Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Дисциплина: Системы управления базами данных

Пояснительная записка

к курсовому проекту

на тему

Система тестирования студентов

КП 1-53 01 02 01 006 ПЗ

Студент гр. 120604 Муртузалиева К. А.

Руководитель Трофимович А. Ф.

Минск 2023

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Цели курсового проекта:

1. Создание единой системы обработки и хранения информации о членах Профсоюза для всей Федерации профсоюзов Беларуси.

2. Повышение результативности работы с базой данных членства Профсоюза путем оптимизации процессов хранения и управления данными.

3. Упрощение процесса отслеживания периодичности сбора взносов от членов Профсоюза.

4. Оптимизация совместной работы с членской базой Профсоюза в режиме многопользовательского взаимодействия.

5. Улучшение эргономики приложения, учитывая метрики исследования удобства пользовательского интерфейса.

Задачи курсового проекта:

1. Разработка инструментов для работы с базой данных членства Профсоюза.

2. Упорядоченное хранение информации о членах Профсоюза на жестком диске компьютера.

3. Реализация инструментов для отслеживания сбора взносов, включая отображение всех взносов на личной карточке каждого члена Профсоюза.

4. Создание инструментов для формирования списков в формате XLSX.

Значимость и актуальность проекта:

Данный курсовой проект имеет большую значимость для профсоюзов различных отраслей, так как позволяет им обзавестись электронным набором инструментов для удобной и эффективной работы с большим количеством членов Профсоюза. Актуальность проекта обусловлена следующими факторами:

1. Отсутствие единой системы учета и управления информацией о членах Профсоюза во многих университетах и других организациях.

2. Необходимость оптимизации работы с базой данных членов Профсоюза и упрощения процесса доступа к информации.

3. Устаревание бумажных форматов и несоответствие современным стандартам эргономики интерфейса.

4. Возможность автоматизировать поиск личных карточек и информации о членах Профсоюза, сократить бумажную работу и минимизировать риски потери личной информации.

Целевые группы проекта:

Целевой группой проекта являются Профсоюзы студентов и работников учреждений образования, фабрик, заводов, колхозов, совхозов, а также других отраслевых профсоюзов.

Ожидаемые результаты проекта:

1. Увеличение скорости и качества работы Профсоюзов.

2. Автоматизация поиска личных карточек и информации о членах Профсоюза.

3. Сокращение бумажной работы и минимизация вероятности потери личной информации.

4. Улучшение совместной работы с членской базой Профсоюза в режиме многопользовательского взаимодействия.

5. Оптимизация эргономики приложения, что приведет к более удобному использованию и повышению удовлетворенности пользователями.

СОДЕРЖАНИЕ

[**ВВЕДЕНИЕ** 4](#_Toc135193312)

[**1.** **ОБЗОР ИССЛЕДУЕМОЙ ОБЛАСТи** 5](#_Toc135193313)

[**1.1.** **ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ** 5](#_Toc135193314)

[**1.2.** **Обзор аналогов** 5](#_Toc135193315)

[**1.3.** **Постановка задачи** 8](#_Toc135193316)

[**2.** **ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ ПРОФСОЮЗА** 10](#_Toc135193317)

[**2.1.** **Проектированние базы данных** 10](#_Toc135193318)

[**2.2.** **Проектирование приложения** 11](#_Toc135193319)

[**3.** **РАЗРАБОТКА БАЗЫ ПРОФСОЮЗА** 13](#_Toc135193320)

[**3.1.** **Выбор средств реализации** 13](#_Toc135193321)

[**3.2.** **Реализация базы данных** 14](#_Toc135193322)

[3.3. Программная реализация приложения 18](#_Toc135193323)

[**3.4.** **Инструкция пользователя** 24](#_Toc135193324)

[**ЗАКЛЮЧЕНИЕ** 28](#_Toc135193325)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 29](#_Toc135193326)

[**ПРИЛОЖЕНИЕ А** 30](#_Toc135193327)

# **ВВЕДЕНИЕ**

Курсовая работа посвящена разработке приложения для управления базой данных членов профсоюза. В современном информационном обществе базы данных играют важную роль в организации, хранении и управлении большим объемом данных. Базы данных служат основой для множества приложений и систем, обеспечивая эффективное и надежное хранение информации.

В контексте данной курсовой работы, мы фокусируемся на применении баз данных в связи с дисциплиной, посвященной базам и банкам данных. Профсоюзы являются организациями, защищающими интересы работников и обеспечивающими социальную поддержку. Членство в профсоюзе предоставляет работникам определенные привилегии и льготы. Управление базой данных членов профсоюза позволяет эффективно вести учет и обработку информации о членах, их личных данных, работе и мероприятиях, связанных с профсоюзной деятельностью.

Целью курсовой работы является разработка функционального приложения, которое облегчит процессы управления базой данных членов профсоюза. Приложение позволит добавлять, редактировать и удалять информацию о членах профсоюза, а также осуществлять поиск и фильтрацию данных в удобном интерфейсе. Оно будет обеспечивать безопасное хранение и доступ к информации, а также предоставлять различные функциональные возможности, например, формирование отчетов и экспорт данных.

# **ОБЗОР ИССЛЕДУЕМОЙ ОБЛАСТи**

* 1. **ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

Предметная область данной курсовой работы связана с управлением базой данных членов профсоюза. Профсоюзы являются организациями, объединяющими работников различных отраслей и защищающими их права и интересы. Членство в профсоюзе дает работникам определенные преимущества и привилегии, такие как коллективное переговоры, защиту от неправомерных действий работодателя, социальную поддержку и другие виды помощи.

Управление базой данных членов профсоюза имеет большое значение для эффективного функционирования профсоюзной организации. База данных содержит информацию о каждом члене профсоюза, включая его личные данные, контактную информацию, дату вступления в профсоюз, историю участия в профсоюзных мероприятиях, информацию о текущей работе и должности, а также другие сведения, которые могут быть полезными для организации и обеспечения поддержки членам профсоюза.

Основной целью управления базой данных членов профсоюза является обеспечение надежного хранения, актуализации и обработки данных о членах профсоюза. Это позволяет профсоюзной организации эффективно управлять своими ресурсами, предоставлять необходимые услуги и льготы членам, проводить статистический анализ и формировать отчетность, а также поддерживать связь и взаимодействие с каждым членом профсоюза.

Предметная область также включает в себя дисциплины, связанные с базами данных и банками данных. Это включает проектирование и разработку структуры базы данных, определение связей между таблицами, оптимизацию запросов, обеспечение безопасности данных, создание пользовательского интерфейса для работы с базой данных и другие аспекты, связанные с управлением и обработкой данных.

Разработка приложения для управления базой данных членов профсоюза в контексте предметной области позволяет углубить знания и навыки в области баз данных, понять специфику работы с данными в профсоюзной среде и применить полученные знания для разработки функционального и эффективного приложения, облегчающего управление информацией о членах профсоюза.

* 1. **Обзор аналогов**

При разработке приложения для управления базой данных членов профсоюза стоит провести обзор аналогичных приложений, которые уже существуют. В рамках этого обзора рассмотрим сравнение базы данных профкома с базой данных университета, выявим их достоинства и недостатки.

База данных профкома (предположим, что она уже существует) специально разработана для управления информацией о членах профсоюза и содержит функционал, связанный с профсоюзными активностями. Она предоставляет возможность хранить и обновлять информацию о членах, отслеживать их участие в мероприятиях и предоставлять доступ к привилегиям, связанным с членством в профсоюзе. Еще некоторые возможности, которые может предоставлять база данных профкома, могут включать формирование отчетов, управление финансами, организацию выборов и т. д. Эти функции специально нацелены на нужды профсоюза и его членов.

С другой стороны, база данных университета, возможно, не имеет такого узкой специализации и ориентирована на общие потребности университетской системы. Она может хранить информацию о студентах, преподавателях, курсах и других аспектах, связанных с образовательным процессом. Однако база данных университета может иметь недостатки в контексте управления информацией о членах профсоюза. Например, она может не предоставлять возможность одновременной работы с несколькими членами профсоюза или иметь ограниченный функционал, специфичный для профсоюза. Кроме того, база данных университета может не обладать гибкими правами доступа и настройками для учета особенностей работы профсоюза.

При разработке приложения для базы данных членов профсоюза можно учесть недостатки базы данных университета и предоставить решения для обеспечения функциональности и гибкости, связанных с профсоюзными нуждами. Это может включать возможность работы с несколькими членами одновременно, расширенный функционал, связанный с профсоюзными мероприятиями и финансовым учетом, а также гибкие права доступа и возможность настройки приложения под конкретные требования профсоюза.

Таким образом, приложение для базы данных членов профсоюза будет иметь преимущества перед базой данных университета в виде специализированного функционала, большей гибкости и настраиваемости, а также возможности эффективно управлять информацией о членах профсоюза.

К примеру, на главной странице базы БГУИР отображается информация ненужная для применения в профсоюзе, что утяжеляет процесс работы с базой членства (см. рис. 1.1).

Также при выгрузке в формат *XLSX* нет какой-либо полезной для профсоюза информации о каждом человеке (см. рис. 1.2).

