsoft, которая позволяет создавать графический интерфейс пользователя (GUI) для приложений, работающих под управлением операционных систем Windows. Ниже представлены основные достоинства этой технологии:

1. ***Простота использования.*** Windows Forms предоставляет разработчикам простой и понятный способ создания графического интерфейса пользователя. Разработчики могут использовать готовые элементы управления, такие как кнопки, текстовые поля, списки и т. д., для создания пользовательского интерфейса.
2. ***Поддержка.*** Windows Forms является частью платформы .NET, поэтому разработчики могут использовать множество готовых библиотек и инструментов для создания приложений.
3. ***Интеграция с Visual Studio.*** Windows Forms интегрирована с Visual Studio, что упрощает процесс разработки и отладки приложений.
4. ***Кроссплатформенность.*** Хотя Windows Forms изначально разрабатывалась для Windows, она также может быть использована для создания кроссплатформенных приложений с использованием Mono.
5. ***Широкое сообщество.*** Windows Forms имеет большое и активное сообщество разработчиков, которые делятся опытом, знаниями и инструментами.
6. ***Надёжность и стабильность.*** Windows Forms обеспечивает надёжную и стабильную работу приложений.
7. ***Поддержка локализации.*** Windows Forms позволяет разработчикам создавать приложения, поддерживающие различные языки и культуры.

Windows Forms является популярным инструментом для создания графических интерфейсов пользователя для приложений, работающих на платформе Windows. Она широко используется в различных областях, таких как бизнес-приложения, игры и образовательные программы.

**Visual Studio** – это интегрированная среда разработки (IDE) от компании Microsoft, которая предоставляет разработчикам инструменты для создания различных приложений, таких как веб-приложения, мобильные приложения, игры и другие. Visual Studio является одним из самых популярных инструментов для разработки программного обеспечения и используется миллионами разработчиков по всему миру. Ниже описаны основные достоинства данной среды разработки:

1. ***Широкий спектр возможностей.*** Visual Studio предоставляет разработчикам множество инструментов для создания приложений, включая редакторы кода, отладчики, профилировщики, дизайнеры пользовательского интерфейса и многое другое.
2. ***Поддержка различных языков программирования.*** Visual Studio поддерживает множество языков программирования, включая C#, C++, JavaScript, Python и другие.
3. ***Интеграция с облачными сервисами.*** Visual Studio позволяет разработчикам интегрировать свои приложения с облачными сервисами, такими как Azure, AWS и Google Cloud.
4. ***Поддержка проектов с открытым исходным кодом.*** Visual Studio имеет встроенную поддержку проектов с открытым исходным кодом, что упрощает разработку и интеграцию с существующими проектами.
5. ***Сообщество и документация.*** Visual Studio имеет большое и активное сообщество разработчиков, которые делятся опытом, знаниями и инструментами. Кроме того, Microsoft предоставляет обширную документацию и учебные материалы по использованию Visual Studio.
6. ***Регулярные обновления.*** Microsoft регулярно выпускает обновления и новые версии Visual Studio, что позволяет разработчикам использовать новые функции и возможности.
7. ***Кроссплатформенность.*** Visual Studio может быть использован для разработки приложений, работающих на различных платформах, включая Windows, macOS и Linux.

Помимо того, в реализации была использована СУБД MySQL. В ней было реализовано хранилище данных, а также все запросы к БД для отображения и изменения данных. Следует также описать данную СУБД.

**MySQL** – это реляционная система управления базами данных (СУБД), разработанная шведской компанией MySQL AB, которую затем приобрела Oracle. MySQL используется для хранения, поиска и обработки данных в различных приложениях и системах. Основные особенности MySQL описаны ниже:

1. ***Простота.*** MySQL имеет простой и понятный синтаксис, что делает его доступным для изучения даже начинающим разработчикам.
2. ***Скорость.*** MySQL оптимизирован для быстрой работы с большими объёмами данных, что делает его подходящим для использования в высоконагруженных системах.
3. ***Масштабируемость.*** MySQL может быть масштабирован для работы с большими объёмами данных и обработки большого количества запросов.
4. ***Безопасность.*** MySQL обеспечивает высокий уровень безопасности данных благодаря встроенным механизмам защиты от несанкционированного доступа и уязвимостей.
5. ***Поддержка.*** MySQL имеет большое и активное сообщество разработчиков, которые делятся опытом, знаниями и инструментами.
6. ***Совместимость.*** MySQL совместим с различными языками программирования и платформами, что делает его универсальным инструментом для разработки.
7. ***Открытость.*** MySQL является открытым исходным кодом, что позволяет разработчикам свободно использовать, модифицировать и распространять его.

### Работа основного функционала

Авторизация

Форма авторизации является первой точкой касания пользователя и информационной системы. В данном случае, форма делится на три блока: авторизация для ученика, авторизация для преподавателя и упрощенная форма регистрации в системе. Помимо этого, строку в поле пароля можно скрыть при помощи вспомогательной кнопки, находящейся справа от поля для ввода пароля.

На Рис. 48 представлена форма для авторизации ученика:

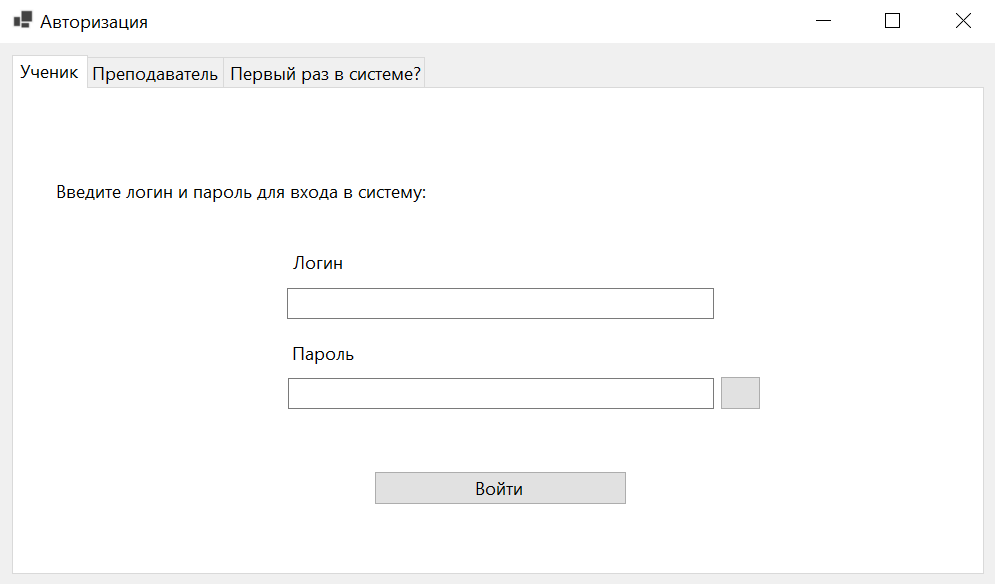


Рис. 48. Форма авторизации ученика в системе

Далее была реализована подобная форма для авторизации преподавателя. На Рис. 49 представлена форма для авторизации преподавателя:

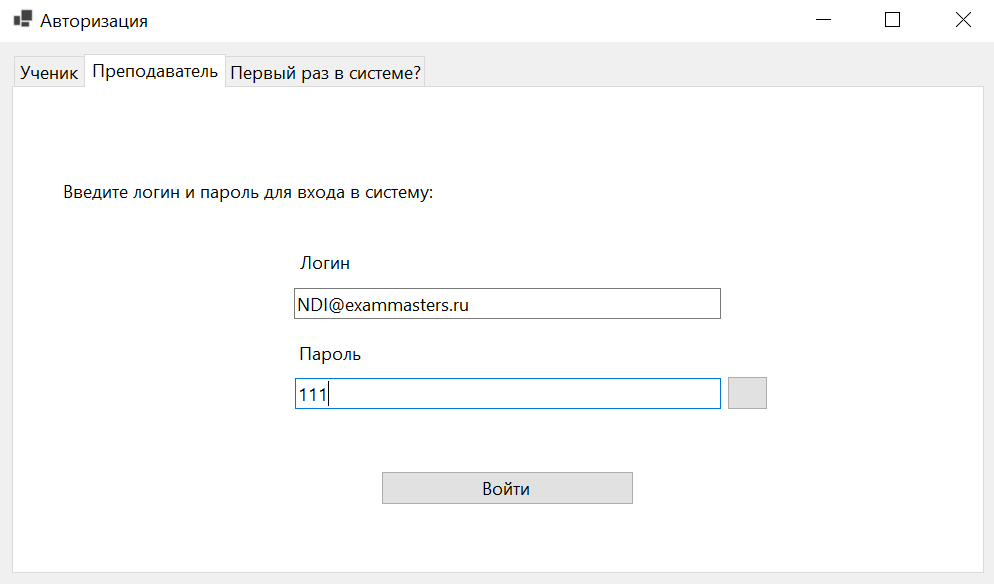


Рис. 49. Форма авторизации преподавателя в системе

Для того, чтобы не нагружать интерфейс системы блоком авторизации администратора, было принято решение оставить авторизацию администратора в блоке преподавателей. Как и преподаватель, администратор должен ввести универсальную почту администратора и пароль. Вход в личный кабинет администратора показан ниже на Рис. 50:

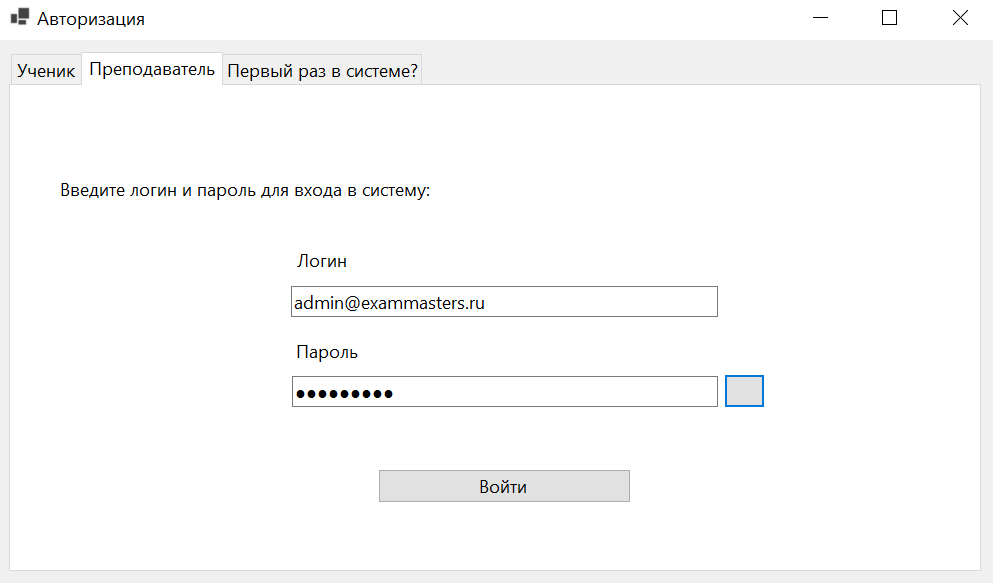


Рис. 50. Вход в личный кабинет администратора

В случае, если администратор добавил пользователя в хранилище данных, однако еще не авторизовался в системе (не установил пароль для своей учетной записи), следует выбрать третий блок «Первый раз в системе?». Для того, чтобы зарегистрироваться, следует придумать пароль и затем повторить ввод пароля. Если два введенных пароля идентичны, устанавливается новый пароль для личного кабинета пользователя. Форма показана ниже на Рис. 51:

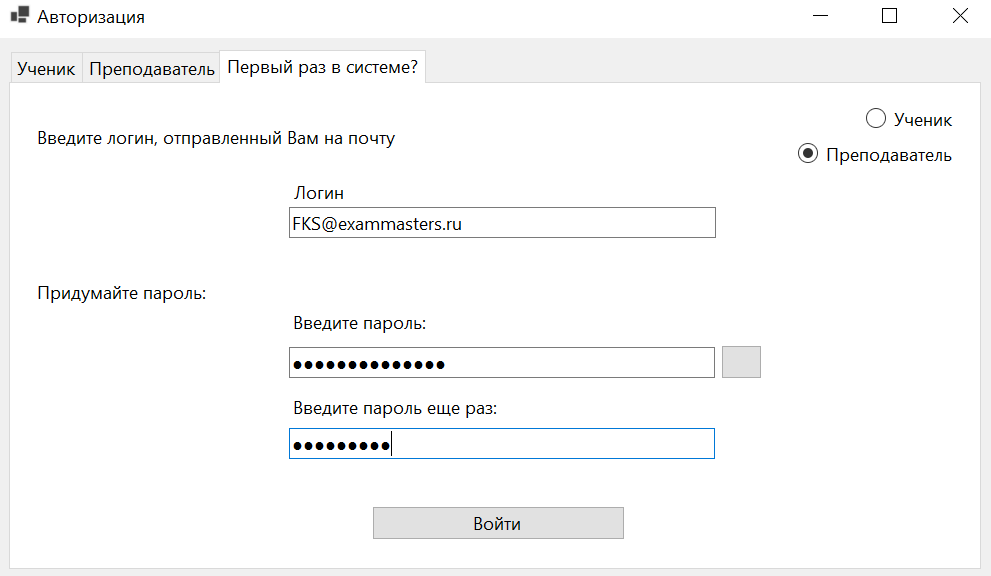


Рис. 51. Установление пароля учетной записи пользователя

Функционал ИС под разными пользователями

После авторизации пользователя в системе открывается форма главной страницы личного кабинета. Далее будет рассмотрены личные кабинеты каждого из видов ролей пользователя. Для начала описан функционал личного ***кабинета студента***.

Ниже на Рис. 52 представлена форма главной страницы личного кабинета ученика:

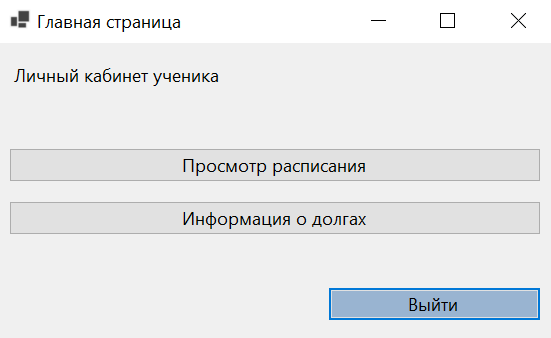


Рис. 52. Главная страница личного кабинета ученика

При нажатии на кнопку «Просмотр расписания» открывается форма вывода расписания для ученика. Эта форма представлена на Рис. 53:

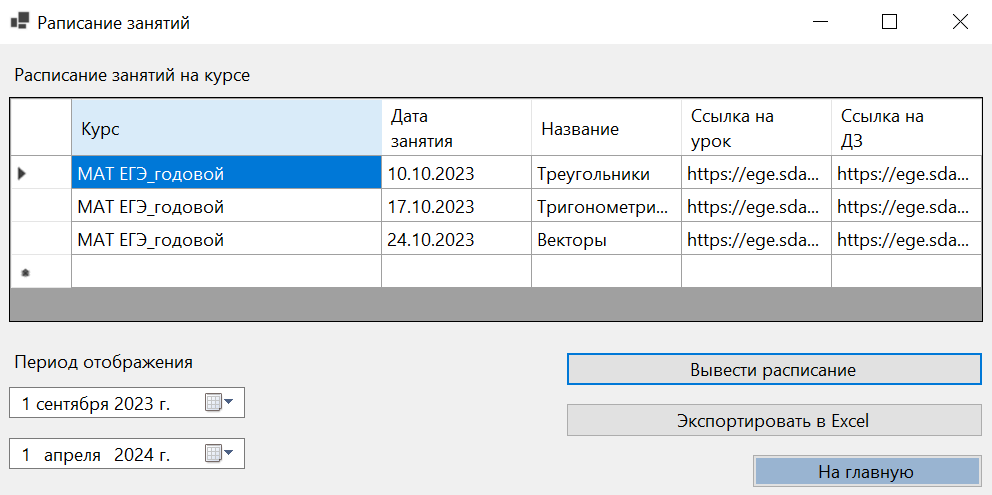


Рис. 53. Форма «Расписание занятий» в личном кабинете ученика

В данном случае, можно сортировать занятия по периоду отображения, а также экспортировать данные в Excel.

При нажатии на кнопку «Информация о долгах» на главной странице личного кабинета открывается форма «Долги по учебе». В таблице долгов отображаются занятия, которые ученик либо не посетил, либо выполнил не все домашнее задание. Эта форма показана ниже на Рис. 54:

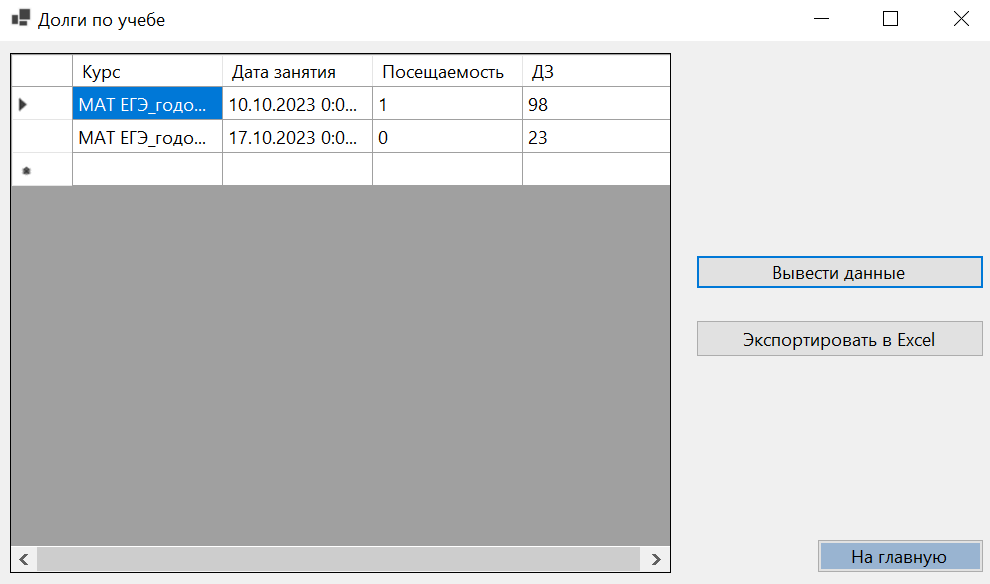


Рис. 54. Форма «Долги по учебе» в личном кабинете ученика

Далее был реализован личный кабинет ***преподавателя***. В него можно перейти из формы «Авторизация» при правильном вводе логина и пароля преподавателя. Ниже на Рис. 55 представлена форма главной страницы личного кабинета преподавателя:

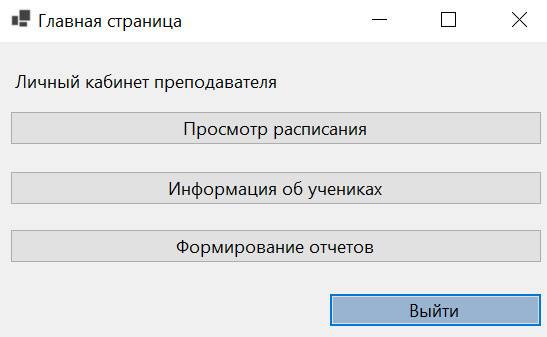


Рис. 55. Главная страница личного кабинета преподавателя

При нажатии кнопки «Просмотр расписания» открывается форма «Расписание преподавателя». На этой форме можно выбрать сортировку по периоду отображения, а также выбрать группы, для которых необходимо вывести расписание. Стоит отметить, что преподаватель может посмотреть расписание только тех групп, в которых он ведет занятия. Эта форма показана ниже на Рис. 56:

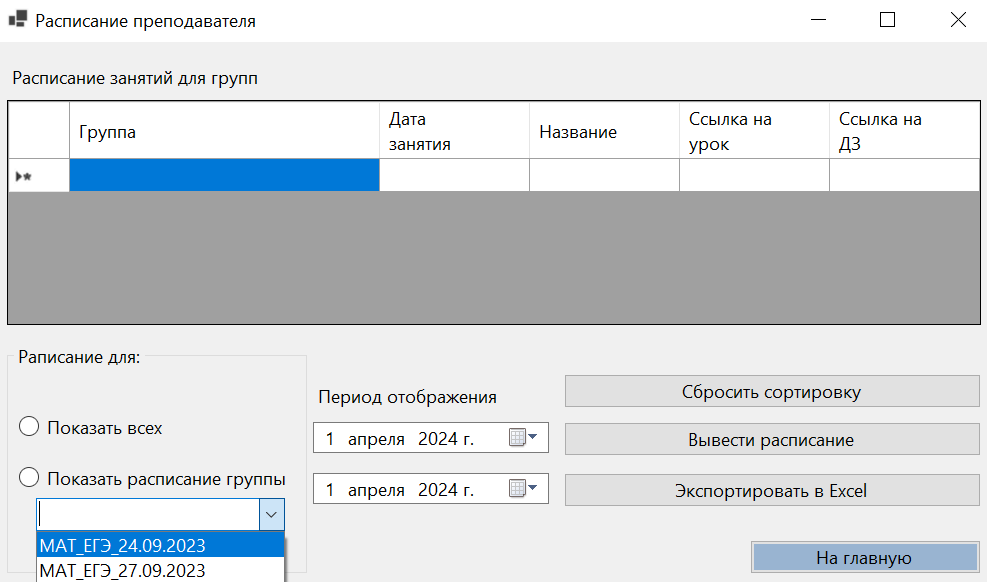


Рис. 56. Форма «Расписание преподавателя»

При нажатии на кнопку «Информация об учениках» открывается одноименная форма. Чтобы вывести данные на этой форме нужно выбрать группу и нажать на кнопку «Вывести данные». Преподаватель может посмотреть расписание только тех групп, в которых он ведет занятия. В таблице выводятся имена учеников группы, а также их номера телефонов и адреса электронных почт. Таким образом, преподаватель может экстренно связаться с учеником в случае необходимости. Форма представлена ниже на Рис. 57:

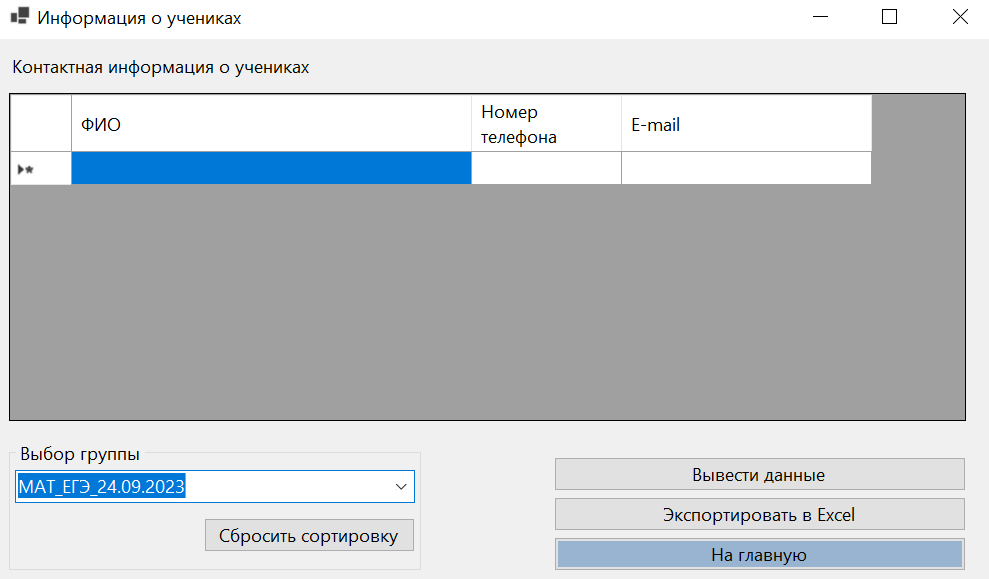


Рис. 57. Форма «Информация о учениках» в личном кабинете преподавателя

При нажатии на кнопку «Формирование отчетов» открывается форма «Сводные статистики по ученикам». Преподаватель может выбрать две функции для работы с данными: вывод сводного отчета средних значений по показателям, а также отчет по показателям каждого занятия. Ниже на Рис. 58 показан блок вывода средней статистики:

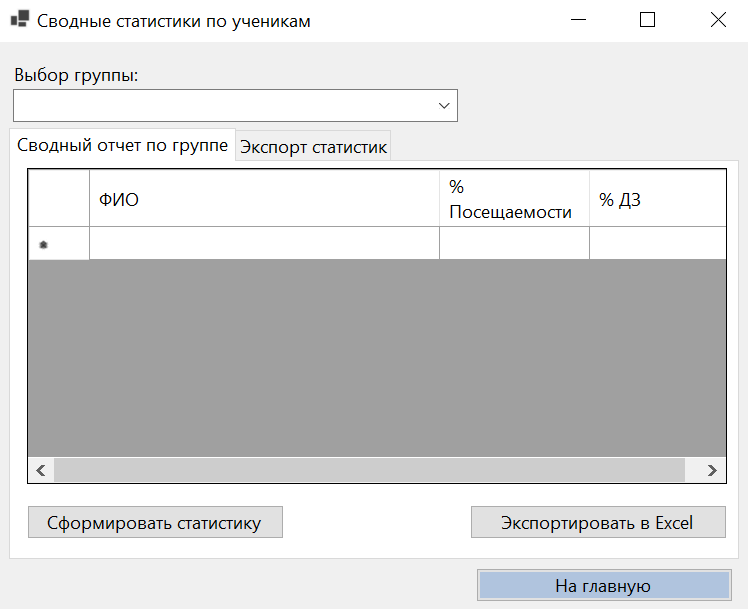


Рис. 58. Форма «Сводные статистики по ученика. Сводный отчет по группе» в личном кабинете преподавателя

Далее был реализован блок вывода статистики по отдельным занятиям группы. Однако, для удобства отображения данных этот блок был реализован как экспорт статистик в Excel. В этом блоке преподаватель может экспортировать проценты домашнего задания или посещаемость по всем занятиям группы. Вид отчетов по занятиям будет рассмотрен более подробно в разделе «Отчеты». Ниже на Рис. 59 представлена реализация этого блока:

