**Частное образовательное учреждение высшего образования**

**«Международный Институт Дизайна и Сервиса»**

**(ЧОУВО МИДиС)**

Кафедра математики и информатики

**КУРСОВАЯ РАБОТА**

По дисциплине

«Объектно-ориентированный анализ и программирование на языке С#»

На тему: «Информационная система автосалона»

Направление: 09.03.03 Прикладная Информатика

Выполнил работу:

A.Я. Гарток

Группа ПИ-226

Проверил работу:

С.С. Чеботарев

Челябинск 2023

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc136355698)

[ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc136355699)

[1.1 Описание предметной области 4](#_Toc136355700)

[1.2 Анализ существующих решений 4](#_Toc136355701)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ 11](#_Toc136355703)

[2.1 Обоснование выбора СУБД 11](#_Toc136355704)

[2.2 Разработка инфологической модели 15](#_Toc136355705)

[2.3 Нормализация базы данных 16](#_Toc136355705)

[2.4 Разработка даталогической модели 18](#_Toc136355705)

[2.5 Инициализация базы данных 19](#_Toc136355705)

[2.6 Заполнение таблиц данными 21](#_Toc136355705)

[2.7 Дополнительные инструменты 23](#_Toc136355705)

[ГЛАВА 3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ 25](#_Toc136355703)

[3.1 Обоснование выбора среды разработки 25](#_Toc136355704)

[3.2 Проектирование приложения 27](#_Toc136355704)

[3.3 Разработка приложения 29](#_Toc136355705)

[ЗАКЛЮЧЕНИЕ 36](#_Toc136355703)

СПИСОК ИСПОЛЬЗОВАННЫХ РЕСУРСОВ………………………………...37

# **ВВЕДЕНИЕ**

Информационная система автосалона — это специализированная система, предназначенная для автоматизации и управления различными аспектами деятельности автосалона. Она объединяет в себе программное обеспечение, данные и процессы, необходимые для эффективного функционирования бизнеса. Цель курсовой работы заключается в приобретении навыков проектирования и разработки базы данных, а также управления данными в базе с использованием программного обеспечения.

Объектом исследования курсовой работы является база данных.

Предметом исследования является информационная система военного округа.

Основные задачи работы:

1. Исследование предметной области.
2. Разработка модели системы военного округа.
3. Выбор СУБД и среды программирования.
4. Проектирование и создание базы данных.
5. Проектирование и создание приложения для управления базой

данных.

По завершению программный продукт будет использоваться для работы с различной информацией о военном округе в базе данных.

# **ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Описание предметной области**

Предметная область автосалона включает в себя информацию об автомобилях (уникальный идентификационный номер, марка авто, модель авто, фото, идентификационный номер конфигурации авто)

У каждого автомобиля есть конфигурация его составных частей (уникальный идентификационный номер, тип двигателя, тип колес, цвет авто)

Важной частью предметной области являются клиенты автосалона, автосалон ведет учет пользователей. У каждого клиента существует личный кабинет с информацией о своем профиле, а также с информацией о своем автомобиле.

Клиент автосалона имеет возможность регистрации и авторизации в приложении, а также оформить заказ на понравившийся автомобиль, используя приложение.

Пользователи приложения, имеющие статус «admin», имеют возможность редактировать список пользователей, обрабатывать заказы пользователей, а также редактировать справочные таблицы.

## **1.2 Анализ существующих решений**

Анализ существующих программных решений, которые выполняют функции схожие с системой управления автосалоном.

Перечень схожих решений:

**«MT DMS»** -это комплексная система управления автодилером, формирует единое информационное пространство на предприятии. MT DMS спроектирована, как универсальное решение, «закрывающее» большинство актуальных потребностей автодилеров, как отдельных, так и в составе дилерской группы.

Основные возможности MT DMS включают:

* MT DMS позволяет руководителю предприятия быть в курсе оперативной деятельности предприятия в любой момент времени, благодаря гибко настраиваемым отчетам. Нет необходимости запрашивать информацию от руководителей направлений и терять время, ожидая ответ.
* Перед обновлением есть возможность предварительно оценить нововведения, не затрагивая рабочую базу данных. Посмотреть на новые функции в тестовом режиме.
* Отчёты в MT DMS интерактивны. Это означает, что сотрудник может не просто просмотреть итоговые данные и распечатать их, но и сделать развёрнутую детализацию по конкретным позициям.
* Программисты создали дружественный пользователю конструктор печатных форм для создания отчётов. С ним возможно автоматическое заполнение практически любых полей из таблиц базы данных.
* Существует возможность отправлять сообщения электронной почты в формате HTML.

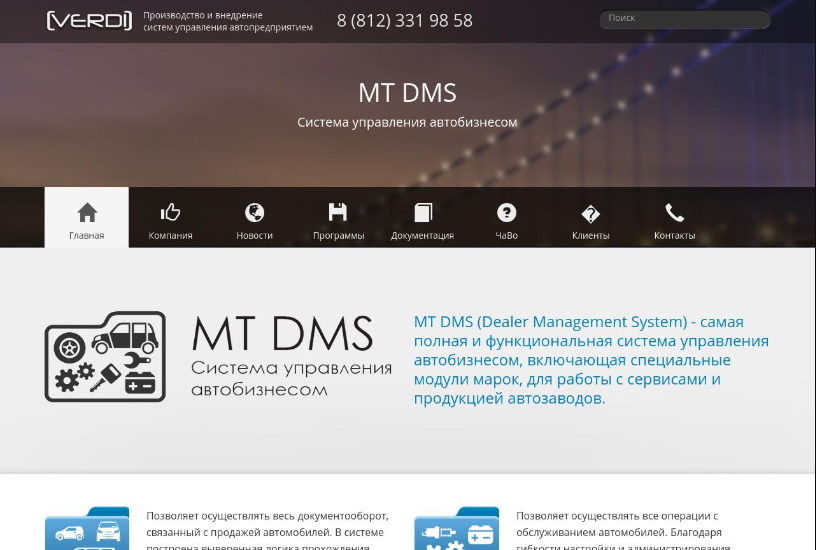


Рисунок 1 – MT DMS

**«1C Альфа-Авто»** - комплексное программное решение на базе 1С, специально разработанное для автосалонов и автосервисов. Программа отличается высоким уровнем функциональности и гибкими настройками под потребности автомобильного бизнеса. Основными функциями «Avaya IP Office» являются: - Управление звонками: IP Office позволяет управлять звонками и маршрутизацией вызовов в соответствии с заданными правилами. Это включает автоматическое распределение вызовов, управление очередями вызовов и перенаправление вызовов.

Основные возможности программы 1C Альфа-Авто:

* Ведение списка аналогов и замен: Альфа-авто позволяет вести список аналогов и замен запасных частей с возможностью быстрого и удобного доступа к данным об их текущих остатках. Это способствует более эффективному управлению запасами и быстрой подборке необходимых деталей для ремонта.
* Заполнение ячеек хранения запчастей: система предоставляет удобный интерфейс для заполнения данных о расположении запчастей на складе. Это позволяет быстрее и проще находить нужные запчасти при выполнении заказов.
* Подключение внешних прайс-листов: система имеет возможность подключения внешних прайс-листов разных форматов (txt, ods, xls, dbf). Это позволяет быстро и удобно обновлять данные о ценах и наличии запчастей у поставщиков без необходимости ручного ввода информации.
* Оформление заказов покупателей: система Альфа-авто позволяет с минимальными усилиями оформлять заказы покупателей с резервированием имеющихся в наличии позиций и заказом отсутствующих.
* Уведомление менеджеров о поступлении товаров: система автоматически уведомляет менеджеров о поступлении товаров по заказам покупателей, что существенно ускоряет процесс выполнения заказов и минимизирует недостачу запчастей.

