Частное учреждение образования

«Колледж бизнеса и права»

РАЗРАБОТКА СИСТЕМЫ ФОРМИРОВАНИЯ ОТЧЁТОВ ПО ОТРАБОТАННЫМ ЧАСАМ ПРЕПОДОАВАТЕЛЕЙ ЧУО «КОЛЛЕДЖ БИЗНЕСА И ПРАВА»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по учебному предмету

«Конструирование программ и языки программирования»

КП Т.292001.401

Руководитель проекта ( Е.Н.Коропа )

Обучающийся ( Д.Р.Буев )

2025

**Этикетка**

для курсовых проектов

**Курсовой проект**

Тема Разработка системы формирования отчётов по отработанным часам преподавателей ЧУО «Колледж бизнеса и права»

КП Т.292001.401

Разработан Буев Д.Р.

Утвержден Коропа Е.Н.

Руководитель: Коропа Е.Н.

Технические средства: ПК с ЦПУ частотой от 2 ГГц и ОП от 4 ГБ, монитор, клавиатура, мышь

Программные средства: ОС Microsoft Windows 11 Pro, Microsoft Visual Studio,

.NET 9.0, .NET Framework 4.7.2, SQL Server Management Studio 2022, Microsoft SQL Server 2022

**Состав документа:**

Пояснительная записка – Буев ПЗ.docx

Графическая часть – Курсовая ДВИ.docx, Курсовая Классы.docx, Курсовая Деятельности.docx

Папка с проектом – Kursach2.0

Установочный пакет программного средства – Setup.msi

Сведения о защите информации: логин Коропа Е.Н., пароль secretPassword

СОДЕРЖАНИЕ

*Изм.*

*Изм.*

*Изм.*

*Изм.*

*Изм.*

*Изм.*

*Изм.*

*Изм.*

*Лист*

*Лист*

*Лист*

*Лист*

*Лист*

*Лист*

*Лист*

*Лист*

*№ докум.*

*№ докум.*

*№ докум.*

*№ докум.*

*№ докум.*

*№ докум.*

*№ докум.*

*№ докум.*

*Подпись*

*Подпись*

*Подпись*

*Подпись*

*Подпись*

*Подпись*

*Подпись*

*Подпись*

*Дата*

*Дата*

*Дата*

*Дата*

*Дата*

*Дата*

*Дата*

*Дата*

*Лист*

*Лист*

*Лист*

*Лист*

*Лист*

*Лист*

*Лист*

*Лист*

*3*

*3*

*3*

*3*

*2*

*3*

*3*

*3*

*КП Т.292001.401 ПЗ*

*КП Т.716007.401 ПЗ*

*КП Т.716009.401 ПЗ*

*КП Т.716007.401 ПЗ*

*ОП Т.893023*

*КП Т.716007.401 ПЗ*

*КП Т.716009.401 ПЗ*

*КП Т.716007.401 ПЗ*

*Разраб.*

*Разраб.*

*Разраб.*

*Разраб.*

*Разраб.*

*Разраб.*

*Разраб.*

*Разраб.*

*Буев Д.Р.*

*Провер.*

*Провер.*

*Провер.*

*Провер.*

*Провер.*

*Провер.*

*Провер.*

*Провер.*

*Коропа Е.Н.*

*Т. контр.*

*Т. контр.*

*Т. контр.*

*Т. контр.*

*Т. контр.*

*Т. контр.*

*Т. контр.*

*Т. контр.*

*Н. контр.*

*Н. контр.*

*Н. контр.*

*Н. контр.*

*Н. контр.*

*Н. контр.*

*Н. контр.*

*Н. контр.*

*Утверд.*

*Утверд.*

*Утверд.*

*Утверд.*

*Утверд.*

*Утверд.*

*Утверд.*

*Утверд.*

*Коропа Е.Н.*

*РАЗРАБОТКА СИСТЕМЫ ФОРМИРОВАНИЯ ОТЧЕТОВ ПО ОТРАБОТАННЫМ ЧАСАМ ПРЕПОДАВАТЕЛЕЙ ЧУО «КОЛЛЕДЖ БИЗНЕСА И ПРАВА»*

*Лит.*

*У*

*Лит.*

*У*

*Лит.*

*У*

*Лит.*

*У*

*Лит.*

*У*

*Лит.*

*У*

*Лит.*

*У*

*Лит.*

*У*

*Листов*

*Листов*

*Листов*

*Листов*

*Листов*

*Листов*

*Листов*

*Листов*

*30*

*55*

*50*

*50*

*55*

*50*

*КБП*

*КБП*

*КБП*

*КБП*

*КБП*

*КБП*

*КБП*

*КБП*

[Введение 4](#_Toc198826740)

[1 Описание задачи 6](#_Toc198826741)

[1.1 Анализ предметной области 6](#_Toc198826742)

[1.2 Постановка задачи 7](#_Toc198826743)

[2 Проектирование системы 8](#_Toc198826744)

[2.1 Требования к приложению 8](#_Toc198826745)

[2.2 Проектирование задачи 8](#_Toc198826746)

[2.3 Концептуальный прототип 11](#_Toc198826747)

[3 Описание реализации программного средства 12](#_Toc198826748)

[3.1 Инструменты разработки и применяемые технологии 12](#_Toc198826749)

[3.2 Порядок авторизации пользователей 13](#_Toc198826750)

[3.3 Организация данных 13](#_Toc198826751)

[3.4 Функции: логическая и физическая организация 16](#_Toc198826752)

[3.5 Входные и выходные данные 19](#_Toc198826753)

[3.6 Функциональное тестирование 19](#_Toc198826754)

[3.7 Описание справочной системы 24](#_Toc198826755)

[4 Применение 27](#_Toc198826756)

[4.1 Назначение программного средства 27](#_Toc198826757)

[4.2 Условия применения 27](#_Toc198826758)

[Заключение 29](#_Toc198826759)

[Список использованных источников 30](#_Toc198826760)

[ПРИЛОЖЕНИЕ А Текст программных модулей 31](#_Toc198826761)

[ПРИЛОЖЕНИЕ Б Формы входных и выходных документов 80](#_Toc198826762)

[ПРИЛОЖЕНИЕ В Результаты работа программы 81](#_Toc198826763)

[ПРИЛОЖЕНИЕ Г Прототипы оконного интерфейса 91](#_Toc198826764)

# Введение

В процессе деятельности образовательных учреждений важной задачей является учет и анализ отработанных часов преподавателей. Эффективное решение этой задачи способствует прозрачности учета, оптимизации планирования и формирования отчетной документации. Основной целью разработки является создание системы формирования отчетов, которая автоматизирует процесс учета и упрощает составление отчетов об отработанных часах.

Актуальность проекта обусловлена необходимостью перехода на современные цифровые технологии в образовательной среде. Автоматизация процессов учета позволяет сократить временные затраты, минимизировать ошибки при вводе данных и обеспечить удобство в эксплуатации системы. Таким образом, данная система имеет значимое практическое применение и является важным этапом совершенствования учебной и административной деятельности.

При разработке курсового проекта будет составлена пояснительная записка, содержащая четыре раздела.

В первом разделе «Описание задачи» рассматривается предметная область и формулируется основная задача, которую необходимо решить в рамках разработки. Особое внимание уделяется анализу текущих аналогов, а также периодичности использования системы.

Во втором разделе «Проектирование системы» приводятся требования к функциональности приложения и осуществляется моделирование архитектуры программного средства. Также описывается структура базы данных и представляется концептуальный прототип пользовательского интерфейса.

Третий раздел «Описание реализации программного средства» посвящен выбору инструментов разработки и технологий, описанию структуры данных и реализации ключевых функций. Включены аспекты логической и физической организации данных, тестирование системы, а также описание справочной документации.

Четвертый раздел «Применение» содержит информацию о назначении системы и условиях её использования, демонстрируя области её применения и влияние на организационные процессы.

Заключение представляет собой анализ созданного программного средства, сопоставление поставленной задачи с достигнутым результатом и обзор возможных несоответствий, если таковые имеются.

Приложение А содержит текст программы.

Приложение Б содержит пример формы выходного документа.

Приложение В содержит изображения с результатами работы программы.

Приложение Г содержит изображения прототипов оконного интерфейса.

В графической части представлены диаграммы вариантов использования, классов и деятельности.

# Описание задачи

## Анализ предметной области

В рамках данной системы исследуются процессы учёта рабочих часов преподавателей, включающие:

* регистрация преподавателей для авторизованного доступа к системе;
* получение информации (импорт) о расписании занятий преподавателей с официального сайта учреждения;
* учет фактического времени работы, а также возможность редактирования данных для исправления ошибок. Преподаватель либо загружает расписание с сайта колледжа, либо вводит его вручную. В результате данные о проведённых занятиях поступают в систему и сохраняются в базе данных, создавая основу для дальнейшего учёта. Каждое занятие фиксируется в базе данных, включая дату, время, дисциплину и количество часов. Это позволяет системе точно отслеживать преподавательскую нагрузку и формировать отчетность без необходимости ручного подсчета. Когда преподаватель переходит на страницу составления отчёта и нажимает кнопку «Составить отчёт», приложение автоматически анализирует сохранённые данные, подсчитывает количество отработанных часов по каждому предмету за определённый период и формирует отчет. После обработки данных приложение генерирует документ в формате Word, в котором структурировано представлена информация о преподавательской нагрузке.
* составление отчетов о рабочих часах, распределенных по месяцам и предметам. Эти процессы являются ключевыми для административной работы образовательного учреждения и должны быть упрощены с помощью автоматизированной системы.

Промежуточные данные формируются путём автоматического извлечения расписания с сайта колледжа и сохранением его в базе данных для последующей обработки.

Итоговые данные формируются путём подсчёта отработанных часов за каждую дату по каждому предмету, подсчёта общего количества отработанных часов по каждому предмету, подсчёта оставшихся часов каждого предмета и экспорт отчёта в формат Word.

Данное программное средство позволит минимизировать ошибки, увеличить скорость обработки и обеспечит удобный интерфейс для работы с большими объёмами данных.

Целевой аудиторией данной системы являются преподаватели ЧУО «Колледж бизнеса и права».

## Постановка задачи

На основе анализа бизнес-процессов и задач, связанных с учетом и анализом рабочего времени преподавателей, выделены следующие функции, которые подлежат автоматизации в разрабатываемой системе:

* авторизация и регистрация пользователей. Реализация надежного механизма для регистрации новых преподавателей с сохранением конфиденциальности их данных. Авторизация преподавателей с проверкой введенных учетных данных. Возможность изменения пароля для повышения безопасности учетной записи;
* извлечение и управление расписанием. Автоматический парсинг расписания преподавателей с официального сайта колледжа. Сохранение расписания в базе данных для анализа и последующего использования. Интерфейс для просмотра и редактирования расписания с удобной навигацией;
* учет и корректировка отработанных часов. Ввод данных о часах, отработанных преподавателями. Возможность добавления, удаления и изменения введенной информации для обеспечения точности данных. Поддержка фильтрации данных по предмету, преподавателю и дате;
* формирование отчетов. Генерация отчетов об отработанных часах по каждому преподавателю за заданный период. Экспорт отчетов в удобные форматы. Расчет итоговых данных, таких как количество часов, отработанных по предметам;
* интерфейс и справочная система. Лаконичный и удобный интерфейс для взаимодействия с системой, включающий меню, кнопочные формы и панели инструментов. Создание справочной системы для объяснения функционала программы и упрощения работы с системой.

На рынке существуют различные аналоги, обеспечивающие учет рабочего времени преподавателей, например Excel, Google Sheets или CRM-системы Bitrix24 и Asana, но ни один из перечисленных аналогов не соответствует требованиям ЧУО «Колледж бизнеса и права».

Предполагаемая система будет интегрирована с сайтом колледжа для автоматического извлечения расписания, чего не предполагают вышеуказанные системы. Система будет поддерживать подсчёт отработанных часов и экспорт их в Word. Также в системе будет присутствовать функционал для редактирования и фильтрации данных. Система будет иметь интуитивно понятный интерфейс и встроенную справочную систему для упрощения работы с программным средством.

# Проектирование системы

## Требования к приложению

В разрабатываемой системе требуется реализовать шифрование паролей с помощью хеширования. Необходимо реализовать механизм авторизации с использованием уникальных учетом данных. Интерфейс должен быть лаконичным, без лишнего визуального шума. Все ключевые функции должны быть доступны за минимальное количество действий. Размеры текстовых полей должны быть ограничены для предотвращения ошибок или перегрузки системы.

Элементы управления должны быть расположены логически, с разделением по функциональным блокам. Кнопки и текстовые поля должны быть достаточно крупными для удобного взаимодействия. Необходимо использовать шрифт Arial размером 16 пикселей. Цветовая палитра должна включать в себя преимущественно нейтральные тона (белый, темно-серый).

Для использования приложения требуется операционная система Windows 11, объёмом памяти минимум 2 ГБ. Необходим доступ к интернету, а также установленная платформа .NET Framework версии 4.7.2 или выше. Также необходимо установить Word и систему управления базами данных (СУБД) SQL Server версии Developer.

Для установки программы будет создано установочное средство.

## Проектирование задачи

Для отражения предметной области была спроектирована ER-диаграмма, которая описывает ключевые сущности и их взаимосвязи.

Основные сущности системы:

* преподаватели;
* предметы;
* проведённые часы.

Сущность «Teacher» имеет атрибуты «Login», «Password» и «Url» для извлечения расписания с сайта колледжа.

Сущность «Subject» имеет атрибуты «Name», «TeacherName» и «CountOfHours».

Сущность «Hours» имеет атрибуты «SubjectName», «Date» и «TeacherName».

Связи между преподавателем и проведёнными часами, а также между предметом и проведёнными часами позволяет преподавателю проводить по несколько предметов в один день.

Диаграмма сущность-связь представлена на рисунке 2.1.

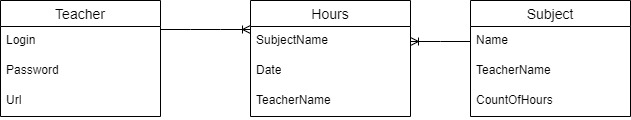


Рисунок 2.1 – Диаграмма сущность-связь

Функциональная модель данных разрабатываемого программного средства представлена в виде диаграммы вариантов использования, смоделированной на унифицированном языке моделирования UML средствами IBM Rational Rose Enterprise Edition, которая предназначена для определения общих границ и контекста моделируемой предметной области на начальных этапах проектирования программы.

Главной целью проектирования моделей является отображение функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями.

В рамках языка UML все представления о модели сложной системы фиксируются в виде специальных графических конструкций – диаграмм. В терминах языка UML определены следующие виды диаграмм: диаграмма вариантов использования, диаграмма классов.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами системы.

Актер представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для достижения определенных целей или решения частных задач.

В данной проектируемой системе в качестве актеров выступают системный администратор и пользователь, которые служат источниками воздействия на моделируемую систему.

Разрабатываемая система будет представлять возможность пользователю выполнять следующие действия:

* авторизоваться;
* зарегистрироваться;
* автоматически извлечь расписание с сайта колледжа;
* сохранить извлечённую информацию в файл;
* сохранить извлечённую информацию в базу данных;
* просмотреть расписание (посредством нажатия на кнопки < и >);
* изменить пароль;
* изменить ссылку для парсинга расписания;
* сформировать месячный отчёт по отработанным часам, включая подсчёт отработанных часов за каждую дату;
* экспорт отчёта в Word;
* добавить информацию об отработанных часах;
* удалить информацию об отработанных часах;
* изменить информацию об отработанных часах;
* отфильтровать данные об отработанных часах по предмету, дате и группе;
* просмотреть справку.

Подробное описание данных функций представлено на диаграмме вариантов использования, представленной в графической части на листе 1.

В ходе разработки программного средства были выделены следующие классы:

* класс Teachers – предназначен для хранения данных о преподавателе (логин, пароль, ссылка для загрузки расписания с сайта колледжа). Аналогичными классами являются Hours, AllHours, Subjects, Dates;
* класс TeachersRepository – служит для работы с таблицей базы данных Teachers. Аналогичными классами являются HoursRepository, DatesRepository, SubjectsRepository, AllHoursRepository;
* класс TeachAuth – служит для авторизации преподавателей. Он проверяет на соответствие логин и пароль, который преподаватель введёт при входе;
* класс TeachReg – служит для регистрации преподавателей. Он проверяет на уникальность логин, который введёт преподаватель, и , если он уникален, то зарегистрирует его;
* класс WordReport – служит для записи отчёта в Word;
* класс WeekSubjects – необходим для хранения недельного расписания преподавателя. Аналогичным классом является класс AllHoursSubjects;
* класс PasswordHasher – служит для шифрования паролей;
* класс HtmlParser – служит для загрузки недельного расписания преподавателя с сайта колледжа.

Детальное описание данных классов представлено на диаграмме классов в графической части на листе 2.

Также во время разработки программы была построена диаграмма деятельности, на которой отражается процесс авторизации преподавателей. Описание данного процесса приведено ниже.

Преподаватель вводит логин и пароль. Затем система проверяет введённые данные на соответствие. Если данные неверны, то система оповещает об этом преподавателя и просит его ввести другие данные. Если введённые данные верны, то система загружает данные преподавателя и перенаправляет его на главную страницу приложения.

Диаграмма деятельности представлена в графической части на листе 3.

## Концептуальный прототип

Система меню должна содержать пункты «Расписание», «Часы», «Отчёт», «Профиль» и «Справка». Основной функцией меню является быстрый доступ к ключевым компонентам системы. Меню должно быть реализовано в виде горизонтальной панели, при нажатии на которую система переключает страницы.

Контекстное меню должно реализовано на страницах и предоставлять быстрый доступ к функциям добавления, удаления и редактирования.

Диалоговое окно должно открываться при добавлении учебного часа для запроса на ввод названия предмета, группы, даты и количества часов. Также диалоговое окно должно выводиться при добавлении нового предмета для указания общего количества часов.

При создании должны быть использованы следующие элементы управления:

* кнопки, для взаимодействия с основными функциями (например парнсинг данных сайта, переход по недельному расписанию);
* текстовые поля, для ввода данных;
* списки выбора, для фильтрации;
* таблицы, для просмотра расписания и работы с базой данных;
* элемент управления для выбора даты, для выбора даты проведения предметов.

Прототипы страниц представлены в приложении Г на рисунках Г.1 – Г.7.

# Описание реализации программного средства

## Инструменты разработки и применяемые технологии

При разработке программного обеспечения для учета отработанных часов преподавателей были выбраны следующие технологии:

* среда разработки – Microsoft Visual Studio 2022;
* язык программирования – C#;
* фреймворк – WPF (Windows Presentation Foundation);
* система управления базами данных – Microsoft SQL Server 2022.

Microsoft Visual Studio поддерживает разработку программного обеспечения для различных платформ, включая десктопные приложения, веб-приложения, мобильные решения и облачные сервисы. Среда обеспечивает удобные механизмы работы с кодом, автоматизированную проверку, интеллектуальные подсказки и возможности рефакторинга. [8]

Выбор C# обусловлен его мощными возможностями для создания десктопных приложений, богатой экосистемой библиотек и удобством работы с данными. [5]

WPF используется для разработки современного и функционального графического интерфейса, позволяя создать удобное взаимодействие пользователя с системой, реализовать гибкость дизайна и поддержку сложных визуальных элементов. [9]

Microsoft SQL Server обеспечивает надежное хранение данных, высокую скорость обработки запросов и возможность структурированного анализа информации, что критически важно для работы с данными о преподавательской нагрузке. [6]

C# и WPF активно применяются в разработке корпоративных приложений, таких как финансовые системы, CRM и ERP-системы.

Microsoft SQL Server используется в банковской сфере, медицинских учреждениях и предприятиях для ведения учета данных, управления статистикой и формирования аналитических отчетов.

В данном проекте C# является основным языком для логики обработки данных, анализа информации и взаимодействия с базой данных. WPF применяется для отображения расписания, формирования отчетов и удобного интерфейса взаимодействия преподавателя с программой. Microsoft SQL Server служит хранилищем данных о занятиях, преподавательской нагрузке и результатах обработки информации.

Выбор данных технологий полностью оправдан задачами проекта: требуется интерактивный пользовательский интерфейс, надежная база данных и эффективные механизмы учета и отчетности. Использование C#, WPF и Microsoft SQL Server позволяет достичь высокой производительности, удобства работы с данными и расширяемости функционала приложения.

## Порядок авторизации пользователей

В данном программном продукте предусмотрена **роль преподавателя**. Он имеет доступ к следующим функциям:

* загрузка недельного расписания с сайта колледжа;
* управление базой данный (добавление, удаление и редактирование информации об отработанных часах);
* генерация месячного отчёта по отработанным часам;
* редактирование своего профиля (изменение пароля и ссылки для загрузки расписания с сайта колледжа).

Процесс аутентификации и авторизации обеспечивает безопасный доступ к функционалу приложения, гарантируя, что каждый преподаватель получает индивидуальные права доступа и может управлять только своими данными, исключая возможность несанкционированного редактирования информации другими преподавателями. Вход преподавателя в систему осуществляется посредством ввода логина и пароля. Пароли хранятся в базе данных в хешированном виде.

## Организация данных

В ходе разработки программного средства для хранения данных было принято решение использовать реляционную модель, в которой каждой сущности, рассмотренной на ERD диаграмме, соответствует своя таблица. Она позволяет эффективно управлять информацией о преподавателях и их отработанных часах.

В основе системы пять таблиц:

* таблица «Teachers» – служит для хранения данных о преподавателях. Детальное описание этой таблицы представлено в таблице 3.1;
* таблица «Subjects» – служит для хранения данных о предметах, которые ведёт преподаватель. Детальное описание этой таблицы представлено в таблице 3.2;
* таблица «AllHours» – служит для хранения общего количества часов по каждому предмету, который ведёт преподаватель. Детальное описание этой таблицы представлено в таблице 3.3;
* таблица «Hours» – служит для хранения информации о проведённых часах. Детальное описание этой таблицы представлено в таблице 3.4;
* таблица «Dates» – служит для хранения дат, по которым проходили предметы преподавателя. Детальное описание этой таблицы представлено в таблице 3.5.

