**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский арктический горно-политехнический колледж»**

**КУРСОВОЙ ПРОЕКТ**

По дисциплине МДК.05.02 Разработка кода информационных систем

**Проектирование информационной системы**

**«Сервис передержки животных»**

Выполнил студент гр. ИСП-22 /\_\_\_\_\_\_\_\_\_\_\_\_\_/Карпова Мария Вадимовна/

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Научный руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Данил Павлович/

(подпись) (Ф.И.О.)

Воркута

2025

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc196828429)

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc196828430)

[1.1 Платформа .NET 4](#_Toc196828431)

[1.2 Язык программирования С# 5](#_Toc196828432)

[1.3 Windows Presentation Foundation (WPF) 6](#_Toc196828433)

[1.4 СУБД SQL server 7](#_Toc196828434)

[1.5 Microsoft SQL Server Management Studio 8](#_Toc196828435)

[1.6 Entity Framework 9](#_Toc196828436)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 10](#_Toc196828437)

[2.1 Разработка диаграммы ERD 10](#_Toc196828438)

[2.2 Разработка базы данных 13](#_Toc196828439)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 15](#_Toc196828440)

[3.1 Программирование информационной системы 15](#_Toc196828441)

[3.1.1 Разработка модуля «Авторизация» 15](#_Toc196828442)

[3.1.2 Разработка модуля «Главное меню» 18](#_Toc196828443)

[3.1.4 Разработка модуля «Осмотры» 25](#_Toc196828444)

[3.1.5 Разработка модуля «Отчет» 30](#_Toc196828445)

[ЗАКЛЮЧЕНИЕ 37](#_Toc196828446)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 38](#_Toc196828447)

# ВВЕДЕНИЕ

Актуальность выбранной темы в том, что в любой современной фирме или организации необходимы автоматизированные информационные системы. Например, чтобы сотрудникам предприятия не пришлось работать с чрезвычайно большим количеством документов для ручной обработки и ускорить работу за счет автоматизации.

Система, автоматизирующая управление центра передержки животных, должна включать в себя функции наблюдения за состоянием животных на время передержки, создания отчетов. Это поможет выполнять работу гораздо быстрее и уменьшить количество ошибок при ее выполнении. В курсовом проекте будут рассмотрены основные принципы создания информационной системы по передержке животных, а также практические аспекты ее внедрения и использования.

Данная система подойдет для людей, которые работают в сервисах по передержке животных.

**Объект:** информационная система «Сервис передержки животных».

**Предмет:** анализ бизнес-процессов «Сервис передержки животных».

**Цель работы:** разработать автоматизированную информационную систему «Сервис передержки животных»

**Задачи:**

* выбрать инструментарий;
* спроектировать базу данных;
* разработать информационную систему.

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

# 1.1 Платформа .NET

**.NET** — это современная и гибкая платформа для разработки приложений, построенная по модульному принципу. Она предоставляет все необходимые инструменты и библиотеки для создания широкого спектра программ: от интерактивных веб-сайтов и удобных мобильных приложений до масштабируемых облачных сервисов и мощных серверных решений.

.NET эволюционировала из .NET Framework, но отличается от нее фундаментальными улучшениями. Главное отличие — это модульность. Вместо монолитной структуры, .NET состоит из отдельных, независимо обновляемых компонентов. Это также кроссплатформенная платформа, что позволяет запускать приложения на Windows, macOS и Linux. Ещё одно важное отличие — ориентация на облачные технологии, что упрощает развертывание и масштабирование приложений в облаке. Архитектура .NET разделяет библиотеку базовых классов (CoreFX) и среду выполнения (CoreCLR), что повышает гибкость и эффективность.

.NET поддерживает несколько языков программирования, что дает разработчикам свободу выбора. Основными языками являются C#, который часто используется для разработки большинства приложений .NET, F#, функциональный язык, подходящий для сложных вычислений и data-driven решений, и Visual Basic .NET, который поддерживается частично и подходит для быстрой разработки приложений с использованием знакомого синтаксиса. Выбор языка зависит от специфики проекта и предпочтений разработчика.

# 1.2 Язык программирования С#

**C#**  — [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения. Разработан в [1998](https://ru.wikipedia.org/wiki/1998)—[2001 годах](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) группой инженеров компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) как язык разработки приложений для платформы [Microsoft .NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) и [.NET Core](https://ru.wikipedia.org/wiki/.NET_Core). Впоследствии был стандартизирован.

C# относится к семье языков с [C-подобным синтаксисом](https://ru.wikipedia.org/wiki/C-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81), из них его синтаксис наиболее близок к [C++](https://ru.wikipedia.org/wiki/C%2B%2B) и [Java](https://ru.wikipedia.org/wiki/Java). Язык имеет [статическую типизацию](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), поддерживает [полиморфизм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [перегрузку операторов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2) (в том числе операторов явного и неявного приведения типа), [делегаты](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%B3%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), атрибуты, [события](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [переменные](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [свойства](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [обобщённые](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) типы и методы, [итераторы](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80), [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) с поддержкой [замыканий](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [LINQ](https://ru.wikipedia.org/wiki/Language_Integrated_Query), [исключения](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), [комментарии](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в формате [XML](https://ru.wikipedia.org/wiki/XML).

Переняв многое от своих предшественников — языков [C++](https://ru.wikipedia.org/wiki/C%2B%2B), [Delphi](https://ru.wikipedia.org/wiki/Delphi_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [Модула](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0-2), [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk) и, в особенности, [Java](https://ru.wikipedia.org/wiki/Java) — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает [множественное наследование](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) классов (между тем допускается [множественная реализация интерфейсов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)#%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B8_%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%8)).

Сейчас язык C# стал опенсорсным: каждый может внести в него свой вклад, но Microsoft при этом сохраняет контроль, поддерживает философию продукта, выбирает лучшие предложения. Обновление языка происходит раз в год.

# 1.3 Windows Presentation Foundation (WPF)

**Windows Presentation Foundation (WPF)** — это мощная и современная система для создания визуально привлекательных и интерактивных клиентских приложений для операционной системы Windows. Подобно своему предшественнику, WinForms, WPF предоставляет платформу для разработки пользовательских интерфейсов, но с более широким набором возможностей и улучшенной архитектурой. Она является графической (презентационной) подсистемой, интегрированной в .NET Framework, начиная с версии 3.0, и продолжает развиваться в составе .NET (ранее .NET Core) и .NET Framework.