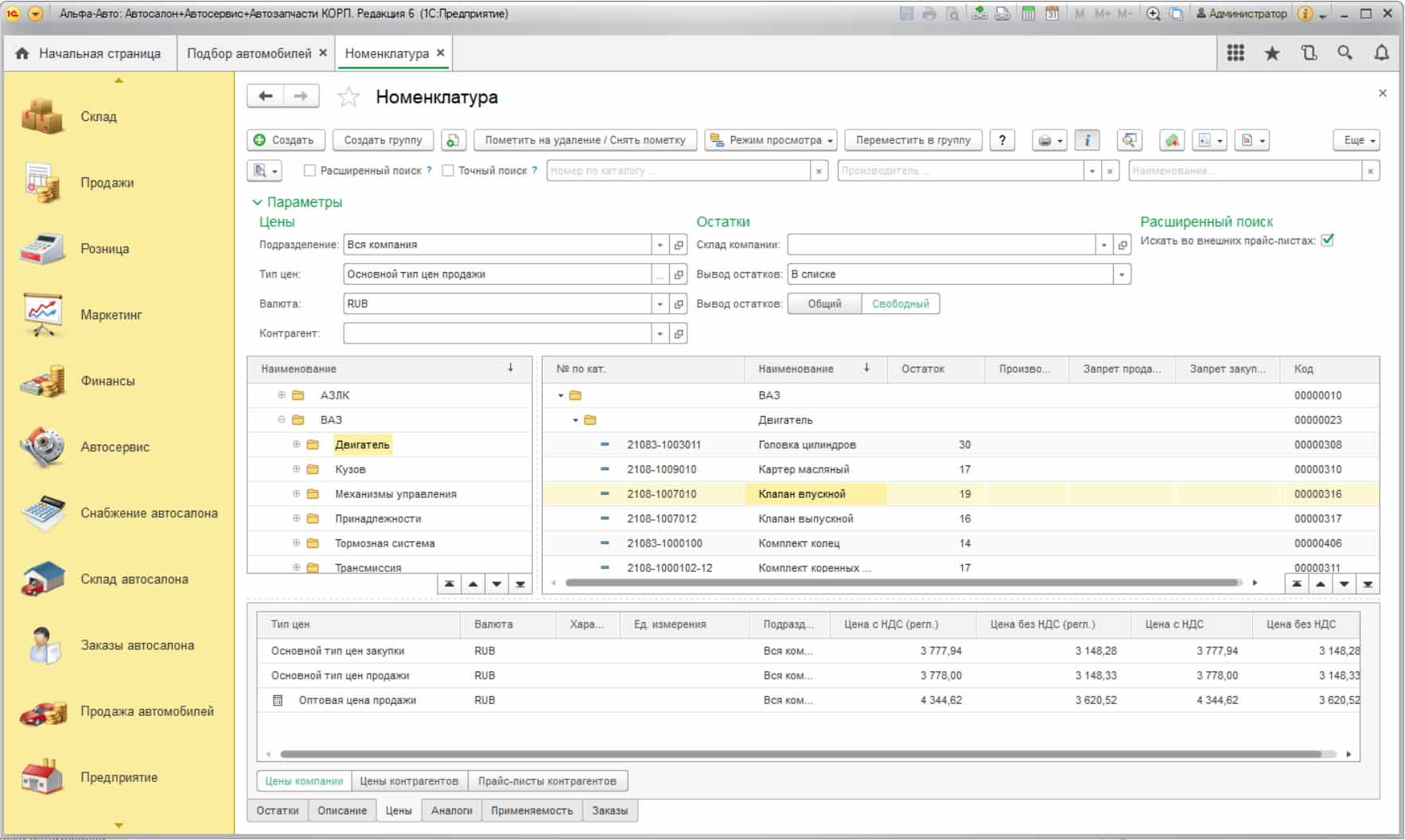


Рисунок 2 – 1C Альфа-Авто

Несмотря на то, что в России на сегодняшний день существует множество программ, ориентированных на автомобильный бизнес, Альфа-Авто продолжает оставаться лидером на рынке.

**«АвтоДилер»** - Онлайн программа для автосервисов. (Склад, проценка, продажа запчастей, CRM, запись на ремонт).

Основные возможности программы АвтоДилер:

* Программа позволяет настраивать распределение товаров по разным складам и стеллажам, оформлять приходные и расходные накладные, размещать заказ клиентов и заказы поставщику.
* Программа автоматически фиксирует обращения клиентов и создает на их основе сделки. Сделки помещаются на канбан, откуда их берут в работу сотрудники.
* Есть история ремонтов и раздел рекомендованных товаров и услуг. Запись клиентов на ремонт возможна в программе и через мобильное приложение.
* Есть интеграция с кассовыми сервисами Эвотор, Штрих-М и Атол, мессенджерами Telegram и WhatsApp, операторами ip-телефонии. Также есть возможность настроить выгрузку данных в 1С:Бухгалтерия.

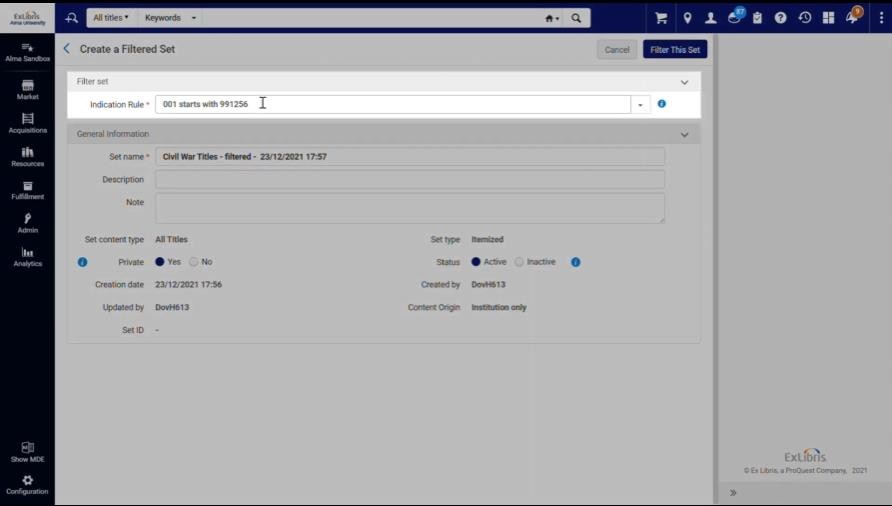


Рисунок 3 – АвтоДилер

# **ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ**

## **2.1 Обоснование выбора СУБД**

Система управления базой данных (СУБД) является критическим компонентом информационной системы. Она необходима для разработки программы на алгоритмическом языке. Основные функции СУБД включают:

* управление данными во внешней памяти (на дисках);
* управление данными в оперативной памяти;
* журнализация изменений и восстановление базы данных после сбоев;

Критерии выбора СУБД:

* Тип данных: необходимо определить, какие типы данных будут храниться в базе данных и выбрать СУБД, поддерживающую эти типы данных.
* Масштаб проекта: в зависимости от размеров проекта необходимо выбрать СУБД, способную эффективно обрабатывать большие объемы данных.
* Скорость обработки данных: если требуется быстрая обработка и доступ к данным, следует выбрать СУБД с высокой производительностью.
* Гибкость: СУБД должна быть гибкой и легко настраиваемой, чтобы соответствовать требованиям проекта.
* Стоимость: необходимо учитывать бюджет проекта при выборе СУБД, чтобы избежать лишних затрат.

