МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 6-05-0612-01 «Программная инженерия»

Направление специальности 6-05-0612-01 «Программная инженерия»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

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

Тема: Программное средство «Фитнес-клуб»

Исполнитель

студент 2 курса группы 10 Жамойдо Артём Игоревич

(Ф.И.О.)

Руководитель работы асс. Мущук Артур Николаевич

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

Курсовой проект защищен с оценкой

Председатель Мущук А.Н.

(подпись)

Минск 2025

Содержание

[ВВЕДЕНИЕ 4](#_Toc198879789)

[1. Аналитический обзор литературы и постановка задачи 5](#_Toc198879790)

[1.1. Анализ прототипов 5](#_Toc198879791)

[1.2. Постановка задачи 10](#_Toc198879792)

[2. Анализ требований к программному средству и разработка функциональных требований 11](#_Toc198879793)

[2.1. Описание средств разработки 11](#_Toc198879794)

[2.1.1. Microsoft Visual Studio 2022 11](#_Toc198879795)

[2.1.2. Программная платформа .NET Core 8.0 11](#_Toc198879796)

[2.1.3. Язык программирования C# 11](#_Toc198879797)

[2.1.4. Технология WPF 12](#_Toc198879798)

[2.1.5. Расширяемый язык разметки XAML 12](#_Toc198879799)

[2.1.6. Технология Entity Framework Core 12](#_Toc198879800)

[2.1.7. MS SQL Server 12](#_Toc198879801)

[2.2. Спецификация функциональных требований к программному средству 13](#_Toc198879802)

[2.3. Спецификация функциональных требований 13](#_Toc198879803)

[2.4. Выводы по разделу 14](#_Toc198879804)

[3. Проектирование программного средства 15](#_Toc198879805)

[3.1. Общая структура 15](#_Toc198879806)

[3.2. Взаимоотношения между классами 15](#_Toc198879807)

[3.3. Модель базы данных 16](#_Toc198879808)

[3.4. Проектирование функционала проекта 17](#_Toc198879809)

[3.5. Проектирование последовательностей проекта 17](#_Toc198879810)

[3.6. Выводы по разделу 17](#_Toc198879811)

[4. Реализация программного средства 18](#_Toc198879812)

[4.1. Основные классы программного средства 18](#_Toc198879813)

[4.2. Описание классов и методов программного средства 18](#_Toc198879814)

[4.2.1. Авторизация и регистрация 18](#_Toc198879815)

[4.2.2. Просмотр и добавление абонементов 19](#_Toc198879816)

[4.2.3. Управление расписанием 19](#_Toc198879817)

[4.2.4. Покупка абонементов 20](#_Toc198879818)

[4.2.5. Управление отзывами пользователей 20](#_Toc198879819)

[4.2.6. Запись на тренировки 21](#_Toc198879820)

[4.3 Выводы по разделу 21](#_Toc198879821)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 22](#_Toc198879822)

[5.1. Тестирование авторизации 22](#_Toc198879823)

[5.2. Тестирование создания абонемента 22](#_Toc198879824)

[5.3. Тестирование блокировки пользователя 23](#_Toc198879825)

[5.4. Тестирование добавления пользователя 24](#_Toc198879826)

[5.5. Тестирование удаления взаимосвязанных данных 25](#_Toc198879827)

[5.6. Выводы по разделу: 26](#_Toc198879828)

[6. Руководство по установке и использованию 27](#_Toc198879829)

[6.1. Использование 27](#_Toc198879830)

[6.2. Выводы по разделу: 31](#_Toc198879831)

[Заключение 33](#_Toc198879832)

[Список литературы 34](#_Toc198879833)

[ПРИЛОЖЕНИЕ А 35](#_Toc198879834)

[ПРИЛОЖЕНИЕ Б 36](#_Toc198879835)

[ПРИЛОЖЕНИЕ В 37](#_Toc198879836)

[ПРИЛОЖЕНИЕ Г 38](#_Toc198879837)

# **ВВЕДЕНИЕ**

Современные технологии стремительно развиваются, и фитнес-клубы остаются востребованными местами для занятий спортом, поддержания здоровья, обучения и общения. Несмотря на рост популярности домашних тренажеров и онлайн-уроков, фитнес-клубы предлагают уникальные преимущества: современное оборудование, профессиональную поддержку тренеров, комфортную атмосферу, возможность участия в групповых тренировках и простое общение с единомышленниками.

Однако эффективное управление таким заведением требует автоматизации многих процессов, включая бронирование тренировок, учет клиентов, контроль расписания и загрузки залов. Специализированное приложение для фитнес-клуба может значительно упростить работу администрации, повысить уровень сервиса и улучшить взаимодействие с посетителями.

Целью данной курсовой работы является разработка программного средства для управления фитнес-клубом, включающего функционал учета клиентов, бронирования тренировок, мониторинга активности. Данное программное средство должно позволить пользователям в лице клиентов записываться на тренировки и просматривать расписание тренировок.

Программное средство будет предоставлять администраторам широкий набор инструментов для контроля посещаемости, ведения базы клиентов, управления абонементами, расписанием тренировок и работы тренеров. Также предусмотрена интеграция с базой данных для хранения информации о клиентах, истории посещений, оплатах, а также отчетности по различным параметрам деятельности клуба.

В процессе разработки будут проанализированы и реализованы следующие ключевые аспекты:

* архитектура приложения и выбор подходящей модели проектирования (например, MVVM);
* разработка современного и понятного пользовательского интерфейса;
* механизмы авторизации и разграничения прав доступа для различных ролей (администратор, клиент);
* функции учета посещаемости, оплаты клиентов;

1. Аналитический обзор литературы и постановка задачи

Для того чтобы окончательно определиться с постановкой задачи курсового проекта, необходимо проанализировать прототипы программных средств выбранной темы.

**1.1. Анализ прототипов**

Gym24.by – платформа, представляющая сеть круглосуточных фитнес-клубов в Минске (рисунок 1.1). Сайт спортивного клуба выполнен в современном и аккуратном стиле. Дизайн выдержан в желто-белой цветовой гамме, которая создает приятное визуальное восприятие и не перегружает пользователя лишними деталями.

Основные разделы четко структурированы, а удобная навигация позволяет быстро находить нужную информацию. Для удобства посетителей реализованы фильтрация и поиск по типу тренировок, что значительно упрощает подбор подходящей программы занятий. Также предусмотрена возможность выбора удобного зала – пользователь может записаться на тренировку в одном из доступных филиалов, исходя из расположения. Процесс покупки абонемента отличается простотой и интуитивностью, что делает его доступным даже для новых пользователей.

Однако, несмотря на удобство функционала, сайт предоставляет ограниченное количество информации о том, какие услуги включены в абонемент. Это может затруднять принятие решения для потенциальных клиентов. Кроме того, на страницах представлено преимущественно текстовое описание, тогда как изображения оборудования и интерьера залов представлены в недостаточном количестве, что снижает визуальную информативность ресурса.

Кроме того, мобильная версия сайта обладает хорошей адаптивностью, что позволяет комфортно пользоваться ресурсом на разных устройствах. Однако некоторые элементы интерфейса могли бы быть оптимизированы для еще большей удобочитаемости и взаимодействия. Также на сайте отсутствует возможность просмотра отзывов посетителей, что могло бы помочь новым клиентам принять более информированное решение.

Было бы полезно добавить интерактивные элементы, такие как виртуальный тур по залу или видеопрезентации тренировок, чтобы улучшить визуальное восприятие и информативность ресурса.

Достоинства:

* В меру яркий дизайн – желто-белая тема.
* Удобная навигация – основные разделы легко доступны.
* Фильтрация и поиск – сортировка по типу тренировок.
* Выбор удобного зала – возможность записи в зал по одному из адресов.
* Простая покупка абонементов.

Недостатки:

* Малое количество информации о услугах, входящих в абонемент.
* Преимущественно текстовая информация, недостаточно изображений оборудования (рисунок 1.1).

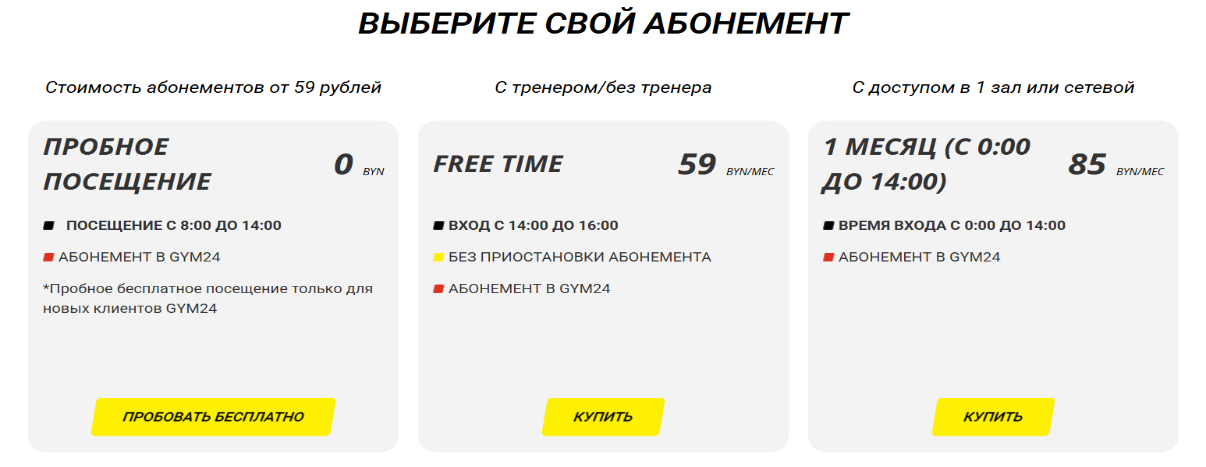


Рисунок 1.1 – Информация о абонементах

Mf.by – удобный и стильный ресурс, представляющий один из самых больших фитнес-центров в Минске. (рисунок 1.4). Сайт фитнес-центра оформлен в светлой цветовой гамме с яркими акцентами, создающими дружелюбную и привлекательную атмосферу. Акционные предложения размещены в главном меню, что позволяет пользователям сразу видеть выгодные варианты и оперативно принимать решения. Навигация по сайту выполнена с учетом удобства посетителей – она проста и интуитивно понятна, что делает поиск нужной информации легким и быстрым.

Визуальное представление ресурса усилено большим количеством качественных изображений, на которых подробно показано оборудование фитнес-центра. Процесс регистрации и покупки абонемента реализован максимально просто, что избавляет пользователя от лишних сложностей при оформлении.

Кроме того, на сайте предусмотрена интеграция с социальными сетями, благодаря чему посетители могут делиться своими достижениями и получать поддержку от сообщества. Интересной возможностью является индивидуальный подбор тренера, что делает процесс тренировок более персонализированным и удобным для каждого клиента.

Однако, несмотря на множество преимуществ, сайт имеет слишком большие элементы верстки. Также было бы полезно добавить отзывы клиентов, чтобы посетители могли ознакомиться с опытом других пользователей перед принятием решения.

Достоинства:

* Дружелюбный дизайн – белая тема с яркими элементами.
* Акционные предложения на главном меню.
* Простая и интуитивная навигация, ориентированная на пользователя.
* Достаточно много качественных изображений, которые хорошо показывают оборудование фитнес-центра.
* Простая регистрация и покупка абонементов.
* Интеграция с социальными сетями (возможность делиться результатами).
* Возможность индивидуального подбора тренера (рисунок 1.2).

Недостатки:

* Слишком большие элементы верстки.
* Зависимость от качества интернет-соединения.

