МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

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

Тема: Программное средство “Студия вокала”

Исполнитель

студент (ка) 2 курса группы 6 Трикоза Татьяна Владиславовна

(Ф.И.О.)

Руководитель работы ст. преп. Сухорукова И.Г.

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

Курсовой проект защищен с оценкой

Председатель Мущук А.Н.

(подпись)

Минск 2024

Содержание

[ВВЕДЕНИЕ 3](#_Toc16633)

[1. Аналитический обзор литературы 4](#_Toc4020)

[1.1 MEDLEY 4](#_Toc2501)

[1.2 Фа Соль 5](#_Toc19515)

[1.3 Cult Academy 8](#_Toc2186)

[1.4 Вывод 9](#_Toc18294)

[2. Анализ требований к программному средству и разработка функциональных требований 10](#_Toc29275)

[2.1 Теоретический анализ и моделирование предметной области 10](#_Toc12573)

[2.2 Описание функциональности ПС 10](#_Toc4920)

[2.3 Спецификация функциональных требований 11](#_Toc15282)

[2.4 Вывод 11](#_Toc6162)

[3. Проектирование программного средства 12](#_Toc31476)

[3.1 Проектирование базы данных 12](#_Toc15867)

[3.2 Проектирование архитектуры проекта 15](#_Toc30415)

[3.3 Вывод 16](#_Toc14276)

[4. Реализация программного средства 17](#_Toc7942)

[4.1 Обобщённая структура приложения 17](#_Toc26003)

[4.2 Реализация архитектуры MVVM 19](#_Toc27875)

[4.3 Реализация паттерна Repository и UnitOfWork 20](#_Toc5696)

[4.4 Реализация хеширования паролей 21](#_Toc11184)

[4.5 Реализация отправки писем 21](#_Toc2166)

[4.6 Реализация паттерна Command 22](#_Toc26066)

[4.7 Вывод 22](#_Toc20965)

[5. Тестирование работоспособности и анализ полученных результатов 23](#_Toc9131)

[5.1 Вывод 26](#_Toc17677)

[6. Руководство по использованию 27](#_Toc21177)

[6.1 Руководство по использованию администраторам 28](#_Toc29295)

[6.2 Руководство по использованию пользователем 32](#_Toc7822)

[6.3 Вывод 36](#_Toc2079)

[ЗАКЛЮЧЕНИЕ 37](#_Toc4749)

[Cписок использованных источников 38](#_Toc11024)

[ПРИЛОЖЕНИЕ А 39](#_Toc21140)

[ПРИЛОЖЕНИЕ Б 39](#_Toc32654)

[ПРИЛОЖЕНИЕ В 40](#_Toc27135)

[ПРИЛОЖЕНИЕ Г 41](#_Toc707)

**ВВЕДЕНИЕ**

В наши дни многим людям известно, что музыка – это неотъемлемая часть жизни, которая может затрагивать самые глубокие эмоции. Через музыку мы можем выразить себя, поделиться своими чувствами и переживаниями. Многие стремятся найти место, где смогут раскрыть свой вокальный потенциал и найти поддержку в своем развитии.

Однако зачастую процесс поиска подходящего преподавателя и записи на занятия может быть затруднительным и требовать значительных временных затрат. Необходимо изучить множество вариантов, сравнить предлагаемые программы обучения и стоимость. Кроме того, традиционный способ записи на занятия часто предполагает личное присутствие в студии или многочисленные звонки, что создает дополнительные неудобства.

В этом контексте актуальность приобретает разработка специализированного приложения, которое позволит упростить процесс поиска и записи на занятия в студии вокала. Такое приложение даст возможность дистанционно ознакомиться с предложениями студии, сравнить тарифы, а также выбрать подходящего преподавателя для занятий, не выходя из дома

Целью данного курсового проекта является разработка приложения для бронирования занятий в студии вокала, которое позволит пользователям удобно выбирать и записываться на подходящие занятия дистанционно.

Для достижения этой цели необходимо выполнить следующие задачи:

* проанализировать существующие аналоги;
* разработать требования к функциональности приложения;
* спроектировать базу данных хранения информации о студиях, преподавателях, занятиях и пользователях;
* спроектировать архитектуру приложения;
* создать программное средство;
* протестировать работоспособность приложения и проанализировать результаты;
* подготовить руководство по использованию для администраторов и пользователей.

