**СОДЕРЖАНИЕ**

Введение 4

1 Технический проект 7

1.1 Анализ объекта исследования 7

1.2 Постановка задачи 8

1.3 Выбор средств разработки 10

1.4 Требования к программному обеспечению и техническим средствам 11

1.5 Проектирование функциональности и структуры программы 11

1.6 Разработка концептуальной модели базы данных 17

2 Рабочий проект 20

2.1 Проектирование физической модели базы данных 20

2.2 Функциональное взаимодействие модулей программы 23

2.3 Входные и выходные данные модулей 32

2.4 Установка и настройка программы 32

2.5 Работа с программой 33

Заключение 40

Список использованных источников 41

Приложение А 42

**ВВЕДЕНИЕ**

Темой моего дипломного проекта является «Разработка и создание приложения для заведующего хозяйственной частью МБДОУ №38 г. Майкопа».

Разрабатываемый программный продукт для заведующего хозяйственной частью в детском саду представляет собой решение, которое позволит автоматизировать учет и планирование работы в хозяйственной части детского сада. С помощью этого продукта заведующий сможет быстро и точно контролировать наличие и расходование материалов, планировать регулярный уход за помещениями и оборудованием, настраивать расписание работы персонала. Также продукт позволит генерировать отчеты и статистику о выполненной работе и оставшихся ресурсах, что поможет в оптимизации работы хозяйственной части и повышении эффективности ее функционирования.

Написание данной дипломной работы является целесообразным, так как программный комплекс представляет собой инновационное решение, которое является актуальным и востребованным в современных условиях детского сада. Наша разработка поможет повысить эффективность управления хозяйственной частью, сократить временные затраты на планирование и контроль, улучшить качество обслуживания и увеличить экономическую эффективность работы. Более того, разработка данного продукта будет иметь большую практическую значимость и внедрение его позволит детскому саду получить конкурентные преимущества на рынке образовательных услуг.

Объект исследования в данной работе - хозяйственная часть детского сада. Она включает в себя все помещения, оборудование, материалы и кадры, которые относятся к поддержанию и обслуживанию жизнедеятельности детского сада. В рамках исследования будет проведен анализ текущих процессов, проблем и потребностей в работе хозяйственной части, а также будет предложено инновационное решение в виде программного комплекса, который поможет автоматизировать и оптимизировать учет и планирование работы в этой сфере.

Предмет исследования в данной работе - программный комплекс для заведующего хозяйственной частью в детском саду. В рамках исследования будет проведен анализ требований пользователей, разработана архитектура и дизайн системы, описаны методы и алгоритмы ее работы, а также проведены тесты и оценка эффективности разработанного решения. Предметом исследования также будет являться внедрение программного комплекса в работу хозяйственной части детского сада и анализ его влияния на улучшение эффективности управления этой сферой, сокращение затрат времени и ресурсов, а также повышение качества обслуживания.

В рамках исследования будут рассмотрены следующие вопросы и задачи:

* анализ текущей работы хозяйственной части детского сада и выявление проблем, с которыми сталкиваются сотрудники;
* изучение потребностей пользователей (заведующего хозяйственной частью, бухгалтера и других сотрудников), их ожиданий и требований к программному комплексу;
* разработка архитектуры и дизайна программного комплекса, учитывающих требования пользователей и особенности работы хозяйственной части детского сада;
* программирование и отладка функционала системы, включая модули учета материалов, кадров, планирование регулярного ухода и обслуживания помещений и оборудования, а также отчетности;
* тестирование программного комплекса для проверки его правильной работы и выявление возможных ошибок и недостатков;
* внедрение программного комплекса в работу хозяйственной части детского сада и оценка его влияния на повышение эффективности управления, упрощение работы сотрудников, сокращение затрат времени и ресурсов;
* анализ результатов исследования и выводы по эффективности программного комплекса, его конкурентоспособности и практической значимости.

При разработке и создании приложения для заведующего хозяйственной части необходимо учесть множество факторов, связанных с потребностями заказчика, требованиям к функционалу и возможностей автоматизации рабочих процессов в учреждении. В рамках исследования будут проведены анализы многих аспектов, включая потребности и требования, сравнение существующих решений на рынке, определение технологий и инструментов для разработки, проектирование дизайна и архитектуры приложения и многие другие.

Данный программный продукт будет предназначен для заведующего хозяйственной частью в детском саду.

**1 ТЕХНИЧЕСКИЙ ПРОЕКТ**

**1.1 Анализ объекта исследования**

Дошкольное образовательное учреждение МБДОУ №38 является учреждением для получения первичного образования ребенка.

В настоящее время параллельно с развитием традиционных форм дошкольного образования апробируются новые модели. Образование в МБДОУ №38 – это высокий уровень подготовки выпускников, результативность деятельности коллектива детского сада.

Основными задачами МБДОУ являются:

* охрана жизни и укрепление физического и психического здоровья воспитанников;
* обеспечение познавательно-речевого, социально-личностного, художественно-эстетического и физического развития воспитанников;
* воспитание с учетом возрастных категорий воспитанников гражданственности, уважения к правам и свободам человека, любви к окружающей природе, Родине, семье;
* осуществление необходимой коррекции недостатков в физическом и (или) психическом развитии воспитанников;
* взаимодействие с семьями воспитанников для обеспечения полноценного развития детей;
* оказание консультативной и методической помощи родителям (законным представителям) по вопросам воспитания, обучения и развития детей.

Преимуществами МБДОУ №38 являются:

* уникальное сочетание классических традиций Российского образования и лучших образцов мировой образовательной практики с инновационными современными подходами;
* современное оснащение образовательного процесса, отлаженная до совершенства система жизнеобеспечения;
* индивидуализация учебно-воспитательного процесса;
* система комплексного сопровождения индивидуального развития ребенка;
* оптимально выстроенная система дополнительного образования.

Вся деятельность коллектива направлена на сохранение и развитие личности ребенка (развитие компетентности, самостоятельности, инициативности, любознательности, способности к творческому самовыражению) и приобщению к общечеловеческим ценностям.

В коллективе нет случайных людей. Сотрудники – люди творческие, без которых невозможен успех процесса воспитания и развития ребенка, профессионалы, которые выбирают свой метод работы, исходя из индивидуальных особенностей детей и уровня их развития.

Наверное, трудно поверить, что в маленьком ауле может быть такой чудесный мир детства, когда не только природные катаклизмы нарушают привычный ритм жизни, но и в образовании существует множество нерешенных проблем. Счастливый «островок детства» готов поделиться опытом своей работы и сотрудничать с другими детскими садами.

В детском саду МБДОУ №38 300 детей в 10 группах.

**1.2** **Постановка задачи**

Необходимо разработать программу для МБДОУ №38, которая обеспечивает хранение, добавление и предоставление всей необходимой информации о сотрудниках, помещениях, материалах, поставщиках, расписании рабочего дня сотрудников, а также создать документационное сопровождение программы, включающее в себя технический и рабочие проекты.

Программа должна обеспечить выполнение следующих функций:

* авторизация;
* переход между страницами;
* хранение информации в БД;
* возможность добавления новой информации;
* возможность быстрого поиска и сортировки данных;
* вывод информации и изображений в текстовые и графические поля.

Программный продукт должен включать в себя две части:

* пользовательскую часть, которая должна содержать все необходимые сведения о хозяйственной части детского сада;
* административную часть, которая должна обеспечивать хранение и добавление новых данных о поставщиках, сотрудниках и их расписании, материалах и помещениях.

Создаваемое приложение не планируется использовать в удаленном режиме работы, поэтому достаточно разработать локальную систему.

Входными данными в приложении будут:

* логин и пароль заведующего хозяйственной частью;
* информация о сотрудниках и их расписании, материалах, помещениях и поставщиках.

Выходными данными в приложении будут:

* изменённая база данных;
* сообщения об ошибках.

Приложение будет состоять из страниц, на которых будет располагаться вся необходимая пользователю информация.

Способ решения поставленных задач зависит от выбора средств разработки.

**1.3 Выбор средств разработки**

Для разработки приложения для заведующего хозяйственной частью МБДОУ №38 города Майкоп можно использовать различные инструменты разработки, такие как Visual Studio, Eclipse, NetBeans и другие. Однако, для данного проекта наиболее подходящим средством разработки является Visual Studio на WPF C#. Вот несколько причин, почему:

* кроссплатформенность - WPF платформа, поддерживающая работу на различных операционных системах, что позволяет создать приложение, работающее на операционных системах Windows, MacOS и Linux;
* удобная визуальная разработка - WPF является одним из наиболее удобных инструментов для визуальной разработки, обладая богатыми возможностями для создания интерфейсов, поддерживая например такие элементы, как DataGrid, ListBox, StackPanel и многие другие;
* обширная библиотека - WPF C# имеет обширную библиотеку, поэтому можно создавать разные элементы интерфейса, что позволяет сократить время разработки;
* надежность и безопасность - В Visual Studio реализован механизм создания безопасных приложений, который защищает ваш код от небезопасных обращений, вместе с этим WPF обеспечивает надежность приложения в целом;
* мощный язык программирования - C# один из самых усовершенствованных языков программирования, имеет сильную типизацию и находится на первом месте по ряду показателей по сравнению с другими языками, такими как Java;
* многопоточность - Разработчик может создать приложение, которое работает с несколькими потоками одновременно, это повышает быстродействие приложения.

Кроме того, Visual Studio и WPF C# являются наиболее популярным инструментом в области разработки Windows-приложений, и имеет большое сообщество разработчиков, которые могут помочь решить возникающие проблемы.

**1.4 Требования к программному обеспечению и техническим средствам**

Для корректной и комфортной разработки приложения для заведующего хозяйственной частью МБДОУ №38 необходимо, чтобы компьютер удовлетворял следующим минимальным системным требованиям:

* процессор 2.5 ГГц и выше;
* 2 Гб ОЗУ;
* свободное пространство на диске: 1 Гб и больше;
* дисплей с разрешением не менее 1366x768 пикселей;
* клавиатура и мышь;
* Microsoft Windows.