## **СУБД - Microsoft SQL Server + MS SSMS**

Так как приложение будет разработано на платформе WPF (Windows Presentation Foundation ) с целью упрощения процесса разработки и оптимизации работы приложения для его разработки была выбрана СУБД Microsoft SQL Server в связке с MS SSMS.

**Microsoft SQL Server** — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

**Microsoft SQL Server Management Studio** (MS SSMS) — утилита из Microsoft SQL Server 2005 и более поздних версий для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server. Удобный графический интерфейс упрощает взаимодействие с СУБД, что ускоряет процесс разработки приложения.

## **2.2 Разработка инфологической модели**

Инфологическая модель – это потоки информации, сущности и связи данной области. В такой модели указываются связи между сущностями данной предметной области.

Сущность – представляет тип объектов, которые должны храниться в базе данных.

Связь – это ассоциирование нескольких сущностей с целью отыскания одних из них по значениям других.

Атрибут – это имя или структура поля записи. Атрибут характеризует размер или тип информации, содержащейся в поле. Его наименование должно быть уникальным для конкретного типа сущностей

Первичный ключ - это специальное поле в таблице, которое однозначно идентифицирует каждую запись или строку в базе данных.

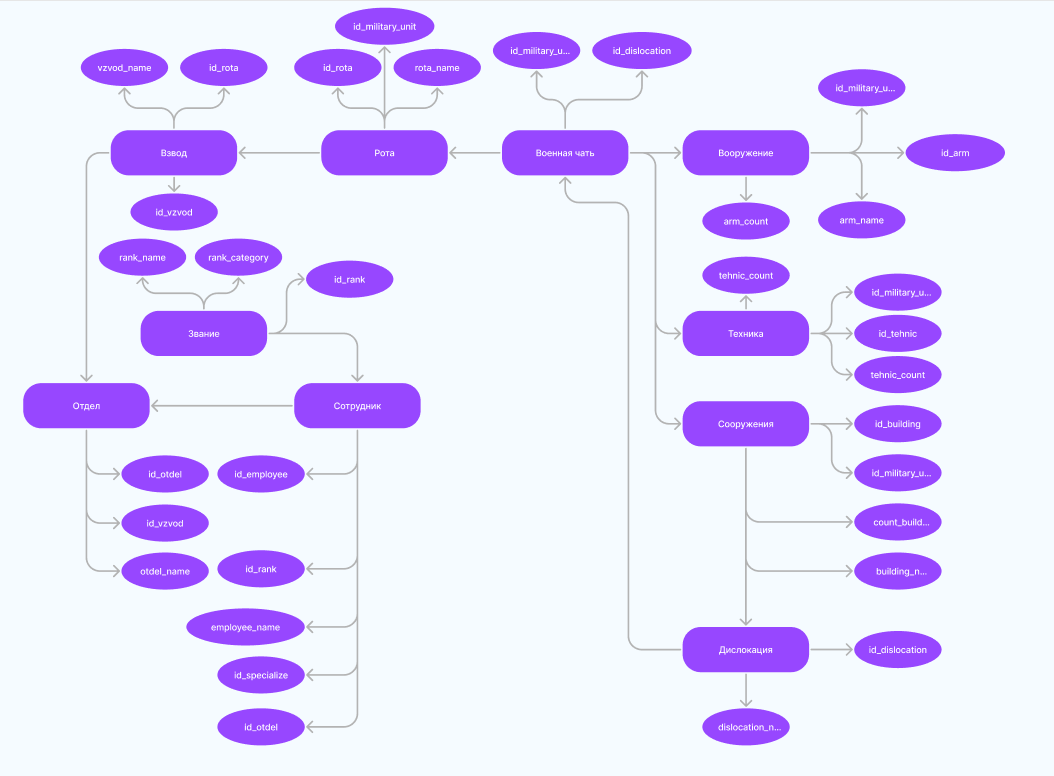


Рисунок 5 – Концептуальная модель

## **2.3 Нормализация базы данных**

Нормализация — это процесс организации данных в базе данных, включающий создание таблиц и установление отношений между ними в соответствии с правилами, которые обеспечивают защиту данных и делают базу данных более гибкой, устраняя избыточность и несогласованные зависимости.

Избыточность данных приводит к непродуктивному расходованию свободного места на диске и затрудняет обслуживание баз данных. Например, если данные, хранящиеся в нескольких местах, потребуется изменить, в них придется внести одни и те же изменения во всех этих местах.

**Первая нормальная форма**

• Устранение повторяющихся групп в отдельных таблицах;

• Создание отдельных таблиц для каждого набора связанных данных;

• Идентификация каждого набора связанных данных с помощью первичного ключа;

**Вторая нормальная форма**

• Создание отдельных таблиц для наборов значений, относящихся к нескольким записям;

• Связь этих таблиц с помощью внешнего ключа;

**Третья нормальная форма**

• Устранение полей, не зависящих от ключа;

Значения, входящие в запись и не являющиеся частью ключа этой записи, не принадлежат таблице (рисунок 2).

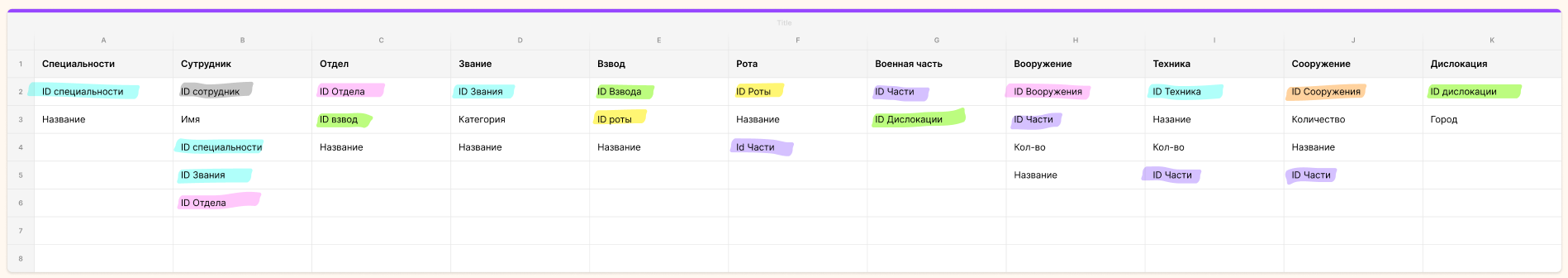


Рисунок 6 – Таблица данных

## **2.4 Разработка даталогической модели базы данных**

Даталогическое проектирование сводится к представлению инфологической модели в терминах выбранной системы управления базами данных (СУБД), т. е. даталогическая модель описывает собственно данные (информацию, которая будет записана в памяти компьютера) и связи между данными. Каждая СУБД опирается на определенную модель данных.