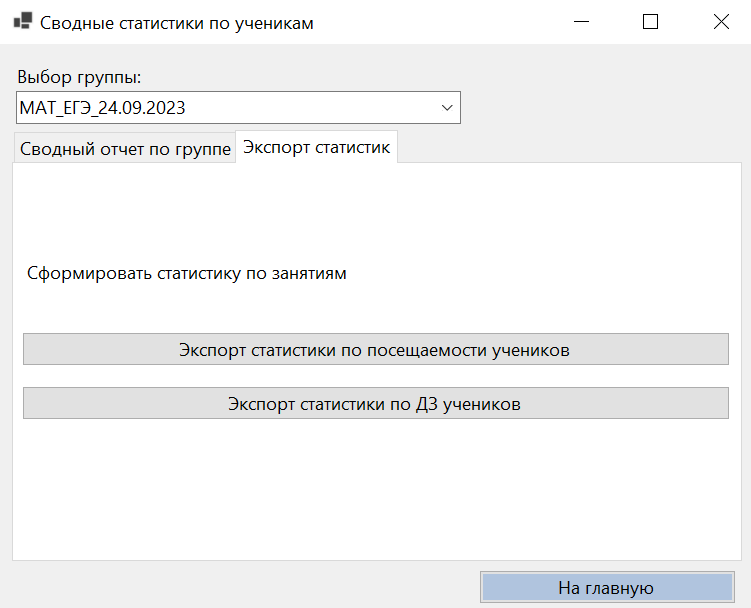


Рис. 59. Форма «Сводные статистики по ученикам. Экспорт статистик» в личном кабинете преподавателя

Далее был реализован функционал личного кабинета ***администратора***. В системе *«Exam Masters»* роль «Администратор» полностью отвечает за работу с данными (удаление, добавление). Помимо этого, администраторы выгружают отчетную статистику всех учеников, преподавателей и групп для анализа и контроля качества преподавания. Ниже на Рис. 60 представлена главная страница личного кабинета администратора:

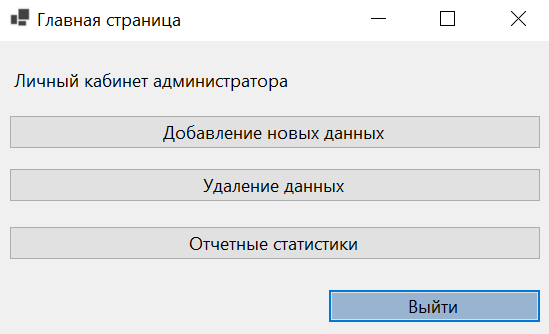


Рис. 60. Главная страница личного кабинета администратора

При нажатии на кнопку «Добавление новых данных» открывается одноименная форма, в которой администратор может создавать новые объекты системы, в дальнейшем необходимые для осуществления образовательного процесса. Для удобства работы с данными объекты были разделены по двум группам: пользователи и курсы. Ниже на Рис. 61 представлен блок «Пользователи»:

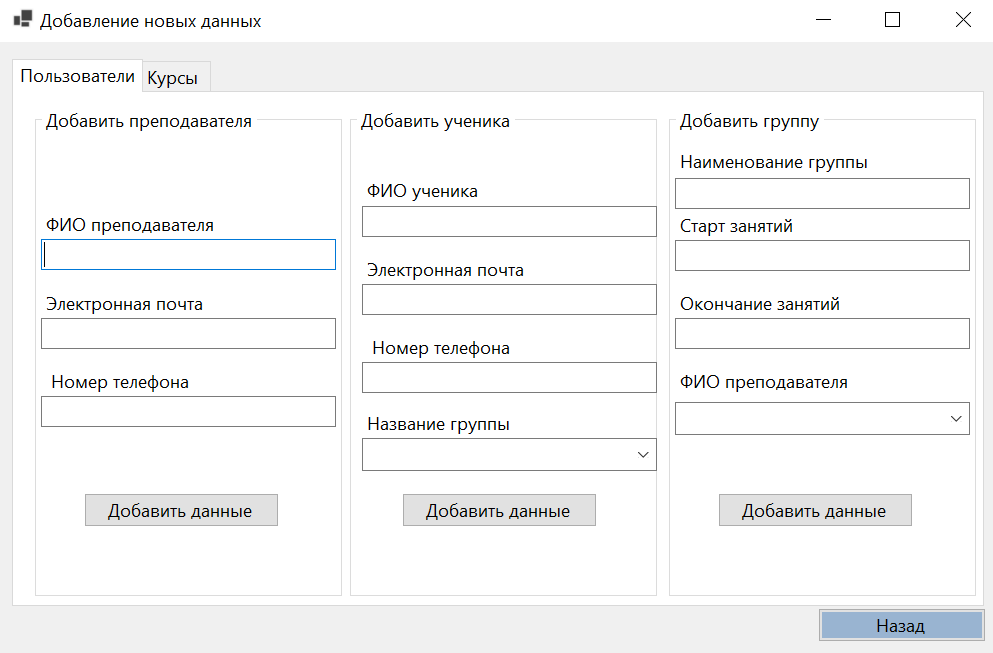


Рис. 61. Форма «Добавление новых данных. Пользователи» в личном кабинете администратора

Ниже на Рис. 15 представлен блок «Курсы» в этой же форме:

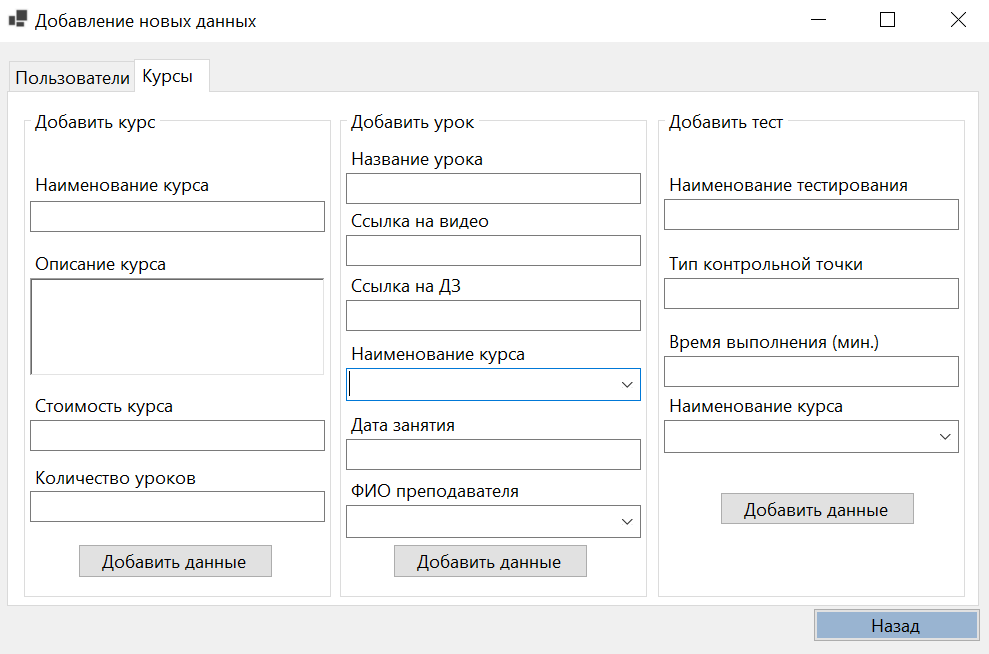


Рис. 15. Форма «Добавление новых данных. Курсы» в личном кабинете администратора

При нажатии на кнопку «Удаление данных» открывается одноименная форма, в которой администратор может удалять объекты системы, в дальнейшем не нужные для функционирования образовательного процесса. Ниже на Рис. 63 представлена эта форма:

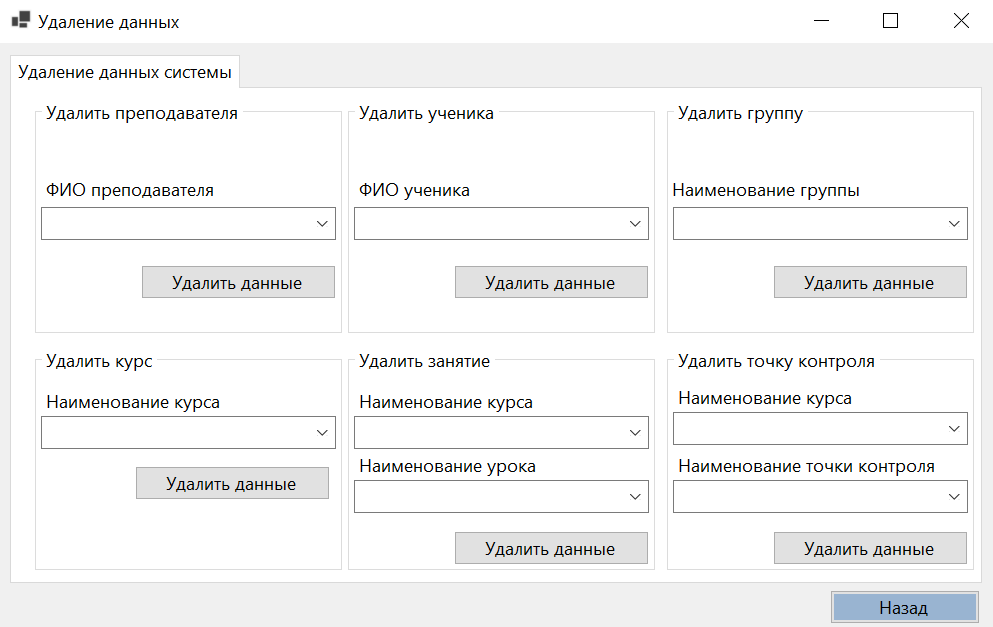


Рис. 63. Форма «Удаление данных» в личном кабинете администратора

При нажатии на кнопку «Отчетные статистики» администратор может вывести статистику по трем объектам: преподавателям, ученикам и группам. Эти объекты разделены по трем разным страницам. В данном случае, можно отсортировать объекты по показателям. Стоит отметить, что если в поля сортировки не вводятся цифры, тогда за верхние границы берутся 100%. Ниже на Рис. 64 представлен раздел «Преподаватели»:

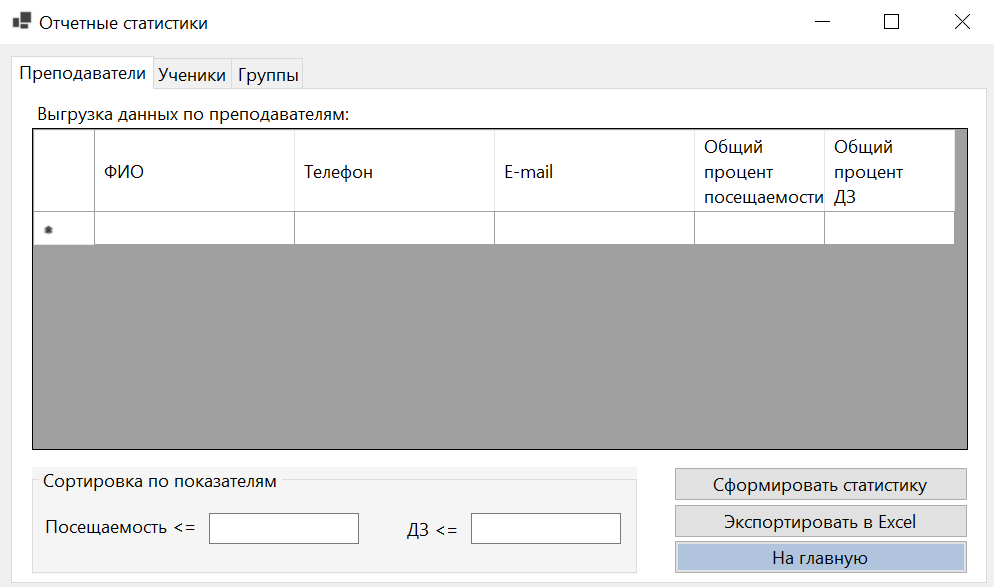


Рис. 64. Форма «Отчетные статистики. Преподаватели» в личном кабинете администратора

На Рис. 65 представлен раздел «Ученики»:

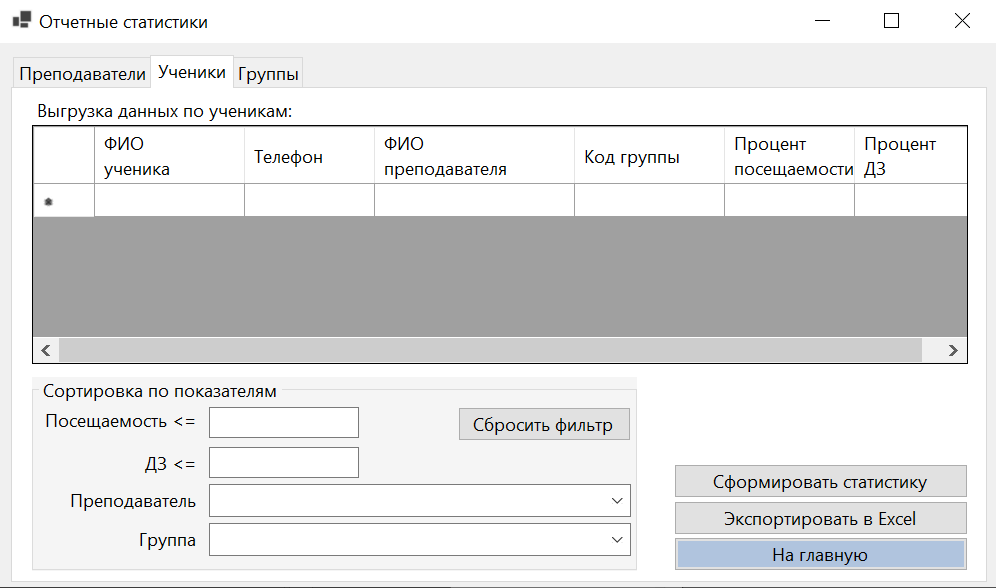


Рис. 65. Форма «Отчетные статистики. Ученики» в личном кабинете администратора

На Рис. 66 представлен раздел «Группы»:

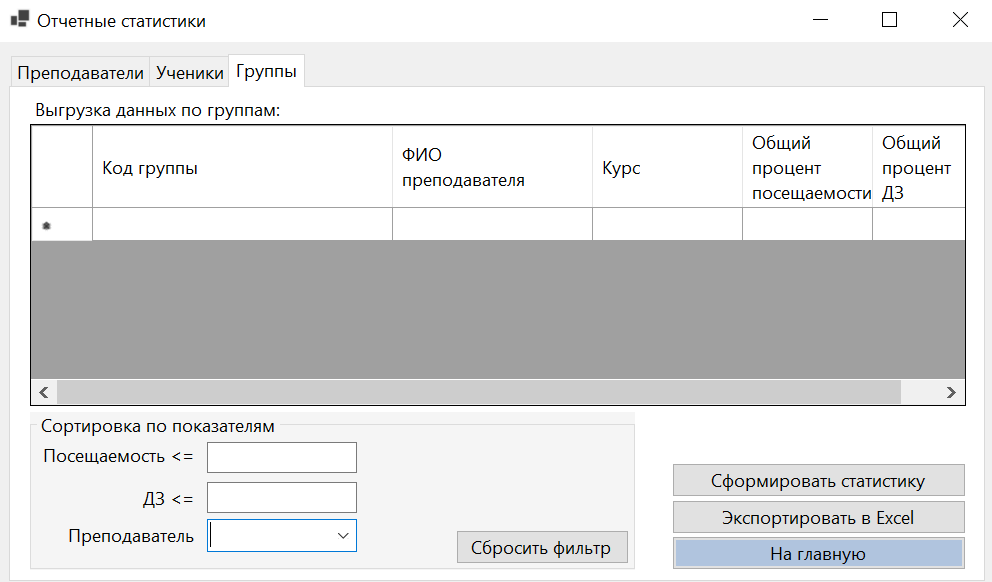


Рис. 66. Форма «Отчетные статистики. Группы» в личном кабинете администратора

### Отчеты

Далее был реализован вывод отчетов для личных кабинетов всех ролей пользователей. Для начала была проведена работа с ***формированием отчетов и экспортом данных в личном кабинете ученика***. Ученик может составлять расписание занятий на курсе, по которому он занимается. Ниже на Рис. 67 представлен вывод этого отчета:

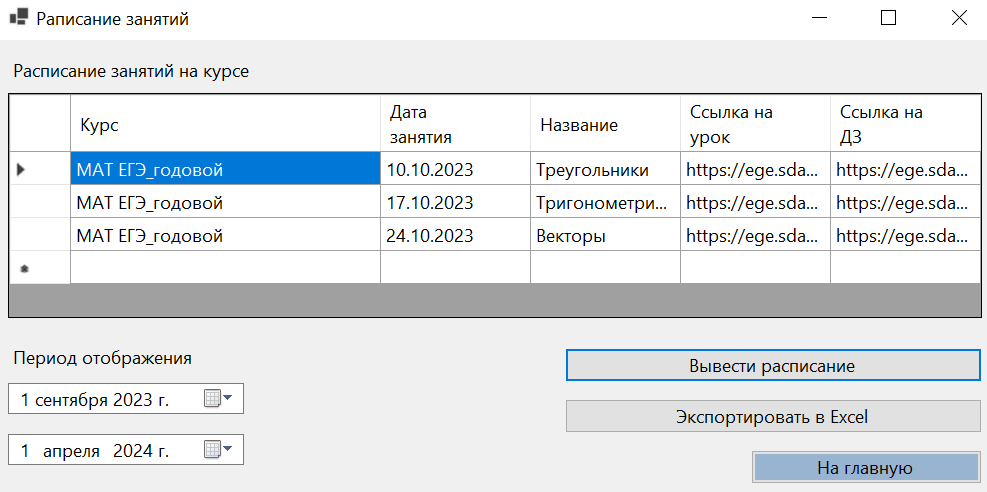


Рис. 67. Вывод отчета на форме «Расписание занятий» в личном кабинете ученика

Далее был реализован экспорт данных в Excel. В данном случае, экспортируются данные, которые представлены в таблицы выше. Данный экспорт представлен ниже на Рис. 68:

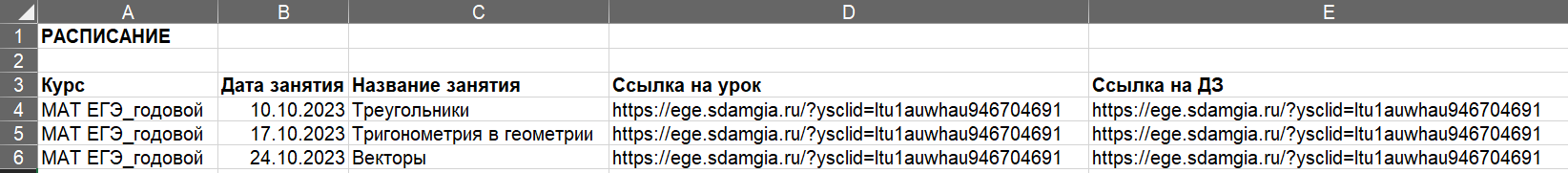


Рис. 68. Экспорт отчета «Расписание занятий»

Помимо этого, ученик может сформировать данные о своих долгах по курсу. Ниже на Рис. 69 представлен пример вывода долгов в личном кабинете ученика:

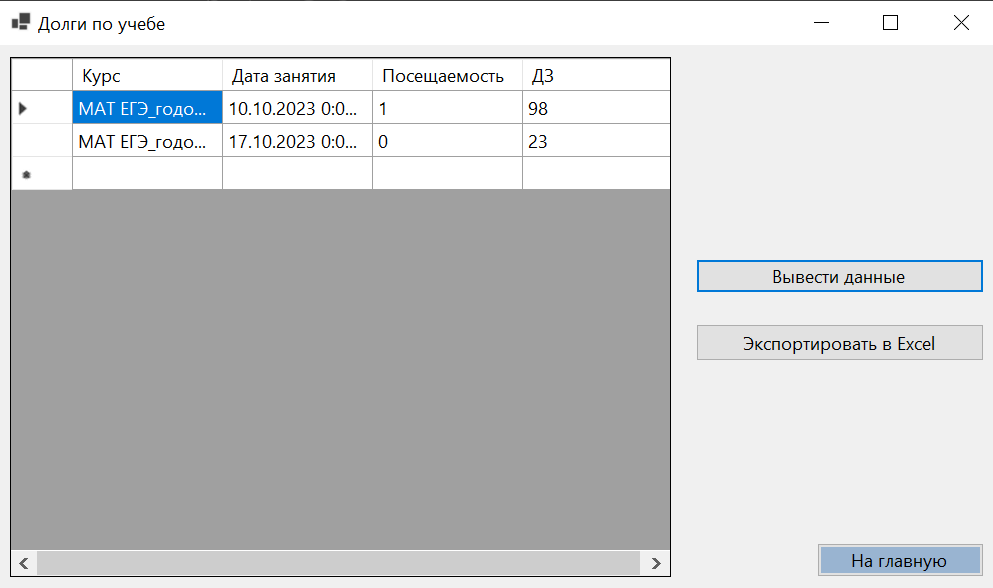


Рис. 69. Вывод отчета на форме «Долги по учебе» в личном кабинете ученика

Эти данные ученик также может экспортировать в Excel. Пример экспортированного файла представлен ниже на Рис. 70:

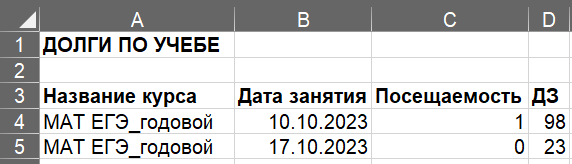


Рис. 70. Экспорт отчета «Долги по учебе»

Далее были реализованы ***отчеты и их экспорт для личного кабинета преподавателя***. Преподаватель может также составлять вывод расписания занятий для групп. Ниже на Рис. 71 представлен пример вывода расписания:

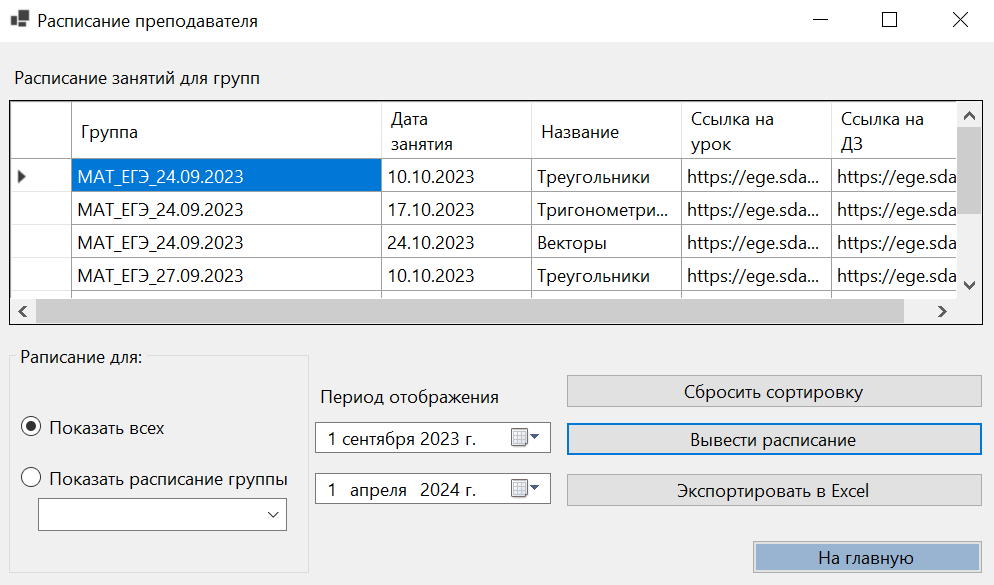


Рис. 71. Вывод отчета на форме «Расписание преподавателя» в личном кабинете преподавателя

Далее был реализован экспорт этого отчета в Excel. Пример экспорта данных представлен ниже на Рис. 72:

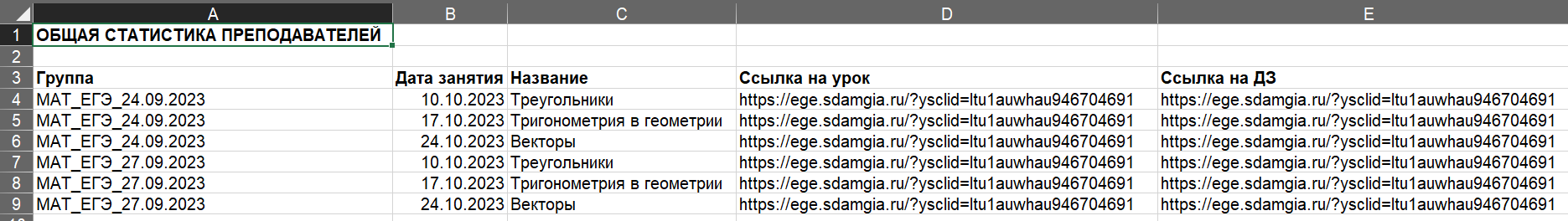


Рис. 72. Экспорт отчета «Расписание преподавателя»

При нажатии на кнопку «Информация о учениках» открывается одноименная форма, на которой преподаватель может сформировать отчет по контактной информации учеников группы. Ниже на Рис. 73 представлен пример такого отчета:

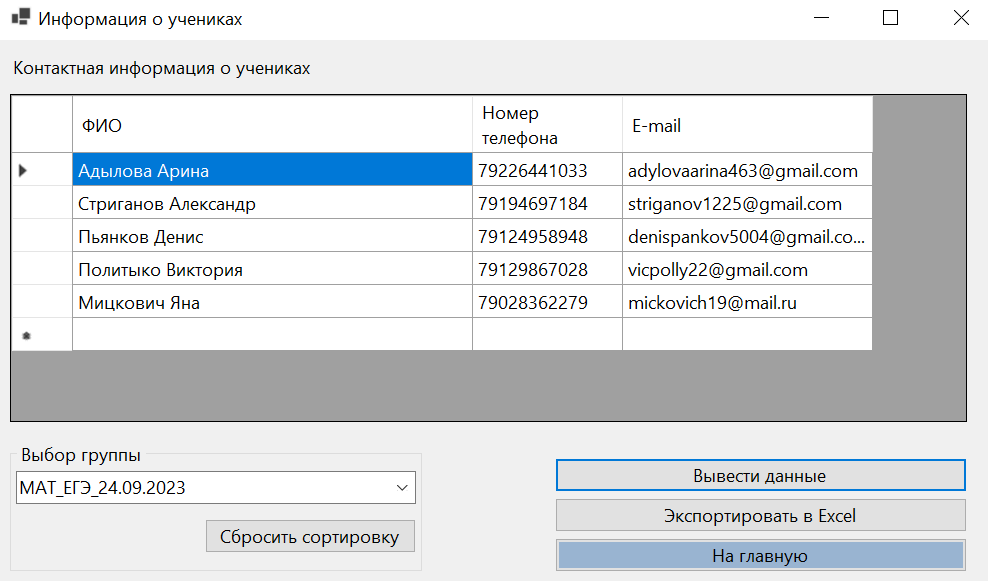


Рис. 73. Вывод отчета на форме «Информация о учениках» в личном кабинете преподавателя