Ключевой особенностью WPF является использование языка разметки XAML (Extensible Application Markup Language). XAML — это декларативный язык, основанный на XML, который предназначен для описания структуры и внешнего вида пользовательского интерфейса. Вместо того чтобы создавать элементы интерфейса программно, разработчики могут определить их в XAML, что делает код более читаемым и простым в обслуживании. XAML позволяет описывать не только базовые элементы управления, но и сложные визуальные эффекты, анимации и стили.

WPF реализует эффективную модель разделения кода и дизайна, что значительно упрощает совместную работу программистов и дизайнеров. Ди и свойств/действий. Эта гранулярность позволяет добиться беспрецедентной гибкости и контроля над внешним видом каждого элемента, позволяя легко задавать сложные стили и темы для всего приложения.

Для разработки приложений WPF можно использовать любой .NET-совместимый язык программирования. В этот обширный список входят такие популярные языки, как C#, F#, VB.NET, а также C++, Ruby, Python (с использованием IronPython), Delphi (Prism), Lua и многие другие. Выбор языка зависит от предпочтений разработчика и требований проекта.

# 1.4 СУБД SQL server

**Microsoft SQL Server** — это мощная система управления реляционными базами данных (РСУБД), созданная компанией Microsoft. Она предназначена для организации, хранения и обработки данных в структурированном формате.

Для взаимодействия с SQL Server используется язык запросов Transact-SQL (T-SQL). T-SQL разработан совместно Microsoft и Sybase и представляет собой расширенную реализацию стандарта ANSI/ISO SQL. Он позволяет выполнять широкий спектр операций с данными, включая создание и изменение таблиц, добавление, удаление и обновление записей, а также сложные запросы для извлечения нужной информации.

SQL Server подходит для работы с базами данных разного размера — от небольших, используемых в персональных приложениях, до огромных, применяемых в крупных корпоративных системах. Он конкурирует с другими популярными СУБД, такими как Oracle, MySQL и PostgreSQL, предлагая широкий набор функций и инструментов для управления данными.

Экосистема SQL Server предоставляет возможность развертывания баз данных в облаке с помощью сервиса Azure SQL. В этом случае Microsoft берет на себя задачи по обслуживанию инфраструктуры, резервному копированию, обеспечению безопасности и масштабированию, позволяя разработчикам сосредоточиться на создании и поддержке клиентских приложений, использующих эти данные.

Поскольку SQL Server разработан компанией Microsoft, он тесно интегрирован с операционной системой Windows и предоставляет оптимальную производительность и совместимость в этой среде. Хотя SQL Server также поддерживает Linux, функциональность версии для Linux ограничена по сравнению с версией для Windows. Некоторые ключевые функции и инструменты могут быть недоступны в Linux-версии.

# 1.5 Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) — это комплексная и мощная утилита, входящая в состав Microsoft SQL Server начиная с версии 2005 и всех последующих версий. Она предназначена для конфигурирования, управления и администрирования всех аспектов и компонентов Microsoft SQL Server. Это незаменимый инструмент для администраторов баз данных, разработчиков и всех, кто работает с SQL Server.

SSMS объединяет в себе два основных компонента: скриптовый редактор и графическую программу. Скриптовый редактор позволяет писать и выполнять SQL-запросы, хранимые процедуры и другие скрипты для управления базой данных. Графическая программа предоставляет интуитивно понятный интерфейс для работы с объектами сервера и изменения его настроек, что делает выполнение административных задач более простым и наглядным.

Ключевым инструментом SSMS является Object Explorer. Object Explorer представляет собой иерархическое представление всех объектов SQL Server, таких как базы данных, таблицы, представления, хранимые процедуры, функции, пользователи, роли и многое другое. Пользователь может просматривать, извлекать и полностью управлять этими объектами, используя контекстное меню и другие функции Object Explorer.

Для бесплатной Express версии SQL Server существует упрощенная версия SSMS под названием SQL Server Management Studio Express. Однако, она имеет ограничения по сравнению с полной версией. SSMS Express не поддерживает ряд компонентов, таких как Analysis Services, Integration Services, Notification Services, Reporting Services, а также SQL Server 2005 Mobile Edition.

# 1.6 Entity Framework

**Entity Framework (EF)** — это мощная и гибкая технология объектно-ориентированного доступа к данным, разработанная Microsoft для платформы .NET. Она является object-relational mapping (ORM) решением, которое упрощает взаимодействие между объектно-ориентированным кодом и реляционными базами данных. EF позволяет разработчикам работать с данными, представленными в базе данных, как с обычными объектами .NET, что повышает продуктивность и снижает количество шаблонного кода, необходимого для доступа к данным.

EF предоставляет два основных способа взаимодействия с данными: LINQ to Entities и Entity SQL. LINQ to Entities позволяет использовать синтаксис LINQ (Language Integrated Query) для запросов к базе данных. Разработчики могут писать запросы на C# или VB.NET, которые автоматически преобразуются в SQL-запросы, выполняемые на сервере базы данных. Entity SQL — это язык, подобный SQL, который позволяет выполнять запросы к концептуальным моделям данных в Entity Framework. Он предоставляет более низкоуровневый контроль над формированием запросов, чем LINQ to Entities.

EF облегчает создание веб-решений благодаря интеграции с различными технологиями .NET. ADO.NET Data Services (ранее известная как Astoria) предоставляет возможность создавать RESTful сервисы, которые позволяют обращаться к данным через HTTP. Кроме того, EF часто используется в связке с Windows Communication Foundation (WCF) и Windows Presentation Foundation (WPF) для построения многоуровневых приложений, реализуя шаблоны проектирования Model-View-Controller (MVC), Model-View-Presenter (MVP) или Model-View-ViewModel (MVVM).

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

# 2.1 Разработка диаграммы ERD

**ER-модель** (от [англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Entity-Relationship model, модель «сущность — связь») — [модель данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), позволяющая описывать [концептуальные схемы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0) [предметной области](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C).

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма «сущность-связь» ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Entity-Relationship diagram, ERD, ER-диаграмма).

