МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 05 01 03 «Информационные системы и

технологии (издательско-полиграфический комплекс)»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема «Программное средство для физкультурно-оздоровительного центра»

Исполнитель

студент 3 курса группы 1 Кальчевский Д. А.

(Ф.И.О.)

Руководитель работы ассистент Панченко О.Л.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Смелов В.В.

(подпись)

Минск 2023

Содержание

[Введение 3](#_Toc154406828)

[1. Аналитический обзор литературы 4](#_Toc154406829)

[1.1 Обзор аналогичных решений 4](#_Toc154406830)

[1.2 Вывод по разделу 6](#_Toc154406831)

[2.1. Определение требований к программному средству 8](#_Toc154406832)

[3. Проектирование архитектуры проекта 10](#_Toc154406833)

[3.1 Проектирование схемы базы данных 10](#_Toc154406834)

[3.2. Описание средств разработки 12](#_Toc154406835)

[3.3. Проектирование окон приложения 12](#_Toc154406836)

[3.4. Проектирование Use Case диаграммы 13](#_Toc154406837)

[3.5 Вывод по разделу 14](#_Toc154406838)

[4. Разработка функциональной модели и модели данных ПС 15](#_Toc154406839)

[4.1. Используемые паттерны 15](#_Toc154406840)

[4.2. Разработка модели данных 21](#_Toc154406841)

[4.3. Регистрация и авторизация 22](#_Toc154406842)

[4.4. Заявка и оформление абонемента 24](#_Toc154406843)

[4.5. Вывод по разделу 25](#_Toc154406844)

[5. Тестирование 26](#_Toc154406845)

[5.1. Вывод по разделу 28](#_Toc154406846)

[6. Руководство по установке и использованию 29](#_Toc154406847)

[6.1. Установка программного средства 29](#_Toc154406848)

[6.2. Руководство пользователя по работе с программным средством 29](#_Toc154406849)

[6.3. Вывод по разделу 29](#_Toc154406850)

[Список литературы 31](#_Toc154406851)

# **Введение**

Физкультурно-оздоровительный центр является одним из наиболее актуальных и востребованных объектов в современном обществе. В условиях современного образа жизни, характеризующегося сидячей работой и недостатком физической активности, поддержание здоровья и физической формы становится особенно важным. Физкультурно-оздоровительные центры предлагают широкий спектр услуг и возможностей для занятий спортом, фитнесом, аэробикой, плаванием и другими видами физической активности.

Целью данного курсового проекта является изучение и анализ физкультурно-оздоровительного центра как современного объекта, способствующего поддержанию и улучшению физического и психологического здоровья людей. В рамках работы будут рассмотрены основные аспекты организации и функционирования физкультурно-оздоровительных центров, их роль в обществе, а также преимущества и вызовы, с которыми они сталкиваются.

Для достижения поставленной цели будут использованы различные методы исследования, включая анализ литературных источников, а также наблюдение и анализ работы конкретного физкультурно-оздоровительного центра.

Ожидается, что результаты данного курсового проекта помогут лучше понять роль и значение физкультурно-оздоровительных центров в современном обществе, а также предложить рекомендации по их улучшению и развитию.

Реализация приложения осуществляется на языке C# с соблюдением основных принципов ООП и предназначаться для ОС Windows. Реализация базы данных должна осуществляться в рамках системы управления базами данных SQL Server.

1. **Аналитический обзор литературы**

Задача данной учебно-проектной работы заключается в разработке программного средства для физкультурно-оздоровительного центра, а также создании реляционной базы данных, которая будет интегрирована с этим приложением. Главной целью является обеспечение клиентам удобного и быстрого доступа к услугам физкультурно-оздоровительного центра.

* 1. **Обзор аналогичных решений**

При планировании физкультурно-оздоровительного центра необходимо учесть существующие аналоги и альтернативные решения, которые также направлены на достижение подобных целей. Этот обзор поможет выявить преимущества и недостатки конкурирующих центров, а также определить, каким образом наш центр может выделяться среди них.

Аналог №1: Приложение «M-Foc» [1]

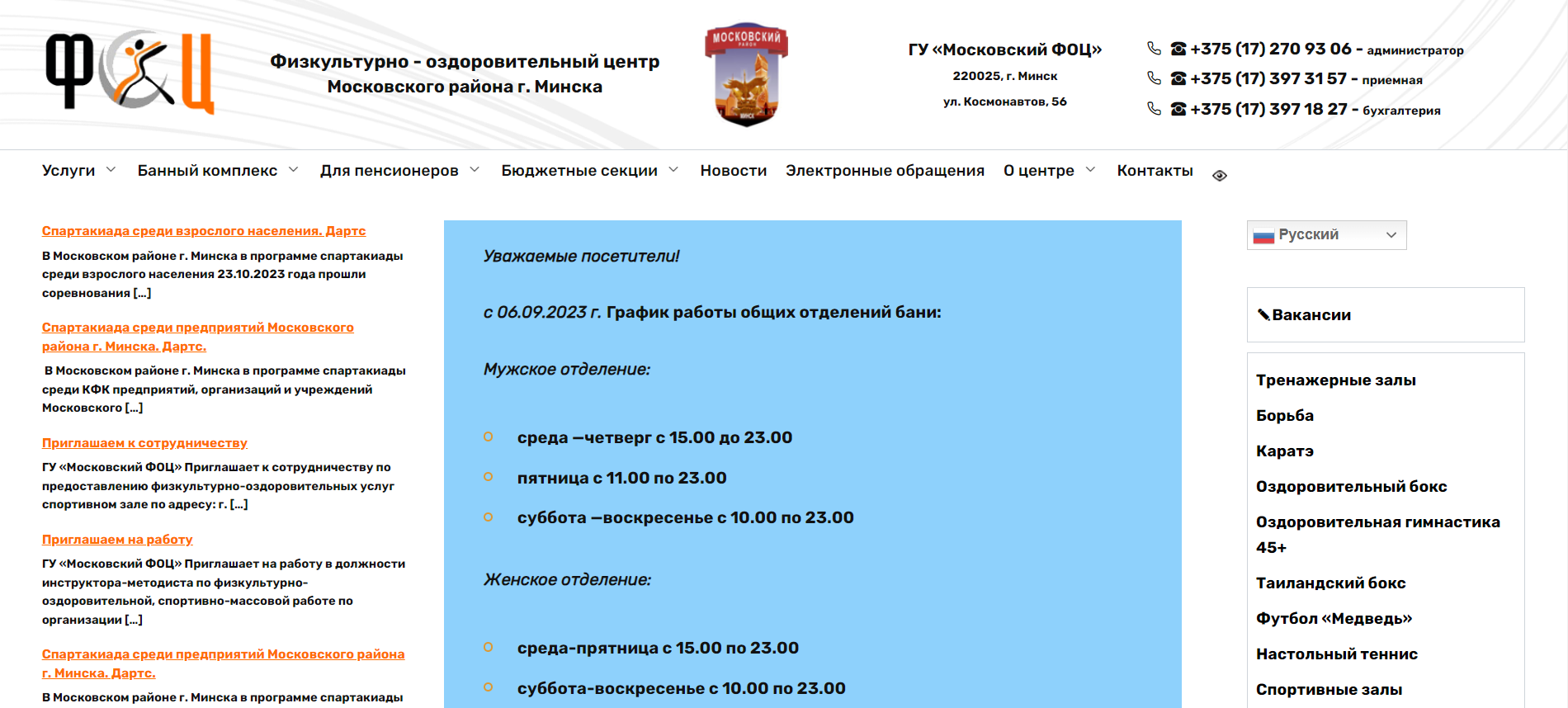


Рисунок 1.1 – Интерфейс приложение «M-Foc»

На рисунке 1.1 представлен интерфейс приложения «M-Foc». Интерфейс оформлен в сдержанной и профессиональной цветовой гамме, что создает впечатление надежности и серьезности комплекса.