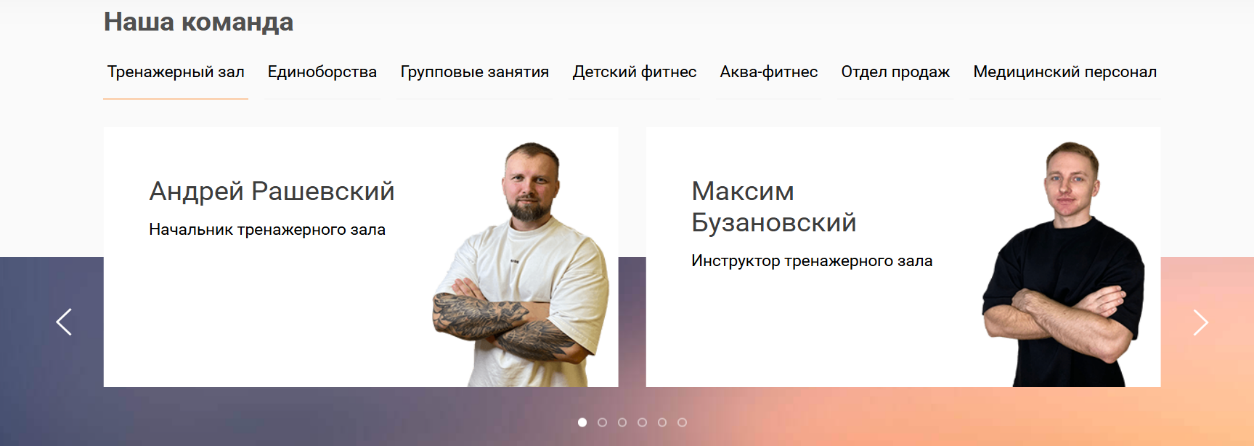


Рисунок 1.2 – Список тренеров

Worldclass.ru – удобное и качественное приложение, представляющее сеть фитнес-клубов в России (рисунок 1.3). Веб-сайт впечатляет стильным и минималистичным дизайном, который сразу привлекает внимание своей элегантностью и нейтральными оттенками.

Удобная навигация позволяет пользователям легко ориентироваться в меню и быстро находить нужную информацию. На главной странице представлено множество акционных предложений, что делает сервис особенно привлекательным для новых и постоянных клиентов.

Дополнительные возможности, такие как интеграция с социальными сетями, позволяют пользователям делиться своими достижениями, а система удобного подбора тренера способствует индивидуальному подходу к тренировкам. Кроме того, программа лояльности поощряет постоянных посетителей, предоставляя бонусные баллы за регулярные занятия.

Однако сайт имеет один заметный недостаток – слишком крупные элементы верстки. Это может затруднять восприятие информации и требовать дополнительного прокручивания страниц, особенно на мобильных устройствах. Несмотря на это, общая функциональность и удобство сайта делают его привлекательным для широкой аудитории.

Достоинства:

* Минималистичный, роскошный дизайн в нейтральных тонах.
* Акционные предложения на главном меню.
* Простая и интуитивная навигация, ориентированная на пользователя.
* Удобная система покупки абонементов.
* Интеграция с социальными сетями (возможность делиться результатами).
* Возможность индивидуального подбора тренера.
* Программа лояльности для постоянных клиентов (баллы за посещения).

Недостатки:

* Слишком большие элементы верстки.

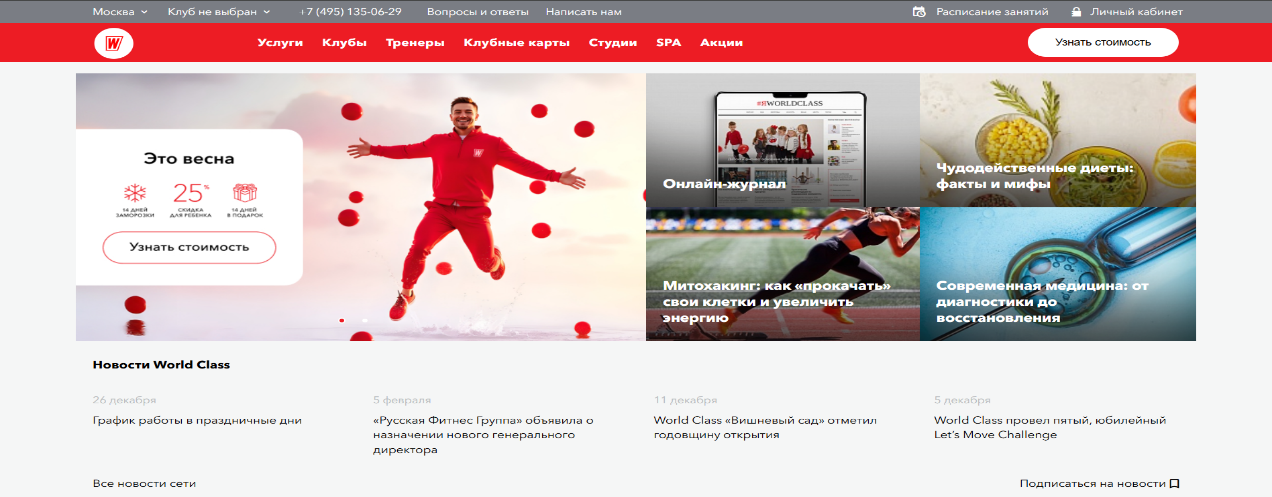


Рисунок 1.3 – Интерфейс веб-приложения «worldclass.ru»

Fitstars.ru – качественное приложение, предоставляющее возможность заниматься онлайн-фитнесом. Веб-сайт привлекает внимание ярким и современным дизайном, который делает процесс поиска и фильтрации контента максимально удобным. Акционные предложения на главной странице позволяют пользователям легко находить выгодные предложения, а интуитивно понятная навигация и продуманная система категорий обеспечивают комфортный доступ ко всем разделам. Особенно впечатляет богатая база видеоуроков по тренировкам, оснащенная удобным фильтром, который помогает подобрать занятия в зависимости от уровня подготовки, целей и типа нагрузки.

Дополнительным плюсом является интеграция с социальными сетями, что позволяет пользователям делиться своими результатами, а также индивидуальный подбор тренера, обеспечивающий персонализированный подход к занятиям.

Однако сайт имеет один заметный недостаток — ограниченность в плане дополнительных услуг. Пользователям может не хватать более разнообразного сервиса, который дополнял бы основной функционал платформы.

Достоинства:

* Яркий и современный дизайн с акцентом на удобство поиска и фильтрации (рисунок 1.4).
* Акционные предложения на главном меню.
* Удобная система категорий и интуитивная навигация.
* База видеоуроков по тренировкам с фильтром по уровню подготовки, целям и типам занятий (кардио, силовые, растяжка).
* Прямые эфиры с тренерами, возможность взаимодействия в реальном времени.
* Интеграция с социальными сетями (возможность делиться результатами).
* Возможность индивидуального подбора тренера.
* Видеоуроки высокого качества с профессиональной съемкой.

Недостатки:

* Ограниченность в плане дополнительных услуг.

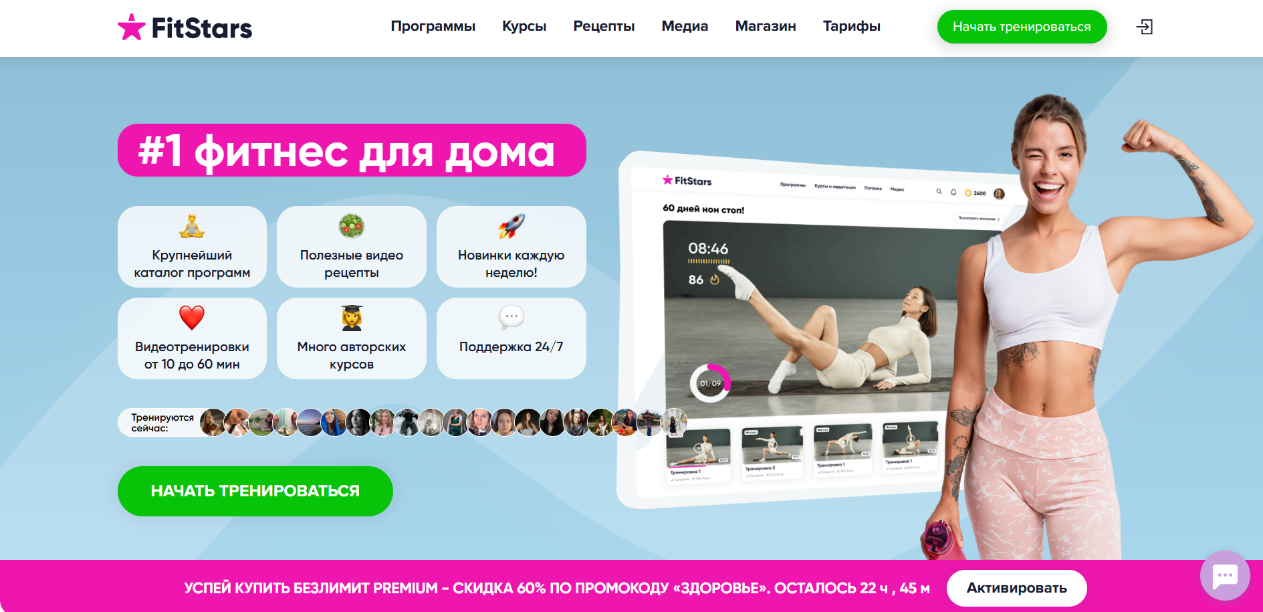


Рисунок 1.4 – Интерфейс веб-приложения «fitstars.ru»

Xfit.ru – стильное и качественное приложение, представляющее сеть фитнес-клубов. Веб-сайт отличается минималистичным дизайном в нейтральных тонах, создавая стильное и ненавязчивое визуальное восприятие. Акционные предложения, размещенные на главной странице, делают платформу привлекательной для пользователей, помогая им находить выгодные варианты тренировок.

Навигация сайта интуитивно понятна, а система категорий упрощает поиск нужных разделов и услуг. Особенно важно, что пользователи могут подобрать тренера, ориентируясь на свои индивидуальные предпочтения и цели, что делает сервис более персонализированным.

Однако одним из недостатков платформы является зависимость от качества интернет-соединения. Для комфортного просмотра и взаимодействия с контентом требуется стабильный и быстрый интернет, что может ограничивать доступность сайта в условиях слабого сигнала.

Кроме того, простота и минимализм в дизайне помогают пользователям сосредоточиться на функционале без лишних отвлекающих элементов. Несмотря на зависимость от интернет-соединения, сайт остается эффективным инструментом для комфортного взаимодействия с тренерами и поиска подходящих тренировок.

Достоинства:

* Минималистичный дизайн в нейтральных тонах.
* Акционные предложения на главном меню.
* Удобная система категорий и интуитивная навигация.
* Возможность индивидуального подбора тренера.

Недостатки:

* Зависимость от качества интернет-соединения.

**1.2. Постановка задачи**

Проанализировав прототипы, были выделены основная задача и функциональные требования проекта.

Основной задачей курсового проекта является разработка десктопного приложения, позволяющего выполнять авторизацию и регистрацию пользователей, предоставлять пользователю в лице администратора возможность управлять заявками, добавлять новости и акции, редактировать расписание тренировок, а также отвечать на отзывы клиентов. Пользователю в лице клиента приложение должно предоставлять функционал для просмотра расписания тренировок, записи на занятия в окне бронирования, оставления комментариев и вопросов.

Приложение должно обеспечивать удобный интерфейс для обеих категорий пользователей, автоматизировать ключевые процессы работы фитнес-клуба и повышать качество обслуживания клиентов.

**2.** **Анализ требований к программному средству и разработка функциональных требований**

**2.1. Описание средств разработки**

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET Core 8.0;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework Core;
* база данных MS SQL Server.

**2.1.1. Microsoft Visual Studio 2022**

Microsoft Visual Studio 2022 — это интегрированная среда разработки для написания, отладки и сборки кода, а также последующей публикации приложений. Данный продукт позволяет разрабатывать не только консольные, но и десктопные приложения, с использованием таких технологий, как WinForms или WPF.

**2.1.2. Программная платформа .NET Core 8.0**

Платформа .NET Core — это созданная Microsoft в 2016 году кроссплатформенная, с открытым исходным кодом технология, предназначенная для создания и выполнения современных приложений. Она поддерживает разработку различных типов приложений, включая веб-приложения, облачные сервисы, десктопные приложения и микросервисы, работая на операционных системах Windows, macOS и Linux.

Основой .NET Core является общеязыковая среда выполнения Common Language Runtime (CLR), которая обеспечивает выполнение кода, написанного на различных поддерживаемых языках программирования, таких как C#, F#, и Visual Basic. .NET Core предлагает модульную архитектуру, где разработчики могут выбирать и использовать только необходимые пакеты, что позволяет создавать более легкие и производительные приложения.

**2.1.3. Язык программирования C#**

В качестве языка программирования используется C# – основной язык разработки в .NET Framework. Язык объектно-ориентированный, имеет строгую статическую типизацию, поддерживает перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения. Используется как основной язык в технологии WPF.

**2.1.4. Технология WPF**

Для предоставления пользовательского интерфейса и разграничения дизайна и бизнес-логики используется технология Microsoft WPF – аналог WinForms, система для построения клиентских приложений Windows с возможностями взаимодействия с пользователем и графическая подсистема в составе .NET Framework, использующая язык разметки XAML.