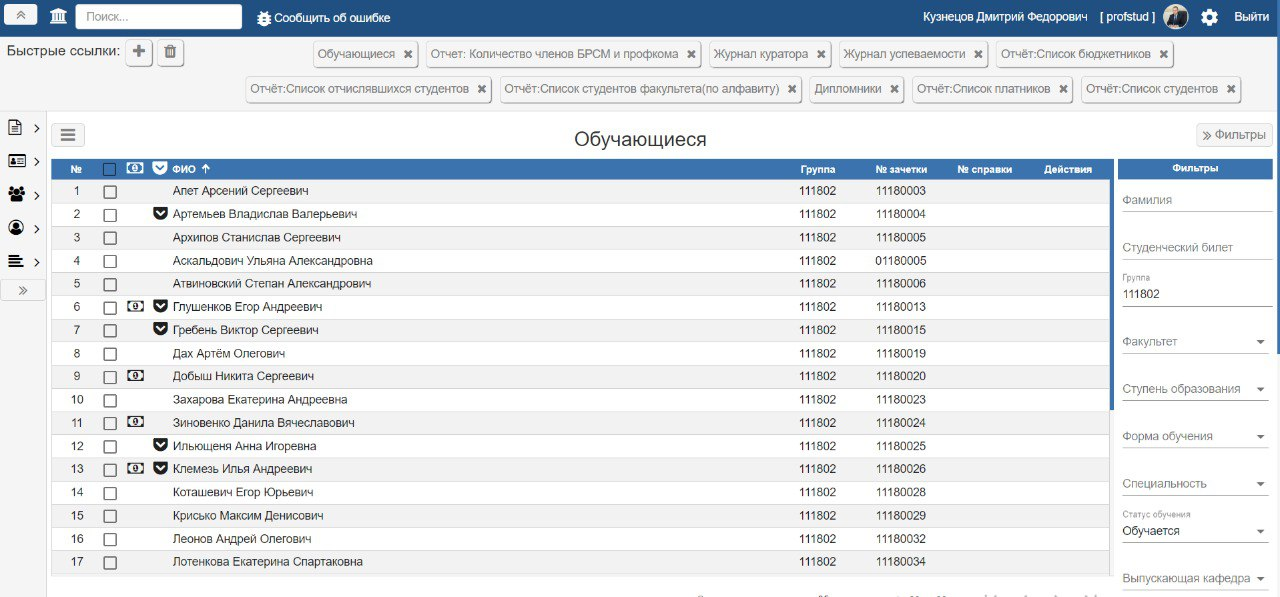


Рисунок 1.1 – Главная панель базы данных БГУИР

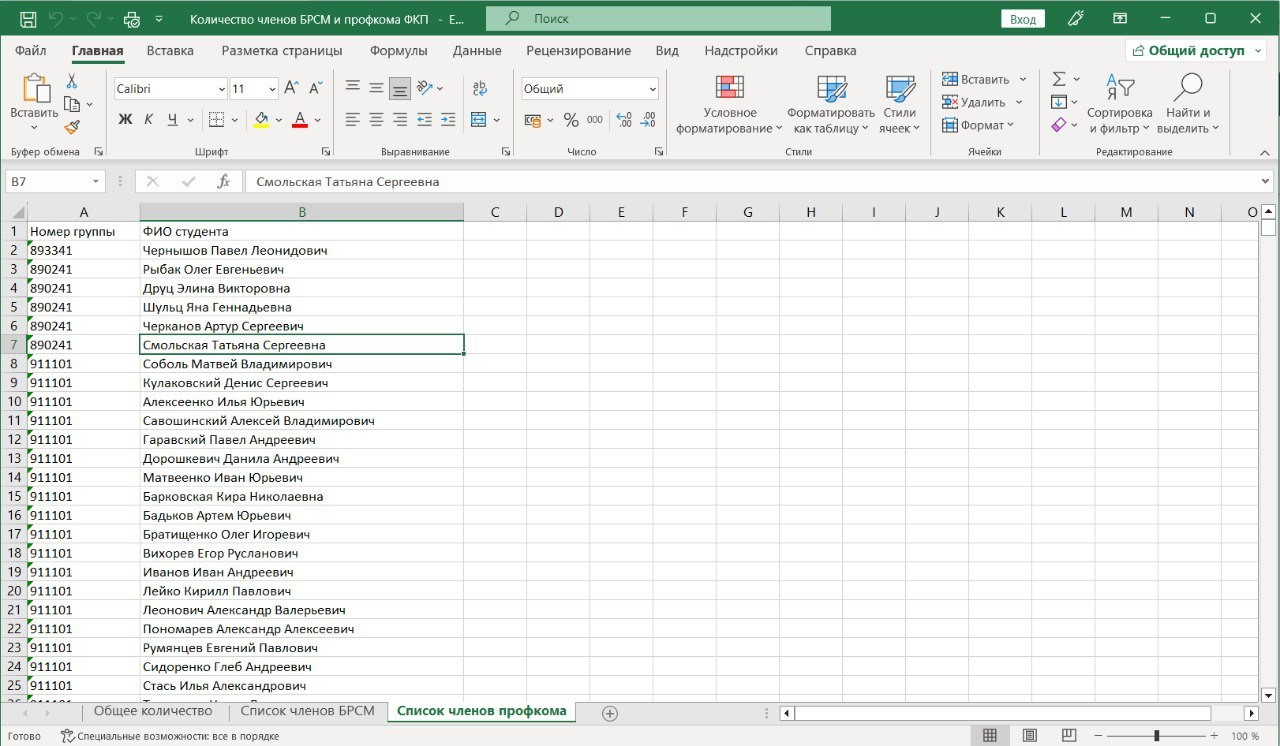


Рисунок 1.2 – Выгрузка в формат *XLSX*

* 1. **Постановка задачи**

В данном разделе мы определим основные требования к функционалу приложения для базы данных членов профсоюза, а также рассмотрим требования к аппаратной и программной платформе, на которых будет работать приложение.

**Требования к функционалу**

Основные требования к функционалу приложения для базы данных членов профсоюза:

1. Управление информацией о членах профсоюза: приложение должно предоставлять возможность добавления, редактирования и удаления записей о членах профсоюза. Пользователи должны иметь доступ к информации о членах, включая персональные данные, контактные данные, статус членства и другую связанную информацию.

2. Организация профсоюзных мероприятий: приложение должно позволять планировать и управлять профсоюзными мероприятиями, такими как собрания, конференции, тренинги и т. д. Это может включать возможность создания событий, установки даты и времени, приглашения участников и отслеживания их участия.

3. Финансовый учет: приложение должно иметь функционал для учета финансовых операций профсоюза, включая взносы членов, расходы на мероприятия, бюджет профсоюза и отчетность по финансам.

4. Права доступа и безопасность: приложение должно обеспечивать гибкие права доступа, позволяющие разграничивать доступ к информации в зависимости от роли пользователя. Также следует предусмотреть механизм аутентификации и защиту данных от несанкционированного доступа.

5. Отчетность и аналитика: приложение должно предоставлять возможность генерации отчетов и аналитической информации о состоянии профсоюза, членстве, финансовых показателях и других аспектах работы профсоюза.

**Требования к аппаратной платформе**

Требования к аппаратной платформе определяют необходимую конфигурацию оборудования, на котором будет работать приложение. Эти требования могут включать:

1. Компьютеры или серверы с достаточным объемом оперативной памяти и процессор

ной мощности для обработки и хранения данных.

2. Доступ к сети Интернет для возможности удаленного доступа к приложению и обмена данными.

3. Надежное хранилище данных, такое как сервер базы данных, с достаточным объемом хранения для всех записей о членах профсоюза и связанной информации.

**Требования к программной платформе**

Требования к программной платформе определяют необходимые программные компоненты и технологии, необходимые для работы приложения. Эти требования могут включать:

1. Операционная система: определенная операционная система, такая как *Windows*, *Linux* или *macOS*, на которой будет работать приложение.

2. Среда выполнения: определенная среда выполнения, такая как .*NET* Framework, *ASP.NET* *Core* или *Node*.*js*, необходимая для запуска и работы приложения.

3. База данных: определенная система управления базами данных, такая как *Microsoft SQL Server, MySQL или PostgreSQL*, необходимая для хранения данных членов профсоюза.

4. Язык программирования: определенный язык программирования, такой как *C*#, *Java* или *Python*, для разработки приложения.

5. Фреймворки и библиотеки: определенные фреймворки и библиотеки, которые могут использоваться для разработки различных компонентов приложения, таких как *ASP.NET Core, Entity Framework, Angular* или *React*.

6. Инструменты разработки: определенные интегрированные среды разработки (*IDE*) или другие инструменты, которые будут использоваться для разработки и отладки приложения.

Определение конкретных требований к программной платформе требует анализа проекта и выбора технологий, наиболее подходящих для реализации функционала приложения для базы данных членов профсоюза.

1. **ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ ПРОФСОЮЗА**
   1. **Проектированние базы данных**

В этом разделе мы рассмотрим процесс проектирования базы данных для приложения членов профсоюза. Мы начнем с описания сущностей и атрибутов, а затем перейдем к нормализации, чтобы обеспечить эффективное хранение и управление данными.

**Описание сущностей**

Сущность Человек:: идентификатор члена профсоюза (первичный ключ), имя, фамилия, отчество, телефона, битовое поле, указывающее на семейное положение, дата рождения,адрес электронной почты, пол, гражданство, фотография (массив байтов.

Сущность Отец: идентификатор отца (первичный ключ), имя, фамилия, отчество, номер телефона, должность. Имеет связь с сущностью Человек один к одному.

3. Сущность Мать: идентификатор матери (первичный ключ), имя, фамилия, отчество, номер телефона, должность. Имеет связь с сущностью Человек один к одному.

4. Сущность Дети: идентификатор ребенка (первичный ключ), имя, фамилия, отчество, номер телефона, дата рождения. Имеет связь с сущностью Человек один ко многим.

5. Сущность Персональные файлы: идентификатор личного дела (первичный ключ),номер личного дела. Имеет связь с сущностью Человек один к одному.

6. Сущность Регистрация: идентификатор регистрации (первичный ключ), страна, регион, город, улица, дом, номер квартиры, почтовый индекс. Имеет связь с сущностью Человек один к одному.

7. Сущность Место жительства: идентификатор местонахождения (первичный ключ), страна, регион, город, улица, дом, номер квартиры, почтовый индекс. Имеет связь с сущностью Человек один к одному.

8. Сущность Должность: идентификатор должности (первичный ключ), название отдела, должность). Имеет связь с сущностью Человек один к одному.

9. Сущность Образование: идентификатор образования (первичный ключ), учебное заведение, специальность, дата окончания обучения, степень. Имеет связь с сущностью Человек один к одному.

10. Сущность Отдел: идентификатор отдела (первичный ключ), название отдела. Имеет связь с сущностью Человек один к одному.

11. Сущность Должность в профсоюзе: идентификатор должности в профкоме (первичный ключ), название должности, дата назначения на должность. Имеет связь с сущностью Человек один к одному.

12. Сущность Членство: идентификатор членства (первичный ключ), дата вступления (по умолчанию - текущая дата), идентификатор отдела (внешний ключ, ссылается на *Departments*.*DepID*). Ссылается на Сущность Должность в профсоюзе, имеет связь один ко многим, дата взноса (по умолчанию - текущая дата). Имеет связь с сущностью Человек один к одному.

13. Сущность Награда: идентификатор награды (первичный ключ), тип награды, дата получения, примечание. Имеет связь с сущностью Членство один ко многим.

14. Таблица Штрафы: идентификатор штрафа (первичный ключ), тип штрафа, дата получения,примечание. Имеет связь с сущностью Членство один ко многим.

Эта база данных позволяет хранить информацию о членах профсоюза, их семейном статусе, детях, личных файлах, адресах проживания, должностях, образовании, членстве в профкоме, наградах и штрафах. Таблицы связаны между собой при помощи внешних ключей, что обеспечивает целостность данных и возможность получения связанной информации из разных таблиц.

* 1. **Проектирование приложения**

При проектировании логики работы приложения для базы данных профсоюза можно рассмотреть следующие аспекты:

Аутентификация и авторизация:

- Реализовать механизм аутентификации пользователей, чтобы они могли войти в систему под своей учетной записью.

- Обеспечить механизм авторизации, чтобы определить права доступа каждого пользователя и ограничить функциональность в соответствии с его ролью (обычный пользователь или администратор).

Управление пользователями:

- Реализовать функционал добавления новых пользователей, включая ввод и сохранение информации о них в базу данных.

- Обеспечить возможность просмотра списка пользователей с возможностью фильтрации и сортировки.

- Разработать функционал выгрузки пользователей в формате *Excel* или другом удобном формате.

3. Управление цеховыми организациями и должностями:

- Разработать механизм добавления новых цеховых организаций и должностей, включая сохранение информации о них в базе данных.

- Обеспечить возможность просмотра списка цеховых организаций и должностей, а также редактирования и удаления существующих записей.

В данной диаграмме вариантов использования (см рис 2.1) представлены два актора - Пользователь и Администратор. Оба актора имеют общие возможности по добавлению, просмотру и выгрузке пользователей. Пользователь может войти в систему под своей учетной записью. Администратор, кроме общих возможностей, также имеет возможность добавлять организации, должности и пользователей.