Таблица 3.1 – Описание таблицы Teachers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Тип данных | Длина поля | Разрешение значения NULL | Описание поля |
| id | int | MAX | Нет | Уникальный идентификатор преподавателя |
| login | varchar | 50 | Нет | Логин преподавателя |
| password | varchar | 50 | Нет | Зашифрованный пароль преподавателя |
| url | varchar | MAX | Да | Ссылка для парсинга раписания преподавателя |

Таблица 3.2 – Описание таблицы Subjects

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Тип данных | Длина поля | Разрешение значения NULL | Описание поля |
| id | int | MAX | Нет | Уникальный идентификатор предмета |
| name | varchar | 50 | Нет | Название предмета |
| teacherId | int | MAX | Нет | Уникальный идентификатор преподавателя, который ведёт данный предмет |

Таблица 3.3 – Описание таблицы AllHours

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Тип данных | Длина поля | Разрешение значения NULL | Описание поля |
| id | int | MAX | Нет | Уникальный идентификатор предмета |
| subjectName | varchar | 50 | Нет | Название предмета |
| teacherId | int | MAX | Нет | Уникальный идентификатор преподавателя, который ведёт данный предмет |
| countOfHours | int | MAX | Нет | Общее количество часов предмета |

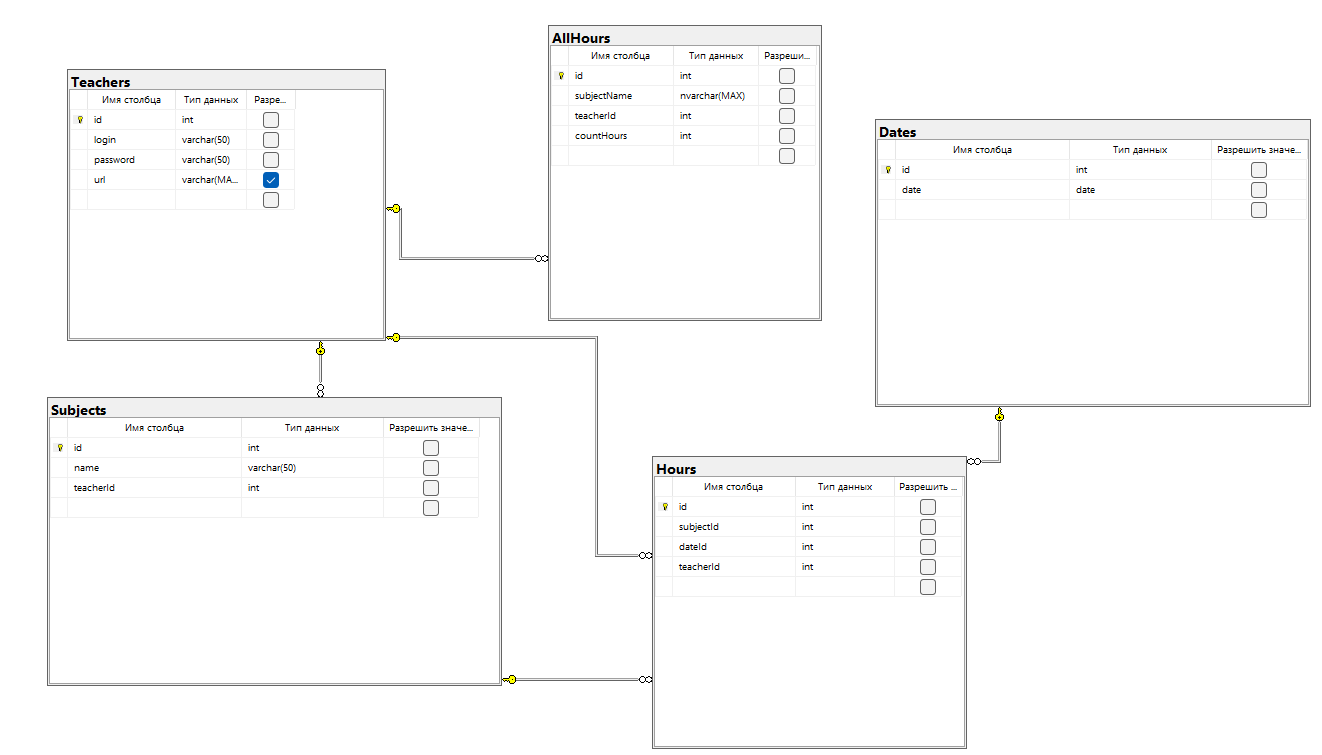
Таблица 3.4 – Описание таблицы Hours

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Тип данных | Длина поля | Разрешение значения NULL | Описание поля |
| id | int | MAX | Нет | Уникальный идентификатор отработанного часа |
| subjectId | int | MAX | Нет | Уникальный идентификатор предмета |
| dateId | int | MAX | Нет | Уникальный идентификатор даты |
| teacherId | int | MAX | Нет | Уникальный идентификатор преподавателя, который ведёт предмет |

Таблица 3.5 – Описание таблицы Dates

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название поля | Тип данных | Длина поля | Разрешение значения NULL | Описание поля |
| id | int | MAX | Нет | Уникальный идентификатор даты |
| date | date | MAX | Нет | Дата проведения предмета |

Схема базы данных представлена на рисунке 3.1.

Рисунок 3.1 – Схема базы данных

## Функции: логическая и физическая организация

В ходе разработки системы для формирования отчётов по отработанным часам преподавателя было разработано множество функций, из которых основными являются функции, представленные ниже.

Функция хеширования паролей, которая вызывается в окнах регистрации (авторизации) и предназначена для шифрования паролей. Код функции ниже.

public static ulong HashFunc(string str)

{

int key = 11; // Ключ для степени (можно изменить для получения других значений хеша)

int count = 9; // Определяет размер хеш-таблицы (10^count)

ulong size = (ulong)Math.Pow(10, count); // Вычисляем размер хеш-таблицы

ulong hash\_code, t\_hash = 0; // Переменные для хранения промежуточного и конечного хеша

for (int i = 0; i < str.Length; i++) // Проход по каждому символу строки

{

t\_hash += (ulong)Math.Pow(key, i) \* (ulong)str[i]; // Добавляем значение символа, умноженное на key^i

t\_hash %= size; // Обрезаем значение, чтобы избежать переполнения

}

hash\_code = t\_hash % size; // Получаем окончательный хеш, ограниченный размером таблицы

return hash\_code; // Возвращаем вычисленный хеш-код

}

Функция загрузки расписания с сайта колледжа, которая служит для парсинга расписания преподавателя и вызывается при нажатии на кнопку «Загрузить расписание». Код функции представлен ниже.

public async static Task<List<string>> GetDayOfWeek(string url, string subjectsTag, string groupsTag)

{

// Создаём конфигурацию для парсинга, используя встроенный загрузчик страниц

var config = Configuration.Default.WithDefaultLoader();

// Создаём новый контекст для обработки веб-страницы

var context = BrowsingContext.New(config);

// Асинхронно загружаем HTML-документ по указанному URL

var doc = await context.OpenAsync(url);

// Создаём список для хранения извлечённых данных (предмет + группа)

var day = new List<string>();

// Получаем список элементов, соответствующих тэгу subjectsTag (например, названия предметов)

var links1 = doc.QuerySelectorAll(subjectsTag);

// Получаем список элементов, соответствующих тэгу groupsTag (например, номера групп)

var links2 = doc.QuerySelectorAll(groupsTag);

// Проходим по извлечённым элементам и объединяем информацию о группе и предмете

for (int j = 0; j < links1.Length; j++)

{

day.Add($"{links2[j].TextContent} {links1[j].TextContent}"); // Формируем строку "Группа Предмет"

}

// Возвращаем список полученных данных

return day;

}

Функция составления отчётов по отработанным часам, которая вызывается при нажатии на кнопку «Составить отчёт» и служит для формирования отчёта по отработанным часам преподавателя за указанный им период. Код функции представлен ниже.

public void GenerateReport(int teacherId, List<List<string>> resultReport, string month, string year)

{

// Получаем путь к каталогу проекта и создаём папку "Отчёты"

string projectDir = AppDomain.CurrentDomain.BaseDirectory;

string dir = System.IO.Path.Combine(projectDir, "Отчёты");

// Настраиваем диалог для сохранения файла

var save = new SaveFileDialog

{

Title = "Сохранить файл как",

Filter = "Документы Word (\*.docx)|\*.docx",

FileName = $"Отчёт.docx",

InitialDirectory = dir

};

string file = $"Отчёт.docx";

// Если пользователь подтвердил сохранение

if (save.ShowDialog() == true)

{

file = save.FileName;

// Перебираем список отчётов и вычисляем количество вычтенных часов

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

{

int sum = 0;

for (int j = 1; j < resultReport[i].Count; j++)

{

string[] temp = resultReport[i][j].Split('-');

if (int.TryParse(temp[1].Trim(), out int num))

{

sum += num;

}

}

// Получаем информацию о преподавателе по имени предмета

string subName = resultReport[i][0];

var sbj = AllHoursRepository.GetTeacherHourByName(teacherId, subName);

// Добавляем итоговые данные в отчёт

resultReport[i].Add($"Всего часов: {sbj.countHours}");

resultReport[i].Add($"Вычтено: {sum}");

resultReport[i].Add($"Осталось: {sbj.countHours - sum}");

// Обновляем данные в репозитории

AllHoursRepository.SubtractHours(sbj, sum);

}

// Определяем максимальное количество строк в таблице

int[] arr = new int[resultReport.Count];

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

{

arr[i] = resultReport[i].Count;

}

int max = arr.Max();

// Создаём объект Word

Microsoft.Office.Interop.Word.Application word = new Microsoft.Office.Interop.Word.Application();

Microsoft.Office.Interop.Word.Document doc = word.Documents.Add();

// Настраиваем параметры страницы (A3, альбомный режим)

doc.PageSetup.PaperSize = WdPaperSize.wdPaperA3;

doc.PageSetup.Orientation = WdOrientation.wdOrientLandscape;

// Добавляем заголовок отчёта

Microsoft.Office.Interop.Word.Paragraph paragraph = doc.Content.Paragraphs.Add();

paragraph.Range.Text = $"Отчёт по отработанным часам за {month} {year} года.";

paragraph.Range.Font.Size = 16;

paragraph.Range.Font.Bold = 1;

paragraph.Alignment = WdParagraphAlignment.wdAlignParagraphCenter;

paragraph.Range.InsertParagraphAfter();

// Создаём таблицу для отчёта

Microsoft.Office.Interop.Word.Range tableRange = doc.Content.Paragraphs.Add().Range;

Microsoft.Office.Interop.Word.Table table = doc.Tables.Add(tableRange, max, resultReport.Count);

// Настраиваем границы таблицы

table.Borders.Enable = 1;

table.Borders.OutsideLineStyle = WdLineStyle.wdLineStyleSingle;

table.Borders.InsideLineStyle = WdLineStyle.wdLineStyleSingle;

table.AutoFitBehavior(WdAutoFitBehavior.wdAutoFitContent);

table.Rows.Alignment = WdRowAlignment.wdAlignRowCenter;

// Заполняем таблицу данными из `resultReport`

for (int col = 0; col < resultReport.Count; col++)

{

for (int row = 0; row < resultReport[col].Count; row++)

{

table.Cell(row + 1, col + 1).Range.Text = resultReport[col][row];

table.Cell(row + 1, col + 1).Range.Font.Bold = 0;

}

}

// Добавляем дату создания отчёта

paragraph.Range.Text = $"Дата создания: {DateTime.Now.ToShortDateString()}";

paragraph.Range.Font.Size = 16;

paragraph.Range.Font.Bold = 1;

paragraph.Alignment = WdParagraphAlignment.wdAlignParagraphLeft;

paragraph.Range.InsertParagraphAfter();

// Сохраняем и закрываем документ

doc.SaveAs2(file);

doc.Close();

word.Quit();

}

}

Все остальные функции представлены в приложении А.

## Входные и выходные данные

В качестве входных данных приложение принимает введённое вручную или загруженное расписание занятий, а также период, в который эти занятия были проведены.

В качестве выходных данных выступает отчёт, который имеет следующую структуру:

* заголовок, в котором указан месяц и год, за который был составлен данный отчёт;
* таблица с данными об отработанных часах;
* дата создания отчёта.

Пример такого отчёта представлен в приложении Б на рисунке Б.1.

## Функциональное тестирование

Функциональное тестирование играет ключевую роль в обеспечении качества программного обеспечения, проверяя его соответствие заявленным требованиям. Оно проводится для выявления ошибок в работе системы, тестирования ее бизнес-логики, оценки удобства взаимодействия с пользователем и гарантии стабильности перед внедрением. В ходе функционального тестирования проверяются основные функции приложения, его корректность при обработке данных, взаимодействие модулей и реакция на некорректные входные данные.

Для проверки корректности работы приложения были разработаны тест-кейсы, которые представлены в таблице 3.6.

Таблица 3.6 – Тест-кейсы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Описание | Шаги | Ожидаемый результат | Фактический результат |
| Т1 | Проверка парсинга расписания по корректной ссылке | 1. Открыть приложение; 2. Ввести логин «TestUser»; 3. Ввести пароль «UserTest230225»; 4. Нажать на кнопку «Войти»; 5. Зайти на сайт колледжа; | Приложение должно загрузить недельное расписание и отобразить его в таблице | Соответствует ожидаемому, представлен в приложении В на рисунке В.1 |

Продолжение таблицы 3.6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Описание | Шаги | Ожидаемый результат | Фактический результат |
|  |  | 1. Открыть страницу с расписанием преподавателя; 2. Скопировать ссылку данной страницы; 3. Закрыть сайт; 4. Вернуться в приложение; 5. Перейти на страницу «Профиль»; 6. Установить скопированную ссылку в поле «Ссылка для парсинга расписания»; 7. Нажать на кнопку «Сохранить»; 8. Перейти на страницу «Расписание»; 9. Нажать на кнопку «Загрузить расписание». |  |  |
| Т2 | Проверка парсинга расписания по некорректной ссылке | 1. Открыть приложение; 2. Ввести логин «TestUser»; 3. Ввести пароль «UserTest230225»; 4. Нажать на кнопку «Войти»; 5. Перейти на страницу «Профиль»; 6. Установить ссылку «https://www.google.by/?hl=ru» в поле «Ссылка для парсинга расписания»; 7. Нажать на кнопку «Сохранить»; 8. Перейти на страницу «Расписание»; 9. Нажать на кнопку «Загрузить расписание». | Приложение должно оповестить об ошибке загрузки расписания | Соответствует ожидаемому, представлен в приложении В на рисунке В.2 |

Продолжение таблицы 3.6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Описание | Шаги | Ожидаемый результат | Фактический результат |
| Т3 | Проверка парсинга расписания с исправным интернет-соединением | 1. Подключиться к сети интернет; 2. Открыть приложение; 3. Ввести логин «TestUser»; 4. Ввести пароль «UserTest230225»; 5. Нажать на кнопку «Войти»; 6. Зайти на сайт колледжа: 7. Открыть страницу с расписанием преподавателя; 8. Скопировать ссылку данной страницы; 9. Закрыть сайт; 10. Вернуться в приложение; 11. Перейти на страницу «Профиль»; 12. Установить скопированную ссылку в поле «Ссылка для парсинга расписания»; 13. Нажать на кнопку «Сохранить»; 14. Перейти на страницу «Расписание»; 15. Нажать на кнопку «Загрузить расписание». | Приложение должно загрузить недельное расписание и отобразить его в таблице | Соответствует ожидаемому, представлен в приложении В на рисунке В.3 |
| Т4 | Проверка парсинга расписания с неисправным интернет-соединением | 1. Отключиться от сети интернет; 2. Открыть приложение; 3. Ввести логин «TestUser»; 4. Ввести пароль «UserTest230225»; 5. Нажать на кнопку «Войти»; 6. Перейти на страницу «Профиль»; | Приложение должно оповестить об ошибке загрузки расписания | Соответствует ожидаемому, представлен в приложении В на рисунке В.4 |

Продолжение таблицы 3.6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Описание | Шаги | Ожидаемый результат | Фактический результат |
|  |  | 1. Установить ссылку «https://www.google.by/?hl=ru» в поле «Ссылка для парсинга расписания»; 2. Нажать на кнопку «Сохранить»; 3. Перейти на страницу «Расписание»; 4. Нажать на кнопку «Загрузить расписание». |  |  |
| Т5 | Проверка составления отчёта с выбранными месяцем и годом | 1. Открыть приложение; 2. Ввести логин «TestUser»; 3. Ввести пароль «UserTest230225»; 4. Нажать на кнопку «Войти»; 5. Перейти на страницу «Отчёт»; 6. Выбрать месяц «Май» в поле «Месяц»; 7. Выбрать год «2025» в поле «Год»; 8. Нажать на кнопку «Составить отчёт»; 9. Указать путь для сохранения отчёта; 10. Нажать на кнопку «Сохранить». | Приложение должно оповестить об успешном составлении и сохранении отчёта | Соответствует ожидаемому, представлен в приложении В на рисунке В.5 |
| Т6 | Проверка составления отчёта с невыбранными месяцем и годом | 1. Открыть приложение; 2. Ввести логин «TestUser»; 3. Ввести пароль «UserTest230225»; 4. Нажать на кнопку «Войти»; 5. Перейти на страницу «Отчёт»; 6. Нажать на кнопку «Составить отчёт». | Приложение должно потребовать выбрать месяц и год | Соответствует ожидаемому, представлен в приложении В на рисунке В.6 |

Продолжение таблицы 3.6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Описание | Шаги | Ожидаемый результат | Фактический результат |
| Т7 | Проверка работы фильтров с некорректными параметрами | 1. Открыть приложение; 2. Ввести логин «TestUser»; 3. Ввести пароль «UserTest230225»; 4. Нажать на кнопку «Войти»; 5. Перейти на страницу «Часы»; 6. Указать в поле «С:» дату «16.05.2025»; 7. Указать в поле «До:» дату «14.05.2025»; 8. Нажать на кнопку «Применить фильтры». | Приложение должно оповестить об ошибке | Соответствует ожидаемому, представлен в приложении В на рисунке В.7 |
| Т8 | Проверка работы фильтров с корректными параметрами | 1. Открыть приложение; 2. Ввести логин «TestUser»; 3. Ввести пароль «UserTest230225»; 4. Нажать на кнопку «Войти»; 5. Перейти на страницу «Часы»; 6. Указать в поле «С:» дату «01.05.2025»; 7. Указать в поле «До:» дату «13.05.2025»; 8. Нажать на кнопку «Применить фильтры». | Приложение должно загрузить в таблицу результаты запроса | Соответствует ожидаемому, представлен в приложении В на рисунке В.8 |
| Т9 | Проверка добавления предмета с корректными данными | 1. Открыть приложение; 2. Ввести логин «TestUser»; 3. Ввести пароль «UserTest230225»; 4. Нажать на кнопку «Войти»; 5. Перейти на страницу «Часы»; | Приложение должно оповестить об успешном добавлении и обновить таблицу | Соответствует ожидаемому, представлен в приложении В на рисунке В.9 |

Продолжение таблицы 3.6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Описание | Шаги | Ожидаемый результат | Фактический результат |
|  |  | 1. Нажать на кнопку «Добавить час»; 2. Ввести в поле «Предмет:» значение «КПиЯП»; 3. Ввести в поле «Группа:» значение «Т-292»; 4. Выбрать дату «14.05.2025»; 5. В поле «Количество часов» указать значение «2»; 6. Нажать на кнопку «Добавить». |  |  |
| Т10 | Проверка добавления предмета с некорректными данными | 1. Открыть приложение; 2. Ввести логин «TestUser»; 3. Ввести пароль «UserTest230225»; 4. Нажать на кнопку «Войти»; 5. Перейти на страницу «Часы»; 6. Нажать на кнопку «Добавить час»; 7. Ничего не указывать в поле «Предмет:»; 8. Ввести в поле «Группа:» значение «Т-292»; 9. Не выбирать дату проведения предмета; 10. В поле «Количество часов» указать значение «2»; 11. Нажать на кнопку «Добавить». | Приложение должно оповестить об ошибке с указанием всех некорректно заполненных полей | Соответствует ожидаемому, представлен в приложении В на рисунке В.10 |

## Описание справочной системы

Справочная система для системы формирование отчётов по отработанным часам преподавателей была разработана с помощью Dr.Explain – программного обеспечения, предназначенного для создания документации, справочных систем и руководств. Оно позволяет разработчикам и техническим специалистам автоматически генерировать справку для программ, веб-сайтов и мобильных приложений.

Справочная система содержит следующие разделы:

* раздел «Окно аутентификации»;
* раздел «Главное окно».

Раздел «Окно аутентификации» содержит два подраздела:

* подраздел «Страница авторизации» – содержит подробное описание процесса авторизации;
* подраздел «Страница регистрации» – содержит подробное описание процесса регистрации.

Раздел «Главное окно» содержит пять подразделов:

* подраздел «Расписание» – содержит подробное описание функций, которые доступны на странице «Расписание»;
* подраздел «Часы» – содержит подробное описание функций, которые доступны на странице «Часы»;
* подраздел «Отчёт» – содержит подробное описание функций, которые доступны на странице «Отчёт»;
* подраздел «Профиль» – содержит подробное описание функций, которые доступны на странице «Профиль»;
* подраздел «Общее количество часов» – содержит подробное описание того, как работать с таблицей общих часов по предметам, которая находится на странице «Профиль».

Открыть справочную систему в приложении можно на любой странице, нажав на кнопку «Справка», которая расположена в правом верхнем углу приложения.

Содержание справочной системы представлено на рисунках 3.2 – 3.3.