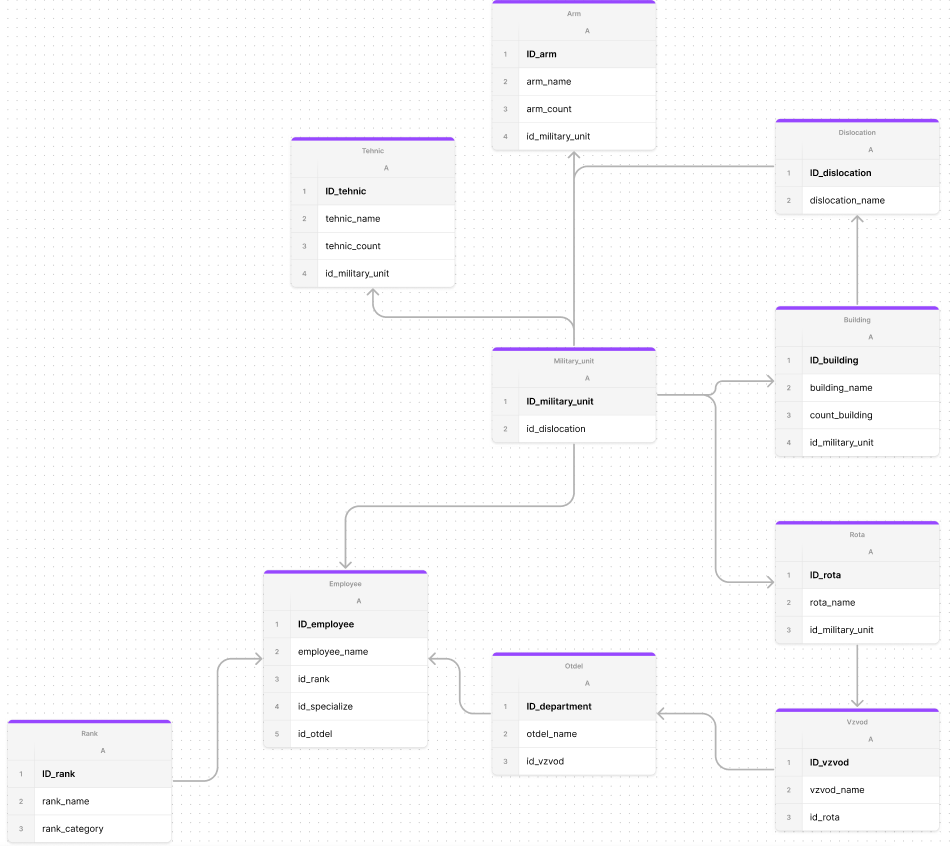


Рисунок 7 – Даталогическая модель

## **2.5 Инициализация базы данных**

Таблицы:

*Звание*

CREATE TABLE [rank] (

[id\_rank] INT CONSTRAINT [PK\_rank\_id] PRIMARY KEY IDENTITY(1, 1),

[rank\_name] NVARCHAR(100) NOT NULL,

[rank\_category] NVARCHAR(100) NOT NULL,

);

*Дислокация*

CREATE TABLE [dislocation] (

[id\_dislocation] INT CONSTRAINT [PK\_dislocation\_id] PRIMARY KEY

IDENTITY(1, 1),

[dislocation\_name] NVARCHAR(100) NOT NULL,

);

*Сооружения*

CREATE TABLE [building] (

[id\_building] INT CONSTRAINT [PK\_building\_id] PRIMARY KEY

IDENTITY(1, 1),

[building\_name] NVARCHAR(100) NOT NULL,

[count\_building] INT,

[id\_military\_unit] INT CONSTRAINT [FK\_military\_unit\_building\_id]

FOREIGN KEY REFERENCES [military\_unit]([id\_military\_unit]),

);

*Военная часть*

CREATE TABLE [military\_unit] (

[id\_military\_unit] INT CONSTRAINT [PK\_military\_unit\_id] PRIMARY

KEY IDENTITY(1, 1),

[id\_dislocation] INT CONSTRAINT [FK\_dislocation\_id] FOREIGN KEY

REFERENCES [dislocation]([id\_dislocation])

);

*Техника*

CREATE TABLE [tehnic] (

[id\_tehnic] INT CONSTRAINT [PK\_tehnic\_id] PRIMARY KEY IDENTITY(1, 1),

[tehnic\_name] NVARCHAR(100) NOT NULL,

[tehnic\_count] INT NOT NULL,

[id\_military\_unit] INT CONSTRAINT [FK\_military\_unit\_id] FOREIGN

KEY REFERENCES [military\_unit]([id\_military\_unit]),

);

*Вооружение*

CREATE TABLE [arm] (

[id\_arm] INT CONSTRAINT [PK\_arm\_id] PRIMARY KEY IDENTITY(1, 1),

[arm\_name] NVARCHAR(100) NOT NULL,

[arm\_count] INT NOT NULL,

[id\_military\_unit] INT CONSTRAINT [FK\_military\_unit\_arm\_id]

FOREIGN KEY REFERENCES [military\_unit]([id\_military\_unit]),

);

*Рота*

CREATE TABLE [rota] (

[id\_rota] INT CONSTRAINT [PK\_rota\_id] PRIMARY KEY IDENTITY(1, 1),

[rota\_name] NVARCHAR(100) NOT NULL,

[id\_military\_unit] INT CONSTRAINT [FK\_military\_unit\_rota\_id]

FOREIGN KEY REFERENCES [military\_unit]([id\_military\_unit]),

);

*Взвод*

CREATE TABLE [vzvod] (

[id\_vzvod] INT CONSTRAINT [PK\_vzvod\_id] PRIMARY KEY IDENTITY(1, 1),

[vzvod\_name] NVARCHAR(100) NOT NULL,

[id\_rota] INT CONSTRAINT [FK\_rota\_vzvod\_id] FOREIGN KEY

REFERENCES [rota]([id\_rota]),

);

*Отдел*

CREATE TABLE [otdel] (

[id\_otdel] INT CONSTRAINT [PK\_otdel\_id] PRIMARY KEY IDENTITY(1, 1),

[otdel\_name] NVARCHAR(100) NOT NULL,

[id\_vzvod] INT CONSTRAINT [FK\_vzvod\_otdel\_id] FOREIGN KEY

REFERENCES [vzvod]([id\_vzvod]),

);

*Сотрудник*

CREATE TABLE [employee] (

[id\_employee] INT CONSTRAINT [PK\_employee\_id] PRIMARY KEY IDENTITY(1, 1),

[employee\_name] NVARCHAR(100) NOT NULL,

[id\_rank] INT CONSTRAINT [FK\_rank\_id] FOREIGN KEY REFERENCES [rank]([id\_rank]),

[id\_specialize] INT CONSTRAINT [FK\_specialize\_employee\_id] FOREIGN KEY REFERENCES [specialize]([id\_specialize]),