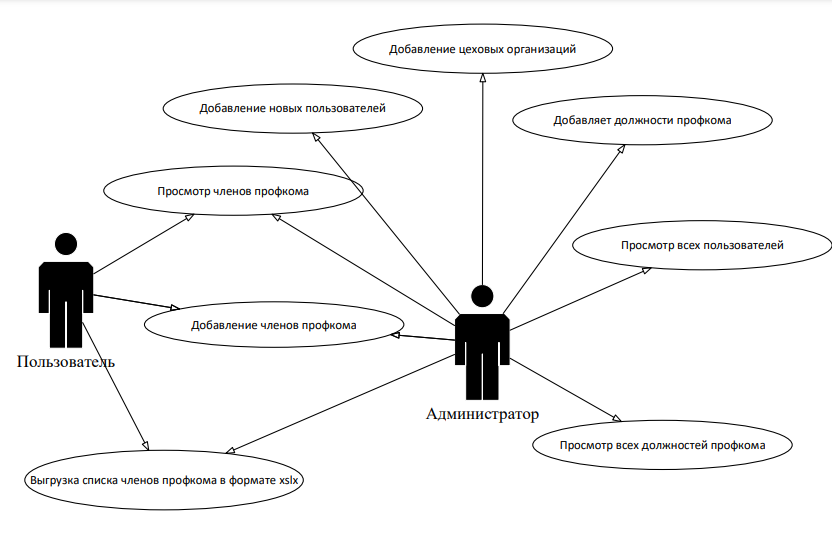


Рисунок 2.1 – Диаграмма вариантов использования

1. **РАЗРАБОТКА БАЗЫ ПРОФСОЮЗА**
   1. **Выбор средств реализации**

Одним из конкурентных решений для реализации базы данных профсоюза может быть использование различных комбинаций СУБД (систем управления базами данных) и ЯП (языков программирования). В данном случае, рассмотрим выбор следующих технологий: *ASP.NET*, *MS SQL* *Server* и фреймворка *ADO.NET.*

1. *ASP.NET:*

- Обоснование выбора: *ASP.NET* является платформой разработки веб-приложений, разработанной *Microsoft*. Она обладает мощными инструментами и возможностями для создания веб-приложений, включая обработку запросов, управление сеансами, создание пользовательского интерфейса и многое другое. *ASP.NET* также предоставляет механизмы для разделения логики приложения и представления данных, что облегчает разработку и поддержку проекта.

- Преимущества:

- Богатый функционал для разработки веб-приложений.

- Поддержка различных языков программирования, включая *C*#.

- Интеграция с другими инструментами и технологиями от Microsoft.

- Недостатки:

- Требуется определенный уровень знаний и опыта для работы с *ASP.NET.*

2. *MS SQL Server*:

- Обоснование выбора: *MS SQL Server* является одной из наиболее популярных и надежных СУБД от Microsoft. Она предлагает широкий набор функций для управления базами данных, обеспечивает высокую производительность и поддерживает расширенные возможности работы с данными, такие как триггеры, хранимые процедуры, представления и другие.

- Преимущества:

- Надежность и стабильность работы.

- Высокая производительность при обработке запросов и масштабируемость.

- Широкий набор инструментов для управления и администрирования базы данных.

- Недостатки:

- Лицензирование может быть затратным для крупных проектов.

3. *ADO.NET*:

- Обоснование выбора: *ADO.NET* (*ActiveX Data Objects .NET)* является фреймворком для работы с данными в приложениях .*NET*. Он предоставляет программистам мощные средства для взаимодействия с базами данных, включая возможность подключения к базе данных, выполнение запросов, обработку результатов и управление транзакциями.

- Преимущества:

- Интеграция с друг

ими инструментами и технологиями от *Microsoft*.

- Поддержка различных провайдеров баз данных, включая *MS SQL Server.*

- Удобный и гибкий способ работы с данными.

- Недостатки:

- Требуется некоторое время для изучения и освоения *ADO.NET*.

Общее обоснование выбора *ASP.NET, MS SQL Server* и фреймворка *ADO.NET:*

- Совместное использование этих технологий позволяет создать мощное и гибкое веб-приложение, способное эффективно управлять базой данных профсоюза.

- *ASP.NET* обеспечивает удобную разработку веб-интерфейса, управление сеансами и обработку запросов.

- *MS SQL Server* предоставляет надежную и производительную СУБД для хранения и управления данными профсоюза.

- *ADO.NET* обеспечивает удобную и гибкую работу с данными, включая выполнение запросов и управление транзакциями.

Выбор данных технологий обоснован их функциональностью, поддержкой со стороны Microsoft и широким использованием в различных проектах. Они обеспечат эффективное и надежное функционирование базы данных профсоюза, а также удобную разработку и поддержку приложения.

* 1. **Реализация базы данных**

Сущность "*Members*" (Таблица "*Members*"):

- *MembersID* (*bigint*) - первичный ключ, идентификатор члена профсоюза

- *FirstName* (*varchar*(20)) - имя члена профсоюза

- *SecondName (varchar(*20)) - фамилия члена профсоюза

- *TherdName (varchar*(20)) - отчество члена профсоюза

- *PhoneNumber (varchar*(20)) - номер телефона члена профсоюза

- *Merital (bit*) - статус брака (0 или 1)

- *BirthDate (date)* - дата рождения члена профсоюза

- *Mail (varchar(30))* - электронная почта члена профсоюза

- *Gender (char)* - пол члена профсоюза

- *Citizenship (varchar(40))* - гражданство члена профсоюза

- *Photo (VARBINARY(MAX))* - фотография члена профсоюза

Сущность "*Father*" (Таблица "*Father*"):

- *FatherID (bigint)* - первичный ключ, идентификатор отца

- *FirstName (varchar(20))* - имя отца

- *SecondName (varchar(20))* - фамилия отца

- *TherdName (varchar(20))* - отчество отца

- *PhoneNumber (varchar(20))* - номер телефона отца

- *JobTitle (varchar(20))* - должность отца

- *MemberID (bigint)* - внешний ключ, ссылается на поле *Members.MembersID*

Сущность "*Mother*" (Таблица "*Mother*"):

- *MotherID (bigint)* - первичный ключ, идентификатор матери

- *FirstName (varchar(20))* - имя матери

- *SecondName (varchar(20))* - фамилия матери

- *TherdName (varchar(20))* - отчество матери

- *PhoneNumber (varchar(20))* - номер телефона матери

- *JobTitle (varchar(20))* - должность матери

- *MemberID (bigint)* - внешний ключ, ссылается на поле *Members.MembersID*

Сущность "*Children*" (Таблица "*Children*"):

*- ChildID (bigint*) - первичный ключ, идентификатор ребенка

- *FirstName (varchar(20))* - имя ребенка

- *SecondName (varchar(20))* - фамилия ребенка

- *TherdName (varchar(20))* - отчество ребенка

- *PhoneNumber (varchar(20*)) - номер телефона ребенка

- *MemberID (bigint)* - внешний ключ, ссылается на поле *Members.MembersID*

- *BirthDate (date)* - дата рождения ребенка

Сущность "*PersonalFile*" (Таблица "*PersonalFile*"):

- *FileID (bigint)* - первичный ключ, идентификатор личного дела

- *PersonalFileNumber (varchar(20))* - номер личного дела

- *MemberID (bigint)* - внешний ключ, ссылается на поле *Members.MembersID*

Сущность

"*Registration*" (Таблица "*Registration*"):

- *RegID (bigint*) - первичный ключ, идентификатор регистрации

- *Country (varchar(20))* - страна регистрации

- *Region (varchar(20))* - регион регистрации

- *City (varchar(20))* - город регистрации

- *Street (varchar(20))* - улица регистрации

- *House (varchar(20))* - дом регистрации

- *ApartmentNumber (varchar(20))* - номер квартиры регистрации

- *Postcode (varchar(20))* - почтовый индекс регистрации

- *MemberID (bigint*) - внешний ключ, ссылается на поле *Members.MembersID*

Сущность "*Location*" (Таблица "*Location*"):

- *LocID (bigint)* - первичный ключ, идентификатор местоположения

- *Country (varchar(20))* - страна

- *Region (varchar(20))* - регион

- *City (varchar(20))* - город

- *Street (varchar(20))* - улица

- *House (varchar(20))* - дом

- *ApartmentNumber (varchar(20))* - номер квартиры

- *Postcode (varchar(20))* - почтовый индекс

- *MemberID (bigint)* - внешний ключ, ссылается на поле *Members.MembersID*

Сущность "*Post*" (Таблица "*Post*"):

- *PostID (bigint*) - первичный ключ, идентификатор должности

- *DepartmentName (varchar(40))* - название отдела

- *Post (varchar(20))* - должность

- *MemberID (bigint)* - внешний ключ, ссылается на поле *Members.MembersID*

Сущность "*Education*" (Таблица "*Education*"):

- *EducID* (*bigint*) - первичный ключ, идентификатор образования

- *EducationalInstitution* *(varchar(30))* - учебное заведение

- *Speciality (varchar(30))* - специальность

- *EndDate (date)* - дата окончания образования

- *Degree (varchar(20))* - степень образования

- *MemberID (bigint)* - внешний ключ, ссылается на поле *Members.MembersID*

Сущность "*Departments*" (Таблица "*Departments*"):

- *DepID (int)* - первичный ключ, идентификатор отдела

- *Name (varchar(20))* - название отдела

Сущность "*ProfkPost*" (Таблица "*ProfkPost*"):

- *PostID (int*) - первичный ключ, идентификатор профком-должности

- *Name (varchar(20))* - название профком-должности

- *PurposeDate (date)* - дата назначения на должность

Сущность "*Memberships*" (Таблица "*Memberships*"):

- *MembershipsID (bigint)* - первичный ключ, идентификатор членства

- *EntryDate (date)* - дата вступления

- *DepID (int)* - внешний ключ, ссылается на поле *Departments.DepID*

- *PostID (int)* - внешний ключ, ссылается на поле *ProfkPost.PostID*

- DueDate (date) - дата окончания членства

- *MemberID (bigint)* - внешний ключ, ссылается на поле *Members.MembersID*

Сущность "*Deserts*" (Таблица "*Deserts*"):

- *DesID (bigint*) - первичный ключ, идентификатор награды

- *TypeDes (varchar(30))* - тип награды

- *GetsDate (date)* - дата получения награды

- *Note (varchar(40))* - примечание

- *MembershipsID (bigint)* - внешний ключ, ссылается на поле *Memberships.MembershipsID*

Сущность "*Penalty*" (Таблица "*Penalty*"):

- *PenaltyID (bigint*) - первичный ключ, идентификатор штрафа

- *TypeDes (varchar(30))* - тип штрафа

- *GetsDate (date)* - дата получения штрафа

- *Note (varchar(40))* - примечание

- *MembershipsID (bigint*) - внешний ключ, ссылается на поле *Memberships.MembershipsID*

В конечном итоге база данных будет иметь следующий вид (см. рис.3.2.1).