Главная страница содержит информацию о различных разделах и услугах, предлагаемых приложением.

В разделе "Спортивные секции" можно ознакомиться с разнообразными видами спорта, которые можно заниматься в комплексе. Также на сайте представлены разделы "Банный комплекс", "Тренажерный зал", где можно узнать подробности о соответствующих услугах и предложениях. Кроме того, в приложение можно найти контактную информацию для связи, а также информацию о графике работы и приглашениях на работу.

Преимущества приложение: логичная навигация, информативность, чёткость и удобство, адаптивность, контактная информация.

Несмотря на преимущества, у приложение также есть некоторые недостатки: отсутствие мультиязычности, отсутствие отзывов и рейтингов, отсутствие интерактивности.

В целом, приложение предоставляет базовую информацию о предложениях, но требует дальнейшего развития для улучшения пользовательского опыта.

Аналог №2: Приложение «YESTODAY» [2]

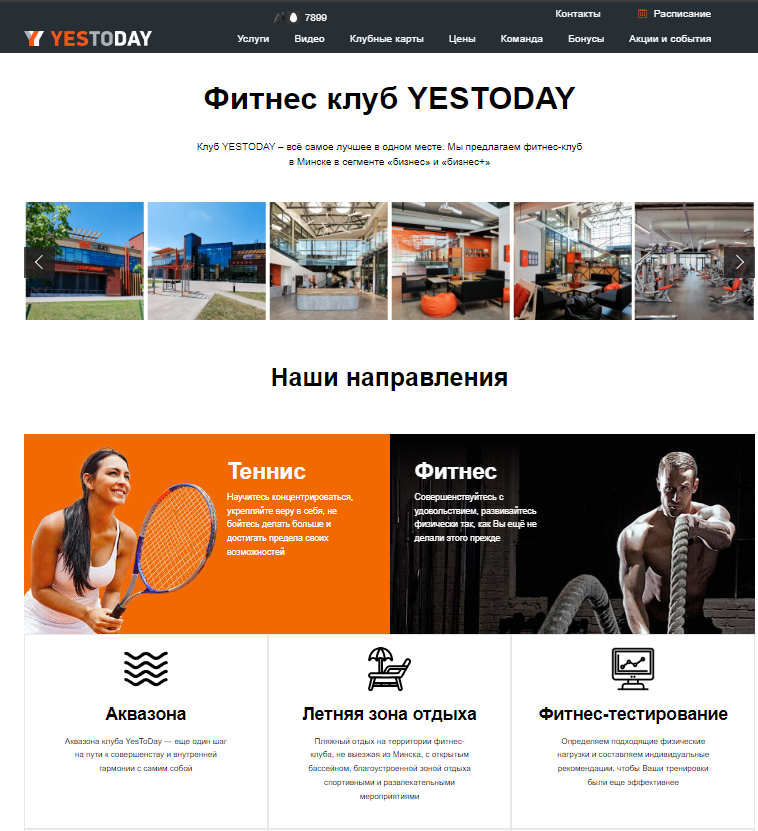


Рисунок 1.2 – Интерфейс приложение «YESTODAY»

На рисунке 1.2 представлен интерфейс приложения «YESTODAY». В приложение присутствует логотип и название оздоровительного центра, а также разделы с информацией о предлагаемых услугах, командной тренеров и отзывами клиентов.

Преимущества приложение: Отзывчивость, удобство использования, визуальный дизайн, информативность.

Недостатки приложение: Медленная загрузка, отсутствие обновление, реклама.

Аналог №3: Приложение «Well-B» [3]

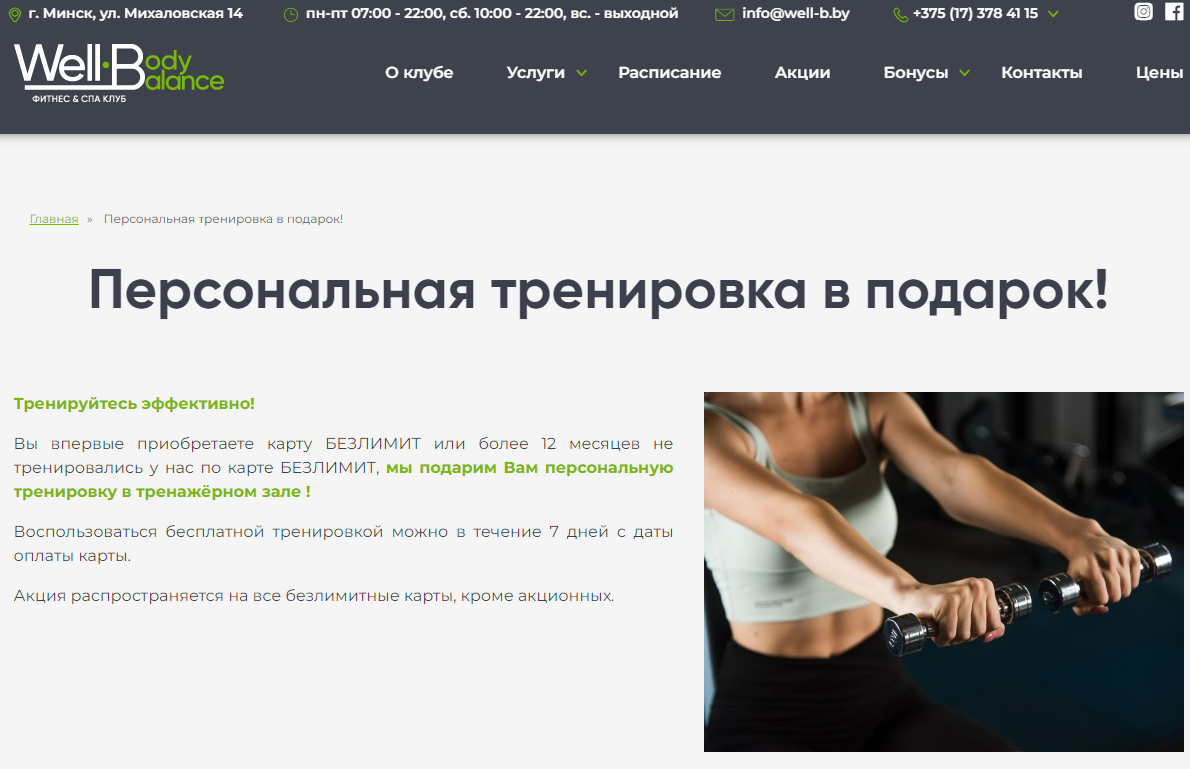


Рисунок 1.3 – Интерфейс приложение «Well-B»

На рисунке 1.3 представлен интерфейс приложения «Well-B».

В приложение представлена информация о контактах центра, его расписании, услугах, новостях, акциях, скидках и ценах. Кроме того, в приложении можно ознакомиться с информацией о центре, его аква-зоне, спа-услугах, групповых занятиях, тренажерном зале, персональных тренировках, детском фитнесе, сертификатах и предложениях для корпоративных клиентов. Также на сайте есть информация о программе лояльности и бонусах для партнеров центра. Особое предложение на сайте - при покупке безлимитной карты или при отсутствии тренировок на протяжении 12 месяцев, предоставляется бесплатная персональная тренировка в тренажерном зале.

Преимущества приложение: удобное расписание, акции и бонусы, разнообразие услуг.

Недостатки приложение: высокая стоимость, ограниченные услуги для детей.

* 1. **Вывод по разделу**

Из описанного раздела можно сделать вывод, что для успешного программного приложения необходимо учитывать следующие аспекты:

Интуитивно понятный интерфейс: Приложение должно иметь простой и легкий в использовании интерфейс, который позволит пользователям быстро ориентироваться и находить нужные функции. Это обеспечит удобство и комфорт при работе с приложением.