Для разработки приложения на компьютере должно быть установлено следующее программное обеспечение:

* Visual Studio 2022;
* ASP.NET Core, Entity Framework;
* Microsoft Windows 10 и выше;
* Microsoft SQL Server, SQL Management Studio.

**1.5 Проектирование функциональности и структуры программы**

Для описания основных функций проектируемой программы воспользуемся методологией SADT с использованием диаграмм IDEF0.

Сначала строим самую общую, контекстную диаграмму (рисунок 1). На ней описываем процесс одним функциональным блоком, определяем входные и выходные данные, управляющие воздействия и механизмы.

Входные данные:

* логин (I1);
* пароль (I2);
* исходная БД (I3);

Выходные данные:

* сообщение об ошибке (O1);
* изменённая БД (O2).

Управление: правила работы с программой (C1).

Механизмом будет являться создаваемое приложение (M1).

Субъект: автоматизированное рабочее место для заведующего хозяйственной частью.



Рисунок 1 – Контекстная IDEF0-диаграмма

Точка зрения: функции системы рассматриваются с точки зрения корректного занесения информации в БД и отображения текстовых и графических данных, а также правильного составления отчета.

Цель: рассмотрение функций системы, выполнение которых позволит корректно вводить информацию и редактировать её в БД, а также осуществлять выборку нужных данных и формировать отчет.

Далее производим декомпозицию функционального блока (рисунок 2), выделяя подфункции «Выполнить аутентификацию в системе» – блок A100, и «Выполнить администрирование системы» – блок A200

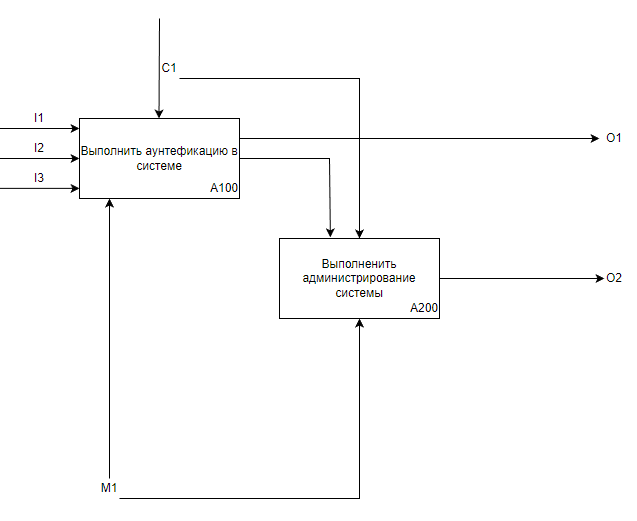


Рисунок 2 – IDEF0-диаграмма основных процессов программы

На вход функции «Выполнить аутентификацию в системе» будет подаваться информация о логине (I1) и его пароль (I2) из БД (I3). После этого будет происходить проверка введённых данных в БД, после чего появится либо вывод об ошибке (O1), либо произведен допуск к следующему блоку.

Для блока «Выполнить администрирование системы» входными данными будут сведения о детях, воспитателях и родителях (I3), хранящиеся в БД (I4). Выходными данными будет изменённая БД (O2).

Обеспечивать выполнение функций «Выполнить аутентификацию в системе» и «Выполнить администрирование системы» будет разрабатываемое приложение.

Теперь на отдельных диаграммах произведем декомпозицию функциональных блоков «Выполнить аутентификацию в системе» и «Выполнить администрирование системы».

На диаграмме A100 (рисунок 3) представлены основные процессы, позволяющие корректно выполнить аутентификацию в системе.

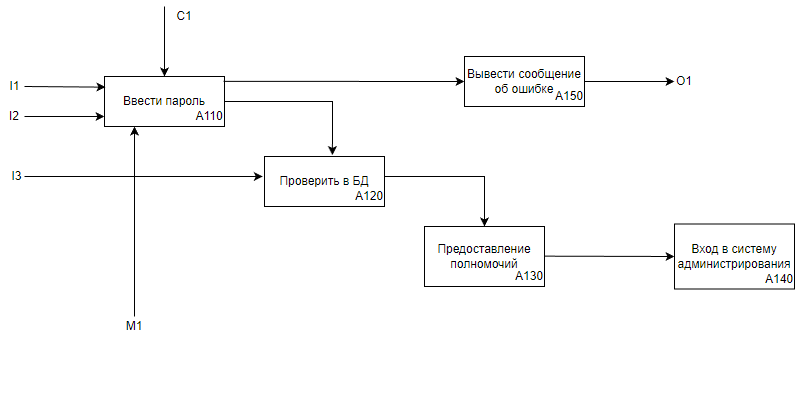


Рисунок 3 – IDEF0-диаграмма основных процессов аутентификации

На диаграмме A200 (рисунок 4) представлены основные процессы, администрирования системы.

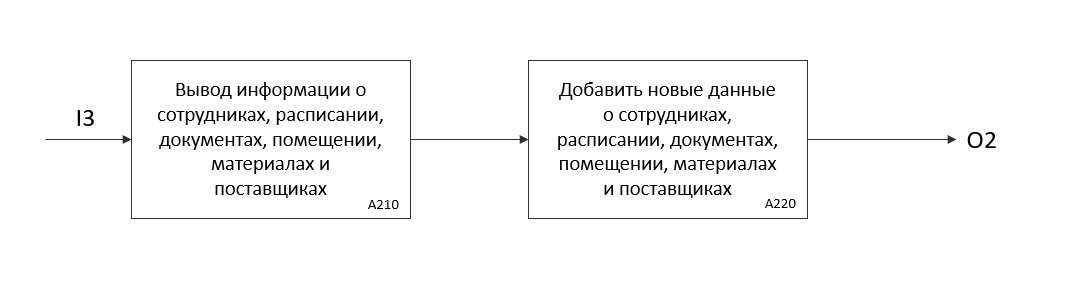


Рисунок 4 – IDEF0-диаграмма основных процессов администрирования системы

На диаграмме A210 (рисунок 5) представлены основные функции, осуществляемые проектируемым приложением, которые позволят корректно провести редактирование информации:

* «Отобразить блоки для ввода информации» (A211);
* «Проверка правильности ввода данных» (A212);
* «Сохранение изменений в БД» (A213).

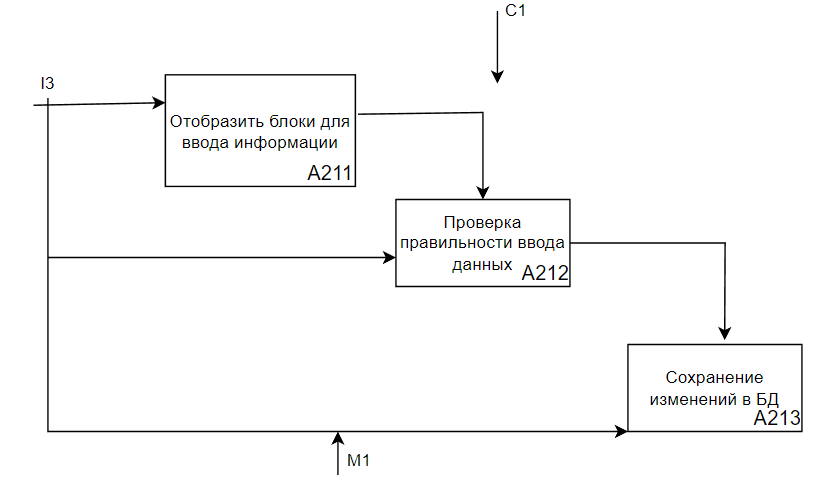


Рисунок 5 – IDEF0-диаграмма основных процессов добавления/редактирования информации

Исходная БД (I3) будет входными данными для первого блока А210. Заполненные поля будут выходными данными для блока А212 и входными для блока А213.

Выходными потоками для блока А213 будет изменённая БД (О2).

Результатом выполнения блока «Выполнить администрирование системы» будет отображение новой информации и сохранения её в БД.

С целью удостоверения предъявляемых к программе требований, приведенных в разделе 1.2, разрабатываемое приложение будет состоять из следующих окон:

* загрузка – будет запускаться перед работой с приложением;
* авторизация – будет содержать блоки для ввода информации и кнопку входа в приложение;
* окно справки – окно со справочной информацией о программе;
* главное окно – окно для просмотра информации о сотрудниках, расписании, документов, поставщиках, материалах и помещениях;
* окно добавления нового расписания – окно с добавлением нового расписания для сотрудника;
* окно добавления нового сотрудника – окно с добавлением нового сотрудника;
* окно добавления нового документа – окно с добавлением нового документа для сотрудника;
* окно добавления нового поставщика – окно с добавлением новых поставщиков материалов;
* окно добавления новых помещений – окно с добавлением новых помещений;
* окно с добавлением новых материалов – окно с добавлением новых материалов;
* окно редактирования информации о расписании – окно, в котором можно изменить данные расписания;
* окно редактирования информации о сотруднике – окно, в котором можно изменить данные о сотруднике;
* окно редактирования информации документа – окно, в котором можно изменить данные документа;
* окно редактирования информации о поставщиках – окно, в котором можно изменить данные о поставщиках;
* окно редактирования информации о помещениях – окно, в котором можно изменить данные о помещениях;
* окно редактирования информации о материалах – окно, в котором можно изменить данные о материалах.

Переход к блоку «Добавление» будет происходить при выборе пункта «Добавить» в главном окне.

Материалы будут храниться в БД. Проектируемая структура приложения представлена на рисунке 6.

Тестирование программы планируется проводить путём выбора каждого пункта меню и проверки корректности получаемого отклика программы.