**2.1.5. Расширяемый язык разметки XAML**

WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление. XAML представляет собой язык декларативного описания интерфейса, основанный на XML.

**2.1.6. Технология Entity Framework Core**

Для взаимодействия с базой данных в приложении WPF используется технология Entity Framework Core — кроссплатформенная и расширяемая версия технологии Entity Framework с открытым исходным кодом, которая активно развивается и дополняется. В то время как традиционные средства ADO.NET обеспечивают создание подключений, выполнение команд и работу с другими объектами для взаимодействия с базами данных, Entity Framework предлагает более высокий уровень абстракции. Эта технология позволяет разработчикам абстрагироваться от конкретной базы данных и работать с данными независимо от типа хранилища. Entity Framework Core является последней версией Entity Framework 8, которая совместима с .NET Core и продолжает активно развиваться.

Entity Framework Core предоставляет два основных способа взаимодействия с базой данных:

* Database First: Entity Framework Core создает набор классов, отражающих модель существующей базы данных;
* Model First: разработчик сначала самостоятельно создает модели данных, на основе которых Entity Framework Core затем генерирует реальную базу данных на сервере.

**2.1.7. MS SQL Server**

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта. Для взаимодействия с базой данных применяется язык SQL. Клиент отправляет запрос 10 на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения. Основной используемый язык запросов — Transact-SQL — реализован на структурированном языке запросов (SQL) с расширениями.

**2.2. Спецификация функциональных требований к программному средству**

Программное средство должно предоставлять следующие функциональные возможности:

Для пользователя:

* регистрация;
* авторизация;
* просмотр списка абонементов;
* детальный просмотр абонементов;
* покупка абонемента;
* подтверждение заказа;
* просмотр приобретенных абонементов;
* возможность оставлять отзыв к занятию;
* возможность поставить рейтинг занятию;
* пополнение баланса;
* изменение темы приложения;
* изменение языка приложения;
* возможность фильтрации абонементов;

Для администратора:

* поиск записей;
* добавление новых абонементов;
* редактирование информации о существующих абонементах;
* удаление существующих абонементов;
* удаление отзывов пользователей;
* удаление занятий;
* блокировка пользователей.

**2.3. Спецификация функциональных требований**

Для функциональности ПС необходимо создание базы данных для хранения информации приложения. Подробно построение базы данных описано в следующем разделе.

В программном средстве необходимо реализовать авторизацию пользователей для доступа ко всем возможностям приложения. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Введенные данные, успешно прошедшие валидацию, сверяются с данными в базе данных.

Пользователь может покупать абонементы, управлять заказами, оставлять отзывы о занятиях, пополнять баланс. Все сведения должны автоматически обновляться в базе данных.

Администратор обладает возможностями просмотра списка абонементов, пользователей, занятий, тренеров, отзывов, заказов, добавления новых абонементов, изменения и удаления абонементов, пользователей, комментариев, заказов.

**2.4. Выводы по разделу**

В данном разделе были выявлены функциональные требования к программному средству для пользователя и администратора.

В разделе были определены инструменты, которые будут использоваться в программном средстве. К таким относятся:

* программная платформа .NET Core 8.0;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework Core;
* база данных MS SQL Server.

**3. Проектирование программного средства**

**3.1. Общая структура**

Программное средство построено по паттерну проектирования MVVM. Есть четкое разделение между бизнес-логикой и представлением. Описание структуры основных папок и файлов проекта представлено в таблице 3.1.

Таблица 3.1 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Helpers | Содержит вспомогательные классы для форматирования, конвертеров, поведения и прочей логики, не связанной напрямую с бизнес-данными. |
| Migrations | Автоматически создаваемые классы миграций Entity Framework — управляют изменениями структуры базы данных. |
| Model | Содержит классы-сущности, которые отображают таблицы в базе данных. |
| Resources | Хранит ресурсы приложения: строки локализации, словари стилей (.xaml), изображения и прочее. |
| View | Представления — визуальные компоненты интерфейса, такие как окна и страницы. |
| ViewModel | Классы ViewModel для MVVM-архитектуры. Содержат свойства, команды и логику, связанную с представлениями. |
| App.xaml | Главный ресурсный файл приложения. Определяет стили и начальное окно. |
| App.xaml.cs | Код-behind для App.xaml, инициализация приложения, глобальные события и службы. |
| AssemblyInfo.cs | Содержит метаданные сборки, такие как версия, название, автор и другие атрибуты. |

Важно отметить, что достичь идеальной архитектуры по паттерну проектирования MVVM довольно сложно, так как в приложении могут присутствовать такие элементы, которые требуют вынесения отдельной логики в представления для корректной работы приложения.

**3.2. Взаимоотношения между классами**

Для визуализации взаимосвязей между классами используется диаграмма UML – графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями).

Для представления внутренней структуры программы в виде классов и связей между ними используется диаграмма классов. Приложение спроектировано таким образом, что каждый класс выполняет свои функции и практически не зависит от других. Диаграмма классов представлена в Приложении А.

**3.3. Модель базы данных**

Для реализации поставленной задачи была создана база данных SporticsDB. Для ее создания использовалась система управления реляционными базами данных MS SQL Server. База данных состоит из таблиц, представленных на рисунке 3.1.

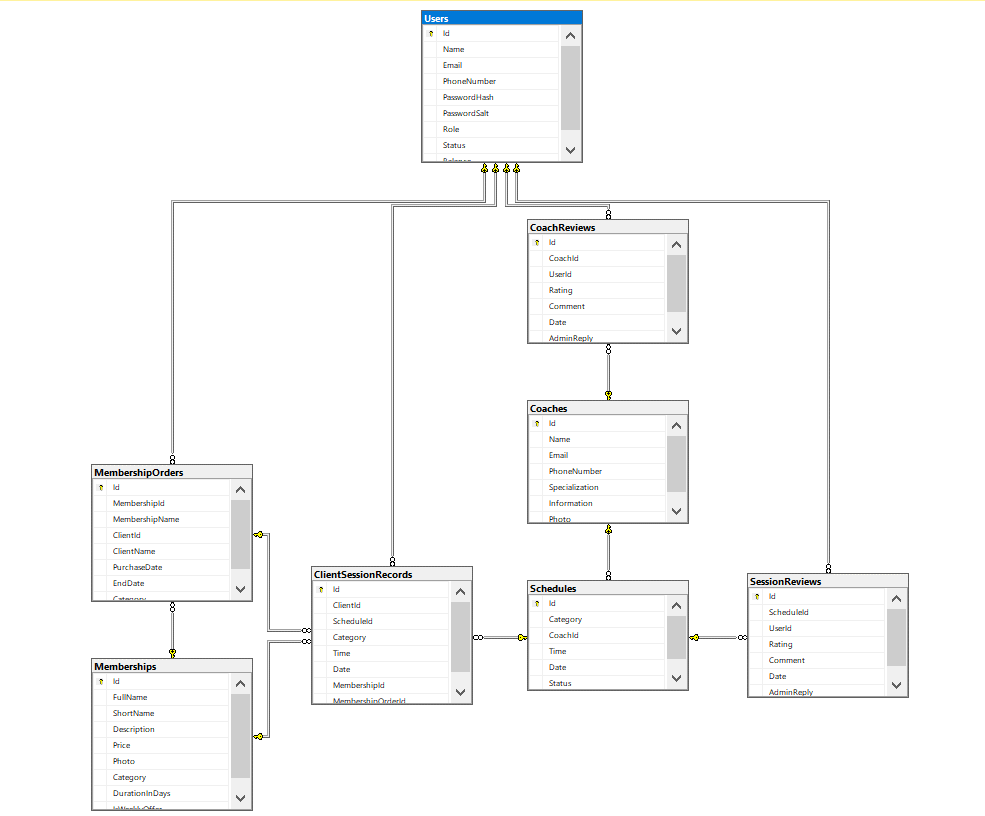


Рисунок 3.3 – Логическая модель базы данных SporticsDB

Описание таблиц базы данных представлено в таблице 3.3

Таблица 3.3 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Таблица | Содержание |
| Users | Содержит информацию о пользователях: имя, email, телефон, пароль, статус, баланс и т.д.. |
| Memberships | Содержит информацию о абонементах: название, категория, цена, описание, фото, срок действия и т.д. |
| Schedules | Расписание занятий с отображением тренера, категории, даты и времени. |
| Coaches | Информация о тренерах: имя, email, телефон, специализация, фото. |
| MembershipOrders | Заказы клиентов (абонементы, разовые тренеровки). |
| ClientSessionRecords | Записи клиентов на групповые занятия. |

**3.4. Проектирование функционала проекта**

Для общего представления функционального назначения системы используется диаграмма использования, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. На диаграмме использования применяются два типа основных сущностей: варианты использования и группы пользователей.

Разные группы пользователей в диаграмме называются актёрами, и обозначают любые сущности, использующие систему. Любая функция системы называется вариантом использования. Каждый вариант использования обозначает набор действий, который может быть использован актёром для взаимодействия с системой, и определяет набор действий, выполняемых этой системой.

Диаграмма использования представлена в приложении Б.

**3.5. Проектирование** **последовательностей проекта**

В целях визуализации взаимодействия объектов системы между собой во времени в едином сценарии использования используется ещё одна UML-диаграмма – диаграмма последовательностей. Данная диаграмма иллюстрирует, как различные части системы взаимодействуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования.

Для отображения течения времени используется линия жизни объекта, которая изображается с помощью штриховой линии, которая проводится вертикально вниз. С помощью линии жизни показывается период, в течение которого объект существует в системе. Сами объекты изображаются в виде прямоугольников, а сообщения, которыми они обмениваются – в виде линий со стрелками. Диаграмма последовательностей представлена в приложении В.

**3.6. Выводы по разделу**

В разделе была выбрана архитектура проекта, которая задаёт общий план проектирование продукта. Также для программного средства были спроектированы сущности базы данных, на основе которых получены их аналоги в языке программирования C#.

**4. Реализация программного средства**

**4.1. Основные классы программного средства**

Для выполнения технических задач программного средства должны быть реализованы следующие функции и соответствующие им классы и методы:

* выполнение авторизации и регистрации;
* добавление абонементов;
* управление заказами;
* покупка абонементов;
* управление отзывами;
* запись на тренировки.

Далее подробно рассмотрены каждые из необходимых для выполнения технических задач функции, а также созданные для их выполнения классы и методы и их функционал и реализация.

**4.2. Описание классов и методов программного средства**

**4.2.1. Авторизация и регистрация**

Вход и регистрация пользователей реализованы в отдельных окнах LoginWindow и RegistrationWindow. Эти окна связаны с LoginViewModel и RegistrationViewModel, которые управляют состоянием форм и обработкой данных. Обе модели используют сервис DataWorker для взаимодействия с базой данных.

При входе в систему используется метод Login, который проверяет введенные пользователем учетные данные. Для обеспечения безопасности применяется хэширование паролей, а затем их сравнение с данными из таблицы Users. В случае успешной авторизации устанавливается текущий пользователь, и открывается главное окно, предоставляя доступ к основному функционалу.

Регистрация новых пользователей осуществляется через метод Register, который проверяет уникальность логина и правильность введенной информации. Перед сохранением учетной записи в базе данных производится генерация соли и хэширование пароля, что повышает надежность хранения данных. Такой подход гарантирует защиту пользовательских учетных записей и предотвращает несанкционированный доступ.

Класс LoginViewModel:

* метод Login проверяет введенные пользователем логин и пароль, используя хэширование и сравнение с данными из таблицы Users;
* при успешной авторизации устанавливается текущий пользователь и открывается главное окно.

Класс RegisterViewModel:

* метод Register добавляет нового пользователя в базу, проверяя уникальность логина и корректность введенных данных;
* используется генерация соли и хэширование пароля перед сохранением в БД.

Хэширование паролей осуществляется через метод HashPassword, реализованный в HashHelper.

**4.2.2.** **Просмотр и добавление абонементов**

Отображение списка товаров осуществляется в представлении MembershipsWindow, управляемом моделью MembershipsViewModel. AddViewModel добавляет новый абонемент.

В модели MembershipsViewModel реализована работа с коллекцией Memberships, которая заполняется данными из соответствующей таблицы базы данных. Это позволяет пользователям получать актуальную информацию о доступных абонементах.