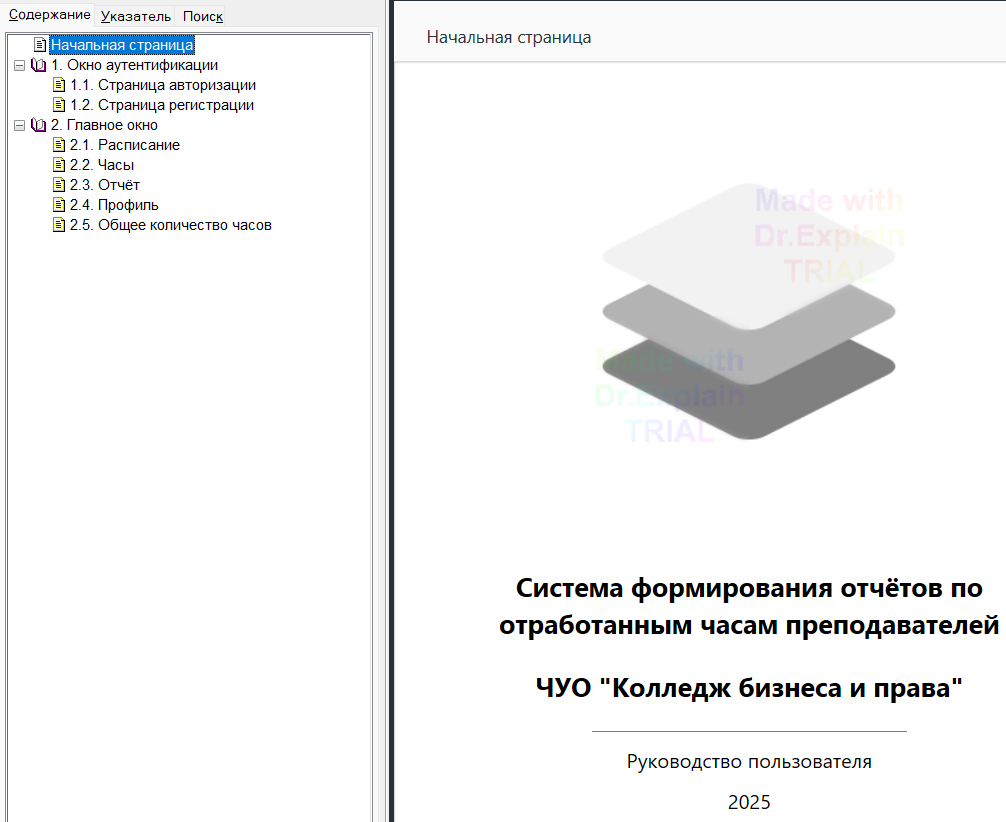


Рисунок 3.2 – Начальная страница справочной системы

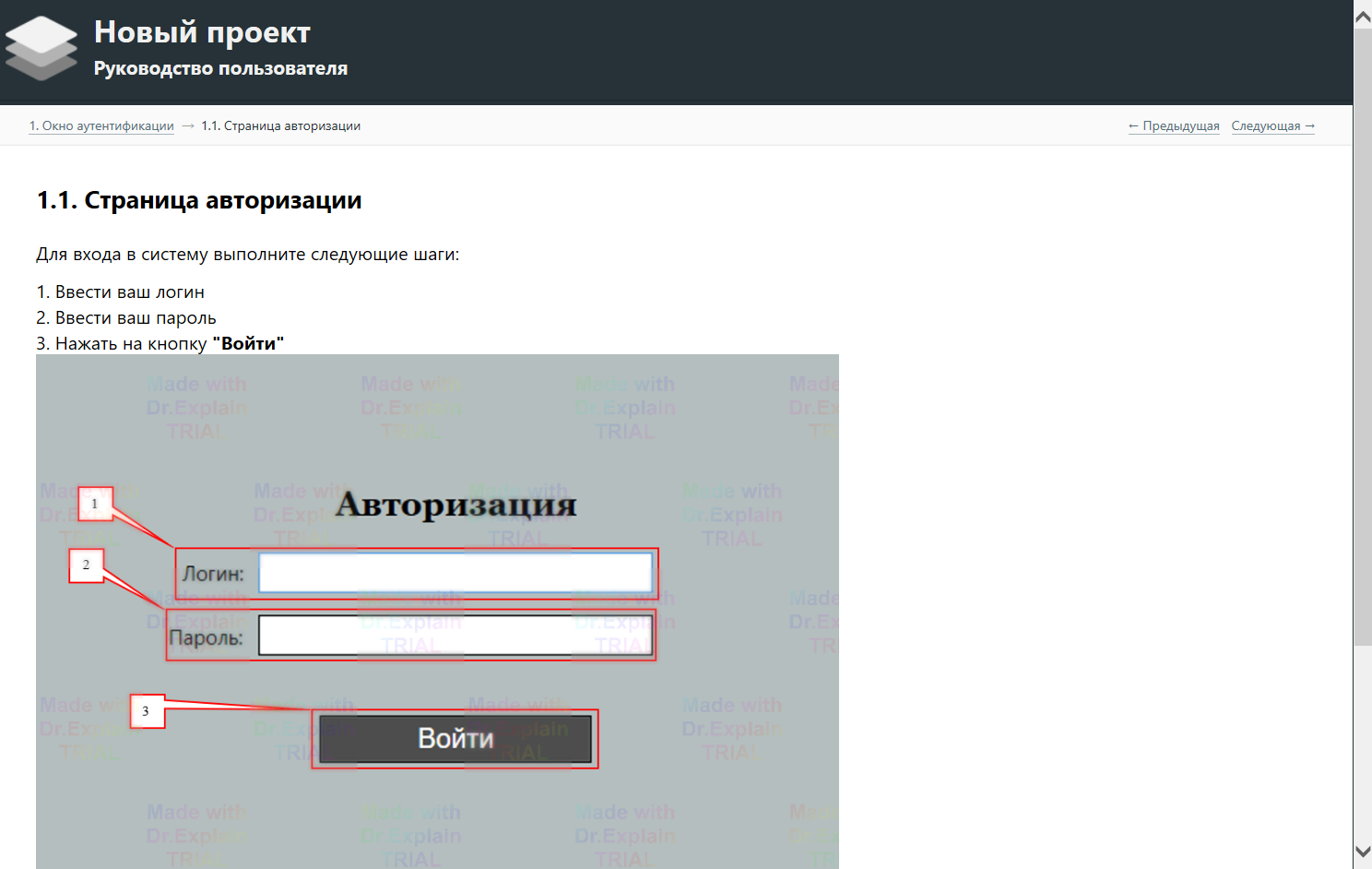


Рисунок 3.3 – Содержание справочной системы

# Применение

## Назначение программного средства

Разработанное программное средство (ПС) получило название «EduTreacker». Оно предназначено для облегчения процесса учёта преподавательской нагрузки, автоматического сбора данных и генерации отчётов. Оно позволяет преподавателям учебного заведения быстро анализировать проведённые занятия и формировать отчетность без необходимости ручного подсчёта.

Основные функции ПС:

* функция авторизации;
* функция регистрации;
* функция автоматического извлечения расписания с сайта колледжа;
* функция сохранения извлечённой информации в файл;
* функция сохранения извлечённой информации в файл в базу данных;
* функция просмотра расписания (посредством нажатия на кнопки «<» и «>»);
* функция изменения пароля;
* функция изменения ссылки для парсинга расписания;
* функция формирования месячного отчёта по отработанным часам, включая подсчёт отработанных часов за каждую дату;
* функция экспорта отчёта в Word;
* функция добавления информации об отработанных часах;
* функция удаления информации об отработанных часах;
* функция изменения информации об отработанных часах;
* функция фильтрации данных об отработанных часах по предмету, дате и группе;
* функция просмотра справки.

Разработанная система ориентирована только на преподавателей ЧУО «Колледж бизнеса и права» и способна загружать расписание преподавателей только с сайта данного учебного заведения.

## Условия применения

Для корректной работы программного средства на компьютере пользователя должно быть установлено следующее ПО:

* операционная система «Windows 7» и выше;
* программа «Microsoft Word» 2016 года выпуска и позже;
* система управления базами данных «Microsoft SQL Server» 2022 года;
* среда выполнения .NET Framework версии 4.7.2 и выше;
* средство установки MSI Installer.

Чтобы установить программу необходимо запустить установочный файл «Setup.msi». После его запуска откроется мастер установки и далее необходимо следовать его инструкциям.

При первом запуске программа автоматически развернёт базу данных на компьютере пользователя и после этого можно будет полноценно пользоваться установленным приложением.

# Заключение

Разработка системы автоматического формирования отчётов по отработанным часам преподавателей ЧУО «Колледж бизнеса и права» была направлена на оптимизацию процесса учёта, упрощение сбора данных о занятиях и создание удобной отчётности. В основе реализации лежали современные технологии, обеспечивающие стабильность и удобство работы.

Для построения системы использовалась среда разработки Microsoft Visual Studio, а программирование осуществлялось на языке C# с использованием технологии WPF для создания интерфейса. В качестве системы управления базами данных применялся Microsoft SQL Server, что позволило эффективно организовать хранение и обработку информации.

Функционал системы включает авторизацию и регистрацию пользователей, автоматическое извлечение расписания с сайта колледжа, сохранение данных как в файлы, так и в базу данных. Также предусмотрены возможности просмотра и редактирования информации об отработанных часах, изменения пароля и настройки параметров парсинга расписания. Дополнительно реализованы механизмы фильтрации данных, формирования месячных отчётов с подсчётом рабочих часов за каждую дату и экспорта отчётов в Word. Для удобства пользователей система включает справочную информацию, облегчающую работу с её функционалом.

Разработанное программное средство полностью соответствует поставленным требованиям, обеспечивая надёжность, стабильность и удобство эксплуатации. Все запланированные функции успешно реализованы, что гарантирует эффективное использование системы для автоматизации учёта преподавательской деятельности.

Преимущества данного ПС:

* автоматизация учёта преподавательской нагрузки, минимизация ручного ввода;
* удобный и интуитивно понятный интерфейс;
* надёжное хранение данных.

Недостатки данного ПС:

* ограниченная область применения (система разработана специально для ЧУО «Колледж бизнеса и права»);
* необходимость наличия Microsoft Word для работы с отчётами.

# Список использованных источников

1. Багласова, Т.Г. Методические указания по оформлению курсовых проектов, дипломных проектов и отчетов для учащихся специальности 2-40 01 01 «Программное обеспечение информационных технологий» // Т.Г. Багласова, К.О. Якимович и др. – Минск : КБП, 2022. – 41 с.
2. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
3. Объектно-ориентированный анализ и проектирование с примерами приложений / Г. Буч [и др.]. – 3-е изд. – М. : ООО «И.Д. Вильямс» , 2008 . – 720 с.
4. Программа и методика испытаний. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.301-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.
5. Руководство по C#. [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/. – Дата доступа: 12.04.2025.
6. Руководство по SQL Server. [Электронный ресурс]. – Режим доступа: https://metanit.com/sql/sqlserver/. – Дата доступа: 10.05.2025.
7. Руководство по SQL. [Электронный ресурс]. – Режим доступа: https://metanit.com/sql/. – Дата доступа: 01.05.2025.
8. Руководство по Visual Studio. [Электронный ресурс]. – Режим доступа: <https://visualstudio.microsoft.com/ru/>. – Дата доступа: 01.04.2025.
9. Руководство по WPF. [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/. – Дата доступа: 20.04.2025.
10. Текст программы. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**Текст программных модулей**

//Скрипт для разворачивания базы данных

USE [master]

GO

/\*\*\*\*\*\* Object: Database [TeachHours] Script Date: 16.05.2025 21:37:50 \*\*\*\*\*\*/

CREATE DATABASE [TeachHours]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'TeachHours', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\TeachHours.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

LOG ON

( NAME = N'TeachHours\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\TeachHours\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

WITH CATALOG\_COLLATION = DATABASE\_DEFAULT, LEDGER = OFF

GO

ALTER DATABASE [TeachHours] SET COMPATIBILITY\_LEVEL = 160

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [TeachHours].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [TeachHours] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [TeachHours] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [TeachHours] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [TeachHours] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [TeachHours] SET ARITHABORT OFF

GO

ALTER DATABASE [TeachHours] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [TeachHours] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [TeachHours] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [TeachHours] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [TeachHours] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [TeachHours] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [TeachHours] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [TeachHours] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [TeachHours] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [TeachHours] SET ENABLE\_BROKER

GO

ALTER DATABASE [TeachHours] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [TeachHours] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [TeachHours] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [TeachHours] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [TeachHours] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [TeachHours] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [TeachHours] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [TeachHours] SET RECOVERY FULL

GO

ALTER DATABASE [TeachHours] SET MULTI\_USER

GO

ALTER DATABASE [TeachHours] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [TeachHours] SET DB\_CHAINING OFF

GO

ALTER DATABASE [TeachHours] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [TeachHours] SET TARGET\_RECOVERY\_TIME = 60 SECONDS

GO

ALTER DATABASE [TeachHours] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [TeachHours] SET ACCELERATED\_DATABASE\_RECOVERY = OFF

GO

EXEC sys.sp\_db\_vardecimal\_storage\_format N'TeachHours', N'ON'

GO

ALTER DATABASE [TeachHours] SET QUERY\_STORE = ON

GO

ALTER DATABASE [TeachHours] SET QUERY\_STORE (OPERATION\_MODE = READ\_WRITE, CLEANUP\_POLICY = (STALE\_QUERY\_THRESHOLD\_DAYS = 30), DATA\_FLUSH\_INTERVAL\_SECONDS = 900, INTERVAL\_LENGTH\_MINUTES = 60, MAX\_STORAGE\_SIZE\_MB = 1000, QUERY\_CAPTURE\_MODE = AUTO, SIZE\_BASED\_CLEANUP\_MODE = AUTO, MAX\_PLANS\_PER\_QUERY = 200, WAIT\_STATS\_CAPTURE\_MODE = ON)

GO

USE [TeachHours]

GO

/\*\*\*\*\*\* Object: Table [dbo].[AllHours] Script Date: 16.05.2025 21:37:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AllHours](

[id] [int] IDENTITY(1,1) NOT NULL,

[subjectName] [nvarchar](50) NOT NULL,

[teacherId] [int] NOT NULL,

[countHours] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Dates] Script Date: 16.05.2025 21:37:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Dates](

[id] [int] IDENTITY(1,1) NOT NULL,

[date] [date] NOT NULL,

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Hours] Script Date: 16.05.2025 21:37:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Hours](

[id] [int] IDENTITY(1,1) NOT NULL,

[subjectId] [int] NOT NULL,

[dateId] [int] NOT NULL,

[teacherId] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Subjects] Script Date: 16.05.2025 21:37:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Subjects](

[id] [int] IDENTITY(1,1) NOT NULL,

[name] [nvarchar](50) NOT NULL,

[teacherId] [int] NOT NULL,

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Teachers] Script Date: 16.05.2025 21:37:50 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Teachers](

[id] [int] IDENTITY(1,1) NOT NULL,

[login] [nvarchar](50) NOT NULL,

[password] [nvarchar](255) NOT NULL,

[url] [nvarchar](max) NULL,

PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[AllHours] WITH CHECK ADD CONSTRAINT [FK\_AllHours\_Teachers] FOREIGN KEY([teacherId])

REFERENCES [dbo].[Teachers] ([id])

GO

ALTER TABLE [dbo].[AllHours] CHECK CONSTRAINT [FK\_AllHours\_Teachers]

GO

ALTER TABLE [dbo].[Hours] WITH CHECK ADD CONSTRAINT [FK\_Hours\_Dates] FOREIGN KEY([dateId])

REFERENCES [dbo].[Dates] ([id])

GO

ALTER TABLE [dbo].[Hours] CHECK CONSTRAINT [FK\_Hours\_Dates]

GO

ALTER TABLE [dbo].[Hours] WITH CHECK ADD CONSTRAINT [FK\_Hours\_Subjects] FOREIGN KEY([subjectId])

REFERENCES [dbo].[Subjects] ([id])

GO

ALTER TABLE [dbo].[Hours] CHECK CONSTRAINT [FK\_Hours\_Subjects]

GO

ALTER TABLE [dbo].[Hours] WITH CHECK ADD CONSTRAINT [FK\_Hours\_Teachers] FOREIGN KEY([teacherId])

REFERENCES [dbo].[Teachers] ([id])

GO

ALTER TABLE [dbo].[Hours] CHECK CONSTRAINT [FK\_Hours\_Teachers]

GO

ALTER TABLE [dbo].[Subjects] WITH CHECK ADD CONSTRAINT [FK\_Subjects\_Teachers] FOREIGN KEY([teacherId])

REFERENCES [dbo].[Teachers] ([id])

GO

ALTER TABLE [dbo].[Subjects] CHECK CONSTRAINT [FK\_Subjects\_Teachers]

GO

USE [master]

GO

ALTER DATABASE [TeachHours] SET READ\_WRITE

GO

//Код файла AllHoursRepository.cs

// Репозиторий для работы с таблицей AllHours в базе данных

internal static class AllHoursRepository

{

// Получает список всех записей о часах для текущего преподавателя

public static List<AllHours> GetAllHoursOfCurrentTeacher(int currentTeacherId) =>

TeachHoursEntities2.GetContext().AllHours.Where(t => t.teacherId == currentTeacherId).ToList();

// Получает запись о часах преподавателя по названию предмета

public static AllHours GetTeacherHourByName(int teachId, string sbjName) =>

TeachHoursEntities2.GetContext().AllHours.FirstOrDefault(s => s.subjectName == sbjName && s.teacherId == teachId);

// Получает запись о часах преподавателя по уникальному идентификатору

public static AllHours GetHourById(int id) =>

TeachHoursEntities2.GetContext().AllHours.FirstOrDefault(h => h.id == id);

// Уменьшает количество часов для конкретного предмета преподавателя

public static void SubtractHours(AllHours hour, int hours)

{

hour.countHours -= hours;

// Если количество часов становится отрицательным, устанавливаем его в 0

if (hour.countHours < 0)

{

hour.countHours = 0;

}

// Сохраняем изменения в базе данных

TeachHoursEntities2.GetContext().SaveChanges();

}

// Добавляет новую запись о часах преподавателя в базу данных

public static void Add(AllHours hour)

{

TeachHoursEntities2.GetContext().AllHours.Add(hour);

TeachHoursEntities2.GetContext().SaveChanges();

}

// Удаляет запись о часах преподавателя из базы данных

public static void Delete(AllHours hour)

{

TeachHoursEntities2.GetContext().AllHours.Remove(hour);

TeachHoursEntities2.GetContext().SaveChanges();

}

// Обновляет количество часов для конкретной записи

public static void ChangeCountOfHours(AllHours hour, int countOfHours)

{

hour.countHours = countOfHours;

// Сохраняем изменения в базе данных

TeachHoursEntities2.GetContext().SaveChanges();

}

}

//Код файла TeachersRepository.cs

// Репозиторий для работы с таблицей Teachers в базе данных

internal static class TeachersRepository

{

// Получает словарь, где логины преподавателей являются ключами, а пароли — значениями

public static Dictionary<string, string> GetTeachersDictionary() =>

TeachHoursEntities2.GetContext().Teachers.ToDictionary(teach => teach.login, teach => teach.password);

// Добавляет нового преподавателя в базу данных

public static bool Save(string login, string hashPassword)

{

Teachers teacher = new Teachers

{

login = login, // Устанавливаем логин преподавателя

password = hashPassword // Сохраняем хэшированный пароль

};

try

{

// Добавляем преподавателя в контекст базы данных и сохраняем изменения

TeachHoursEntities2.GetContext().Teachers.Add(teacher);

TeachHoursEntities2.GetContext().SaveChanges();

return true; // Возвращаем успех при успешном сохранении

}

catch (Exception)

{

return false; // В случае ошибки возвращаем false

}

}

// Получает преподавателя по его уникальному идентификатору

public static Teachers GetTeacherById(int id) =>

TeachHoursEntities2.GetContext().Teachers.FirstOrDefault(t => t.id == id);

// Получает список всех преподавателей из базы данных

public static List<Teachers> GetTeachersList() =>

TeachHoursEntities2.GetContext().Teachers.ToList();

// Изменяет пароль преподавателя

public static bool ChangePassword(Teachers teacher, string password)

{

try

{

teacher.password = password; // Обновляем пароль

TeachHoursEntities2.GetContext().SaveChanges(); // Сохраняем изменения

return true; // Возвращаем успех

}

catch (Exception)

{

return false; // В случае ошибки возвращаем false

}

}

// Изменяет URL-адрес (например, ссылку на профиль) преподавателя

public static bool ChangeUrl(Teachers teacher, string url)

{

try

{

teacher.url = url; // Обновляем URL-адрес

TeachHoursEntities2.GetContext().SaveChanges(); // Сохраняем изменения

return true; // Возвращаем успех

}

catch (Exception)

{

return false; // В случае ошибки возвращаем false

}

}

}

//Код файла TeachAuth.cs

// Класс TeachAuth отвечает за аутентификацию преподавателей

internal class TeachAuth

{

// Поля для хранения логина и хэшированного пароля преподавателя

string login;

string hashPassword;

// Пустой конструктор (используется при необходимости создания объекта без параметров)

public TeachAuth() { }

// Конструктор с параметрами для инициализации логина и пароля

public TeachAuth(string login, string hashPas)

{

this.login = login;

this.hashPassword = hashPas;

}

// Проверяет, существует ли преподаватель с данным логином в переданном словаре

public bool ValidateLogin(Dictionary<string, string> teachers) => teachers.Keys.Contains(login);

// Проверяет, соответствует ли логин и пароль преподавателя сохранённым данным

// Возвращает идентификатор преподавателя, если найден, иначе -1

public int ValidatePassword(List<Teachers> teachers)

{

// Ищем преподавателя с совпадающим логином и паролем

int? teacherId = teachers.FirstOrDefault(teach => teach.login == login && teach.password == hashPassword)?.id;

// Если преподаватель не найден, возвращаем -1

if (teacherId == null)

{

return -1;

}

// Возвращаем идентификатор преподавателя

return teacherId.Value;

}

}

//Код файла TeachReg.cs

// Класс TeachReg предназначен для регистрации преподавателей

internal class TeachReg

{

// Поля для хранения логина и пароля

string login;

string password;

string repeatPassword;

// Конструктор с параметрами для инициализации объекта

public TeachReg(string login, string password, string repeatPassword)

{

this.login = login; // Сохраняем логин пользователя

this.password = password; // Основной пароль

this.repeatPassword = repeatPassword; // Повторный ввод пароля

}

// Пустой конструктор для возможного создания объекта без параметров

public TeachReg() { }

// Проверяет, совпадают ли введённые пароли

public bool ValidatePasswords() => password == repeatPassword ? true : false;