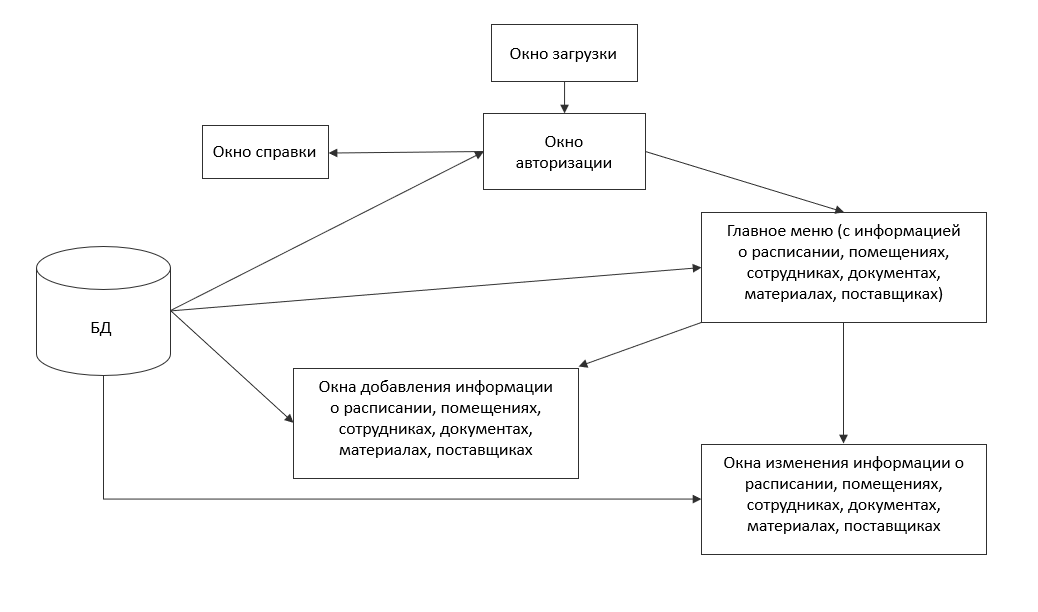


Рисунок 6 – Схема проектируемой структуры приложения

**1.6 Разработка концептуальной модели базы данных**

Для хранения информации будет разработана БД. Она будет состоять из таблиц: «Пользователи», «Роли», «Группы», «Дети» и «Воспитатели и родители», которые включают в себя следующие поля:

Пользователи:

* Id – служебное поле для обеспечения уникальности записей;
* Login – логин администратора;
* Password – пароль администратора;
* FullName – Ф.И.О администратора.

Должность:

* PostId – служебное поле для обеспечения уникальности записей;
* PostName – наименование должности.

Документы сотрудника:

* DocumentId – служебное поле для обеспечения уникальности записей;
* EmployeeId – обозначение, кому принадлежит этот документ, внешний ключ к таблице «Сотрудники»;
* DocumentName – наименование документа;
* Documents – путь хранения документа.

Расписание рабочего дня сотрудников:

* ScheduleId – служебное поле для обеспечения уникальности записей;
* IdEmployee – сотрудник, внешний ключ к таблице «Сотрудники»;
* Date – дата рабочего дня;
* TimeStart – начало рабочего дня;
* TimeEnd – конец рабочего дня.

Сотрудник:

* EmployeeId – служебное поле для обеспечения уникальности записей;
* NameEmployee – имя сотрудника;
* SurnameEmployee – фамилия сотрудника;
* IdPost – должность сотрудника, внешний ключ к таблице «Должности»;
* SerialPassportEmployee – серия паспорта сотрудника;
* NumberPassportEmployee – номер паспорта сотрудника;
* Contact – номер телефона сотрудника;
* Address – адрес, место жительства сотрудника.

Поставщик:

* SupplierId – служебное поле для обеспечения уникальности записей;
* NameSupplier – наименование организации поставщика;
* Contact – контактные данные, номер телефона поставщика;
* Comments – описание компании поставщика.

Помещения:

* FacilitiesId – служебное поле для обеспечения уникальности записей;
* NameFacilities – название помещения.

Материалы:

* EquipmentId – служебное поле для обеспечения уникальности записей;
* NameEquipment – наименование материала;
* Count – количество материала на складе;
* PurchaseDate – дата поставки;
* Cost – стоимость материала;
* IdFacilities – помещение, в котором хранится материал;
* IdSuppliers – поставщик, что поставил материал.

Концептуальная модель БД представлена на рисунке 7.

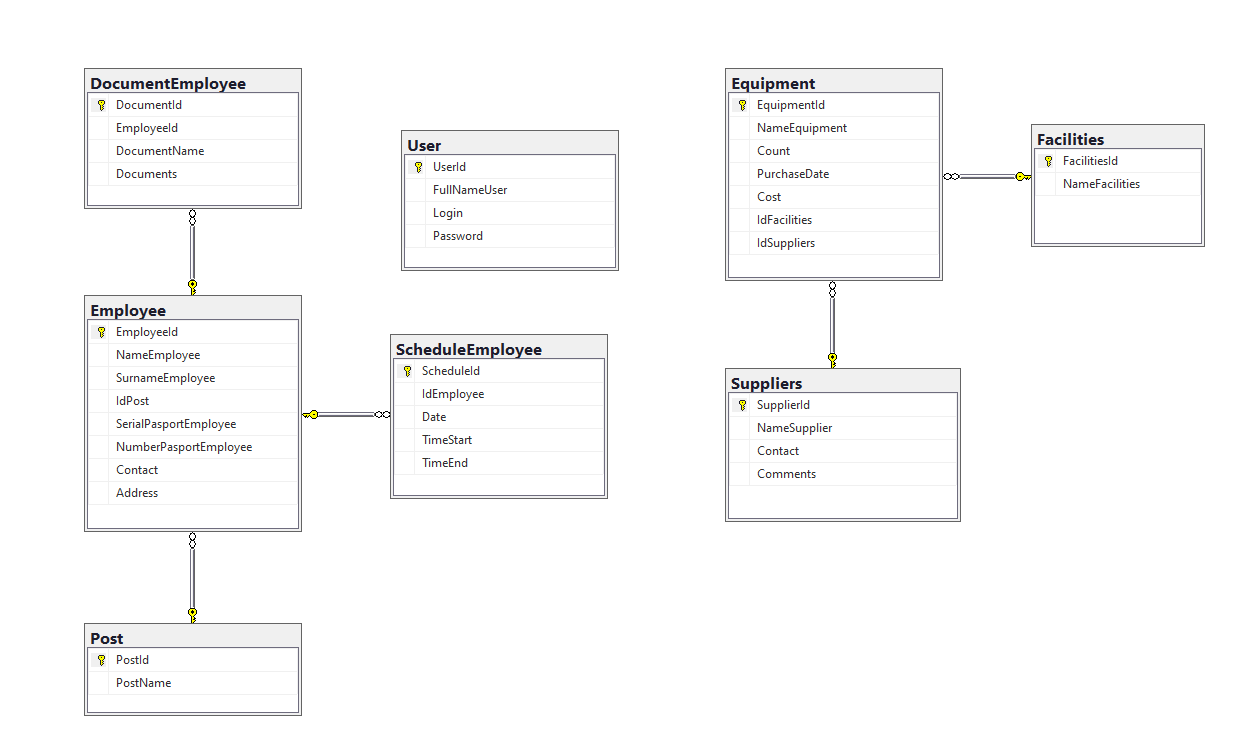


Рисунок 7 – Концептуальная модель БД

**2 РАБОЧИЙ ПРОЕКТ**

**2.1 Проектирование физической модели базы данных**

Физическая модель данных описывает то, как данные хранятся в компьютере, представляя информацию о структуре записей, их упорядоченности и существующих путях доступа. Также описываются типы, идентификаторы и разрядность полей.

На основе логической модели были создана физическая модель БД, которая хранится в компьютере в файле KinderGartenHoz.mdf (таблица 1).

Таблица 1 – Физическая модель таблицы «Пользователи»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| UserId | Уникальный номер | int | Первичный |
| Login | Логин | nvarchar(255) | - |
| Password | Пароль | char(55) | - |
| FullNameUser | Ф.И.О пользователя | nvarchar(55) | - |

Таблица 2 – Физическая модель таблицы «Помещения»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| FacilitiesId | Уникальный номер | int | Первичный |
| FacilitiesName | Наименования помещения | nvarchar(255) | - |

Таблица 3 – Физическая модель таблицы «Поставщики»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| SuppliersId | Уникальный номер | int | Первичный |
| SuppliersName | Наименования организации | nvarchar(255) | - |
| Contact | Номер телефона поставщика | nvarchar(11) | - |
| Comments | Описание организации | nvarchar(max) | - |

Таблица 4 – Физическая модель таблицы «Материалы»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| EquupmentId | Уникальный номер | int | Первичный |
| NameEquipment | Наименование материала | nvarchar(255) | - |
| Count | Количество на складе | int | - |
| PurschaseDate | Дата поставки | date | - |
| Cost | Стоимость материала | decimal(10,2) | - |
| IdFacilities | Место хранения | int | Внешний |
| IdSuppliers | Поставщик | int | Внешний |

Таблица 5 – Физическая модель таблицы «Должности»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| PostId | Уникальный номер | int | Первичный |
| PostName | Наименование должности | nvarchar(255) | - |

Таблица 6 – Физическая модель таблицы «Сотрудники»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| EmployeeId | Уникальный номер | int | Первичный |
| EmployeeName | Имя сотрудника | nvarchar(255) | - |
| EmployeeSurname | Фамилия сотрудника | nvarchar(255) | - |
| IdPost | Должность сотрудника | int | Внешний |
| SerialPasportEmployee | Серия паспорта | int | - |
| NumberPasportEmployee | Номер паспорт | int | - |
| Contact | Номер телефона | char(11) | - |
| Address | Адрес сотрудника | nvarchar(255) | - |

Таблица 7 – Физическая модель таблицы «Документы сотрудника»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| DocumentId | Уникальный номер | int | Первичный |
| EmployeeId | Сотрудник | int | Внешний |
| DocumentName | Наименование документа | nvarchar(255) | - |
| Documents | Путь к хранилищу документа | nvarchar(max) | - |