[id\_otdel] INT CONSTRAINT [FK\_otdel\_employee\_id] FOREIGN KEY REFERENCES [otdel]([id\_otdel]),

);

*Специализация*

CREATE TABLE [specialize] (

[id\_specialize] INT CONSTRAINT [PK\_specialize\_id] PRIMARY KEY IDENTITY(1, 1),

[specialize\_name] NVARCHAR(100) NOT NULL,

);

## **2.6 Заполнение таблиц данными**

Таблицы:

*Звание*

INSERT INTO [rank] ([rank\_name], [rank\_category]) VALUES(N'Рядовой', N'Сержанский состав');

INSERT INTO [rank] ([rank\_name], [rank\_category]) VALUES(N'Ефрейтор', N'Сержанский состав');

INSERT INTO [rank] ([rank\_name], [rank\_category]) VALUES(N'Прапорщик', N'Сержанский состав');

INSERT INTO [rank] ([rank\_name], [rank\_category]) VALUES(N'Сержант', N'Сержанский состав');

INSERT INTO [rank] ([rank\_name], [rank\_category]) VALUES(N'Старшин', N'Сержанский состав');

INSERT INTO [rank] ([rank\_name], [rank\_category]) VALUES(N'Лейтенант', N'Офицерский состав');

INSERT INTO [rank] ([rank\_name], [rank\_category]) VALUES(N'Капитан', N'Офицерский состав');

INSERT INTO [rank] ([rank\_name], [rank\_category]) VALUES(N'Майор', N'Офицерский состав');

INSERT INTO [rank] ([rank\_name], [rank\_category]) VALUES(N'Подполковник', N'Офицерский состав');

INSERT INTO [rank] ([rank\_name], [rank\_category]) VALUES(N'Полковник', N'Офицерский состав');

INSERT INTO [rank] ([rank\_name], [rank\_category]) VALUES(N'Генералы', N'Офицерский состав');

*Дислокация*

INSERT INTO [dislocation] ([dislocation\_name])

VALUES(N'Москва');

INSERT INTO [dislocation] ([dislocation\_name]) VALUES(N'Челябинск');

*Сооружения*

INSERT INTO [building] ([building\_name], [count\_building], [id\_military\_unit]) VALUES(N'Штаб', 1, 2);

INSERT INTO [building] ([building\_name], [count\_building], [id\_military\_unit]) VALUES(N'казарма', 2, 2);

*Военная часть*

INSERT INTO [military\_unit] ([id\_dislocation]) VALUES(2);

INSERT INTO [military\_unit] ([id\_dislocation]) VALUES(1);

*Техника*

INSERT INTO [tehnic] ([tehnic\_name], [tehnic\_count], [id\_military\_unit]) VALUES(N'Тягач', 5, 1);

INSERT INTO [tehnic] ([tehnic\_name], [tehnic\_count], [id\_military\_unit]) VALUES(N'БМП', 10, 1);

INSERT INTO [tehnic] ([tehnic\_name], [tehnic\_count], [id\_military\_unit]) VALUES(N'Танк', 2, 1);

INSERT INTO [tehnic] ([tehnic\_name], [tehnic\_count], [id\_military\_unit]) VALUES(N'Тягач', 15, 2);

INSERT INTO [tehnic] ([tehnic\_name], [tehnic\_count], [id\_military\_unit]) VALUES(N'БМП', 40, 2);

INSERT INTO [tehnic] ([tehnic\_name], [tehnic\_count], [id\_military\_unit]) VALUES(N'Танк', 11, 2);

*Вооружение*

INSERT INTO [arm] ([arm\_name], [arm\_count], [id\_military\_unit]) VALUES(N'Карабин', 200, 2);

INSERT INTO [tehnic] ([tehnic\_name], [tehnic\_count], [id\_military\_unit]) VALUES(N'Артиллерия', 70, 2);

INSERT INTO [tehnic] ([tehnic\_name], [tehnic\_count], [id\_military\_unit]) VALUES(N'Ракетное вооружение', 30, 2);

*Рота*

INSERT INTO [rota] ([rota\_name], [id\_military\_unit]) VALUES(N'Мотострелковая', 2);

INSERT INTO [rota] ([rota\_name], [id\_military\_unit]) VALUES(N'Десантно-штурмовая', 2);

INSERT INTO [rota] ([rota\_name], [id\_military\_unit]) VALUES(N'Танковая', 2);

INSERT INTO [rota] ([rota\_name], [id\_military\_unit]) VALUES(N'Разведывательная', 2);

INSERT INTO [rota] ([rota\_name], [id\_military\_unit]) VALUES(N'Ремонтная', 2);

INSERT INTO [rota] ([rota\_name], [id\_military\_unit]) VALUES(N'Связь', 2);

*Взвод*

INSERT INTO [vzvod] ([vzvod\_name], [id\_rota]) VALUES(N'Противотанковый', 12);

*Отдел*

INSERT INTO [otdel] ([otdel\_name], [id\_vzvod]) VALUES(N'Связь', 2);

INSERT INTO [otdel] ([otdel\_name], [id\_vzvod]) VALUES(N'Обеспечение', 2);

*Сотрудник*

INSERT INTO [employee] ([employee\_name], [id\_rank], [id\_specialize], [id\_otdel]) VALUES(N'Горбачев Роман Андреевич', 2, 1, 1);

*Специализация*

INSERT INTO [specialize] ([specialize\_name]) VALUES(N'Врач');

INSERT INTO [specialize] ([specialize\_name]) VALUES(N'Инженер');

INSERT INTO [specialize] ([specialize\_name]) VALUES(N'Летчик');

INSERT INTO [specialize] ([specialize\_name]) VALUES(N'Солдат');

INSERT INTO [specialize] ([specialize\_name]) VALUES(N'Радист');

INSERT INTO [specialize] ([specialize\_name]) VALUES(N'Картограф');

**2.7 Дополнительные инструменты**

**Figma** — онлайн-сервис для разработки интерфейсов и прототипирования с возможностью организации совместной работы в режиме реального времени.

Сервис доступен по подписке, предусмотрен бесплатный тарифный план для одного пользователя. Имеются офлайн-версии для Windows, macOS. Реализована интеграция с корпоративным мессенджером Slack и инструментом прототипирования Framer.

Используется как для создания упрощённых прототипов интерфейсов, так и для детальной проработки дизайна интерфейсов мобильных приложений, веб-сайтов, корпоративных порталов.

# **ГЛАВА 3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ БАЗОЙ ДАННЫХ**

## **3.1 Обоснование выбора среды разработки**

**Microsoft Visual Studio** — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов.

Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, WPF, UWP а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, .NET Core, .NET, MAUI, Xbox, Windows Phone .NET Compact Framework и Silverlight.

После покупки компании Xamarin корпорацией Microsoft появилась возможность разработки IOS и Android программ.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня.

Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных.

Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Учитывая все вышесказанное, я выбрал данный инструмент для реализации приложения.