Для удобства поиска предусмотрен метод ApplyFilters, который фильтрует товары по категории и цене, помогая пользователям быстро находить подходящие предложения. Добавление новых абонементов осуществляется через модель AddViewModel с использованием метода ExecuteAdd, который отвечает за корректное внесение данных в систему.

Такая структура обеспечивает удобное управление абонементами и гибкость в выборе подходящих вариантов.

Класс MembershipsViewModel (приложение Г):

* содержит коллекцию Memberships, заполняемую из таблицы Memberships;
* метод ApplyFilters фильтрует товары по категории и цене;

Класс AddViewModel (приложение Г):

* метод ExecuteAdd добавляет абонементы

**4.2.3.** **Управление расписанием**

Данный функционал доступен администраторам в AdminWindow и SchedulesWindow, логика которых реализована в AdminViewModel и SchedulesViewModel.

В модели AdminViewModel реализованы методы EditSchedule и DeleteSchedule, которые позволяют эффективно управлять расписанием, редактируя и удаляя записи по необходимости. Это обеспечивает гибкость в настройке графика и удобство администрирования.

Модель SchedulesViewModel включает методы AddSchedule и GetDetails, предназначенные для добавления новых расписаний и просмотра детальной информации о них. Это помогает пользователям легко ориентироваться в записях и оперативно вносить изменения.

Взаимодействие с базой данных осуществляется через DataWorker, который обеспечивает надежную обработку данных и их корректное хранение. Такой подход гарантирует стабильность работы системы и удобство управления расписанием. Кроме того, использование DataWorker позволяет оптимизировать работу с данными, обеспечивая их быструю обработку и минимизируя вероятность ошибок. Такой подход делает систему более надежной и удобной для пользователей.

Класс AdminViewModel:

* методы EditSchedule, DeleteSchedule для управления расписанием;

Класс SchedulesViewModel:

* метод AddSchedule, GetDetails – для добавления и просмотра информации.

Взаимодействие с базой данных осуществляется через класс DataWorker.

**4.2.4. Покупка абонементов**

Покупка абонементов реализована через таблицу Memberships. Пользователь может оформить покупку выбранного абонемента при наличии необходимого баланса.

В модели ClientMembershipsViewModel реализован метод GetDetails, который позволяет пользователям открывать окно с подробной информацией о товаре. Это обеспечивает удобный доступ к характеристикам и деталям выбранного абонемента. Модель ClientMembershipInfoViewModel включает метод BuyMembership, который отвечает за добавление заказа и автоматическое списание баланса у пользователя. Такой механизм упрощает процесс покупки и делает его быстрым и удобным.

Взаимодействие с базой данных осуществляется через DataWorker, который обеспечивает надежное сохранение и извлечение информации о покупках. Это гарантирует корректную обработку данных и стабильную работу системы. Благодаря такому подходу пользователи могут легко управлять своими абонементами и отслеживать историю покупок.

Класс ClientMembershipsViewModel:

* метод GetDetails открывает окно информации о товаре;

Класс ClientMembershipInfoViewModel:

* метод BuyMembership добавляет заказ и списывает баланс у пользователя;

Класс DataWorker обеспечивает сохранение и извлечение данных о покупках.

**4.2.5. Управление отзывами пользователей**

Отзывы пользователей хранятся в таблицах CoachReview и SessionReview и отображаются в представлениях ReviewWindow и ReviewScheduleWindow.

В модели ReviewViewModel реализован метод SubmitReview, который дает пользователю возможность оставить отзыв о товаре. Этот процесс ограничен одним комментарием на единицу товара, что позволяет поддерживать объективность и точность оценок.

Метод LoadCoachReviews загружает отзывы о тренере, включая имя клиента, его комментарий и выставленный рейтинг. Это помогает пользователям ознакомиться с мнением других клиентов и принять более взвешенное решение при выборе специалиста.

Администратор платформы может взаимодействовать с отзывами с помощью метода SubmitAdminReply, который позволяет оставлять комментарии к пользовательским оценкам. Такой подход делает систему отзывов прозрачной и удобной для всех участников.

Класс ReviewViewModel:

* метод SubmitReview позволяет пользователю один раз оставить отзыв о товаре;
* метод LoadCoachReviews загружает отзывы о тренере, включая имя клиента, комментарий, рейтинг;
* метод SubmitAdminReply позволяет администратору оставить комментарий на отзыв.

**4.2.6. Запись на тренировки**

Запись на тренировки реализована через таблицу ClientSessionRecord. Пользователь может записаться на тренировку при наличии абонемента с подходящей категорией.

В модели ClientScheduleInfoViewModel реализован метод EvaluateAccess, который проверяет, подписан ли пользователь на выбранное занятие. Он также анализирует время проведения тренировки, чтобы предоставить доступ к отзывам только после ее завершения, обеспечивая объективность оценок.

Метод SignUp позволяет пользователям записываться на занятия, предварительно проверяя, не зарегистрированы ли они уже на тренировку. Кроме того, он учитывает временные ограничения, исключая возможность записи на прошедшие занятия, что помогает поддерживать актуальность расписания.

Такой подход обеспечивает удобное управление тренировочным процессом, позволяя пользователям легко планировать свое расписание и оставлять отзывы только после фактического посещения занятий. Это делает систему более прозрачной и удобной для всех участников.

Класс ClientScheduleInfoViewModel:

* метод EvaluateAccess проверяет, не подписан ли пользователь уже на данное занятие, определяет время, чтобы дать доступ к отзывам только после окончания тренировки;
* метод SignUp позволяет записаться на занятие, проверяет, записан ли пользователь уже на тренировку, определяет время, чтобы, не проводилась запись на прошедшие занятия;

**4.3 Выводы по разделу**

В данном разделе представлен процесс реализации программного средства Sportics, спроектированного на основе требований и логики предметной области. Были описаны ключевые классы и методы, отвечающие за выполнение авторизации и регистрации, просмотр и добавление абонементов, управление расписанием, покупку абонементов и управление отзывами пользователей. Все элементы реализованы по паттерну MVVM.

**5. Тестирование, проверка работоспособности и анализ полученных результатов**

**5.1. Тестирование авторизации**

При авторизации программное средство учитывает многие факторы. Одним из таких является проверка корректности заполнения полей. Например, поля почты и пароля не могут быть пустым. Пример попытки ввода некорректных данных представлен на рисунке 5.1.

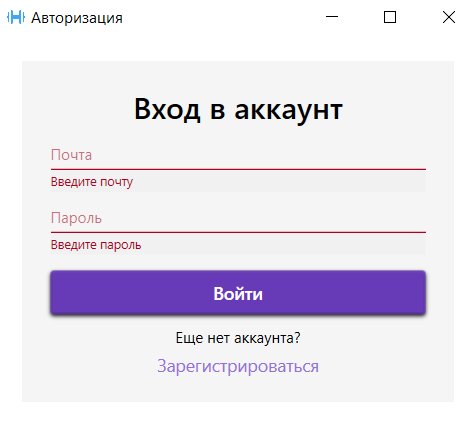


Рисунок 5.1 – Обработка пустых полей

Таким образом программное средство подсказывает пользователю, что ему необходимо изменить чтобы успешно войти в систему.

**5.2. Тестирование создания абонемента**

В форме создания абонемента применяются следующие правила валидации:

* все поля обязательны к заполнению;
* значения должны соответствовать типу данных или формату;
* числовые значения должны входить в определенный диапазон.

Тестирование системы валидации окна приведено на рисунке 5.2.

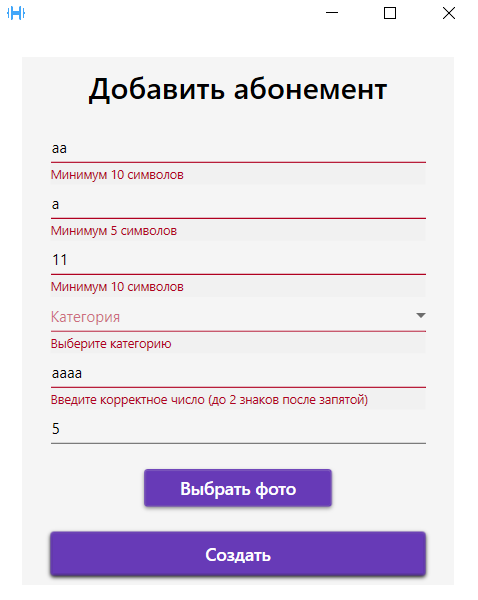


Рисунок 5.2 – Обработка полей добавления

Таким образом, пользователь не сможет указать дату конца бронирования раньше даты начала, так как такое поведение подозрительно и, предположительно, не соответствует ожидаемому результату.

**5.3. Тестирование блокировки пользователя**

В случае недобросовестных действий админ имеет возможность заблокировать конкретного пользователя. В этом случае программное средство изменит статус пользователя на заблокирован и при следующей авторизации пользователь увидит сообщение, которое представлено на рисунке 5.3.

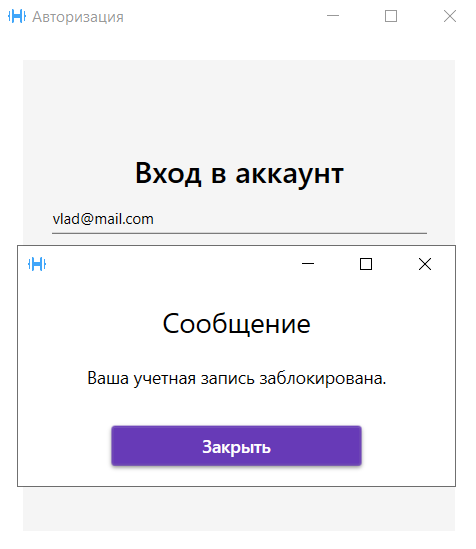


Рисунок 5.3 – Сообщение о блокировке

Благодаря такого рода проверке, программное средство позволяет поддерживать порядок и обеспечивать безопасность использования.

**5.4. Тестирование добавления пользователя**

При регистрации пользователь должен соблюдать множество правил, так как процесс регистрации – это основа взаимодействия с приложением для новых пользователей. К таковым правилам можно отнести:

* заполнить все поля;
* минимальная длина каждого поля 2 символа;
* максимальная длина каждого поля 50 символов;
* минимальная длина пароля 5 символов;
* номер должен соответствовать формату;
* пароль и повтор пароля должны совпадать;
* имя, фамилия, отчество должны содержать только буквы;
* в базе данных не должно быть пользователей с такой же почтой.

В случае несоблюдения хотя бы одного правила, программное средство будет выводить сообщения об ошибках, которое представлено на рисунке 5.4.

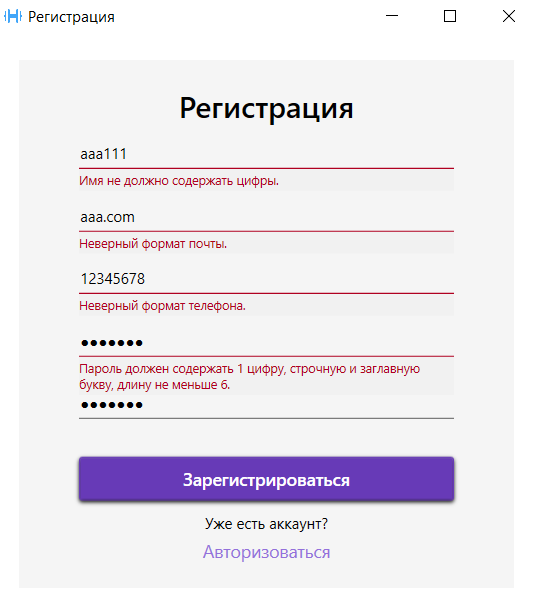


Рисунок 5.5 – Обработка полей добавления сотрудника

**5.5. Тестирование удаления взаимосвязанных данных**

При удаление определенного элемента одной таблицы базы данных происходит проверка на связи с элементами других таблиц. Например, при удалении тренера происходит проверка на связь с занятиями (расписанием), а при удалении заказа происходит проверка на запись клиента. Пример попытки удаления тренера представлен на рисунке 5.6.