Таблица 8 – Физическая модель таблицы «Расписание рабочего дня сотрудника»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Подпись поля | Тип данных | Ключ |
| ScheduleId | Уникальный номер | int | Первичный |
| IdEmployee | Сотрудник | int | Внешний |
| Date | Дата рабочего дня | date | - |
| TimeStart | Время начала рабочего дня | time | - |
| TimeEnd | Время окончания рабочего дня | time | - |

**2.2 Функциональное взаимодействие модулей программы**

На основе разработанного технического проекта было написано приложение по теме «Разработка и создание приложения для заведующего хозяйственной частью «МБДОУ №38» г. Майкопа». Для создания программы был использован язык программирования C# и ИСР Visual studio 2022. В программе используются следующие языковые модули:

* System − содержит фундаментальные и базовые классы, определяющие часто используемые типы значений и ссылочных данных, события и обработчики событий, интерфейсы, атрибуты и исключения обработки;
* System.Windows − содержит типы, используемые в приложениях.
* System.IO − содержит типы, поддерживающие ввод и вывод, включая возможности чтения и записи данных в потоках как синхронно, так и асинхронно, реализации пользовательского ведения журнала и обработки входящих и исходящих потоков данных в последовательных портах;
* System.Data − содержит типы для доступа к данным из различных источников и для управления этими данными. Пространство имен верхнего уровня и несколько дочерних пространств имен образуют архитектуру ADO.NET и поставщиков данных ADO.NET;
* System.Threading − содержат типы, обеспечивающие возможности многопотокового программирования. Дочернее пространство имен предоставляет типы, которые упрощают задачу написания параллельного и асинхронного кода;
* System.Collections.Generic – работа с обобщёнными коллекциями (List<string>) [1].

Также были разработаны следующие пользовательские модули, взаимодействие которых представлено на рисунке 8:

* EntryWindow.xaml.cs – окно авторизации;
* MainWindow.xaml.cs – главное окно приложения;
* ScheduleEmployeePage.xaml.cs – страница с информацией о расписании;
* EmployeePage.xaml.cs – страница с информацией о сотрудниках;
* EquipmentPage.xaml.cs – страница с информацией о материалах;
* DocumentWindow.xaml.cs – окно с информацией о документах сотрудника;
* FacilitiesPage.xaml.cs – страница с информацией о помещениях;
* SuppliersPage.xaml.cs – страница с информацией о поставщиках;
* AddScheduleEmployeeWindow.xaml.cs – окно с добавлением нового расписания;
* AddEmployeeWindow.xaml.cs – окно с добавлением нового сотрудника;
* AddEquipmentWindow.xaml.cs – окно с добавлением новых материалов;
* AddDocumentEmployeeWindow.xaml.cs – окно с добавлением нового документа сотрудника;
* AddFacilities.xaml.cs – окно с добавлением нового помещения;
* AddSuppliers.xaml.cs – окно с добавлением нового поставщика;
* EditScheduleEmployeeWindow.xaml.cs – окно с изменением расписания сотрудников;
* EditEmployeeWindow.xaml.cs – окно с изменением данных о сотруднике;
* EditEquipmentWindow.xaml.cs – окно с изменением данных о материалах;
* EditDocumentEmployeeWindow.xaml.cs – окно с изменением данных о документах сотрудника;
* EditFacilitiesWindow.xaml.cs – окно с изменением данных о помещениях;
* EditSuppliersWindow.xaml.cs – окно с изменением данных о поставщиках;
* CurrentUser.cs – класс, хранилище и передачи информации об имени и фамилии пользователя;
* HouseHoldPartAccess.cs – класс, позволяющий хранить, изменять, добавлять и удалять информацию из базы данных;
* TimeSpanToStringConverter.cs – класс, который конвертирует тип «TimeSpan» в тип «String»;
* BaseViewModel.cs – расширенная реализация, для работы со свойствами;
* RelayCommand.cs – класс, позволяющий взаимодействовать с элементами в окнах/страницах из других классов.

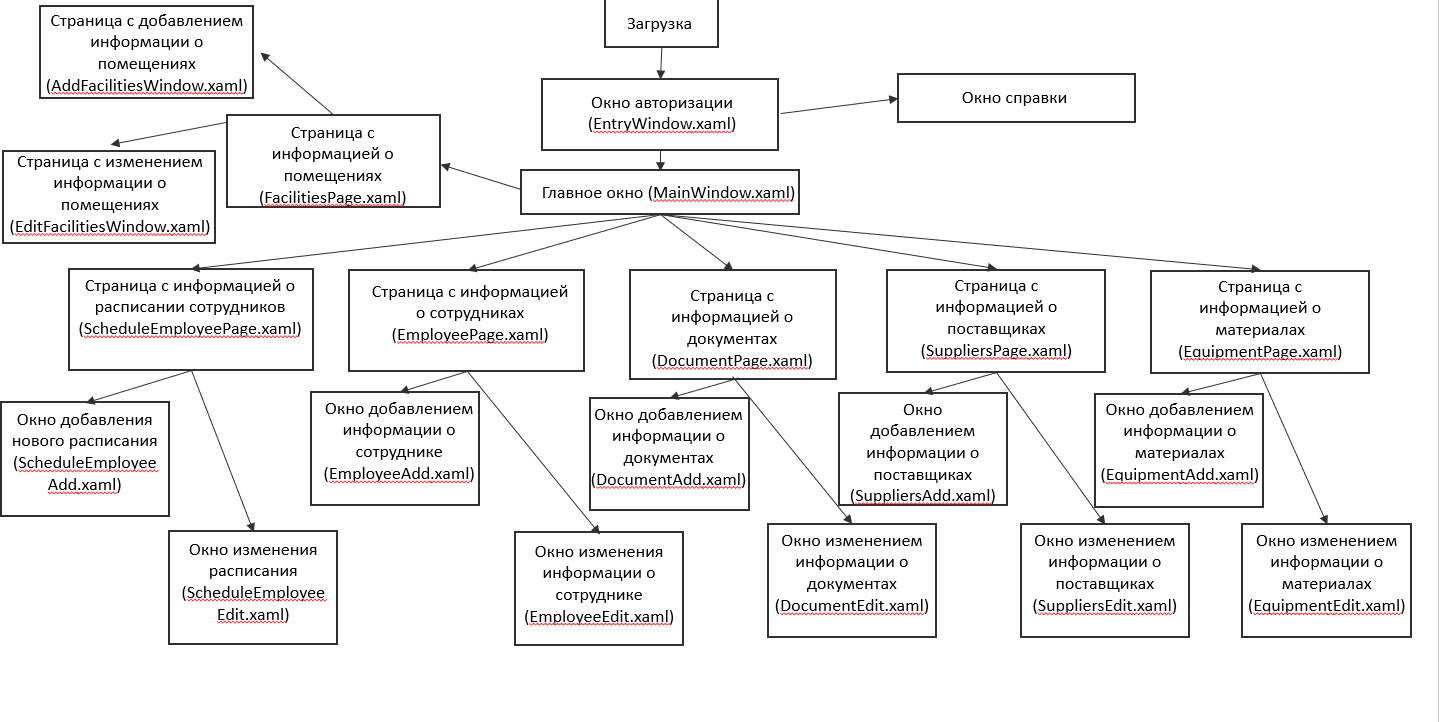


Рисунок 8 – Схема взаимодействия основных модулей программы

В модуле EntryWindow.xaml.cs описаны и используются следующие процедуры:

* private void ExiteBtn\_Click(object sender, RoutedEventArgs e) – выход из программы;
* private void PasswordBox\_PasswordChanged(object sender, RoutedEventArgs e) – передача информации из TextBox в PasswordBox;
* private void CheckBox\_Checked(object sender, RoutedEventArgs e) – Передача информации из PasswordBox в TextBox. Смена видимости у PasswordBox в режим невидимки и TextBox в режим видимости;
* private void CheckBox\_Unchecked(object sender, RoutedEventArgs e) – смена видимости у TextBox в режим невидимки и PasswordBox в режим видимости;
* private void Window\_Loaded(object sender, RoutedEventArgs e) – заполнение полей логина и пароля, если пользователь сохранял пароль;
* private void LoginButton\_Click(object sender, RoutedEventArgs e) – проверка данных на правильность ввода логина и пароля, переход в главное окно.

В модуле MainWindow.xaml.cs описаны и используются следующие процедуры:

* private void ExiteBtn\_Click(object sender, RoutedEventArgs e) – выход из программы;
* private void EmployeeBtn\_Click(object sender, RoutedEventArgs e) – переход на страницу с информацией о сотрудниках;
* private void ScheduleEmployeeBtn\_Click(object sender, RoutedEventArgs e) – переход на страницу с расписанием;
* private void EquipmentBtn\_Click(object sender, RoutedEventArgs e) – переход на страницу с информацией об инвентаре;
* private void FacilitiesBtn\_Click(object sender, RoutedEventArgs e) – переход на страницу с информацией о помещениях;
* private void SuppliersBtn\_Click(object sender, RoutedEventArgs e) – переход на страницу с информацией о поставщиках;
* private void DocumentsBtn\_Click(object sender, RoutedEventArgs e) – переход на страницу с документами сотрудников.