// Проверяет, соответствует ли длина пароля минимальному требованию (не менее 6 символов)

public bool ValidatePasswordLength() => password.Length >= 6 ? true : false;

// Проверяет, не существует ли уже логин в списке зарегистрированных преподавателей

public bool ValidateLogin(Dictionary<string, string> teachers) => !teachers.Keys.Contains(login);

}

//Код файла AllSubjectHours.cs

// Класс AllSubjectHours предназначен для хранения информации о часах занятий по предметам

internal class AllSubjectHours

{

// Уникальный идентификатор записи

public int Id { get; set; }

// Дата проведения занятия в строковом формате

public string Date { get; set; }

// Название предмета

public string Subject { get; set; }

// Конструктор с параметрами для инициализации объекта

public AllSubjectHours(int id, DateTime date, string subject)

{

this.Id = id; // Устанавливаем ID записи

this.Date = date.ToShortDateString(); // Конвертируем дату в строковый формат (например, "18.05.2025")

this.Subject = subject; // Сохраняем название предмета

}

}

//Код файла WeekSubjects.cs

// Класс WeekSubjects используется для хранения списка предметов по дням недели

public class WeekSubjects

{

// Свойства, содержащие названия предметов для каждого дня недели

public string Monday { get; set; }

public string Tuesday { get; set; }

public string Wednesday { get; set; }

public string Thursday { get; set; }

public string Friday { get; set; }

public string Saturday { get; set; }

// Конструктор класса, принимающий названия предметов для каждого дня

public WeekSubjects(string monday, string tuesday, string wednesday, string thursday, string friday, string saturday)

{

this.Monday = monday; // Устанавливаем предмет для понедельника

this.Tuesday = tuesday; // Устанавливаем предмет для вторника

this.Wednesday = wednesday;// Устанавливаем предмет для среды

this.Thursday = thursday; // Устанавливаем предмет для четверга

this.Friday = friday; // Устанавливаем предмет для пятницы

this.Saturday = saturday; // Устанавливаем предмет для субботы

}

}

//Код файла WordReport.cs

// Класс WordReport реализует интерфейс IReporter и отвечает за генерацию отчетов в формате Word

internal class WordReport : IReporter

{

// Метод для создания отчета по отработанным часам преподавателя

public void GenerateReport(int teacherId, List<List<string>> resultReport, string month, string year)

{

// Определяем директорию, в которой будут сохраняться отчеты

string projectDir = AppDomain.CurrentDomain.BaseDirectory;

string dir = System.IO.Path.Combine(projectDir, "Отчёты");

// Окно выбора пути сохранения файла

var save = new SaveFileDialog

{

Title = "Сохранить файл как",

Filter = "Документы Word (\*.docx)|\*.docx",

FileName = $"Отчёт.docx",

InitialDirectory = dir

};

// Имя файла по умолчанию

string file = $"Отчёт.docx";

// Если пользователь выбрал путь сохранения

if (save.ShowDialog() == true)

{

file = save.FileName;

// Проходим по списку отчетных данных и обрабатываем количество отработанных часов

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

{

int sum = 0;

// Обрабатываем строки с данными о количестве часов

for (int j = 1; j < resultReport[i].Count; j++)

{

string[] temp = resultReport[i][j].Split('-');

// Пытаемся извлечь число из строки

if (int.TryParse(temp[1].Trim(), out int num))

{

sum += num;

}

}

// Получаем данные о преподавателе и обновляем отчет

string subName = resultReport[i][0];

var sbj = AllHoursRepository.GetTeacherHourByName(teacherId, subName);

resultReport[i].Add($"Всего часов: {sbj.countHours}");

resultReport[i].Add($"Вычтено: {sum}");

resultReport[i].Add($"Осталось: {sbj.countHours - sum}");

// Корректируем количество оставшихся часов в базе данных

AllHoursRepository.SubtractHours(sbj, sum);

}

// Определяем максимальное количество строк в отчете

int[] arr = new int[resultReport.Count];

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

{

arr[i] = resultReport[i].Count;

}

int max = arr.Max();

// Создаем документ Word и настраиваем его параметры

Microsoft.Office.Interop.Word.Application word = new Microsoft.Office.Interop.Word.Application();

Microsoft.Office.Interop.Word.Document doc = word.Documents.Add();

doc.PageSetup.PaperSize = WdPaperSize.wdPaperA3;

doc.PageSetup.Orientation = WdOrientation.wdOrientLandscape;

// Добавляем заголовок отчета

Microsoft.Office.Interop.Word.Paragraph paragraph = doc.Content.Paragraphs.Add();

paragraph.Range.Text = $"Отчёт по отработанным часам за {month} {year} года.";

paragraph.Range.Font.Size = 16;

paragraph.Range.Font.Bold = 1;

paragraph.Alignment = WdParagraphAlignment.wdAlignParagraphCenter;

paragraph.Range.InsertParagraphAfter();

// Создаем таблицу для отображения данных отчета

Microsoft.Office.Interop.Word.Range tableRange = doc.Content.Paragraphs.Add().Range;

Microsoft.Office.Interop.Word.Table table = doc.Tables.Add(tableRange, max, resultReport.Count);

table.Borders.Enable = 1;

table.Borders.OutsideLineStyle = WdLineStyle.wdLineStyleSingle;

table.Borders.InsideLineStyle = WdLineStyle.wdLineStyleSingle;

table.AutoFitBehavior(WdAutoFitBehavior.wdAutoFitContent);

table.Rows.Alignment = WdRowAlignment.wdAlignRowCenter;

// Заполняем таблицу данными

for (int col = 0; col < resultReport.Count; col++)

{

for (int row = 0; row < resultReport[col].Count; row++)

{

table.Cell(row + 1, col + 1).Range.Text = resultReport[col][row];

table.Cell(row + 1, col + 1).Range.Font.Bold = 0;

}

}

// Добавляем дату создания отчета

paragraph.Range.Text = $"Дата создания: {DateTime.Now.ToShortDateString()}";

paragraph.Range.Font.Size = 16;

paragraph.Range.Font.Bold = 1;

paragraph.Alignment = WdParagraphAlignment.wdAlignParagraphLeft;

paragraph.Range.InsertParagraphAfter();

// Сохраняем и закрываем документ

doc.SaveAs2(file);

doc.Close();

word.Quit();

}

}

}

//Код файла HtmlParser.cs

// Статический класс HtmlParser предназначен для парсинга веб-страниц и получения данных о расписании

internal static class HtmlParser

{

// Асинхронный метод для получения информации о занятиях по дням недели с указанного URL

public async static Task<List<string>> GetDayOfWeek(string url, string subjectsTag, string groupsTag)

{

// Создаём конфигурацию с загрузчиком страниц по умолчанию

var config = Configuration.Default.WithDefaultLoader();

// Создаём контекст для обработки HTML-документа

var context = BrowsingContext.New(config);

// Асинхронно загружаем HTML-документ по указанному URL

var doc = await context.OpenAsync(url);

// Проверка, успешно ли загружен документ

if (doc != null) // Можно также проверять статус ответа, если необходимо

{

// Создаём список для хранения данных о занятиях

var day = new List<string>();

// Получаем элементы с информацией о предметах

var links1 = doc.QuerySelectorAll(subjectsTag);

// Получаем элементы с информацией о группах

var links2 = doc.QuerySelectorAll(groupsTag);

// Формируем список предметов и групп для каждого дня

for (int j = 0; j < links1.Length; j++)

{

// Добавляем название группы и предмета в список

day.Add($"{links2[j].TextContent} {links1[j].TextContent}");

}

// Возвращаем список занятий

return day;

}

else

{

// Если документ не был загружен, возвращаем пустой список вместо null

return new List<string>();

}

}

}

//Код файла PasswordHasher.cs

// Статический класс для хеширования паролей

internal static class PasswordHasher

{

// Метод для вычисления хеш-кода строки (используется простой алгоритм хеширования)

public static ulong HashFunc(string str)

{

int key = 11; // Ключ для расчета хеша (основание степени)

int count = 9; // Количество разрядов, используемых для ограничения размера хеша

ulong size = (ulong)Math.Pow(10, count); // Ограничение максимального размера хеш-кода

ulong hash\_code, t\_hash = 0;

// Проход по каждому символу строки

for (int i = 0; i < str.Length; i++)

{

// Вычисление хеш-значения на основе символа и позиции в строке

t\_hash += (ulong)Math.Pow(key, i) \* (ulong)str[i];

// Ограничение размера хеша, чтобы избежать переполнения

t\_hash %= size;

}

// Финальное ограничение хеш-кода

hash\_code = t\_hash % size;

return hash\_code;

}

}

//Код файла IReporter.cs

// Интерфейс IReporter предназначен для реализации механизмов генерации отчётов

internal interface IReporter

{

// Метод GenerateReport должен быть реализован в классах, которые поддерживают создание отчётов

void GenerateReport(int teacherId, List<List<string>> resultReport, string month, string year);

}

//Код файла Hours.cs

// Класс Hours представляет собой сущность базы данных, содержащую информацию о проведённых занятиях

public partial class Hours

{

// Уникальный идентификатор записи

public int id { get; set; }

// Идентификатор предмета (связан с таблицей Subjects)

public int subjectId { get; set; }

// Идентификатор даты занятия (связан с таблицей Dates)

public int dateId { get; set; }

// Идентификатор преподавателя (связан с таблицей Teachers)

public int teacherId { get; set; }

// Навигационные свойства для связи с другими таблицами

public virtual Dates Dates { get; set; } // Связь с таблицей дат занятий

public virtual Subjects Subjects { get; set; } // Связь с таблицей предметов

public virtual Teachers Teachers { get; set; } // Связь с таблицей преподавателей

}

//Код файла Teachers.cs

// Класс Teachers представляет сущность преподавателя в базе данных

public partial class Teachers

{

// Отключает предупреждение о вызове виртуальных методов в конструкторе

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

// Конструктор класса, инициализирующий коллекции для навигационных свойств

public Teachers()

{

this.AllHours = new HashSet<AllHours>(); // Инициализация списка часов работы преподавателя

this.Hours = new HashSet<Hours>(); // Инициализация списка записей о занятиях

this.Subjects = new HashSet<Subjects>(); // Инициализация списка предметов преподавателя

}

// Уникальный идентификатор преподавателя

public int id { get; set; }

// Логин преподавателя для аутентификации

public string login { get; set; }

// Пароль преподавателя (лучше хранить в хэшированном виде!)

public string password { get; set; }

// URL-адрес (может использоваться для профиля или связанной информации)

public string url { get; set; }

// Отключает предупреждение о изменяемых коллекциях

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

// Связь один-ко-многим: преподаватель имеет множество записей о часах работы

public virtual ICollection<AllHours> AllHours { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

// Связь один-ко-многим: преподаватель ведёт занятия (Hours)

public virtual ICollection<Hours> Hours { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

// Связь один-ко-многим: преподаватель может вести несколько предметов

public virtual ICollection<Subjects> Subjects { get; set; }

}

//Код файла Subjects.cs

// Класс Subjects представляет собой сущность базы данных для хранения информации о предметах

public partial class Subjects

{

// Отключает предупреждение о вызове виртуальных методов в конструкторе

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

// Конструктор класса, инициализирующий коллекцию связей с записями о занятиях

public Subjects()

{

this.Hours = new HashSet<Hours>(); // Инициализация коллекции занятий по предмету

}

// Уникальный идентификатор предмета

public int id { get; set; }

// Название предмета

public string name { get; set; }

// Идентификатор преподавателя, ведущего этот предмет

public int teacherId { get; set; }

// Отключает предупреждение о модифицируемых коллекциях

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

// Связь один-ко-многим: предмет может иметь множество записей о проведённых занятиях

public virtual ICollection<Hours> Hours { get; set; }

// Связь один-ко-одному: предмет привязан к конкретному преподавателю

public virtual Teachers Teachers { get; set; }

}

//Код файла AllHours.cs

// Класс AllHours представляет собой сущность базы данных, содержащую информацию о преподавательских часах

public partial class AllHours

{

// Уникальный идентификатор записи

public int id { get; set; }

// Название предмета, к которому относится запись

public string subjectName { get; set; }

// Идентификатор преподавателя, связанного с этой записью

public int teacherId { get; set; }

// Количество отработанных часов по данному предмету

public int countHours { get; set; }

// Навигационное свойство для связи с преподавателем

public virtual Teachers Teachers { get; set; }

}

//Код файла Dates.cs

// Класс Dates представляет сущность базы данных для хранения информации о датах занятий

public partial class Dates

{

// Отключает предупреждение о вызове виртуальных методов в конструкторе

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

// Конструктор класса, инициализирующий коллекцию связанных занятий

public Dates()

{

this.Hours = new HashSet<Hours>(); // Инициализация коллекции занятий, связанных с датой

}

// Уникальный идентификатор даты

public int id { get; set; }

// Значение даты (использует System.DateTime)

public System.DateTime date { get; set; }

// Отключает предупреждение о модифицируемых коллекциях

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

// Связь один-ко-многим: одна дата может содержать несколько занятий

public virtual ICollection<Hours> Hours { get; set; }

}

//Код файла AddHourWindow.xaml.cs

public partial class AddHourWindow : Window

{

public AddHourWindow()

{

InitializeComponent();

}

public AddHourWindow(int teachId)

{

InitializeComponent();

this.id = teachId;

List<Hours> hours = HoursRepository.GetTeacherHours(id);

foreach (var h in hours)

{

int id = h.id;

string subject = SubjectsRepository.GetSubjectById(h.subjectId).name;

DateTime date = DatesRepository.GetDateById(h.dateId).date;

allSubjectHours.Add(new AllSubjectHours(id, date, subject));

}

List<string> uniqGroups = new List<string>();

List<string> uniqSubjects = new List<string>();

foreach (var s in allSubjectHours)

{

var temp = s.Subject.Split(' ');

if (!uniqGroups.Contains(temp[0]))

{

uniqGroups.Add(temp[0]);

}

if (!uniqSubjects.Contains(temp[1]))

{

uniqSubjects.Add(temp[1]);

}

}

NameBox.ItemsSource = uniqSubjects;

GroupBox.ItemsSource = uniqGroups;

}

int id;

BindingList<AllSubjectHours> allSubjectHours = new BindingList<AllSubjectHours>();

private void Button\_Click(object sender, RoutedEventArgs e)

{

try

{

List<string> errors = new List<string>();

if (!Regex.IsMatch(GroupBox.Text, @"[А-Я]{1}-[0-9]{3}"))

{

errors.Add("Некорретное название группы.\nНазвание должно соответствовать шаблону \"А-000\"");

}

if(NameBox.Text.Length == 0 || NameBox.Text.Trim().ToLower().Contains("Урок снят".Trim().ToLower()))

{

errors.Add("Некорретное название предмета.\nНазвание предмета не может быть пустым.\nПредмет не может называться \"Урок снят\"");

}

if(datePicker.SelectedDate == null)

{

errors.Add("Некорректная дата.");

}

if(!(int.TryParse(DateBox.Text, out int count) == true && count >= 1 && count <= 16))

{

errors.Add("Некорректное количество часов. Часы должны находиться в диапазоне от 1 до 16.");

}

if(errors.Count > 0)

{

StringBuilder sb = new StringBuilder("");

foreach (var t in errors)

{

sb.AppendLine(t);

}

MessageBox.Show(sb.ToString(), "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

DateTime.TryParse(datePicker.SelectedDate.Value.ToString(), out DateTime date);

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

{

Hours hour = new Hours();

Subjects subjects = new Subjects();

subjects.name = $"{GroupBox.Text.Trim()} {NameBox.Text.Trim()}";

subjects.teacherId = this.id;

SubjectsRepository.Add(subjects);

Dates dates = new Dates();

dates.date = date;

DatesRepository.Add(dates);

hour.subjectId = subjects.id;

hour.teacherId = this.id;

hour.dateId = dates.id;

HoursRepository.Add(hour);

}

MessageBox.Show("Час успешно добавлен.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

this.DialogResult = true;

this.Close();

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void Date\_Click(object sender, MouseButtonEventArgs e)

{

DateBox.Text = "";

}

}

//Код файла App.xaml.cs

public partial class App : Application

{

protected override void OnStartup(StartupEventArgs e)

{

base.OnStartup(e);

try

{

string connectString = @"Data Source=localhost;Initial Catalog=master;Integrated Security=True";

if (!DatabaseExists("TeachHours", connectString))

{

string scriptPath = "script.sql"; // путь к вашему SQL-скрипту

string script = File.ReadAllText(scriptPath);

string[] commands = Regex.Split(script, @"^\s\*GO\s\*$", RegexOptions.Multiline | RegexOptions.IgnoreCase);

using (SqlConnection connection = new SqlConnection(connectString))

{

connection.Open();

foreach (string command in commands)

{

if (!string.IsNullOrWhiteSpace(command))

{

using (SqlCommand sqlCommand = new SqlCommand(command, connection))

{

sqlCommand.ExecuteNonQuery();

}

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при развёртывании базы данных. Вероятнее всего у вас не установлен Sql Server. Пожалуйста, устанвоите его и повторите попытку.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

//MessageBox.Show(ex.Message);

Shutdown();

}

}

bool DatabaseExists(string databaseName, string connectionStringMaster)

{

using (SqlConnection connection = new SqlConnection(connectionStringMaster))

{

connection.Open();

string query = $"SELECT database\_id FROM sys.databases WHERE name = @name";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@name", databaseName);

object result = command.ExecuteScalar();

return (result != null);

}

}

}

}

//Код файла BaseWindow.xaml.cs

public partial class BaseWindow : System.Windows.Window

{

int id;

DateTime reportDate = new DateTime(1920, 12, 31);

string fileName = "";

Teachers teacher;

BindingList<WeekSubjects> weekSubjects;

BindingList<AllSubjectHours> allSubjectHours;

BindingList<AllSubjectHours> filterSubjectHours = new BindingList<AllSubjectHours>();

public delegate void GetAllHourForSubject(int t);

int allHoursOfSubject = -1;

List<string> filesInDirectory;

bool sourceSave = true;

int indexOfFile = -1;

DateTime week = DateTime.Now;

public BaseWindow()

{

InitializeComponent();

}

public BaseWindow(int teacherId)

{

InitializeComponent();

id = teacherId;

teacher = TeachersRepository.GetTeacherById(id);

CreateTeachFolder(id);

}

private void SavePassButton\_Click(object sender, RoutedEventArgs e)

{

if(MessageBox.Show("Вы точно хотите изменить свой пароль?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Warning) == MessageBoxResult.Yes)

{

TeachReg validator = new TeachReg(teacher.login, this.NewPassBox.Text, this.NewPassBox.Text);

if (validator.ValidatePasswordLength())

{

string newPass = PasswordHasher.HashFunc(this.NewPassBox.Text).ToString();

if (TeachersRepository.ChangePassword(teacher, newPass))

{

MessageBox.Show("Новый пароль сохранён.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show("Во время сохранения пароля возникла ошибка. Повторите попытку.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Длина пароля должна составлять минимум 6 символов.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

if(MessageBox.Show("Вы действительно хотите сохранить иземенения?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Warning) == MessageBoxResult.Yes)

{

var button = sender as Button;

var hour = button?.DataContext as AllSubjectHours;

var selectedHour = HoursRepository.GetHourById(hour.Id);

try

{

Dates date = DatesRepository.GetDateById(selectedHour.dateId);

Subjects subject = SubjectsRepository.GetSubjectById(selectedHour.subjectId);

if (DateTime.TryParse(hour.Date, out DateTime dt) && hour.Subject.Length > 0)

{

if(DatesRepository.ChangeDate(date.id, dt) && SubjectsRepository.ChangeNameSubject(subject.id, hour.Subject))

{

MessageBox.Show("Данные успешно изменены.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

CreateAllHoursTable();

}

else

{

MessageBox.Show("При сохранении изменений произошла ошибка.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Данные введены неверно.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

UpdateHoursGrid();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void Base\_Load(object sender, RoutedEventArgs e)

{

this.LoginBox.Text = teacher.login;

this.UrlBox.Text = teacher.url;

List<string> years = new List<string>();

for(int i = DateTime.Now.Year; i >= DateTime.Now.Year-5; i--)

{

years.Add(i.ToString());

}

YearBox.ItemsSource = years;

List<string>months = new List<string>(new string[] {"Январь", "Февраль" , "Март" , "Апрель" , "Май" , "Июнь", "Июль", "Август", "Сентябрь", "Ноябрь", "Декабрь" });

MonthBox.ItemsSource = months;

List<Hours> hours = HoursRepository.GetTeacherHours(teacher.id);//TeachHoursEntities2.GetContext().Hours.Where(h => h.teacherId == teacher.id).ToList();

allSubjectHours = new BindingList<AllSubjectHours>();

foreach(var h in hours)

{

int id = h.id;

string subject = SubjectsRepository.GetSubjectById(h.subjectId).name;

DateTime date = DatesRepository.GetDateById(h.dateId).date;

allSubjectHours.Add(new AllSubjectHours(id, date, subject));

}

CreateAllHoursTable();

this.HoursGrid.ItemsSource = allSubjectHours;

while(week.DayOfWeek != DayOfWeek.Monday)

{

week = week.AddDays(-1);

}

DateTime week1 = week;

DateTime date1 = DateTime.Now;

DateTime date2 = DateTime.Now;

for(int i = 1; i < 7; i++, week1 = week1.AddDays(1))

{

if (i == 1)

{

var column = new DataGridTextColumn

{

Header = $"Понедельник\n{week1.ToShortDateString()}",

Binding = new Binding("Monday")

};

this.SourceGrid.Columns.Add(column);

date1 = week1;

}

if (i == 2)

{

var column = new DataGridTextColumn

{

Header = $"Вторник\n{week1.ToShortDateString()}",

Binding = new Binding("Tuesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 3)

{

var column = new DataGridTextColumn

{

Header = $"Среда\n{week1.ToShortDateString()}",

Binding = new Binding("Wednesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 4)

{

var column = new DataGridTextColumn

{

Header = $"Четверг\n{week1.ToShortDateString()}",

Binding = new Binding("Thursday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 5)

{

var column = new DataGridTextColumn

{

Header = $"Пятница\n{week1.ToShortDateString()}",

Binding = new Binding("Friday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 6)