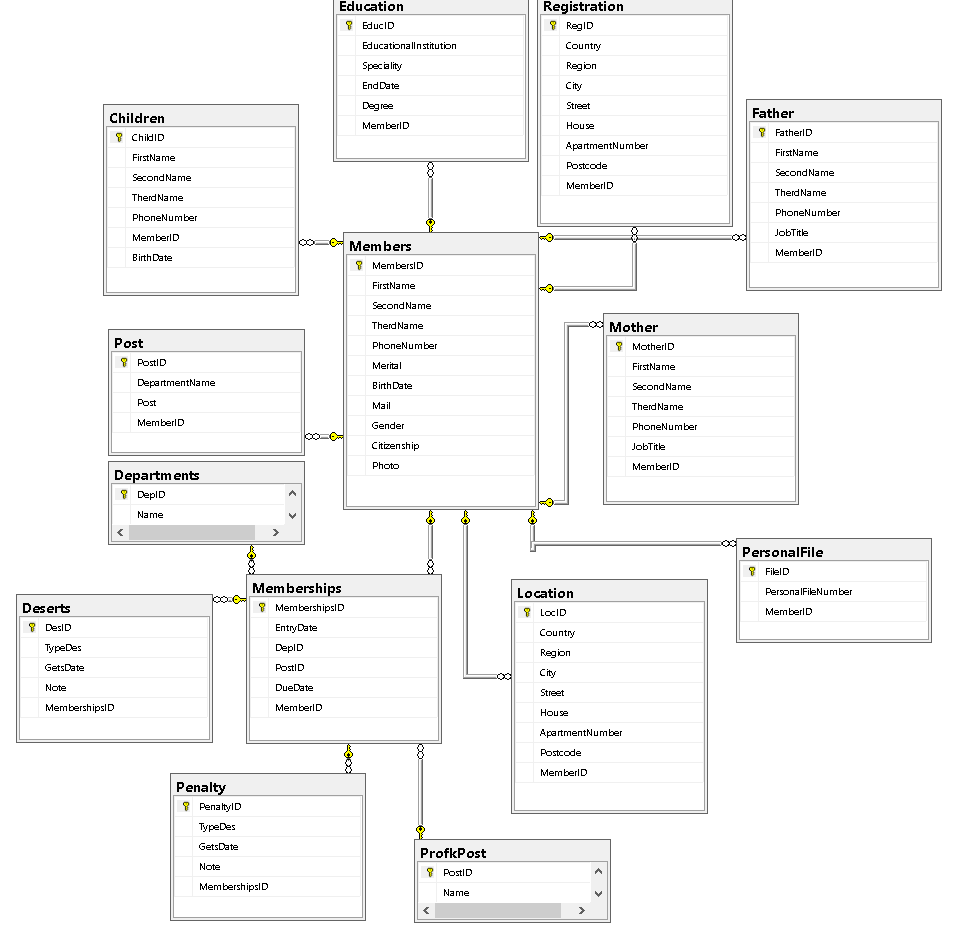


Рисунок 3.2.1 – физичесхая схема базы данных.

* 1. Программная реализация приложения

*private static void CreateDB()*

*{*

*string connectionString2 = "Server=(localdb)\\MSSQLLocalDB;Database=master;Trusted\_Connection=True;";*

*SqlConnection connection = new SqlConnection(connectionString2);*

*connection.Open();*

*string createNewDBCommand = "CREATE DATABASE Profkom";*

*string useDB = "use Profkom";*

*string createTableCommand = createDBString;*

*SqlCommand command = new SqlCommand(createNewDBCommand, connection);*

*command.ExecuteNonQuery();*

*command = new SqlCommand(useDB, connection);*

*command.ExecuteNonQuery();*

*command = new SqlCommand(createTableCommand, connection);*

*command.ExecuteNonQuery();*

*}*

Сначала, устанавливается строка подключения *connectionString2*,

указывающая на сервер базы данных *(localdb)\MSSQLLocalDB* и использующая аутентификацию с доверенным подключением.

Затем, создается новый объект *SqlConnection* с использованием указанной строки подключения.

Подключение к серверу базы данных открывается вызовом connection.Open().

Выполняется *SQL*-запрос *CREATE DATABASE* Profkom, который создает новую базу данных с названием "Profkom" на сервере базы данных.

Следующий *SQL*-запрос *USE Profkom* устанавливает контекст использования только что созданной базы данных "*Profkom*".

После этого, переменная createTableCommand содержит *SQL*-запросы для создания таблиц в базе данных "*Profkom*". Однако, в предоставленном коде отсутствует сам запрос.

Создается новый объект *SqlCommand* с *SQL*-запросом *createNewDBCommand* и соединением connection, и выполняется с помощью command.*ExecuteNonQuery*(). Запрос создает новую базу данных "*Profkom*".

Создается новый объект *SqlCommand* с *SQL*-запросом *useDB* и соединением connection, и выполняется с помощью command.*ExecuteNonQuery*(). Запрос устанавливает контекст использования базы данных "*Profkom*".

Создается новый объект *SqlCommand* с *SQL*-запросом createTableCommand и соединением connection, и выполняется с помощью command.*ExecuteNonQuery*(). Запрос создает таблицы в базе данных "*Profkom*".

*public static List<Member> GetFromDBMembers(ref List<Member> members)*

*{*

*SqlConnection connection = new SqlConnection(connectionString);*

*connection.Open();*

*string Command = "select\r\n m.MembersID,\r\n m.FirstName,\r\n m.SecondName,\r\n m.TherdName,\r\n m.PhoneNumber,\r\n m.Merital,\r\n m.BirthDate,\r\n m.Mail,\r\n m.Gender,\r\n m.Citizenship,\r\n m.Photo,\r\n mom.FirstName as momName,\r\n mom.SecondName as mom2Name,\r\n mom.TherdName as mom3Name,\r\n mom.PhoneNumber as momNumber,\r\n mom.JobTitle as momJob,\r\n dad.FirstName as dadName,\r\n dad.SecondName as dad2Name,\r\n dad.TherdName as dad3Name,\r\n dad.PhoneNumber as dadPhone,\r\n dad.JobTitle as dadJob,\r\n pf.PersonalFileNumber as FileNumber,\r\n Post.DepartmentName as DepWork,\r\n Post.Post as PostWork,\r\n loc.Country as locCountry,\r\n loc.Region as locRegion,\r\n loc.City as locCity,\r\n loc.Street as locStreet,\r\n loc.House as locHouse,\r\n loc.ApartmentNumber as locApNum,\r\n loc.Postcode as locPostCode,\r\n reg.Country as regCountry,\r\n reg.Region as regRegion,\r\n reg.City as regCity,\r\n reg.Street as regStreet,\r\n reg.House as regHouse,\r\n reg.ApartmentNumber as regApNum,\r\n reg.Postcode as regPostCode,\r\n edu.EducationalInstitution as Zavedenie,\r\n edu.Speciality as spec,\r\n edu.EndDate as endDateEd,\r\n edu.Degree as degree,\r\n memsh.MembershipsID as memshID,\r\n memsh.EntryDate as etnry,\r\n memsh.DepID as depID,\r\n memsh.PostID as postID,\r\n memsh.DueDate as dueDate\r\nfrom\r\n Members as m\r\n left join Location as loc on loc.MemberID = m.MembersID\r\n left join Registration as reg on reg.MemberID = m.MembersID\r\n left join Education as edu on edu.MemberID = m.MembersID\r\n left join Mother as mom on mom.MemberID = m.MembersID\r\n left join post on post.MemberID = m.MembersID\r\n left join Father as dad on dad.MemberID = m.MembersID\r\n left join PersonalFile as pf on pf.MemberID = m.MembersID\r\n left join Memberships as memsh on memsh.MemberID = m.MembersID";*

*using (SqlCommand command = new SqlCommand(Command, connection))*

*{*

*using (SqlDataReader reader = command.ExecuteReader())*

*{*

*List<Member> membe = new List<Member>();*

*while (reader.Read())*

*{*

*Member mem = new Member();*

*mem.Id = Convert.ToInt32(reader["MembersID"]);*

*mem.FirstName = (string)reader["FirstName"];*

*mem.SecondName = (string)reader["SecondName"];*

*mem.ThirdName = (string)reader["TherdName"];*

*mem.PhoneNumber = (string)reader["PhoneNumber"];*

*//MaritalStatus = bool.Parse((string)reader["Merital"]),*

*mem.BirthDate = (DateTime)reader["BirthDate"];*

*mem.Email = (string)reader["Mail"];*

*//Gender = bool.Parse((string)reader["Gender"]),*

*mem.Citizenship = (string)reader["Citizenship"];*

*mem.Photo = reader["Photo"] != DBNull.Value ? (byte[])reader["Photo"] : null;*

*mem.MaritalStatus = reader["Merital"] != DBNull.Value && Convert.ToBoolean(reader["Merital"]);*

*if (reader["Gender"] != DBNull.Value)*

*{*

*if ((string)reader["Gender"] == "1")*

*mem.Gender = true;*

*else*

*mem.Gender = false;*

*}*

*if (reader["momName"] != DBNull.Value)*

*{*

*mem.Mother.FirstName = (string)reader["momName"];*

*mem.Mother.SecondName = (string)reader["mom2Name"];*

*mem.Mother.ThirdName = (string)reader["mom3Name"];*

*mem.Mother.PhoneNumber = (string)reader["momNumber"];*

*mem.Mother.JobTitle = (string)reader["momJob"];*

*}*

*if (reader["dadName"] != DBNull.Value)*

*{*

*mem.Father.FirstName = (string)reader["dadName"];*

*mem.Father.SecondName = (string)reader["dad2Name"];*

*mem.Father.ThirdName = (string)reader["dad3Name"];*

*mem.Father.PhoneNumber = (string)reader["dadPhone"];*

*mem.Father.JobTitle = (string)reader["dadJob"];*

*}*

*mem.PersonalFiles.NumberPersonalFile = (string)reader["FileNumber"];*

*mem.PostW.Post = (string)reader["PostWork"];*

*mem.PostW.Department = (string)reader["DepWork"];*

*mem.FactAddress.Country = (string)reader["locCountry"];*

*mem.FactAddress.Region = (string)reader["locRegion"];*

*mem.FactAddress.City = (string)reader["locCity"];*

*mem.FactAddress.Street = (string)reader["locStreet"];*

*mem.FactAddress.NumberHouse = (string)reader["locHouse"];*

*mem.FactAddress.ApartmentNumber = (string)reader["locApNum"];*

*mem.FactAddress.Postcode = (string)reader["locPostCode"];*

*mem.DocumetAddress.Country = (string)reader["regCountry"];*

*mem.DocumetAddress.Region = (string)reader["regRegion"];*

*mem.DocumetAddress.City = (string)reader["regCity"];*

*mem.DocumetAddress.Street = (string)reader["regStreet"];*

*mem.DocumetAddress.NumberHouse = (string)reader["regHouse"];*

*mem.DocumetAddress.ApartmentNumber = (string)reader["regApNum"];*

*mem.DocumetAddress.Postcode = (string)reader["regPostCode"];*

*mem.Educations.EducationalInstitution = (string)reader["Zavedenie"];*

*mem.Educations.Speciality = (string)reader["spec"];*

*var year = (DateTime)reader["endDateEd"];*

*mem.Educations.EndDate = (int)year.Year;*

*mem.Educations.Degree = (string)reader["degree"];*

*mem.Memberships.Id = Convert.ToInt32(reader["memshID"]);*

*mem.Memberships.DateJoin = (DateTime)reader["etnry"];*

*mem.Memberships.OrganizationID = (int)reader["depID"];*

*mem.Memberships.PostOrgID = (int)reader["postID"];*

*mem.Memberships.LastPayment = (DateTime)reader["dueDate"];*

*membe.Add(mem);*

*}*

*reader.Close();*

*foreach (var mem in membe)*

*{*

*SqlCommand command2 = new SqlCommand("select \r\n\tFirstName,\r\n\tSecondName,\r\n\tTherdName,\r\n\tPhoneNumber,\r\n\tBirthDate\r\nfrom Children\r\nwhere MemberID = @memID", connection);*