В модулях ScheduleEmployeePage.xaml.cs, EquipmentPage.xaml.cs описаны и используются следующие общие процедуры:

* private void SearchDate\_SelectedDateChanged(object sender, SelectionChangedEventArgs e) – процедура по выбранной пользователем дате. Если пользователь выбирает дату, то происходит фильтрация записей по дате, а если пользователь не выбирает дату, то отображаются все занятия. При этом используется ICollectionView для представления коллекции данных и ScheduleEmployee для работы с объектами расписания занятий.фильтрует данные в таблице;
* private void SearchText\_TextChanged(object sender, TextChangedEventArgs e) – процедура осуществляет поиск по разным критериям. Вначале получается текст поиска в нижнем регистре. Затем используя LINQ-запрос, происходит поиск при этом фрагменты поискового текста разбиваются на отдельные слова, которые должны содержаться в одном из полей (например: имя или фамилия);
* private void CopyMenuItem\_Click(object sender, RoutedEventArgs e) – процедура вызывается при нажатии на пункт меню "Копировать", который отображается при нажатии правой кнопкой мыши на строке таблицы. Происходит получение выбранной строки таблицы;
* private void DeleteMenuItem\_Click(object sender, RoutedEventArgs e) – процедура вызывается при нажатии на пункт меню «Удалить», который отображается при нажатии правой кнопкой мыши на строке таблице. Она удаляет полностью всю строку из базы данных;
* private void EditMenuItem\_Click(object sender, RoutedEventArgs e) – процедура открывает окно редактирования выбранной записи в таблице;
* public static class VisualTreeHelpers – класс содержит статический метод GetParentOfType(), который помогает получить родительский элемент нужного типа T из визуального дерева элементов WPF. Метод GetParentOfType() принимает в качестве параметра элемент типа DependencyObject, для которого необходимо получить родительский элемент нужного типа T. Этот метод используется для получения родительского элемента нужного типа в событиях и привязках данных в WPF;
* private void ScheduleDG\_RowPreviewMouseRightButtonDown(object sender, MouseButtonEventArgs e) – процедура обрабатывает событие клика правой кнопкой мыши на строке таблицы ;
* private void AddScheduleButton\_Click(object sender, RoutedEventArgs e) – процедура открывает окно с добавлением нового расписания.

В модулях ScheduleEmployeeAdd.xaml.cs, ScheduleEmployeeEdit.xaml.cs описаны и используются следующие процедуры:

* private void LoadEmployee() – процедура загружает список сотрудников из базы данных и отображает его в элементе управления "Имя сотрудника" (EmployeeName). Используется объект dbAccess, который предоставляет доступ к данным из базы данных;
* private void SaveButton\_Click(object sender, RoutedEventArgs e) – процедура вызывается при нажатии на кнопку "Сохранить". Сначала создаются объекты db и dbAccess для доступа к базе данных. Затем происходит проверка, является ли свойство DataContext типом ScheduleEmployee. Если это так, то происходит проверка заполнения всех полей в объекте schedule (EmployeeName, Date, TimeStart, TimeEnd). Если какое-то из полей не заполнено, то выводится сообщение об ошибке.

В модулях EmployeeAddWindow.xaml.cs и EmployeeEditWindow.xaml.cs описаны и используются следующие методы:

* private void LoadPost() – процедура загружает список должностей из базы данных и отображает его в элементе управления «Должность» (PostName). Используется объект dbAccess, который предоставляет доступ к данным из базы данных;
* private void SaveButton\_Click(object sender, RoutedEventArgs e) – процедура вызывается при нажатии на кнопку «Сохранить». Сначала создаются объекты db и dbAccess для доступа к базе данных. Затем происходит проверка, является ли свойство DataContext типом Employee. Если это так, то происходит проверка заполнения всех полей в объекте employee (EmployeeName, EmployeeSurname, PostName, SerialPasportEmployee, NumberPasportEmployee, Contact, Address). Если какое-то из полей не заполнено, то выводится сообщение об ошибке.

В модулях FacilitiesAddWindow.xaml.cs, FacilitiesEditWindow.xaml.cs, SuppliersAddWindow.xaml.cs, SuppliersEditWindow.xaml.cs описаны и используются процедура private void SaveButton\_Click(object sender, RoutedEventArgs e) – процедура вызывается при нажатии на кнопку «Сохранить». Сохраняет изменённые и добавленные новые данные о поставщиках и помещениях.

В модулях DocumentEmployeeAddWindow.xaml.cs, DocumentEmployeeEditWindow.xaml.cs описаны и используются следующие процедуры:

* private LoadEmployee() – процедура загружает список сотрудников из базы данных и отображает его в элементе управления "Имя сотрудника" (EmployeeName). Используется объект dbAccess, который предоставляет доступ к данным из базы данных;
* private void AddDocumentButton\_Click(object sender, RoutedEventArgs e) – процедура добавляет новый документ для сотрудников. Сначала проводится проверка документа на расширение и вес, после чего добавляется в базу данных путь к файлу, а сам файл сохраняется в проекте;
* private void SaveButton(object sender, RoutedEventArgs e) – процедура вызывается при нажатии на кнопку «Сохранить». После чего она сохраняет изменённые и добавленные новые документы сотрудников.

В модулях EquipmentAddWindow.xaml.cs, EquipmentEditWindow.xaml.cs описаны и используются следующие процедуры:

* private LoadFacilities() – процедура загружает список помещений из базы данных и отображает его в элементе управления «Наименование помещения» (FacilitiesName). Используется объект dbAccess, который предоставляет доступ к данным из базы данных;
* private LoadSuppliers() – процедура загружает список поставщиков из базы даных и отображает его в элементе управления «Поставщики» (Suppliers). Используется объект dbAccess, который предоставляет доступ к данным из базы данных;
* private void SaveButton\_Click(object sender, RoutedEventArgs e) – процедура вызывается при нажатии на кнопку «Сохранить». После чего она сохраняет изменённые и добавленные новые данные о материалах в базу данных.

В модуле HouseHoldPartAccess описаны и используется следующие процедуры:

* public List<Employee> GetEmployee() – процедура обеспечивает хранение данных из таблицы Employee;
* public void AddScheduleEmployee(ScheduleEmployee schedule) – процедура обеспечивает добавления новых данных в таблицу Employee;
* public List<ScheduleEmployee> GetScheduleEmployee() – процедура обеспечивает хранение данных из таблицы ScheduleEmployee;
* public void RemoveScheduleEmployee(int ScheduleId) – процедура обеспечивает удаление данных из таблицы ScheduleEmployee;
* public void UpdateScheduleEmployee(ScheduleEmployee schedule) – процедура обеспечивает обновление данных из таблицы ScheduleEmployee;
* public List<Employee> GetEmployeesByPost(int idPost) – процедура обеспечивает хранение данных из таблицы Employee;
* public void AddEmployee(Employee employee) – процедура обеспечивает добавление новых данных в таблицу Employee;
* public void UpdateEmployee(Employee employee) – процедура обеспечивает обновление данных в таблице Employee;
* public void DeleteEmployee(int id) – процедура обеспечивает удаление данных из таблицы Employee;
* public List<Equipment> GetAllEquipment() – процедура обеспечивает хранение данных из таблицы Equipment;
* public List<Equipment> GetEquipmentByFacilities(int idFacilities) – процедура обеспечивает хранение данных из таблицы Facilities;
* public void AddEquipment(Equipment equipment) – процедура обеспечивает добавление новых данных в таблицу Equipment;
* public void UpdateEquipment(Equipment equipment) – процедура обеспечивает обновление данных в таблице Equipment;
* public void DeleteEquipment(int id) – процедура обеспечивает удаление данных из таблицы Equipment.

**2.3 Входные и выходные данные модулей**

Входными данными для модуля MainWindow.xaml.cs являются данные со страниц ScheduleEmployeePage.xaml, EmployeePage.xaml, FacilitiesPage.xaml, SuppliersPage.xaml, EquipmentPage.xaml.

Входными данными для страниц ScheduleEmployeePage.xaml, EmployeePage.xaml, FacilitiesPage.xaml, SuppliersPage.xaml, EquipmentPage.xaml являются данные из базы данных, которая загружается и выводиться благодаря ObservableCollection и классу HouseHoldPartAccess.cs. Информация выводиться в DataGrid и ListView.

**2.4 Установка и настройка программы**

Для установки программы необходимо запустить файл инсталляции HouseHoldWorkers.exe. В результате в выбранном пользователем каталоге будет создана папка «HouseHoldWorkers», в которой будут размещены исполняемый файл программы (HouseHoldWorkers.exe). Также будут созданы ярлыки для запуска приложения на рабочем столе и в меню «Пуск». Ярлык для деинсталляции программы будет размещён в меню «Пуск».

Для нормальной работы программы необходимо:

* 300МЬ свободного места на жёстком диске;
* операционная система7/8/10;
* NetFramework 4.5+.

Для создания инсталляции была использована программа SmartInstallMaker 5.04. В ходе инсталляции пользователю показываются следующие окна:

* выбор языка установки;
* лицензионное соглашение – с ним необходимо согласиться;
* краткая информация об устанавливаемом приложении – назначение, минимальные требования к программному обеспечению и техническим средствам;
* выбор папки установки;
* создание ярлыков в меню «Пуск» – позволяет выбрать или создать папку в меню «Пуск», куда будут помещаться ярлыки, или отказаться от создания ярлыков;
* создание дополнительных ярлыков – позволяет создать ярлык на рабочем столе или отказаться от создания ярлыка;
* ход выполнения процесса установки;
* завершение установки – позволяет выбрать файлы, которые нужно запустить после установки.

**2.5 Работа с программой**

Приложение было разработано на языке C# на платформе .NET Framework 4.8. Для создания графического интерфейса использовалась технология WPF. На рисунке 9 представлено окно загрузки программы.

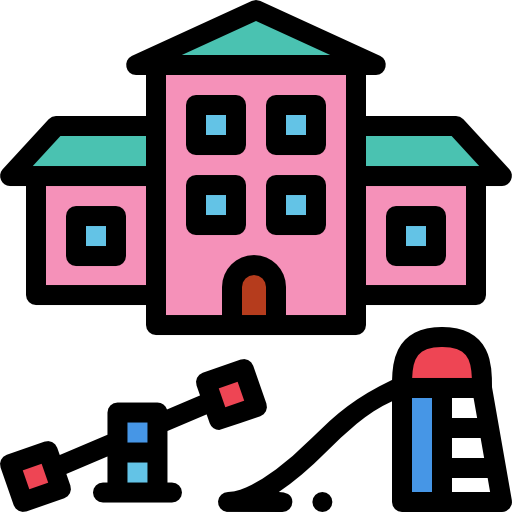


Рисунок 9 – Окно запуска программы

После завершения загрузки, откроется окно с авторизацией в приложении (рисунок 10).