**Назначение:**

ER-модель используется при высокоуровневом (концептуальном) [проектировании баз данных](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B0%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время [проектирования баз данных](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B0%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) происходит преобразование схемы, созданной на основе ER-модели, в конкретную [схему базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) на основе выбранной модели данных ([реляционной](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), [объектной](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85&action=edit&redlink=1), [сетевой](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) или др.).

**Элементы:**

Для построения каждой диаграммы взаимосвязи объектов используется набор специальных символов. Чаще всего используются следующие символы:

**Прямоугольники**. Представляют родительские объекты. С ними часто связаны слабые объекты (двойные прямоугольники), зависящие от них.

**Овалы.** Представляют атрибуты. Они описывают характеристики объекта.

**Ромбы.** Используются для отображения взаимосвязей между объектами.

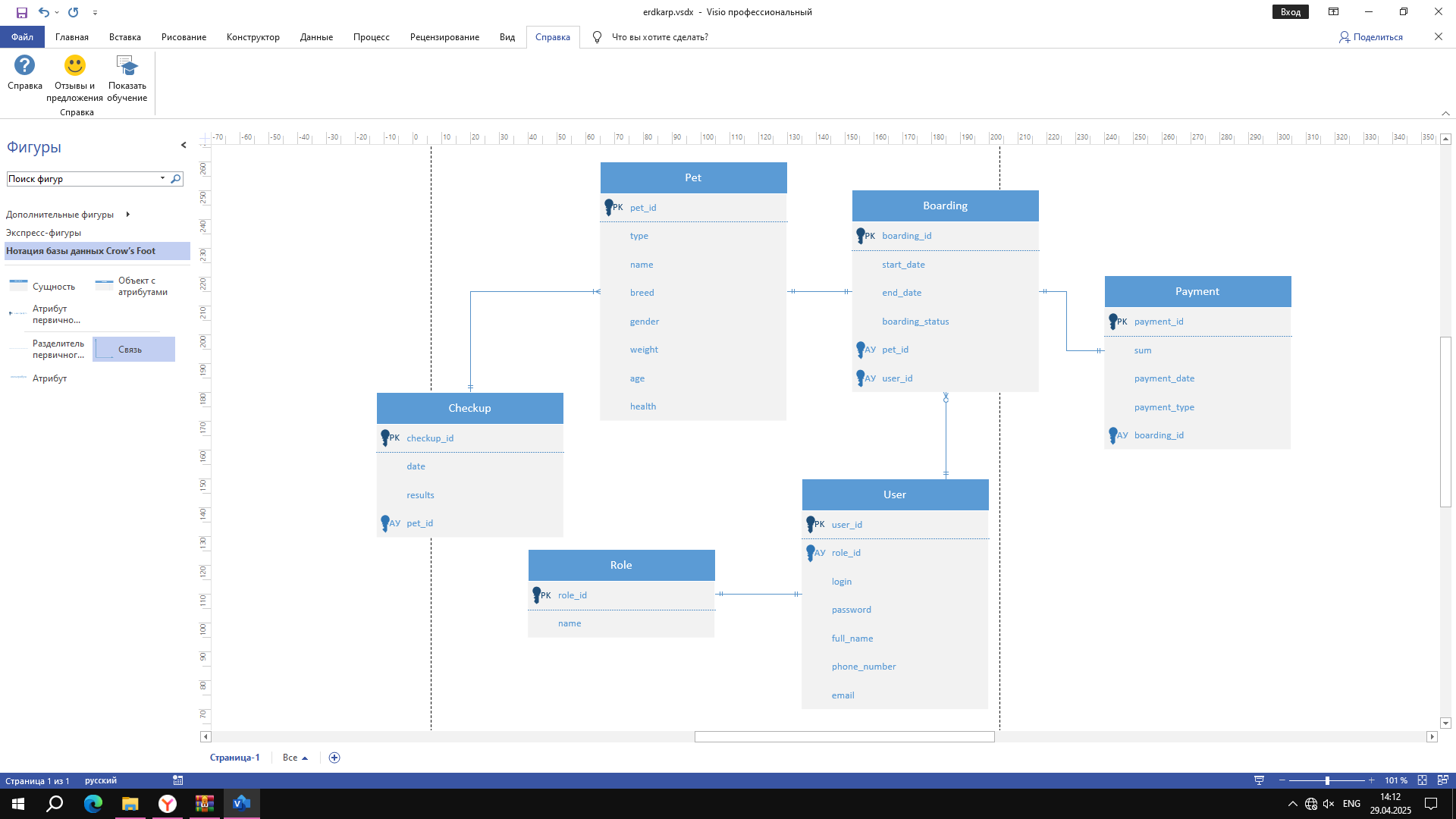
**Линии и стрелки.** Используются, чтобы показать, какие объекты связаны друг с другом.

**Показатели мощности связей.** Отражают объем данных, переходящих от одного объекта к другому.

**Компоненты:**

Объекты обычно отображаются в виде прямоугольников и представляют собой существительные (например, предметы, люди, понятия или события). Использование разных символов позволяет отображать разные типы объектов. Так, сильные объекты обычно передаются прямоугольниками, а слабые — двойными прямоугольниками.

**Атрибуты**— это характеристики объекта. Они определяются в классе и могут быть уникальными для каждого объекта. Атрибуты позволяют хранить состояние объекта и его характеристики.

**Взаимосвязи.** На ER-диаграммах соединительные линии и стрелки имеют метки. Метки указывают на характер взаимодействия объектов, демонстрируя разработчику взаимосвязи между ними. Разные типы соединительных линий также имеют разное значение и показывают, какой объем данных передается от одного объекта к другому.

*Р**ис. 2.1 ER Диаграмма (Entity-Relationship Diagram)*

На рис. 2.1 показана ER диаграмма информационной системы «Сервис передержки животных». Она показывает сущности и как они связаны между собой внутри. Сущностями являются: питомец, пользователь, роль, передержка, осмотр, оплата.

# 2.2 Разработка базы данных

Разработка базы данных будет происходить в программе SSMS (SQL Server Management Studio).

**SQL Server Management Studio (SSMS)** — это мощная и многофункциональная интегрированная среда, предназначенная для управления инфраструктурой Microsoft SQL на всех уровнях. С её помощью можно получать доступ к серверам, конфигурировать их, администрировать, разрабатывать и управлять всеми ключевыми компонентами, включая SQL Server, базы данных Azure SQL, управляемые экземпляры Azure SQL, SQL Server на базе виртуальных машин в Azure, а также Azure Synapse Analytics.