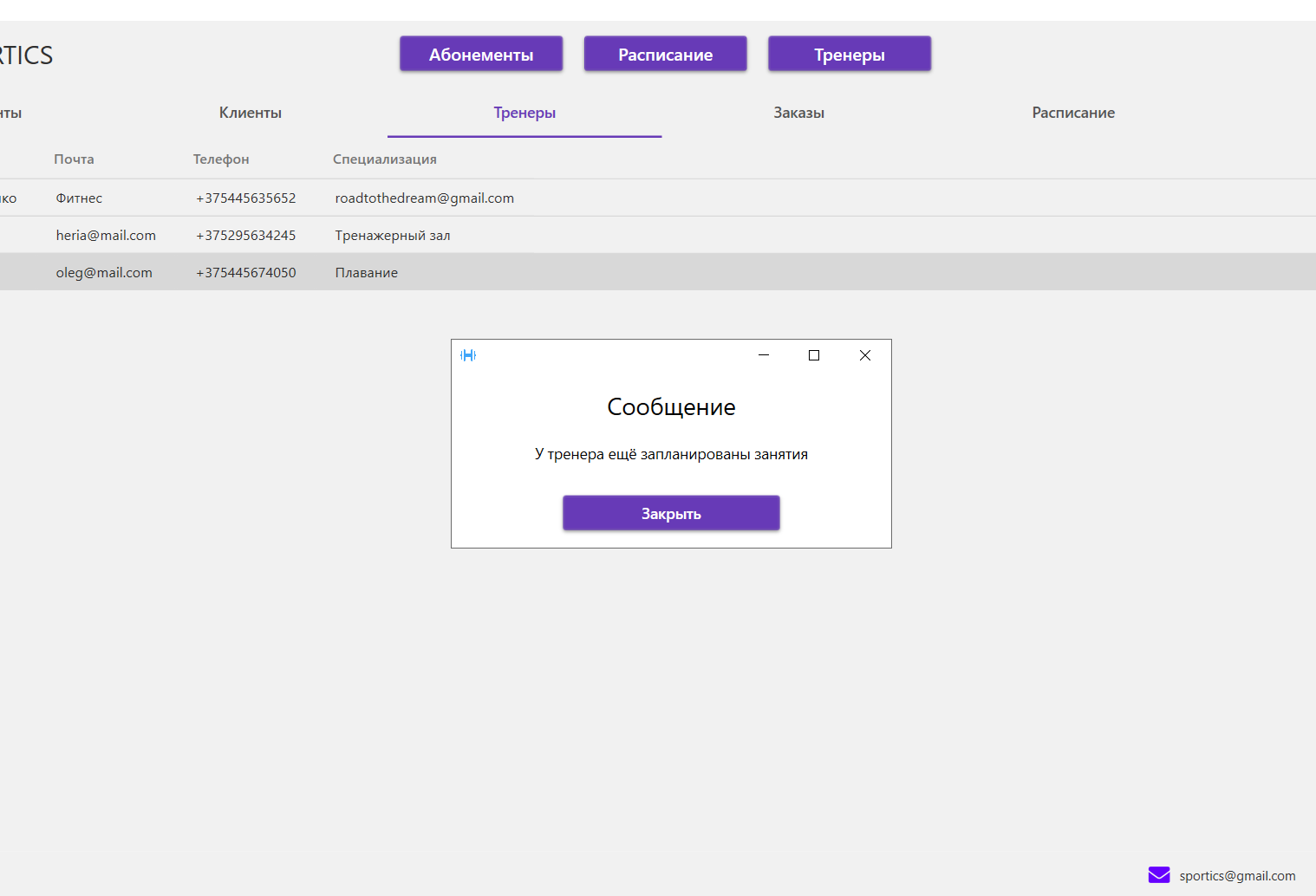


Рисунок 5.6 – Попытка удаления тренера с занятием

**5.6. Выводы по разделу:**

В разделе описан процесс тестирования продукта. Были проверены случаи негативного и положительного тестирования. Были проверены все случаи валидации вводимых пользователем данных.

1. **Руководство по установке и использованию**
   1. **Использование**

При запуске программного средства пользователь попадает на страницу авторизации, содержащая форму входа.

Стартовая страница позволяет зарегистрированному пользователю совершить вход в аккаунт, для этого ему необходимо ввести корректные логин и пароль, после чего пользователь будет перенаправлен на главную страницу.

Окно авторизации представлено на рисунках 6.1.

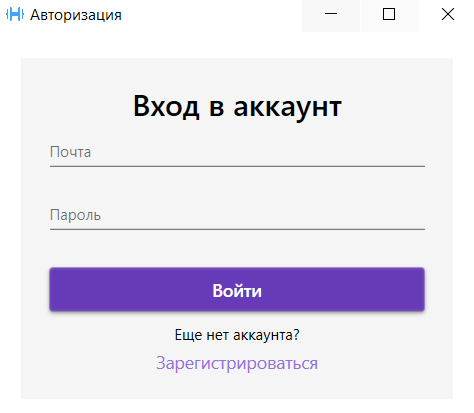


Рисунок 6.1 – Окно авторизации

Находясь на главной странице пользователь получает доступ к каталогу предложений недели. Главная страница представлена на рисунке рисунке 6.2.

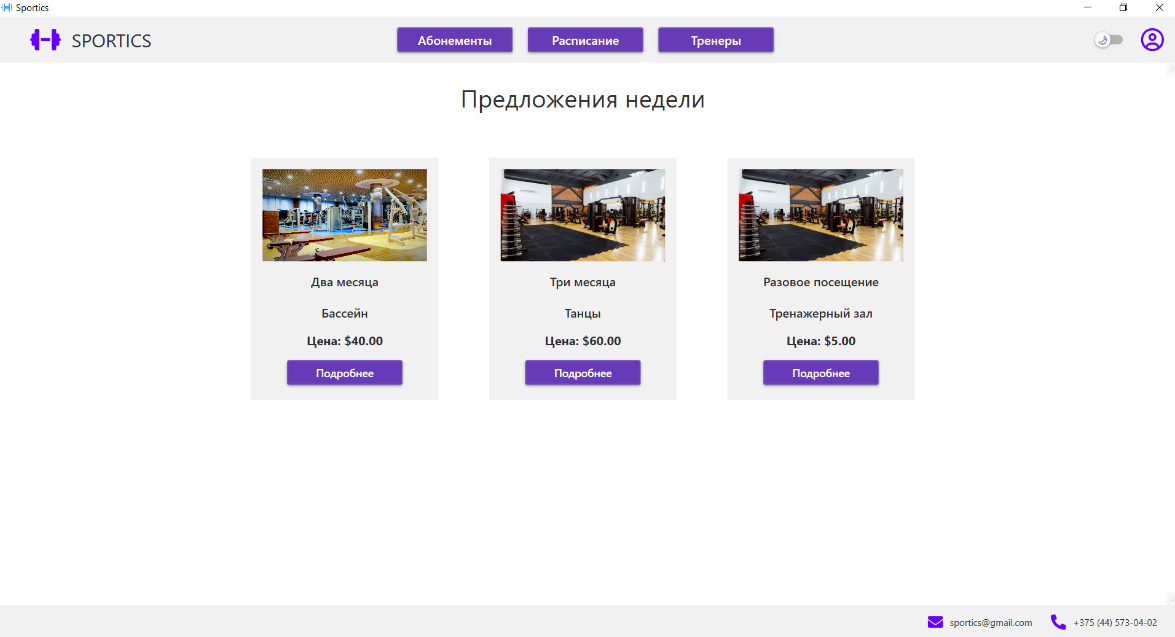


Рисунок 6.2 – Главная страница пользователя

Вверху страницы находятся кнопки маршрутизации, по которым можно перейти на нужные страницы. В правом верхнем углу находятся кнопки темы, личного кабинета, а также выбор языка. Личный кабинет предоставляет возможность выйти из приложения.

На странице абонементов находятся их элементы с кнопкой, позволяющей узнать о предложении подробнее, а также купить этот абонемент. Это представлено на рисунке 6.3.

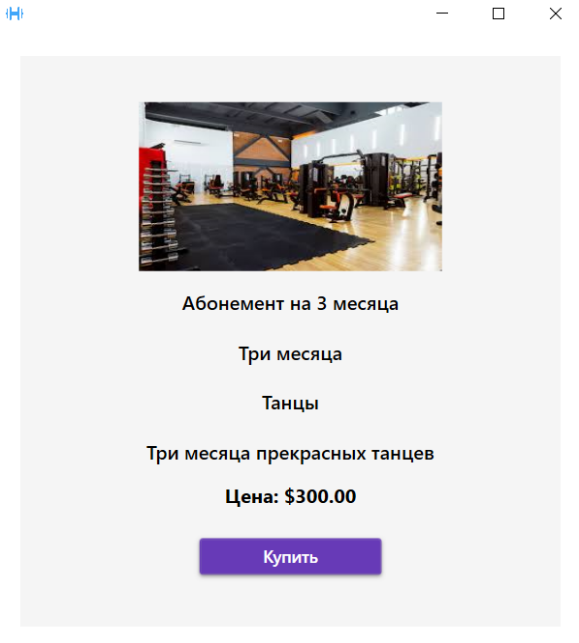


Рисунок 6.3 – Окно абонемента

После успешного заказа пользователь может перейти в личный кабинет (приложение Г), где будут находиться все активные заказы. Окно личного кабинета предсталвено на рисунке 6.4.

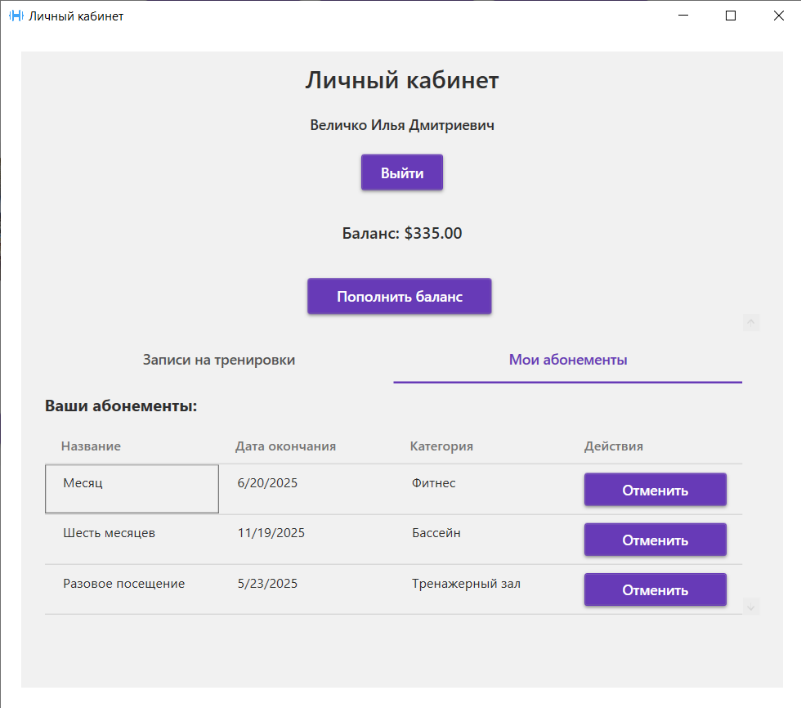


Рисунок 6.4 – Окно личного кабинета

Так же пользователь может выбрать таблицу, в которой будут находиться записи на групповые тренировки. Пример окна представлен на рисунке 6.5.

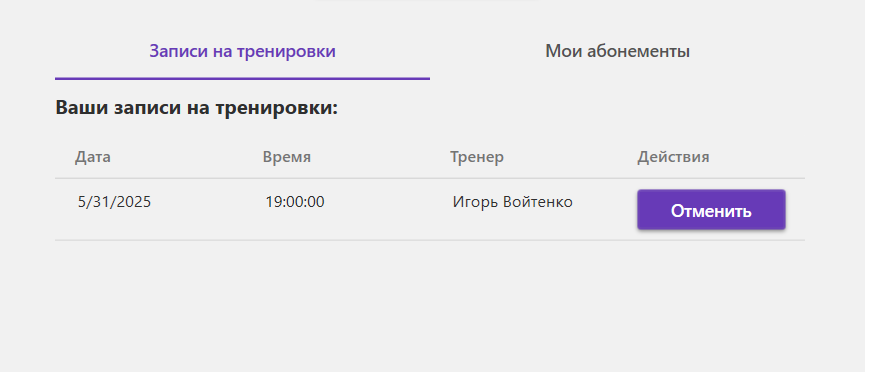


Рисунок 6.5 – Окно записей на тренировки

В этом же окне пользователь может пополнить свой счет. Окно пополнения баланса представлено на рисунке 6.6.