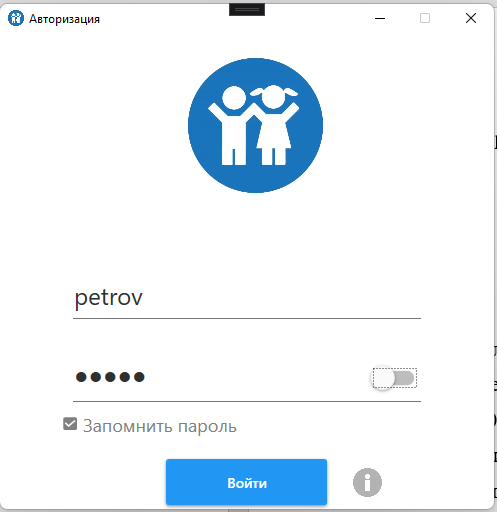


Рисунок 10 – Окно авторизации

В окне авторизации пользователь может просмотреть пароль нажав на «кнопку-переключатель» в поле для ввода пароля (рисунок 11).

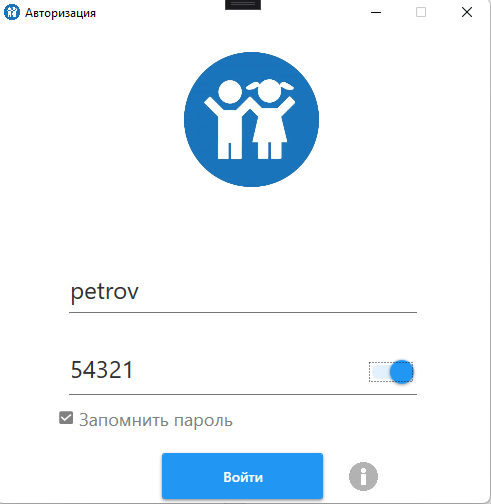


Рисунок 11 – Активная кнопка просмотра пароля

Также в окне авторизации пользователь может нажать на кнопку «Запомнить пароль», после чего при следующих запусках программы поля для ввода логина и пароля будут заполнены.

Из окна авторизации пользователь может перейти в окно справки, нажав на иконку справочника или же авторизоваться, нажав на кнопку «Войти». При неправильном вводе логина или пароля пользователь увидит диалоговое окно с сообщением об ошибке (рисунок 12).

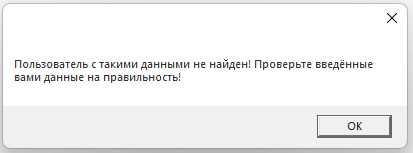


Рисунок 12 – Диалоговое окно, с сообщением о неправильном вводе логина или пароля

После успешной авторизации пользователь увидит сообщение об успешном входе, после чего откроется главное окно, в котором сразу отображается информация о расписании сотрудников (рисунок 13)

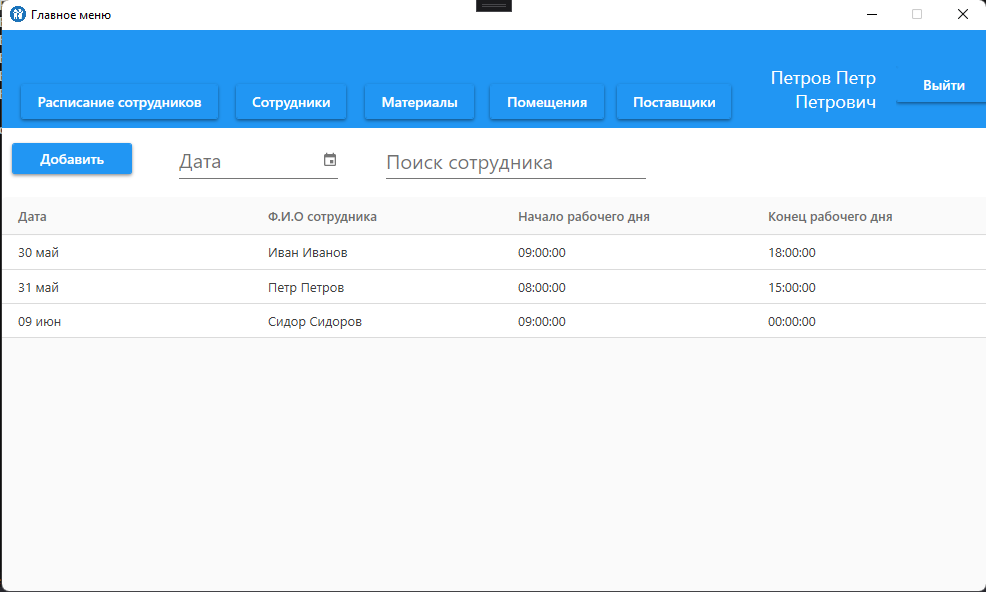


Рисунок 13 – Главное меню, с информацией о расписании сотрудников

В главном меню пользователь увидит кнопки для перехода по страницам (Расписание сотрудников, сотрудники, материалы, помещения и поставщики). Также справа сверху пользователь может увидеть свои данные и кнопку «Выход», для выхода назад в окно авторизации.

На странице «Расписание сотрудников» пользователь может совершить поиск данных по дате или по фамилии и имени сотрудника (рисунок 14).

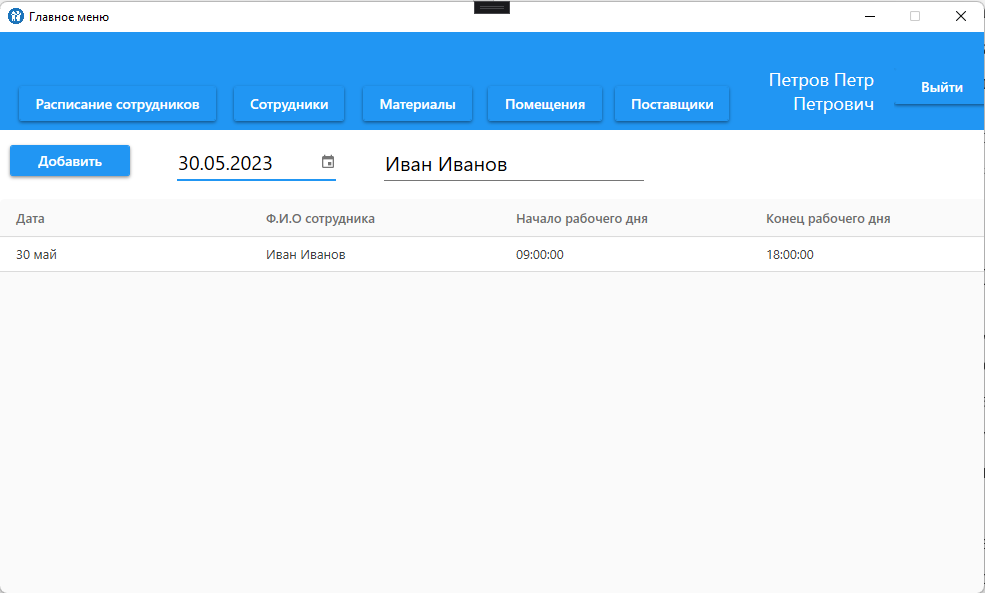


Рисунок 14 – Поиск по дате, фамилии и имени

При нажатии правой кнопкой мыши по любому поле в таблице с информацией о расписании сотрудника пользователь увидит контекстное меню (рисунок 15), в котором он может:

* Открыть окно редактирования данных выбранной строки расписания;
* Удалить строку с выбранным расписанием из базы данных;
* Копировать фамилию и имя сотрудника.

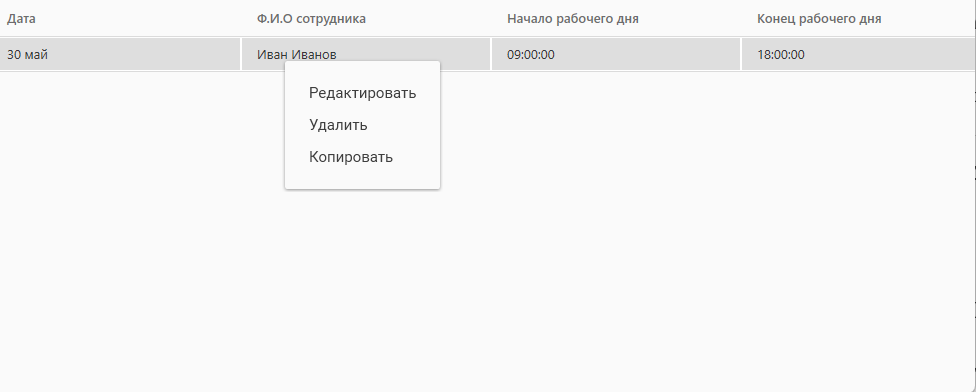


Рисунок 16 – Контекстное меню в расписании сотрудников

При нажатии на кнопку «Редактировать» откроется окно редактирования расписания (рисунок 17).

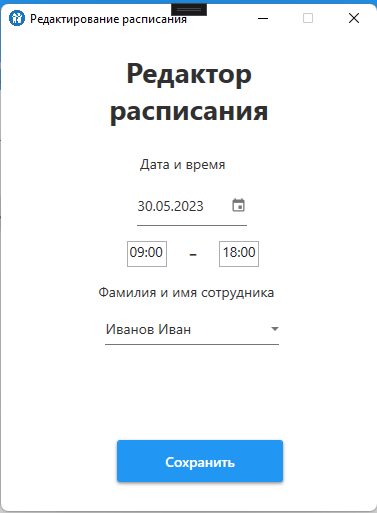


Рисунок 17 – Окно редактирования расписания

Если пользователь не заполнит все данные, то откроется диалоговое окно с информацией, что не все поля были заполнены (рисунок 18).