SSMS представляет собой универсальный инструмент, объединяющий в себе удобный графический интерфейс и расширенные редакторы сценариев, что позволяет эффективно работать как начинающим специалистам, так и опытным администраторам баз данных и разработчикам. Эта среда предоставляет все необходимые средства для создания запросов, анализа данных, мониторинга производительности, настройки безопасности и автоматизации задач администрирования.

Кроме того, SQL Server Management Studio поддерживает работу с объектами служб Analysis Services (SSAS), предоставляя возможность выполнять такие действия, как резервное копирование, восстановление и обработка кубов и других аналитических объектов.

*Рис. 2.1 ER Диаграмма (Entity-Relationship Diagram)*

В данной таблице используется 6 таблиц базы данных. Сущность User (Пользователь) имеет атрибуты: user\_id (первичный ключ), role\_id (внешний ключ), login (логин), password (пароль), full\_name (ФИО), phone\_number (номер телефона), email (электронная почта).

Сущность Role (Роль) имеет атрибуты: role\_id (первичный ключ), name (название роли).

Сущность Payment (Оплата) имеет атрибуты: payment\_id (первичный ключ), sum (сумма оплаты), payment\_date (дата оплаты), payment\_type (тип оплаты), boarding\_id (внешний ключ), user\_id (внешний ключ).

Сущность Boarding (Передержка) имеет атрибуты: boarding\_id (первичный ключ), start\_date (дата начала), end\_date (дата конца), boarding\_status (статус передержки), pet\_id (внешний ключ), user\_id (внешний ключ).

Сущность Pet (Питомец) имеет атрибуты: pet\_id (первичный ключ), type (тип животного), name (кличка), breed (порода), gender (пол), weight (вес), age (возраст), health (здоровье).

Сущность Checkup (Осмотр) имеет атрибуты: checkup\_id (первичный ключ), date (дата осмотра), results (результаты осмотра), pet\_id (внешний ключ).

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

# 3.1 Программирование информационной системы

# 3.1.1 Разработка модуля «Авторизация»

*Рис. 3.1.1.1 Окно «Авторизация»*

Методы проверки логина, пароля и капчи:

public authoriz()

{

InitializeComponent();

GenerateNewCaptcha();

}

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

{

//проверка заполнения всех полей

if (!CheckRequiredFields())

{

ShowErrorMessage("Все поля обязательны для заполнения!");

return;

}

if (txtCaptcha.Text != lblCaptcha.Text)

{

ShowErrorMessage("Неверная капча! Попробуйте еще раз.");

GenerateNewCaptcha();

return;

}

if (IsValidCredentials(txtUsername.Text, txtPassword.Password))

{

MessageBoxResult result = MessageBox.Show(

"Здравствуйте, Мария!",

"Успешная авторизация",

MessageBoxButton.OK,

MessageBoxImage.Information);

if (result == MessageBoxResult.OK)

{

OpenMainWindow();

this.Close();

}

}

else

{

ShowErrorMessage("Неверный логин или пароль!");

GenerateNewCaptcha();

}

}

private bool CheckRequiredFields()

{

return !string.IsNullOrWhiteSpace(txtUsername.Text) &&

!string.IsNullOrWhiteSpace(txtPassword.Password) &&

!string.IsNullOrWhiteSpace(txtCaptcha.Text);

}

private bool IsValidCredentials(string login, string password)

{

return login == "маря1989" && password == "маря1989";

}

private void ShowErrorMessage(string message)

{

MessageBox.Show(

message,

"Ошибка авторизации",

MessageBoxButton.OK,

MessageBoxImage.Error);

}

private void OpenMainWindow()

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

}

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

{

GenerateNewCaptcha();

}

private void GenerateNewCaptcha()

{

const string chars = "ABCDEFGHJKLMNPQRSTUVWXYZ23456789";

var random = new Random();

var captcha = new StringBuilder(6);

for (int i = 0; i < 6; i++)

{

captcha.Append(chars[random.Next(chars.Length)]);

}

lblCaptcha.Text = captcha.ToString();

txtCaptcha.Clear();

txtCaptcha.Focus();

}

private void Border\_MouseDown(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

if (e.ChangedButton == System.Windows.Input.MouseButton.Left)

this.DragMove();

}

}

}

В данном модуле было разработано окно авторизации и капча с проверкой логина и пароля пользователя.

# 3.1.2 Разработка модуля «Главное меню»

*Рис. 3.1.1.2 Окно «Главное меню»*

using System;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Controls;

namespace boarding.Views

{

public partial class dashboardpage : Page

{

private string connectionString = "Server=localhost\\SQLEXPRESS;Database=boarding\_app;Trusted\_Connection=True;";

public dashboardpage()

{

InitializeComponent();

LoadStatistics();

}

private void LoadStatistics()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand petCmd = new SqlCommand("SELECT COUNT(\*) FROM Pet", connection);

int petCount = (int)petCmd.ExecuteScalar();

AnimalsCountText.Text = petCount.ToString();

SqlCommand checkupCmd = new SqlCommand(

"SELECT COUNT(\*) FROM Checkup WHERE date >= @startDate", connection);

checkupCmd.Parameters.AddWithValue("@startDate", DateTime.Today.AddMonths(-1));

int checkupCount = (int)checkupCmd.ExecuteScalar();

InspectionsCountText.Text = checkupCount.ToString();

}

}

catch (Exception ex)

{

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

}

}

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

{

try

{

var vetWindow = new vetpage();

vetWindow.Show();

Window.GetWindow(this)?.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при открытии ветеринарной панели: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

В данном модуле было разработано главное меню с кнопками навигации по страницам программы.