Расширенный функционал: Приложение должно предоставлять широкий спектр функций и возможностей, соответствующих потребностям пользователей. Оно должно позволять эффективно и удобно выполнять основные задачи. Это обеспечит удовлетворение потребностей клиентов и повысит их удовлетворенность.

Высокая производительность: Приложение должно работать быстро и отзывчиво. Загрузка страниц и выполнение операций должны происходить плавно и без задержек. Это минимизирует время ожидания пользователей и создаст позитивное впечатление о приложении.

При соблюдении всех этих аспектов, разработанное программное приложение будет представлять готовый продукт, который исчерпывающе предоставляет функционал для клиентов. Оно будет обеспечивать удобство, эффективность и удовлетворение потребностей пользователей, что является ключевыми факторами успеха.

В завершение разработки программного приложения также крайне важными факторами являются непрерывная поддержка и обновление. Регулярные обновления не только устраняют возможные ошибки и улучшают безопасность, но и добавляют новые функциональности, следуя требованиям рынка и ожиданиям пользователей.

Кроме того, важно предусмотреть механизмы обратной связи, чтобы пользователи могли сообщать о проблемах, предлагать улучшения и задавать вопросы. Активная взаимодействие с пользовательским сообществом поможет лучше понять потребности пользователей и адаптировать приложение под их требования.

Наконец, эффективная маркетинговая стратегия также сыграет важную роль в успехе программного продукта. Предоставление четкой информации о преимуществах приложения, привлечение внимания целевой аудитории и создание положительного имиджа могут существенно повысить его популярность и конкурентоспособность на рынке. Таким образом, для обеспечения успеха программного приложения необходимо не только разработать функциональный и удобный продукт, но и эффективно управлять им в долгосрочной перспективе.

1. **Анализ требований к программному продукту т разработка функциональных требований**

Для того чтобы приступить к этапу разработки проекта, необходимо четко сформулировать функциональные требования к программному средству.

**2.1. Определение требований к программному средству**

Функции администратора сервиса:

* поддерживать работу с базой данных;
* добавление, изменение, удаление абонементов;
* управление статуса об абонементе(заморозка/отмена).

Функции клиента:

* авторизация, регистрация пользователя;
* поиск, фильтрация абонементов;
* просмотр информации об абонементе;
* оформление абонемента;
* просмотр истории оформленных абонементов.

**2.2 Спецификация функциональных требований**

Для функциональности разрабатываемого программного средства необходимо реализовать регистрацию и авторизацию пользователя. Данная функция позволяет идентифицировать каждого пользователя.

При определении данных пользователя, ему будут предоставлены соответствующие функциональные возможности в программном средстве.

На диаграмме 2.1 представлена диаграмма последовательности:

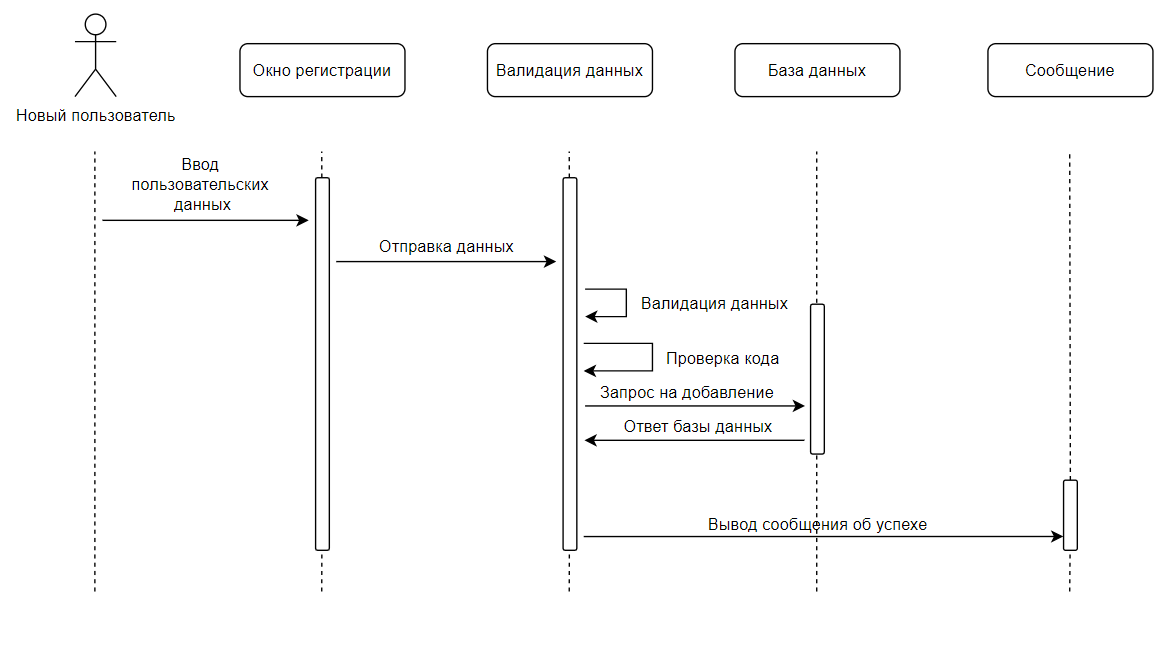
****

Рисунок 2.1 – Диаграмма последовательности

Данная диаграмма последовательности описывает взаимодействие пользователя с системой. Данная диаграмма отображает шаги регистрации пользователя. Программный продукт должен быть реализовать на объектно-ориентированном языке программирования C# с использованием технологии WPF.

Windows Presentation Foundation (WPF) — 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.

База данных была разработана в Microsoft SQL Server. SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире.

SQL Server характеризуется такими особенностями как: - производительность,

SQL Server работает очень быстро; - надежность и безопасность, SQL Server предоставляет шифрование данных; - простота, с данной СУБД относительно легко работать и вести администрирование.

**2.3 Вывод по разделу**

В данном разделе были определены и проанализированы функциональные требования к разрабатываемому программному средству. С помощью диаграммы последовательности были отображены функциональные требования к регистрации в разрабатываемом функциональном продукте.

1. **Проектирование архитектуры проекта**

Проектирование архитектуры проекта является важным этапом разработки программного средства. Оно определяет структуру, компоненты и взаимодействие между ними, а также обеспечивает основу для успешной реализации проекта.

* 1. **Проектирование схемы базы данных**

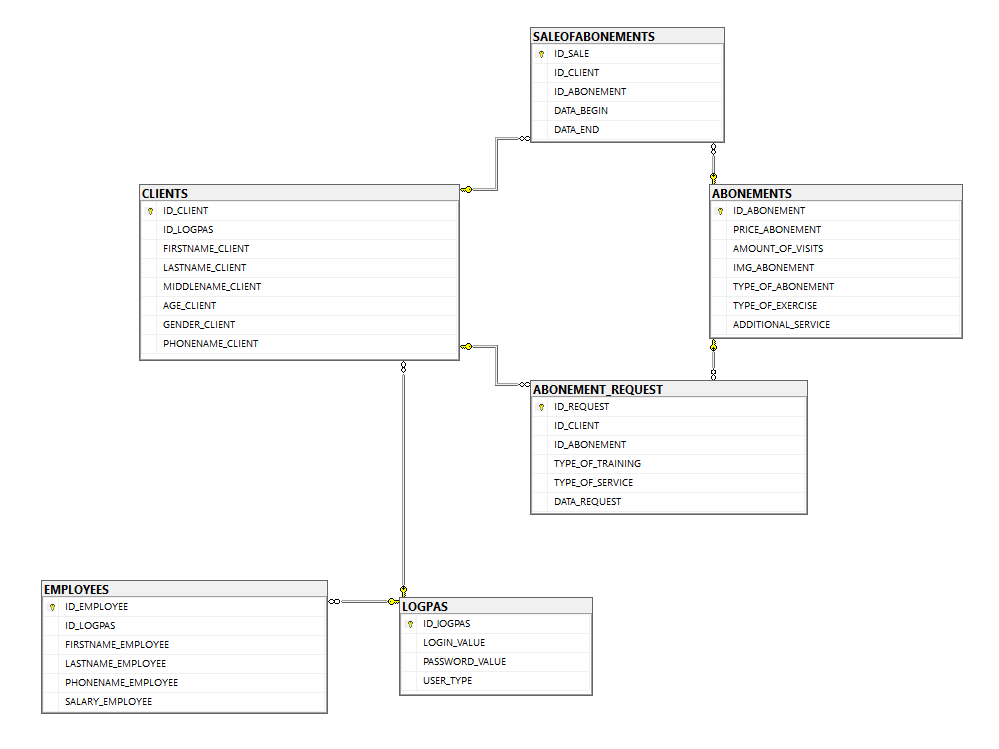
****

Рисунок 3.1– Логическая схема БД

Таблица CLIENTS – хранит информацию о клиентах и состоит из следующих столбцов:

ID\_CLIENT: уникальный идентификатор клиентов;

ID\_LOGPAS: уникальный идентификатор данных и является внешним ключом, который ссылается на таблицу LOGPAS;

FIRSTNAME\_CLIENT: фамилия клиента;

LASTNAME\_CLIENT: имя клиента;

MIDDLENAME\_CLIENT: отчество клиента;

AGE\_CLIENT: возраст клиента;

GENDER\_CLIENT: пол клиента;

PHONENAME\_CLIENT: номер телефона клиента;

Таблица EMPLOYEES – хранит информацию о сотрудниках и состоит из следующих столбцов:

ID\_EMPLOYEE: уникальный идентификатор сотрудников;

ID\_LOGPAS: уникальный идентификатор данных и является внешним ключом, который ссылается на таблицу LOGPAS;

FIRSTNAME\_EMPLOYEE: фамилия сотрудника;

LASTNAME\_EMPLOYEE: имя сотрудника;

PHONENAME\_EMPLOYEE: номер телефона сотрудника;

SALARY\_EMPLOYEE: заработная плата сотрудника;

Таблица ABONEMENTS – хранит информацию об абонементах и состоит из следующих столбцов:

ID\_ABONEMENTS: уникальный идентификатор абонементов;

PRICE\_ABONEMENTS: цена абонемента;

AMOUNT\_OF\_VISITS: количество посещений;

IMG\_ABONEMENTS: картинка абонемента;

TYPEOFABONEMENT: тип абонемента;

TYPEOFEXIRCISE: вид занятий;

ADDITIONALSERVICE – услуга;

Таблица LOGPAS – хранит информацию о логинах и паролях и содержит столбцы:

ID\_LOGPAS: уникальный идентификатор данных;

LOGIN\_VALUE: логин пользователя;

PASSWORD\_VALUE: пароль пользователя;

USER\_TYPE: тип пользователя (сотрудник или клиент);

Таблица SALEOFABONEMENTS – хранит информацию о купленных абонементах и имеет столбцы:

ID\_SALE: уникальный идентификатор продажи;

ID\_CLIENT: уникальный идентификатор клиента и является внешним ключом, который ссылается на таблицу CLIENTS;

ID\_ABONEMENT: уникальный идентификатор абонемента и является внешним ключом, который ссылается на таблицу ABONEMENTS;

DATA\_BEGIN: дата начала абонемента;

DATA\_END: конец даты абонемента;

Таблица ABONEMENTREQUEST – хранит информацию об заявках абонементов и содержит столбцы:

ID\_REQUEST: уникальный идентификатор заявки;

ID\_CLIENT: уникальный идентификатор клиента и является внешним ключом, который ссылается на таблицу CLIENTS;

ID\_ABONEMENT: уникальный идентификатор абонемента и является внешним ключом, который ссылается на таблицу ABONEMENTS;

TYPE\_OF\_TRAINING: тип тренировки;

TYPE\_OF\_SERVICES: тип услуги;

DATA\_REQUEST: дата заявки;

**3.2. Описание средств разработки**

При разработке программного средства были использованы следующие средства:

* язык программирования C#;
* платформа .NET;
* среда разработки Microsoft Visual Studio 2022;
* система управления базами данных Microsoft SQL Server;
* инструмент для работы с базой данных Entity Framework;
* технология разработки пользовательских интерфейсов WPF.

Microsoft Visual Studio 2022 – интегрированная среда разработки (IDE), предоставляемая Microsoft. Она предлагает широкий набор инструментов и функций для разработки приложений на различных платформах, включая C# и .NET. Visual Studio обеспечивает удобную среду для написания кода, отладки, тестирования и сборки приложений.

Язык программирования C# является объектно-ориентированным языком программирования, разработанным Microsoft. Он широко используется для разработки приложений на платформе .NET.

Платформа .NET – это платформа разработки приложений, разработанная Microsoft. Она предоставляет среду выполнения и набор библиотек для разработки и запуска приложений на различных операционных системах.

Entity Framework – это инструмент для работы с базами данных в приложениях .NET. Он предоставляет удобный способ взаимодействия с базой данных через объектно-ориентированную модель данных. Entity Framework упрощает создание, чтение, обновление и удаление данных из базы данных, а также обеспечивает механизмы миграции и управления схемой базы данных.

WPF (Windows Presentation Foundation) – это технология разработки пользовательского интерфейса для приложений Windows. Она позволяет создавать богатые и интерактивные пользовательские интерфейсы с использованием различных элементов управления, стилей и шаблонов. WPF интегрируется с языком C# и платформой .NET, обеспечивая мощные возможности для создания современных приложений с привлекательным дизайном.

SQL Server – это система управления базами данных (СУБД) от Microsoft. Она предоставляет надежное и масштабируемое решение для хранения и управления данными. SQL Server поддерживает язык SQL для выполнения запросов и манипуляции данными. Он интегрируется с Entity Framework и позволяет взаимодействовать с базой данных в приложении.

**3.3. Проектирование окон приложения**

Приложение имеет несколько страниц, которые предоставляют различный функционал для пользователей.

Пользователи могут начать с авторизации, где они вводят свои учетные данные для доступа к приложению. Если у пользователя нет учетной записи, он может зарегистрироваться, заполнив соответствующую форму.

После успешной авторизации или регистрации, пользователи попадают на главную страницу, где они могут увидеть общую информацию о физкультурно-оздоровительном центре и доступные абонементы. Они могут выбрать интересующий их абонемент и перейти на страницу с подробной информацией о абонементе.

Для клиентов доступен личный кабинет, где они могут просмотреть свою историю покупок и заявок абонементов. Они также могут просмотреть детали предыдущих покупок и отменить заявки, если это возможно.

Для администраторов доступна страница одобрения заявок, где они могут просмотреть список ожидающих одобрения заявок и принять или отклонить их. Также у администраторов есть страница изменения данных об абонементе, где они могут изменить информацию об абонементах, такую как описание, фотографии, стоимость и даты.

Пользователи могут свободно перемещаться между страницами в зависимости от своих потребностей и роли в приложении.

**3.4. Проектирование Use Case диаграммы**

Для визуального осмысления задач системы применяется схема использования, иллюстрирующая, какие возможности предоставляемой программной системы предоставлены для каждой категории пользователей, данная схема представлена на рисунке 3.2.