{

var column = new DataGridTextColumn

{

Header = $"Суббота\n{week1.ToShortDateString()}",

Binding = new Binding("Saturday")

};

this.SourceGrid.Columns.Add(column);

date2 = week1;

}

}

var teachLogin = teacher.login;

this.WeekName.Content = $"Неделя с {date1.ToShortDateString()} - {date2.ToShortDateString()}";

fileName = $"Неделя с {date1.ToShortDateString()} - {date2.ToShortDateString()}.json";

string projectDir1 = AppDomain.CurrentDomain.BaseDirectory;

string dir1 = System.IO.Path.Combine(projectDir1, $"Расписание\\{teachLogin}");

filesInDirectory = Directory.GetFiles(dir1).OrderBy(file => File.GetCreationTime(file)).ToList();

if(filesInDirectory.Count > 0)

{

indexOfFile = filesInDirectory.Count - 1;

try

{

string json = File.ReadAllText(filesInDirectory[indexOfFile]);

weekSubjects = JsonConvert.DeserializeObject<BindingList<WeekSubjects>>(json);

FileInfo fileInfo = new FileInfo(filesInDirectory[indexOfFile]);

this.FileName.Content = $"Файл от {File.GetLastWriteTime(fileInfo.FullName).ToShortDateString()}";

SourceGrid.ItemsSource = weekSubjects;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке данных: {ex.Message}");

}

}

List<Hours> hs = TeachHoursEntities2.GetContext().Hours.Where(h => h.teacherId == teacher.id).ToList();

allSubjectHours = new BindingList<AllSubjectHours>();

UpdateHoursGrid();

}

private void SaveUrlButton\_Click(object sender, RoutedEventArgs e)

{

if (MessageBox.Show("Вы точно хотите изменить ссылку для расписания?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Warning) == MessageBoxResult.Yes)

{

if (this.UrlBox.Text.Length == 0)

{

MessageBox.Show("Поле для расписания пустое.");

}

else

{

if(TeachersRepository.ChangeUrl(teacher, this.UrlBox.Text))

{

MessageBox.Show("Новая ссылка сохранена.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show("При сохранении новой ссылки произошла ошибка.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

static bool IsInternetAvailable()

{

try

{

using (var ping = new Ping())

{

var reply = ping.Send("8.8.8.8", 3000); // Используется адрес DNS-сервера Google для проверки подключения

return reply.Status == IPStatus.Success;

}

}

catch

{

return false;

}

}

private async void LoadButton\_Click(object sender, RoutedEventArgs e)

{

if(IsInternetAvailable())

{

try

{

var url = this.teacher.url;

if (url!=null && !url.Contains("teacher"))

{

throw new Exception();

}

var monday = await HtmlParser.GetDayOfWeek(url, $"tbody tr div.pair.lw\_1:not(.removed) div.subject", $"tbody tr div.pair.lw\_1:not(.removed) div.group span.group-span a");

var tuesday = await HtmlParser.GetDayOfWeek(url, $"tbody tr div.pair.lw\_2:not(.removed) div.subject", $"tbody tr div.pair.lw\_2:not(.removed) div.group span.group-span a");

var wednesday = await HtmlParser.GetDayOfWeek(url, $"tbody tr div.pair.lw\_3:not(.removed) div.subject", $"tbody tr div.pair.lw\_3:not(.removed) div.group span.group-span a");

var thursday = await HtmlParser.GetDayOfWeek(url, $"tbody tr div.pair.lw\_4:not(.removed) div.subject", $"tbody tr div.pair.lw\_4:not(.removed) div.group span.group-span a");

var friday = await HtmlParser.GetDayOfWeek(url, $"tbody tr div.pair.lw\_5:not(.removed) div.subject", $"tbody tr div.pair.lw\_5:not(.removed) div.group span.group-span a");

var satuday = await HtmlParser.GetDayOfWeek(url, $"tbody tr div.pair.lw\_6:not(.removed) div.subject", $"tbody tr div.pair.lw\_6:not(.removed) div.group span.group-span a");

if(monday == null || tuesday == null || wednesday == null || thursday == null || friday == null || satuday == null)

{

throw new Exception();

}

int maxValue = new int[] { monday.Count, tuesday.Count, wednesday.Count, thursday.Count, friday.Count, satuday.Count }.Max();

int indMon = 0, indTues = 0, indWed = 0, indThurs = 0, indFrid = 0, indSat = 0;

weekSubjects = new BindingList<WeekSubjects>();

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

{

string mon, tues, wed, thurs, frid, sat;

if (indMon > monday.Count - 1)

{

mon = "";

}

else

{

mon = monday[indMon++];

}

if (indTues > tuesday.Count - 1)

{

tues = "";

}

else

{

tues = tuesday[indTues++];

}

if (indWed > wednesday.Count - 1)

{

wed = "";

}

else

{

wed = wednesday[indWed++];

}

if (indThurs > thursday.Count - 1)

{

thurs = "";

}

else

{

thurs = thursday[indThurs++];

}

if (indFrid > friday.Count - 1)

{

frid = "";

}

else

{

frid = friday[indFrid++];

}

if (indSat > satuday.Count - 1)

{

sat = "";

}

else

{

sat = satuday[indSat++];

}

weekSubjects.Add(new WeekSubjects(mon, tues, wed, thurs, frid, sat));

}

SourceGrid.Columns.Clear();

SourceGrid.ItemsSource = null;

SourceGrid.Items.Clear();

week = DateTime.Now;

//CreateWeekGrid(week);

while (week.DayOfWeek != DayOfWeek.Monday)

{

week = week.AddDays(-1);

}

DateTime week1 = week;

DateTime date1 = DateTime.Now;

DateTime date2 = DateTime.Now;

for (int i = 1; i < 7; i++, week1 = week1.AddDays(1))

{

if (i == 1)

{

var column = new DataGridTextColumn

{

Header = $"Понедельник\n{week1.ToShortDateString()}",

Binding = new Binding("Monday")

};

this.SourceGrid.Columns.Add(column);

date1 = week1;

}

if (i == 2)

{

var column = new DataGridTextColumn

{

Header = $"Вторник\n{week1.ToShortDateString()}",

Binding = new Binding("Tuesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 3)

{

var column = new DataGridTextColumn

{

Header = $"Среда\n{week1.ToShortDateString()}",

Binding = new Binding("Wednesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 4)

{

var column = new DataGridTextColumn

{

Header = $"Четверг\n{week1.ToShortDateString()}",

Binding = new Binding("Thursday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 5)

{

var column = new DataGridTextColumn

{

Header = $"Пятница\n{week1.ToShortDateString()}",

Binding = new Binding("Friday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 6)

{

var column = new DataGridTextColumn

{

Header = $"Суббота\n{week1.ToShortDateString()}",

Binding = new Binding("Saturday")

};

this.SourceGrid.Columns.Add(column);

date2 = week1;

}

}

indexOfFile = filesInDirectory.Count - 1;

SourceGrid.ItemsSource = weekSubjects;

sourceSave = false;

}

catch(Exception ex)

{

MessageBox.Show("Ошибка соединения с сайтом колледжа.\nПроверьте корректность ссылки.\nСсылка должна указывать на раписание преподавателя.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Отсутствует интернет-соединение.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void SaveFile\_Click(object sender, RoutedEventArgs e)

{

if(weekSubjects != null && weekSubjects.Count > 0)

{

var teachLogin = teacher.login;

string projectDir = AppDomain.CurrentDomain.BaseDirectory;

string dir = System.IO.Path.Combine(projectDir, $"Расписание\\{teachLogin}");

string file = fileName;

string json = JsonConvert.SerializeObject(weekSubjects, Formatting.Indented);

File.WriteAllText($"{dir}\\{file}", json);

MessageBox.Show("Раписание успешно сохранено." , "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

string projectDir1 = AppDomain.CurrentDomain.BaseDirectory;

string dir1 = System.IO.Path.Combine(projectDir1, $"Расписание\\{teachLogin}");

filesInDirectory = Directory.GetFiles(dir1).OrderBy(file1 => File.GetCreationTime(file1)).ToList();

indexOfFile = filesInDirectory.Count - 1;

string json1 = File.ReadAllText(filesInDirectory[indexOfFile]);

weekSubjects = JsonConvert.DeserializeObject<BindingList<WeekSubjects>>(json1);

FileInfo fileinfo = new FileInfo(filesInDirectory[indexOfFile]);

this.FileName.Content = $"Файл от {File.GetLastWriteTime(fileinfo.FullName).ToShortDateString()}";

week = DateTime.Now;

SourceGrid.Columns.Clear();

SourceGrid.ItemsSource = null;

SourceGrid.Items.Clear();

week = DateTime.Now;

while (week.DayOfWeek != DayOfWeek.Monday)

{

week = week.AddDays(-1);

}

DateTime week1 = week;

DateTime date1 = DateTime.Now;

DateTime date2 = DateTime.Now;

for (int i = 1; i < 7; i++, week1 = week1.AddDays(1))

{

if (i == 1)

{

var column = new DataGridTextColumn

{

Header = $"Понедельник\n{week1.ToShortDateString()}",

Binding = new Binding("Monday")

};

this.SourceGrid.Columns.Add(column);

date1 = week1;

}

if (i == 2)

{

var column = new DataGridTextColumn

{

Header = $"Вторник\n{week1.ToShortDateString()}",

Binding = new Binding("Tuesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 3)

{

var column = new DataGridTextColumn

{

Header = $"Среда\n{week1.ToShortDateString()}",

Binding = new Binding("Wednesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 4)

{

var column = new DataGridTextColumn

{

Header = $"Четверг\n{week1.ToShortDateString()}",

Binding = new Binding("Thursday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 5)

{

var column = new DataGridTextColumn

{

Header = $"Пятница\n{week1.ToShortDateString()}",

Binding = new Binding("Friday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 6)

{

var column = new DataGridTextColumn

{

Header = $"Суббота\n{week1.ToShortDateString()}",

Binding = new Binding("Saturday")

};

this.SourceGrid.Columns.Add(column);

date2 = week1;

}

}

this.WeekName.Content = $"Неделя с {date1.ToShortDateString()} - {date2.ToShortDateString()}";

SourceGrid.ItemsSource = weekSubjects;

sourceSave = true;

}

else

{

MessageBox.Show("Сначала загрузите расписание с сайта колледжа.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

void AddSubjectsToDB(List<string> day, DateTime date)

{

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

{

Dates dayDate = new Dates();

dayDate.date = date;

DatesRepository.Add(dayDate);

Subjects subject = new Subjects();

subject.name = day[i];

subject.teacherId = teacher.id;

SubjectsRepository.Add(subject);

Hours hour = new Hours();

hour.dateId = dayDate.id;

hour.subjectId = subject.id;

hour.teacherId = teacher.id;

HoursRepository.Add(hour);

}

}

private void SaveDB\_Click(object sender, RoutedEventArgs e)

{

if (weekSubjects != null && weekSubjects.Count > 0)

{

DateTime date = DateTime.Now;

while(date.DayOfWeek != DayOfWeek.Monday)

{

date = date.AddDays(-1);

}

var monday = (from i in weekSubjects

where i.Monday != "" && !i.Monday.Trim().ToLower().Contains("Урок снят".Trim().ToLower())

select i.Monday).ToList<string>();

var tuesday = (from i in weekSubjects

where i.Tuesday != "" && !i.Tuesday.Trim().ToLower().Contains("Урок снят".Trim().ToLower())

select i.Tuesday).ToList<string>();

var wednesday = (from i in weekSubjects

where i.Wednesday != "" && !i.Wednesday.Trim().ToLower().Contains("Урок снят".Trim().ToLower())

select i.Wednesday).ToList<string>();

var thursday = (from i in weekSubjects

where i.Thursday != "" && !i.Thursday.Trim().ToLower().Contains("Урок снят".Trim().ToLower())

select i.Thursday).ToList<string>();

var friday = (from i in weekSubjects

where i.Friday != "" && !i.Friday.Trim().ToLower().Contains("Урок снят".Trim().ToLower())

select i.Friday).ToList<string>();

var saturday = (from i in weekSubjects

where i.Saturday != "" && !i.Saturday.Trim().ToLower().Contains("Урок снят".Trim().ToLower())

select i.Saturday).ToList<string>();

try

{

AddSubjectsToDB(monday, date);

date = date.AddDays(1);

AddSubjectsToDB(tuesday, date);

date = date.AddDays(1);

AddSubjectsToDB(wednesday, date);

date = date.AddDays(1);

AddSubjectsToDB(thursday, date);

date = date.AddDays(1);

AddSubjectsToDB(friday, date);

date = date.AddDays(1);

AddSubjectsToDB(saturday, date);

MessageBox.Show("Расписание успешно сохранено.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

UpdateHoursGrid();

CreateAllHoursTable();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Сначала загрузите расписание с сайта колледжа.", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

var removeHours = this.HoursGrid.SelectedItems.Cast<AllSubjectHours>().ToList();

if(MessageBox.Show($"Вы действительно хотите удалить {removeHours.Count} часов?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

try

{

List<Hours> remove = (from i in removeHours

select HoursRepository.GetHourById(i.Id)).ToList();

List<Dates> removeDates = (from i in remove

select DatesRepository.GetDateById(i.dateId)).ToList();

List<Subjects> removeSubjects = (from i in remove

select SubjectsRepository.GetSubjectById(i.subjectId)).ToList();

HoursRepository.DeleteRange(remove);

DatesRepository.DeleteRange(removeDates);

SubjectsRepository.DeleteRange(removeSubjects);

foreach(var hour in removeHours)

{

allSubjectHours.Remove(hour);

}

UpdateHoursGrid();

MessageBox.Show("Часы успешно удалены.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

CreateAllHoursTable();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

public void UpdateHoursGrid()

{

List<Hours> hours = HoursRepository.GetTeacherHours(teacher.id);

allSubjectHours.Clear();

foreach (var h in hours)

{

int id = h.id;

string subject = SubjectsRepository.GetSubjectById(h.subjectId).name;

DateTime date = DatesRepository.GetDateById(h.dateId).date;

allSubjectHours.Add(new AllSubjectHours(id, date, subject));

}

HoursGrid.ItemsSource = allSubjectHours;

List<string> uniqGroups = new List<string>();

List<string> uniqSubjects = new List<string>();

foreach (var s in allSubjectHours)

{

var temp = s.Subject.Split(' ');

if (!uniqGroups.Contains(temp[0]))

{

uniqGroups.Add(temp[0]);

}

if (!uniqSubjects.Contains(temp[1]))

{

uniqSubjects.Add(temp[1]);

}

}

this.ViewSubjects.ItemsSource = uniqSubjects;

this.GroupsBox.ItemsSource = uniqGroups;

}

private void ButtonAddHour\_Click(object sender, RoutedEventArgs e)

{

AddHourWindow page = new AddHourWindow(teacher.id);

if(page.ShowDialog() == true)

{

UpdateHoursGrid();

CreateAllHoursTable();

}

}

private void ReportButton\_Click(object sender, RoutedEventArgs e)

{

if(MonthBox.SelectedItem != null && YearBox.SelectedItem != null)

{

switch (MonthBox.SelectedItem.ToString())

{

case "Январь":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 1, 1);

break;

case "Февраль":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 2, 1);

break;

case "Март":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 3, 1);

break;

case "Апрель":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 4, 1);

break;

case "Май":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 5, 1);

break;

case "Июнь":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 6, 1);

break;

case "Июль":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 7, 1);

break;

case "Август":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 8, 1);

break;

case "Сентябрь":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 9, 1);

break;

case "Октябрь":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 10, 1);

break;

case "Ноябрь":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 11, 1);

break;

case "Декабрь":

reportDate = new DateTime(int.Parse(YearBox.SelectedItem.ToString()), 12, 1);

break;

}

List<string> subjectsInGroups = new List<string>();

List<string> datesOfSubjects = new List<string>();

foreach (var s in allSubjectHours)

{

DateTime data = DateTime.Parse(s.Date);

if (!subjectsInGroups.Contains(s.Subject) && data.Month == reportDate.Month && data.Year == reportDate.Year)

{

subjectsInGroups.Add(s.Subject);

}

if (!datesOfSubjects.Contains(s.Date) && DateTime.Parse(s.Date).Year == reportDate.Year)

{

datesOfSubjects.Add(s.Date);

}

}

List<List<string>> resultReport = new List<List<string>>();

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

{

resultReport.Add(new List<string>());

resultReport[i].Add(subjectsInGroups[i]);

int countOfHours = 0;

for(int j = 0; j < datesOfSubjects.Count; j++)

{

foreach(var s in allSubjectHours)

{

if(s.Subject == subjectsInGroups[i] && s.Date.CompareTo(datesOfSubjects[j])==0)

{

countOfHours++;

}

}

if(countOfHours > 0)

{

resultReport[i].Add($"{datesOfSubjects[j]} - {countOfHours}");

countOfHours = 0;

}

}

}

if(resultReport.Count > 0)

{

WordReport reporter = new WordReport();

reporter.GenerateReport(id, resultReport, MonthBox.SelectedItem.ToString(), YearBox.SelectedItem.ToString());

CreateAllHoursTable();

MessageBox.Show("Отчёт успешно сохранён!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show("Предметов на данный месяц и год не найдено.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Выберите месяц и год для формирования отчёта.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

public void GetHour(int t)

{

this.allHoursOfSubject = t;

}

private void FilterButton\_Click(object sender, RoutedEventArgs e)

{

filterSubjectHours = allSubjectHours;

if(ViewSubjects.SelectedItems.Count > 0)

{

var tempList = new BindingList<AllSubjectHours>();

foreach(var t in ViewSubjects.SelectedItems)

{

foreach(var item in filterSubjectHours)

{

if (item.Subject.Contains(t.ToString()))

{

tempList.Add(item);

}

}

}

filterSubjectHours = tempList;

}

if(GroupsBox.SelectedItem != null)

{

var tempList = new BindingList<AllSubjectHours>();

var group = GroupsBox.SelectedItem.ToString();

foreach (var item in filterSubjectHours)

{

if (item.Subject.Contains(group))

{

tempList.Add(item);

}

}

filterSubjectHours = tempList;

}

if(PickerStart.SelectedDate != null)

{

var tempList = new BindingList<AllSubjectHours>();

var start = PickerStart.SelectedDate;

foreach(var sbj in filterSubjectHours)

{

DateTime.TryParse(sbj.Date, out DateTime date);

if(date >= start)

{

tempList.Add(sbj);

}

}

filterSubjectHours = tempList;

}

if(PickerEnd.SelectedDate != null)

{

var tempList = new BindingList<AllSubjectHours>();

var end = PickerEnd.SelectedDate;

foreach (var sbj in filterSubjectHours)

{

DateTime.TryParse(sbj.Date, out DateTime date);

if (date <= end)

{

tempList.Add(sbj);

}

}

filterSubjectHours = tempList;

}

if(filterSubjectHours.Count > 0)

{

HoursGrid.ItemsSource = filterSubjectHours;

}

else

{

MessageBox.Show("По вашему запросу ничего не найдено.");

ViewSubjects.SelectedItems.Clear();

GroupsBox.SelectedItem = null;

PickerStart.SelectedDate = null;

PickerEnd.SelectedDate = null;

UpdateHoursGrid();

}

}

private void SubjectTitle\_Click(object sender, MouseButtonEventArgs e)

{

ViewSubjects.SelectedItems.Clear();

}

private void Prev\_Click(object sender, RoutedEventArgs e)

{

if(indexOfFile <= 0)

{

Prev.IsEnabled = false;

}

else

{

if(sourceSave == false)

{

if (MessageBox.Show($"Расписание не сохранено, перелистывание приведёт к потере данных.\nСохранить раписание?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

if (weekSubjects != null && weekSubjects.Count > 0)

{

var teachLogin = teacher.login;

string projectDir = AppDomain.CurrentDomain.BaseDirectory;

string dir = System.IO.Path.Combine(projectDir, $"Расписание\\{teachLogin}");

string file = fileName;

string json = JsonConvert.SerializeObject(weekSubjects, Formatting.Indented);

File.WriteAllText($"{dir}\\{file}", json);

MessageBox.Show("Файл успешно сохранён.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

string projectDir1 = AppDomain.CurrentDomain.BaseDirectory;

string dir1 = System.IO.Path.Combine(projectDir1, $"Расписание\\{teachLogin}");

filesInDirectory = Directory.GetFiles(dir1).OrderBy(file1 => File.GetCreationTime(file1)).ToList();

indexOfFile = filesInDirectory.Count - 1;

string json1 = File.ReadAllText(filesInDirectory[indexOfFile]);

weekSubjects = JsonConvert.DeserializeObject<BindingList<WeekSubjects>>(json1);

FileInfo fileinfo = new FileInfo(filesInDirectory[indexOfFile]);

this.FileName.Content = $"Файл от {File.GetLastWriteTime(fileinfo.FullName).ToShortDateString()}";

SourceGrid.Columns.Clear();

SourceGrid.ItemsSource = null;

SourceGrid.Items.Clear();

week = DateTime.Now;

while (week.DayOfWeek != DayOfWeek.Monday)

{

week = week.AddDays(-1);

}

DateTime week1 = week;

DateTime date1 = DateTime.Now;

DateTime date2 = DateTime.Now;

for (int i = 1; i < 7; i++, week1 = week1.AddDays(1))

{

if (i == 1)

{

var column = new DataGridTextColumn

{

Header = $"Понедельник\n{week1.ToShortDateString()}",

Binding = new Binding("Monday")

};

this.SourceGrid.Columns.Add(column);

date1 = week1;

}

if (i == 2)

{

var column = new DataGridTextColumn

{

Header = $"Вторник\n{week1.ToShortDateString()}",

Binding = new Binding("Tuesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 3)

{

var column = new DataGridTextColumn

{

Header = $"Среда\n{week1.ToShortDateString()}",

Binding = new Binding("Wednesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 4)

{

var column = new DataGridTextColumn

{

Header = $"Четверг\n{week1.ToShortDateString()}",

Binding = new Binding("Thursday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 5)

{

var column = new DataGridTextColumn

{

Header = $"Пятница\n{week1.ToShortDateString()}",

Binding = new Binding("Friday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 6)

{

var column = new DataGridTextColumn

{

Header = $"Суббота\n{week1.ToShortDateString()}",

Binding = new Binding("Saturday")