*command2.Parameters.Add("@memID", System.Data.SqlDbType.Int).Value = mem.Id;*

*using (SqlDataReader reader2 = command2.ExecuteReader())*

*{*

*while (reader2.Read())*

*{*

*Children children = new Children*

*{*

*FirstName = (string)reader["FirstName"],*

*SecondName = (string)reader["SecondName"],*

*ThirdName = (string)reader["TherdName"],*

*PhoneNumber = (string)reader["PhoneNumber"],*

*BirthDate = (DateTime)reader["BirthDate"]*

*};*

*mem.Childrens.Add(children);*

*}*

*reader2.Close();*

*}*

*}*

*foreach (var mem in membe)*

*{*

*SqlCommand command2 = new SqlCommand("select\r\n\tTypeDes,\r\n\tGetsDate,\r\n\tNote\r\nfrom Deserts\r\nwhere MembershipsID = @memshID", connection);*

*command2.Parameters.Add("@memshID", System.Data.SqlDbType.Int).Value = mem.Memberships.Id;*

*using (SqlDataReader reader2 = command2.ExecuteReader())*

*{*

*while (reader2.Read())*

*{*

*Deserts des = new Deserts*

*{*

*DesertTypes = (string)reader2["TypeDes"],*

*Date = (DateTime)reader2["GetsDate"],*

*Note = (string)reader2["Note"]*

*};*

*mem.Memberships.Deserts.Add(des);*

*}*

*reader2.Close();*

*}*

*}*

*foreach (var mem in membe)*

*{*

*SqlCommand command2 = new SqlCommand("select\r\n\tTypeDes,\r\n\tGetsDate,\r\n\tNote\r\nfrom Penalty\r\nwhere MembershipsID = @memshID", connection);*

*command2.Parameters.Add("@memshID", System.Data.SqlDbType.Int).Value = mem.Memberships.Id;*

*using (SqlDataReader reader2 = command2.ExecuteReader())*

*{*

*while (reader2.Read())*

*{*

*Penalty pen = new Penalty*

*{*

*DesertTypes = (string)reader2["TypeDes"],*

*Date = (DateTime)reader2["GetsDate"],*

*Note = (string)reader2["Note"]*

*};*

*mem.Memberships.Penaltys.Add(pen);*

*}*

*reader2.Close();*

*}*

*members.Add(mem);*

*}*

*}*

*connection.Close();*

*}*

*return members;*

*}*

Функция `*GetFromDBMembers*` получает данные о членах из базы данных и заполняет объекты класса `*Member*`. Основные шаги, выполняемые функцией:

1. Создается новое подключение к базе данных с использованием строки подключения `connectionString`.

2. Выполняется *SQL*-запрос, который объединяет несколько *таблиц (`Members`, `Location`, `Registration`, `Education`, `Mother`, `Post`, `Father`, `PersonalFile`, `Memberships`)* с помощью операторов `*LEFT JOIN*` для получения полной информации о членах.

3. Создается объект `*SqlCommand*` с указанным *SQL*-запросом и соединением, а затем выполняется с помощью метода `*ExecuteReader*()`.

4. Создается список объектов `*Member*` для хранения полученных данных.

5. В цикле по каждой записи результата запроса выполняется следующее:

- Создается новый объект `*Member*`.

- Извлекаются значения полей из текущей записи результата запроса и устанавливаются соответствующим свойствам объекта `*Member*`.

- Проверяются значения полей, связанных с родителями, и при наличии данных устанавливаются свойства объектов `*Mother*` и `*Father*`.

- Извлекается информация о адресе проживания и документальном адресе, а также об образовании, и устанавливаются соответствующие свойства объекта `*Member*`.

- Извлекается информация о членстве и добавляется в свойство `*Memberships*` объекта `*Member*`.

- Для каждого члена выполняется запрос для получения информации о детях, и данные о детях добавляются в свойство `*Childrens*` объекта `*Member*`.

- Для каждого члена выполняется запрос для получения информации о наградах, и данные о наградах добавляются в свойство `*Deserts*` объекта `*Memberships*`.

- Для каждого члена выполняется запрос для получения информации о нарушениях, и данные о нарушениях добавляются в свойство `*Penaltys*` объекта `*Memberships*`.

- Заполненный объект `*Member*` добавляется в список `*membe*`.

6. Закрывается объект `*SqlDataReader*`.

7. Выполняется дополнительный цикл, в котором для каждого объекта `*Member*` в списке `*membe*` выполняются дополнительные запросы для получения информации о детях, наградах и нарушениях, и эти данные добавляются соответствующим свойствам объекта `*Member*`.

8. Закрывается соединение с базой данных.

9. Возвращается список `*members*`, содержащий заполненные объекты `Member

* 1. **Инструкция пользователя**

При входе вы видите интерфейс авторизации (см. рис. 3.4.1) на котором вы должны ввести логин и пароль, в случае, если у вас нет их попросите у человека с правами администратора добавить учётную запись для вас.

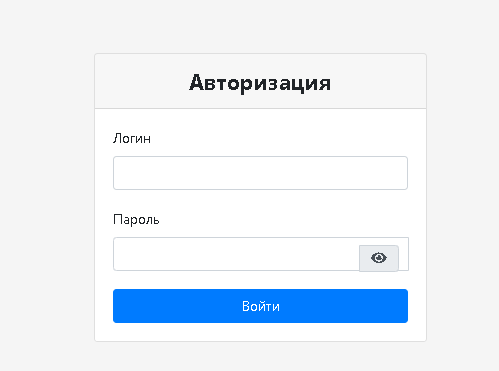


Рисунок 3.4.1 – Интерфейс авторизации пользователя.

После успешной авторизации вы можете увидеть меню из меню выбора действийя (см. рис. 3.4.2) в котором нужно выбрать что-то в зависимости от того, что вы хотите сделать.

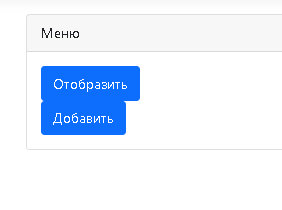


Рисунок 3.4.2 – Интерфейс меню пользователя.

При выборе пункта отобразить перед нами появляется интерфейс отображения (см. рис 3.4.3) членов профсоюза, в котором предусмотрен поиск, и сортировка. Так же есть возможность выгрузить данные в формате XLSX.

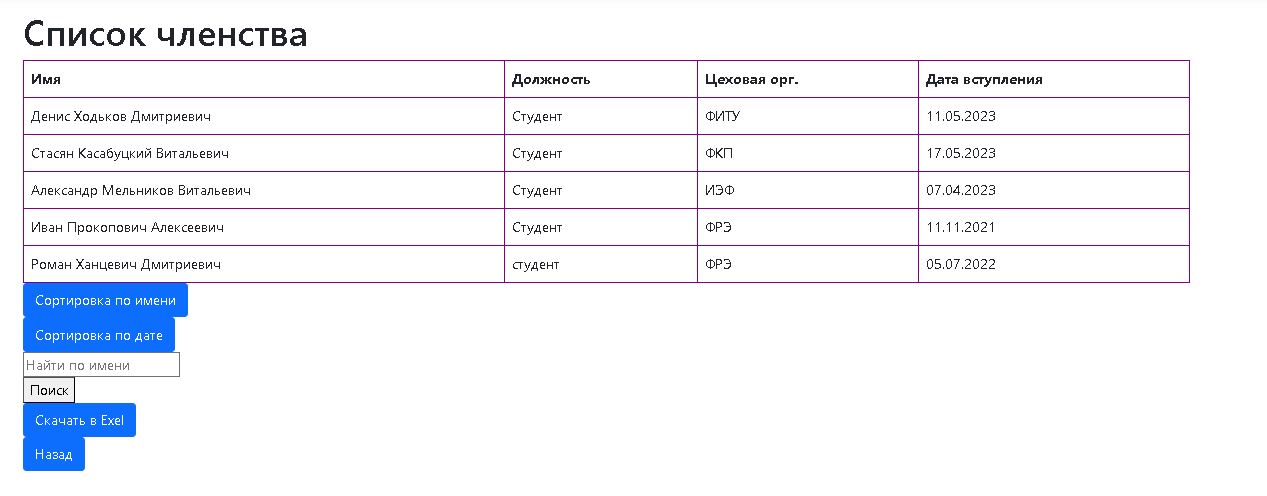


Рисунок 3.4.3 – Интерфейс отображения членов профсоюза.

Если выбрать добавление то открывается интерфейс добавления нового члена профсоюза в базу данных (см. рис 3.4.4) , на форме после заполнения всех данных кнопкой сохраняется новый член в базу данных.

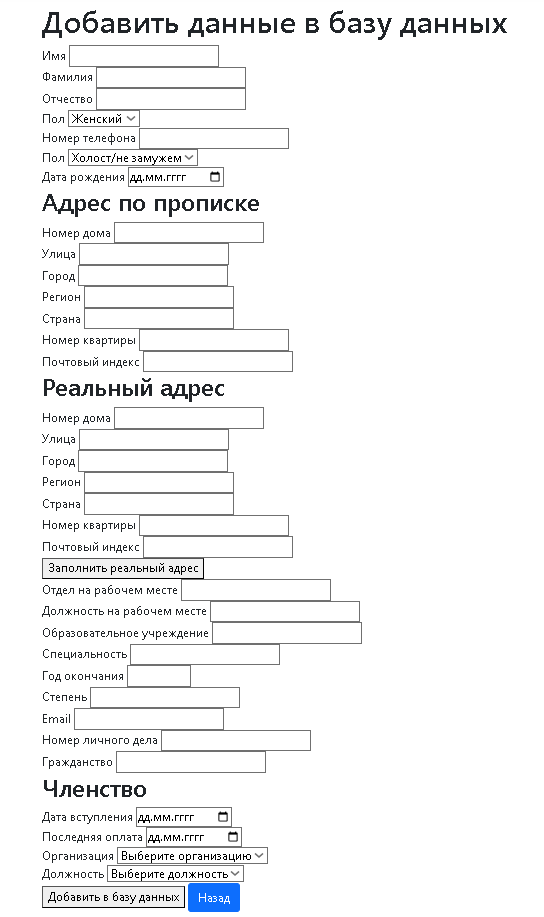


Рисунок 3.4.4 – Интерфейс добавления нового члена профсоюза.

Если авторизоваться с правами админа, то появляется ещё 3 дополнительных возможности (см.рис.3.4.5).

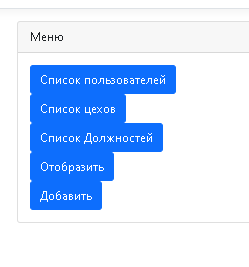


Рисунок 3.4.5 – Интерфейс выбора действия с правами Администратора.

При выборе меню “список пользователей” появляется интерфейс отображения всех пользователей и добавления новых (см. рис. 3.4.6).



Рисунок 3.4.6 – Интерфейс отображения пользователей.

При выборе в меню пункта “список цехов” у нас появляется возможность просмотреть все цеха и добавить новые, аналогично и с должностями.

# **ЗАКЛЮЧЕНИЕ**

В рамках данного курсового проекта была разработана база данных для профсоюза. Целью проекта было создание эффективной и надежной системы управления информацией о членах профсоюза, их персональных данных, членских взносах, наградах и нарушениях.

Для реализации базы данных была выбрана платформа *ASP.NET*, язык программирования *C#* и *СУБД MS SQL Server.* Фреймворк *ADO.NET* использовался для взаимодействия с базой данных. Этот выбор был обоснован широкими возможностями и инструментами, предоставляемыми этими технологиями, а также их надежностью и распространенностью.