Далее был реализован экспорт этого отчета. Он представлен на Рис. 74:



Рис. 74. Экспорт отчета «Информация о учениках»

Далее был реализован вывод и экспорт данных для формы «Сводные статистики по ученикам» в личном кабинете преподавателя. На Рис. 75 представлен вариант вывода сводной статистики по группе:

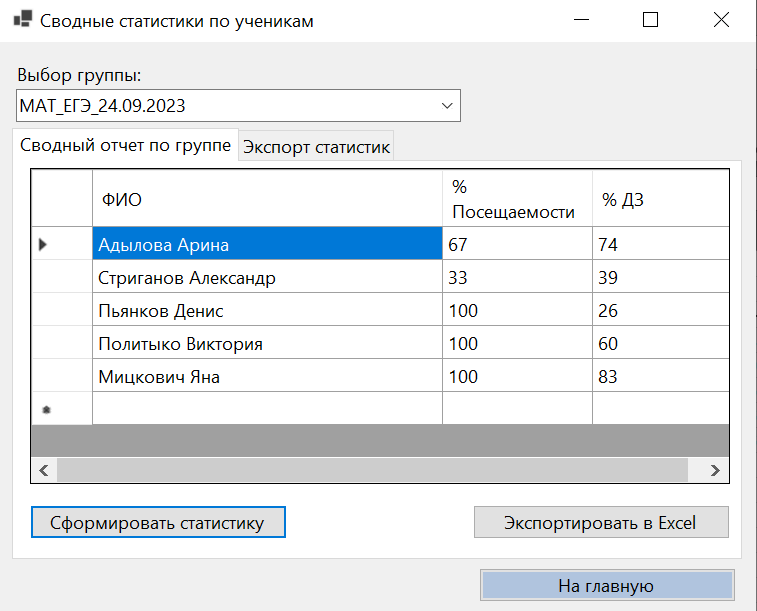


Рис. 75. Вывод отчета на форме «Сводный отчет по группе» в личном кабинете преподавателя

Экспорт сводной статистики по группе представлен ниже на Рис. 76:

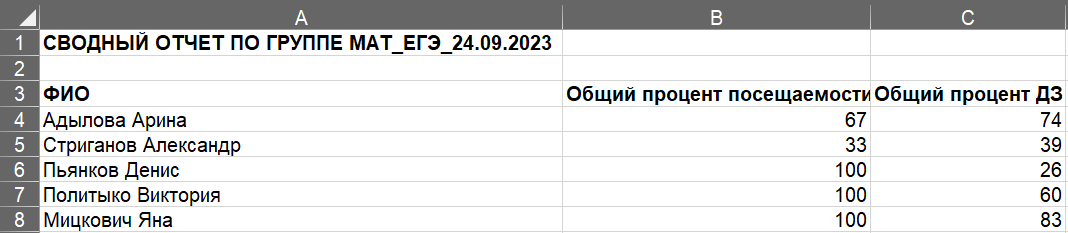


Рис. 76. Экспорт отчета «Сводный отчет по группе»

Далее был реализован экспорт статистики по занятиям. Ниже на Рис. 77 представлен экспорт статистики по посещаемости:

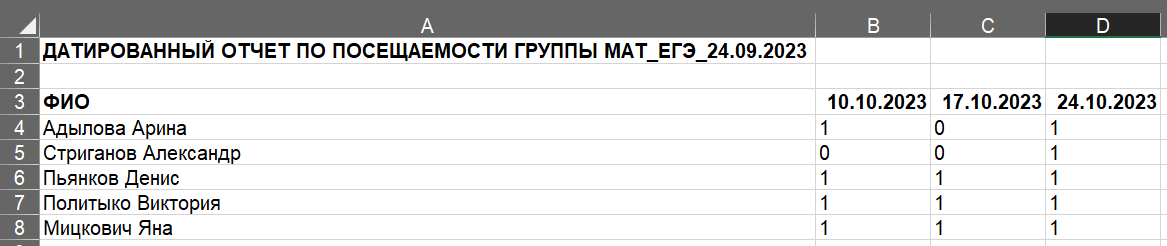


Рис. 77. Экспорт отчета «Датированная статистика по посещаемости группы»

После был реализован экспорт статистики по процентам выполнения домашнего задания. Ниже на Рис. 78 представлен экспорт статистики по ДЗ:

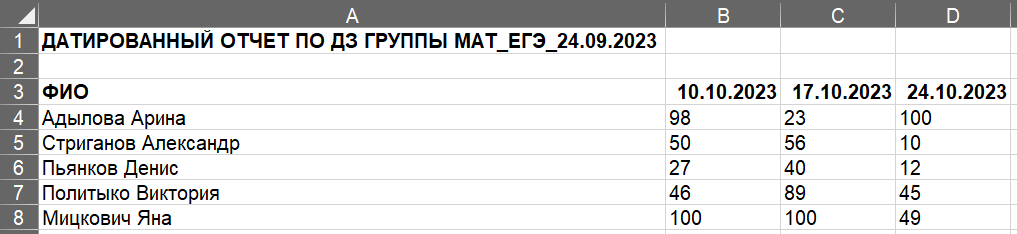


Рис. 78. Экспорт отчета «Датированная статистика по процентам домашнего задания группы»

Далее были реализованы ***отчеты и их экспорт для личного кабинета администратора***. Администратор может формировать и экспортировать статистики преподавателей. Эти функции показаны ниже на Рис. 79, Рис. 80:

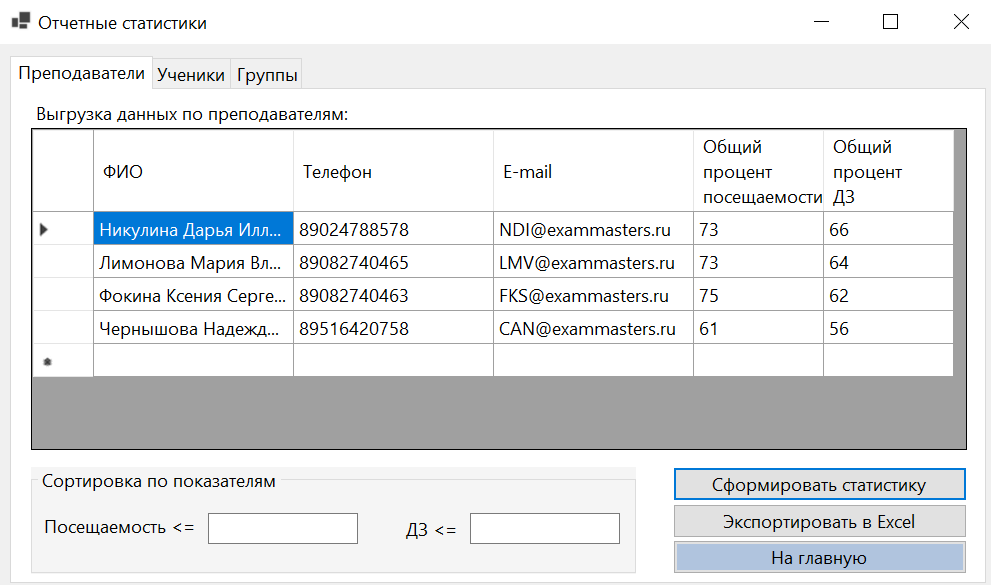


Рис. 79. Вывод отчета на форме «Отчетные статистики. Преподаватели» в личном кабинете администратора

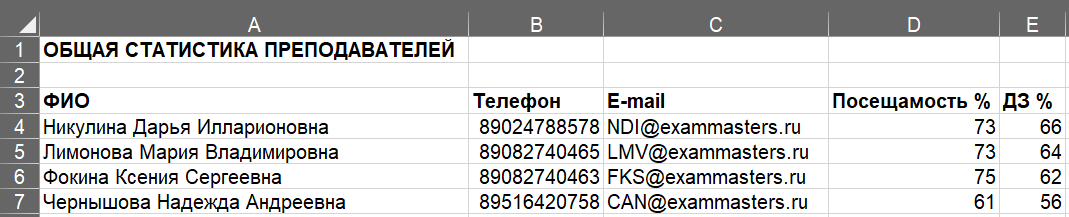


Рис. 80. Экспорт отчета «Общая статистика преподавателей»

Та же самая функция возможна для вывода статистики по ученикам. Вывод и экспорт данных представлен ниже на Рис. 81, Рис. 82:

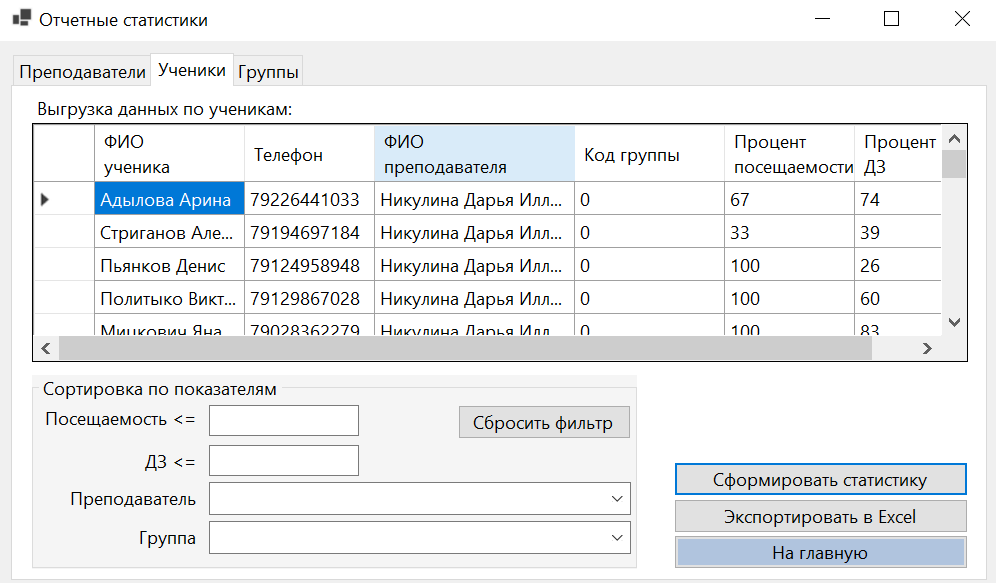


Рис. 81. Вывод отчета на форме «Отчетные статистики. Ученики» в личном кабинете администратора

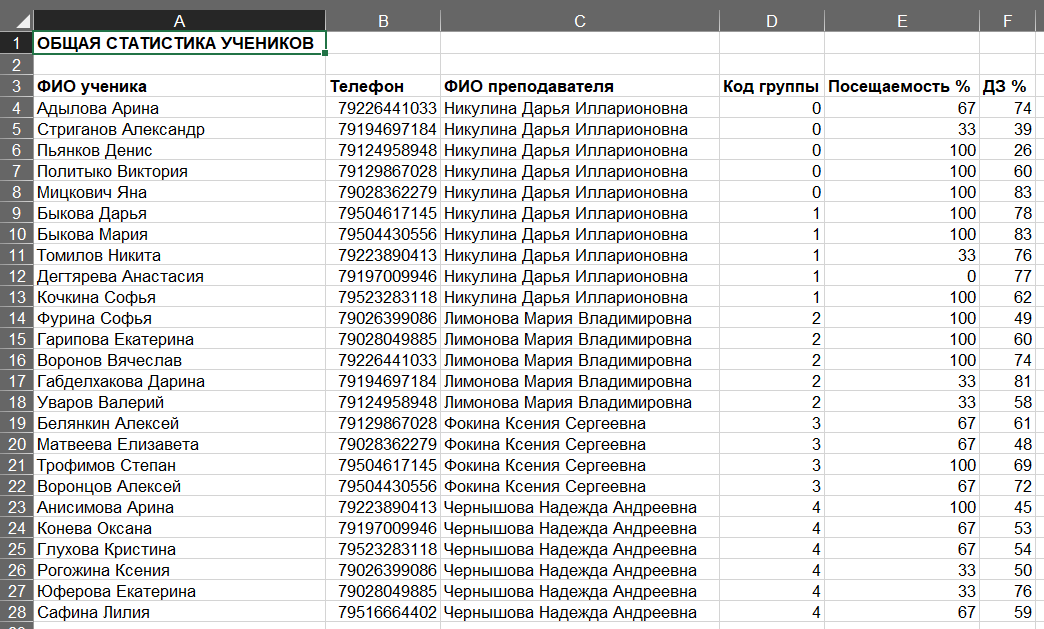


Рис. 82. Экспорт отчета «Общая статистика учеников». Отрывок отчета

Последняя функция вывода и экспорта была реализована для групп учеников. На Рис. 83, Рис. 84 представлены примеры вывода и экспорта отчетов:

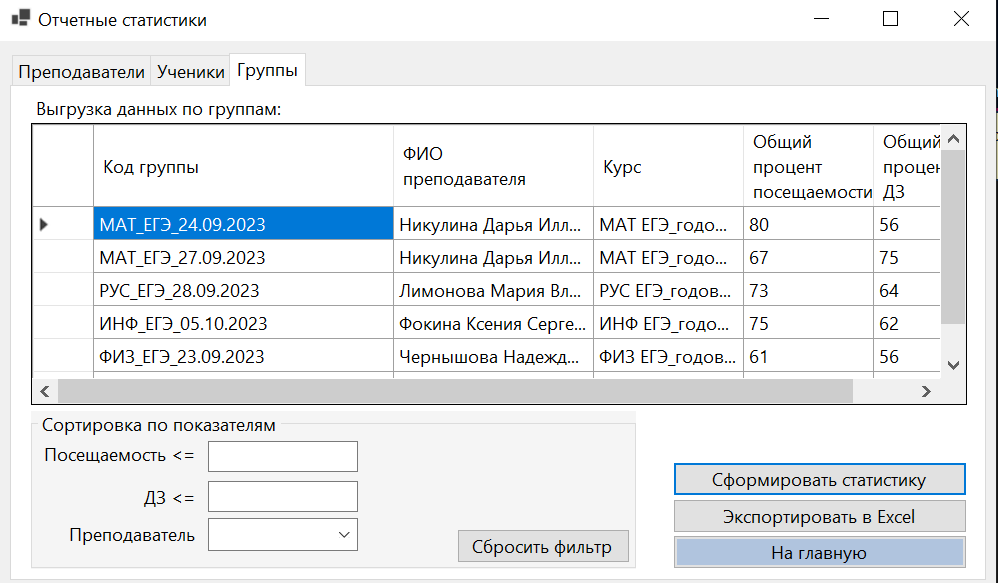


Рис. 83. Вывод отчета на форме «Отчетные статистики. Группы» в личном кабинете администратора

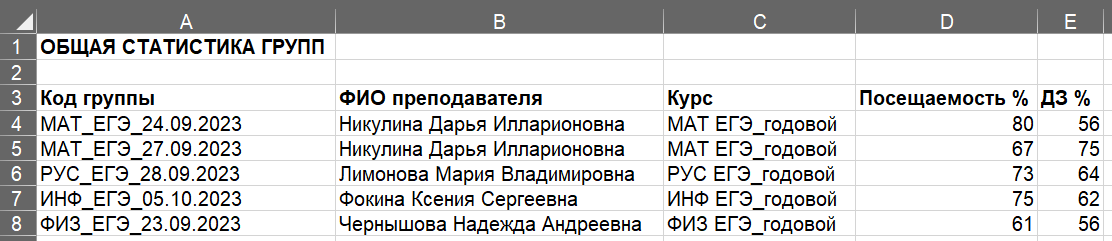


Рис. 84. Экспорт отчета «Общая статистика групп»

### Реакция ИС на ошибочный ввод данных

В форме «Авторизация» присутствует проверка введенных данных и ошибочный ввод данных можно классифицировать следующим образом:

1. **Незаполненные обязательные поля.** Пример представлен ниже на Рис. 85:

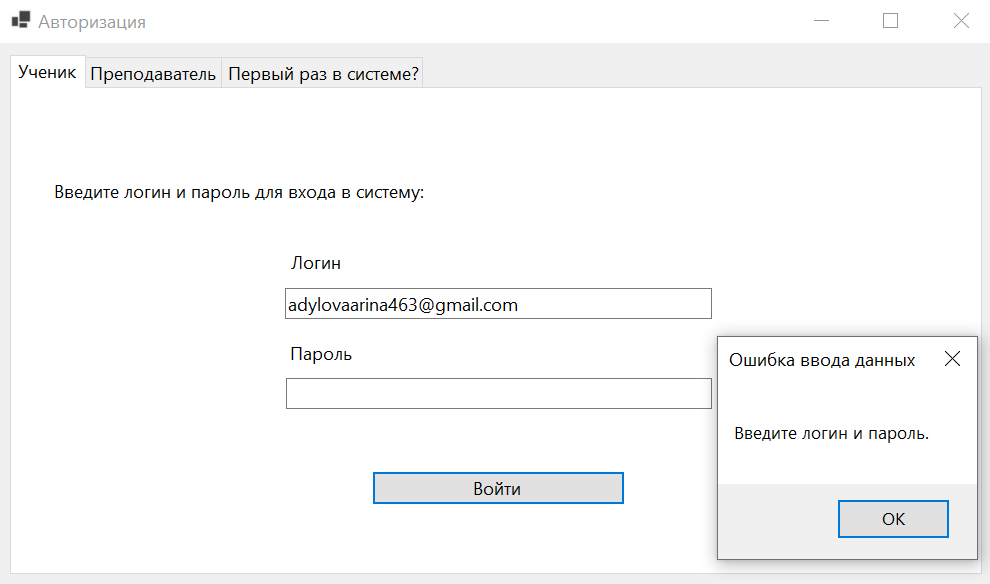


Рис. 85. Незаполненные поля в форме «Авторизация»

1. **Ввод неправильного пароля.** Пример представлен на Рис. 86:

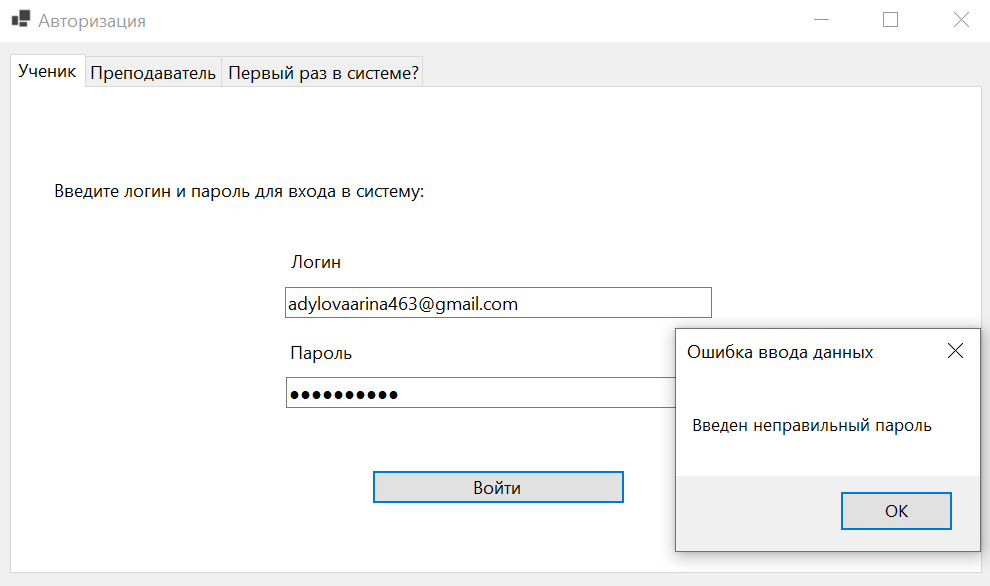


Рис. 86. Ввод неправильного пароля в форме «Авторизация»

1. **Несоответствие пароля и его проверки.** Пример представлен ниже на Рис. 87:

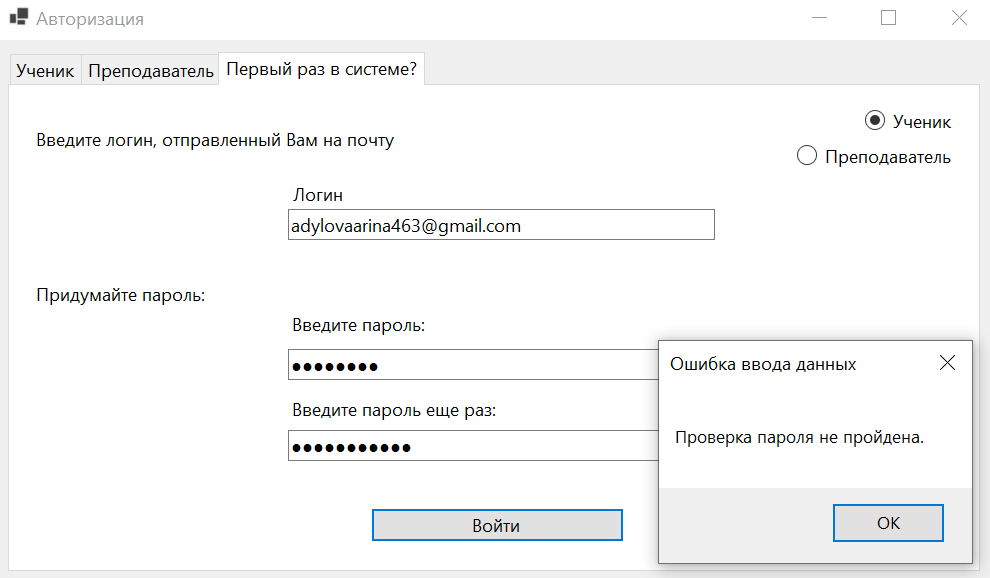


Рис. 87. Несоответствие пароля и его проверки

Далее были прописана обработка неправильного ввода данных в личном кабинете преподавателя. Основной ошибкой ввода данных являются незаполненные обязательные поля. Наиболее важным полем ввода является выбор группы и сортировки. Это прослеживается на протяжении работы со всем личным кабинетом преподавателя.

На форме «Расписание преподавателя» существует два варианта формирования данных: по всем группам и для одной конкретной группы. Таким образом, один из этих вариантов точно должен быть выбран. Ниже на Рис. 88 представлен ошибочный ввод данных в этой форме:

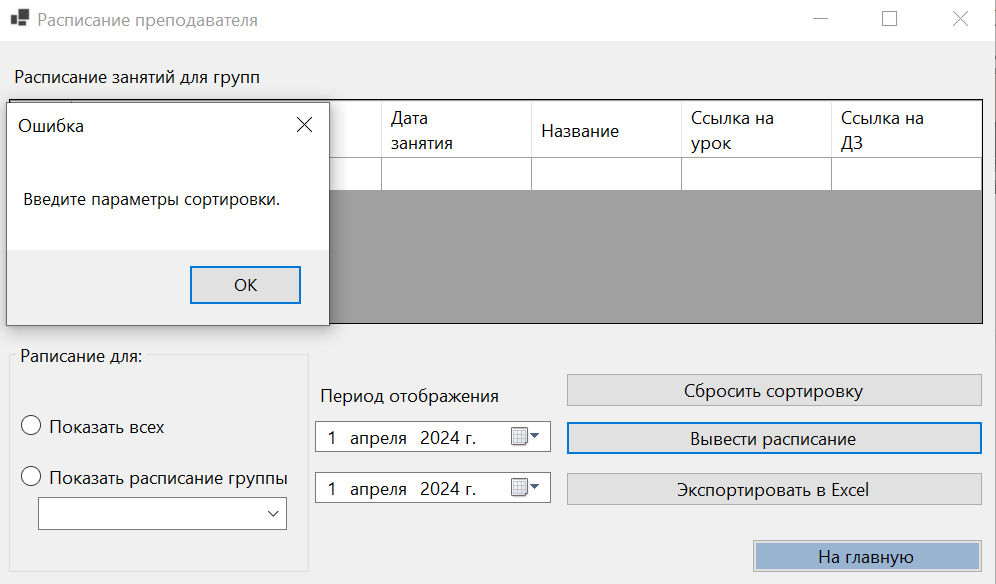


Рис. 88. Ошибочный ввод данных в форме «Расписание преподавателя»

В форме «Информация о учениках» пользователь обязательно должен выбрать группу для формирования списка данных. Поэтому была реализована реакция на ошибочное использование данных. Это показано ниже на Рис. 89:

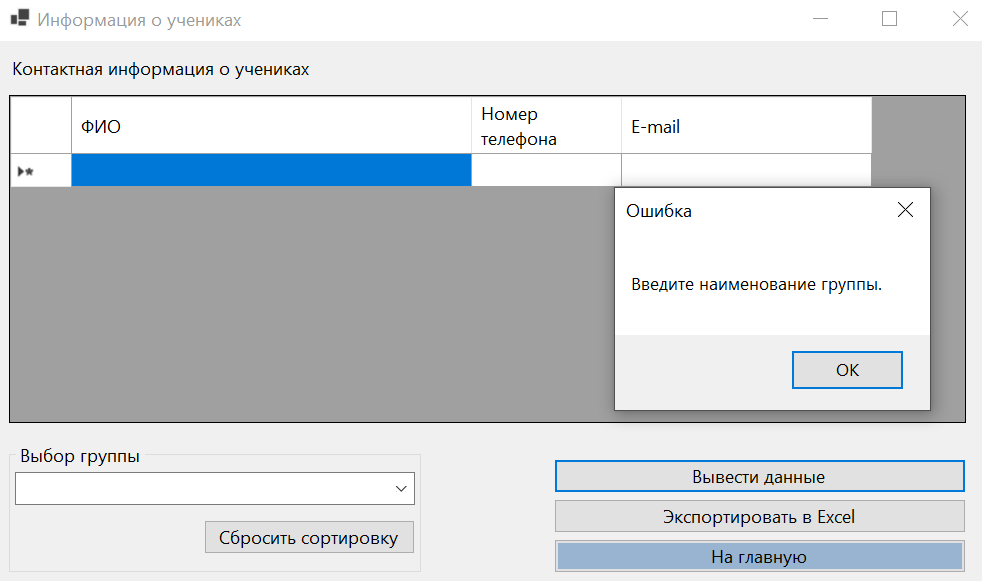


Рис. 89. Ошибочный ввод данных в форме «Информация о учениках»