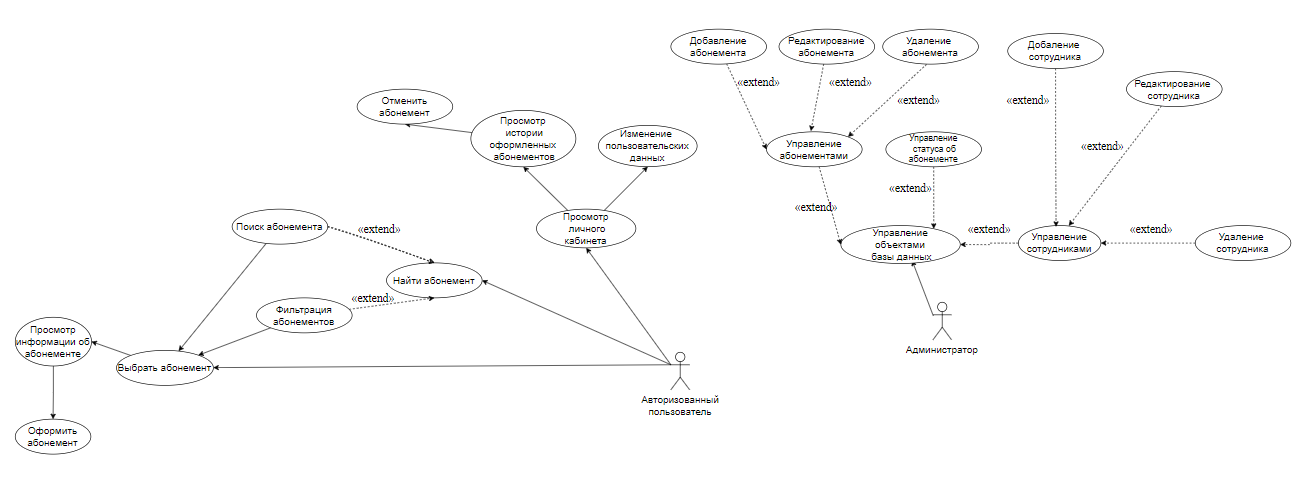
****

Рисунок 3.2– Use Case диаграмма

**3.5 Вывод по разделу**

В данном разделе мы провели важные этапы проектирования физкультурно-оздоровительного центра, которые включали проектирование структуры базы данных, создание Use Case диаграммы, описание средств разработки и проектирование интерфейса центра.

Проектирование структуры базы данных позволило нам определить необходимые таблицы, связи между ними и атрибуты для хранения информации о клиентах, сотрудниках, а также о доступных услугах и оборудовании в центре.

Создание Use Case диаграммы помогло нам идентифицировать основных актеров, таких как клиенты, сотрудники и администраторы, и определить их взаимодействие с системой. Мы также описали функциональные возможности, которые система предоставляет для каждого актера, и сценарии использования.

Описание средств разработки позволило нам указать, какие инструменты и технологии мы использовали при разработке физкультурно-оздоровительного центра. Мы упомянули использование среды разработки, такой как Visual Studio, языка программирования C#, реализация базы данных в рамках системы управления базами данных SQL Server.

Проектирование интерфейса физкультурно-оздоровительного центра позволило нам определить основные страницы и путь взаимодействия пользователей с системой. Мы учли удобство использования и эргономику интерфейса, чтобы обеспечить комфортные условия для клиентов и сотрудников.

В целом, проведенные этапы проектирования позволят нам создать надежный, функциональный и удобный в использовании физкультурно-оздоровительный центр для наших клиентов.

1. **Разработка функциональной модели и модели данных ПС**

Разработка функциональной модели и модели данных программного средства позволяет определить функциональные возможности разрабатываемого программного средства, а также построить абстрактную структуру.

Функциональная модель – абстракция, описывающая функции, операции и взаимодействия в системе. Это представление того, что система должна делать, фокусируясь на функциональности, а не на реализации. Она включает в себя функции, взаимодействия, потоки данных, структуру системы и иерархию функций, обеспечивая понимание работы системы на уровне функциональности.

## **4.1. Используемые паттерны**

Использование паттернов программирования является важным аспектом разработки программного обеспечения, поскольку это способствует повышению качества, управляемости и эффективности кода. Вот несколько ключевых причин, почему паттерны в программировании так важны:

* улучшение масштабируемости: паттерны помогают создавать гибкие и масштабируемые системы. Разделяя функциональность на независимые компоненты, паттерны позволяют легко добавлять новые возможности и модифицировать систему без значительных изменений в коде.
* обеспечение надежности и стабильности: паттерны разработаны и отлажены сообществом опытных разработчиков. Используя эти паттерны, разработчики могут быть уверены в надежности и стабильности своих решений.
* переиспользование кода: паттерны предоставляют проверенные и оптимальные решения для распространенных проблем. Используя эти паттерны, разработчики могут эффективно переиспользовать существующий код, избегая необходимости решать одни и те же задачи заново.
* сокращение времени разработки: паттерны предоставляют готовые к использованию абстракции, что сокращает время, необходимое для разработки. Разработчики могут сосредоточиться на бизнес-логике приложения, а не на решении технических деталей, которые уже решены в рамках паттерна.

В данном курсовом проекте были использованы два паттерна: MVVM и Сommand.

MVVM (Model-View-ViewModel) - это архитектурный паттерн проектирования, который используется в разработке программного обеспечения для построения пользовательских интерфейсов. MVVM предлагает четкое разделение ответственности между компонентами приложения, что упрощает его структуру и поддержку. Вот основные компоненты MVVM:

Модель (Model): представляет собой слой данных и бизнес-логики приложения. Отвечает за доступ к данным, их обработку и валидацию. Не зависит от пользовательского интерфейса.

Представление (View): отображает данные и взаимодействует с пользователем. Не содержит бизнес-логики и не обрабатывает данные напрямую. Реагирует на действия пользователя, такие как клики, ввод текста и другие события.

Модель-представление (ViewModel): предоставляет прослойку между Моделью и Представлением. Содержит логику представления (в том числе форматирование данных и валидацию) без привязки к конкретному пользовательскому интерфейсу. Осуществляет двустороннюю связь с Моделью и Представлением.

Преимущества MVVM:

* Разделение ответственности: Четкое разделение слоев упрощает поддержку, тестирование и модификацию кода.
* Повторное использование кода: Логика представления может быть повторно использована в различных частях приложения.
* Улучшенная тестируемость: Model и ViewModel могут быть легко тестированы отдельно, поскольку они не зависят от пользовательского интерфейса.
* Гибкость в дизайне пользовательского интерфейса: Изменения в пользовательском интерфейсе могут быть внесены без модификации бизнес-логики.
* Повышенная читаемость кода: Структура MVVM делает код более понятным и поддерживаемым.

На рисунке 4.1 представлена структура паттерна MVVM:

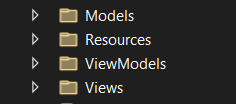


Рисунок 4.1 – Структура паттерна MVVM

Данная структура паттерна позволила разработать все элементы данного проекта.

На листинге 4.2 представлен пример реализации модели данных:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ApplicationWPF.Models

{

public partial class ABONEMENTS

{

public ABONEMENTS()

{

this.ABONEMENT\_REQUEST = new HashSet<ABONEMENT\_REQUEST>();

this.SALEOFABONEMENTS = new HashSet<SALEOFABONEMENTS>();

}

[Key]

public int ID\_ABONEMENT { get; set; }

public decimal PRICE\_ABONEMENT { get; set; }

public int AMOUNT\_OF\_VISITS { get; set; }

public byte[] IMG\_ABONEMENT { get; set; }

public string TYPE\_OF\_ABONEMENT { get; set; }

public string TYPE\_OF\_EXERCISE { get; set; }

public string ADDITIONAL\_SERVICE { get; set; }

public virtual ICollection<ABONEMENT\_REQUEST> ABONEMENT\_REQUEST { get; set; }

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

}

}

Листинг 4.2 – Модель Abonements

Для полной реализации паттерна, стоит добавить реализации компонентов, таких как View и ViewModel. В качестве примера возьмем файл Registration.xaml и RegistrationViewModel.