В качестве технологии реализации была выбрана платформа “WPF” в cвязке с пакетом “Entity framework”

**Windows Presentation Foundation (WPF)** — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML. WPF предустановлена в Windows Vista (.NET Framework 3.0), Windows 7 (.NET Framework 3.5 SP1), Windows 8 (.NET Framework 4.0 и 4.5), Windows 8.1 (.NET Framework 4.5.1) и Windows 10 (.NET Framework 4.7). С помощью WPF можно создавать широкий спектр как автономных, так и запускаемых в браузере приложений.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Windows Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX. Также существует урезанная версия CLR, называющаяся WPF/E, она же известна как Silverlight.

**ADO.NET Entity Framework (EF)** — объектно-ориентированная технология доступа к данным, является object-relational mapping (ORM) решением для .NET Framework от Microsoft. Предоставляет возможность взаимодействия с объектами как посредством LINQ в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как ADO.NET Data Services (Astoria), так и связка из Windows Communication Foundation и Windows Presentation Foundation, позволяющая строить многоуровневые приложения, реализуя один из шаблонов проектирования MVC, MVP или MVVM.

## **3.2 Проектирование приложения**

Для проектирования структуры и дизайна интерфейса выбрал приложение – “Figma”, которое указывал ранее.

Было отрисовано два основных экрана приложения, для представления архитектуры приложения.

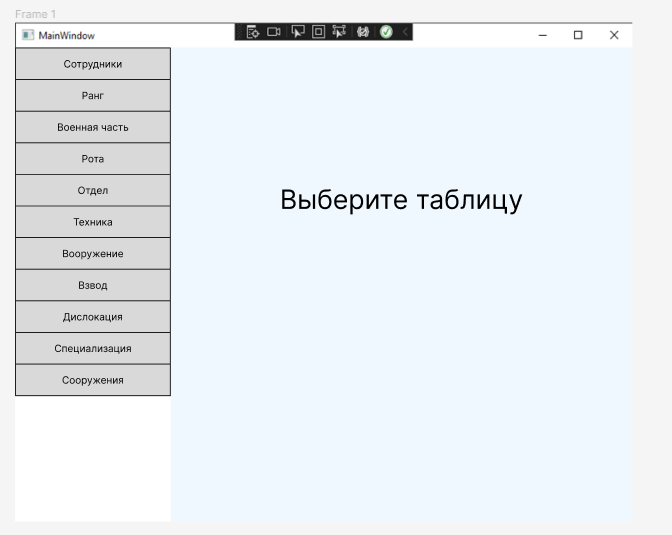


Рисунок 8 – Главный экран приложения

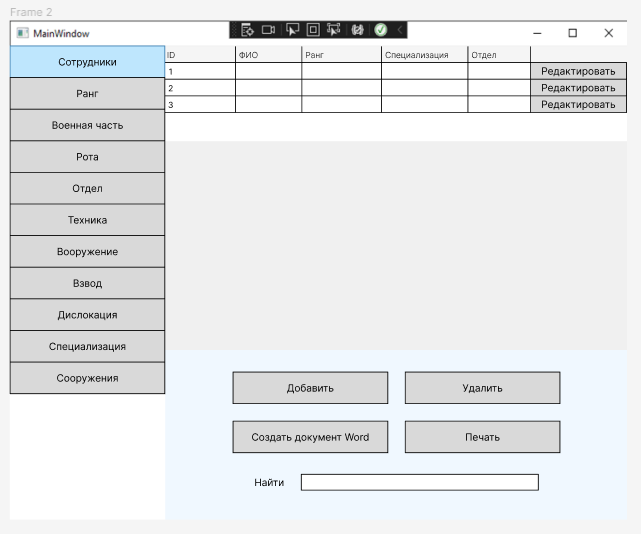


Рисунок 9 – Экран выбранной таблицы

Также были отрисованы состояния кнопок при взаимодействии пользователя с ними.

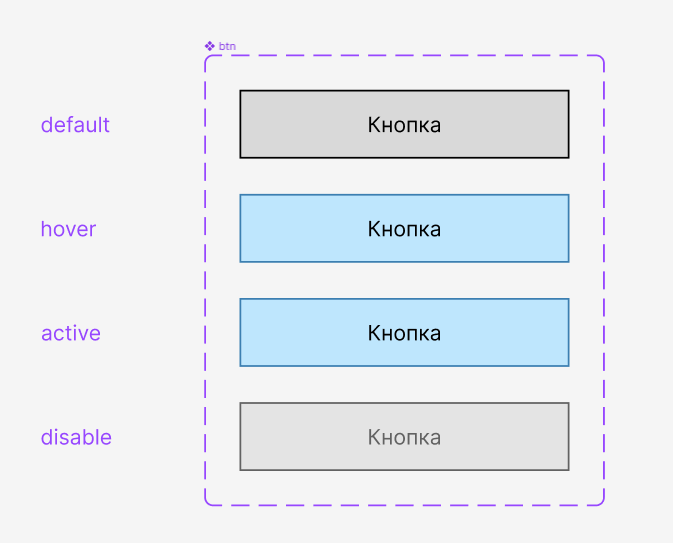


Рисунок 10 – Состояния кнопок

## **3.3 Разработка приложения**

Для реализации подключения базы данных к приложению, был создан класс, который содержит в себе статический метод получения данных из базы данных GetContext().

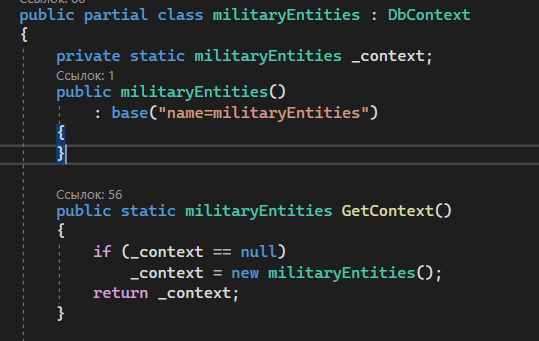


Рисунок 11 – Класс militaryEntities

Также были созданы классы под каждую таблицу.

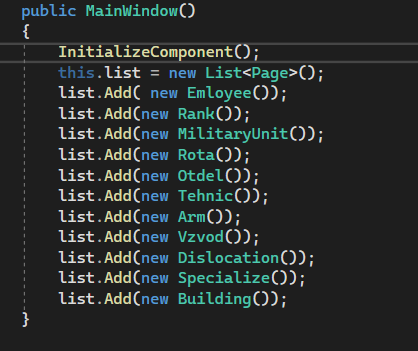


Рисунок 12 – Список табличных классов

Добавление новых данных в базу реализовано по средствам открытия нового WPF окна с полями для ввода данных.

Создаем новый экземпляр класса, в котором будет содержаться информация о текущем объекте. Связываем поля для ввода с полями в таблице.

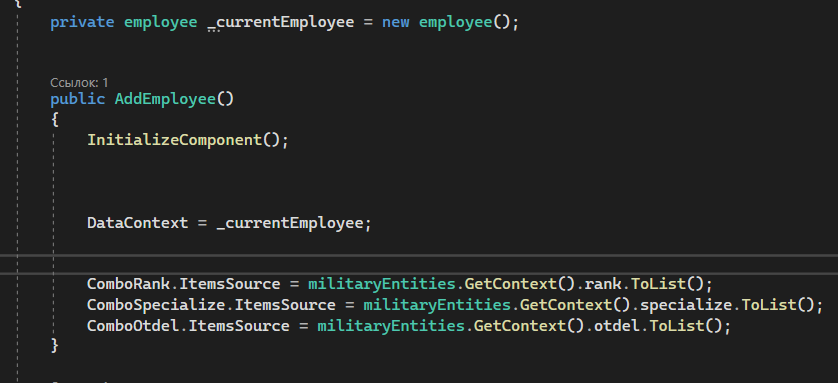


Рисунок 13 – Получение данных о новом объекте

Также реализуем проверку на отсутствие пустых полей при отправке данных.