Была разработана физическая схема базы данных, которая включает таблицы для хранения информации о членах профсоюза, их личных данных, родителях, адресах проживания и документальных адресах, образовании, членских взносах, наградах и нарушениях. Каждая таблица имеет соответствующие поля с определенными типами данных и допустимыми значениями.

Также были разработаны функции для создания базы данных, заполнения ее тестовыми данными и извлечения информации из базы данных. Функция получения информации о членах из базы данных производит сложные запросы, объединяющие несколько таблиц для получения полной информации о каждом члене профсоюза, а также связанные данные о детях, наградах и нарушениях.

Разработанная база данных позволяет эффективно управлять информацией о членах профсоюза, облегчает поиск и анализ данных, а также предоставляет удобные инструменты для внесения, обновления и удаления информации.

В результате выполнения данного курсового проекта была создана функциональная и надежная база данных для профсоюза, которая может быть использована в реальной среде. Однако, для обеспечения безопасности и оптимизации производительности базы данных, рекомендуется дальнейшая работа по реализации механизмов резервного копирования данных, защиты от несанкционированного доступа и оптимизации запросов.

В целом, данный курсовой проект позволил ознакомиться

с основами проектирования баз данных, использованием современных технологий для разработки баз данных и их взаимодействия, а также приобрести практические навыки в области управления информацией.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

* + 1. Б. А. Залесский, О. В. Хоменко. Базы данных: учебное пособие. - Москва: Горячая линия-Телеком, 2018.
    2. К. Дейт. Введение в системы баз данных. - Москва: Вильямс, 2017.
    3. Х. Гаррисон. SQL. Руководство по изучению языка. - Санкт-Петербург: Питер, 2019.
    4. М. Граслин. Реляционные базы данных. - Москва: ДМК Пресс, 2017.
    5. Д. Колл. Изучаем SQL. - Москва: ДМК Пресс, 2019.
    6. М. Херштейн. Проектирование баз данных. - Москва: ДМК Пресс, 2018.
    7. Д. Вукович. SQL и реляционная теория для программистов. - Москва: ДМК Пресс, 2020.
    8. C. J. Date. An Introduction to Database Systems. - Boston: Addison-Wesley Professional, 2019.
    9. A. Silberschatz, H. F. Korth, S. Sudarshan. Database System Concepts. - New York: McGraw-Hill Education, 2019.
    10. R. Elmasri, S. B. Navathe. Fundamentals of Database Systems. - Boston: Pearson, 2016.

**ПРИЛОЖЕНИЕ А**

**(справочное)**

**Выдержки из кода**

Класс для работы с БД.

*using Kursach.BL.Model;*

*using System;*

*using System.Collections.Generic;*

*using System.Data.SqlClient;*

*using System.Diagnostics.Metrics;*

*using System.Linq.Expressions;*

*using System.ComponentModel;*

*using System.Configuration;*

*using System.Data;*

*namespace Kursach.BL.Controller*

*{*

*/// <summary>*

*/// Класс для работы с ms сервером.*

*/// </summary>*

*public class WorkWithDB*

*{*

*/// <summary>*

*/// строка подключения к бд.*

*/// </summary>*

*public static string connectionString = "Server=(localdb)\\MSSQLLocalDB;Database=Profkom;Trusted\_Connection=True;";*

*/// <summary>*

*/// Создание базы данных с таблицами.*

*/// </summary>*

*/// <returns></returns>*

*private static void CreateDB()*

*{*

*string connectionString2 = "Server=(localdb)\\MSSQLLocalDB;Database=master;Trusted\_Connection=True;";*

*SqlConnection connection = new SqlConnection(connectionString2);*

*connection.Open();*

*string createNewDBCommand = "CREATE DATABASE Profkom";*

*string useDB = "use Profkom";*

*string createTableCommand = "create table Users(\r\nLogin NVarChar(255) primary key,\r\nPassword NVarChar(255),\r\nEmail NVarChar(255),\r\nAccess int,\r\nDepartment NVarChar(255),\r\nPost NVarChar(255),\r\nName NVarChar(255)\r\n)\r\n\r\n\r\ninsert Users(Login, Password, Email, Access, Department, Post, Name)\r\nvalues ('Admin','020604','den.hodkov@gmail.com',0, 'Университет', 'Председатель', 'Денис')\r\n create table Members(\r\nMembersID bigint primary key identity(1,1),\r\nFirstName NVarChar(20),\r\nSecondName NVarChar(20),\r\nTherdName NVarChar(20),\r\nPhoneNumber NVarChar(20),\r\nMerital bit,\r\nBirthDate date,\r\nMail NVarChar(30),\r\nGender char,\r\nCitizenship NVarChar(40),\r\nPhoto VARBINARY(MAX),\r\n)\r\n\r\ncreate table Father (\r\nFatherID bigint primary key identity(1,1),\r\nFirstName NVarChar(20),\r\nSecondName NVarChar(20),\r\nTherdName NVarChar(20),\r\nPhoneNumber NVarChar(20),\r\nJobTitle NVarChar(20),\r\nMemberID bigint not null,\r\nFOREIGN KEY (MemberID) REFERENCES Members(MembersID)\r\n)\r\n\r\ncreate table Mother (\r\nMotherID bigint primary key identity(1,1),\r\nFirstName NVarChar(20),\r\nSecondName NVarChar(20),\r\nTherdName NVarChar(20),\r\nPhoneNumber NVarChar(20),\r\nJobTitle NVarChar(20),\r\nMemberID bigint not null,\r\nFOREIGN KEY (MemberID) REFERENCES Members(MembersID)\r\n)\r\n\r\ncreate table Children (\r\nChildID bigint primary key identity(1,1),\r\nFirstName NVarChar(20),\r\nSecondName NVarChar(20),\r\nTherdName NVarChar(20),\r\nPhoneNumber NVarChar(20),\r\nMemberID bigint not null,\r\nBirthDate date,\r\nFOREIGN KEY (MemberID) REFERENCES Members(MembersID)\r\n)\r\n\r\n\r\ncreate table PersonalFile (\r\nFileID bigint primary key identity(1,1),\r\nPersonalFileNumber NVarChar(20),\r\nMemberID bigint not null,\r\nFOREIGN KEY (MemberID) REFERENCES Members(MembersID)\r\n)\r\n\r\ncreate table Registration (\r\nRegID bigint primary key identity(1,1),\r\nCountry NVarChar(20),\r\nRegion NVarChar(20),\r\nCity NVarChar(20),\r\nStreet NVarChar(20),\r\nHouse NVarChar(20),\r\nApartmentNumber NVarChar(20),\r\nPostcode NVarChar(20),\r\nMemberID bigint not null,\r\nFOREIGN KEY (MemberID) REFERENCES Members(MembersID)\r\n)\r\n\r\ncreate table Location (\r\nLocID bigint primary key identity(1,1),\r\nCountry NVarChar(20),\r\nRegion NVarChar(20),\r\nCity NVarChar(20),\r\nStreet NVarChar(20),\r\nHouse NVarChar(20),\r\nApartmentNumber NVarChar(20),\r\nPostcode NVarChar(20),\r\nMemberID bigint not null,\r\nFOREIGN KEY (MemberID) REFERENCES Members(MembersID)\r\n)\r\n\r\n\r\ncreate table Post (\r\nPostID bigint primary key identity(1,1),\r\nDepartmentName NVarChar(40),\r\nPost NVarChar(20),\r\nMemberID bigint not null,\r\nFOREIGN KEY (MemberID) REFERENCES Members(MembersID)\r\n)\r\n\r\ncreate table Education (\r\nEducID bigint primary key identity(1,1),\r\nEducationalInstitution NVarChar(30),\r\nSpeciality NVarChar(30),\r\nEndDate date,\r\nDegree NVarChar(20),\r\nMemberID bigint not null,\r\nFOREIGN KEY (MemberID) REFERENCES Members(MembersID)\r\n)\r\n\r\ncreate table Departments (\r\nDepID int primary key identity(1,1),\r\nName NVarChar(20)\r\n)\r\n\r\ncreate table ProfkPost (\r\nPostID int primary key identity(1,1),\r\nName NVarChar(20))\r\n\r\ncreate table Memberships (\r\nMembershipsID bigint primary key identity(1,1),\r\nEntryDate date default getdate(),\r\nDepID int,\r\nPostID int,\r\nDueDate date default getdate(),\r\nMemberID bigint not null,\r\nFOREIGN KEY (MemberID) REFERENCES Members(MembersID),\r\nFOREIGN KEY (DepID) REFERENCES Departments(DepID),\r\nFOREIGN KEY (PostID) REFERENCES ProfkPost(PostID)\r\n)\r\n\r\ncreate table Deserts (\r\nDesID bigint primary key identity(1,1),\r\nTypeDes NVarChar(30),\r\nGetsDate date,\r\nNote NVarChar(40),\r\nMembershipsID bigint not null,\r\nFOREIGN KEY (MembershipsID) REFERENCES Memberships(MembershipsID),\r\n)\r\n\r\ncreate table Penalty (\r\nPenaltyID bigint primary key identity(1,1),\r\nTypeDes NVarChar(30),\r\nGetsDate date,\r\nNote NVarChar(40),\r\nMembershipsID bigint not null,\r\nFOREIGN KEY (MembershipsID) REFERENCES Memberships(MembershipsID),\r\n)";*

*SqlCommand command = new SqlCommand(createNewDBCommand, connection);*

*command.ExecuteNonQuery();*

*command = new SqlCommand(useDB, connection);*

*command.ExecuteNonQuery();*

*command = new SqlCommand(createTableCommand, connection);*

*command.ExecuteNonQuery();*

*}*

*/// <summary>*

*/// проверка существования нужной дб.*

*/// </summary>*

*/// <returns>Если она существует вернёт true, если она не существует вернёт false.</returns>*

*public static bool CheckDB()*

*{*

*using (SqlConnection connection = new SqlConnection(connectionString))*

*{*

*try*

*{*

*connection.Open();*

*// Если соединение установлено успешно, то база данных существует*

*return true;*

*}*

*catch (SqlException)*

*{*

*// Если возникла ошибка, значит база данных не существует*

*CreateDB();*

*return false;*

*}*

*}*

*}*

*public static List<Member> GetFromDBMembers(ref List<Member> members)*

*{*

*SqlConnection connection = new SqlConnection(connectionString);*

*connection.Open();*

*string Command = "select\r\n m.MembersID,\r\n m.FirstName,\r\n m.SecondName,\r\n m.TherdName,\r\n m.PhoneNumber,\r\n m.Merital,\r\n m.BirthDate,\r\n m.Mail,\r\n m.Gender,\r\n m.Citizenship,\r\n m.Photo,\r\n mom.FirstName as momName,\r\n mom.SecondName as mom2Name,\r\n mom.TherdName as mom3Name,\r\n mom.PhoneNumber as momNumber,\r\n mom.JobTitle as momJob,\r\n dad.FirstName as dadName,\r\n dad.SecondName as dad2Name,\r\n dad.TherdName as dad3Name,\r\n dad.PhoneNumber as dadPhone,\r\n dad.JobTitle as dadJob,\r\n pf.PersonalFileNumber as FileNumber,\r\n Post.DepartmentName as DepWork,\r\n Post.Post as PostWork,\r\n loc.Country as locCountry,\r\n loc.Region as locRegion,\r\n loc.City as locCity,\r\n loc.Street as locStreet,\r\n loc.House as locHouse,\r\n loc.ApartmentNumber as locApNum,\r\n loc.Postcode as locPostCode,\r\n reg.Country as regCountry,\r\n reg.Region as regRegion,\r\n reg.City as regCity,\r\n reg.Street as regStreet,\r\n reg.House as regHouse,\r\n reg.ApartmentNumber as regApNum,\r\n reg.Postcode as regPostCode,\r\n edu.EducationalInstitution as Zavedenie,\r\n edu.Speciality as spec,\r\n edu.EndDate as endDateEd,\r\n edu.Degree as degree,\r\n memsh.MembershipsID as memshID,\r\n memsh.EntryDate as etnry,\r\n memsh.DepID as depID,\r\n memsh.PostID as postID,\r\n memsh.DueDate as dueDate\r\nfrom\r\n Members as m\r\n left join Location as loc on loc.MemberID = m.MembersID\r\n left join Registration as reg on reg.MemberID = m.MembersID\r\n left join Education as edu on edu.MemberID = m.MembersID\r\n left join Mother as mom on mom.MemberID = m.MembersID\r\n left join post on post.MemberID = m.MembersID\r\n left join Father as dad on dad.MemberID = m.MembersID\r\n left join PersonalFile as pf on pf.MemberID = m.MembersID\r\n left join Memberships as memsh on memsh.MemberID = m.MembersID";*