};

this.SourceGrid.Columns.Add(column);

date2 = week1;

}

}

this.WeekName.Content = $"Неделя с {date1.ToShortDateString()} - {date2.ToShortDateString()}";

SourceGrid.ItemsSource = weekSubjects;

sourceSave = true;

}

else

{

MessageBox.Show("Сначала откройте или загрузите расписание.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

sourceSave = true;

}

}

else

{

Next.IsEnabled = true;

indexOfFile--;

try

{

string json = File.ReadAllText(filesInDirectory[indexOfFile]);

weekSubjects = JsonConvert.DeserializeObject<BindingList<WeekSubjects>>(json);

FileInfo fileinfo = new FileInfo(filesInDirectory[indexOfFile]);

this.FileName.Content = $"Файл от {File.GetLastWriteTime(fileinfo.FullName).ToShortDateString()}";

SourceGrid.Columns.Clear();

SourceGrid.ItemsSource = null;

SourceGrid.Items.Clear(); week = week.AddDays(-7);

DateTime week1 = week;

DateTime date1 = DateTime.Now;

DateTime date2 = DateTime.Now;

for (int i = 1; i < 7; i++, week1 = week1.AddDays(1))

{

if (i == 1)

{

var column = new DataGridTextColumn

{

Header = $"Понедельник\n{week1.ToShortDateString()}",

Binding = new Binding("Monday")

};

this.SourceGrid.Columns.Add(column);

date1 = week1;

}

if (i == 2)

{

var column = new DataGridTextColumn

{

Header = $"Вторник\n{week1.ToShortDateString()}",

Binding = new Binding("Tuesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 3)

{

var column = new DataGridTextColumn

{

Header = $"Среда\n{week1.ToShortDateString()}",

Binding = new Binding("Wednesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 4)

{

var column = new DataGridTextColumn

{

Header = $"Четверг\n{week1.ToShortDateString()}",

Binding = new Binding("Thursday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 5)

{

var column = new DataGridTextColumn

{

Header = $"Пятница\n{week1.ToShortDateString()}",

Binding = new Binding("Friday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 6)

{

var column = new DataGridTextColumn

{

Header = $"Суббота\n{week1.ToShortDateString()}",

Binding = new Binding("Saturday")

};

this.SourceGrid.Columns.Add(column);

date2 = week1;

}

}

this.WeekName.Content = $"Неделя с {date1.ToShortDateString()} - {date2.ToShortDateString()}";

SourceGrid.ItemsSource = weekSubjects;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке данных: {ex.Message}", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

private void Next\_Click(object sender, RoutedEventArgs e)

{

if(indexOfFile >= filesInDirectory.Count - 1)

{

Next.IsEnabled = false;

}

else

{

if (sourceSave == false)

{

if (MessageBox.Show($"Расписание не сохранено, перелистывание приведёт к потере данных.\nСохранить раписание?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

if (weekSubjects != null && weekSubjects.Count > 0)

{

var teachLogin = teacher.login;

string projectDir = AppDomain.CurrentDomain.BaseDirectory;

string dir = System.IO.Path.Combine(projectDir, $"Расписание\\{teachLogin}");

string file = fileName;

string json = JsonConvert.SerializeObject(weekSubjects, Formatting.Indented);

File.WriteAllText($"{dir}\\{file}", json);

MessageBox.Show("Файл успешно сохранён.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

string projectDir1 = AppDomain.CurrentDomain.BaseDirectory;

string dir1 = System.IO.Path.Combine(projectDir1, $"Расписание\\{teachLogin}");

filesInDirectory = Directory.GetFiles(dir1).OrderBy(file1 => File.GetCreationTime(file1)).ToList();

indexOfFile = filesInDirectory.Count-1;

string json1 = File.ReadAllText(filesInDirectory[indexOfFile]);

weekSubjects = JsonConvert.DeserializeObject<BindingList<WeekSubjects>>(json1);

FileInfo fileinfo = new FileInfo(filesInDirectory[indexOfFile]);

this.FileName.Content = $"Файл от {File.GetLastWriteTime(fileinfo.FullName).ToShortDateString()}";

SourceGrid.Columns.Clear();

SourceGrid.ItemsSource = null;

SourceGrid.Items.Clear(); week = DateTime.Now;

while(week.DayOfWeek != DayOfWeek.Monday)

{

week = week.AddDays(-1);

}

DateTime week1 = week;

DateTime date1 = DateTime.Now;

DateTime date2 = DateTime.Now;

for (int i = 1; i < 7; i++, week1 = week1.AddDays(1))

{

if (i == 1)

{

var column = new DataGridTextColumn

{

Header = $"Понедельник\n{week1.ToShortDateString()}",

Binding = new Binding("Monday")

};

this.SourceGrid.Columns.Add(column);

date1 = week1;

}

if (i == 2)

{

var column = new DataGridTextColumn

{

Header = $"Вторник\n{week1.ToShortDateString()}",

Binding = new Binding("Tuesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 3)

{

var column = new DataGridTextColumn

{

Header = $"Среда\n{week1.ToShortDateString()}",

Binding = new Binding("Wednesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 4)

{

var column = new DataGridTextColumn

{

Header = $"Четверг\n{week1.ToShortDateString()}",

Binding = new Binding("Thursday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 5)

{

var column = new DataGridTextColumn

{

Header = $"Пятница\n{week1.ToShortDateString()}",

Binding = new Binding("Friday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 6)

{

var column = new DataGridTextColumn

{

Header = $"Суббота\n{week1.ToShortDateString()}",

Binding = new Binding("Saturday")

};

this.SourceGrid.Columns.Add(column);

date2 = week1;

}

}

this.WeekName.Content = $"Неделя с {date1.ToShortDateString()} - {date2.ToShortDateString()}";

SourceGrid.ItemsSource = weekSubjects;

sourceSave = true;

}

else

{

MessageBox.Show("Сначала загрузите расписание с сайте колледжа.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

sourceSave = true;

}

}

else

{

Prev.IsEnabled = true;

indexOfFile++;

try

{

string json = File.ReadAllText(filesInDirectory[indexOfFile]);

weekSubjects = JsonConvert.DeserializeObject<BindingList<WeekSubjects>>(json);

FileInfo fileinfo = new FileInfo(filesInDirectory[indexOfFile]);

this.FileName.Content = $"Файл от {File.GetLastWriteTime(fileinfo.FullName).ToShortDateString()}";

SourceGrid.Columns.Clear();

SourceGrid.ItemsSource = null;

SourceGrid.Items.Clear(); week = week.AddDays(7);

DateTime week1 = week;

DateTime date1 = DateTime.Now;

DateTime date2 = DateTime.Now;

for (int i = 1; i < 7; i++, week1 = week1.AddDays(1))

{

if (i == 1)

{

var column = new DataGridTextColumn

{

Header = $"Понедельник\n{week1.ToShortDateString()}",

Binding = new Binding("Monday")

};

this.SourceGrid.Columns.Add(column);

date1 = week1;

}

if (i == 2)

{

var column = new DataGridTextColumn

{

Header = $"Вторник\n{week1.ToShortDateString()}",

Binding = new Binding("Tuesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 3)

{

var column = new DataGridTextColumn

{

Header = $"Среда\n{week1.ToShortDateString()}",

Binding = new Binding("Wednesday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 4)

{

var column = new DataGridTextColumn

{

Header = $"Четверг\n{week1.ToShortDateString()}",

Binding = new Binding("Thursday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 5)

{

var column = new DataGridTextColumn

{

Header = $"Пятница\n{week1.ToShortDateString()}",

Binding = new Binding("Friday")

};

this.SourceGrid.Columns.Add(column);

}

if (i == 6)

{

var column = new DataGridTextColumn

{

Header = $"Суббота\n{week1.ToShortDateString()}",

Binding = new Binding("Saturday")

};

this.SourceGrid.Columns.Add(column);

date2 = week1;

}

}

this.WeekName.Content = $"Неделя с {date1.ToShortDateString()} - {date2.ToShortDateString()}";

SourceGrid.ItemsSource = weekSubjects;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке данных: {ex.Message}", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

private void ClearFilters\_Click(object sender, RoutedEventArgs e)

{

ViewSubjects.SelectedItems.Clear();

GroupsBox.SelectedItem = null;

PickerStart.SelectedDate = null;

PickerEnd.SelectedDate = null;

UpdateHoursGrid();

}

private void CreateAllHoursTable()

{

var allHours = AllHoursRepository.GetAllHoursOfCurrentTeacher(teacher.id);

List<string> subs = new List<string>();

var currentSubs = allHours.Select(s => s.subjectName.ToString()).ToList();

if(allSubjectHours.Count > 0)

{

foreach (var sub in allSubjectHours)

{

if (!subs.Contains(sub.Subject.ToString()))

{

subs.Add(sub.Subject.ToString());

}

}

foreach (var sub in subs)

{

if (!currentSubs.Contains(sub.ToString()))

{

currentSubs.Add(sub.ToString());

AllHoursOfSubjects window = new AllHoursOfSubjects(sub);

window.temp = GetHour;

window.ShowDialog();

var h = new AllHours();

h.subjectName = sub;

h.countHours = allHoursOfSubject;

h.teacherId = id;

AllHoursRepository.Add(h);

}

}

allHours = AllHoursRepository.GetAllHoursOfCurrentTeacher(teacher.id);

currentSubs = allHours.Select(s => s.subjectName.ToString()).ToList();

foreach (var sub in currentSubs)

{

if (!subs.Contains(sub))

{

var sb = AllHoursRepository.GetTeacherHourByName(id, sub);

AllHoursRepository.Delete(sb);

}

}

}

ResultGrid.ItemsSource = AllHoursRepository.GetAllHoursOfCurrentTeacher(id);

}

public void CreateTeachFolder(int id)

{

var teachLogin = teacher.login;

string projectDir = AppDomain.CurrentDomain.BaseDirectory;

string dir = System.IO.Path.Combine(projectDir, $"Расписание\\{teachLogin}");

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

}

private void ChangeHour\_Click(object sender, RoutedEventArgs e)

{

try

{

var button = sender as Button;

var hour = button?.DataContext as AllHours;

var subj = AllHoursRepository.GetHourById(hour.id);

if (hour.countHours < 0)

{

MessageBox.Show("Общее кол-во часов не может быть меньше 0", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

AllHoursRepository.ChangeCountOfHours(subj, hour.countHours);

MessageBox.Show("Изменения сохранены!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

CreateAllHoursTable();

}

}

catch(Exception)

{

MessageBox.Show("Введите не корректные данные.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void Window\_Close(object sender, CancelEventArgs e)

{

if(sourceSave == false)

{

if (MessageBox.Show($"Расписание не сохранено, перелистывание приведёт к потере данных.\nСохранить раписание?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

if (weekSubjects != null && weekSubjects.Count > 0)

{

var teachLogin = teacher.login;

string projectDir = AppDomain.CurrentDomain.BaseDirectory;

string dir = System.IO.Path.Combine(projectDir, $"Расписание\\{teachLogin}");

string file = fileName;

string json = JsonConvert.SerializeObject(weekSubjects, Formatting.Indented);

File.WriteAllText($"{dir}\\{file}", json);

MessageBox.Show("Файл успешно сохранён.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

}

private void Spravka\_Click(object sender, RoutedEventArgs e)

{

string helpPath = System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "spravka.chm");

System.Diagnostics.Process.Start(new System.Diagnostics.ProcessStartInfo(helpPath) { UseShellExecute = true });

}

}

//Код файла MainWindow.xaml.cs

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void RegButton\_Click(object sender, RoutedEventArgs e)

{

TeachReg validator = new TeachReg(this.RegLogText\_Box.Text, this.FirstRegPasText\_Box.Password, this.SecondRegPasText\_Box.Password);

if (validator.ValidatePasswords())

{

if(validator.ValidatePasswordLength())

{

var teachers = TeachersRepository.GetTeachersDictionary();

if (validator.ValidateLogin(teachers))

{

var login = this.RegLogText\_Box.Text;

var hashPas = PasswordHasher.HashFunc(this.FirstRegPasText\_Box.Password);

if(TeachersRepository.Save(login, hashPas.ToString()))

{

MessageBox.Show("Вы зарегистрированы!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

}

else

{

MessageBox.Show("Во время регитсрации произршла непредвиденная ошибка. Повторите попытку.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Данный логин уже занят.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Длина пароля должна составлять минимум 6 символов.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Пароли не совпадают.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void AuthButton\_Click(object sender, RoutedEventArgs e)

{

var hashPas = PasswordHasher.HashFunc(this.AuthPasText\_Box.Password);

TeachAuth validator = new TeachAuth(this.AuthLogText\_Box.Text, hashPas.ToString());

var teachersDict = TeachersRepository.GetTeachersDictionary();

var teachersList = TeachersRepository.GetTeachersList();

if (validator.ValidateLogin(teachersDict))

{

int teachId = validator.ValidatePassword(teachersList);

if (teachId > 0)

{

BaseWindow page = new BaseWindow(teachId);

page.Show();

this.Close();

}

else

{

MessageBox.Show("Неверный пароль.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Неверный логин.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void AuthPasTextBox\_Click(object sender, MouseButtonEventArgs e)

{

this.AuthPasText\_Box.Password = "";

}

private void AuthLogTextBox\_Click(object sender, MouseButtonEventArgs e)

{

this.AuthLogText\_Box.Text = "";

}

private void RegLog\_Click(object sender, MouseButtonEventArgs e)

{

this.RegLogText\_Box.Text = "";

}

private void RegPas1\_Click(object sender, MouseButtonEventArgs e)

{

this.FirstRegPasText\_Box.Password = "";

}

private void RegPas2\_Click(object sender, MouseButtonEventArgs e)

{

this.SecondRegPasText\_Box.Password = "";

}

private void Spravka\_Click(object sender, RoutedEventArgs e)

{

string helpPath = System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "spravka.chm");

System.Diagnostics.Process.Start(new System.Diagnostics.ProcessStartInfo(helpPath) { UseShellExecute = true });

}

}

//Код файла AllHoursOfSubjects.xaml.cs

public partial class AllHoursOfSubjects : Window

{

public AllHoursOfSubjects()

{

InitializeComponent();

}

public AllHoursOfSubjects(string title)

{

InitializeComponent();

this.titleSubject = title;

}

string titleSubject;

public BaseWindow.GetAllHourForSubject temp;

private void Window\_Load(object sender, RoutedEventArgs e)

{

this.Label1.Content = $"Количество часов по предмету\n{this.titleSubject}:";

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

try

{

int t = Convert.ToInt32(this.TextBox.Text);

if (t <= 0 || t > 1200)

{

throw new Exception();

}

temp(t);

this.Close();

}

catch (Exception ex)

{

MessageBox.Show("Некорректные данные.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void Box\_Click(object sender, MouseButtonEventArgs e)

{

this.TextBox.Text = "";

}

}

# ПРИЛОЖЕНИЕ Б

**(справочное)**

**Формы входных и выходных документов**



Рисунок Б.1 – Пример отчёта за выбранный месяц по отработанным часам преподавателя