На листинге 4.3 представлен пример реализации компонента View:

<TextBlock Text="Registration" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Row="2"/>

<StackPanel Grid.Row="3">

<TextBlock Text="LastName" Style="{StaticResource {x:Type TextBlock}}"/>

<TextBox Text="{Binding LastName}"/>

<TextBlock Text="FirstName" Style="{StaticResource {x:Type TextBlock}}"/>

<TextBox Text ="{Binding FirstName}"/>

<TextBlock Text="MiddleName" Style="{StaticResource {x:Type TextBlock}}"/>

<TextBox Text ="{Binding MiddleName}"/>

<TextBlock Text="Age" Style="{StaticResource {x:Type TextBlock}}"/>

<TextBox Text ="{Binding Age}"/>

<TextBlock Text="Gender" Style="{StaticResource {x:Type TextBlock}}"/>

<TextBox Text ="{Binding Gender}"/>

<TextBlock Text="Phone" Style="{StaticResource {x:Type TextBlock}}"/>

<TextBox Text ="{Binding Phone}"/>

<TextBlock Text="Username" Style="{StaticResource {x:Type TextBlock}}"/>

<TextBox Text ="{Binding Username}"/>

<TextBlock Text="Password" Style="{StaticResource {x:Type TextBlock}}"/>

<TextBox Text ="{Binding Password}"/>

<TextBlock Text="ConfirmPassword" Style="{StaticResource {x:Type TextBlock}}"/>

<TextBox/>

<Button Content="Register" Command="{Binding RegistrationCommand}" Height="35" Width="123"/>

</StackPanel>

<TextBlock Text="© 2023 Fitness Center" Grid.Row="4" HorizontalAlignment="Center" VerticalAlignment="Center"/>

Листинг 4.3 – Компонент Registration

При условии рассмотренных листингов для модели данных и View, необходимо также рассмотреть само ViewModel.

На листинге 4.4 представлен пример реализации компонента ModelView:

private void Registration(object parameter)

{

bool isSuccess = false;

try

{

var logpas = new LOGPAS

{

LOGIN\_VALUE = \_username,

PASSWORD\_VALUE = \_password,

USER\_TYPE = "client"

};

\_dbContext.LOGPAS.Add(logpas);

\_dbContext.SaveChanges();

var idLogpas = \_dbContext.LOGPAS

.Where(a => a.LOGIN\_VALUE == \_username && a.PASSWORD\_VALUE == \_password)

.Select(a => a.ID\_lOGPAS)

.FirstOrDefault();

var client = new CLIENTS

{

ID\_LOGPAS = idLogpas,

FIRSTNAME\_CLIENT = \_firstname,

LASTNAME\_CLIENT = \_lastname,

MIDDLENAME\_CLIENT = \_middlename,

AGE\_CLIENT = int.Parse(\_age),

GENDER\_CLIENT = \_gender,

PHONENAME\_CLIENT = \_phone

};

\_dbContext.CLIENTS.Add(client);

\_dbContext.SaveChanges();

isSuccess = true;

}

catch (Exception)

{

MessageBox.Show("Invalid credentials. Please try again.");

}

if (isSuccess)

{

MessageBox.Show("Registration successful! ");

}

else

{

MessageBox.Show("Registration failed. Please check your credentials.");

}

}

Листинг 4.4 – Компонент RegistrationViewModel

Паттерн Command (команда) — это поведенческий паттерн проектирования, который превращает запросы в объекты, позволяя передавать их как аргументы при вызове методов, ставить запросы в очередь, логировать их, а также поддерживать отмену операций.

Основные участники паттерна "Команда":

* команда (Command), определяет общий интерфейс для выполнения определенной операции. Включает в себя метод execute, который вызывается для выполнения команды;
* конкретная команда (ConcreteCommand), реализует интерфейс команды и связывает с собой конкретное действие и объект, который должен выполнить это действие;
* отправитель (Invoker), знает о команде и вызывает её метод execute. Не обязан знать, как конкретно реализована команда;
* получатель (Receiver), знает, как выполнять операции, связанные с выполнением команды. Выполняет фактическую работу, когда вызывается метод execute команды;
* клиент (Client), создаёт объекты команды и устанавливает их отправителя.

На листинге 4.5 представлена реализация паттерна Command.

public ICommand RegistrationCommand { get; }

public ICommand ExitCommand { get; private set; }

public ICommand ToLogin { get; private set; }

public RegistrationViewModel()

{

RegistrationCommand = new RelayCommand(Registration, CanRegistration);

ToLogin = new RelayCommand(parameter => LoginApp());

ExitCommand = new RelayCommand(parameter => ExitApp());

}

Листинг 4.5 – Паттерн Command

По итогу использование данных паттернов, код выглядит структурированным, надежным, масштабируемым.

## **4.2. Разработка модели данных**

Для определения модели данных в проекте используются классы, представляющие собой модели данных. Данные модели соотносимы с таблицами базы данных. Для разработки модели данных необходимо представь проект в виде диаграммы классов.

Диаграмма классов – это структурный вид диаграммы в языке моделирования UML (Unified Modeling Language), который используется для визуализации структуры и отношений между классами в системе. Классы представляют абстракции объектов или сущностей, а отношения между ними показывают, как эти классы взаимодействуют друг с другом.

Основные элементы на диаграмме классов включают:

классы, представляют сущности в системе и содержат атрибуты (переменные) и методы (функции), которые описывают поведение объекта;

отношения, показывают, как классы взаимодействуют друг с другом. Например, отношения могут быть ассоциациями (связи между объектами), агрегациями (часть-целое отношение, где один объект включает в себя другой), композициями (строгий вид агрегации), наследованиями и реализациями интерфейсов;

атрибуты и методы, отображаются внутри классов и предоставляют дополнительную информацию о состоянии и поведении объектов;

интерфейсы, отображают, какие методы класса доступны для других классов;

На рисунке 4.6 представлена диаграмма классов, разрабатываемого проекта.