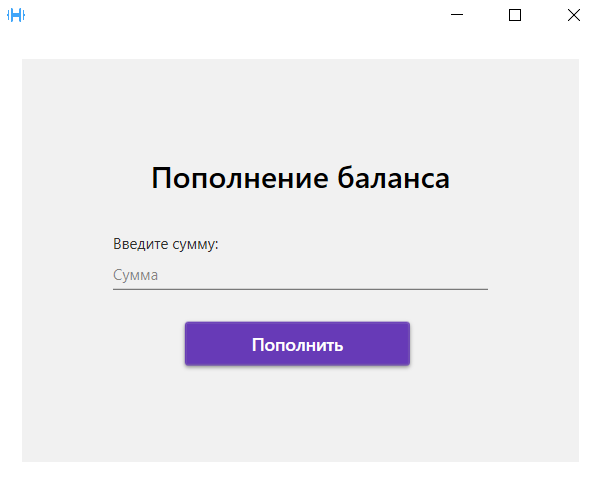


Рисунок 6.6 – Окно пополнения баланса

Окно адниминстратора сильно отличается от окна пользователя. Вверху окна содержатся кнопки маршрутизации, темы и личного кабинета. С их помощью можно перейти в окна редактирования абонементов, тренеров и расписания.

Внешний вид страницы аднимистратора представлен на рисунке 6.7.

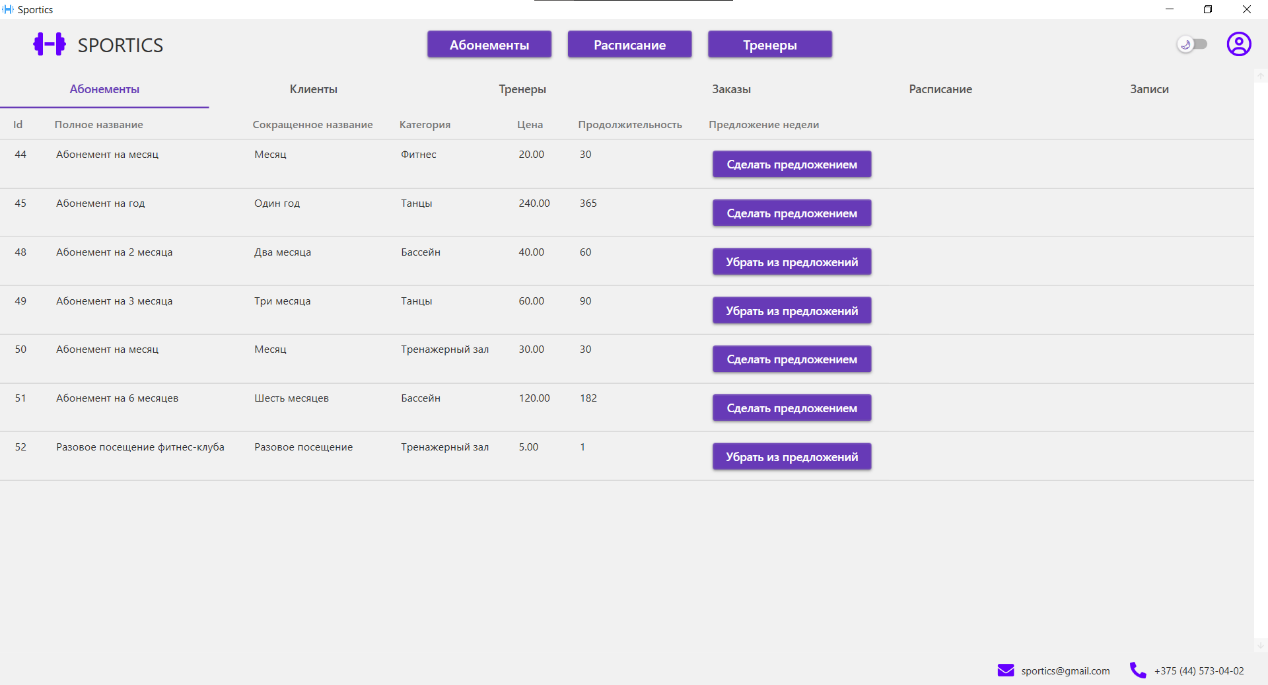


Рисунок 6.7 – Страница администратора

Администратор имеет возможность управлять заказами. В окне выводятся все заказанные абонементы всех пользователей. Администратор может отменить заказ, если это необходимо. Окно представлено на рисунке 6.8.

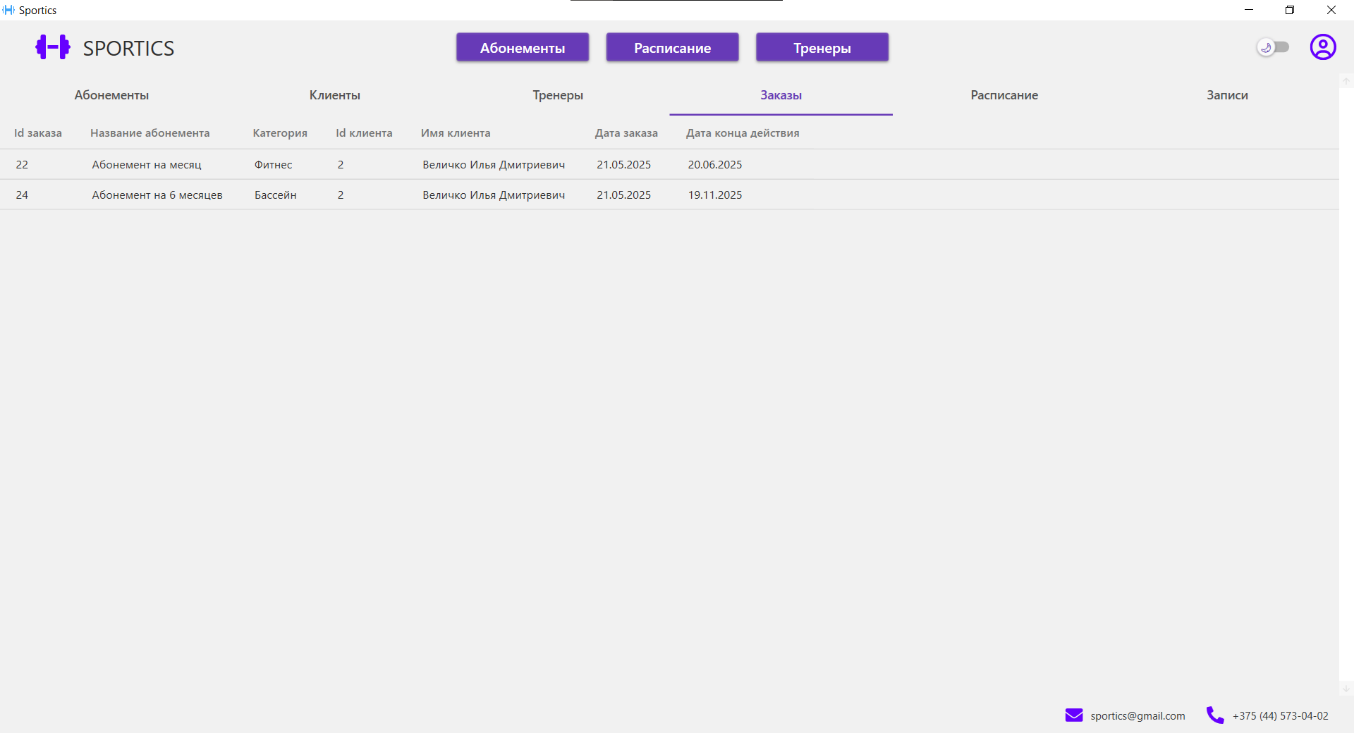


Рисунок 6.7 – Окно заказы

Администратор ответственен в том числе и за управление отзывами. Он может ответить или же вовсе удалить отзыв из базы данных. Окно отзывов на тренеров представлено на рисунке 6.8.

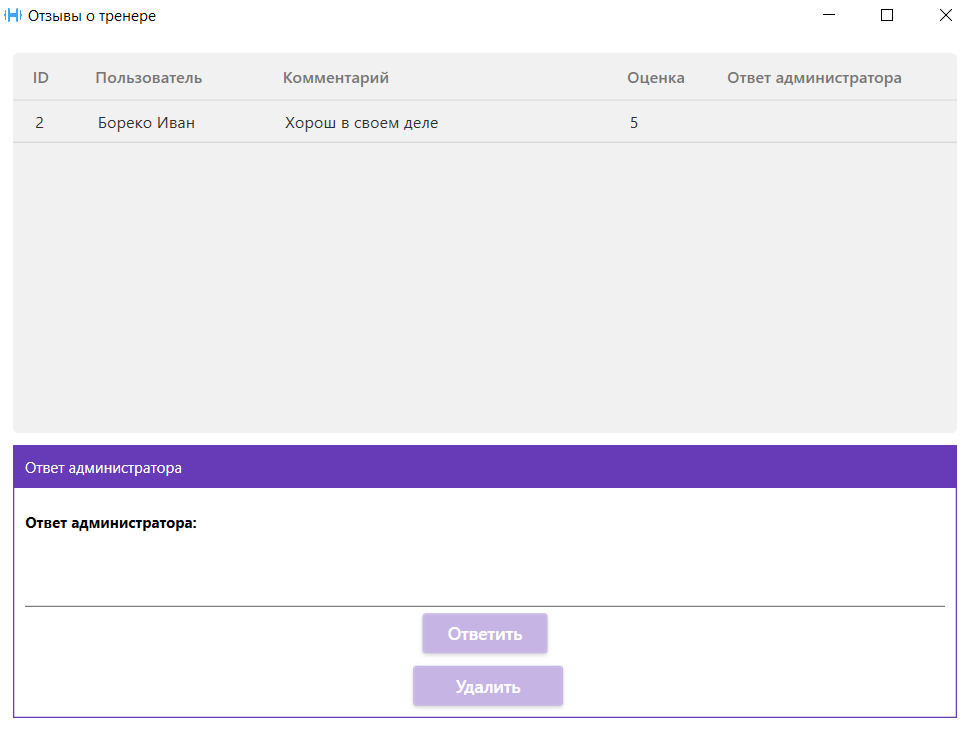


Рисунок 6.8 – Окно отзывы о тренере

Администрирование комментариев позволяет поддерживать порядок в приложении, предотвращая появление недопустимого контента и обеспечивая качественную обратную связь между пользователями. Администраторы могут модерировать отзывы, удалять некорректные или оскорбительные сообщения, а также отвечать на вопросы пользователей, поддерживая активное взаимодействие на платформе.

* 1. **Выводы по разделу:**

Этот раздел подробно объясняет процесс использования программного средства, предоставляя наглядные примеры работы с каждой функцией. В нем описаны основные возможности программного средства и принципы его работы, а также ключевые аспекты взаимодействия с интерфейсом. Кроме того, рассматриваются практические сценарии применения, которые позволяют пользователю эффективно решать различные задачи.

Каждая функция программного средства сопровождается инструкцией, демонстрирующей последовательность действий, необходимых для достижения желаемого результата. В разделе также приведены рекомендации по оптимальному использованию функционала, а также советы по устранению возможных затруднений.

Таким образом, данный раздел служит подробным руководством, позволяющим пользователю уверенно и эффективно использовать весь доступный функционал.

**Заключение**

В ходе выполнения курсового проекта было разработано программное средство для управления фитнесс-клубом на языке C# с использованием технологий Entity Framework Core, WPF.

При разработке программного средства были выполнены все пункты из указанного списка предполагаемого основного функционала приложения, а именно:

* создана база данных;
* реализована авторизация и регистрация;
* просмотр абонементов;
* управление абонементами;
* просмотр заказов;
* запись на тренировку
* просмотр каталога тренеров;
* редактирование информации о абонементах;
* удаление занятий;
* добавление отзывов;
* управление статусом пользователя;
* управление отзывами.