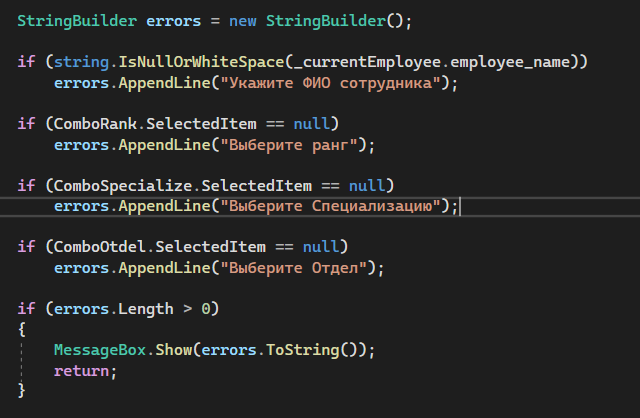


Рисунок 14 – Валидация полей

Создаем кнопку сохранения и реализуем событие нажатия на эту кнопку. Заносим данные о текущем объекте в общую таблицу с данными и сохраняем изменения с помощью специального метода из библиотеки Entity. Все эти действия оборачиваем в конструкцию try-catch, для обработки возможных ошибок, поскольку данные операции с базой данных являются потенциально опасными.

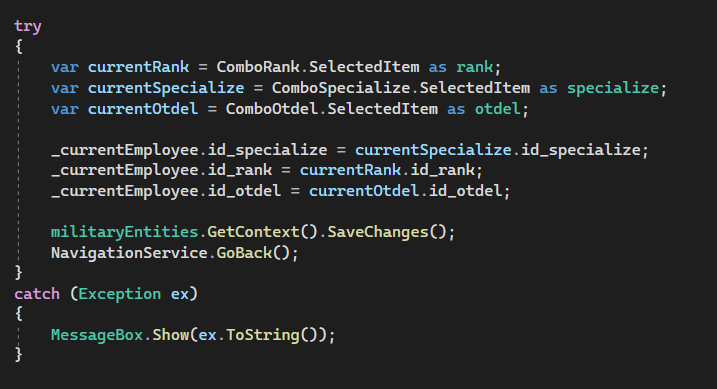


Рисунок 15 – Сохранение данных о новом объекте

Удаление данных из базы происходит путем выделения строки с удаляемым объектом и нажатия на кнопку “Удалить”.

При нажатии на кнопку срабатывает событие, в котором получаем выбранный объект. После вызываем диалоговое окно с вопросом об удалении данных из таблицы, если пользователь соглашается, то вызывается метод удаления данных, после этого сохраняем все изменения и заново отрисовываем нашу таблицу. Все эти операции также оборачиваем в конструкцию “try-catch”.

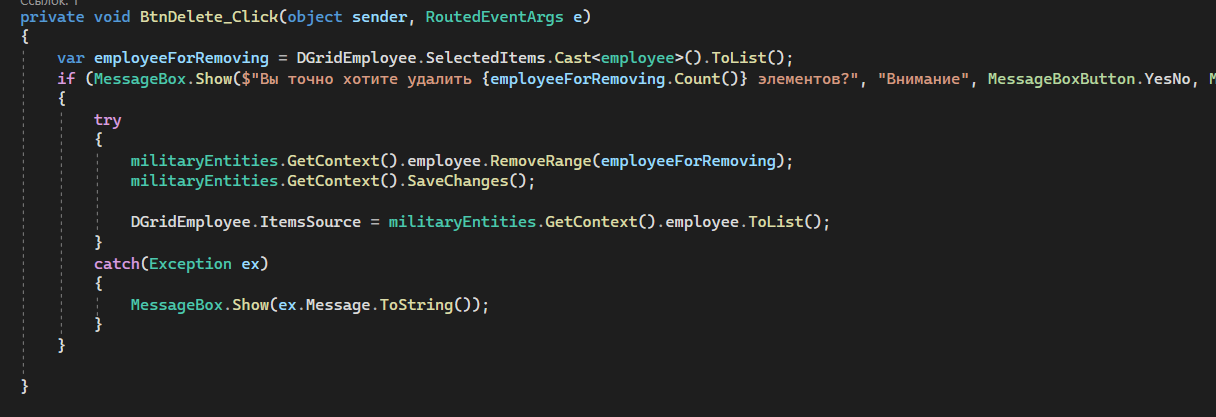


Рисунок 16 – Удаление объекта из таблицы

Реализация функции поиска представлена на рисунке 16.

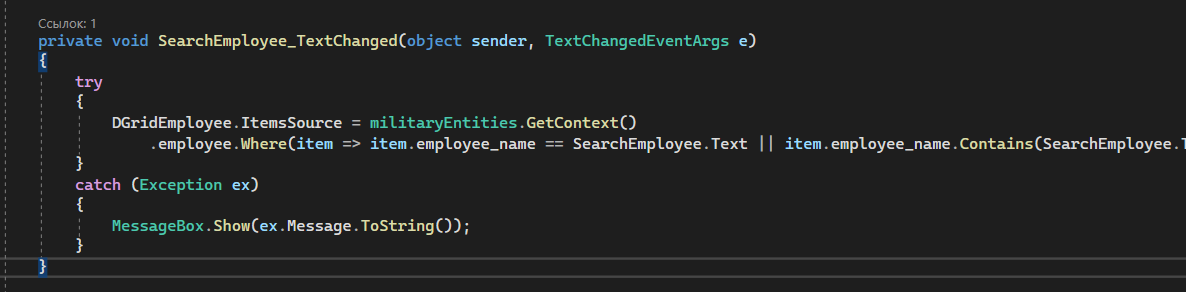


Рисунок 17 – Функция поиска

Также были реализованы функции генерации отчета word и вывода таблицы на печать.

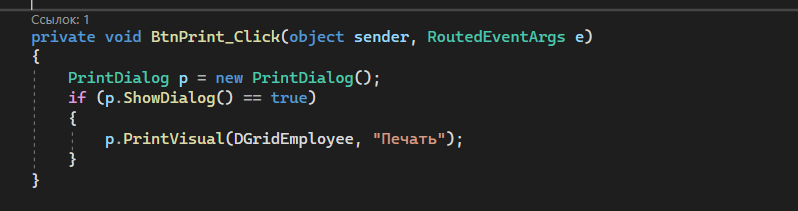


Рисунок 18 – Функция печати

Генерация отчета реализуется с помощью специальной кнопки, которая обращается к созданному классу “Word” (рисунок 18), который добавляет информацию в документ, создавая новый параграф.