Эта реакция была также прописана в форме «Сводные статистики по ученикам». Без выбора группы невозможно сформировать ее статистику. Эта реакция на ошибочный ввод данных представлен ниже на Рис. 90:

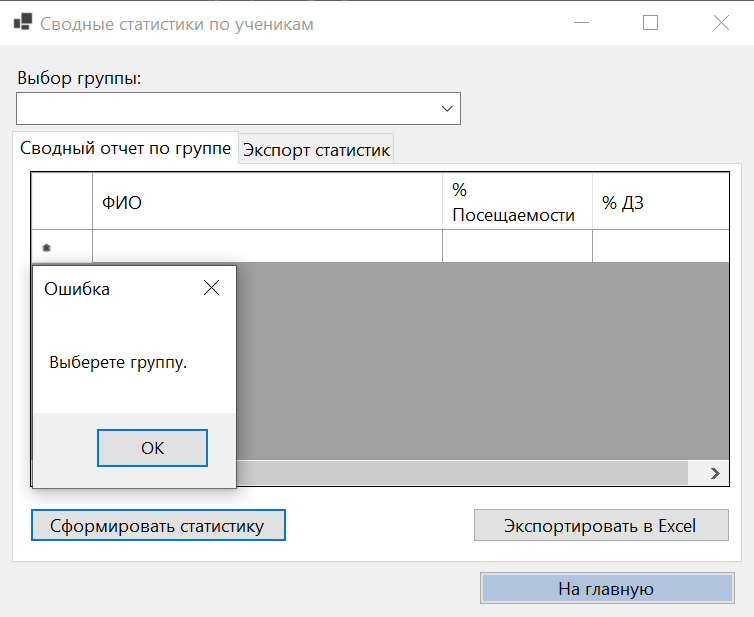


Рис. 90. Ошибочный ввод данных в форме «Сводные статистики по ученикам. Сводный отчет по группе»

В личном кабинете администратора можно совершить ошибку при добавлении и удалении данных. Все поля, представленные на формах, нужно заполнить, прежде чем добавлять или удалять данные в системе. Реакцию на ошибочное действие в системе представлено ниже на Рис. 91, Рис. 92:

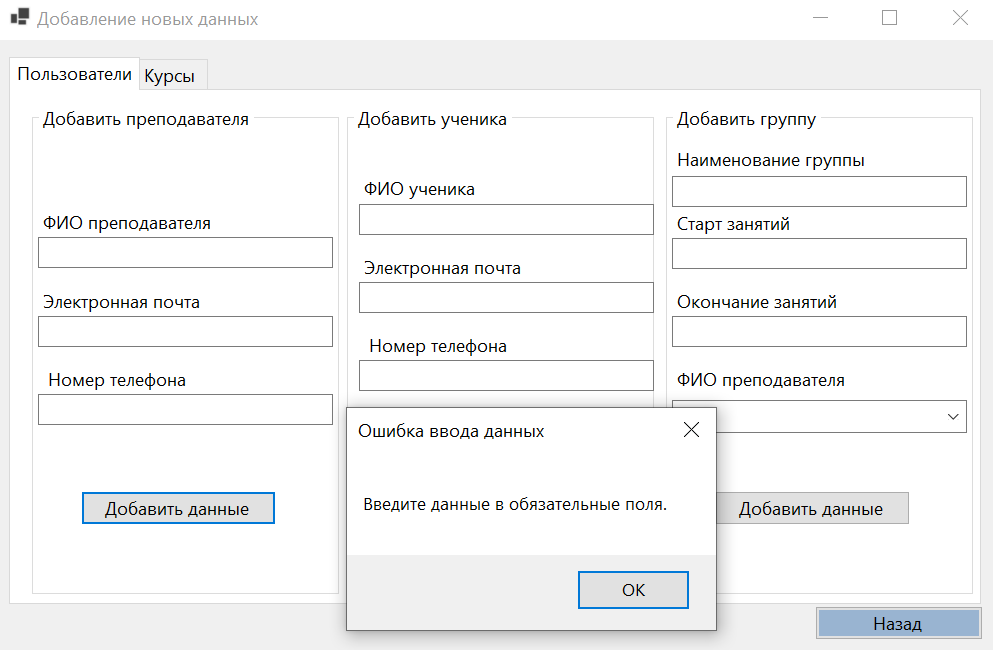


Рис. 91. Ошибочный ввод данных в форме «Добавление новых данных»

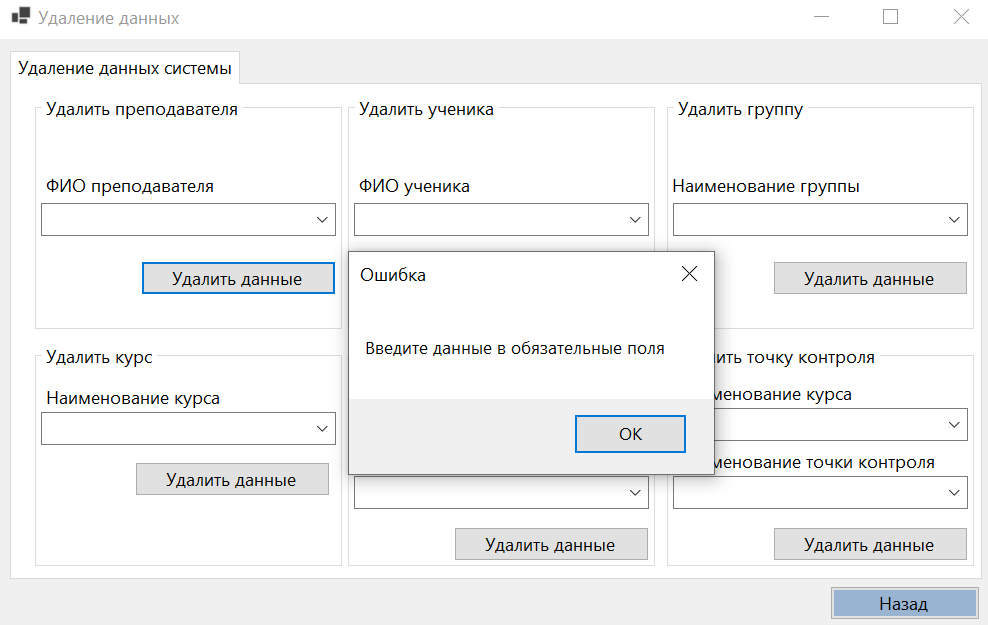


Рис. 92. Ошибочный ввод данных в форме «Удаление данных»

## Вывод

В ходе выполнения настоящей лабораторной работы было реализовано приложение корпоративной информационной системы *«Exam Masters»*. Были реализованы требования и условия, которые описаны на предыдущих этапах разработки.

Дальнейшим этапом, согласно ТЗ, является тестирование приложения на корректность работы.

# Тестирование КИС

## Постановка задачи

### Описание КИС и основного функционала

Наименование системы: «Модуль корпоративной информационной системы функционирования филиалов онлайн-школы “Exam Masters”».

Модуль корпоративной информационной системы онлайн-школы предназначен для обеспечения пользователям удобного учебного процесса как со стороны учеников, так и со стороны преподавателей и руководства.

Внутри системы пользователи смогут:

1. Добавлять данные в разделы в соответствии с ТЗ;
2. Просматривать статистики;
3. Выгружать отчеты, описанные в ТЗ.

В системе существует разграничение по правам пользователей. Преподавателям доступны следующие функции:

1. Просмотр групп, у которых он ведет занятия;
2. Просмотр каждого отдельного ученика в группе, анализ его статистики;
3. Выставление оценок, корректирование данных в соответствии с ТЗ;
4. Выгрузка отчетов по успеваемости, посещаемости и т.д.;
5. Функции по просмотру статистики, описанной в ТЗ;
6. Функции по выгрузке отчетов из разделов, описанных в ТЗ.

Ученикам доступны следующие функции:

1. Добавление данных по разделам, описанным в ТЗ;
2. Удаление данных по разделам, описанным в ТЗ;
3. Просмотр статистики по персональной успеваемости, посещаемости и т.д.;
4. Выгрузка отчетов по персональной успеваемости.

### Цель работы и задачи

**Цель работы:** спроектировать диаграмму классов, помогающей реализовать модуль корпоративно-информационной системы онлайн-школы *«Exam Masters».*

**Задачи:**

* + - 1. Выбрать средство реализации диаграммы классов;
      2. Согласно ТЗ определить необходимые в системе классы;
      3. Определить связи между классами и компонентами;
      4. Разработать диаграмму классов;
      5. Составить описание диаграммы классов.

## Реализация тестирования

### Вид и параметры тестирования

В ходе данного этапа разработки КИС было реализовано три вида тестирования: ручное тестирование, тестирование на соответствие требованиям, тестирование на наличие ошибок. Ниже подробно описаны виды тестирований.

**Ручное тестирование** – это метод тестирования программного обеспечения, при котором QA-инженер вручную проверяет работоспособность программы, следуя заранее определённому сценарию.

Ручное тестирование можно описать следующим алгоритмом действий:

1. **Анализ требований.** QA-инженер изучает требования к программному обеспечению, чтобы понять, какие функции и возможности должны быть протестированы.
2. **Разработка тестовых сценариев.** QA-инженер разрабатывает тестовые сценарии, которые описывают, как будет проводиться тестирование.
3. **Выполнение тестовых сценариев.** QA-инженер выполняет тестовые сценарии, проверяя работоспособность программного обеспечения.
4. **Фиксация результатов.** QA-инженер фиксирует результаты тестирования, отмечая обнаруженные ошибки и дефекты.
5. **Составление отчётов.** QA-инженер составляет отчёты о результатах тестирования, включая информацию об обнаруженных ошибках и дефектах.

Ниже в Табл. 24 представлены основные достоинства и недостатки ручного тестирования:

Табл. 24. Достоинства и недостатки ручного тестирования

|  |  |
| --- | --- |
| **Достоинства ручного тестирования** | **Недостатки ручного тестирования** |
| **Гибкость.** QA-инженер может адаптировать тестовые сценарии под конкретные требования и условия. | **Трудоёмкость.** Ручное тестирование требует больших затрат времени и усилий. |
| **Детальное тестирование.** QA-инженер может проверить работоспособность программы на более глубоком уровне. | **Вероятность ошибок.** QA-инженер может допустить ошибку при выполнении тестовых сценариев. |
| **Обнаружение нестандартных ситуаций.** QA-инженер может обнаружить ошибки и дефекты, которые не были предусмотрены тестовыми сценариями. |

**Тестирование на соответствие требованиям** – это процесс проверки того, насколько продукт или система соответствуют определённым требованиям. Тестирование на соответствие требованиям является одним из основных видов тестирования в процессе разработки программного обеспечения.

Тестирование на соответствие требованиям может проводиться на различных этапах разработки, включая этап проектирования, разработки и внедрения. Оно помогает обеспечить качество продукта или системы и соответствие их ожиданиям пользователей. Этот вид тестирования может быть выполнен вручную или с использованием автоматизированных инструментов. В зависимости от сложности продукта или системы, тестирование на соответствие требованиям может потребовать значительных усилий и времени.

**Тестирование на наличие ошибок** – это процесс проверки программного обеспечения на предмет ошибок и дефектов. Оно является одним из основных этапов разработки программного обеспечения и направлено на обеспечение его качества. Тестирование на наличие ошибок может проводиться на различных уровнях: модульном, интеграционном, системном и приемочном.

Тестирование на наличие ошибок является важным этапом разработки программного обеспечения, который позволяет обеспечить его качество и надёжность. Оно помогает выявить ошибки и дефекты на ранних этапах разработки, когда их исправление обходится дешевле.

Данные виды тестирования были реализованы разработчиками проекта для выявления ошибок системы, а также несоответствия первоначальному ТЗ.

### Тестирование основного функционала

Авторизация.

Первоначально было выполнено тестирование формы «Авторизация». Ниже на Табл. 25, Табл. 26 представлен результат проведения тестирования для блоков авторизации ученика, преподавателя, а также упрощенной формы регистрации:

Табл. 25. Тест-кейс №1. Тестирование формы «Авторизация. Ученик»

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Запуск приложения. | Открывается форма авторизации. | Открывается форма авторизации. |
| 2 | Ввод логина | В поле «Логин» появляется строка символов. | В поле «Логин» появляется строка символов. |
| 3 | Ввод пароля | В поле «Пароль» появляется скрытая строка символов. | В поле «Пароль» появляется скрытая строка символов. |
| 4\* | Нажатие на кнопку «Показать пароль» | В поле «Логин» появляется строка введенных символов. | В поле «Логин» появляется строка введенных символов. |
| 5 | Нажатие на кнопку «Войти» | Открывается главная страница личного кабинета ученика. | Открывается главная страница личного кабинета ученика. |

Табл. 26. Тест-кейс №2. Тестирование формы «Авторизация. Преподаватель»

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Запуск приложения. | Открывается форма авторизации. | Открывается форма авторизации. |
| 2 | Ввод логина | В поле «Логин» появляется строка символов. | В поле «Логин» появляется строка символов. |
| 3 | Ввод пароля | В поле «Пароль» появляется скрытая строка символов. | В поле «Пароль» появляется скрытая строка символов. |
| 4\* | Нажатие на кнопку «Показать пароль» | В поле «Логин» появляется строка введенных символов. | В поле «Логин» появляется строка введенных символов. |
| 5 | Нажатие на кнопку «Войти» | Открывается главная страница личного кабинета преподавателя. | Открывается главная страница личного кабинета преподавателя. |

***Дополнительная информация по тест-кейсам №1 и №2:***

1. Для ролей пользователей – «Преподаватель», «Ученик», «Администратор» авторизация проходит успешно.
2. Если введен логин пользователя, которого нет в хранилище данных, система выдает сообщение: «Пользователь не найден» и очищает заполненные поля.
3. Если пароль не подходит, при условии, что логин пользователя есть в хранилище, система выдает сообщение: «Введен неправильный пароль», и очищает поля ввода логина и пароля.
4. Если авторизоваться после неудачной попытки с верными данными, авторизация проходит успешно.

Табл. 27. Тест-кейс №3. Тестирование формы упрощенной регистрации

|  |  |  |  |
| --- | --- | --- | --- |
| **Предусловие выполнения тест-кейса №3.** Вводимый логин должен присутствовать в хранилище данных. В данном случае, под логином подразумевается адрес электронной почты, к которой привязывается личный кабинет пользователя. | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Выбор роли пользователя: «Ученик» / «Преподаватель» | Отмечается выбранная кнопка. | Отмечается выбранная кнопка. |
| 2 | Ввод логина | В поле «Логин» появляется строка символов. | В поле «Логин» появляется строка символов. |
| 3 | Ввод пароля | В поле «Пароль» появляется скрытая строка символов. | В поле «Пароль» появляется скрытая строка символов. |
| 4 | Повторный ввод пароля | В поле «Повторный ввод пароля» появляется скрытая строка символов. | В поле «Повторный ввод пароля» появляется скрытая строка символов. |
| 5\* | Нажатие на кнопку «Показать пароль» | Открывается главная страница личного кабинета преподавателя. | Открывается главная страница личного кабинета преподавателя. |
| 6 | Нажатие на кнопку «Войти» | Пароль успешно добавлен в систему. Система уведомляет об этом пользователя. | Пароль успешно добавлен в систему. Система уведомляет об этом пользователя. |

***Дополнительная информация по тест-кейсу №3:***

1. Для ролей пользователей – «Преподаватель», «Ученик» – упрощенная регистрация проходит успешно.
2. В случае, если пользователь не выбрал роль, система уведомляет об этом пользователя.
3. В случае, если пользователь ввел не существующий в хранилище логин, система уведомляет его об это и очищает поле.
4. В случае, если поля «Пароль» и «Проверка пароля» не совпадают, система уведомляет пользователя об этом.

Функционал ученика.

Для начала были протестированы основные функции главной страницы личного кабинета студента. Ниже в Табл. 28 описан тест-кейс для проверки работы функционала:

Табл. 28. Тест-кейс №4. Тестирование запуска функций отображения расписания и долгов

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №3.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Ученик». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1а | Нажатие на кнопку «Просмотр расписания» | Открывается форма «Расписание занятий». | Открывается форма «Расписание занятий». |
| 1б | Нажатие на кнопку «Информация о долгах» | Открывается форма «Информация о долгах». | Открывается форма «Информация о долгах». |

Далее была проведена проверка корректной работы формы «Расписание занятий». Тест-кейс и его результаты представлены в Табл. 29:

Табл. 29.Тест-кейс №5. Тестирование функционала формы «Расписание занятий»

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №5.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Ученик» и после перешел на форму «Расписание занятий». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1\* | Выбор периода отображения: установление нижней и верхней границы сортировки | В полях появляются нужные даты для сортировки занятий. | В полях появляются нужные даты для сортировки занятий. |
| 2 | Нажатие на кнопку «Вывести расписание» | В таблице появляются данные о занятиях. | В таблице появляются данные о занятиях. |
| 3 | Нажатие на кнопку «Экспортировать в Excel» | Открывается диалоговое окно для сохранения файла. | Открывается диалоговое окно для сохранения файла. |
| 4 | Выбор названия файла и места его сохранения в диалоговом окне. Нажатие кнопка «Сохранить». | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. |

Такая же проверка была проведена для формы «Долги по учебе». Тест-кейс и его результаты представлены в Табл. 30:

Табл. 30. Тест-кейс №6. Тестирование функционала формы «Долги по учебе»

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №6.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Ученик» и после перешел на форму «Долги по учебе». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Нажатие на кнопку «Вывести данные» | В таблице появляются данные о долгах по учебе. | В таблице появляются данные о долгах по учебе. |
| 2 | Нажатие на кнопку «Экспортировать в Excel» | Открывается диалоговое окно для сохранения файла. | Открывается диалоговое окно для сохранения файла. |
| 3 | Выбор названия файла и места его сохранения в диалоговом окне. Нажатие кнопка «Сохранить». | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. |

Функционал преподавателя

Для начала были протестированы основные функции главной страницы личного кабинета преподавателя. Ниже в Табл. 31 описан тест-кейс для проверки работы функционала:

Табл. 31. Тест-кейс №7. Тестирование запуска функций главной страницы личного кабинета преподавателя

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №7.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Преподаватель». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1а | Нажатие на кнопку «Просмотр расписания» | Открывается форма «Расписание преподавателя». | Открывается форма «Расписание преподавателя». |
| 1б | Нажатие на кнопку «Информация о учениках» | Открывается форма «Информация о учениках». | Открывается форма «Информация о учениках». |
| 1в | Нажатие на кнопку «Формирование отчетов» | Открывается форма «Сводная статистика по ученикам». | Открывается форма «Сводная статистика по ученикам». |

Далее был протестирован функционал личного кабинета преподавателя. Для начала на грамотность работы была проверена форма «Расписание преподавателя». Ее тестирование и результаты представлены ниже на Табл. 32:

Табл. 32. Тест-кейс №8. Тестирование функционала формы «Расписание преподавателя»

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №8.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Преподаватель» и после перешел на форму «Расписание преподавателя» нажатием кнопки «Просмотр расписания». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Выбор сортировки: «Для всех групп» / «Для отдельной группы» | Загорается кнопка соответствующего списка. | Загорается кнопка соответствующего списка. |
| 2\* | В случае, если нужно сформировать данные по отдельной группе, нажатие на кнопку «Показать расписание группы». Выбрать группу. | Появляется список групп. Выбранная группа автоматически записывается в поле. | Появляется список групп. Выбранная группа автоматически записывается в поле. |
| 3 | Выбор периода отображения: установление нижней и верхней границы сортировки | В полях появляются нужные даты для сортировки занятий. | В полях появляются нужные даты для сортировки занятий. |
| 4\* | Нажатие кнопку «Сбросить сортировку» | Сортировка по группе сбрасывается, поле для группы очищается. | Сортировка по группе сбрасывается, поле для группы очищается. |
| 5 | Нажатие на кнопку «Вывести расписание» | В таблице появляются данные о занятиях. | В таблице появляются данные о занятиях. |
| 6 | Нажатие на кнопку «Экспортировать в Excel» | Открывается диалоговое окно для сохранения файла. | Открывается диалоговое окно для сохранения файла. |
| 7 | Выбор названия файла и места его сохранения в диалоговом окне. Нажатие кнопка «Сохранить». | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. |

***Дополнительная информация по тест-кейсу №8***

1. В случае, если преподаватель нажимает кнопку «Экспортировать в Excel» без вывода расписания в форме, система сохраняет пустой файл.
2. В случае, если пользователь введет неправильный порядок дат (сначала большую, затем меньшую), система не выдает ошибку, однако не формирует расписание и не уведомляет пользователя о ошибочном вводу данных.
3. В случае, если пользователь не выбирает группы, в которых для которых нужно сформировать расписание, система уведомляет пользователя о некорректном вводе данных и просит выбрать формат сортировки.

Была также проверена грамотность работы функции «Контактная информация учеников» личного кабинета преподавателя. Тестирование это функции представлено ниже в Табл. 33:

Табл. 33. Тест-кейс №9. Тестирование функционала формы «Информация о учениках»

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №9.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Преподаватель» и после перешел на форму «Информация об учениках» нажатием одноименной кнопки. | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Просмотр списка доступных групп | Появляется падающий список групп. | Появляется падающий список групп. |
| 2 | Выбор группу. | Появляется список групп. Выбранная группа автоматически записывается в поле. | Появляется список групп. Выбранная группа автоматически записывается в поле. |
| 3\* | Нажатие кнопку «Сбросить сортировку» | Сортировка по группе сбрасывается, поле для группы очищается. | Сортировка по группе сбрасывается, поле для группы очищается. |
| 4 | Нажатие на кнопку «Вывести данные» | В таблице появляются данные о учениках. | В таблице появляются данные о учениках. |
| 5 | Нажатие на кнопку «Экспортировать в Excel» | Открывается диалоговое окно для сохранения файла. | Открывается диалоговое окно для сохранения файла. |
| 6 | Выбор названия файла и места его сохранения в диалоговом окне. Нажатие кнопка «Сохранить». | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. |

***Дополнительная информация по тест-кейсу №9***

1. В случае, если пользователь не выберет группу из падающего списка, система выведет уведомление о том, что без наименования группы данные не будут сформированы.
2. В случае, если пользователь нажмет кнопку «Экспортировать в Excel» без вывода данных, система сохранит пустой файл.

Далее была проверена последняя функция личного кабинета преподавателя – «Формирование отчетов». Для начала был протестирован раздел «Сводный отчет по группе» на наличие ошибок. Тест-кейс представлен ниже на Табл. 34:

Табл. 34. Тест-кейс №10. Тестирование функционала формы «Формирование отчетов. Сводный отчет по группе»

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №10.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Преподаватель» и после перешел на форму «Формирование отчетов» нажатием одноименной кнопки. | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Просмотр списка доступных групп | Появляется падающий список групп. | Появляется падающий список групп. |
| 2 | Выбор группу. | Появляется список групп. Выбранная группа автоматически записывается в поле. | Появляется список групп. Выбранная группа автоматически записывается в поле. |
| 4 | Нажатие на кнопку «Сформировать статистику» | В таблице появляются данные о показателях учеников. | В таблице появляются данные о показателях учеников. |
| 5 | Нажатие на кнопку «Экспортировать в Excel» | Открывается диалоговое окно для сохранения файла. | Открывается диалоговое окно для сохранения файла. |
| 6 | Выбор названия файла и места его сохранения в диалоговом окне. Нажатие кнопка «Сохранить». | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. |

***Дополнительная информация по тест-кейсу №8***

1. В случае, если пользователь не выберет группу из падающего списка, система выведет уведомление о том, что без наименования группы данные не будут сформированы.
2. В случае, если пользователь нажмет кнопку «Экспортировать в Excel» без вывода данных, система сохранит пустой файл.
3. В случае, если пользователь ввел в поле «Посещаемость» или «ДЗ» строковый формат данных или отрицательное значение, система выдает исключение.

Далее была протестирована функция «Экспорт статистики по посещаемости занятий» на наличие ошибок. Ее результат представлен ниже на Табл. 35:

Табл. 35. Тест-кейс №11. Тестирование функционала формы «Формирование отчетов. Экспорт статистики по посещаемости занятий»

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №11.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Преподаватель» и после перешел на форму «Формирование отчетов. Экспорт статистик» нажатием одноименной кнопки. | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Просмотр списка доступных групп | Появляется падающий список групп. | Появляется падающий список групп. |
| 2 | Выбор группу. | Появляется список групп. Выбранная группа автоматически записывается в поле. | Появляется список групп. Выбранная группа автоматически записывается в поле. |
| 4 | Нажатие на кнопку «Экспорт статистики по посещаемости занятий» | Открывается диалоговое окно для сохранения файла. | Открывается диалоговое окно для сохранения файла. |
| 5 | Выбор названия файла и места его сохранения в диалоговом окне. Нажатие кнопка «Сохранить». | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. |

Далее была протестирована функция «Экспорт статистики по ДЗ учеников» на наличие ошибок. Ее результат представлен ниже на Табл. 36:

Табл. 36. Тест-кейс №12. Тестирование функционала формы «Формирование отчетов. Экспорт статистики по ДЗ учеников»

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №12.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Преподаватель» и после перешел на форму «Формирование отчетов. Экспорт статистик» нажатием одноименной кнопки. | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Просмотр списка доступных групп | Появляется падающий список групп. | Появляется падающий список групп. |
| 2 | Выбор группу. | Появляется список групп. Выбранная группа автоматически записывается в поле. | Появляется список групп. Выбранная группа автоматически записывается в поле. |
| 4 | Нажатие на кнопку «Экспорт статистики по ДЗ учеников» | Открывается диалоговое окно для сохранения файла. | Открывается диалоговое окно для сохранения файла. |
| 5 | Выбор названия файла и места его сохранения в диалоговом окне. Нажатие кнопка «Сохранить». | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. |

***Дополнительная информация по тест-кейсу №11 и №12:***

В случае, если пользователь не выберет группу из падающего списка, система выведет уведомление о том, что без наименования группы.