# 3.1.3 Разработка модуля «Список животных»

*Рис. 3.1.1.3 Окно «Список животных»*

Методы создания таблицы и загрузки данных в эту таблицу с возможностью изменения и добавления новых данных:

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Controls;

namespace boarding.Views

{

public partial class animalspage : Page

{

private string connectionString = "Server=localhost\\SQLEXPRESS;Database=boarding\_app;Trusted\_Connection=True;";

public animalspage()

{

InitializeComponent();

LoadPetsData();

}

private void LoadPetsData()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT \* FROM Pet";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dt = new DataTable();

adapter.Fill(dt);

petsDataGrid.ItemsSource = dt.DefaultView;

}

}

catch (Exception ex)

{

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

}

}

private int GetNextPetId()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT ISNULL(MAX(pet\_id), 0) + 1 FROM Pet";

SqlCommand command = new SqlCommand(query, connection);

int nextId = (int)command.ExecuteScalar();

return nextId;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении нового ID: {ex.Message}");

return 1; // Если ошибка, вернём 1 по умолчанию

}

}

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

{

int newPetId = GetNextPetId();

var addWindow = new AddEditPetWindow(newPetId);

if (addWindow.ShowDialog() == true)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"INSERT INTO Pet (pet\_id, type, name, breed, gender, weight, age, health)

VALUES (@pet\_id, @type, @name, @breed, @gender, @weight, @age, @health)";

SqlCommand cmd = new SqlCommand(query, connection);

cmd.Parameters.AddWithValue("@pet\_id", addWindow.PetId);

cmd.Parameters.AddWithValue("@type", addWindow.PetType);

cmd.Parameters.AddWithValue("@name", addWindow.PetName);

cmd.Parameters.AddWithValue("@breed", addWindow.Breed);

cmd.Parameters.AddWithValue("@gender", addWindow.Gender);

cmd.Parameters.AddWithValue("@weight", addWindow.Weight);

cmd.Parameters.AddWithValue("@age", addWindow.Age);

cmd.Parameters.AddWithValue("@health", addWindow.Health);

cmd.ExecuteNonQuery();

}

LoadPetsData();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка добавления: {ex.Message}");

}

}

}

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

{

if (petsDataGrid.SelectedItem == null)

{

MessageBox.Show("Выберите животное для редактирования");

return;

}

DataRowView row = (DataRowView)petsDataGrid.SelectedItem;

var editWindow = new AddEditPetWindow(

row["type"].ToString(),

row["name"].ToString(),

row["breed"].ToString(),

row["gender"].ToString(),

Convert.ToDouble(row["weight"]),

Convert.ToInt32(row["age"]),

row["health"].ToString(),

Convert.ToInt32(row["pet\_id"]));

if (editWindow.ShowDialog() == true)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"UPDATE Pet SET

type = @type,

name = @name,

breed = @breed,

gender = @gender,

weight = @weight,

age = @age,

health = @health

WHERE pet\_id = @pet\_id";

SqlCommand cmd = new SqlCommand(query, connection);

cmd.Parameters.AddWithValue("@type", editWindow.PetType);

cmd.Parameters.AddWithValue("@name", editWindow.PetName);

cmd.Parameters.AddWithValue("@breed", editWindow.Breed);

cmd.Parameters.AddWithValue("@gender", editWindow.Gender);

cmd.Parameters.AddWithValue("@weight", editWindow.Weight);

cmd.Parameters.AddWithValue("@age", editWindow.Age);

cmd.Parameters.AddWithValue("@health", editWindow.Health);

cmd.Parameters.AddWithValue("@pet\_id", editWindow.PetId);

cmd.ExecuteNonQuery();

}

LoadPetsData();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка редактирования: {ex.Message}");

}

}

}

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

{

LoadPetsData();

}

}

}

using System.Windows;

namespace boarding.Views

{

public partial class AddEditPetWindow : Window

{

public string PetType => txtType.Text;

public string PetName => txtName.Text;

public string Breed => txtBreed.Text;

public string Gender => txtGender.Text;

public double Weight => double.Parse(txtWeight.Text);

public int Age => int.Parse(txtAge.Text);

public string Health => txtHealth.Text;

public int PetId { get; private set; }

public AddEditPetWindow()

{

InitializeComponent();

Title = "Добавить животное";

}

public AddEditPetWindow(int petId) : this()

{

PetId = petId;

}

public AddEditPetWindow(string type, string name, string breed, string gender,

double weight, int age, string health, int petId)

: this()

{

txtType.Text = type;

txtName.Text = name;

txtBreed.Text = breed;

txtGender.Text = gender;

txtWeight.Text = weight.ToString();

txtAge.Text = age.ToString();

txtHealth.Text = health;

PetId = petId;

Title = "Изменить животное";

}

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

{

if (ValidateInput())

{

DialogResult = true;

Close();

}

}

private bool ValidateInput()

{

if (string.IsNullOrWhiteSpace(PetType) ||

string.IsNullOrWhiteSpace(PetName) ||

string.IsNullOrWhiteSpace(Breed) ||

string.IsNullOrWhiteSpace(Gender))

{

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

return false;

}

if (!double.TryParse(txtWeight.Text, out \_) ||

!int.TryParse(txtAge.Text, out \_))

{

MessageBox.Show("Вес и возраст должны быть числами");

return false;

}

return true;