*using (SqlCommand command = new SqlCommand(Command, connection))*

*{*

*using (SqlDataReader reader = command.ExecuteReader())*

*{*

*List<Member> membe = new List<Member>();*

*while (reader.Read())*

*{*

*Member mem = new Member();*

*mem.Id = Convert.ToInt32(reader["MembersID"]);*

*mem.FirstName = (string)reader["FirstName"];*

*mem.SecondName = (string)reader["SecondName"];*

*mem.ThirdName = (string)reader["TherdName"];*

*mem.PhoneNumber = (string)reader["PhoneNumber"];*

*//MaritalStatus = bool.Parse((string)reader["Merital"]),*

*mem.BirthDate = (DateTime)reader["BirthDate"];*

*mem.Email = (string)reader["Mail"];*

*//Gender = bool.Parse((string)reader["Gender"]),*

*mem.Citizenship = (string)reader["Citizenship"];*

*mem.Photo = reader["Photo"] != DBNull.Value ? (byte[])reader["Photo"] : null;*

*mem.MaritalStatus = reader["Merital"] != DBNull.Value && Convert.ToBoolean(reader["Merital"]);*

*if (reader["Gender"] != DBNull.Value)*

*{*

*if ((string)reader["Gender"] == "1")*

*mem.Gender = true;*

*else*

*mem.Gender = false;*

*}*

*if (reader["momName"] != DBNull.Value)*

*{*

*mem.Mother.FirstName = (string)reader["momName"];*

*mem.Mother.SecondName = (string)reader["mom2Name"];*

*mem.Mother.ThirdName = (string)reader["mom3Name"];*

*mem.Mother.PhoneNumber = (string)reader["momNumber"];*

*mem.Mother.JobTitle = (string)reader["momJob"];*

*}*

*if (reader["dadName"] != DBNull.Value)*

*{*

*mem.Father.FirstName = (string)reader["dadName"];*

*mem.Father.SecondName = (string)reader["dad2Name"];*

*mem.Father.ThirdName = (string)reader["dad3Name"];*

*mem.Father.PhoneNumber = (string)reader["dadPhone"];*

*mem.Father.JobTitle = (string)reader["dadJob"];*

*}*

*mem.PersonalFiles.NumberPersonalFile = (string)reader["FileNumber"];*

*mem.PostW.Post = (string)reader["PostWork"];*

*mem.PostW.Department = (string)reader["DepWork"];*

*mem.FactAddress.Country = (string)reader["locCountry"];*

*mem.FactAddress.Region = (string)reader["locRegion"];*

*mem.FactAddress.City = (string)reader["locCity"];*

*mem.FactAddress.Street = (string)reader["locStreet"];*

*mem.FactAddress.NumberHouse = (string)reader["locHouse"];*

*mem.FactAddress.ApartmentNumber = (string)reader["locApNum"];*

*mem.FactAddress.Postcode = (string)reader["locPostCode"];*

*mem.DocumetAddress.Country = (string)reader["regCountry"];*

*mem.DocumetAddress.Region = (string)reader["regRegion"];*

*mem.DocumetAddress.City = (string)reader["regCity"];*

*mem.DocumetAddress.Street = (string)reader["regStreet"];*

*mem.DocumetAddress.NumberHouse = (string)reader["regHouse"];*

*mem.DocumetAddress.ApartmentNumber = (string)reader["regApNum"];*

*mem.DocumetAddress.Postcode = (string)reader["regPostCode"];*

*mem.Educations.EducationalInstitution = (string)reader["Zavedenie"];*

*mem.Educations.Speciality = (string)reader["spec"];*

*var year = (DateTime)reader["endDateEd"];*

*mem.Educations.EndDate = (int)year.Year;*

*mem.Educations.Degree = (string)reader["degree"];*

*mem.Memberships.Id = Convert.ToInt32(reader["memshID"]);*

*mem.Memberships.DateJoin = (DateTime)reader["etnry"];*

*mem.Memberships.OrganizationID = (int)reader["depID"];*

*mem.Memberships.PostOrgID = (int)reader["postID"];*

*mem.Memberships.LastPayment = (DateTime)reader["dueDate"];*

*membe.Add(mem);*

*}*

*reader.Close();*

*foreach (var mem in membe)*

*{*

*SqlCommand command2 = new SqlCommand("select \r\n\tFirstName,\r\n\tSecondName,\r\n\tTherdName,\r\n\tPhoneNumber,\r\n\tBirthDate\r\nfrom Children\r\nwhere MemberID = @memID", connection);*

*command2.Parameters.Add("@memID", System.Data.SqlDbType.Int).Value = mem.Id;*

*using (SqlDataReader reader2 = command2.ExecuteReader())*

*{*

*while (reader2.Read())*

*{*

*Children children = new Children*

*{*

*FirstName = (string)reader["FirstName"],*

*SecondName = (string)reader["SecondName"],*

*ThirdName = (string)reader["TherdName"],*

*PhoneNumber = (string)reader["PhoneNumber"],*

*BirthDate = (DateTime)reader["BirthDate"]*

*};*

*mem.Childrens.Add(children);*

*}*

*reader2.Close();*

*}*

*}*

*foreach (var mem in membe)*

*{*

*SqlCommand command2 = new SqlCommand("select\r\n\tTypeDes,\r\n\tGetsDate,\r\n\tNote\r\nfrom Deserts\r\nwhere MembershipsID = @memshID", connection);*

*command2.Parameters.Add("@memshID", System.Data.SqlDbType.Int).Value = mem.Memberships.Id;*

*using (SqlDataReader reader2 = command2.ExecuteReader())*

*{*

*while (reader2.Read())*

*{*

*Deserts des = new Deserts*

*{*

*DesertTypes = (string)reader2["TypeDes"],*

*Date = (DateTime)reader2["GetsDate"],*

*Note = (string)reader2["Note"]*

*};*

*mem.Memberships.Deserts.Add(des);*

*}*

*reader2.Close();*

*}*

*}*

*foreach (var mem in membe)*

*{*

*SqlCommand command2 = new SqlCommand("select\r\n\tTypeDes,\r\n\tGetsDate,\r\n\tNote\r\nfrom Penalty\r\nwhere MembershipsID = @memshID", connection);*

*command2.Parameters.Add("@memshID", System.Data.SqlDbType.Int).Value = mem.Memberships.Id;*

*using (SqlDataReader reader2 = command2.ExecuteReader())*

*{*

*while (reader2.Read())*

*{*

*Penalty pen = new Penalty*

*{*

*DesertTypes = (string)reader2["TypeDes"],*

*Date = (DateTime)reader2["GetsDate"],*

*Note = (string)reader2["Note"]*

*};*

*mem.Memberships.Penaltys.Add(pen);*

*}*

*reader2.Close();*

*}*

*members.Add(mem);*

*}*

*}*

*connection.Close();*

*}*

*return members;*

*}*

*public static bool AddMember*

*(*

*ref Member member*

*)*

*{*

*using (SqlConnection connection = new SqlConnection(connectionString))*

*{*

*connection.Open();*

*using (SqlTransaction transaction = connection.BeginTransaction())*

*{*

*SqlCommand command = new SqlCommand("insert into Members (\r\nFirstName,\r\nSecondName,\r\nTherdName,\r\nPhoneNumber,\r\nMerital,\r\nBirthDate,\r\nMail,\r\nGender,\r\nCitizenship)\r\nVALUES (\r\n@1name,\r\n@2name,\r\n@3name,\r\n@phNumber,\r\n@Merital,\r\n@BirthDate,\r\n@Mail,\r\n@Gender,\r\n@Citizenship);\r\nselect SCOPE\_IDENTITY();", connection, transaction);*

*command.Parameters.AddWithValue("@1name", member.FirstName);*

*command.Parameters.AddWithValue("@2name", member.SecondName);*

*command.Parameters.AddWithValue("@3name", member.ThirdName);*

*command.Parameters.AddWithValue("@phNumber", member.PhoneNumber);*

*command.Parameters.AddWithValue("@Merital", member.MaritalStatus);*

*command.Parameters.AddWithValue("@BirthDate", member.BirthDate);*

*command.Parameters.AddWithValue("@Mail", member.Email);*

*command.Parameters.AddWithValue("@Gender", member.Gender);*

*command.Parameters.AddWithValue("@Citizenship", member.Citizenship);*

*//command.Parameters.AddWithValue("@Photo", member.Photo);*

*int membersId = Convert.ToInt32(command.ExecuteScalar());*

*if (member.Father != null)*

*{*

*command = new SqlCommand("insert into Father (\r\nFirstName,\r\nSecondName,\r\nTherdName,\r\nPhoneNumber,\r\nJobTitle,\r\nMemberID)\r\nVALUES (\r\n@1name,\r\n@2name,\r\n@3name,\r\n@phNumber,\r\n@JobTitle,\r\n@memID)", connection, transaction);*

*command.Parameters.AddWithValue("@1name", member.Father.FirstName);*

*command.Parameters.AddWithValue("@2name", member.Father.SecondName);*

*command.Parameters.AddWithValue("@3name", member.Father.ThirdName);*

*command.Parameters.AddWithValue("@JobTitle", member.Father.JobTitle);*

*command.Parameters.AddWithValue("@phNumber", member.Father.PhoneNumber);*

*command.Parameters.AddWithValue("@memID", membersId);*

*command.ExecuteNonQuery();*

*}*

*if (member.Mother != null)*

*{*

*command = new SqlCommand("insert into Mother (\r\nFirstName,\r\nSecondName,\r\nTherdName,\r\nPhoneNumber,\r\nJobTitle,\r\nMemberID)\r\nVALUES (\r\n@1name,\r\n@2name,\r\n@3name,\r\n@phNumber,\r\n@JobTitle,\r\n@memID)", connection, transaction);*