# **Аналитический обзор литературы**

Одним из важных критериев и выбора стратегии разработки приложения является анализ конкурентов. Если создать приложение типичное с другими, это не поспособствует расширению клиентской базы. При помощи анализа конкурентов и реализации востребованных пользователями требований, позволит приложению стать удобнее и в тоже время эффективнее.

## **MEDLEY**

Приложение вокальной студии Medley [1] обеспечивает простую запись на уроки, предоставляет всю необходимую информацию о занятиях, включая расписание и стоимость, позволяет выбрать преподавателя по их профилю и методикам, предоставляет актуальные новости и уведомления, позволяет пользователю читать отзывы и оценивать качество занятий. Также в приложении есть информация об оборудовании и условиях в студии, а также доступ к обучающим материалам и видеоурокам для самостоятельной практики.

Интерфейс приложения вокальной студии обладает интуитивной понятностью и простотой для всех пользователей. Окно просмотра (рисунок 1.1) предоставляет возможность просматривать информацию о занятиях и событиях в студии, а пользователь может легко перемещаться между различными разделами.



Рисунок 1.1 — Окно просмотра.

Вкладка программ обучения (рисунок 1.2) и настройки персональных данных (рисунок 1.3), а также весь интерфейс, выполнены в минималистичном стиле, чтобы обеспечить максимальную наглядность и удобство использования.

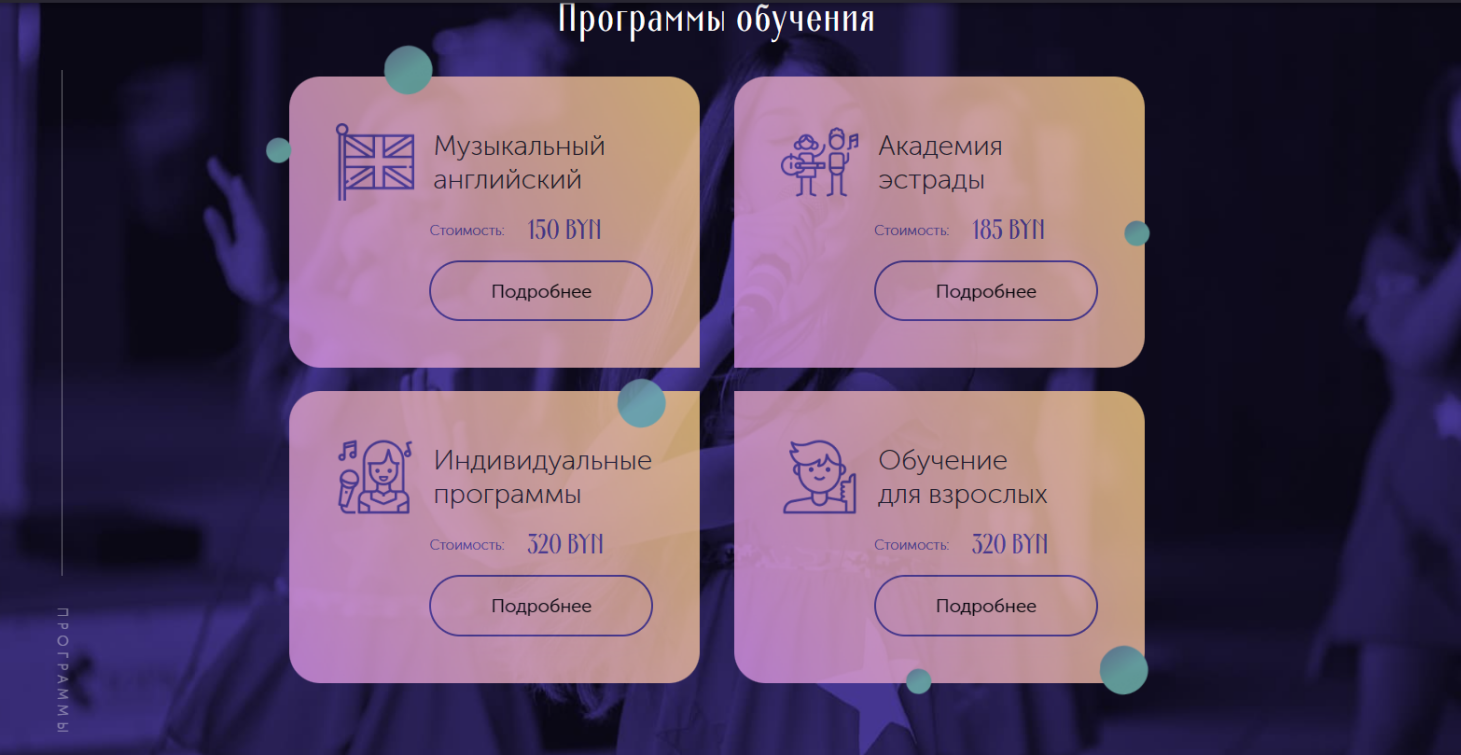


Рисунок 1.2 — Вкладка программ обучения.