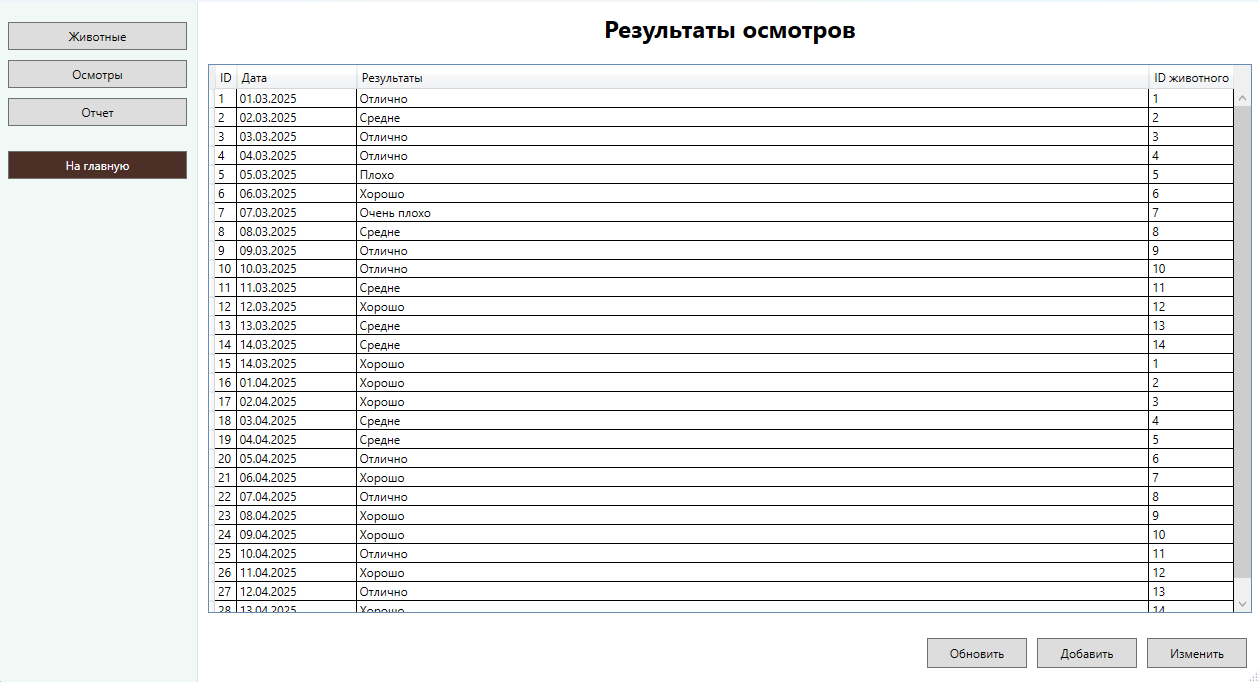
}

}

}

В данном модуле была разработана таблица питомцев, которая берет данные из базы данных и имеется возможность их изменить, добавить и отсортировать.

# 3.1.4 Разработка модуля «Осмотры»

 *Рис. 3.1.1.3 Окно «Список животных»*

Методы создания таблицы и загрузки данных в эту таблицу с возможностью изменения и добавления новых данных:

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Controls;

namespace boarding.Views

{

public partial class examinationspage : Page

{

private string connectionString = "Server=localhost\\SQLEXPRESS;Database=boarding\_app;Trusted\_Connection=True;";

public examinationspage()

{

InitializeComponent();

LoadCheckupsData();

}

private void LoadCheckupsData()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT \* FROM Checkup";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dt = new DataTable();

adapter.Fill(dt);

checkupsDataGrid.ItemsSource = dt.DefaultView;

}

}

catch (Exception ex)

{

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

}

}

private int GetNextCheckupId()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT ISNULL(MAX(checkup\_id), 0) + 1 FROM Checkup";

SqlCommand command = new SqlCommand(query, connection);

return (int)command.ExecuteScalar();

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при получении нового ID: {ex.Message}");

return 1;

}

}

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

{

int newCheckupId = GetNextCheckupId();

var addWindow = new AddEditCheckupWindow(newCheckupId);

if (addWindow.ShowDialog() == true)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"INSERT INTO Checkup (checkup\_id, date, results, pet\_id)

VALUES (@checkup\_id, @date, @results, @pet\_id)";

SqlCommand cmd = new SqlCommand(query, connection);

cmd.Parameters.AddWithValue("@checkup\_id", addWindow.CheckupId);

cmd.Parameters.AddWithValue("@date", addWindow.CheckupDate);

cmd.Parameters.AddWithValue("@results", addWindow.Results);

cmd.Parameters.AddWithValue("@pet\_id", addWindow.PetId);

cmd.ExecuteNonQuery();

}

LoadCheckupsData();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка добавления: {ex.Message}");

}

}

}

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

{

if (checkupsDataGrid.SelectedItem == null)

{

MessageBox.Show("Выберите осмотр для редактирования");

return;

}

DataRowView row = (DataRowView)checkupsDataGrid.SelectedItem;

var editWindow = new AddEditCheckupWindow(

Convert.ToDateTime(row["date"]),

row["results"].ToString(),

Convert.ToInt32(row["pet\_id"]),

Convert.ToInt32(row["checkup\_id"]));

if (editWindow.ShowDialog() == true)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"UPDATE Checkup SET

date = @date,

results = @results,

pet\_id = @pet\_id

WHERE checkup\_id = @checkup\_id";

SqlCommand cmd = new SqlCommand(query, connection);

cmd.Parameters.AddWithValue("@date", editWindow.CheckupDate);

cmd.Parameters.AddWithValue("@results", editWindow.Results);

cmd.Parameters.AddWithValue("@pet\_id", editWindow.PetId);

cmd.Parameters.AddWithValue("@checkup\_id", editWindow.CheckupId);

cmd.ExecuteNonQuery();

}

LoadCheckupsData();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка редактирования: {ex.Message}");

}

}

}

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

{

LoadCheckupsData();

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

namespace boarding.Views

{

public partial class AddEditCheckupWindow : Window

{

private string connectionString = "Server=localhost\\SQLEXPRESS;Database=boarding\_app;Trusted\_Connection=True;";

public DateTime CheckupDate => dpDate.SelectedDate ?? DateTime.Now;

public string Results => txtResults.Text;

public int PetId => (cbPets.SelectedItem as DataRowView)?["pet\_id"] as int? ?? 0;

public int CheckupId { get; private set; }

public AddEditCheckupWindow()

{

InitializeComponent();

dpDate.SelectedDate = DateTime.Today;

Title = "Добавить осмотр";

LoadPets();

}

public AddEditCheckupWindow(int checkupId) : this()

{

CheckupId = checkupId;

}

public AddEditCheckupWindow(DateTime date, string results, int petId, int checkupId)

: this()

{

dpDate.SelectedDate = date;

txtResults.Text = results;

foreach (DataRowView item in cbPets.Items)

{

if ((int)item["pet\_id"] == petId)

{

cbPets.SelectedItem = item;

break;

}

}

CheckupId = checkupId;

Title = "Изменить осмотр";

}

private void LoadPets()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT pet\_id FROM Pet ORDER BY pet\_id";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dt = new DataTable();

adapter.Fill(dt);

cbPets.ItemsSource = dt.DefaultView;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка загрузки списка животных: {ex.Message}");

}

}

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

{

if (ValidateInput())

{

DialogResult = true;

Close();

}

}

private bool ValidateInput()

{

if (dpDate.SelectedDate == null ||

string.IsNullOrWhiteSpace(Results))

{

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

return false;

}

if (cbPets.SelectedItem == null)

{

MessageBox.Show("Выберите животное");

return false;

}

return true;

}

}

}

В данном модуле была разработана таблица питомцев, которая берет данные из базы данных и имеется возможность их изменить, добавить и отсортировать.