Функционал администратора

Для начала были протестированы основные функции главной страницы личного кабинета администратора. Ниже в Табл. 37 описан тест-кейс для проверки работы функционала:

Табл. 37. Тест-кейс №13. Тестирование запуска функций главной страницы личного кабинета администратора

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №13.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Администратор». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1а | Нажатие на кнопку «Добавление новых данных» | Открывается форма «Добавление новых данных». | Открывается форма «Добавление новых данных». |
| 1б | Нажатие на кнопку «Удаление данных» | Открывается форма «Удаление данных». | Открывается форма «Удаление данных». |
| 1в | Нажатие на кнопку «Отчетные статистики» | Открывается форма «Отчетные статистики». | Открывается форма «Отчетные статистики» |

Далее функционал каждой формы был проверен по отдельности. Для начала была протестирована на наличие ошибок форма «Добавление новых данных». Результаты представлены ниже на Табл. 38 – Табл. 43:

Табл. 38. Тест-кейс №14. Тестирование функции добавления преподавателя

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №14.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Добавление новых данных. Пользователи». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «ФИО» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 2 | Ввод данных в поле «Электронная почта» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 3 | Ввод данных в поле «Номер телефона» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 4 | Нажатие на кнопку «Добавить данные» | Система добавляет преподавателя в систему и хранилище данных, а также выводит сообщение о успешном добавлении. | Система добавляет преподавателя в систему и хранилище данных, а также выводит сообщение о успешном добавлении. |

Табл. 39. Тест-кейс №15. Тестирование функции добавления ученика

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №15.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Добавление новых данных. Пользователи». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «ФИО» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 2 | Ввод данных в поле «Электронная почта» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 3 | Ввод данных в поле «Номер телефона» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 4 | Ввод данных в поле «Название группы» | Появляется падающий список, из которого наименование группы записывается в поле. | Появляется падающий список, из которого наименование группы записывается в поле. |
| 5 | Нажатие на кнопку «Добавить данные» | Система добавляет преподавателя в систему и хранилище данных | Система добавляет преподавателя в систему и хранилище данных |

Табл. 40. Тест-кейс №16. Тестирование функции добавления группы

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №16.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Добавление новых данных. Пользователи». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «Наименование группы» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 2 | Ввод данных в поле «Старт занятий» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 3 | Ввод данных в поле «Окончание занятий» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 4 | Ввод данных в поле «ФИО преподавателя» | Появляется падающий список, из которого ФИО преподавателя записывается в поле. | Появляется падающий список, из которого ФИО преподавателя записывается в поле. |
| 5 | Нажатие на кнопку «Добавить данные» | Система добавляет преподавателя в систему и хранилище данных, а также выводит сообщение о успешном добавлении. | Система добавляет преподавателя в систему и хранилище данных, а также выводит сообщение о успешном добавлении. |

Табл. 41. Тест-кейс №17. Тестирование функции добавления курса

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №17.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Добавление новых данных. Курсы». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «Наименование группы» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 2 | Ввод данных в поле «Старт занятий» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 3 | Ввод данных в поле «Окончание занятий» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 4 | Ввод данных в поле «ФИО преподавателя» | Появляется падающий список, из которого ФИО преподавателя записывается в поле. | Появляется падающий список, из которого ФИО преподавателя записывается в поле. |
| 5 | Нажатие на кнопку «Добавить данные» | Система добавляет преподавателя в систему и хранилище данных, а также выводит сообщение о успешном добавлении. | Система добавляет преподавателя в систему и хранилище данных, а также выводит сообщение о успешном добавлении. |

Табл. 42. Тест-кейс №18. Тестирование функции добавления урока

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №18.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Добавление новых данных. Курсы». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «Название урока» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 2 | Ввод данных в поле «Ссылка на занятие» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 3 | Ввод данных в поле «Ссылка на ДЗ» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 4 | Ввод данных в поле «Наименование курса» | Появляется падающий список, из которого наименование курса записывается в поле. | Появляется падающий список, из которого наименование курса записывается в поле. |
| 5 | Ввод данных в поле «Дата занятия» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 6 | Ввод данных в поле «ФИО преподавателя» | Появляется падающий список, из которого ФИО преподавателя записывается в поле. | Появляется падающий список, из которого ФИО преподавателя записывается в поле. |
| 7 | Нажатие на кнопку «Добавить данные» | Система добавляет преподавателя в систему и хранилище данных, а также выводит сообщение о успешном добавлении. | Система добавляет преподавателя в систему и хранилище данных, а также выводит сообщение о успешном добавлении. |

Табл. 43. Тест-кейс №19. Тестирование функции добавления контрольной точки

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №19.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Добавление новых данных. Курсы». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «Наименование теста» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 2 | Ввод данных в поле «Тип контрольной точки» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 3 | Ввод данных в поле «Ссылка на ДЗ» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 4 | Ввод данных в поле «Время выполнения» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 5 | Ввод данных в поле «Дата занятия» | В поле появляется строка с данными. | В поле появляется строка с данными. |
| 6 | Ввод данных в поле «Наименование курса» | Появляется падающий список, из которого наименование курса записывается в поле. | Появляется падающий список, из которого наименование курса записывается в поле. |
| 7 | Нажатие на кнопку «Добавить данные» | Система добавляет преподавателя в систему и хранилище данных, а также выводит сообщение о успешном добавлении. | Система добавляет преподавателя в систему и хранилище данных, а также выводит сообщение о успешном добавлении. |

***Дополнительная информация по тест-кейсам №14–19:***

1. В случае, если пользователь вводит не все данные, система уведомляет пользователя об этом и просит заполнить оставшиеся поля.
2. В системе не предусмотрена проверка на тип вводимых данных. Так, к примеру, в поле «Номер телефона» можно ввести строковый формат данных.

Далее была протестирована форма «Удаление данных». Ниже на Табл. 44 – Табл. 49 представлены результаты тестирования на наличие ошибок основных функций удаления:

Табл. 44. Тест-кейс №20. Тестирование функции удаления преподавателя

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №20.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Удаление данных». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «Фио преподавателя» | Появляется падающий список, из которого ФИО преподавателя записывается в поле. | Появляется падающий список, из которого ФИО преподавателя записывается в поле. |
| 2 | Нажатие на кнопку «Удалить данные» | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. |

Табл. 45. Тест-кейс №21. Тестирование функции удаления ученика

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №21.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Удаление данных». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «Фио ученика» | Появляется падающий список, из которого ФИО ученика записывается в поле. | Появляется падающий список, из которого ФИО ученика записывается в поле. |
| 2 | Нажатие на кнопку «Удалить данные» | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. |

Табл. 46. Тест-кейс №22. Тестирование функции удаления группы

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №22.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Удаление данных». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «Наименование группы» | Появляется падающий список, из которого наименование группы записывается в поле. | Появляется падающий список, из которого наименование группы записывается в поле. |
| 2 | Нажатие на кнопку «Удалить данные» | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. |

Табл. 47. Тест-кейс №23. Тестирование функции удаления группы

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №23.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Удаление данных». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «Наименование курса» | Появляется падающий список, из которого наименование курса записывается в поле. | Появляется падающий список, из которого наименование курса записывается в поле. |
| 2 | Нажатие на кнопку «Удалить данные» | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. |

Табл. 48. Тест-кейс №24. Тестирование функции удаления занятия

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №24.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Удаление данных». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «Наименование курса» | Появляется падающий список, из которого наименование курса записывается в поле. | Появляется падающий список, из которого наименование курса записывается в поле. |
| 2 | Ввод данных в поле «Наименование урока» | Появляется падающий список, из которого наименование урока записывается в поле. | Появляется падающий список, из которого наименование урока записывается в поле. |
| 3 | Нажатие на кнопку «Удалить данные» | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. |

Табл. 49. Тест-кейс №25. Тестирование функции удаления точки контроля

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №25.*** Пользователь прошел форму авторизации, зашел в свой личный кабинет под ролью «Администратор» и перешел на форму «Удаление данных». | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Ввод данных в поле «Наименование курса» | Появляется падающий список, из которого наименование курса записывается в поле. | Появляется падающий список, из которого наименование курса записывается в поле. |
| 2 | Ввод данных в поле «Наименование точки контроля» | Появляется падающий список, из которого ТК записывается в поле. | Появляется падающий список, из которого ТК записывается в поле. |
| 3 | Нажатие на кнопку «Удалить данные» | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. | Система удаляет преподавателя из хранилища данных, а также выводит сообщение о успешном удалении. |

***Дополнительная информация по тест-кейсам №20–25:***

1. В случае, если пользователь вводит не все данные, система уведомляет пользователя об этом и просит заполнить оставшиеся поля.
2. В системе не предусмотрена функция уточнения у пользователя, действительно ли он хочет удалить какие-либо данные. Таким образом, важные данные могут быть утеряны в случае неаккуратной эксплуатации системы.
3. В системе не предусмотрена обработка исключений, вызванных сильной связанностью данных.

Далее была протестирована форма «Отчетные статистики». Эта форма делится на три раздела: «Преподаватели», «Ученики», «Группы». На Табл. 50 представлено тестирование раздела «Преподаватели»:

Табл. 50. Тест-кейс №26. Тестирование работы раздела «Отчетные статистики. Преподаватели»

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №26.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Администратор» и после перешел на форму «Отчетные статистики» нажатием одноименной кнопки. | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1\* | Выбор сортировки: Установление верхней границы показателей посещаемости и ДЗ | Загорается кнопка соответствующего списка. | Загорается кнопка соответствующего списка. |
| 2 | Нажатие на кнопку «Сформировать статистику» | В таблице появляются данные о показателях преподавателей. | В таблице появляются данные о показателях преподавателей. |
| 3 | Нажатие на кнопку «Экспортировать в Excel» | Открывается диалоговое окно для сохранения файла. | Открывается диалоговое окно для сохранения файла. |
| 4 | Выбор названия файла и места его сохранения в диалоговом окне. Нажатие кнопка «Сохранить». | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. |

***Дополнительная информация по тест-кейсу №26***

1. В случае, если пользователь не установит верхние границы для отображения показателей преподавателей, автоматически система выведет всех преподавателей.
2. В случае, если пользователь нажмет кнопку «Экспортировать в Excel» без вывода данных, система сохранит пустой файл.
3. В случае, если пользователь ввел в поле «Посещаемость» или «ДЗ» строковый формат данных или отрицательное значение, система выдает исключение.

Далее был протестирован раздел «Ученики». Результаты тестирования приведены ниже на Табл. 51:

Табл. 51. Тест-кейс №27. Тестирование работы раздела «Отчетные статистики. Ученики»

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №27.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Администратор» и после перешел на форму «Отчетные статистики» нажатием одноименной кнопки. | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1\* | Выбор сортировки: Установление верхней границы показателей посещаемости и ДЗ | Загорается кнопка соответствующего списка. | Загорается кнопка соответствующего списка. |
| 2\* | Ввод данных в поле «ФИО преподавателя» | Появляется падающий список, из которого ФИО преподавателя записывается в поле. | Появляется падающий список, из которого ФИО преподавателя записывается в поле. |
| 3\* | Ввод данных в поле «Группа» | Появляется падающий список, из которого группа записывается в поле. | Появляется падающий список, из которого группа записывается в поле. |
| 4 | Нажатие на кнопку «Сформировать статистику» | В таблице появляются данные о показателях учеников. | В таблице появляются данные о показателях учеников. |
| 5 | Нажатие на кнопку «Экспортировать в Excel» | Открывается диалоговое окно для сохранения файла. | Открывается диалоговое окно для сохранения файла. |
| 6 | Выбор названия файла и места его сохранения в диалоговом окне. Нажатие кнопка «Сохранить». | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. |

***Дополнительная информация по тест-кейсу №26***

1. В случае, если пользователь не установит верхние границы для отображения показателей учеников, автоматически система выведет всех учеников.
2. В случае, если пользователь ввел в поле «Посещаемость» или «ДЗ» строковый формат данных или отрицательное значение, система выдает исключение.
3. В случае, если не выбрать преподавателя и группу в падающих списках, будут выведены все ученики.
4. В случае, если выбрать только преподавателя, будет выведен список учеников именно этого преподавателя.
5. В случае, если выбрать только группу, то будет выведен список учеников, принадлежащих этой группе.
6. В случае, если выбран преподаватель, в падающем списке групп будут отображены только те, в которых этот преподаватель ведет занятия.
7. В случае, если пользователь нажмет кнопку «Экспортировать в Excel» без вывода данных, система сохранит пустой файл.

Далее был протестирован раздел «Группы». Результаты тестирования приведены ниже на Табл. 52:

Табл. 52. Тест-кейс №28. Тестирование работы раздела «Отчетные статистики. Группы»

|  |  |  |  |
| --- | --- | --- | --- |
| ***Предусловие выполнения тест-кейса №28.*** Пользователь прошел форму авторизации и зашел в свой личный кабинет под ролью «Администратор» и после перешел на форму «Отчетные статистики» нажатием одноименной кнопки. | | | |
| **№** | **Шаг** | **Ожидаемый результат** | **Фактический результат** |
| 1\* | Выбор сортировки: Установление верхней границы показателей посещаемости и ДЗ | Загорается кнопка соответствующего списка. | Загорается кнопка соответствующего списка. |
| 2\* | Ввод данных в поле «ФИО преподавателя» | Появляется падающий список, из которого ФИО преподавателя записывается в поле. | Появляется падающий список, из которого ФИО преподавателя записывается в поле. |
| 4 | Нажатие на кнопку «Сформировать статистику» | В таблице появляются данные о показателях учеников. | В таблице появляются данные о показателях учеников. |
| 5 | Нажатие на кнопку «Экспортировать в Excel» | Открывается диалоговое окно для сохранения файла. | Открывается диалоговое окно для сохранения файла. |
| 6 | Выбор названия файла и места его сохранения в диалоговом окне. Нажатие кнопка «Сохранить». | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. | Файл сохраняется. Система уведомляет пользователя о успешном сохранении файла. |

***Дополнительная информация по тест-кейсу №27***

1. В случае, если пользователь не установит верхние границы для отображения показателей групп, автоматически система выведет всех групп.
2. В случае, если не выбрать преподавателя, будут выведены все группы.
3. В случае, если пользователь нажмет кнопку «Экспортировать в Excel» без вывода данных, система сохранит пустой файл.
4. В случае, если пользователь ввел в поле «Посещаемость» или «ДЗ» строковый формат данных или отрицательное значение, система выдает исключение.

## Задачи на доработку

Форма «Преподаватель. Расписание преподавателя»

1. В случае, если преподаватель нажимает кнопку «Экспортировать в Excel» без вывода расписания, следует выводить сообщение о том, что данные не сформированы и не могут быть выведены.
2. В случае, если пользователь введет неправильный порядок дат (сначала большую, затем меньшую), следует вывести сообщение о том, что порядок дат введен неверно.

Форма «Преподаватель. Информация о учениках»

1. В случае, если пользователь нажмет кнопку «Экспортировать в Excel» без вывода данных, следует выводить сообщение о том, что данные не сформированы и не могут быть выведены.

Форма «Преподаватель. Формирование отчетов. Сводный отчет по группе»

1. В случае, если пользователь нажмет кнопку «Экспортировать в Excel» без вывода данных, следует выводить сообщение о том, что данные не сформированы и не могут быть выведены.
2. В случае, если в поля «Посещаемость» и «ДЗ» введено отрицательное значение или строковый формат данных, система не должна выдавать исключение, а отправлять сообщение пользователю о том, что введен неправильный формат данных.

Форма «Администратор. Добавление новых данных»

1. Добавить проверку правильности ввода номера телефона для преподавателя и ученика.
2. Добавить проверку правильности ввода электронной почты для преподавателя и ученика.
3. Добавить проверку правильности ввода дат начала и окончания занятий для группы.
4. Добавить проверку правильности ввода стоимости курса и количества уроков для курса.

Форма «Администратор. Удаление данных»

1. В случае, если пользователь нажимает кнопку «Удалить данные», следует ввести проверку, действительно ли нужно удалить объект.
2. В системе не предусмотрена обработка исключений, вызванных сильной связанностью данных. Добавить исключения при удалении каждого объекта в соответствии с базой данных и связями таблиц в ней.

Форма «Администратор. Отчетные статистики. Ученики»

1. В случае, если в поля «Посещаемость» и «ДЗ» введено отрицательное значение или строковый формат данных, система не должна выдавать исключение, а отправлять сообщение пользователю о том, что введен неправильный формат данных.
2. В случае, если пользователь нажмет кнопку «Экспортировать в Excel» без вывода данных, следует выводить сообщение о том, что данные не сформированы и не могут быть выведены.

Форма «Администратор. Отчетные статистики. Преподаватели»

1. В случае, если в поля «Посещаемость» и «ДЗ» введено отрицательное значение или строковый формат данных, система не должна выдавать исключение, а отправлять сообщение пользователю о том, что введен неправильный формат данных.
2. В случае, если пользователь нажмет кнопку «Экспортировать в Excel» без вывода данных, следует выводить сообщение о том, что данные не сформированы и не могут быть выведены.

Форма «Администратор. Отчетные статистики. Группы»

1. В случае, если в поля «Посещаемость» и «ДЗ» введено отрицательное значение или строковый формат данных, система не должна выдавать исключение, а отправлять сообщение пользователю о том, что введен неправильный формат данных.
2. В случае, если пользователь нажмет кнопку «Экспортировать в Excel» без вывода данных, следует выводить сообщение о том, что данные не сформированы и не могут быть выведены.

# Выводы

В ходе тестирования корпоративной информационной системы онлайн-модуля сети школ *«Exam Masters»* было выявлено 17 ошибок разной тяжести, которые были описаны в разделе «Задачи на доработку». Несмотря на это, система покрывает большее количество функций и нестандартных сценариев работы, которые удовлетворяет требованиям из ТЗ.

Ошибки, которые были найдены в ходе проведения тестирования, будут исправлены для дальнейшего развития проекта.

# Приложение

## Классы форм

### Autorization

using ExamMasters.Admin;

using ExamMasters.Student;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics.Eventing.Reader;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ExamMasters

{

public partial class Autorization : Form

{

public string Student\_login = "";

public string Student\_password = "";

public string Teacher\_login = "";

public string Teacher\_password = "";

public string login = "";

public string password = "";

public string passwordCheck = "";

bool clck\_check1 = true;

bool clck\_check2 = true;

bool clck\_check3 = true;

public Autorization()

{

InitializeComponent();

}

private void label11\_Click(object sender, EventArgs e)

{

}

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

{

// НЕ ВВЕДЕН ЛОГИН ИЛИ ПАРОЛЬ

if (tb\_studentLogin.Text == "" || tb\_studentPassword.Text == "")

{

string messageBoxText = "Введите логин и пароль.";

string caption = "Ошибка ввода данных";

MessageBox.Show(messageBoxText, caption);

}

else

{

int student\_ID = DB\_Do.GetStudent\_ID\_ByEmail(Student\_login);

StudentClass student = new StudentClass(student\_ID);

student.Email = Student\_login;

// Проверка на наличие пользователя в БД с указанным ID

if (student\_ID == -1)

{

string messageBoxText = "Пользователь не найден.";

string caption = "Ошибка ввода данных";

MessageBox.Show(messageBoxText, caption);

tb\_studentPassword.Clear();

tb\_studentLogin.Clear();

}

student.Password = DB\_Do.GetStudent\_Password\_ByID(student\_ID);

// НЕПРАВИЛЬНЫЙ ЛОИГН ИЛИ ПАРОЛЬ

if (student.Password != Student\_password)

{

string messageBoxText = "Введен неправильный пароль";

string caption = "Ошибка ввода данных";

MessageBox.Show(messageBoxText, caption);

tb\_studentPassword.Clear();

tb\_studentLogin.Clear();

}

// Вход в систему при корректном наборе данных

else if (student.Password == Student\_password && Student\_password != "" && Student\_login != "")

{

//Student.Student\_Main student\_Main = new Student();

Student.Student\_Main student\_Main = new Student.Student\_Main(student);

student\_Main.Show();

Hide();

}

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

Student\_login = tb\_studentLogin.Text;

}

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

{

Student\_password = tb\_studentPassword.Text;

}

private void tabPage1\_Click(object sender, EventArgs e)

{

}

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

{

// НЕ ВВЕДЕН ЛОГИН ИЛИ ПАРОЛЬ

if (tb\_teacherLogin.Text == "" || tb\_teacherPassword.Text == "")

{

string messageBoxText = "Введите логин и пароль.";

string caption = "Ошибка ввода данных";

MessageBox.Show(messageBoxText, caption);

}

else

{

int teacher\_ID = DB\_Do.GetTeacher\_ID\_ByEmail(Teacher\_login);

TeacherClass teacher = new TeacherClass(teacher\_ID);

teacher.Email = Teacher\_login;

// Проверка на наличие пользователя в БД с указанным ID

if (teacher.Id == -1)

{

string messageBoxText = "Пользователь не найден.";

string caption = "Ошибка ввода данных";

MessageBox.Show(messageBoxText, caption);

tb\_teacherPassword.Clear();

tb\_teacherLogin.Clear();

}

teacher.Password = DB\_Do.GetTeacher\_Password\_ByID(teacher\_ID);

// НЕПРАВИЛЬНЫЙ ЛОИГН ИЛИ ПАРОЛЬ

if (teacher.Password != Teacher\_password)

{

string messageBoxText = "Введен неправильный пароль";

string caption = "Ошибка ввода данных";

MessageBox.Show(messageBoxText, caption);

tb\_teacherPassword.Clear();

tb\_teacherLogin.Clear();

}

// ВХОД В ЛК АДМИНИСТРАТОРА

else if (teacher.Email == "admin@exammasters.ru" && Teacher\_password == teacher.Password)

{

Admin\_Main adminMain = new Admin\_Main();

adminMain.Show();

Hide();

}

// Вход в систему при корректном наборе данных

else if (teacher.Password == Teacher\_password && Teacher\_password != "" && Teacher\_login != "")

{

Teacher.Teacher\_Main teacher\_Main = new Teacher.Teacher\_Main(teacher);

teacher\_Main.Show();

Hide();

}

}

}

private void button5\_Click(object sender, EventArgs e)

{

// Установление пароля без выбора роли пользователя

if (!rbt\_student.Checked && !rbt\_teacher.Checked)

{

string messageBoxText = "Выберете роль пользователя.";

string caption = "Ошибка ввода данных";

MessageBox.Show(messageBoxText, caption);

}

// Смена пароля для пользователя роли "Ученик"

else if (rbt\_student.Checked)

{

int id = DB\_Do.GetStudent\_ID\_ByEmail(login);

if (id == -1)

{

string messageBoxText = "Пользователь не найден.";

string caption = "Ошибка ввода данных";

MessageBox.Show(messageBoxText, caption);

tb\_login.Clear();

tb\_password.Clear();

tb\_passwordCheck.Clear();

}

else if (password == passwordCheck)

{

DB\_Do.SetStudentPassword(id, password);

string messageBoxText = "Пароль добавлен успешно.";

string caption = "Уведомление";

MessageBox.Show(messageBoxText, caption);

tb\_login.Clear();

tb\_password.Clear();

tb\_passwordCheck.Clear();

}

else

{

string messageBoxText = "Проверка пароля не пройдена.";

string caption = "Ошибка ввода данных";

MessageBox.Show(messageBoxText, caption);

tb\_password.Clear();

tb\_passwordCheck.Clear();

}

}

// Смена пароля для пользователя роли "Преподаватель"

else if (rbt\_teacher.Checked)

{

int id = DB\_Do.GetTeacher\_ID\_ByEmail(login);

if (id == -1)

{

string messageBoxText = "Пользователь не найден.";

string caption = "Ошибка ввода данных";

MessageBox.Show(messageBoxText, caption);

tb\_login.Clear();

tb\_password.Clear();

tb\_passwordCheck.Clear();

}

if (password == passwordCheck)

{

DB\_Do.SetTeacherPassword(id, password);

string messageBoxText = "Пароль добавлен успешно.";

string caption = "Уведомление";

MessageBox.Show(messageBoxText, caption);

tb\_login.Clear();

tb\_password.Clear();

tb\_passwordCheck.Clear();

}

else

{

string messageBoxText = "Проверка пароля не пройдена.";

string caption = "Ошибка ввода данных";

MessageBox.Show(messageBoxText, caption);

tb\_password.Clear();

tb\_passwordCheck.Clear();

}

}

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

Teacher\_login = tb\_teacherLogin.Text;

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

Teacher\_password = tb\_teacherPassword.Text;

}

private void tb\_login\_TextChanged(object sender, EventArgs e)

{

login = tb\_login.Text;

}

private void tb\_password\_TextChanged(object sender, EventArgs e)

{

password = tb\_password.Text;

}

private void tb\_passwordCheck\_TextChanged(object sender, EventArgs e)

{

passwordCheck = tb\_passwordCheck.Text;

}

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

{

if (!clck\_check1)

{

tb\_studentPassword.UseSystemPasswordChar = true;

clck\_check1 = true;

return;

}

if (clck\_check1)

{

tb\_studentPassword.UseSystemPasswordChar = false;

clck\_check1 = false;

return;

}

}

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

{

if (!clck\_check2)

{

tb\_teacherPassword.UseSystemPasswordChar = true;

clck\_check2 = true;

return;

}

if (clck\_check2)

{

tb\_teacherPassword.UseSystemPasswordChar = false;

clck\_check2 = false;

return;

}

}

private void button6\_Click(object sender, EventArgs e)

{

if (!clck\_check3)

{

tb\_password.UseSystemPasswordChar = true;

tb\_passwordCheck.UseSystemPasswordChar = true;

clck\_check3 = true;

return;

}

if (clck\_check3)

{

tb\_password.UseSystemPasswordChar = false;

tb\_passwordCheck.UseSystemPasswordChar = false;

clck\_check3 = false;

return;

}

}

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

{

}

}

}

Admin\_Main

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;

namespace ExamMasters.Admin

{

public partial class Admin\_Main : Form

{

public Admin\_Main()

{

InitializeComponent();

}

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

{

}

private void bt\_exit\_Click(object sender, EventArgs e)

{

Autorization autorization = new Autorization();

autorization.Show();

Hide();

}

private void bt\_addData\_Click(object sender, EventArgs e)

{

Admin\_AddData addData = new Admin\_AddData();

addData.Show();

Hide();

}

private void bt\_DeleteData\_Click(object sender, EventArgs e)

{

Admin\_DeleteData deleteData = new Admin\_DeleteData();

deleteData.Show();

Hide();

}

private void bt\_adminStatistics\_Click(object sender, EventArgs e)

{

Admin\_Statistics statistics = new Admin\_Statistics();

statistics.Show();

Hide();

}

}

}

Admin\_AddData

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 static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace ExamMasters.Admin

{

public partial class Admin\_AddData : Form

{

public Admin\_AddData()