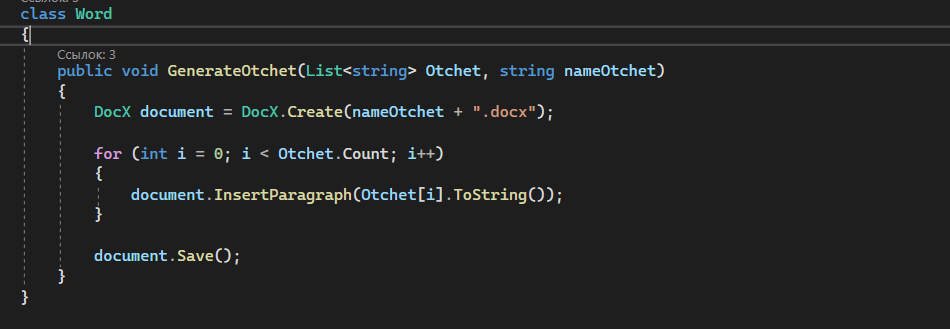


Рисунок 19 – Класс “Word”

Приложение представляет из себя удобный модульный интерфейс, который состоит из так называемых WPF страниц, которые изменяют свое отображение на экране в зависимости от действий пользователя.

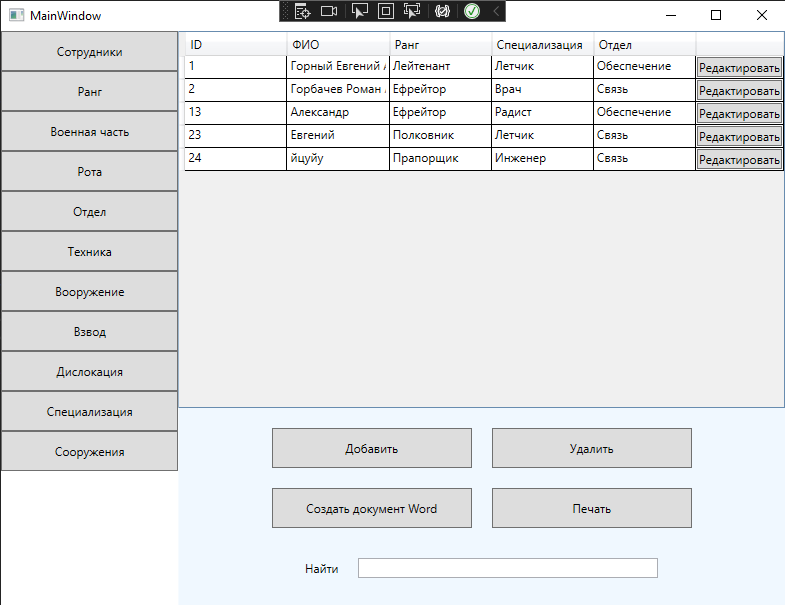


Рисунок 20 – Приложение(общ. вид)

Данные подгружаются на страницы и импортируются в таблицы из базы данных, которая подключается к приложению с помощью специального кода.

Благодаря возможностям Entity Framework, реализованы такие функции управления базой, как – добавление, удаление, редактирование всех данных, через специальные кнопки и другие элементы управления.

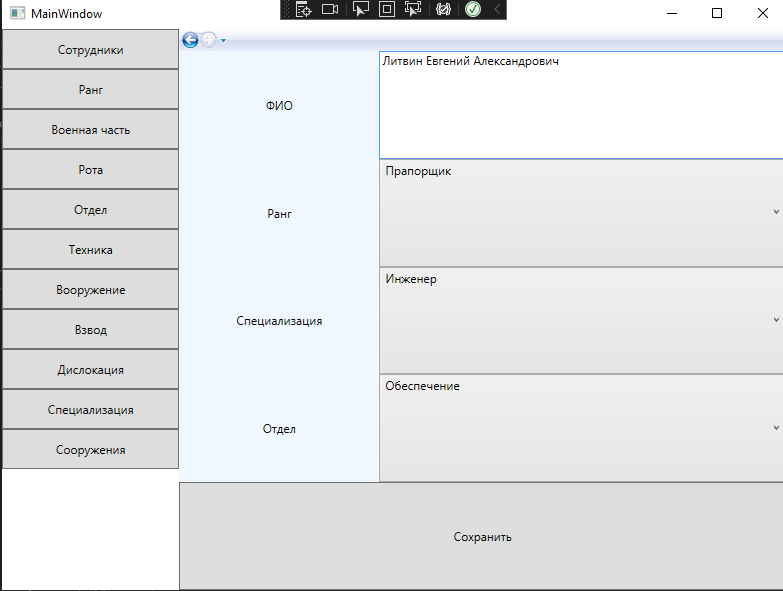


Рисунок 21 – Приложение(Добавление данных в таблицу)

В качестве дополнительных функций была реализована возможность вывода каждой таблицы данных на печать и создание word – отчета, через специализированную библиотеку Xceed – сервис для работы с документами word.

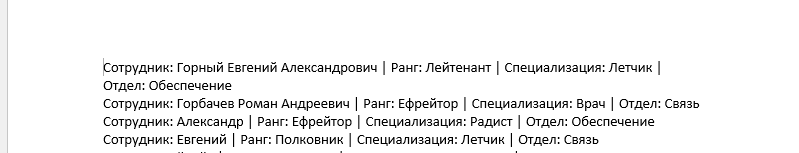


Рисунок 22 – Приложение(Генерация отчета)

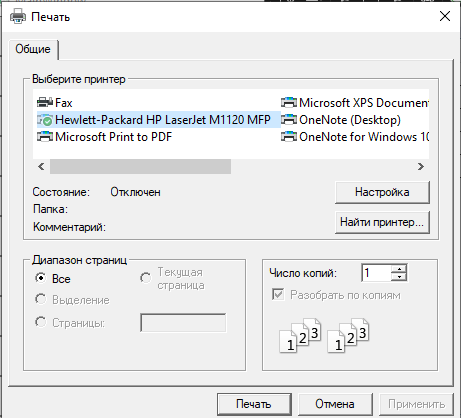


Рисунок 23 – Приложение(Вывод таблиц на печать)

Помимо этого, была реализована функция поиска данных по конкретным таблицам, для более удобного взаимодействия с данными.

Напоследок, реализовал небольшую валидацию данных, для того чтобы избежать отправку пустых форм.

# **ЗАКЛЮЧЕНИЕ**

Рассмотрим результаты проведенной работы, целью которого являлось создание базы данных и приложения для реализации управления этой базой.

В перечень выполненных работ попадает:

• Анализ действующих решений;

• Анализ возможных решений для данной задачи;

• Проектирование и создание реляционной базы данных;

• Проектирование и создание приложения для управления базой данных;

Все эти работы в комплексе помогли научиться создавать правильные с точки зрения логики построения структуры базы данных, разобраться в многообразии инструментов проектирования и программирования баз данных. Освоить для себя новые инструменты для взаимодействия с данными.

На данном этапе развития функционал приложения логически завершён.

# **СПИСОК ИСПОЛЬЗУЕМЫХ РЕСУРСОВ**

1. «Введение в системы баз данных» К. Дж. Дейт.
2. «MySQL по максимуму» Бэрон Шварц, Вадим Ткаченко, Петр Зайцев.
3. WPF: Windows Presentation Foundation в .NET 4.5. Мэтью Макдональд.
4. Основы Windows Presentation Foundation. Крис Андерсон.