# 3.1.5 Разработка модуля «Отчет»

*Рис. 3.1.1.4 Окно «Отчет»*

Код экспорта данных из базы данных в Word, PDF и в Excel документ:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.IO;

using System.Windows;

using System.Windows.Controls;

using Word = Microsoft.Office.Interop.Word;

using Excel = Microsoft.Office.Interop.Excel;

namespace boarding.Views

{

public partial class ReportPage : Page

{

private string connectionString = "Server=localhost\\SQLEXPRESS;Database=boarding\_app;Trusted\_Connection=True;";

public ReportPage()

{

InitializeComponent();

dpStartDate.SelectedDate = DateTime.Today.AddMonths(-1);

dpEndDate.SelectedDate = DateTime.Today;

this.Loaded += ReportPage\_Loaded;

}

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

{

cbAllData.Checked += cbAllData\_Checked;

cbAllData.Unchecked += cbAllData\_Checked;

}

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

{

bool isChecked = cbAllData?.IsChecked ?? false;

if (cbPets != null) cbPets.IsChecked = isChecked;

if (cbCheckups != null) cbCheckups.IsChecked = isChecked;

if (cbPets != null) cbPets.IsEnabled = !isChecked;

if (cbCheckups != null) cbCheckups.IsEnabled = !isChecked;

}

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

{

try

{

if (!dpStartDate.SelectedDate.HasValue || !dpEndDate.SelectedDate.HasValue)

{

MessageBox.Show("Укажите период отчета", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

if (dpStartDate.SelectedDate > dpEndDate.SelectedDate)

{

MessageBox.Show("Дата начала не может быть позже даты окончания", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

var selectedTables = new List<string>();

if (cbPets.IsChecked ?? false) selectedTables.Add("Pets");

if (cbCheckups.IsChecked ?? false) selectedTables.Add("Checkups");

if (selectedTables.Count == 0)

{

MessageBox.Show("Выберите хотя бы один тип данных для отчета", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

string format = rbWord.IsChecked ?? false ? "Word" :

rbExcel.IsChecked ?? false ? "Excel" : "PDF";

bool isTableFormat = rbTable.IsChecked ?? false;

var saveFileDialog = new Microsoft.Win32.SaveFileDialog();

saveFileDialog.Filter = format == "Word" ? "Word Documents|\*.docx" :

format == "Excel" ? "Excel Files|\*.xlsx" : "PDF Files|\*.pdf";

saveFileDialog.FileName = $"Отчет\_по\_животным\_{DateTime.Now:yyyyMMdd}";

if (saveFileDialog.ShowDialog() == true)

{

if (format == "Excel")

{

ExportToExcel(saveFileDialog.FileName, selectedTables,

dpStartDate.SelectedDate, dpEndDate.SelectedDate);

}

else

{

ExportToWord(saveFileDialog.FileName, selectedTables,

dpStartDate.SelectedDate, dpEndDate.SelectedDate,

isTableFormat, format == "PDF");

}

MessageBox.Show("Отчет успешно сформирован", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

catch (Exception ex)

{

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

}

}

private void ExportToWord(string filePath, List<string> selectedTables,

DateTime? startDate, DateTime? endDate,

bool isTableFormat, bool saveAsPdf)

{

Word.Application wordApp = null;

Word.Document doc = null;

try

{

wordApp = new Word.Application();

wordApp.Visible = false;

wordApp.DisplayAlerts = Word.WdAlertLevel.wdAlertsNone;

doc = wordApp.Documents.Add();

var title = doc.Paragraphs.Add();

title.Range.Text = "Отчет по животным и осмотрам";

title.Range.Font.Bold = 1;

title.Range.Font.Size = 16;

title.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

title.Range.InsertParagraphAfter();

var period = doc.Paragraphs.Add();

period.Range.Text = $"Период: {startDate:dd.MM.yyyy} - {endDate:dd.MM.yyyy}";

period.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

period.Range.InsertParagraphAfter();

var date = doc.Paragraphs.Add();

date.Range.Text = $"Сформирован: {DateTime.Now:dd.MM.yyyy HH:mm}";

date.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

date.Range.InsertParagraphAfter();

doc.Paragraphs.Add().Range.InsertParagraphAfter();

System.Data.DataTable petsTable = null;

System.Data.DataTable checkupsTable = null;

using (var connection = new SqlConnection(connectionString))

{

connection.Open();

if (selectedTables.Contains("Pets"))

{

var petsAdapter = new SqlDataAdapter(

"SELECT pet\_id, type, name, breed, gender, weight, age, health FROM Pet",

connection);

petsTable = new System.Data.DataTable();

petsAdapter.Fill(petsTable);

}

if (selectedTables.Contains("Checkups"))

{

string checkupQuery = @"SELECT c.checkup\_id, c.date, c.results,

p.pet\_id, p.name as pet\_name

FROM Checkup c

JOIN Pet p ON c.pet\_id = p.pet\_id";

if (startDate.HasValue && endDate.HasValue)

{

checkupQuery += $" WHERE c.date BETWEEN '{startDate:yyyy-MM-dd}' AND '{endDate:yyyy-MM-dd}'";

}

var checkupsAdapter = new SqlDataAdapter(checkupQuery, connection);

checkupsTable = new System.Data.DataTable();

checkupsAdapter.Fill(checkupsTable);

}

}

foreach (var table in selectedTables)

{

var section = doc.Paragraphs.Add();

section.Range.Text = table == "Pets" ? "Данные о животных" : "Данные об осмотрах";

section.Range.Font.Bold = 1;

section.Range.Font.Size = 14;

section.Range.InsertParagraphAfter();

if (isTableFormat)

{

if (table == "Pets" && petsTable != null)

{

ExportDataTableToWord(doc, petsTable);

}

else if (table == "Checkups" && checkupsTable != null)

{

ExportDataTableToWord(doc, checkupsTable);

}

}

else

{

if (table == "Pets" && petsTable != null)

{

ExportDataToWordText(doc, petsTable);

}

else if (table == "Checkups" && checkupsTable != null)

{

ExportDataToWordText(doc, checkupsTable);

}

}

doc.Paragraphs.Add().Range.InsertParagraphAfter();

}

Word.WdSaveFormat format = saveAsPdf ? Word.WdSaveFormat.wdFormatPDF : Word.WdSaveFormat.wdFormatDocumentDefault;

c.SaveAs2(filePath, format);

object doNotSave = Word.WdSaveOptions.wdDoNotSaveChanges;

doc.Close(ref doNotSave);

wordApp.Quit();