{

InitializeComponent();

// Формирование списка групп

List<Group> groups = DB\_Do.GetGroups();

for (int i = 0; i < groups.Count; i++)

{

comboBox1.Items.Add(groups[i].Name);

}

// Формирование списка преподавателей

List<TeacherClass> teacher = DB\_Do.GetTeachers();

for (int i = 0; i < teacher.Count; i++)

{

cmbx\_groupTeacher.Items.Add(teacher[i].Name);

cmbx\_lessonTeacher.Items.Add(teacher[i].Name);

}

// Формирование списка курсов

List<Course> course = DB\_Do.GetCourses();

for (int i = 0; i < course.Count; i++)

{

tb\_lessonKursName.Items.Add(course[i].Name);

tb\_testKursName.Items.Add(course[i].Name);

}

}

string TeacherName = ""; string TeacherLogin = ""; string TeacherPhone = "";

string StudentName = ""; string StudentLogin = ""; string StudentPhone = ""; string StudentGroupName = "";

string GroupName = ""; string GroupStart = ""; string GroupFinish = ""; string GroupTeacherName = "";

string CourseName = ""; string CourseDescr = ""; string CourseCount = ""; string CourseLessonCount = "";

string LessonName = ""; string LessonVideo = ""; string LessonHometask = ""; string LessonCourse = ""; string LessonDate = ""; string LessonTeacher = "";

string TestName = ""; string TestType = ""; string TestTime = ""; string TestCourse = "";

private void groupBox6\_Enter(object sender, EventArgs e)

{

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void label3\_Click(object sender, EventArgs e)

{

}

private void Admin\_AddData\_Load(object sender, EventArgs e)

{

}

private void label22\_Click(object sender, EventArgs e)

{

}

private void label20\_Click(object sender, EventArgs e)

{

}

private void label21\_Click(object sender, EventArgs e)

{

}

private void button7\_Click(object sender, EventArgs e)

{

if (tn\_groupName.Text == "" || tn\_groupStart.Text == "" || tb\_groupFinish.Text == "" || cmbx\_groupTeacher.Text == "")

{

Message.Show("Ошибка ввода данных", "Введите данные в обязательные поля.");

return;

}

int teacher\_id = DB\_Do.GetTeacher\_ID\_ByFIO(GroupTeacherName);

DB\_Do.AddGroup(GroupName, GroupStart, GroupFinish, teacher\_id);

Message.Show("Уведомление", "Группа добавлена в базу.");

tn\_groupName.Clear();

tn\_groupStart.Clear();

tb\_groupFinish.Clear();

cmbx\_groupTeacher.Text = "";

}

private void comboBox6\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void label5\_Click(object sender, EventArgs e)

{

}

private void tb\_teacherName\_TextChanged(object sender, EventArgs e)

{

TeacherName = tb\_teacherName.Text;

}

private void tb\_teacherLogin\_TextChanged(object sender, EventArgs e)

{

TeacherLogin = tb\_teacherLogin.Text;

}

private void tb\_teacherPhone\_TextChanged(object sender, EventArgs e)

{

TeacherPhone = tb\_teacherPhone.Text;

}

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

{

Admin\_Main main = new Admin\_Main();

main.Show();

Close();

}

private void bt\_addTeacher\_Click(object sender, EventArgs e)

{

if(tb\_teacherName.Text =="" || tb\_teacherLogin.Text == "" || tb\_teacherLogin.Text == "")

{

Message.Show("Ошибка ввода данных", "Введите данные в обязательные поля.");

return;

}

DB\_Do.AddTeacher(TeacherName, TeacherLogin, TeacherPhone);

Message.Show("Уведомление", "Преподаватель добавлен в базу.");

tb\_teacherName.Clear();

tb\_teacherLogin.Clear();

tb\_teacherPhone.Clear();

}

private void tb\_studentName\_TextChanged(object sender, EventArgs e)

{

StudentName = tb\_studentName.Text;

}

private void tb\_lessonName\_TextChanged(object sender, EventArgs e)

{

LessonName = tb\_lessonName.Text;

}

private void tb\_studentLogin\_TextChanged(object sender, EventArgs e)

{

StudentLogin = tb\_studentLogin.Text;

}

private void tb\_studentPhone\_TextChanged(object sender, EventArgs e)

{

StudentPhone = tb\_studentPhone.Text;

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

StudentGroupName = comboBox1.SelectedItem.ToString();

}

private void bt\_addStudent\_Click(object sender, EventArgs e)

{

if (tb\_studentName.Text == "" || tb\_studentLogin.Text == "" || tb\_studentPhone.Text == "" || comboBox1.Text == "")

{

Message.Show("Ошибка ввода данных", "Введите данные в обязательные поля.");

return;

}

int group\_id = DB\_Do.GetGroup\_ID\_ByName(StudentGroupName);

DB\_Do.AddStudent(StudentName, StudentLogin, StudentPhone, group\_id);

Message.Show("Уведомление", "Ученик добавлен в базу.");

tb\_studentName.Clear();

tb\_studentLogin.Clear();

tb\_studentPhone.Clear();

comboBox1.Text = "";

}

private void tn\_groupName\_TextChanged(object sender, EventArgs e)

{

GroupName = tn\_groupName.Text;

}

private void tn\_groupStart\_TextChanged(object sender, EventArgs e)

{

GroupStart = tn\_groupStart.Text;

}

private void tb\_groupFinish\_TextChanged(object sender, EventArgs e)

{

GroupFinish = tb\_groupFinish.Text;

}

private void cmbx\_groupTeacher\_SelectedIndexChanged(object sender, EventArgs e)

{

GroupTeacherName = cmbx\_groupTeacher.SelectedItem.ToString();

}

private void tb\_kursName\_TextChanged(object sender, EventArgs e)

{

CourseName = tb\_kursName.Text;

}

private void tb\_kursDescr\_TextChanged(object sender, EventArgs e)

{

CourseDescr = tb\_kursDescr.Text;

}

private void tb\_kursCost\_TextChanged(object sender, EventArgs e)

{

CourseCount = tb\_kursCost.Text;

}

private void tb\_kursLessonCount\_TextChanged(object sender, EventArgs e)

{

CourseLessonCount = tb\_kursLessonCount.Text;

}

private void tb\_lessonVideo\_TextChanged(object sender, EventArgs e)

{

LessonVideo = tb\_lessonVideo.Text;

}

private void tb\_lessonHometask\_TextChanged(object sender, EventArgs e)

{

LessonHometask = tb\_lessonHometask.Text;

}

private void tb\_lessonKursName\_SelectedIndexChanged(object sender, EventArgs e)

{

LessonCourse = tb\_lessonKursName.Text;

}

private void bt\_addKurs\_Click(object sender, EventArgs e)

{

if (tb\_kursName.Text == "" || tb\_kursDescr.Text == "" || tb\_kursLessonCount.Text == "" || tb\_kursCost.Text == "")

{

Message.Show("Ошибка ввода данных", "Введите данные в обязательные поля.");

return;

}

DB\_Do.AddCourse(CourseName, CourseDescr, Convert.ToInt32(CourseCount), Convert.ToInt32(CourseLessonCount));

tb\_kursName.Clear();

tb\_kursDescr.Clear();

tb\_kursCost.Clear();

tb\_kursLessonCount.Clear();

Message.Show("Уведомление", "Курс успешно добавлен в базу.");

}

private void tb\_addLesson\_Click(object sender, EventArgs e)

{

if (tb\_lessonName.Text == "" || tb\_lessonVideo.Text == "" || tb\_lessonHometask.Text == "" || tb\_lessonDate.Text == "" || cmbx\_lessonTeacher.Text == "")

{

Message.Show("Ошибка","Введите данные в обязательные поля");

return;

}

int id\_kurs = DB\_Do.GetCourse\_ID\_ByName(LessonCourse);

int id\_teacher = DB\_Do.GetTeacher\_ID\_ByFIO(LessonTeacher);

DB\_Do.AddLesson(id\_kurs, LessonDate, LessonName, LessonVideo, LessonHometask, id\_teacher);

Message.Show("Уведомление", "Урок успешно добавлен в курс");

tb\_lessonName.Clear();

tb\_lessonDate.Clear();

tb\_lessonVideo.Clear();

tb\_lessonHometask.Clear();

tb\_lessonKursName.Text = "";

cmbx\_lessonTeacher.Text = "";

}

private void tb\_lessonDate\_TextChanged(object sender, EventArgs e)

{

LessonDate = tb\_lessonDate.Text;

}

private void cmbx\_lessonTeacher\_SelectedIndexChanged(object sender, EventArgs e)

{

LessonTeacher = cmbx\_lessonTeacher.Text;

}

private void tb\_addTest\_Click(object sender, EventArgs e)

{

if (tb\_testName.Text == "" || tb\_testTime.Text == "" || tb\_testType.Text == "" || tb\_testKursName.Text == "")

{

Message.Show("Ошибка", "Введите данные в обязательные поля");

return;

}

int id\_kurs = DB\_Do.GetCourse\_ID\_ByName(LessonCourse);

DB\_Do.AddTest(TestName, TestType, TestTime, id\_kurs);

Message.Show("Уведомление", "Тестирование успешно добавлено в курс");

tb\_testKursName.Text = "";

tb\_testName.Clear();

tb\_testType.Clear();

tb\_testTime.Clear();

}

private void tb\_testKursName\_SelectedIndexChanged(object sender, EventArgs e)

{

TestCourse = tb\_testKursName.Text;

}

private void tb\_testName\_TextChanged(object sender, EventArgs e)

{

TestName = tb\_testName.Text;

}

private void tb\_testType\_TextChanged(object sender, EventArgs e)

{

TestType = tb\_testType.Text;

}

private void tb\_testTime\_TextChanged(object sender, EventArgs e)

{

TestTime = tb\_testTime.Text;

}

}

}

Admin\_DeleteData

using MySqlX.XDevAPI;

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;

namespace ExamMasters.Admin

{

public partial class Admin\_DeleteData : Form

{

List<TeacherClass> teachers = new List<TeacherClass>();

List<StudentClass> students = new List<StudentClass>();

List<Group> groups = new List<Group>();

List<Course> courses = new List<Course>();

List<Lesson> lessons = new List<Lesson>();

List<Test> tests = new List<Test>();

public Admin\_DeleteData()

{

InitializeComponent();

// Формирование списка преподавателей

List<TeacherClass> teachers = DB\_Do.GetTeachers();

for (int i = 0; i < teachers.Count; i++)

{

cmbx\_teachers.Items.Add(teachers[i].Name);

}

// Формирование списка учеников

List<StudentClass> students = DB\_Do.GetStudents();

for (int i = 0; i < students.Count; i++)

{

cmbx\_students.Items.Add(students[i].Name);

}

// Формирование списка групп

List<Group> group = DB\_Do.GetGroups();

for (int i = 0; i < group.Count; i++)

{

cmbx\_groups.Items.Add(group[i].Name);

}

// Формирование списка курсов

List<Course> courses = DB\_Do.GetCourses();

for (int i = 0; i < courses.Count; i++)

{

cmbx\_Courses.Items.Add(courses[i].Name);

cmbx\_CourseForLesson.Items.Add(courses[i].Name);

cmbx\_CourseForTest.Items.Add(courses[i].Name);

}

}

TeacherClass teacher = new TeacherClass(-1);

StudentClass student = new StudentClass(-1);

Group group = new Group(-1);

Course course = new Course(-1);

Lesson lesson = new Lesson(-1);

Test test = new Test(-1);

private void bt\_exit\_Click(object sender, EventArgs e)

{

Admin\_Main admin = new Admin\_Main();

admin.Show();

Close();

}

private void cmbx\_teachers\_SelectedIndexChanged(object sender, EventArgs e)

{

teacher.Name = cmbx\_teachers.Text;

}

private void bt\_deleteTeacher\_Click(object sender, EventArgs e)

{

if (cmbx\_teachers.Text == "")

{

Message.Show("Ошибка", "Введите данные в обязательные поля");

return;

}

teacher.Id = DB\_Do.GetTeacher\_ID\_ByFIO(teacher.Name);

DB\_Do.DeleteTeacher(teacher.Id);

cmbx\_teachers.Text = "";

teachers.Remove(teacher);

cmbx\_teachers.Items.Remove(teacher.Name);

Message.Show("Уведомление", "Преподаватель успешно удален из базы.");

}

private void cmbx\_students\_SelectedIndexChanged(object sender, EventArgs e)

{

student.Name = cmbx\_students.Text;

}

private void bt\_deleteStudent\_Click(object sender, EventArgs e)

{

if (cmbx\_students.Text == "")

{

Message.Show("Ошибка", "Введите данные в обязательные поля");

return;

}

student.Id = DB\_Do.GetStudent\_ID\_ByName(student.Name);

DB\_Do.DeleteStudent(student.Id);

cmbx\_students.Text = "";

students.Remove(student);

cmbx\_students.Items.Remove(student.Name);

Message.Show("Уведомление", "Ученик успешно удален из базы.");

}

private void cmbx\_groups\_SelectedIndexChanged(object sender, EventArgs e)

{

group.Name = cmbx\_groups.Text;

}

private void bt\_deleteGroup\_Click(object sender, EventArgs e)

{

if (cmbx\_groups.Text == "")

{

Message.Show("Ошибка", "Введите данные в обязательные поля");

return;

}

group.Id = DB\_Do.GetGroup\_ID\_ByName(group.Name);

DB\_Do.DeleteGroup(group.Id);

cmbx\_groups.Text = "";

groups.Remove(group);

cmbx\_groups.Items.Remove(group.Name);

Message.Show("Уведомление", "Группа успешно удалена из базы.");

}

private void cmbx\_Courses\_SelectedIndexChanged(object sender, EventArgs e)

{

course.Name = cmbx\_Courses.Text;

}

private void tb\_deleteCourse\_Click(object sender, EventArgs e)

{

if (cmbx\_Courses.Text == "")

{

Message.Show("Ошибка", "Введите данные в обязательные поля");

return;

}

course.Id = DB\_Do.GetCourse\_ID\_ByName(course.Name);

DB\_Do.DeleteCourse(course.Id);

cmbx\_Courses.Text = "";

courses.Remove(course);

cmbx\_Courses.Items.Remove(course.Name);

cmbx\_CourseForLesson.Items.Remove(course.Name);

cmbx\_CourseForTest.Items.Remove(course.Name);

Message.Show("Уведомление", "Курс успешно удален из базы.");

}

private void cmbx\_CourseForLesson\_SelectedIndexChanged(object sender, EventArgs e)

{

course.Name = cmbx\_CourseForLesson.Text;

course.Id = DB\_Do.GetCourse\_ID\_ByName(course.Name);

List<Lesson> lessons = DB\_Do.GetLessons(course.Id);

cmbx\_lessonName.Items.Clear();

// Формирование списка уроков

for (int i = 0; i < lessons.Count; i++)

{

cmbx\_lessonName.Items.Add(lessons[i].Name);

}

}

private void cmbx\_lessonName\_SelectedIndexChanged(object sender, EventArgs e)

{

lesson.Name = cmbx\_lessonName.Text;

}

private void bt\_delete\_Lesson\_Click(object sender, EventArgs e)

{

if (cmbx\_CourseForLesson.Text == "" || cmbx\_lessonName.Text == "")

{

Message.Show("Ошибка", "Введите данные в обязательные поля");

return;

}

lesson.Id = DB\_Do.GetLessonId\_ByName(lesson.Name);

DB\_Do.DeleteLesson(lesson.Id);

cmbx\_lessonName.Text = "";

cmbx\_CourseForLesson.Text = "";

lessons.Remove(lesson);

cmbx\_lessonName.Items.Remove(lesson.Name);

Message.Show("Уведомление", "Занятие успешно удалено из базы.");

}

private void cmbx\_CourseForTest\_SelectedIndexChanged(object sender, EventArgs e)

{

course.Name = cmbx\_CourseForTest.Text;

course.Id = DB\_Do.GetCourse\_ID\_ByName(course.Name);

List<Test> tests = DB\_Do.GetTest(course.Id);

cmbx\_testName.Items.Clear();

// Формирование списка уроков

for (int i = 0; i < tests.Count; i++)

{

cmbx\_testName.Items.Add(tests[i].Name);

}

}

private void cmbx\_testName\_SelectedIndexChanged(object sender, EventArgs e)

{

test.Name = cmbx\_testName.Text;

}

private void bt\_deleteTest\_Click(object sender, EventArgs e)

{

if (cmbx\_CourseForTest.Text == "" || cmbx\_testName.Text == "")

{

Message.Show("Ошибка", "Введите данные в обязательные поля");

return;

}

test.Id = DB\_Do.GetTestId\_ByName(test.Name);

DB\_Do.DeleteTest(test.Id);

cmbx\_testName.Text = "";

cmbx\_CourseForTest.Text = "";

tests.Remove(test);

cmbx\_testName.Items.Remove(test.Name);

Message.Show("Уведомление", "Тестирование успешно удалено из базы.");

}

}

}

Admin\_Statistics

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics.Eventing.Reader;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using ExamMasters.Student;

using Spire.Xls;

namespace ExamMasters.Admin

{

public partial class Admin\_Statistics : Form

{

int teacherAttendance;

int teacherHometask;

int studentAttendance;

int studentHometask;

string studentTeacher;

string studentGroup;

int groupAttendance;

int groupHometask;

string groupTeacher;

string[,] exportData;

SaveFileDialog SaveFileDialog1 = new SaveFileDialog();

public Admin\_Statistics()

{

InitializeComponent();

// Формирование списка преподавателей

List<TeacherClass> teacher = DB\_Do.GetTeachers();

for (int i = 0; i < teacher.Count; i++)

{

cmbx\_teachers.Items.Add(teacher[i].Name);

cmbx\_groupTeacher.Items.Add(teacher[i].Name);

}

// Формирование списка групп

groups = DB\_Do.GetGroups();

for (int i = 0; i < groups.Count; i++)

{

cmbx\_groups.Items.Add(groups[i].Name);

}

studentAttendance = 100;

studentHometask = 100;

teacherAttendance = 100;

teacherHometask = 100;

groupAttendance = 100;

groupHometask = 100;

}

List<Group> groups = new List<Group>();

List<TeacherClass> teacher = new List<TeacherClass>();

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

{

}

private void dataGridView1\_CellContentClick\_1(object sender, DataGridViewCellEventArgs e)

{

}

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

{

dataGridView1.Rows.Clear();

if (tb\_attendance.Text == "")

{

teacherAttendance = 100;

}

if (tb\_hometask.Text == "")

{

teacherHometask = 100;

}

List<TeacherClass> teachers = new List<TeacherClass>();

teachers = DB\_Do.GetAllStatTeacher();

for (int i = 0; i < teachers.Count; i++)

{

if (Math.Round(teachers[i].Attendance \* 100) <= teacherAttendance && Math.Round(teachers[i].Hometask) <= teacherHometask)

{

dataGridView1.Rows.Add(teachers[i].Name, teachers[i].Phone, teachers[i].Email, Math.Round(teachers[i].Attendance \* 100), Math.Round(teachers[i].Hometask));

}

}

}

private void textBox6\_TextChanged(object sender, EventArgs e)

{

if (tb\_groupAttendance.Text != "")

{

groupAttendance = Convert.ToInt32(tb\_groupAttendance.Text);

}

else { groupAttendance = 100; }

}

private void label8\_Click(object sender, EventArgs e)

{

}

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

{

}

private void textBox5\_TextChanged(object sender, EventArgs e)

{

if (tb\_groupHometask.Text != "")

{

groupHometask = Convert.ToInt32(tb\_groupHometask.Text);

}

else { groupAttendance = 100; }

}

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

{

Admin\_Main admin\_Main = new Admin\_Main();

admin\_Main.Show();

Close();

}

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

{

if (tb\_attendance.Text != "") { teacherAttendance = Convert.ToInt32(tb\_attendance.Text); };

}

private void tb\_hometask\_TextChanged(object sender, EventArgs e)

{

if (tb\_hometask.Text != "") { teacherHometask = Convert.ToInt32(tb\_hometask.Text); }

}

private void bt\_export\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить файл Excel";

saveFileDialog.FileName = "Сводная статистика по преподавателям.xlsx"; // Имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

//Создание экземпляра Workbook

Workbook workbook = new Workbook();

//Получение первой рабочей страницы

Worksheet worksheet = workbook.Worksheets[0];

//Создание объекта DataTable

DataTable dataTable = new DataTable();

int rows = dataGridView1.Rows.Count;

int columns = dataGridView1.ColumnCount;

exportData = new string[rows + 1, columns];

exportData[0, 0] = "ФИО";

exportData[0, 1] = "Телефон";

exportData[0, 2] = "E-mail";

exportData[0, 3] = "Посещамость %";

exportData[0, 4] = "ДЗ %";

for (int i = 1; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

exportData[i, j] = Convert.ToString(dataGridView1[j, i - 1].Value);

}

}

worksheet.Range[1, 1].Value = "ОБЩАЯ СТАТИСТИКА ПРЕПОДАВАТЕЛЕЙ";

worksheet.Range[2, 2].Value = "";

//Запись массива в таблицу, начиная с ячейки A3

worksheet.InsertArray(exportData, 3, 1);

//Автоматическое подгонка ширины столбцов

worksheet.AllocatedRange.AutoFitColumns();

//Применение стиля к первой и третьей строке

CellStyle style = workbook.Styles.Add("newStyle");

style.Font.IsBold = true;

worksheet.Range[1, 1, 1, 6].Style = style;

worksheet.Range[3, 1, 3, 6].Style = style;

//Сохранение в файл Excel

workbook.SaveToFile(saveFileDialog.FileName, ExcelVersion.Version2016);

Message.Show("Уведомление", "Данные успешно экспортированы.");

}

catch

{

MessageBox.Show("Файл не сохранен", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

Message.Show("Уведомление", "Сохранение файла отменено.");

}

}

private void bt\_studentExit\_Click(object sender, EventArgs e)

{

Admin\_Main admin\_Main = new Admin\_Main();

admin\_Main.Show();

Hide();

}

private void bt\_groupExit\_Click(object sender, EventArgs e)

{

Admin\_Main admin\_Main = new Admin\_Main();

admin\_Main.Show();

Hide();

}

private void tb\_studentAttendance\_TextChanged(object sender, EventArgs e)

{

if (tb\_studentAttendance.Text != "")

{

studentAttendance = Convert.ToInt32(tb\_studentAttendance.Text);

}

else { studentAttendance = 100; }

}

private void tb\_studentHometask\_TextChanged(object sender, EventArgs e)

{

if (tb\_studentHometask.Text != "")

{

studentHometask = Convert.ToInt32(tb\_studentHometask.Text);

}

else { studentHometask = 100; }

}

private void cmbx\_teachers\_SelectedIndexChanged(object sender, EventArgs e)

{

studentTeacher = cmbx\_teachers.Text;

int teacher\_ID = DB\_Do.GetTeacher\_ID\_ByFIO(studentTeacher);

if (studentTeacher != "")

{

cmbx\_groups.Items.Clear();

List<Group> listOfGroup = new List<Group>();

listOfGroup = DB\_Do.GetGroupsOfTeacher(teacher\_ID);

for (int i = 0; i < listOfGroup.Count; i++)

{

cmbx\_groups.Items.Add(listOfGroup[i].Name);

}

}

}

private void cmbx\_groups\_SelectedIndexChanged(object sender, EventArgs e)

{

studentGroup = cmbx\_groups.Text;

}

private void bt\_resetSort\_Click(object sender, EventArgs e)

{

tb\_studentAttendance.Text = "";

tb\_studentHometask.Text = "";

cmbx\_teachers.Text = "";

cmbx\_groups.Text = "";

cmbx\_groups.Items.Clear();

groups = DB\_Do.GetGroups();

for (int i = 0; i < groups.Count; i++)

{

cmbx\_groups.Items.Add(groups[i].Name);

}

}

private void bt\_studentStetistics\_Click(object sender, EventArgs e)

{

dataGridView2.Rows.Clear();

List<StudentClass> students = new List<StudentClass>();

students = DB\_Do.GetAllStatStudent();

List<StudentClass> studentSort = students.FindAll(students => studentAttendance >= Math.Round(students.Attendance \* 100) && studentHometask >= Math.Round(students.Hometask)).ToList();

// Статистика учеников одного преподавателя (включая сортировку по показателям)

if (cmbx\_teachers.Text != "")

{

studentSort = studentSort.FindAll(studentSort => studentSort.TeacherName == cmbx\_teachers.Text).ToList();

}

if (cmbx\_groups.Text != "")

{

int group\_ID = DB\_Do.GetGroup\_ID\_ByName(studentGroup);

studentSort = studentSort.FindAll(studentSort => studentSort.GroupID == group\_ID).ToList();

}

for (int i = 0; i < studentSort.Count; i++)

{

dataGridView2.Rows.Add(studentSort[i].Name, studentSort[i].Phone, studentSort[i].TeacherName, studentSort[i].GroupID, Math.Round(studentSort[i].Attendance \* 100), Math.Round(studentSort[i].Hometask));

}

}

private void bt\_studentExport\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить файл Excel";

saveFileDialog.FileName = "Сводная статистика по ученикам.xlsx"; // Имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

//Создание экземпляра Workbook

Workbook workbook = new Workbook();

//Получение первой рабочей страницы

Worksheet worksheet = workbook.Worksheets[0];

//Создание объекта DataTable

DataTable dataTable = new DataTable();

int rows = dataGridView2.Rows.Count;

int columns = dataGridView2.ColumnCount;

exportData = new string[rows + 1, columns];

exportData[0, 0] = "ФИО ученика";

exportData[0, 1] = "Телефон";

exportData[0, 2] = "ФИО преподавателя";

exportData[0, 3] = "Код группы";

exportData[0, 4] = "Посещаемость %";

exportData[0, 5] = "ДЗ %";

for (int i = 1; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

exportData[i, j] = Convert.ToString(dataGridView2[j, i - 1].Value);

}

}

worksheet.Range[1, 1].Value = "ОБЩАЯ СТАТИСТИКА УЧЕНИКОВ";

worksheet.Range[2, 2].Value = "";

//Запись массива в таблицу, начиная с ячейки A3

worksheet.InsertArray(exportData, 3, 1);

//Автоматическое подгонка ширины столбцов

worksheet.AllocatedRange.AutoFitColumns();

//Применение стиля к первой и третьей строке

CellStyle style = workbook.Styles.Add("newStyle");

style.Font.IsBold = true;

worksheet.Range[1, 1, 1, 6].Style = style;

worksheet.Range[3, 1, 3, 6].Style = style;