Тестирование программы показало, что она работает корректно и выполняет все свои функции.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

**Список литературы**

1. Онлайн– ресурс “smartshell.gg” [Электронный ресурс] – Режим доступа smartshell.gg – Дата доступа: 25.04.2025
2. Онлайн– ресурс “astrumsoft.com” [Электронный ресурс] – Режим доступа astrumsoft.com – Дата доступа: 25.04.2025
3. Онлайн– ресурс “lightshell.ru” [Электронный ресурс] – Режим доступа lightshell.ru – Дата доступа: 27.04.2025
4. Онлайн– ресурс “ggLeap” [Электронный ресурс] – Режим доступа esportsentertainmentgroup.com – Дата доступа: 26.04.2025
5. Онлайн– ресурс “smartlaunch” [Электронный ресурс] – Режим доступа smartlaunch.com – Дата доступа: 26.04.2025
6. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
7. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 28.04.2025
8. Работа с Entity Framework Core [Электронный ресурс] – https://professorweb.ru/my/entity– framework/6/level1/ – Дата доступа 23.04.2025

**ПРИЛОЖЕНИЕ А**

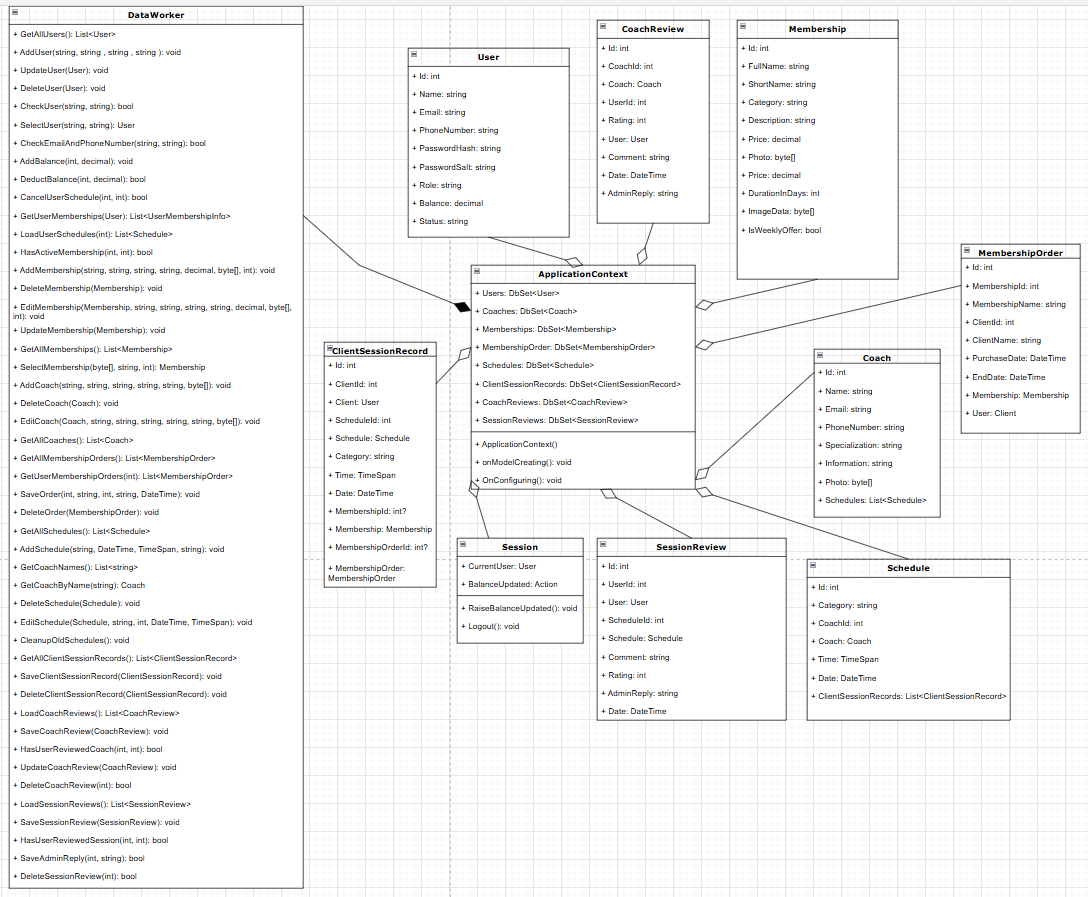


Рисунок А.1 – диаграмма классов

**ПРИЛОЖЕНИЕ Б**

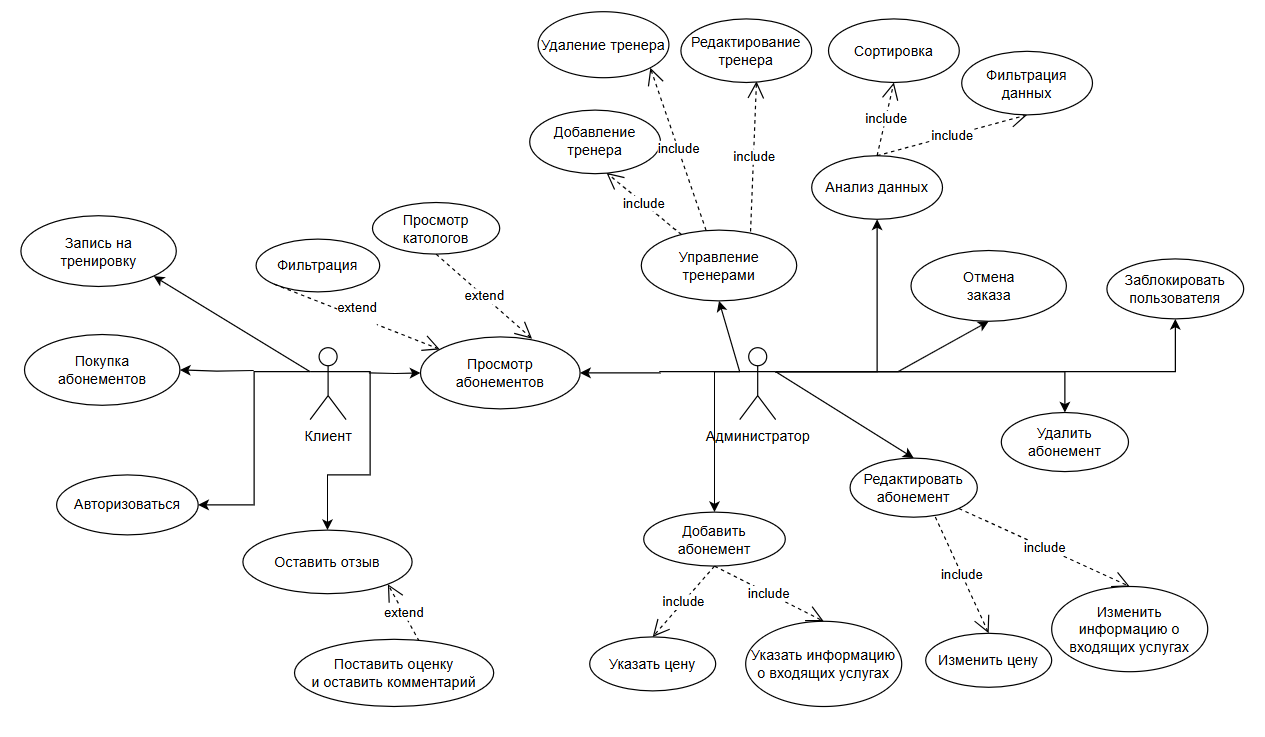


Рисунок Б.1 – диаграмма вариантов использования

**ПРИЛОЖЕНИЕ В**

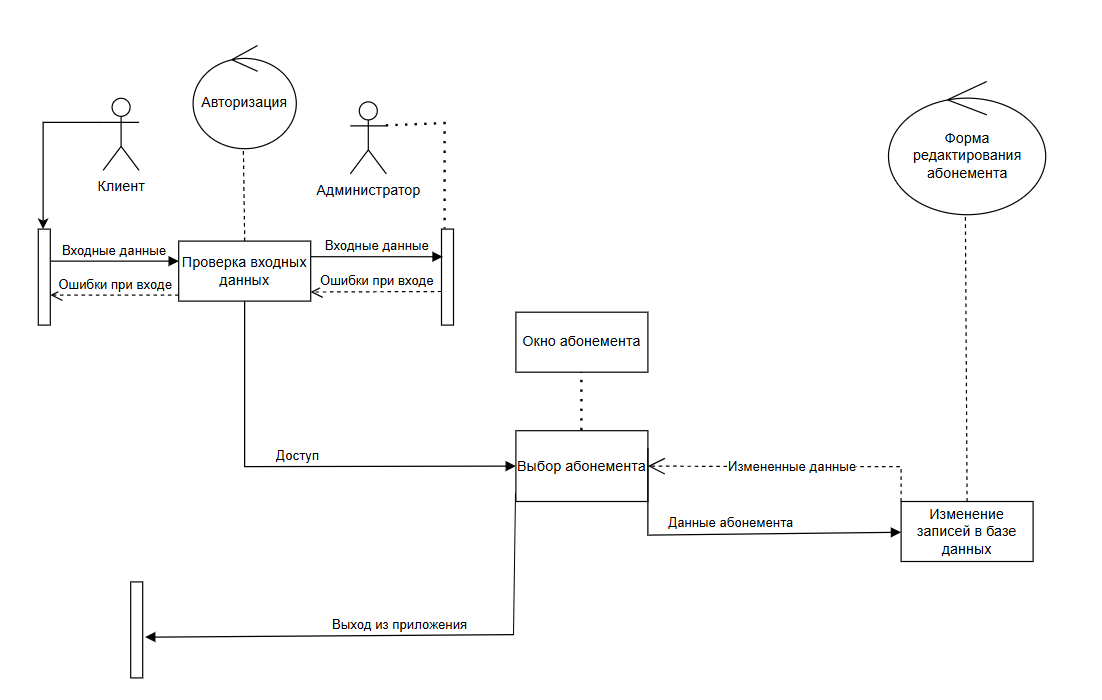


Рисунок В.1 – диаграмма последовательности

**ПРИЛОЖЕНИЕ Г**

Листинг 1: Класс AddViewModel

using Microsoft.Win32;

using Sportics.Model;

using Sportics.View;

using System;

using System.ComponentModel;

using System.Globalization;

using System.IO;

using System.Text.RegularExpressions;

using System.Windows;

using System.Windows.Forms;

using System.Windows.Input;

namespace Sportics.ViewModel

{

public class AddViewModel : BaseViewModel, IDataErrorInfo

{

public string FullName

{

get => fullName;

set

{

fullName = value;

OnPropertyChanged(nameof(FullName));

IsValidationActive = false;

}

}

public string ShortName

{

get => shortName;

set

{

shortName = value;

OnPropertyChanged(nameof(ShortName));

IsValidationActive = false;

}

}

public string Category

{

get => category;

set

{

category = value;

OnPropertyChanged(nameof(Category));

IsValidationActive = false;

}

}

public string Description

{

get => description;

set

{

description = value;

OnPropertyChanged(nameof(Description));

IsValidationActive = false;

}

}

public string Price

{

get => price;

set

{

price = value;

OnPropertyChanged(nameof(Price));

IsValidationActive = false;

}

}

public byte[] PhotoData

{

get => photoData;

set

{

photoData = value;

OnPropertyChanged(nameof(PhotoData));

IsValidationActive = false;

}

}

public string DurationInDays

{

get => durationInDays;

set

{

durationInDays = value;

OnPropertyChanged(nameof(DurationInDays));

IsValidationActive = false;

}

}

private string fullName;

private string shortName;

private string category;

private string description;

private string price;

private byte[] photoData;

private string durationInDays;

private bool isValidationActive = false;

public bool IsValidationActive

{

get => isValidationActive;

set

{

isValidationActive = value;

OnPropertyChanged(nameof(IsValidationActive));

}

}

public string Error => null;

public string this[string columnName]

{

get

{

if (!IsValidationActive) return null;

switch (columnName)

{

case nameof(FullName):

if (string.IsNullOrWhiteSpace(FullName) || FullName.Length < 10)

return "Минимум 10 символов";

break;

case nameof(ShortName):

if (string.IsNullOrWhiteSpace(ShortName) || ShortName.Length < 5)

return "Минимум 5 символов";

break;

case nameof(Description):

if (string.IsNullOrWhiteSpace(Description) || Description.Length < 10)

return "Минимум 10 символов";

break;

case nameof(Category):

if (string.IsNullOrWhiteSpace(Category))

return "Выберите категорию";

break;

case nameof(Price):

if (string.IsNullOrWhiteSpace(Price))

return "Введите цену";

if (!Regex.IsMatch(Price, @"^\d+([.,]\d{1,2})?$"))

return "Введите корректное число (до 2 знаков после запятой)";

if (!decimal.TryParse(Price, NumberStyles.Number, CultureInfo.CurrentCulture, out decimal priceVal))

return "Неверный формат цены";

if (priceVal <= 0)

return "Цена должна быть больше 0";

if (priceVal > 1500)

return "Максимум 1500";

break;

case nameof(PhotoData):

if (PhotoData == null || PhotoData.Length == 0)

return "Выберите фото";

break;

case nameof(DurationInDays):

if (int.Parse(DurationInDays) <= 0)

return "Длительность должна быть больше нуля";

if (int.Parse(DurationInDays) >= 366)

return "Слишком большая длительность";

break;