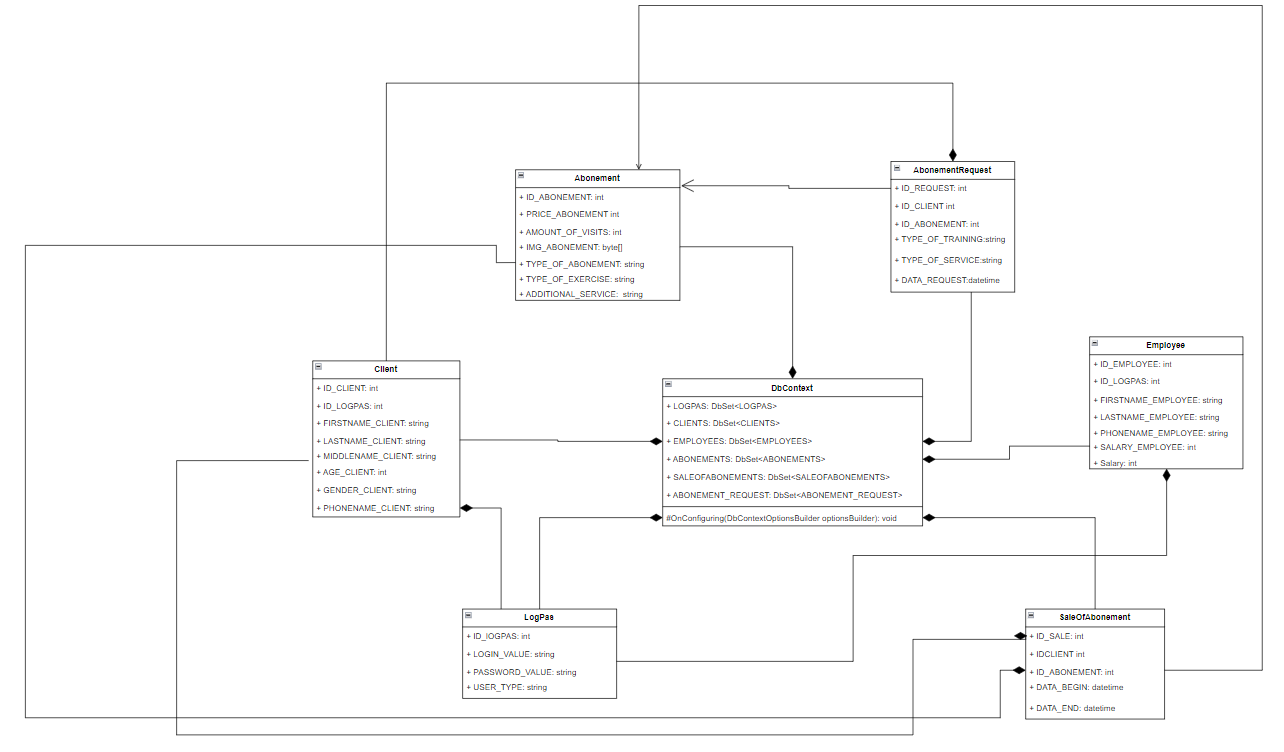


Рисунок 4.6 – Диаграмма классов

Данная диаграмма классов отображает всю структуру модели данных разрабатываемого проекта.

## **4.3. Регистрация и авторизация**

Сразу после запуска программу пользователь попадает в окно авторизации. Если поля логина и пароля будут пустыми, в окне будет выведено сообщение об ошибке. После программа обращается к базе данных и проверяет существует ли пользователь с соответствующим логином и паролем. Если пользователь не найден, будет выведено сообщение об ошибке, в противном случае вход в приложение.

На листинге 4.7 представлен LoginViewMode(код авторизации):

public class LoginViewModel : ViewModelBase, IDataErrorInfo

{

private string \_userName;

private string \_password;

private readonly MyDbContext \_dbContext = new MyDbContext();

private bool \_isPasswordVisible;

public string UserName

{

get => \_userName;

set

{

\_userName = value;

OnPropertyChanged(nameof(UserName));

((RelayCommand)LoginCommand).RaiseCanExecuteChanged();

}

}

public string Password

{

get => \_password;

set

{

\_password = value;

OnPropertyChanged(nameof(Password));

((RelayCommand)LoginCommand).RaiseCanExecuteChanged();

}

}

public bool IsPasswordVisible

{

get { return \_isPasswordVisible; }

set

{

if (\_isPasswordVisible != value)

{

\_isPasswordVisible = value;

OnPropertyChanged(nameof(IsPasswordVisible));

}

}

}

public string this[string columnName]

{

get

{

string error = null;

switch (columnName)

{

case nameof(UserName):

if (string.IsNullOrWhiteSpace(UserName))

error = "Имя пользователя не может быть пустым.";

break;

case nameof(Password):

if (string.IsNullOrWhiteSpace(Password))

error = "Пароль не может быть пустым.";

break;

}

return error;

}

}

Листинг 4.7 – Код авторизации

При отсутствии аккаунта пользователя, ему необходимо перейти на страницу регистрации. Если поля регистрации останутся пустыми, программа выдаст сообщение об ошибке.

Если поля регистрации не соответствуют определённой валидации, программа сотрёт значение поля. Далее приложение осуществляет запрос к базе данных для проверки наличия пользователя с указанным логином и паролем.

Если такой пользователь найден, появится сообщение об ошибке. В противном случае произойдет успешная регистрация и переадресация пользователя на страницу авторизации.

Реализация данной логики представлена в листинге 4.8.

private void Registration(object parameter)

{

bool isSuccess = false;

try

{

var logpas = new LOGPAS

{

LOGIN\_VALUE = \_username,

PASSWORD\_VALUE = \_password,

USER\_TYPE = "client"

};

\_dbContext.LOGPAS.Add(logpas);

\_dbContext.SaveChanges();

var idLogpas = \_dbContext.LOGPAS

.Where(a => a.LOGIN\_VALUE == \_username && a.PASSWORD\_VALUE == \_password)

.Select(a => a.ID\_lOGPAS)

.FirstOrDefault();

var client = new CLIENTS

{

ID\_LOGPAS = idLogpas,

FIRSTNAME\_CLIENT = \_firstname,

LASTNAME\_CLIENT = \_lastname,

MIDDLENAME\_CLIENT = \_middlename,

AGE\_CLIENT = int.Parse(\_age),

GENDER\_CLIENT = \_gender,

PHONENAME\_CLIENT = \_phone

};

\_dbContext.CLIENTS.Add(client);

\_dbContext.SaveChanges();

isSuccess = true;

}

catch (Exception)

{

MessageBox.Show("Invalid credentials. Please try again.");

}

if (isSuccess)

{

MessageBox.Show("Registration successful! ");

}

else

{

MessageBox.Show("Registration failed. Please check your credentials.");

}

}

Листинг 4.8 – Код регистрации

Данные возможности позволяют точно идентифицировать пользователя.

## **4.4. Заявка и оформление абонемента**

Каждый из клиентов имеет возможность перейти на страницу абонементов, отсортировать, найти или выбрать абонемент из списка. После этапа выбора абонемента, пользователь имеет возможность перейти на страницу конкретного абонемента и ознакомиться с его информацией более подробно. На странице абонемента присутствует кнопка для оформления абонемента. Данная кнопка отправит пользователя на страницу с подтверждением заявки.

Реализация данной логики представлена в листинге 4.9.

private void ConfirmRequest(object parameter)

{

if (!IsConfirmationChecked)

{

MessageBox.Show("Ошибка! Подтвердите заявку на оформление абонемента.");

return;

}

CloseWindow();

}

Листинг 4.9 – Код подтверждение заявки клиента

После успешного подтверждение заявки, клиент должен дождаться успешного подтверждение от администратора.

Реализация данной логики представлена в листинге 4.10.

private void Approve(object parameter)

{

if (\_selectedRequest != null)

{

using (var dbContext = new DBContext())

{

var request = dbContext.Abonements\_request.Find(\_selectedRequest.IdBooking);

if (request!= null)

{

var newSale = new SALES

{

sale\_date = DateTime.Now,

id\_client = existingBooking.id\_client,

id\_tour = existingBooking.id\_tour,

id\_employee = currentEmployee.id\_employee

};

dbContext.SALE.Add(newSale);

dbContext.SaveChanges();

dbContext.BOOKING.Remove(existingBooking);

dbContext.SaveChanges();

LoadRequest();

}

}

}

else

{

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

}

Листинг 4.10 – Код подтверждение заявки администратора

Данные методы реализуют функциональность заявки и оформления абонементов для клиента.

## **4.5. Вывод по разделу**

В ходе анализа и проектирования функциональных моделей были определены основные задачи, функции и взаимосвязи между компонентами системы. Это позволило создать четкое представление о том, как программа будет взаимодействовать с пользователями и другими системами. Разработка модели данных помогла лучше понять потребности приложения в хранении и обработке информации.

1. **Тестирование**

Для проверки работоспособности программного средства было проведено его тщательное тестирование, в ходе которого осуществлялась проверка заполняемости базы данных, корректность извлекаемых из базы данных значений, а также реакция программы на вводимые, заведомо неверные, данные. Было проработано множество ситуаций, которые могут возникнуть в ходе работы с программным средством. Так, при входе осуществляется проверка на действительные логин и пароль.