//Сохранение в файл Excel

workbook.SaveToFile(saveFileDialog.FileName, ExcelVersion.Version2016);

Message.Show("Уведомление", "Данные успешно экспортированы.");

}

catch (Exception ex)

{

MessageBox.Show("Файл не сохранен" + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

Console.WriteLine("Сохранение файла отменено.");

}

}

private void bt\_resetSortGroup\_Click(object sender, EventArgs e)

{

tb\_groupAttendance.Text = "";

groupAttendance = 100;

tb\_groupHometask.Text = "";

groupHometask = 100;

cmbx\_groupTeacher.Text = "";

}

private void bt\_groupStatistics\_Click(object sender, EventArgs e)

{

dataGridView3.Rows.Clear();

List<Group> groups = new List<Group>();

groups = DB\_Do.AllStatGroup();

List<Group> groupsSort = groups.FindAll(groups => groupAttendance >= Math.Round(groups.Attendance \* 100) && groupHometask >= Math.Round(groups.Hometask)).ToList();

if (cmbx\_groupTeacher.Text != "")

{

groupsSort = groupsSort.FindAll(groupsSort => groupTeacher == groupsSort.TeacherName).ToList();

}

for (int i = 0; i < groupsSort.Count; i++)

{

dataGridView2.Rows.Add();

dataGridView3.Rows.Add(groupsSort[i].Name, groupsSort[i].TeacherName, groupsSort[i].CourseName, Math.Round(groupsSort[i].Attendance \* 100), Math.Round(groupsSort[i].Hometask));

}

}

private void cmbx\_groupTeacher\_SelectedIndexChanged(object sender, EventArgs e)

{

groupTeacher = cmbx\_groupTeacher.Text;

}

private void bt\_groupExport\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить файл Excel";

saveFileDialog.FileName = "Сводная статистика по группам.xlsx"; // Имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

//Создание экземпляра Workbook

Workbook workbook = new Workbook();

//Получение первой рабочей страницы

Worksheet worksheet = workbook.Worksheets[0];

//Создание объекта DataTable

DataTable dataTable = new DataTable();

int rows = dataGridView3.Rows.Count;

int columns = dataGridView3.ColumnCount;

exportData = new string[rows + 1, columns];

exportData[0, 0] = "Код группы";

exportData[0, 1] = "ФИО преподавателя";

exportData[0, 2] = "Курс";

exportData[0, 3] = "Посещаемость %";

exportData[0, 4] = "ДЗ %";

for (int i = 1; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

exportData[i, j] = Convert.ToString(dataGridView3[j, i - 1].Value);

}

}

worksheet.Range[1, 1].Value = "ОБЩАЯ СТАТИСТИКА ГРУПП";

worksheet.Range[2, 2].Value = "";

//Запись массива в таблицу, начиная с ячейки A3

worksheet.InsertArray(exportData, 3, 1);

//Автоматическое подгонка ширины столбцов

worksheet.AllocatedRange.AutoFitColumns();

//Применение стиля к первой и третьей строке

CellStyle style = workbook.Styles.Add("newStyle");

style.Font.IsBold = true;

worksheet.Range[1, 1, 1, 6].Style = style;

worksheet.Range[3, 1, 3, 6].Style = style;

//Сохранение в файл Excel

workbook.SaveToFile(saveFileDialog.FileName, ExcelVersion.Version2016);

Message.Show("Уведомление", "Данные успешно экспортированы.");

}

catch (Exception ex)

{

MessageBox.Show("Файл не сохранен" + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

Console.WriteLine("Сохранение файла отменено.");

}

}

}

}

Teacher\_Main

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;

namespace ExamMasters.Teacher

{

public partial class Teacher\_Main : Form

{

TeacherClass Teacher;

public Teacher\_Main(TeacherClass t)

{

InitializeComponent();

Teacher = new TeacherClass(t.Id, t.Name, t.Email, t.Phone, t.Password);

}

private void bt\_exit\_Click(object sender, EventArgs e)

{

Autorization auto = new Autorization();

auto.Show();

Close();

}

private void Teacher\_Main\_Load(object sender, EventArgs e)

{

}

private void bt\_statistics\_Click(object sender, EventArgs e)

{

Teacher\_Statistics teacher\_Statistics = new Teacher\_Statistics(Teacher);

teacher\_Statistics.Show();

Hide();

}

private void bt\_schedule\_Click(object sender, EventArgs e)

{

Teacher\_Schedule teacher\_Schedule = new Teacher\_Schedule(Teacher);

teacher\_Schedule.Show();

Hide();

}

private void bt\_info\_Click(object sender, EventArgs e)

{

Teacher\_Info teacher\_Info = new Teacher\_Info(Teacher);

teacher\_Info.Show();

Hide();

}

}

}

Teacher\_Info

using Spire.Xls;

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;

namespace ExamMasters.Teacher

{

public partial class Teacher\_Info : Form

{

public TeacherClass Teacher;

public string GroupName;

public int GroupId;

string[,] exportData;

public Teacher\_Info(TeacherClass teacher)

{

InitializeComponent();

Teacher = teacher;

Teacher.Id = teacher.Id;

List<Group> groups = new List<Group>();

groups = DB\_Do.GetGroups\_ByTeacherID(Teacher.Id);

for (int i = 0; i < groups.Count; i++)

{

cmbx\_group.Items.Add(groups[i].Name);

}

}

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

{

}

private void Teacher\_Info\_Load(object sender, EventArgs e)

{

}

private void bt\_exit\_Click(object sender, EventArgs e)

{

Teacher\_Main teacher = new Teacher\_Main(Teacher);

teacher.Show();

Close();

}

private void bt\_resetSort\_Click(object sender, EventArgs e)

{

cmbx\_group.Text = "";

}

private void cmbx\_group\_SelectedIndexChanged(object sender, EventArgs e)

{

GroupName = cmbx\_group.Text;

}

private void bt\_ShowData\_Click(object sender, EventArgs e)

{

if (cmbx\_group.Text != "")

{

dataGridView1.Rows.Clear();

GroupId = DB\_Do.GetGroup\_ID\_ByName(GroupName);

List<StudentClass> students = DB\_Do.GetListOfStudentsInGroup(GroupId);

for (int i = 0; i < students.Count; i++)

{

dataGridView1.Rows.Add(students[i].Name, students[i].Phone, students[i].Email);

}

}

else

{

Message.Show("Ошибка", "Введите наименование группы.");

}

}

private void bt\_export\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить файл Excel";

saveFileDialog.FileName = "Контактная информация о учеников группы " + GroupName+ ".xlsx"; // Имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

//Создание экземпляра Workbook

Workbook workbook = new Workbook();

//Получение первой рабочей страницы

Worksheet worksheet = workbook.Worksheets[0];

//Создание объекта DataTable

DataTable dataTable = new DataTable();

int rows = dataGridView1.Rows.Count;

int columns = dataGridView1.ColumnCount;

exportData = new string[rows + 1, columns];

exportData[0, 0] = "ФИО";

exportData[0, 1] = "Телефон";

exportData[0, 2] = "E-mail";

for (int i = 1; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

exportData[i, j] = Convert.ToString(dataGridView1[j, i - 1].Value);

}

}

worksheet.Range[1, 1].Value = "КОНТАКТНАЯ ИНФОРМАЦИЯ УЧЕНИКОВ ГРУППЫ " + GroupName;

worksheet.Range[2, 2].Value = "";

//Запись массива в таблицу, начиная с ячейки A3

worksheet.InsertArray(exportData, 3, 1);

//Автоматическое подгонка ширины столбцов

worksheet.AllocatedRange.AutoFitColumns();

//Применение стиля к первой и третьей строке

CellStyle style = workbook.Styles.Add("newStyle");

style.Font.IsBold = true;

worksheet.Range[1, 1, 1, 6].Style = style;

worksheet.Range[3, 1, 3, 6].Style = style;

//Сохранение в файл Excel

workbook.SaveToFile(saveFileDialog.FileName, ExcelVersion.Version2016);

Message.Show("Уведомление", "Данные успешно экспортированы.");

}

catch

{

MessageBox.Show("Файл не сохранен", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

Message.Show("Уведомление", "Сохранение файла отменено.");

}

}

}

}

Teacher\_Schedule

using Spire.Xls;

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;

namespace ExamMasters.Teacher

{

public partial class Teacher\_Schedule : Form

{

public TeacherClass teacher = new TeacherClass(-1);

public string groupNameSelected = "";

public string[,] exportData;

public Teacher\_Schedule(TeacherClass t)

{

InitializeComponent();

teacher = new TeacherClass(t.Id, t.Name, t.Email, t.Phone, t.Password);

List<Group> groups = DB\_Do.GetGroupsOfTeacher(teacher.Id);

foreach (Group group in groups)

{

cmbx\_groups.Items.Add(group.Name);

}

}

private void Teacher\_Schedule\_Load(object sender, EventArgs e)

{

}

private void bt\_show\_Click(object sender, EventArgs e)

{

dataGridView1.Rows.Clear();

List<Lesson> lessons = DB\_Do.GetAllLessonsByTeacherId(teacher.Id);

List<Lesson> lessonsSort = lessons.FindAll(lesson => lesson.Date.Date >= dateTimePicker1.Value.Date && lesson.Date.Date <= dateTimePicker2.Value.Date).ToList();

if (rbt\_selectAll.Checked || rbt\_selectOne.Checked)

{

if (rbt\_selectOne.Checked && cmbx\_groups.Text == "")

{

Message.Show("Ошибка", "Введите название группы.");

}

if (rbt\_selectOne.Checked && cmbx\_groups.Text != "")

{

lessonsSort = lessonsSort.FindAll(lesson => lesson.GroupName == groupNameSelected).ToList();

}

foreach (Lesson lesson in lessonsSort)

{

dataGridView1.Rows.Add(lesson.GroupName, lesson.Date.ToString("d"), lesson.Name, lesson.Video, lesson.Hometask);

}

}

else { Message.Show("Ошибка", "Введите параметры сортировки."); }

}

private void bt\_exit\_Click(object sender, EventArgs e)

{

Teacher\_Main teacher\_Main = new Teacher\_Main(teacher);

teacher\_Main.Show();

Close();

}

private void cmbx\_groups\_SelectedIndexChanged(object sender, EventArgs e)

{

groupNameSelected = cmbx\_groups.Text;

}

private void bt\_resetSort\_Click(object sender, EventArgs e)

{

rbt\_selectAll.Checked = false;

rbt\_selectOne.Checked = false;

dateTimePicker1.Value = DateTime.Now;

dateTimePicker2.Value = DateTime.Now;

cmbx\_groups.Text = "";

}

private void bt\_export\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить файл Excel";

saveFileDialog.FileName = "Расписание групп.xlsx"; // Имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

//Создание экземпляра Workbook

Workbook workbook = new Workbook();

//Получение первой рабочей страницы

Worksheet worksheet = workbook.Worksheets[0];

//Создание объекта DataTable

DataTable dataTable = new DataTable();

int rows = dataGridView1.Rows.Count;

int columns = dataGridView1.ColumnCount;

exportData = new string[rows + 1, columns];

exportData[0, 0] = "Группа";

exportData[0, 1] = "Дата занятия";

exportData[0, 2] = "Название";

exportData[0, 3] = "Ссылка на урок";

exportData[0, 4] = "Ссылка на ДЗ";

for (int i = 1; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

exportData[i, j] = Convert.ToString(dataGridView1[j, i - 1].Value);

}

}

worksheet.Range[1, 1].Value = "ОБЩАЯ СТАТИСТИКА ПРЕПОДАВАТЕЛЕЙ";

worksheet.Range[2, 2].Value = "";

//Запись массива в таблицу, начиная с ячейки A3

worksheet.InsertArray(exportData, 3, 1);

//Автоматическое подгонка ширины столбцов

worksheet.AllocatedRange.AutoFitColumns();

//Применение стиля к первой и третьей строке

CellStyle style = workbook.Styles.Add("newStyle");

style.Font.IsBold = true;

worksheet.Range[1, 1, 1, 6].Style = style;

worksheet.Range[3, 1, 3, 6].Style = style;

//Сохранение в файл Excel

workbook.SaveToFile(saveFileDialog.FileName, ExcelVersion.Version2016);

Message.Show("Уведомление", "Данные успешно экспортированы.");

}

catch

{

MessageBox.Show("Файл не сохранен", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

Message.Show("Уведомление", "Сохранение файла отменено.");

}

}

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

{

}

}

}

Teacher\_Statistics

using Microsoft.VisualBasic;

using Spire.Xls;

using Spire.Xls.AI;

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;

namespace ExamMasters.Teacher

{

public partial class Teacher\_Statistics : Form

{

TeacherClass Teacher;

public string GroupName;

public int GroupId;

public string[,] exportData;

public Teacher\_Statistics(TeacherClass t)

{

InitializeComponent();

Teacher = new TeacherClass(t.Id, t.Name, t.Email, t.Phone, t.Password);

List<Group> groups = new List<Group>();

groups = DB\_Do.GetGroupsOfTeacher(Teacher.Id);

foreach (Group group in groups)

{

cmbx\_groups.Items.Add(group.Name);

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

GroupName = cmbx\_groups.Text;

GroupId = DB\_Do.GetGroup\_ID\_ByName(GroupName);

}

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

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить файл Excel";

saveFileDialog.FileName = "Сводный отчет по группе " + GroupName + ".xlsx"; // Имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

//Создание экземпляра Workbook

Workbook workbook = new Workbook();

//Получение первой рабочей страницы

Worksheet worksheet = workbook.Worksheets[0];

//Создание объекта DataTable

DataTable dataTable = new DataTable();

int rows = dataGridView1.Rows.Count;

int columns = dataGridView1.ColumnCount;

exportData = new string[rows + 1, columns];

exportData[0, 0] = "ФИО";

exportData[0, 1] = "Общий процент посещаемости";

exportData[0, 2] = "Общий процент ДЗ";

for (int i = 1; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

exportData[i, j] = Convert.ToString(dataGridView1[j, i - 1].Value);

}

}

worksheet.Range[1, 1].Value = "СВОДНЫЙ ОТЧЕТ ПО ГРУППЕ " + GroupName;

worksheet.Range[2, 2].Value = "";

//Запись массива в таблицу, начиная с ячейки A3

worksheet.InsertArray(exportData, 3, 1);

//Автоматическое подгонка ширины столбцов

worksheet.AllocatedRange.AutoFitColumns();

//Применение стиля к первой и третьей строке

CellStyle style = workbook.Styles.Add("newStyle");

style.Font.IsBold = true;

worksheet.Range[1, 1, 1, 6].Style = style;

worksheet.Range[3, 1, 3, 6].Style = style;

//Сохранение в файл Excel

workbook.SaveToFile(saveFileDialog.FileName, ExcelVersion.Version2016);

Message.Show("Уведомление", "Данные успешно экспортированы.");

}

catch

{

MessageBox.Show("Файл не сохранен", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

Message.Show("Уведомление", "Сохранение файла отменено.");

}

}

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

{

}

private void tabPage3\_Click(object sender, EventArgs e)

{

}

private void Teacher\_Statistics\_Load(object sender, EventArgs e)

{

}

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

{

Teacher\_Main teacher\_Main = new Teacher\_Main(Teacher);

teacher\_Main.Show();

Close();

}

private void bt\_ShowData\_Click(object sender, EventArgs e)

{

if (cmbx\_groups.Text != "")

{

List<StudentClass> students = DB\_Do.GetAllStatStudent();

students = students.FindAll(student => student.GroupID == GroupId).ToList();

foreach (StudentClass student in students)

{

dataGridView1.Rows.Add(student.Name, Math.Round(student.Attendance \* 100), Math.Round(student.Hometask));

}

}

else { Message.Show("Ошибка", "Выберете группу."); }

}

private void bt\_attendanceExport\_Click(object sender, EventArgs e)

{

if (cmbx\_groups.Text == "")

{

Message.Show("Ошибка", "Выберете группу.");

return;

}

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить файл Excel";

saveFileDialog.FileName = "Датированный отчет по посещаемости группы " + GroupName + ".xlsx"; // Имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

//Создание экземпляра Workbook

Workbook workbook = new Workbook();

//Получение первой рабочей страницы

Worksheet worksheet = workbook.Worksheets[0];

GroupId = DB\_Do.GetGroup\_ID\_ByName(GroupName);

List<StudentClass> students = DB\_Do.GetListOfStudentsInGroup(GroupId);

List<List<string>> exportData = new List<List<string>>();

List<string> headers = new List<string>();

headers.Add("ФИО");

foreach (StudentClass student in students)

{

List<Lesson> lessons = DB\_Do.StudentGetAttendanceHomework(student.Id);

List<string> studentAttendance = new List<string>();

studentAttendance.Add(student.Name);

foreach (Lesson lesson in lessons)

{

studentAttendance.Add(lesson.Attendance.ToString());

if (headers.Count < lessons.Count + 1)

{

headers.Add(lesson.Date.ToString("d"));

}

}

exportData.Add(studentAttendance);

}

exportData.Insert(0, headers);

string[,] arrayData = new string[exportData.Count, exportData[0].Count];

for (int i = 0; i < exportData.Count; i++)

{

for (int j = 0; j < exportData[i].Count; j++)

{

arrayData[i, j] = exportData[i][j];

}

}

worksheet.Range[1, 1].Value = "ДАТИРОВАННЫЙ ОТЧЕТ ПО ПОСЕЩАЕМОСТИ ГРУППЫ " + GroupName;

worksheet.Range[2, 2].Value = "";

//Запись массива в таблицу, начиная с ячейки A3

worksheet.InsertArray(arrayData, 3, 1);

//Автоматическое подгонка ширины столбцов

worksheet.AllocatedRange.AutoFitColumns();

//Применение стиля к первой и третьей строке

CellStyle style = workbook.Styles.Add("newStyle");

style.Font.IsBold = true;

worksheet.Range[1, 1, 1, 6].Style = style;

worksheet.Range[3, 1, 3, 6].Style = style;

for (int j = 0; j < exportData[0].Count - 1; j++)

{

worksheet.Range[3, j + 2].Style.NumberFormat = System.Globalization.DateTimeFormatInfo.CurrentInfo.ShortDatePattern;

for (int i = 0; i < exportData.Count - 1; i++)

{

worksheet.Range[4 + i, j + 2].Style.NumberFormat = "@";

}

}

//Сохранение в файл Excel

workbook.SaveToFile(saveFileDialog.FileName, ExcelVersion.Version2016);

Message.Show("Уведомление", "Данные успешно экспортированы.");

}

catch

{

MessageBox.Show("Файл не сохранен", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

Message.Show("Уведомление", "Сохранение файла отменено.");

}

}

private void bt\_hometaskExport\_Click(object sender, EventArgs e)

{

if (cmbx\_groups.Text == "")

{

Message.Show("Ошибка", "Выберете группу.");

return;

}

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить файл Excel";

saveFileDialog.FileName = "Датированный отчет по ДЗ группы " + GroupName + ".xlsx"; // Имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

//Создание экземпляра Workbook

Workbook workbook = new Workbook();

//Получение первой рабочей страницы

Worksheet worksheet = workbook.Worksheets[0];

GroupId = DB\_Do.GetGroup\_ID\_ByName(GroupName);

List<StudentClass> students = DB\_Do.GetListOfStudentsInGroup(GroupId);

List<List<string>> exportData = new List<List<string>>();

List<string> headers = new List<string>();

headers.Add("ФИО");

foreach (StudentClass student in students)

{

List<Lesson> lessons = DB\_Do.StudentGetAttendanceHomework(student.Id);

List<string> studentAttendance = new List<string>();

studentAttendance.Add(student.Name);

foreach (Lesson lesson in lessons)

{

studentAttendance.Add(lesson.HometaskProcent.ToString());

if (headers.Count < lessons.Count + 1)

{

headers.Add(lesson.Date.ToString("d"));

}

}

exportData.Add(studentAttendance);

}

exportData.Insert(0, headers);

string[,] arrayData = new string[exportData.Count, exportData[0].Count];

for (int i = 0; i < exportData.Count; i++)

{

for (int j = 0; j < exportData[i].Count; j++)

{

arrayData[i, j] = exportData[i][j];

}

}

worksheet.Range[1, 1].Value = "ДАТИРОВАННЫЙ ОТЧЕТ ПО ДЗ ГРУППЫ " + GroupName;

worksheet.Range[2, 2].Value = "";

//Запись массива в таблицу, начиная с ячейки A3

worksheet.InsertArray(arrayData, 3, 1);

//Автоматическое подгонка ширины столбцов

worksheet.AllocatedRange.AutoFitColumns();

//Применение стиля к первой и третьей строке

CellStyle style = workbook.Styles.Add("newStyle");

style.Font.IsBold = true;

worksheet.Range[1, 1, 1, 6].Style = style;

worksheet.Range[3, 1, 3, 6].Style = style;

for (int j = 0; j < exportData[0].Count - 1; j++)

{

worksheet.Range[3, j + 2].Style.NumberFormat = System.Globalization.DateTimeFormatInfo.CurrentInfo.ShortDatePattern;

for (int i = 0; i < exportData.Count - 1; i++)

{

worksheet.Range[4 + i, j + 2].Style.NumberFormat = "@";

}

}

//Сохранение в файл Excel

workbook.SaveToFile(saveFileDialog.FileName, ExcelVersion.Version2016);

Message.Show("Уведомление", "Данные успешно экспортированы.");

}

catch

{

MessageBox.Show("Файл не сохранен", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

Message.Show("Уведомление", "Сохранение файла отменено.");

}

}

}

}

Student\_Main

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;

namespace ExamMasters.Student

{

public partial class Student\_Main : Form

{

public StudentClass student = new StudentClass(-1);

public Student\_Main(StudentClass st)

{

InitializeComponent();

student = new StudentClass(st.Id, st.Name, st.Email, st.Phone, st.Password);

}

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

{

Autorization autorization = new Autorization();

autorization.Show();

Close();

}

private void bt\_showSchedule\_Click(object sender, EventArgs e)

{

Student\_Schedule student\_Schedule = new Student\_Schedule(student);

student\_Schedule.Show();

Hide();

}

private void bt\_showDebts\_Click(object sender, EventArgs e)

{

Student\_Debts student\_Debts = new Student\_Debts(student);

student\_Debts.Show();

Hide();

}

private void Student\_Main\_Load(object sender, EventArgs e)

{

}

}

}

Student\_Debts

using Spire.Xls;

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;

namespace ExamMasters.Student

{

public partial class Student\_Debts : Form

{

StudentClass student;

string[,] exportData;

public Student\_Debts(StudentClass st)

{

InitializeComponent();

student = new StudentClass(st.Id, st.Name, st.Email, st.Phone, st.Password);

}

private void Student\_Debts\_Load(object sender, EventArgs e)

{

}

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

{

Student\_Main studentMain = new Student\_Main(student);

studentMain.Show();

Close();

}

private void bt\_debtsShowData\_Click(object sender, EventArgs e)

{

dgv\_debtsData.Rows.Clear();

List<Lesson> debts = DB\_Do.StudentGetAttendanceHomework(student.Id);

debts = debts.FindAll(lesson => lesson.Attendance != 1 || lesson.HometaskProcent < 100);

foreach (Lesson lesson in debts)

{

dgv\_debtsData.Rows.Add(lesson.Name, lesson.Date, lesson.Attendance, lesson.HometaskProcent);

}

}

private void bt\_export\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить файл Excel";

saveFileDialog.FileName = "Долги по учебе.xlsx"; // Имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

//Создание экземпляра Workbook

Workbook workbook = new Workbook();

//Получение первой рабочей страницы

Worksheet worksheet = workbook.Worksheets[0];

//Создание объекта DataTable

DataTable dataTable = new DataTable();

int rows = dgv\_debtsData.Rows.Count;

int columns = dgv\_debtsData.ColumnCount;

exportData = new string[rows + 1, columns];

exportData[0, 0] = "Название курса";

exportData[0, 1] = "Дата занятия";

exportData[0, 2] = "Посещаемость";

exportData[0, 3] = "ДЗ";

for (int i = 1; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

exportData[i, j] = Convert.ToString(dgv\_debtsData[j, i - 1].Value);

}

}

worksheet.Range[1, 1].Value = "ДОЛГИ ПО УЧЕБЕ";

worksheet.Range[2, 2].Value = "";

//Запись массива в таблицу, начиная с ячейки A3

worksheet.InsertArray(exportData, 3, 1);

//Автоматическое подгонка ширины столбцов

worksheet.AllocatedRange.AutoFitColumns();

//Применение стиля к первой и третьей строке

CellStyle style = workbook.Styles.Add("newStyle");

style.Font.IsBold = true;

worksheet.Range[1, 1, 1, 6].Style = style;

worksheet.Range[3, 1, 3, 6].Style = style;

//Сохранение в файл Excel

workbook.SaveToFile(saveFileDialog.FileName, ExcelVersion.Version2016);

Message.Show("Уведомление", "Данные успешно экспортированы.");

}

catch

{

MessageBox.Show("Файл не сохранен", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

Message.Show("Уведомление", "Сохранение файла отменено.");

}

}

}

}

Student\_Schedule

using MySqlX.XDevAPI;

using Spire.Xls;

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;

namespace ExamMasters.Student

{

public partial class Student\_Schedule : Form

{

StudentClass student;

public Student\_Schedule(StudentClass st)

{

InitializeComponent();

student = new StudentClass(st.Id, st.Name, st.Email, st.Phone, st.Password);

}

private void Student\_Schedule\_Load(object sender, EventArgs e)

{

}

private void bt\_exit\_Click(object sender, EventArgs e)

{

Student\_Main student\_Main = new Student\_Main(student);

student\_Main.Show();

Close();

}

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

{

}

private void bt\_ShowSchedule\_Click(object sender, EventArgs e)

{

dataGridView1.Rows.Clear();

List<Lesson> lessons = DB\_Do.GetAllLessonsByStudentId(student.Id);

List<Lesson> lessonsSort = lessons.FindAll(lesson => lesson.Date.Date >= dateTimePicker1.Value.Date && lesson.Date.Date <= dateTimePicker2.Value.Date).ToList();

foreach (Lesson lesson in lessonsSort)

{

dataGridView1.Rows.Add(lesson.GroupName, lesson.Date.ToString("d"), lesson.Name, lesson.Video, lesson.Hometask);

}

}

private void bt\_export\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Excel files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить файл Excel";

saveFileDialog.FileName = "Расписание.xlsx"; // Имя файла по умолчанию

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

//Создание экземпляра Workbook

Workbook workbook = new Workbook();