}

return null;

}

}

public ICommand SelectPhotoCommand { get; }

public ICommand AddCommand { get; }

public AddViewModel()

{

SelectPhotoCommand = new RelayCommand(obj => ExecuteSelectPhoto());

AddCommand = new RelayCommand(obj => ExecuteAdd());

}

private void ExecuteSelectPhoto()

{

Microsoft.Win32.OpenFileDialog openFileDialog = new Microsoft.Win32.OpenFileDialog

{

Filter = "Image files (\*.jpg;\*.png)|\*.jpg;\*.png"

};

if (openFileDialog.ShowDialog() == true)

{

PhotoData = File.ReadAllBytes(openFileDialog.FileName);

}

}

public event Action RequestClose;

private void ExecuteAdd()

{

IsValidationActive = true;

// Проверка всех полей

OnPropertyChanged(nameof(FullName));

OnPropertyChanged(nameof(ShortName));

OnPropertyChanged(nameof(Category));

OnPropertyChanged(nameof(Description));

OnPropertyChanged(nameof(Price));

OnPropertyChanged(nameof(PhotoData));

OnPropertyChanged(nameof(DurationInDays));

if (!string.IsNullOrEmpty(this[nameof(FullName)]) ||

!string.IsNullOrEmpty(this[nameof(ShortName)]) ||

!string.IsNullOrEmpty(this[nameof(Category)]) ||

!string.IsNullOrEmpty(this[nameof(Description)]) ||

!string.IsNullOrEmpty(this[nameof(Price)]) ||

!string.IsNullOrEmpty(this[nameof(DurationInDays)]))

return;

if (!string.IsNullOrEmpty(this[nameof(PhotoData)]))

{

string message = "Выберите фото";

MessageWindow messageWindow = new MessageWindow();

MessageViewModel viewModel = new MessageViewModel(message);

messageWindow.DataContext = viewModel;

viewModel.RequestClose += () => messageWindow.Close();

messageWindow.Owner = System.Windows.Application.Current.MainWindow;

messageWindow.WindowStartupLocation = WindowStartupLocation.CenterOwner;

messageWindow.ShowDialog();

return;

}

DataWorker.AddMembership(FullName, ShortName, Category, Description, decimal.Parse(Price), PhotoData, int.Parse(DurationInDays));

RequestClose?.Invoke();

}

}

}

Листинг 2: Класс AccountViewModel

using Sportics.Model;

using Sportics.View;

using System;

using System.Collections.ObjectModel;

using System.Linq;

using System.Windows;

using System.Windows.Input;

namespace Sportics.ViewModel

{

public class AccountViewModel : BaseViewModel

{

public bool IsClient => Session.CurrentUser?.Role?.ToLower() == "клиент";

public string UserName => Session.CurrentUser?.Name ?? "Админ";

public decimal UserBalance => Session.CurrentUser?.Balance ?? 0;

public ICommand ExitCommand { get; }

public ICommand BalanceCommand { get; }

public ICommand CancelRegistrationCommand { get; }

public ICommand CancelScheduleCommand { get; }

private ObservableCollection<Schedule> \_userSchedules;

public ObservableCollection<Schedule> UserSchedules

{

get => \_userSchedules;

set

{

\_userSchedules = value;

OnPropertyChanged();

}

}

private ObservableCollection<UserMembershipInfo> \_userMemberships;

public ObservableCollection<UserMembershipInfo> UserMemberships

{

get => \_userMemberships;

set

{

\_userMemberships = value;

OnPropertyChanged();

}

}

private Schedule \_selectedSchedule;

public Schedule SelectedSchedule

{

get => \_selectedSchedule;

set

{

\_selectedSchedule = value;

OnPropertyChanged();

}

}

public AccountViewModel()

{

ExitCommand = new RelayCommand(\_ => Exit());

BalanceCommand = new RelayCommand(\_ => OpenBalance());

CancelScheduleCommand = new RelayCommand(obj => CancelSchedule(obj));

Session.BalanceUpdated += OnBalanceUpdated;

LoadUserSchedules();

}

private void LoadUserSchedules()

{

var userId = Session.CurrentUser?.Id ?? 0;

var schedules = DataWorker.LoadUserSchedules(userId);

UserSchedules = new ObservableCollection<Schedule>(schedules);

UserMemberships = new ObservableCollection<UserMembershipInfo>(

DataWorker.GetUserMemberships(Session.CurrentUser));

OnPropertyChanged(nameof(UserSchedules));

}

private void CancelSchedule(object obj)

{

if (obj is Schedule schedule)

{

bool success = DataWorker.CancelUserSchedule(Session.CurrentUser.Id, schedule.Id);

if (success)

{

UserSchedules.Remove(schedule);

OnPropertyChanged(nameof(UserSchedules));

ShowMessage("Запись отменена");

}

else

{

ShowMessage("Не удалось отменить запись");

}

}

}

private void OnBalanceUpdated()

{

OnPropertyChanged(nameof(UserBalance));

}

private void Exit()

{

Session.Logout();

LoginWindow loginWindow = new LoginWindow();

Application.Current.MainWindow = loginWindow;

Application.Current.Windows

.OfType<Window>()

.FirstOrDefault(w => w is MainWindow || w is AdminWindow || w is AccountWindow

|| w is CoachesWindow || w is MembershipsWindow || w is SchedulesWindow || w is ClientCoachesWindow

|| w is ClientMembershipsWindow || w is ClientSchedulesWindow)?

.Close();

Application.Current.MainWindow.Show();

}

private void OpenBalance()

{

BalanceWindow balanceWindow = new BalanceWindow();

BalanceViewModel balanceViewModel = new BalanceViewModel();

balanceWindow.DataContext = balanceViewModel;

balanceViewModel.RequestClose += () => balanceWindow.Close();

balanceWindow.Owner = Application.Current.MainWindow;

balanceWindow.WindowStartupLocation = WindowStartupLocation.CenterOwner;

balanceWindow.ShowDialog();

}

private void ShowMessage(string message)

{

var messageWindow = new MessageWindow();

var viewModel = new MessageViewModel(message);

messageWindow.DataContext = viewModel;

viewModel.RequestClose += () => messageWindow.Close();

messageWindow.Owner = Application.Current.MainWindow;

messageWindow.WindowStartupLocation = WindowStartupLocation.CenterOwner;

messageWindow.ShowDialog();

}

~AccountViewModel()

{

Session.BalanceUpdated -= OnBalanceUpdated;

}

}

public class UserMembershipInfo

{

public Membership Membership { get; set; }

public DateTime EndDate { get; set; }

}

}

Листинг 3: Класс MembershipsViewModel

using Sportics.Helper;

using Sportics.Model;

using Sportics.View;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Linq;

using System.Text.RegularExpressions;

using System.Windows;

using System.Windows.Input;

namespace Sportics.ViewModel

{

public class MembershipsViewModel : BaseViewModel, IDataErrorInfo

{

public List<Membership> Memberships { get; set; }

public ObservableCollection<Membership> FilteredMemberships { get; set; }

public List<string> Categories { get; set; } = new List<string>

{

"Все категории", "Фитнес", "Йога", "Бассейн", "Тренажерный зал", "Танцы"

};

public ObservableCollection<string> Languages { get; } = new ObservableCollection<string> { "RU", "EN" };

private string \_selectedLanguage = "RU";

public string SelectedLanguage

{

get => \_selectedLanguage;

set

{

if (\_selectedLanguage != value)

{

\_selectedLanguage = value;

OnPropertyChanged();

LocalizationManager.ChangeCulture(value);

}

}

}

private readonly ThemeService ThemeService = ThemeService.Instance;

public bool IsDarkTheme

{

get => ThemeService.IsDarkTheme;

set

{

if (value)

ThemeService.SetDarkTheme();

else

ThemeService.SetLightTheme();

OnPropertyChanged();

}

}

private string priceFrom;

public string PriceFrom

{

get => priceFrom;

set

{

priceFrom = value;

OnPropertyChanged(nameof(PriceFrom));

IsValidationActive = false;

}

}

private string priceTo;

public string PriceTo

{

get => priceTo;

set

{

priceTo = value;

OnPropertyChanged(nameof(PriceTo));

IsValidationActive = false;

}

}

private string selectedCategory = "Все категории";

public string SelectedCategory

{

get => selectedCategory;

set

{

selectedCategory = value;

OnPropertyChanged(nameof(SelectedCategory));

}

}

private bool isValidationActive = false;

public bool IsValidationActive

{

get => isValidationActive;

set

{

isValidationActive = value;

OnPropertyChanged(nameof(IsValidationActive));

}

}

public string Error => null;

public string this[string columnName]

{

get

{

if (!IsValidationActive) return null;

switch (columnName)

{

case nameof(PriceFrom):

if (!string.IsNullOrWhiteSpace(PriceFrom) && !Regex.IsMatch(PriceFrom, @"^\d+$"))

return "Введите положительное число";

break;

case nameof(PriceTo):

if (!string.IsNullOrWhiteSpace(PriceTo) && !Regex.IsMatch(PriceTo, @"^\d+$"))

return "Введите положительное число";

break;

}

return null;

}

}

public ICommand OpenAccountCommand { get; }

public ICommand OpenAddMembershipCommand { get; }

public ICommand OpenAdminCommand { get; }

public ICommand DetailsCommand { get; }

public ICommand ApplyFilterCommand { get; }

public MembershipsViewModel()

{

OpenAccountCommand = new RelayCommand(obj => OpenAccount());

OpenAddMembershipCommand = new RelayCommand(obj => OpenAddMembership());

OpenAdminCommand = new RelayCommand(obj => OpenAdmin());

DetailsCommand = new RelayCommand(obj => GetDetails((Membership)obj));

ApplyFilterCommand = new RelayCommand(obj => ApplyFilter());

AllMemberships();

}

private void AllMemberships()

{

Memberships = DataWorker.GetAllMemberships();

FilteredMemberships = new ObservableCollection<Membership>(Memberships);

OnPropertyChanged(nameof(Memberships));

OnPropertyChanged(nameof(FilteredMemberships));

}

private void ApplyFilter()

{

IsValidationActive = true;

OnPropertyChanged(nameof(PriceFrom));

OnPropertyChanged(nameof(PriceTo));

if (!string.IsNullOrEmpty(this[nameof(PriceFrom)]) || !string.IsNullOrEmpty(this[nameof(PriceTo)]))

return;

decimal.TryParse(PriceFrom, out decimal from);

decimal.TryParse(PriceTo, out decimal to);

List<Membership> filtered = Memberships.Where(m =>

(string.IsNullOrEmpty(SelectedCategory) || SelectedCategory == "Все категории" || m.Category == SelectedCategory) &&

(string.IsNullOrWhiteSpace(PriceFrom) || m.Price >= from) &&

(string.IsNullOrWhiteSpace(PriceTo) || m.Price <= to)).ToList();

FilteredMemberships = new ObservableCollection<Membership>(filtered);

OnPropertyChanged(nameof(FilteredMemberships));

}

private void OpenAddMembership()

{

AddWindow window = new AddWindow

{

Owner = Application.Current.MainWindow,

WindowStartupLocation = WindowStartupLocation.CenterOwner

};

window.ShowDialog();

AllMemberships();

}

private void OpenAdmin()

{

AdminWindow adminWindow = new AdminWindow();

Application.Current.MainWindow = adminWindow;

Application.Current.Windows

.OfType<Window>()

.FirstOrDefault(w => w is MembershipsWindow)?

.Close();

Application.Current.MainWindow.Show();

}

private void OpenAccount()

{

AccountWindow accountWindow = new AccountWindow();

accountWindow.Owner = Application.Current.MainWindow;

accountWindow.WindowStartupLocation = WindowStartupLocation.CenterOwner;

accountWindow.ShowDialog();

}

private void GetDetails(Membership membership)

{

MembershipInfoWindow window = new MembershipInfoWindow();

MembershipInfoViewModel viewModel = new MembershipInfoViewModel(membership);

window.DataContext = viewModel;

viewModel.RequestClose += () => window.Close();

window.Owner = Application.Current.MainWindow;

window.WindowStartupLocation = WindowStartupLocation.CenterOwner;

window.ShowDialog();

AllMemberships();

}

}

}