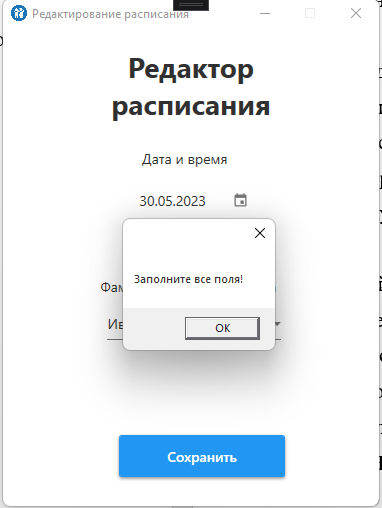


Рисунок 18 – Диалоговое окно, с сообщением о том, что необходимо заполнить все поля в окне редактора расписания

При правильном заполнении всех данных, информация о расписании обновится.

При нажатии на кнопку «Удалить», пользователь увидит диалоговое окно, где идёт уточнение, точно ли пользователь хочет удалить расписание (рисунок 19).

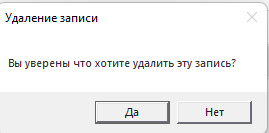


Рисунок 19 – Диалоговое окно «Удаление записи»

Если пользователь выберет «Да», выбранное расписание удалиться из программы и базы данных, если «Нет», то ничего не произойдёт.

При нажатии на кнопку «Добавить расписание» откроется окно добавления нового расписания (рисунок 20), которое выглядит также как и окно «Редактирования расписания».

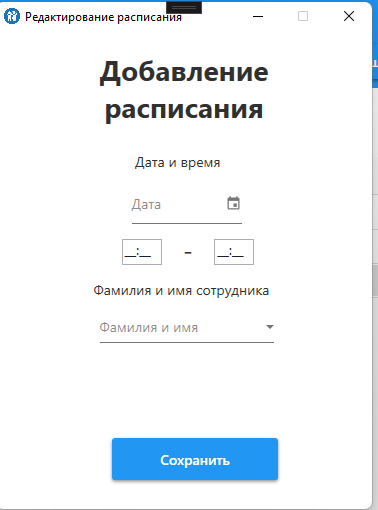


Рисунок 20 – Окно добавления нового расписания

При правильном заполнении данных новое расписание будет успешно загружено в программу и базу данных, о чём пользователю сообщит высветившееся диалоговое окно (рисунок 21).

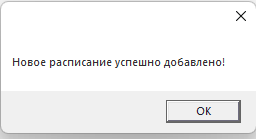


Рисунок 21 – Диалоговое окно, с сообщением об успешном добавлении нового расписания

**ЗАКЛЮЧЕНИЕ**

В рамках дипломного проекта было разработано приложение для заведующего хозяйственной частью МБДОУ №38 Г.Майкоп, оно значительно повышает эффективность работы сотрудников и повышает качество обслуживания. Приложение позволяет не только вести учет материальных ресурсов, но и контролировать выполнение работ по обслуживанию объектов учреждения.

В результате создания данного приложения для заведующего хозяйственной частью МБДОУ №38 Г.Майкоп получил удобный и эффективный инструмент для управления хозяйственной деятельностью. Благодаря автоматизации учета материальных ресурсов и контролю выполнения работ, удалось существенно ускорить процесс подготовки документов и устранить ошибки в учете.

Некоторыми недостатками приложения для заведующего хозяйственной частью МБДОУ №38 Г. Майкоп являются необходимость постоянного обновления базы данных, сложность использования для непрофильных пользователей и отсутствие возможности интеграции с другими софтверными продуктами.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Албахари Б., Албахари Дж. С# 7.0. Справочник. Полное описание языка. М.: Вильямс, 2018. 1024 с.
2. Дарвин Я. Ф. Android. Сборник рецептов: задачи и решения для разработчиков приложений. 2-е изд. Пер. с англ. СПб.: ООО «Альфа-книга», 2018. 768 с.
3. Кугаевских А. Проектирование информационных систем. Системная и бизнес-аналитика. Новосибирск: НГТУ, 2018. 256 с.
4. Лотка Р. C# и CSLA .NET Framework. Разработка бизнес-объектов. М.: Диалектика / Вильямс, 2018. 842 c.
5. Нейгел К., Ивьен Б., Глинн Д., Уотсон К., Скиннер М. C# 5.0 и платформа .NET 4.5 для профессионалов. М.: Вильямс, 2018. 693 c.
6. Фуфаев Э. В. Базы данных. М.: Академия, 2021. 320 c.
7. Библиотека классов .NET Framework. URL: https://msdn.microsoft.com/ ru-ru/library/mt472912(v=vs.110) (дата обращения: 10.05.2023).
8. Общие сведения о платформе Android. URL: https://developer. android.com/guide (дата обращения: 10.05.2023).
9. Официальный сайт HTML HelpWorkShop. URL: https:// www.microsoft.com (дата обращения: 10.05.2023).
10. Официальный сайт Smart InstallMaker. URL: (дата обращения: 10.05.2023).
11. О REST API простыми словами. URL: https://smoff.ru/howitworks/rest-api-prostymi-slovami? (дата обращения: 10.05.2023).
12. Простое руководство по UML-диаграммам и моделированию баз данных. URL: https://www.microsoft.com/ru-ru/microsoft-365/business-insights-ideas/resources/ guide-to-uml-diagramming-and-database-modeling (дата обращения: 10.05.2023).

**ПРИЛОЖЕНИЕ А  
(справочное)**

**Листинг модулей**

**Модуль BaseViewModel.cs**

public abstract class BaseViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

// Стандартная реализация

protected virtual void OnPropertyChanged([CallerMemberName] string

propertyName = null)

{

PropertyChanged?.Invoke(this, new

PropertyChangedEventArgs(propertyName));

}

// Расширеная реализация, для упрощения работы со свойствами.

protected virtual bool SetPropertyChanged<T>(ref T source, T value,

[CallerMemberName] string propertyName = null)

{

if (EqualityComparer<T>.Default.Equals(source, value))

return false;

source = value;

OnPropertyChanged(propertyName);

return true;

}

}

**Модуль RelayCommand.cs**

public class RelayCommand : ICommand

{

private Action<object> execute;

private Func<object, bool> canExecute;

public event EventHandler CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

public RelayCommand(Action<object> execute, Func<object, bool>

canExecute = null)

{

this.execute = execute;

this.canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return this.canExecute == null || this.canExecute(parameter);

}

public void Execute(object parameter)

{

this.execute(parameter);

}

}

**Модуль EntryWindow.xaml.cs**

private void PasswordCheck\_Checked(object sender, RoutedEventArgs e)

{

TextPasswordInput.Visibility = Visibility.Visible;

PasswordInput.Visibility = Visibility.Hidden;

TextPasswordInput.Text = PasswordInput.Password;

}

private void PasswordCheck\_Unchecked(object sender, RoutedEventArgs e)

{

TextPasswordInput.Visibility = Visibility.Hidden;

PasswordInput.Visibility = Visibility.Visible;

PasswordInput.Password = TextPasswordInput.Text;

}

private void PasswordInput\_PasswordChanged(object sender, RoutedEventArgs e)

{

TextPasswordInput.Text = PasswordInput.Password;

}

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

{

using (var context = new HouseholdPartEntities())

{

var user = context.User.Where(x => x.Login == LoginText.Text && x.Password == TextPasswordInput.Text).FirstOrDefault();

if (user != null)

{

if (SavePass.IsChecked == true)

{

string login = LoginText.Text;

string password = PasswordInput.Password;

string filePath = "savedlogin.txt";

using (StreamWriter writer = new StreamWriter(filePath))

{

writer.WriteLine(login);

writer.WriteLine(password);

}

}

CurrentUser.Name = user.FullNameUser;

MessageBox.Show("Вы успешно авторизовались!");

MainWindow main = new MainWindow();

main.Show();

this.Close();

}

else { MessageBox.Show("Пользователь с такими данными не найден! Проверьте введённые вами данные на правильность!"); }

}

}

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

{

string filePath = "savedlogin.txt";

if (File.Exists(filePath))

{

// читаем сохраненные значения

string[] lines = File.ReadAllLines(filePath);

if (lines.Length >= 2)

{

// загружаем значения в поля логина и пароля

LoginText.Text = lines[0];

PasswordInput.Password = lines[1];

// устанавливаем галочку "Запомнить пароль"

SavePass.IsChecked = true;

}

}

**Модуль MainWindow.xaml.cs**

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

{

EntryWindow entry = new EntryWindow();

entry.Show();

this.Close();

}

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

{

MainFrame.Navigate(new EmployeePage());

}

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

{

MainFrame.Navigate(new ScheduleEmployeePage());

}

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

{

MainFrame.Navigate(new FacilitiesPage());

}

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

{

MainFrame.Navigate(new SuppliersPage());

}

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

{

MainFrame.Navigate(new EquipmentPage());

}

**Модуль ScheduleEmployeePage.xaml.cs**

public ScheduleEmployeePage()

{

InitializeComponent();

using (var context = new HouseholdPartEntities())

{

scheduleTable = new ObservableCollection<ScheduleEmployee>(context.ScheduleEmployee.ToList());

}

ScheduleDG.ItemsSource= scheduleTable;

}

public ObservableCollection<ScheduleEmployee> scheduleTable { get; set; } = new ObservableCollection<ScheduleEmployee>();

public ObservableCollection<ScheduleEmployee> ScheduleTable { get { return scheduleTable; } }

private void SearchDate\_SelectedDateChanged(object sender, SelectionChangedEventArgs e)

{

DateTime? selectedDate = SearchDate.SelectedDate; // Получаем выбранную пользователем дату

ICollectionView view = CollectionViewSource.GetDefaultView(ScheduleDG.ItemsSource); // Получаем представление коллекции

if (selectedDate.HasValue) // Если пользователь выбрал дату

{

view.Filter = item =>

{

ScheduleEmployee schedule = item as ScheduleEmployee; // Получаем текущий объект Schedule

return schedule.Date == selectedDate.Value.Date; // Фильтруем записи по дате

};

}

else // Если пользователь не выбрал дату

{

view.Filter = null; // Отображаем все занятия

}

}

private void SearchText\_TextChanged(object sender, TextChangedEventArgs e)