//Получение первой рабочей страницы

Worksheet worksheet = workbook.Worksheets[0];

//Создание объекта DataTable

DataTable dataTable = new DataTable();

int rows = dataGridView1.Rows.Count;

int columns = dataGridView1.ColumnCount;

string[,] exportData;

exportData = new string[rows + 1, columns];

exportData[0, 0] = "Курс";

exportData[0, 1] = "Дата занятия";

exportData[0, 2] = "Название занятия";

exportData[0, 3] = "Ссылка на урок";

exportData[0, 4] = "Ссылка на ДЗ";

for (int i = 1; i < rows; i++)

{

for (int j = 0; j < columns; j++)

{

exportData[i, j] = Convert.ToString(dataGridView1[j, i - 1].Value);

}

}

worksheet.Range[1, 1].Value = "РАСПИСАНИЕ";

worksheet.Range[2, 2].Value = "";

//Запись массива в таблицу, начиная с ячейки A3

worksheet.InsertArray(exportData, 3, 1);

//Автоматическое подгонка ширины столбцов

worksheet.AllocatedRange.AutoFitColumns();

//Применение стиля к первой и третьей строке

CellStyle style = workbook.Styles.Add("newStyle");

style.Font.IsBold = true;

worksheet.Range[1, 1, 1, 6].Style = style;

worksheet.Range[3, 1, 3, 6].Style = style;

//Сохранение в файл Excel

workbook.SaveToFile(saveFileDialog.FileName, ExcelVersion.Version2016);

Message.Show("Уведомление", "Данные успешно экспортированы.");

}

catch

{

MessageBox.Show("Файл не сохранен", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

Message.Show("Уведомление", "Сохранение файла отменено.");

}

}

}

}

## Объекты системы

StudentClass

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ExamMasters

{

public class StudentClass

{

public int Id;

public string Name;

public string Email;

public string Phone;

public string Password;

public string TeacherName;

public int GroupID;

public double Attendance;

public double Hometask;

public StudentClass(int id, string name, string email, string phone, string password)

{

Id = id;

Name = name;

Email = email;

Phone = phone;

Password = password;

TeacherName = "";

GroupID = -1;

Attendance = -1;

Hometask = -1;

}

public StudentClass(int id)

{

Id = id;

Name = "";

Email = "";

Phone = "";

Password = "";

TeacherName = "";

GroupID = -1;

Attendance = -1;

Hometask = -1;

}

public StudentClass()

{

Id = -1;

Name = "";

Email = "";

Phone = "";

Password = "";

TeacherName = "";

GroupID = -1;

Attendance = -1;

Hometask = -1;

}

}

}

TeacherClass

using MySqlX.XDevAPI;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ExamMasters

{

public class TeacherClass

{

public int Id;

public string Name;

public string Email;

public string Phone;

public string Password;

public double Attendance;

public double Hometask;

List<Group> ListOfGroups;

public TeacherClass(int id, string name, string email, string phone, string password)

{

Id = id;

Name = name;

Email = email;

Phone = phone;

Password = password;

Attendance = 0;

Hometask = 0;

ListOfGroups = new List<Group>();

}

public TeacherClass(int id, string name, string email, string phone, double attendance, double hometask)

{

Id = id;

Name = name;

Email = email;

Phone = phone;

Password = "";

Attendance = attendance;

Hometask = hometask;

ListOfGroups = new List<Group>();

}

public TeacherClass(int id)

{

Id = id;

Name = "";

Email = "";

Phone = "";

Password = "";

Attendance = 0;

Hometask = 0;

ListOfGroups = new List<Group>();

}

public List<Group> ShowMyGroups()

{

return ListOfGroups;

}

public List<StudentClass> ShowStudentsInGroup(int group\_id)

{

List<StudentClass> ListOfStudents = new List<StudentClass>();

// КОД :)

//:\_( Ищем по айдишнику

return ListOfStudents;

}

public List<StudentClass> ShowAttendance(int group\_id)

{

List<StudentClass> ListOfStudents = new List<StudentClass>();

return ListOfStudents;

}

public List<StudentClass> ShowHometask(int group\_id)

{

List<StudentClass> ListOfStudents = new List<StudentClass>();

return ListOfStudents;

}

public List <Lesson> ShowLessons(int group\_id)

{

List <Lesson> ListOfLesson = new List<Lesson>();

return ListOfLesson;

}

}

}

Course

using MySqlX.XDevAPI;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ExamMasters

{

internal class Course

{

public int Id;

public string Name;

List<Lesson> ListOfLessons;

public Course(int id, string name)

{

this.Id = id;

this.Name = name;

ListOfLessons = new List<Lesson>();

}

public Course(int id)

{

this.Id = id;

this.Name = "";

ListOfLessons = new List<Lesson>();

}

}

}

Group

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ExamMasters

{

public class Group

{

public int Id;

public string Name;

public string TeacherName;

public string CourseName;

public int CourseId;

public double Attendance;

public double Hometask;

List<StudentClass> ListOfStudents;

public Group(int id, string name)

{

Id = id;

Name = name;

TeacherName = "";

CourseId = -1;

CourseName = "";

Attendance = 0;

Hometask = 0;

ListOfStudents = new List<StudentClass>();

}

public Group(int id)

{

Id = id;

Name = "";

TeacherName = "";

CourseName = "";

CourseId = -1;

Attendance = 0;

Hometask = 0;

ListOfStudents = new List<StudentClass>();

}

public int[] ShowAttendance()

{

int[] ints = new int[4];

return ints;

}

public int[] ShowHometask()

{

int[] ints = new int[4];

return ints;

}

}

}

Lesson

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ExamMasters

{

public class Lesson

{

public int Id;

public string Name;

public string Video;

public string Hometask;

public DateTime Date;

public string GroupName;

public int Attendance;

public int HometaskProcent;

public Lesson(int id, string theme, string video, string hometask)

{

this.Id = id;

this.Name = theme;

this.Video = video;

this.Hometask = hometask;

}

public Lesson(int id)

{

this.Id = id;

this.Name = "";

this.Video = "";

this.Hometask = "";

}

}

}

Test

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ExamMasters

{

internal class Test

{

public int Id;

public string Name;

public string Type;

public string Time;

public Test(int id, string name, string type, string time)

{

this.Id = id;

this.Name = name;

this.Type = type;

this.Time = time;

}

public Test(int id)

{

this.Id = id;

this.Name = "";

this.Type = "";

this.Time = "";

}

}

}

## Классы для работы с данными и БД

### Message

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace ExamMasters

{

internal class Message

{

public static string Caption;

public static string Text;

public Message()

{

Caption = "";

Text = "";

}

public Message(string caption, string text)

{

Caption = caption;

Text = text;

}

public static void Show(string caption, string text)

{

MessageBox.Show(text, caption);

}

}

}

DBConnection

namespace ExamMasters

{

internal class DBConnection

{

static MySqlConnection connection = new MySqlConnection("server=localhost; username=root; pwd=NiKa10042002; database=exammasters");

public void OpenConnection()

{

if (connection.State == System.Data.ConnectionState.Closed)

{

connection.Open();

}

}

public void CloseConnection()

{

if (connection.State == System.Data.ConnectionState.Open)

{

connection.Close();

}

}

public MySqlConnection GetConnection()

{

return connection;

}

}

}

DB\_Do

internal class DB\_Do : DBConnection

{

public static DBConnection db = new DBConnection();

public static MySqlCommand command;

//

// Поиск ID по электронной почте (Ученик)

public static int GetStudent\_ID\_ByEmail(string email)

{

string student\_id;

db.OpenConnection();

string sql = "SELECT ID FROM student WHERE email=" + "'" + email + "'";

command = new MySqlCommand(sql, db.GetConnection());

if (command.ExecuteScalar() == null)

{

student\_id = "-1";

}

else

{

student\_id = command.ExecuteScalar().ToString();

}

db.CloseConnection();

return Convert.ToInt32(student\_id);

}

//

// Поиск пароля по ID (Ученик)

public static string GetStudent\_Password\_ByID(int id)

{

string student\_password = "";

db.OpenConnection();

string sql = "SELECT password FROM student WHERE ID=" + "'" + id + "'";

command = new MySqlCommand(sql, db.GetConnection());

if (command.ExecuteScalar() == null)

{

student\_password = "";

}

else

{

student\_password = command.ExecuteScalar().ToString();

}

db.CloseConnection();

return student\_password;

}

//

// Поиск ID по электронной почте (Преподаватель)

public static int GetTeacher\_ID\_ByEmail(string email)

{

string teacher\_id;

db.OpenConnection();

string sql = "SELECT ID FROM teacher WHERE email=" + "'" + email + "'";

command = new MySqlCommand(sql, db.GetConnection());

if (command.ExecuteScalar() == null)

{

teacher\_id = "-1";

}

else

{

teacher\_id = command.ExecuteScalar().ToString();

}

db.CloseConnection();

return Convert.ToInt32(teacher\_id);

}

//

// Поиск ID по ФИО (Преподаватель)

public static int GetTeacher\_ID\_ByFIO(string name)

{

string teacher\_id;

db.OpenConnection();

string sql = "SELECT ID FROM teacher WHERE FIO=" + "'" + name + "'";

command = new MySqlCommand(sql, db.GetConnection());

if (command.ExecuteScalar() == null)

{

teacher\_id = "-1";

}

else

{

teacher\_id = command.ExecuteScalar().ToString();

}

db.CloseConnection();

return Convert.ToInt32(teacher\_id);

}

//

// Поиск пароля по ID (Преподаватель)

public static string GetTeacher\_Password\_ByID(int id)

{

string teacher\_password = "";

db.OpenConnection();

string sql = "SELECT password FROM teacher WHERE ID=" + "'" + id + "'";

command = new MySqlCommand(sql, db.GetConnection());

if (command.ExecuteScalar() == null)

{

teacher\_password = "";

}

else

{

teacher\_password = command.ExecuteScalar().ToString();

}

db.CloseConnection();

return teacher\_password;

}

//

// Установление пароля (Ученик)

public static void SetStudentPassword(int id, string password)

{

db.OpenConnection();

string proc\_view = "SetPasswordStudent";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

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

command.Parameters.AddWithValue("newpassword", password);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// Установление пароля (Преподаватель)

public static void SetTeacherPassword(int id, string password)

{

db.OpenConnection();

string proc\_view = "SetPasswordTeacher";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

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

command.Parameters.AddWithValue("newpassword", password);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// ДОБАВЛЕНИЕ ДАННЫХ. Добавление преподавателя

public static void AddTeacher(string name, string email, string phone)

{

db.OpenConnection();

string proc\_view = "AddTeacher";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("fio", name);

command.Parameters.AddWithValue("MyEmail", email);

command.Parameters.AddWithValue("MyPhone", phone);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// ДОБАВЛЕНИЕ ДАННЫХ. Добавление ученика

public static void AddStudent(string name, string email, string phone, int gr\_id)

{

db.OpenConnection();

string proc\_view = "AddStudent";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("fio", name);

command.Parameters.AddWithValue("MyEmail", email);

command.Parameters.AddWithValue("MyPhone", phone);

command.Parameters.AddWithValue("MyGroupId", gr\_id);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// Вывод списка групп

public static List <Group> GetGroups()

{

List<Group> ListOfGroups = new List<Group>();

int ID;

string name;

db.OpenConnection();

string proc\_view = "GetGroupsList";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

ID = Convert.ToInt32(read["ID"]);

name = Convert.ToString(read["gr\_name"]);

Group group = new Group(ID, name);

ListOfGroups.Add(group);

}

}

read.Close();

db.CloseConnection();

return ListOfGroups;

}

//

// Поиск ID по названию группы

public static int GetGroup\_ID\_ByName(string gr\_name)

{

string id;

db.OpenConnection();

string sql = "SELECT ID FROM exammasters.group WHERE gr\_name=" + "'" + gr\_name + "'";

command = new MySqlCommand(sql, db.GetConnection());

id = command.ExecuteScalar().ToString();

db.CloseConnection();

return Convert.ToInt32(id);

}

//

// Поиск названия группы по ID

public static string GetGroupNameByID(int ID)

{

db.OpenConnection();

string proc\_view = "GetGroupNameByID";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

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

command.CommandType = CommandType.StoredProcedure;

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

db.CloseConnection();

return name;

}

//

// Вывод списка преподавателей

public static List<TeacherClass> GetTeachers()

{

List<TeacherClass> ListOfTeacher = new List<TeacherClass>();

int ID;

string name;

db.OpenConnection();

string proc\_view = "GetListOfTeacher";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

ID = Convert.ToInt32(read["ID"]);

name = Convert.ToString(read["FIO"]);

TeacherClass teacher = new TeacherClass(ID);

teacher.Name = name;

ListOfTeacher.Add(teacher);

}

}

read.Close();

db.CloseConnection();

return ListOfTeacher;

}

//

// ДОБАВЛЕНИЕ ДАННЫХ. Добавление группы

public static void AddGroup(string group\_name, string group\_start, string group\_finish, int group\_TeacherId)

{

db.OpenConnection();

string proc\_view = "AddGroup";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("group\_name", group\_name);

command.Parameters.AddWithValue("group\_start", group\_start);

command.Parameters.AddWithValue("group\_finish", group\_finish);

command.Parameters.AddWithValue("group\_TeacherId", group\_TeacherId);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// Вывод списка курсов

public static List<Course> GetCourses()

{

List<Course> ListOfCourses = new List<Course>();

int ID;

string name = "";

db.OpenConnection();

string proc\_view = "GetKursNAME ";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

ID = Convert.ToInt32(read["ID"]);

name = Convert.ToString(read["name"]);

Course course = new Course(ID, name);

ListOfCourses.Add(course);

}

}

read.Close();

db.CloseConnection();

return ListOfCourses;

}

//

// ДОБАВЛЕНИЕ ДАННЫХ. Добавление курса

public static void AddCourse(string kurs\_name, string kurs\_descr, int kurs\_cost, int kurs\_lessoncol)

{

db.OpenConnection();

string proc\_view = "AddKurs";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("kurs\_name", kurs\_name);

command.Parameters.AddWithValue("kurs\_descr", kurs\_descr);

command.Parameters.AddWithValue("kurs\_cost", kurs\_cost);

command.Parameters.AddWithValue("kurs\_lessoncol", kurs\_lessoncol);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// Поиск ID по наименованию курса

public static int GetCourse\_ID\_ByName(string name)

{

int id = -1;

db.OpenConnection();

string sql = "SELECT ID FROM kurs WHERE name=" + "'" + name + "'";

command = new MySqlCommand(sql, db.GetConnection());

id = Convert.ToInt32(command.ExecuteScalar());

db.CloseConnection();

return id;

}

//

// ДОБАВЛЕНИЕ ДАННЫХ. Добавление урока

public static void AddLesson(int lesson\_kurs\_id, string lesson\_date, string lesson\_name, string lesson\_video, string lesson\_hometask, int lesson\_teacher\_id)

{

db.OpenConnection();

string proc\_view = "AddLesson";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("lesson\_kurs\_id", lesson\_kurs\_id);

command.Parameters.AddWithValue("lesson\_date", lesson\_date);

command.Parameters.AddWithValue("lesson\_name", lesson\_name);

command.Parameters.AddWithValue("lesson\_video", lesson\_video);

command.Parameters.AddWithValue("lesson\_hometask", lesson\_hometask);

command.Parameters.AddWithValue("lesson\_teacher\_id", lesson\_teacher\_id);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

public static void AddTest(string test\_name, string test\_type, string test\_time, int test\_kurs\_id)

{

db.OpenConnection();

string proc\_view = "AddTest";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("test\_name", test\_name);

command.Parameters.AddWithValue("test\_type", test\_type);

command.Parameters.AddWithValue("test\_time", test\_time);

command.Parameters.AddWithValue("test\_kurs\_id", test\_kurs\_id);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// УДАЛЕНИЕ ДАННЫХ. Удаление преподавателя.

public static void DeleteTeacher(int teacher\_id)

{

db.OpenConnection();

string proc\_view = "DeleteTeacher";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("teacher\_id", teacher\_id);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// Поиск ID ученика по ФИО

public static int GetStudent\_ID\_ByName(string student\_name)

{

db.OpenConnection();

string proc\_view = "GetStudentIdFIO";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("MyFIO", student\_name);

command.CommandType = CommandType.StoredProcedure;

int id = Convert.ToInt32(command.ExecuteScalar());

db.CloseConnection();

return id;

}

//

// УДАЛЕНИЕ ДАННЫХ. Удаление ученика.

public static void DeleteStudent(int student\_id)

{

db.OpenConnection();

string proc\_view = "DeleteStudent";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("student\_id", student\_id);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// Вывод списка учеников

public static List<StudentClass> GetStudents()

{

List<StudentClass> ListOfStudents = new List<StudentClass>();

int ID;

string name = "";

db.OpenConnection();

string proc\_view = "GetListOfStudents";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

ID = Convert.ToInt32(read["ID"]);

name = Convert.ToString(read["FIO"]);

StudentClass student = new StudentClass(ID);

student.Name = name;

ListOfStudents.Add(student);

}

}

read.Close();

db.CloseConnection();

return ListOfStudents;

}

//

// УДАЛЕНИЕ ДАННЫХ. Удаление группы.

public static void DeleteGroup(int group\_id)

{

db.OpenConnection();

string proc\_view = "DeleteGroup";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("group\_id", group\_id);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// УДАЛЕНИЕ ДАННЫХ. Удаление курса.

public static void DeleteCourse(int kurs\_id)

{

db.OpenConnection();

string proc\_view = "DeleteKurs";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("kurs\_id", kurs\_id);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// Формирование таблицы занятий по ID курса

public static List<Lesson> GetLessons(int id)

{

List<Lesson> ListOfLessons = new List<Lesson>();

int ID;

string name = "";

db.OpenConnection();

string proc\_view = "GetLessonsKursID";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

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

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

ID = Convert.ToInt32(read["ID"]);

name = Convert.ToString(read["name"]);

Lesson lesson = new Lesson(ID);

lesson.Name = name;

ListOfLessons.Add(lesson);

}

}

read.Close();

db.CloseConnection();

return ListOfLessons;

}

//

// УДАЛЕНИЕ ДАННЫХ. Удаление занятия.

public static void DeleteLesson(int lesson\_id)

{

db.OpenConnection();

string proc\_view = "DeleteLesson";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("lesson\_id", lesson\_id);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// Поиск ID занятия по названию

public static int GetLessonId\_ByName(string LessonName)

{

db.OpenConnection();

string proc\_view = "GetLessonIdNAME";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

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

command.CommandType = CommandType.StoredProcedure;

int id = Convert.ToInt32(command.ExecuteScalar());

db.CloseConnection();

return id;

}

//

// Формирование таблицы тестов по ID курса

public static List<Test> GetTest(int id)

{

List<Test> ListOfTests= new List<Test>();

int ID;

string name = "";

db.OpenConnection();

string proc\_view = "GetTestsKursID";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

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

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

ID = Convert.ToInt32(read["id"]);

name = Convert.ToString(read["name"]);

Test test = new Test(ID);

test.Name = name;

ListOfTests.Add(test);

}

}

read.Close();

db.CloseConnection();

return ListOfTests;

}

//

// Поиск ID тестирования по названию

public static int GetTestId\_ByName(string KTName)

{

db.OpenConnection();

string proc\_view = "GetKTIdNAME";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

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

command.CommandType = CommandType.StoredProcedure;

int id = Convert.ToInt32(command.ExecuteScalar());

db.CloseConnection();

return id;

}

//

// УДАЛЕНИЕ ДАННЫХ. Удаление тестирования.

public static void DeleteTest(int test\_id)

{

db.OpenConnection();

string proc\_view = "DeleteTest";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("test\_id", test\_id);

command.CommandType = CommandType.StoredProcedure;

command.ExecuteNonQuery();

db.CloseConnection();

}

//

// ОТЧЕТНЫЕ СТАТИСТИКИ. Вывод статистики по преподавателям

public static List<TeacherClass> GetAllStatTeacher()

{

List<TeacherClass> teachers= new List<TeacherClass>();

int ID;

db.OpenConnection();

string proc\_view = "AllStatTeacher";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

ID = Convert.ToInt32(read["ID"]);

TeacherClass teacher = new TeacherClass(ID);

teacher.Name = Convert.ToString(read["FIO"]);

teacher.Phone = Convert.ToString(read["phone\_number"]);

teacher.Email = Convert.ToString(read["email"]);

teacher.Attendance = Convert.ToDouble(read["avg(exammasters.group\_stat.attendance)"]);

teacher.Hometask = Convert.ToDouble(read["avg(exammasters.group\_stat.hometask)"]);

teachers.Add(teacher);

}

}

read.Close();

db.CloseConnection();

return teachers;

}

public static List<Group> GetGroupsOfTeacher(int id)

{

List<Group> groups = new List<Group>();

int ID;

db.OpenConnection();

string proc\_view = "GetGroupsTeacherID";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

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

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

ID = Convert.ToInt32(read["ID"]);

Group group = new Group(ID);

group.Name = Convert.ToString(read["gr\_name"]);

groups.Add(group);

}

}

read.Close();

db.CloseConnection();

return groups;

}

//

// ОТЧЕТНЫЕ СТАТИСТИКИ. Вывод статистики по ученикам

public static List<StudentClass> GetAllStatStudent()

{

List<StudentClass> students = new List<StudentClass>();

db.OpenConnection();

string proc\_view = "AllStatsStudents";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

StudentClass student = new StudentClass();

student.Name = Convert.ToString(read["FIO"]);

student.Phone = Convert.ToString(read["phone\_number"]);

student.GroupID = Convert.ToInt32(read["ID"]);

student.TeacherName = Convert.ToString(read["teacher\_FIO"]);

student.Attendance = Convert.ToDouble(read["attendance"]);

student.Hometask= Convert.ToDouble(read["hometask"]);

students.Add(student);

}

}

read.Close();

db.CloseConnection();

return students;

}

//

// ОТЧЕТНЫЕ СТАТИСТИКИ. Вывод статистики по группам

public static List<Group> AllStatGroup()

{

List<Group> groups = new List<Group>();

int group\_ID;

db.OpenConnection();

string proc\_view = "AllStatGroup";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

group\_ID = Convert.ToInt32(read["group\_ID"]);

Group group = new Group(group\_ID);

group.TeacherName = Convert.ToString(read["FIO"]);

group.CourseId = Convert.ToInt32(read["ID"]);

group.Attendance = Convert.ToDouble(read["attendance"]);

group.Hometask = Convert.ToDouble(read["hometask"]);

groups.Add(group);

}

}

read.Close();

db.CloseConnection();

for (int i = 0; i < groups.Count; i++)

{

groups[i].Name = GetGroupNameByID(groups[i].Id);

groups[i].CourseName = GetCourseName\_ByID(groups[i].CourseId);

}

return groups;

}

public static string GetCourseName\_ByID(int id)

{

string courseName = "";

string proc\_view = "GetKursNAME\_ByID";

db.OpenConnection();

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

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

command.CommandType = CommandType.StoredProcedure;

courseName = Convert.ToString(command.ExecuteScalar());

db.CloseConnection();

return courseName;

}

//

// Составление таблицы групп по ID преподавателя

public static List<Group> GetGroups\_ByTeacherID(int id)

{

List<Group> groups = new List<Group>();

db.OpenConnection();

string proc\_view = "GetGroupsTeacherID";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

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

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

Group group = new Group(id);

group.Name = Convert.ToString(read["gr\_name"]);

groups.Add(group);

}

}

read.Close();

db.CloseConnection();

return groups;

}

//

// Формирование информации об учениках группы

public static List<StudentClass> GetListOfStudentsInGroup(int id)

{

List<StudentClass> students = new List<StudentClass>();

db.OpenConnection();

string proc\_view = "GetListOfStudentsInGroup";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

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

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

StudentClass student = new StudentClass(Convert.ToInt32(read["ID"]));

student.Name = Convert.ToString(read["FIO"]);

student.Email = Convert.ToString(read["email"]);

student.Phone = Convert.ToString(read["phone\_number"]);

students.Add(student);

}

}

read.Close();

db.CloseConnection();

return students;

}

public static List<Lesson> GetAllLessonsByTeacherId(int id)

{

List<Lesson> lessons = new List<Lesson>();

db.OpenConnection();

string proc\_view = "GetAllLessonsByTeacherId";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("teacher\_id", id);

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

Lesson lesson = new Lesson(0);

lesson.Name = Convert.ToString(read["name"]);

lesson.Video = Convert.ToString(read["video"]);

lesson.Hometask = Convert.ToString(read["hometask"]);

lesson.Date = DateTime.Parse(Convert.ToString(read["date"]));

lesson.GroupName = Convert.ToString(read["gr\_name"]);

lessons.Add(lesson);

}

}

read.Close();

db.CloseConnection();

return lessons;

}

public static List<Lesson> StudentGetAttendanceHomework(int st\_id)

{

List<Lesson> statistic = new List<Lesson>();

db.OpenConnection();

string proc\_view = "StudentGetAttendanceHomework";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("st\_id", st\_id);

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

Lesson lesson = new Lesson(0);

lesson.Name = Convert.ToString(read["name"]);

lesson.Attendance = Convert.ToInt32(read["attendance"]);

lesson.HometaskProcent = Convert.ToInt32(read["hometask"]);

lesson.Date = DateTime.Parse(Convert.ToString(read["date"]));

statistic.Add(lesson);

}

}

read.Close();

db.CloseConnection();

return statistic;

}

public static List<Lesson> GetAllLessonsByStudentId(int student\_id)

{

List<Lesson> lessons = new List<Lesson>();

db.OpenConnection();

string proc\_view = "GetAllLessonsByStudentId";

MySqlCommand command = new MySqlCommand(proc\_view, db.GetConnection());

command.Parameters.AddWithValue("student\_id", student\_id);

command.CommandType = CommandType.StoredProcedure;

MySqlDataReader read = command.ExecuteReader();

if (read.HasRows)

{

while (read.Read())

{

Lesson lesson = new Lesson(0);

lesson.Name = Convert.ToString(read["name"]);

lesson.Video = Convert.ToString(read["video"]);

lesson.Hometask = Convert.ToString(read["hometask"]);

lesson.Date = DateTime.Parse(Convert.ToString(read["date"]));

lesson.GroupName = Convert.ToString(read["kurs\_name"]);

lessons.Add(lesson);

}

}

read.Close();

db.CloseConnection();

return lessons;

}

}