# ПРИЛОЖЕНИЕ В

**(справочное)**

**Результаты работы приложения**

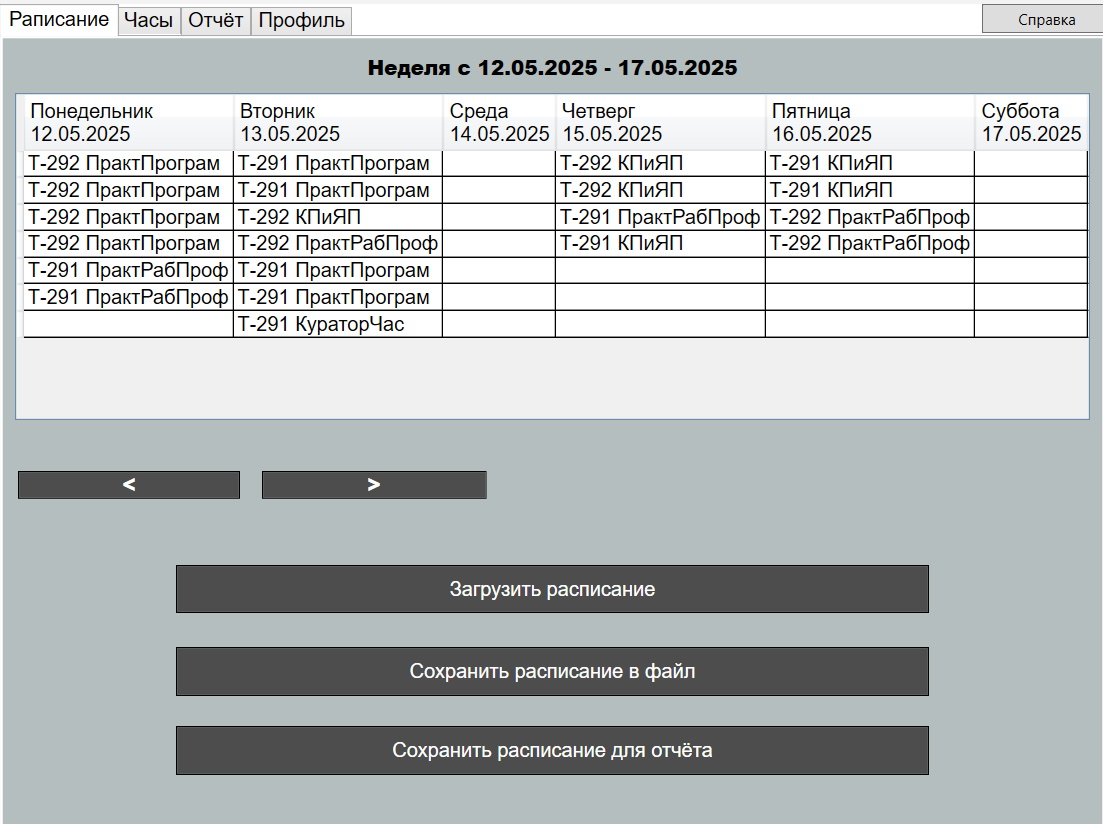


Рисунок В.1 – Проверка парсинга расписания по корректной ссылке

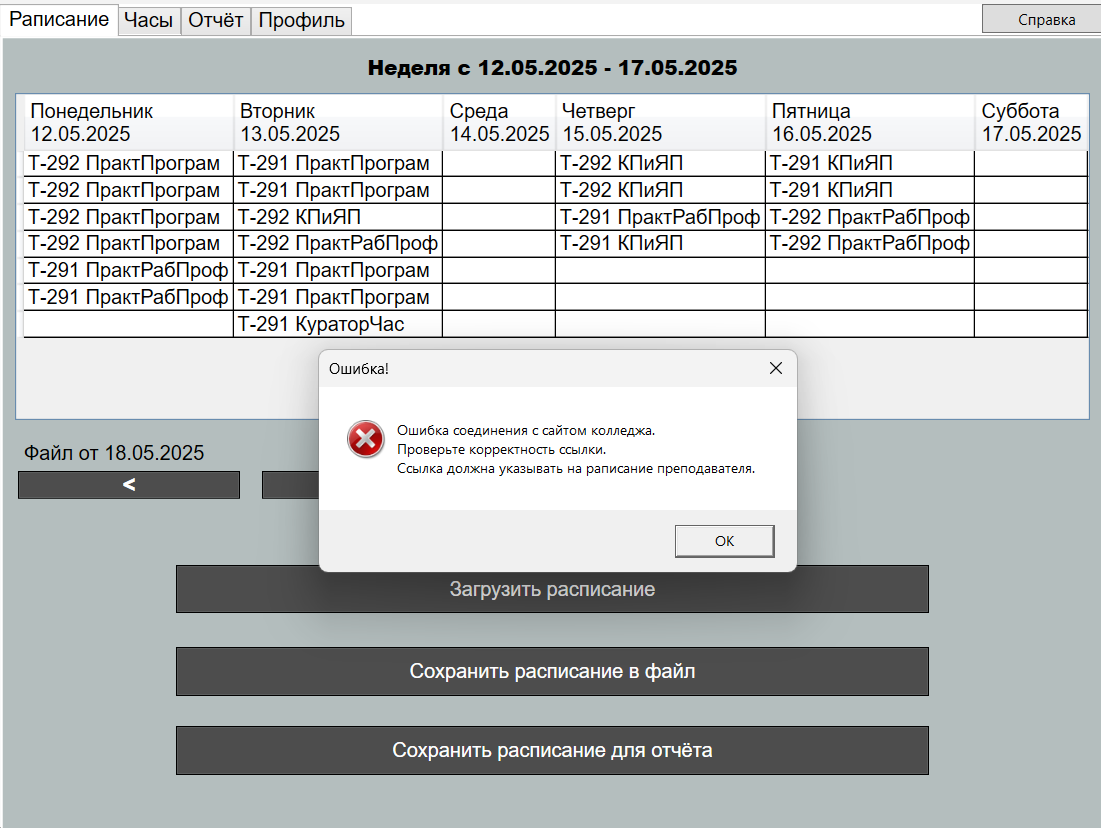


Рисунок В.2 – Проверка парсинга расписания по некорректной ссылке

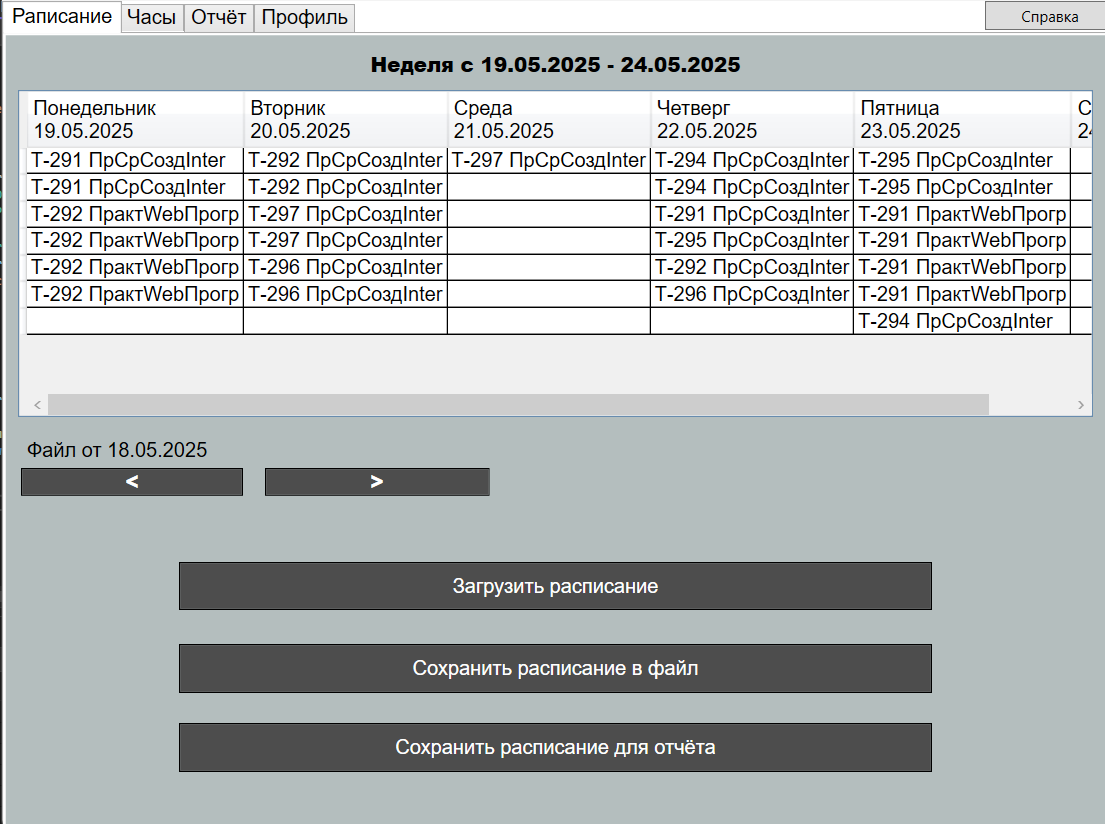


Рисунок В.3 – Проверка парсинга расписания с исправным интернет-соединением

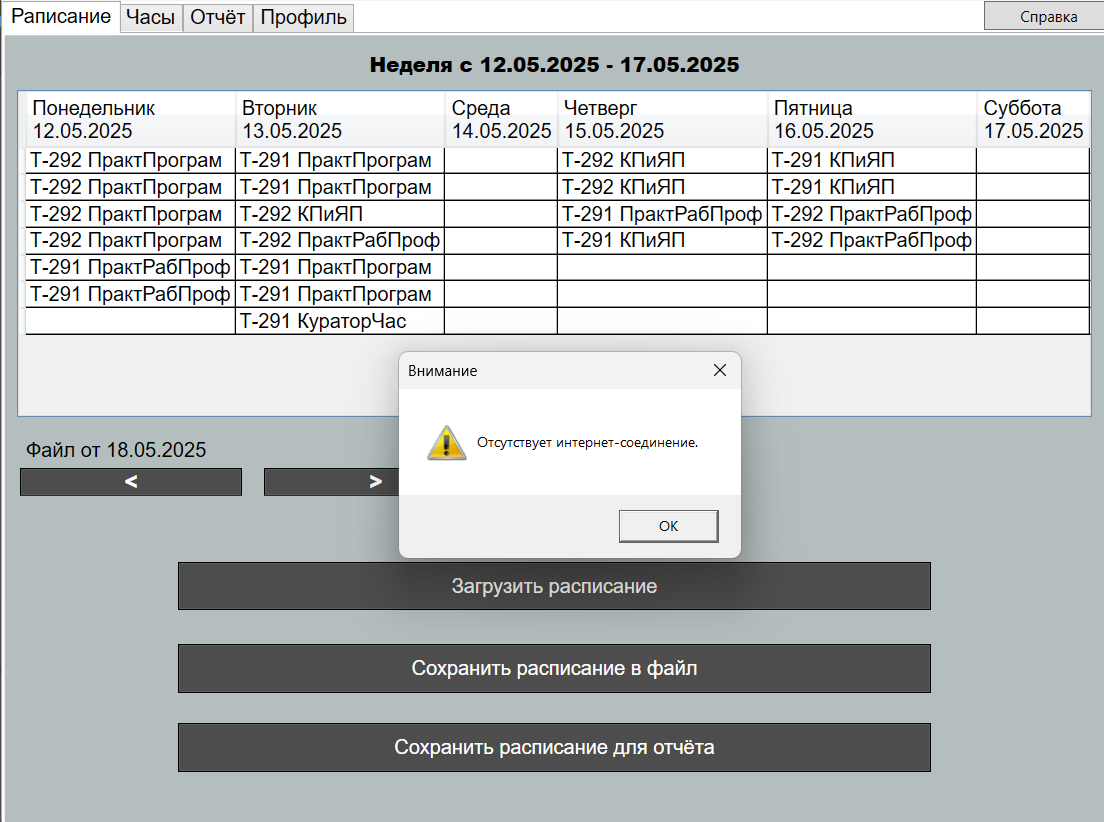


Рисунок В.4 – Проверка парсинга расписания с неисправным интернет-соединением

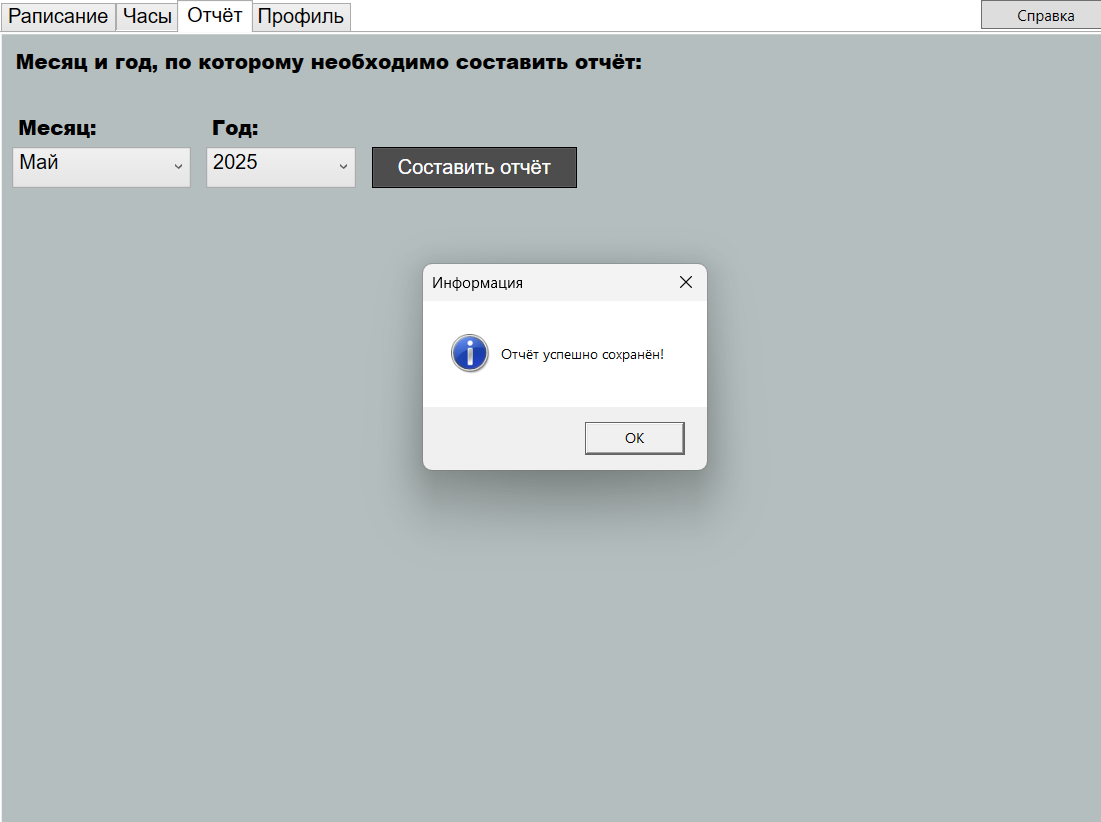


Рисунок В.5 – Проверка составления отчёта с выбранными месяцем и годом

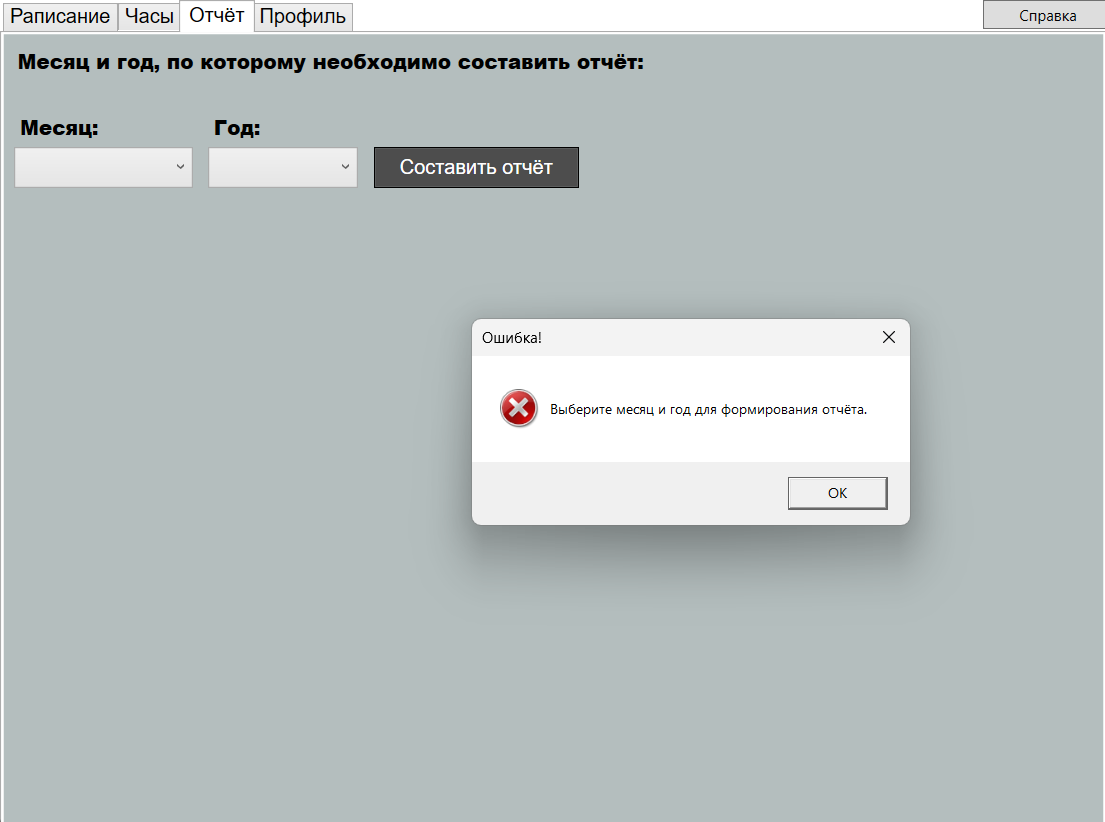


Рисунок В.6 – Проверка составления отчёта с невыбранными месяцем и годом

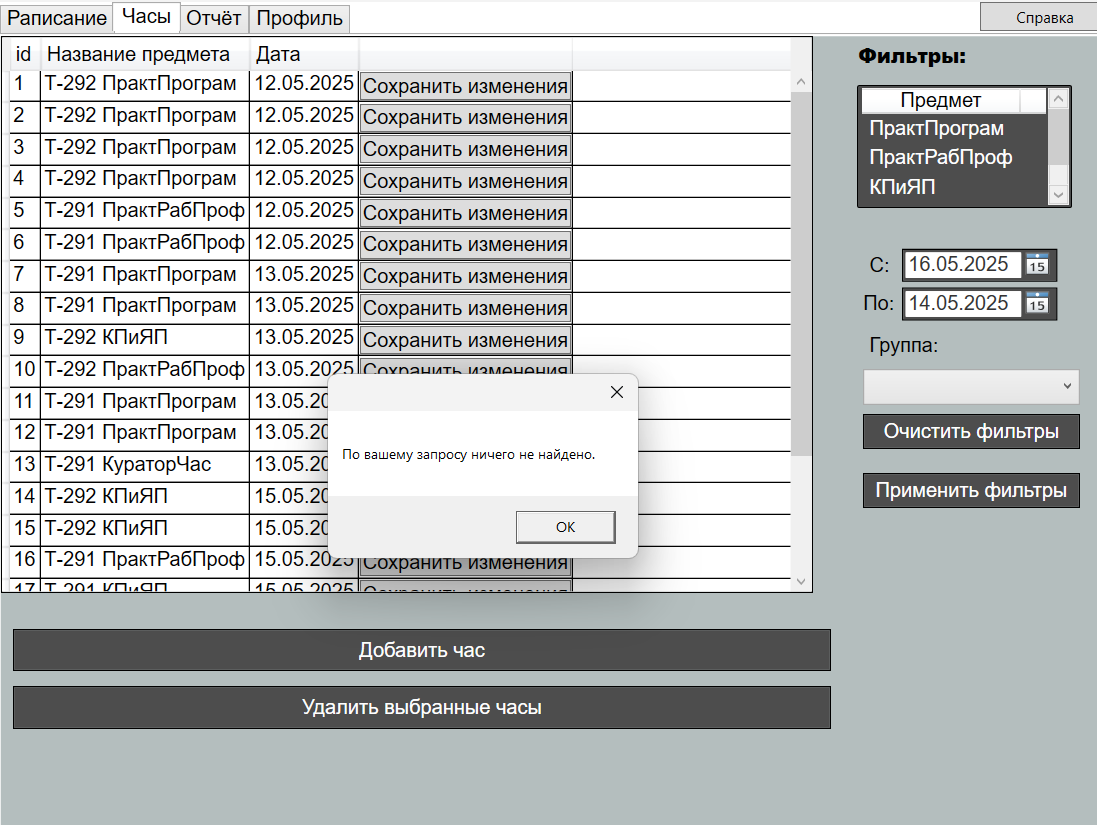


Рисунок В.7 – Проверка работы фильтров с некорректными параметрами

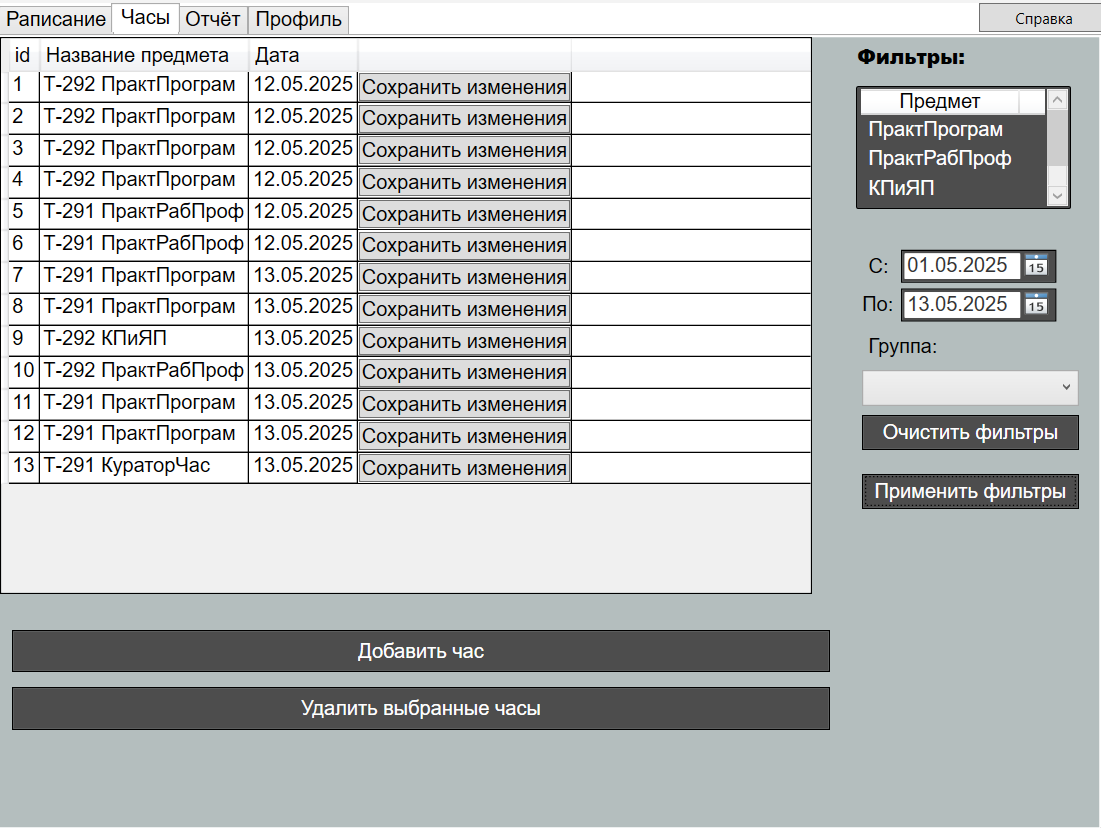


Рисунок В.8 – Проверка работы фильтров с корректными параметрами

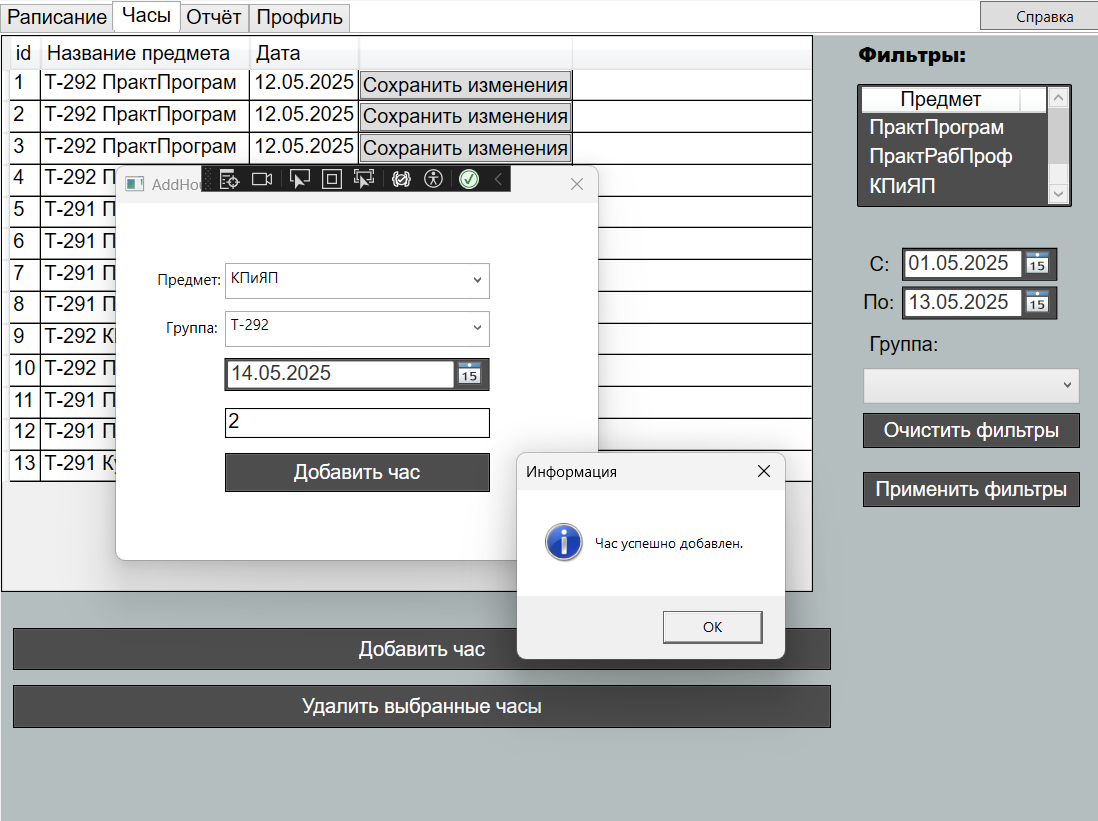


Рисунок В.9 – Проверка добавления предмета с корректными данными

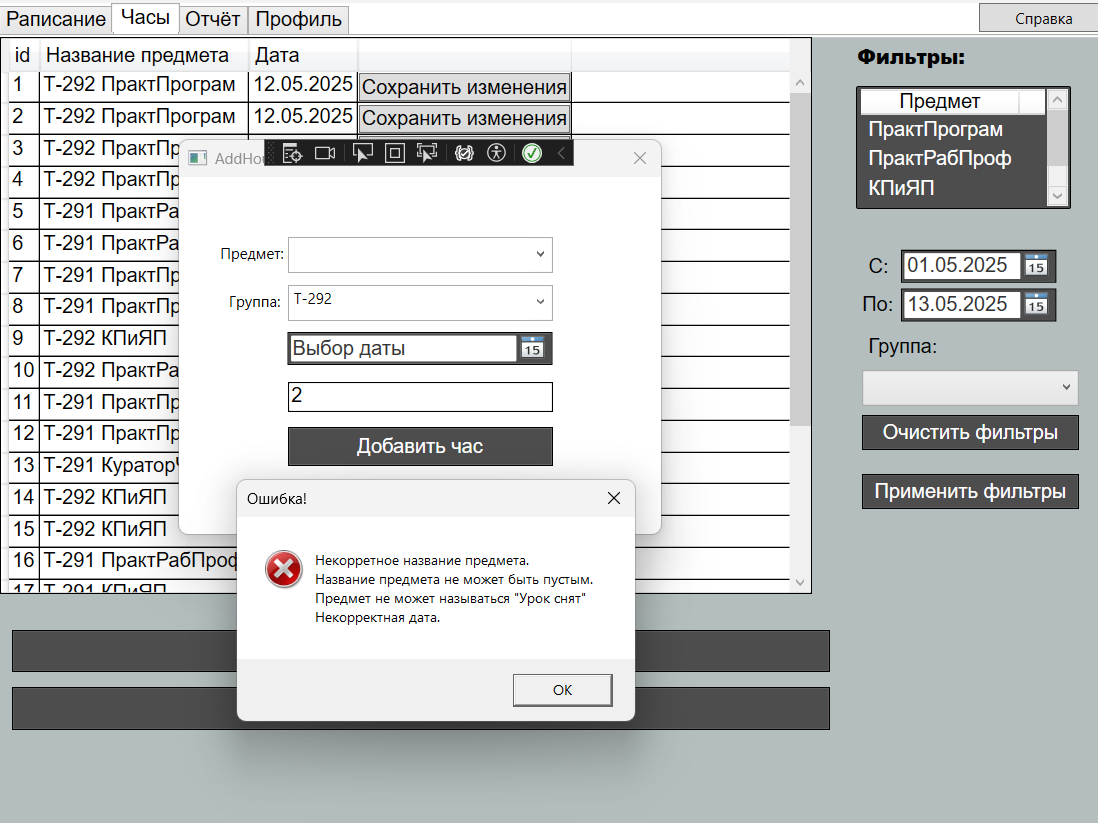


Рисунок В.10 – Проверка добавления предмета с некорректными данными

# ПРИЛОЖЕНИЕ Г

**(справочное)**

**Прототипы оконного интерфейса**

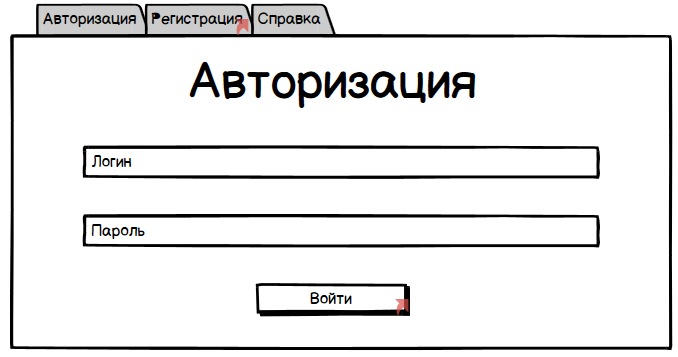


Рисунок Г.1 – Прототип страницы авторизации

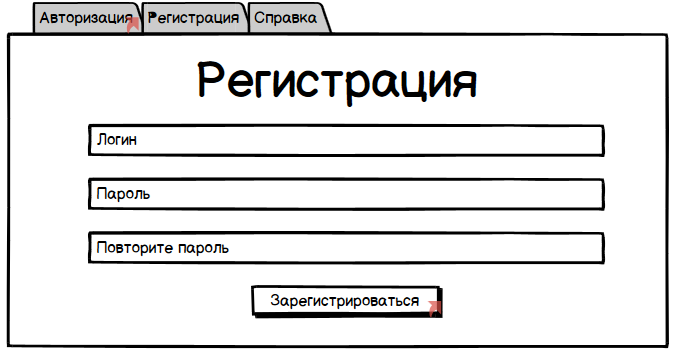


Рисунок Г.2 – Прототип страницы регистрации



Рисунок Г.3 – Прототип страницы расписания

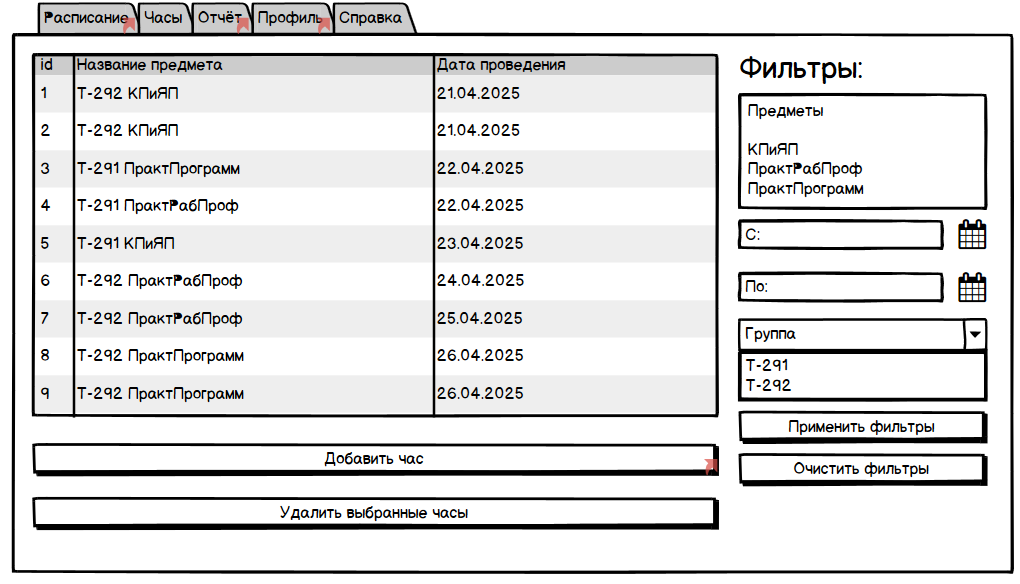


Рисунок Г.4 – Прототип страницы часов

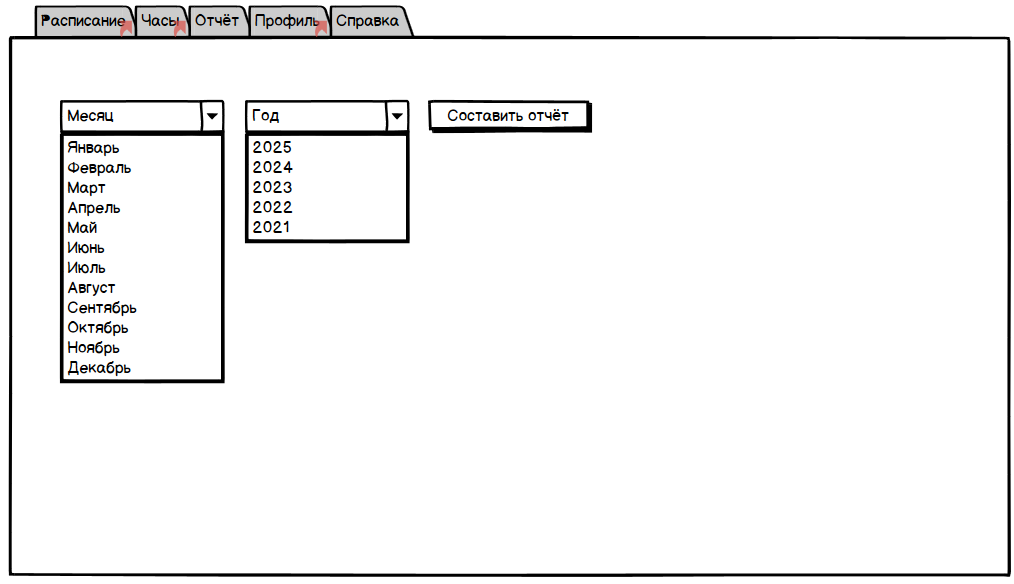


Рисунок Г.5 – Прототип страницы отчёта

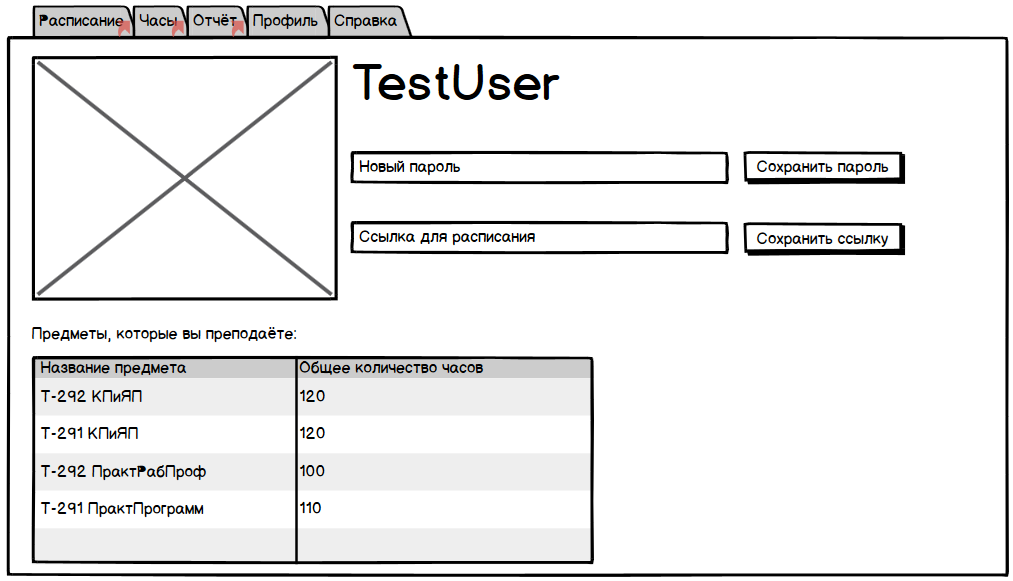


Рисунок Г.6 – Прототип страницы профиля

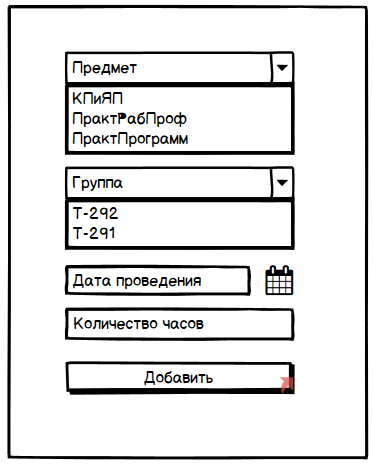
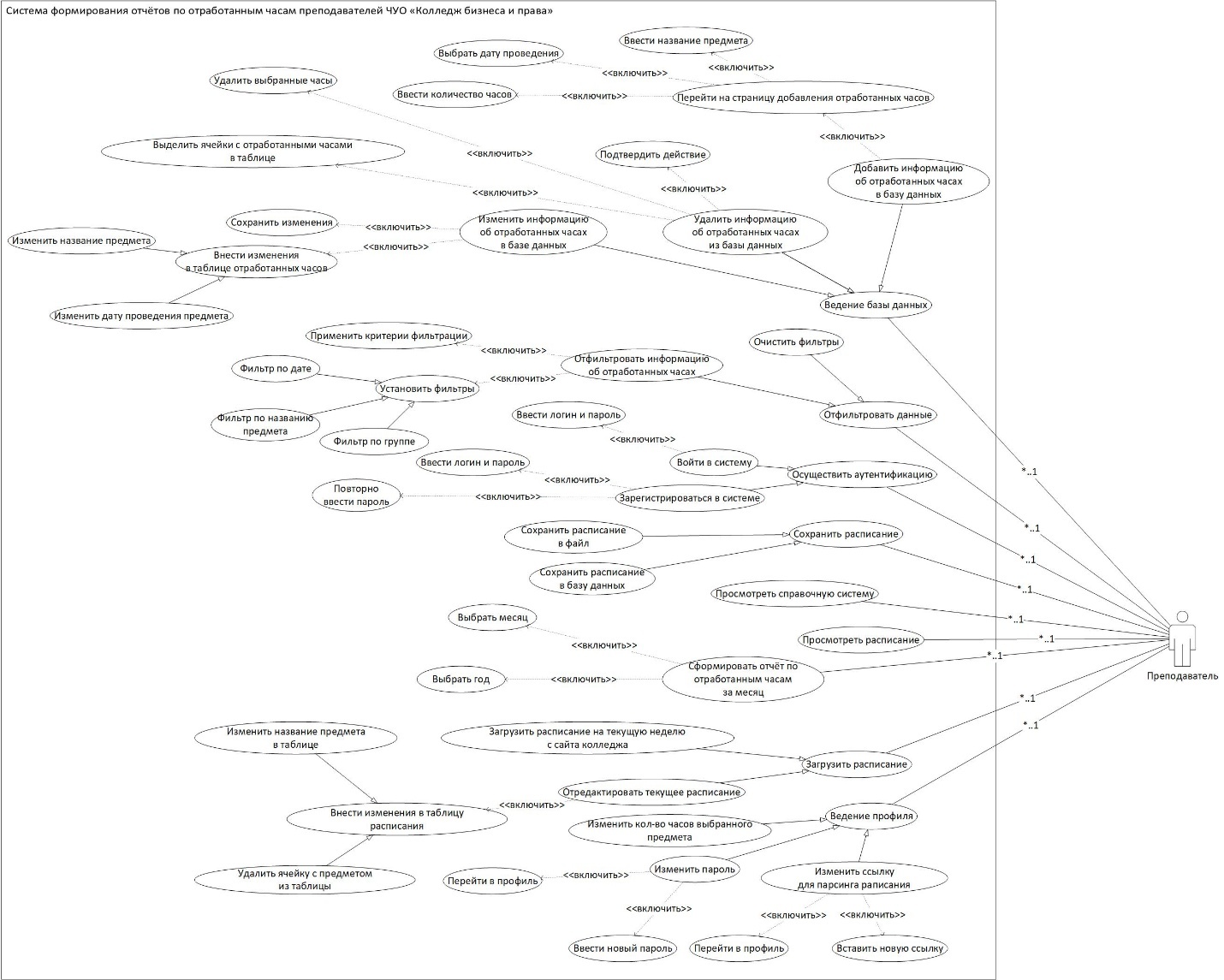


Рисунок Г.7 – Прототип страницы добавления часов



*КП Т.292001.401 ГЧ*

КБП

У

Т. Контр.

Разраб. Разраб.

Буев Д.Р.

Провер. Провер.

Коропа Е.Н.

Реценз.

Утверд.

*Коропа Е.Н.*

Н. Контр.

*РАЗРАБОТКА СИСТЕМЫ ФОРМИРОВАНИЯ ОТЧЕТОВ ПО ОТРАБОТАННЫМ ЧАСАМ ПРЕПОДАВАТЕЛЕЙ ЧУО «КОЛЛЕДЖ БИЗНЕСА И ПРАВА»*