Например, пользователь ввел неверный пароль или логин (результат на рис 5.1):

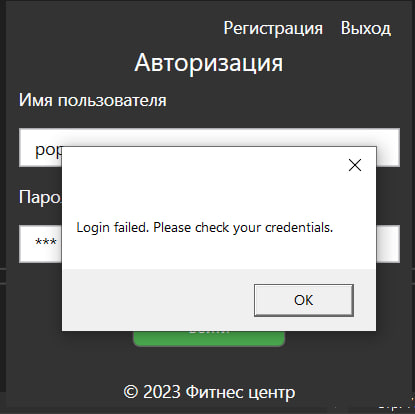


Рисунок 5.1 − Результат входа с неверным логином или паролем

Или вы упустили пустое поле и нажали войти (результат на рис. 5.2):

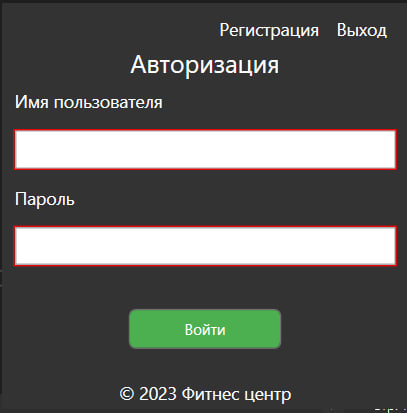


Рисунок 5.2 − Пустые поля при попытке авторизации

При регистрации пользователь обязан заполнить все поля, иначе форма не пройдёт проверку на пустые поля.

Результат представлен на рисунке 5.3.

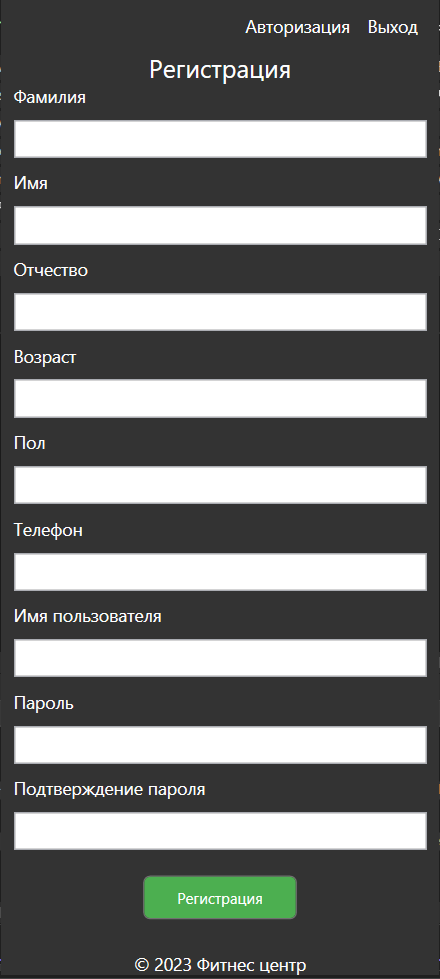


Рисунок 5.3 – Проверка на пустоту полей

При попытке изменить значения абонемента на отрицательное, выводиться окно предупреждения.

Результат валидации представлен на рисунке 5.4:

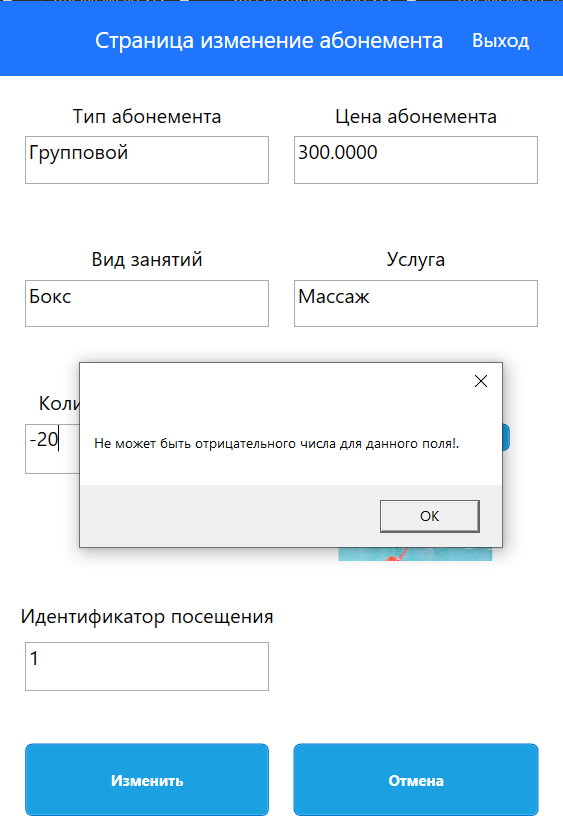


Рисунок 5.4 – Обработка отрицательного числа

Если при регистрации пользователь выберет логин который уже существует, у него не получится зарегистрироваться, пока он его не изменит.

Представлено на рисунке 5.5:

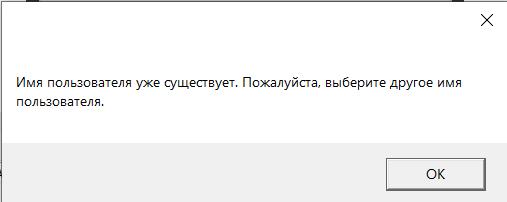


Рисунок 5.5 – Логин уже существует

Тестирование позволяет минимизировать риски возникновения ошибок и обеспечивает более стабильную работу системы.

## **5.1. Вывод по разделу**

Таким образом в проекте были обработаны все возможные ошибки, что позволяет минимизировать шансы некорректной работы приложения и повысить комфорт его использования.

В результате тщательного тестирования обнаружены и обработаны различные сценарии использования, что улучшает надежность и устойчивость программного средства.

# **6. Руководство по установке и использованию**

**6.1. Установка программного средства**

Для работы программного средства необходима первоначальная настройка. Для начала, необходимо наличие установленной на компьютере системы управления базами данных Microsoft SQL Server, а также утилиты SQL Server Management Studio для управления и администрирования компонентов Microsoft SQL Server.

Необходимо войти в SQL Server Management Studio, где открыть файл с sql-запросами и создать бд.

После всех проделанных шагов в раскрывающемся списке «Базы данных» обозревателя объектов должна появиться база данных “PUBL”.

Далее необходимо запустить установщик приложения и установить на свой компьютер.

На рисунке 6.1 представлен установщик:



Рисунок 6.1 − Установщик приложения PUBL

Теперь программное средство готово к запуску.

**6.2. Руководство пользователя по работе с программным средством**

После успешного присоединения базы данных и установки приложения пользователю необходимо запустить файл ApplicationWpf.exe. На рисунке 6.2 представлен файл.



Рисунок 6.2 − Запускаемый файл приложения ApplicationWpf

После открытия окна авторизации новому пользователю требуется перейти к окну регистрации, в котором правильно заполнить все поля. После чего пройти авторизацию.

**6.3. Вывод по разделу**

Данный раздел представляет собой установку и руководство пользования данным приложением. Были представлены основные функции пользования, а также базовая инструкция по установке.

**Заключение**

В ходе выполнения данного курсового проекта было реализовано программное средство для физкультурно-оздоровительного центра, позволяющее оформить абонемент для клиента, а также производить модерацию и управление для сотрудника и клиента.

Была спроектирована схема реляционной базы данных, а также создана база данных по данной схеме.

**Список литературы**

1. WPF - система "Авторизации и Регистрации" [Электронный ресурс]. – Режим доступа: https://you-hands.ru/2018/08/31/wpf-sistema-avtorizacii-i-registracii/ – Дата доступа: 28.10.2023
2. Microsoft .NET [Электронный ресурс]. – Режим доступа: https://dotnet.microsoft.com/en-us/ – Дата доступа: 14.11.2023
3. Metanit [Электронный ресурс]. ⎯ Электронные данные . ⎯ Режим доступа: 30.09.2023.
4. https://metanit.com/sharp/wpf/?ysclid=lqckqsfx58539466315 .Дата доступа: 30.09.2023.
5. С# 6.0. Справочник. Полное описание языка 6-е издание / Жозеф Албахари, Бен