System.Diagnostics.Process.Start(filePath);

}

catch (Exception)

{

if (doc != null)

{

object doNotSave = Word.WdSaveOptions.wdDoNotSaveChanges;

doc.Close(ref doNotSave);

}

if (wordApp != null) wordApp.Quit();

throw;

}

}

private void ExportDataTableToWord(Word.Document doc, System.Data.DataTable dataTable)

{

Word.Table table = doc.Tables.Add(doc.Range(doc.Content.End - 1),

dataTable.Rows.Count + 1,

dataTable.Columns.Count);

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

{

table.Cell(1, i + 1).Range.Text = dataTable.Columns[i].ColumnName;

}

for (int row = 0; row < dataTable.Rows.Count; row++)

{

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

{

table.Cell(row + 2, col + 1).Range.Text = dataTable.Rows[row][col].ToString();

}

}

table.Range.Font.Size = 10;

table.Rows[1].Range.Font.Bold = 1;

table.Rows[1].Range.Shading.BackgroundPatternColor = Word.WdColor.wdColorGray15;

table.Borders.Enable = 1;

table.AllowAutoFit = true;

table.AutoFitBehavior(Word.WdAutoFitBehavior.wdAutoFitContent);

}

private void ExportDataToWordText(Word.Document doc, System.Data.DataTable dataTable)

{

foreach (System.Data.DataRow row in dataTable.Rows)

{

var paragraph = doc.Paragraphs.Add();

foreach (System.Data.DataColumn col in dataTable.Columns)

{

paragraph.Range.Text += $"{col.ColumnName}: {row[col]}\t";

}

paragraph.Range.InsertParagraphAfter();

}

}

private void ExportToExcel(string filePath, List<string> selectedTables,

DateTime? startDate, DateTime? endDate)

{

Excel.Application excelApp = null;

Excel.Workbook workbook = null;

try

{

excelApp = new Excel.Application();

excelApp.Visible = false;

workbook = excelApp.Workbooks.Add();

System.Data.DataTable petsTable = null;

System.Data.DataTable checkupsTable = null;

using (var connection = new SqlConnection(connectionString))

{

connection.Open();

if (selectedTables.Contains("Pets"))

{

var petsAdapter = new SqlDataAdapter(

"SELECT pet\_id, type, name, breed, gender, weight, age, health FROM Pet",

connection);

petsTable = new System.Data.DataTable();

petsAdapter.Fill(petsTable);

}

if (selectedTables.Contains("Checkups"))

{

string checkupQuery = @"SELECT c.checkup\_id, c.date, c.results,

p.pet\_id, p.name as pet\_name

FROM Checkup c

JOIN Pet p ON c.pet\_id = p.pet\_id";

if (startDate.HasValue && endDate.HasValue)

{

checkupQuery += $" WHERE c.date BETWEEN '{startDate:yyyy-MM-dd}' AND '{endDate:yyyy-MM-dd}'";

}

var checkupsAdapter = new SqlDataAdapter(checkupQuery, connection);

checkupsTable = new System.Data.DataTable();

checkupsAdapter.Fill(checkupsTable);

}

}

foreach (var table in selectedTables)

{

Excel.Worksheet worksheet;

if (workbook.Sheets.Count < selectedTables.IndexOf(table) + 1)

{

worksheet = (Excel.Worksheet)workbook.Sheets.Add();

}

else

{

worksheet = (Excel.Worksheet)workbook.Sheets[selectedTables.IndexOf(table) + 1];

}

worksheet.Name = table == "Pets" ? "Животные" : "Осмотры";

System.Data.DataTable dataToExport = table == "Pets" ? petsTable : checkupsTable;

if (dataToExport != null)

{

ExportDataTableToExcel(worksheet, dataToExport);

}

}

while (workbook.Sheets.Count > selectedTables.Count)

{

((Excel.Worksheet)workbook.Sheets[workbook.Sheets.Count]).Delete();

}

workbook.SaveAs(filePath);

System.Diagnostics.Process.Start(filePath);

}

catch (Exception)

{

if (workbook != null) workbook.Close(false);

if (excelApp != null) excelApp.Quit();

throw;

}

finally

{

if (workbook != null) workbook.Close(false);

if (excelApp != null) excelApp.Quit();

}

}

private void ExportDataTableToExcel(Excel.Worksheet worksheet, System.Data.DataTable dataTable)

{

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

{

worksheet.Cells[1, i + 1] = dataTable.Columns[i].ColumnName;

}

for (int row = 0; row < dataTable.Rows.Count; row++)

{

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

{

worksheet.Cells[row + 2, col + 1] = dataTable.Rows[row][col].ToString();

}

}

Excel.Range headerRange = worksheet.Range[worksheet.Cells[1, 1], worksheet.Cells[1, dataTable.Columns.Count]];

headerRange.Font.Bold = true;

worksheet.Columns.AutoFit();

}

}

}

# ЗАКЛЮЧЕНИЕ

В рамках выполнения курсового проекта была разработана автоматизированная информационная система "Сервис передержки животных". При создании системы использовались технологии: платформа .NET, язык программирования C#, WPF для интерфейса, SQL Server, в качестве СУБД, и Entity Framework для работы с данными.

Система включает модули авторизации с проверкой логина, пароля и капчи. функционал для работы с информацией о животных на передержке, а также формирование отчетов с возможностью экспорта в Word и Excel.

Внедрение системы позволит сократить время своевременного заполнения информации до 1-2 минут, а формирование отчетов до 3-4 минут, что значительно повысит эффективность работы архива. Для дальнейшего развития системы рекомендуется интеграция с MS Office и расширение функционала, например. Разработанная информационная система успешно реализована, соответствует поставленным задачам и готова к внедрению, обеспечивая автоматизацию ключевых процессов учета сервиса передержки животных.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

<https://ru.wikipedia.org/wiki/.NET>

<https://ru.wikipedia.org/wiki/C_Sharp>

<https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation>

<https://ru.wikipedia.org/wiki/Microsoft_SQL_Server>

<https://ru.wikipedia.org/wiki/SQL_Server_Management_Studio>

<https://ru.wikipedia.org/wiki/ADO.NET_Entity_Framework>