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 1

Листов 3

Инв.№подл.

Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

*КП Т.292001.401 ГЧ*

*Диаграмма вариантов использования*

*Диаграмма вариантов использования*

Диаграмма вариантов использования

*КП Т.292001.401 ГЧ*

КБП

У

Т. Контр.

Разраб. Разраб.

Буев Д.Р.

Провер. Провер.

Коропа Е.Н.

Реценз.

Утверд.

*Коропа Е.Н.*

Н. Контр.

*РАЗРАБОТКА СИСТЕМЫ ФОРМИРОВАНИЯ ОТЧЕТОВ ПО ОТРАБОТАННЫМ ЧАСАМ ПРЕПОДАВАТЕЛЕЙ ЧУО «КОЛЛЕДЖ БИЗНЕСА И ПРАВА»*

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 2

Листов 3

Инв.№подл.

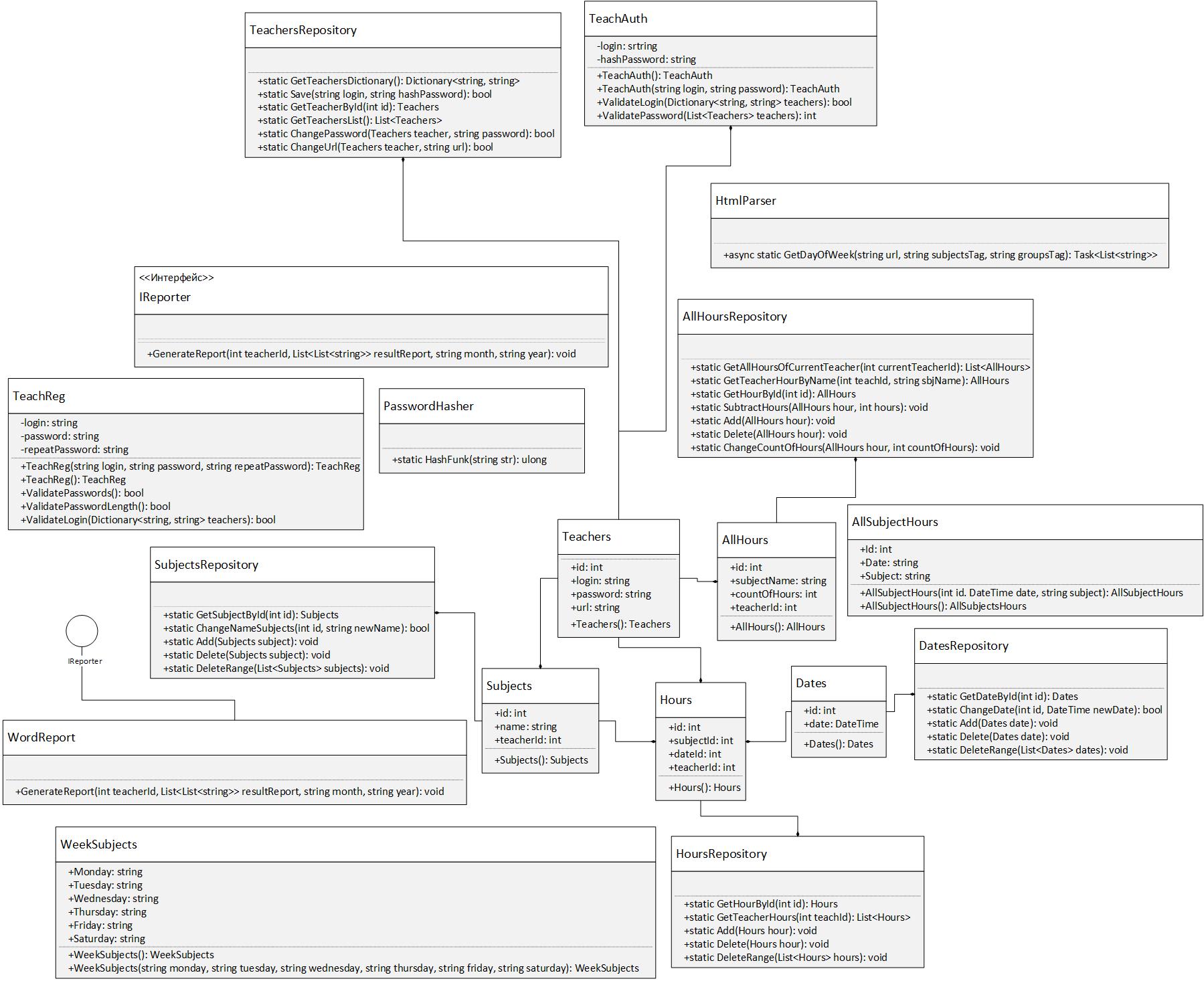
Подп. и дата

Взам.инв.№

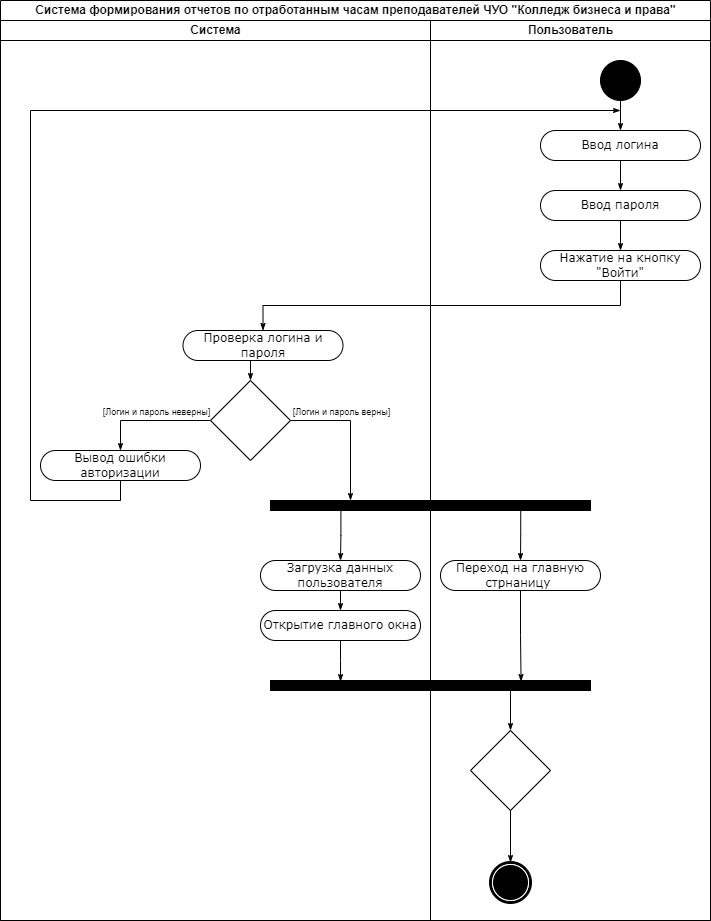
Инв.№дубл.

Подп. и дата

*КП Т.292001.401 ГЧ*



*Диаграмма классов*



*КП Т.292001.401 ГЧ*

*Диаграмма деятельности*

*(процесс авторизации преподавателя)*

КБП

У

Т. Контр.

Разраб. Разраб.

Буев Д.Р.

Провер. Провер.

Коропа Е.Н.

Реценз.

Утверд.

*Коропа Е.Н.*

Н. Контр.

*РАЗРАБОТКА СИСТЕМЫ ФОРМИРОВАНИЯ ОТЧЕТОВ ПО ОТРАБОТАННЫМ ЧАСАМ ПРЕПОДАВАТЕЛЕЙ ЧУО «КОЛЛЕДЖ БИЗНЕСА И ПРАВА»*

№ докум.

Подпись

Дата

Изм.

Лист

Масса

Лит.

Масштаб

Лист 3

Листов 3

Инв.№подл.

Подп. и дата

Взам.инв.№

Инв.№дубл.

Подп. и дата

*КП Т.292001.401 ГЧ*

**Удостоверяющий лист**

электронного документа – курсовой проект

Тема «Разработка системы формирования отчётов по отработанным часа преподавателей ЧУО «Колледж бизнеса и права».

Обозначение КП Т.292001.401

Разработчик Буев Д.Р. Руководитель Коропа Е.Н.

(Ф.И.О.) (Ф.И.О.)

Подписи лиц, ответственных за разработку электронного документа

|  |  |  |
| --- | --- | --- |
| Состав электронного документа | Разработчик | Руководитель |
| Пояснительная записка (на бумажном носителе формата А4), Пояснительная записка Буев ПЗ.docх |  |  |
| Папка с проектом «Kursach 2.0» |  |  |
| ГЧ, Курсовая Классы.docx |  |  |
| ГЧ, Курсовая Деятельности.docx |  |  |
| ГЧ, Курсовая ДВИ.docx |  |  |
| Установочный файл Setup.msi |  |  |
| Тип носителя: USB-флеш-накопитель |  |  |