*command.Parameters.AddWithValue("@1name", member.Mother.FirstName);*

*command.Parameters.AddWithValue("@2name", member.Mother.SecondName);*

*command.Parameters.AddWithValue("@3name", member.Mother.ThirdName);*

*command.Parameters.AddWithValue("@JobTitle", member.Mother.JobTitle);*

*command.Parameters.AddWithValue("@phNumber", member.Mother.PhoneNumber);*

*command.Parameters.AddWithValue("@memID", membersId);*

*command.ExecuteNonQuery();*

*}*

*command = new SqlCommand("insert into PersonalFile (PersonalFileNumber,MemberID)\r\nvalues (\r\n@pesNum,\r\n@memID)", connection, transaction);*

*command.Parameters.AddWithValue("@pesNum", member.PersonalFiles.NumberPersonalFile);*

*command.Parameters.AddWithValue("@memID", membersId);*

*command.ExecuteNonQuery();*

*command = new SqlCommand("insert into Post (DepartmentName,post,MemberID)\r\nvalues (\r\n@depName,\r\n@Post,\r\n@memID)", connection, transaction);*

*command.Parameters.AddWithValue("@depName", member.PostW.Department);*

*command.Parameters.AddWithValue("@Post", member.PostW.Post);*

*command.Parameters.AddWithValue("@memID", membersId);*

*command.ExecuteNonQuery();*

*command = new SqlCommand("insert into Location (\r\nCountry,\r\nRegion,\r\nCity,\r\nStreet,\r\nHouse,\r\nApartmentNumber,\r\nPostcode,\r\nMemberID)\r\nvalues (\r\n@Country,\r\n@Region,\r\n@City,\r\n@Street,\r\n@House,\r\n@ApartmentNumber,\r\n@Postcode,\r\n@MemberID)", connection, transaction);*

*command.Parameters.AddWithValue("@Country", member.FactAddress.Country);*

*command.Parameters.AddWithValue("@Region", member.FactAddress.Region);*

*command.Parameters.AddWithValue("@City", member.FactAddress.City);*

*command.Parameters.AddWithValue("@Street", member.FactAddress.Street);*

*command.Parameters.AddWithValue("@House", member.FactAddress.NumberHouse);*

*command.Parameters.AddWithValue("@ApartmentNumber", member.FactAddress.ApartmentNumber);*

*command.Parameters.AddWithValue("@Postcode", member.FactAddress.Postcode);*

*command.Parameters.AddWithValue("@MemberID", membersId);*

*command.ExecuteNonQuery();*

*command = new SqlCommand("insert into Registration (\r\nCountry,\r\nRegion,\r\nCity,\r\nStreet,\r\nHouse,\r\nApartmentNumber,\r\nPostcode,\r\nMemberID)\r\nvalues (\r\n@Country,\r\n@Region,\r\n@City,\r\n@Street,\r\n@House,\r\n@ApartmentNumber,\r\n@Postcode,\r\n@MemberID)", connection, transaction);*

*command.Parameters.AddWithValue("@Country", member.DocumetAddress.Country);*

*command.Parameters.AddWithValue("@Region", member.DocumetAddress.Region);*

*command.Parameters.AddWithValue("@City", member.DocumetAddress.City);*

*command.Parameters.AddWithValue("@Street", member.DocumetAddress.Street);*

*command.Parameters.AddWithValue("@House", member.DocumetAddress.NumberHouse);*

*command.Parameters.AddWithValue("@ApartmentNumber", member.DocumetAddress.ApartmentNumber);*

*command.Parameters.AddWithValue("@Postcode", member.DocumetAddress.Postcode);*

*command.Parameters.AddWithValue("@MemberID", membersId);*

*command.ExecuteNonQuery();*

*command = new SqlCommand("insert into Education (\r\nEducationalInstitution,\r\nSpeciality,\r\nEndDate,\r\nDegree,\r\nMemberID)\r\nvalues (\r\n@EducationalInstitution,\r\n@Speciality,\r\n@EndDate,\r\n@Degree,\r\n@MemberID\r\n)", connection, transaction);*

*command.Parameters.AddWithValue("@EducationalInstitution", member.Educations.EducationalInstitution);*

*command.Parameters.AddWithValue("@Speciality", member.Educations.Speciality);*

*command.Parameters.AddWithValue("@EndDate", DateTime.ParseExact(member.Educations.EndDate.ToString(),"yyyy",null));*

*command.Parameters.AddWithValue("@Degree", member.Educations.Degree);*

*command.Parameters.AddWithValue("@MemberID", membersId);*

*command.ExecuteNonQuery();*

*if (member.Childrens != null)*

*{*

*foreach (var child in member.Childrens)*

*{*

*command = new SqlCommand("insert into Children (\r\nFirstName,\r\nSecondName,\r\nTherdName,\r\nPhoneNumber,\r\nMemberID,\r\nBirthDate)\r\nvalues (\r\n@FirstName,\r\n@SecondName,\r\n@TherdName,\r\n@PhoneNumber,\r\n@MemberID,\r\n@BirthDate)", connection, transaction);*

*command.Parameters.AddWithValue("@FirstName", child.FirstName);*

*command.Parameters.AddWithValue("@SecondName", child.SecondName);*

*command.Parameters.AddWithValue("@TherdName", child.ThirdName);*

*command.Parameters.AddWithValue("@PhoneNumber", child.PhoneNumber);*

*command.Parameters.AddWithValue("@BirthDate", child.BirthDate);*

*command.Parameters.AddWithValue("@MemberID", membersId);*

*command.ExecuteNonQuery();*

*}*

*}*

*command = new SqlCommand("insert into Memberships (\r\nEntryDate,\r\nDepID,\r\nPostID,\r\nDueDate,\r\nMemberID)\r\nvalues (\r\n@EntryDate,\r\n@DepID,\r\n@PostID,\r\n@DueDate,\r\n@MemberID)\r\nselect SCOPE\_IDENTITY();", connection, transaction);*

*command.Parameters.AddWithValue("@EntryDate", member.Memberships.DateJoin);*

*command.Parameters.AddWithValue("@DepID", member.Memberships.OrganizationID);*

*command.Parameters.AddWithValue("@PostID", member.Memberships.PostOrgID);*

*command.Parameters.AddWithValue("@DueDate", member.Memberships.LastPayment);*

*command.Parameters.AddWithValue("@MemberID", membersId);*

*int membershipId = Convert.ToInt32(command.ExecuteScalar());*

*member.Memberships.Id = membershipId;*

*transaction.Commit();*

*return true;*

*}*

*}*

*}*

*public static void AddOrganization(ref Organization organization)*

*{*

*using (SqlConnection connection = new SqlConnection(connectionString))*

*{*

*connection.Open();*

*SqlCommand command = new SqlCommand("insert into Departments (Name)\r\nvalues (@name)\r\nselect SCOPE\_IDENTITY();", connection);*

*command.Parameters.Add("@name", SqlDbType.NVarChar).Value = organization.Name;*

*organization.Id = Convert.ToInt32(command.ExecuteScalar());*

*connection.Close();*

*}*

*}*

*public static void AddOrgPost(ref OrgPost orgPost)*

*{*

*using (SqlConnection connection = new SqlConnection(connectionString))*

*{*

*connection.Open();*

*SqlCommand command = new SqlCommand("insert into ProfkPost (Name)\r\nvalues (@name)\r\nselect SCOPE\_IDENTITY();", connection);*

*command.Parameters.Add("@name", SqlDbType.NVarChar).Value = orgPost.Name;*

*orgPost.Id = Convert.ToInt32(command.ExecuteScalar());*

*connection.Close();*

*}*

*}*

*public static bool GetFromDBPostProfk(ref List<OrgPost> orgPosts)*

*{*

*SqlConnection connection = new SqlConnection(connectionString);*

*connection.Open();*

*string Command = "select Name, PostID from ProfkPost;";*

*using (SqlCommand command = new SqlCommand(Command, connection))*

*{*

*using (SqlDataReader reader = command.ExecuteReader())*

*{*

*while (reader.Read())*

*{*

*OrgPost post = new OrgPost()*

*{*

*Name = (string)reader["Name"],*

*Id = (int)reader["PostID"]*

*};*

*orgPosts.Add(post);*

*}*

*reader.Close();*

*}*

*connection.Close();*

*}*

*return true;*

*}*

*public static bool GetFromDBOrganization(ref List<Organization> organizations)*

*{*

*SqlConnection connection = new SqlConnection(connectionString);*

*connection.Open();*

*string Command = "Select Name, DepID from Departments;";*

*using (SqlCommand command = new SqlCommand(Command, connection))*

*{*

*using (SqlDataReader reader = command.ExecuteReader())*

*{*

*while (reader.Read())*

*{*

*Organization org = new Organization*

*{*

*Name = (string)reader["Name"],*

*Id = (int)reader["DepID"]*

*};*

*organizations.Add(org);*

*}*

*reader.Close();*

*}*

*connection.Close();*

*}*

*return true;*

*}*

*public static void AddUser(ref Users user)*

*{*

*using (SqlConnection connection = new SqlConnection(connectionString))*

*{*

*connection.Open();*

*SqlCommand command = new SqlCommand("insert Users(Login, Password, Email, Access, Department, Post, Name)\r\nvalues (@login, @password, @email, @access, @dep, @post, @name)", connection);*

*command.Parameters.Add("@login", SqlDbType.NVarChar).Value = user.Login;*

*command.Parameters.Add("@password", SqlDbType.NVarChar).Value = user.Password;*

*command.Parameters.Add("@email", SqlDbType.NVarChar).Value = user.Email;*

*command.Parameters.Add("@access", SqlDbType.Int).Value = user.Access;*

*command.Parameters.Add("@dep", SqlDbType.NVarChar).Value = user.Department;*

*command.Parameters.Add("@post", SqlDbType.NVarChar).Value = user.Post;*

*command.Parameters.Add("@name", SqlDbType.NVarChar).Value = user.Name;*

*command.ExecuteNonQuery();*

*connection.Close();*

*}*

*}*

*public static bool GetUsersFromDB(ref List<Users> users)*

*{*

*SqlConnection connection = new SqlConnection(connectionString);*

*connection.Open();*

*string Command = "select \* from Users;";*

*using (SqlCommand command = new SqlCommand(Command, connection))*

*{*

*using (SqlDataReader reader = command.ExecuteReader())*

*{*

*while (reader.Read())*

*{*

*Users user = new Users*

*{*

*Login = (string)reader["Login"],*

*Password = (string)reader["Password"],*

*Email = (string)reader["Email"],*

*reader.Close();*

*}*

*connection.Close();*

*}*

**ВЕДОМОСТЬ** **КУРСОВОГО ПРОЕКТА**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | *Дополнительные сведения* | | | | |
|  | | | | | *Текстовые документы* |  | | | | |
|  | | | | |  |  | | | | |
| *БГУИР КП 1-53 01 02 01 006 ПЗ* | | | | | *Пояснительная записка* | *41c.* | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  |  |  |  |  | *БГУИР КП 1-53 01 02 06 005 Д1* | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм.* | *Л* | *№ докум.* | *Подп* | *Дата* | *База данных профсоюза*  *Ведомость курсового проекта* | *Лит* | | | *Лист* | *Листов* |
| *Разраб.* | | *Ходьков Милько* |  |  |  |  |  | *41* | *41* |
| *Провер.* | | *Трофимович* |  |  | *Кафедра ИТАС*  *гр. 020604* | | | | |
| *Т.контр.* | |  |  |  |
| *Н.контр.* | |  |  |  |
| *Утв.* | |  |  |  |
| *Реценз.* | |  |  |  |