{

string searchText = SearchText.Text.ToLower(); // Получаем текст поиска в нижнем регистре

ICollectionView view = CollectionViewSource.GetDefaultView(ScheduleDG.ItemsSource); // Получаем представление коллекции

if (!string.IsNullOrEmpty(searchText)) // Если поисковая строка не пуста

{

view.Filter = item =>

{

ScheduleEmployee schedule = item as ScheduleEmployee; // Получаем текущий объект Schedule

using (var context = new HouseholdPartEntities()) // Создаем новый контекст базы данных

{

// Используем LINQ-запрос для поиска сотрудника по имени или фамилии

var searchWords = searchText.Split(' ');

var employee = context.Employee.FirstOrDefault(g => searchWords.All(w => g.NameEmployee.ToLower().Contains(w) || g.SurnameEmployee.ToLower().Contains(w)));

return schedule.ScheduleId == employee?.EmployeeId; // Возвращаем true, если ID сотрудника соответствует ID сотрудника в расписании

}

};

}

else // Если поисковая строка пуста

{

view.Filter = null; // Отображаем все занятия

}

}

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

{

// Получаем выбранную строку

ScheduleEmployee selectedItem = (ScheduleEmployee)((ContextMenu)FindResource("ItemContextMenu")).Tag;

// Получаем значение ячейки столбца "Ф.И.О сотрудника"

string employeeName = selectedItem.EmployeeName;

// Копируем значение в буфер обмена

Clipboard.SetText(employeeName);

}

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

{

ScheduleEmployee selectedItem = (ScheduleEmployee)ScheduleDG.SelectedItem;

if (selectedItem != null && MessageBox.Show("Вы уверены что хотите удалить эту запись?", "Удаление записи", MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

var db = new HouseholdPartEntities();

var dbAccess = new HouseHoldPartAccess(db);

int ScheduleId = selectedItem.ScheduleId;

dbAccess.RemoveScheduleEmployee(ScheduleId);

ScheduleDG.ItemsSource = dbAccess.GetScheduleEmployee();

db.Dispose();

MessageBox.Show("Расписание успешно удалено!");

}

}

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

{

if (ScheduleDG.SelectedItem != null)

{

// Получение данных выбранной строки

ScheduleEmployee selectedItem = (ScheduleEmployee)ScheduleDG.SelectedItem;

// Создание нового экземпляра окна редактирования

ScheduleEmployeeEditWindow editWindow = new ScheduleEmployeeEditWindow(selectedItem);

// Открытие окна

editWindow.ShowDialog();

using (var context = new HouseholdPartEntities())

{

scheduleTable = new ObservableCollection<ScheduleEmployee>(context.ScheduleEmployee.ToList());

}

ScheduleDG.ItemsSource = scheduleTable;

}

}

public static class VisualTreeHelpers

{

public static T GetParentOfType<T>(DependencyObject element) where T : DependencyObject

{

DependencyObject parent = VisualTreeHelper.GetParent(element);

if (parent == null)

return null;

T parentOfType = parent as T;

if (parentOfType != null)

return parentOfType;

return GetParentOfType<T>(parent);

}

}

private void ScheduleDG\_RowPreviewMouseRightButtonDown(object sender, MouseButtonEventArgs e)

{

if (e.RightButton == MouseButtonState.Pressed)

{

DataGridRow row = VisualTreeHelpers.GetParentOfType<DataGridRow>(e.OriginalSource as FrameworkElement);

if (row != null)

{

ScheduleDG.SelectedItem = row.DataContext;

// Присваиваем Tag выбранной строке

((ContextMenu)FindResource("ItemContextMenu")).Tag = ScheduleDG.SelectedItem;

// Открываем контекстное меню

((ContextMenu)FindResource("ItemContextMenu")).IsOpen = true;

e.Handled = true;

}

}

}

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

{

var db = new HouseholdPartEntities();

var dbAccess = new HouseHoldPartAccess(db);

AddScheduleEmployeeWindow add = new AddScheduleEmployeeWindow();

add.ShowDialog();

ScheduleDG.ItemsSource = dbAccess.GetScheduleEmployee();

db.Dispose();

}

}

**Модуль ScheduleEmployeeAddWindow.xaml.cs**

public AddScheduleEmployeeWindow()

{

InitializeComponent();

LoadEmployee();

}

private void LoadEmployee()

{

var db = new HouseholdPartEntities();

var dbAccess = new HouseHoldPartAccess(db);

var employee = dbAccess.GetEmployee();

EmployeeName.ItemsSource = employee;

}

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

{

if (Date.SelectedDate == null || TimeText.Text == "" || TimeEndText.Text == "" || EmployeeName.SelectedItem == null)

{

MessageBox.Show("Пожалуйста заполните все поля");

return;

}

var db = new HouseholdPartEntities();

var dbAccess = new HouseHoldPartAccess(db);

// Create new schedule object using inputted data

var newSchedule = new ScheduleEmployee

{

Date = Date.SelectedDate.Value,

TimeStart = TimeSpan.Parse(TimeText.Text),

TimeEnd = TimeSpan.Parse(TimeEndText.Text),

IdEmployee = (int)EmployeeName.SelectedValue

};

// Add new schedule to database

dbAccess.AddScheduleEmployee(newSchedule);

db.SaveChanges();

MessageBox.Show("Новое расписание успешно добавлено!");

this.Close();

}

}

**Модуль ScheduleEmployeeEditWindow.xaml.cs**

public ScheduleEmployeeEditWindow(ScheduleEmployee selectedItem)

: base()

{

InitializeComponent();

DataContext = selectedItem;

LoadEmployee();

}

private void LoadEmployee()

{

var db = new HouseholdPartEntities();

var dbAccess = new HouseHoldPartAccess(db);

var employee = dbAccess.GetEmployee();

EmployeeName.ItemsSource = employee;

}

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

{

var db = new HouseholdPartEntities();

var dbAccess = new HouseHoldPartAccess(db);

if (DataContext is ScheduleEmployee schedule)

{

if (string.IsNullOrEmpty(schedule.EmployeeName) || !schedule.Date.HasValue || !schedule.TimeStart.HasValue || !schedule.TimeEnd.HasValue)

{

MessageBox.Show("Заполните все поля!");

}

else

{

dbAccess.UpdateScheduleEmployee(schedule);

MessageBox.Show("Данные успешно сохранены!");

this.Close();

}

}

}

**Модуль TimeSpanToStringConverter.cs**

public class TimeSpanToStringConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is TimeSpan timeSpan)

{

// Преобразование TimeSpan в строку в формате "чч:мм"

return timeSpan.ToString("hh':'mm");

}

return null;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

if (value is string timeString && TimeSpan.TryParseExact(timeString, "hh':'mm", culture, out TimeSpan timeSpan))

{

// Преобразование строки в TimeSpan

return timeSpan;

}

return null;

}

}

**Модуль HouseHoldPartAccess.cs**

public class HouseHoldPartAccess

{

private readonly HouseholdPartEntities \_db;

public HouseHoldPartAccess(HouseholdPartEntities db)

{

\_db= db;

}

public List<User> GetAllUsers()

{

return \_db.User.ToList();

}

public User GetUserById(int id)

{

return \_db.User.FirstOrDefault(u => u.UserId == id);

}

public void AddUser(User user)

{

\_db.User.Add(user);

\_db.SaveChanges();

}

public void UpdateUser(User user)

{

\_db.Entry(user).State = EntityState.Modified;

\_db.SaveChanges();

}

public void DeleteUser(int id)

{

User user = \_db.User.Find(id);

\_db.User.Remove(user);

\_db.SaveChanges();

}

public List<Employee> GetEmployee()

{

return \_db.Employee.ToList();

}

public void AddScheduleEmployee(ScheduleEmployee schedule)

{

\_db.ScheduleEmployee.Add(schedule);

\_db.SaveChanges();

}

public List<ScheduleEmployee> GetScheduleEmployee()

{

return \_db.ScheduleEmployee.ToList();

}

public void RemoveScheduleEmployee(int ScheduleId)

{

var schedule = \_db.ScheduleEmployee.Find(ScheduleId);

if (schedule != null )

{

\_db.ScheduleEmployee.Remove(schedule);

\_db.SaveChanges();

}

}

public void UpdateScheduleEmployee(ScheduleEmployee schedule)

{

var dbSchedule = \_db.ScheduleEmployee.Find(schedule.ScheduleId);

if (dbSchedule != null)

{

dbSchedule.Date = schedule.Date;

dbSchedule.TimeStart= schedule.TimeStart;

dbSchedule.TimeEnd= schedule.TimeEnd;

dbSchedule.IdEmployee = schedule.IdEmployee;

\_db.SaveChanges();

}

}

public List<Employee> GetEmployeesByPost(int idPost)

{

return \_db.Employee.Where(e => e.IdPost == idPost).ToList();

}

public void AddEmployee(Employee employee)

{

\_db.Employee.Add(employee);

\_db.SaveChanges();

}

public void UpdateEmployee(Employee employee)

{

\_db.Entry(employee).State = EntityState.Modified;

\_db.SaveChanges();

}

public void DeleteEmployee(int id)

{

Employee employee = \_db.Employee.Find(id);

\_db.Employee.Remove(employee);

\_db.SaveChanges();

}

public List<Equipment> GetAllEquipment()

{

return \_db.Equipment.ToList();

}

public List<Equipment> GetEquipmentByFacilities(int idFacilities)

{

return \_db.Equipment.Where(e => e.IdFacilities == idFacilities).ToList();

}

public void AddEquipment(Equipment equipment)

{

\_db.Equipment.Add(equipment);

\_db.SaveChanges();

}

public void UpdateEquipment(Equipment equipment)

{

\_db.Entry(equipment).State = EntityState.Modified;

\_db.SaveChanges();

}

public void DeleteEquipment(int id)

{

Equipment equipment = \_db.Equipment.Find(id);

\_db.Equipment.Remove(equipment);

\_db.SaveChanges();

}

}