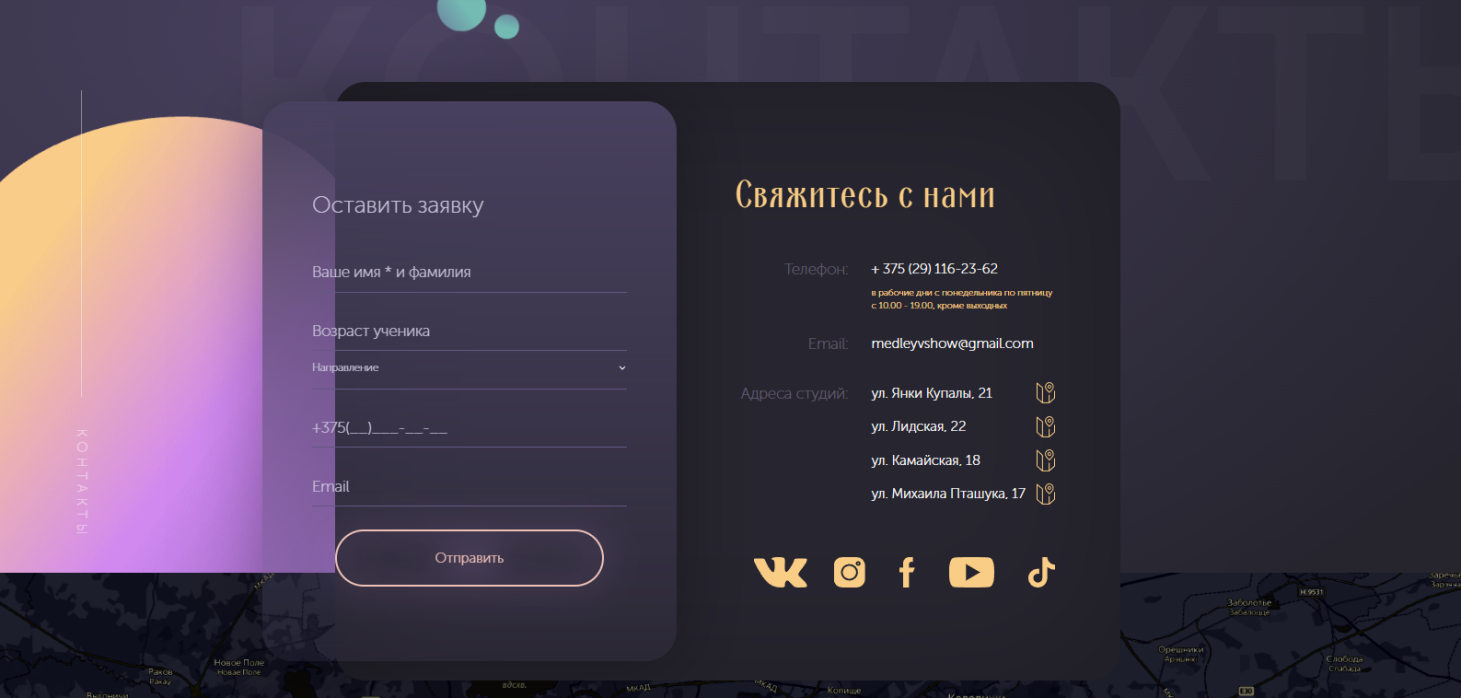


Рисунок 1.3 — Вкладка персональных данных.

Говоря о недостатках сайта, при заполнении заявки мы встречаемся с формой, где нас просят ввести свои данные. Нет возможности выбрать определенного преподавателя, а лишь только направление. Также нет возможности ввести свой опыт в этой сфере.

## **Фа Соль**

Фа Соль [2] - приложение вокальной студии, где пользователи могут не только записываться на уроки, получать информацию о занятиях и преподавателях, но и воспользоваться уникальной методикой Шерил Портер. Эта методика, разработанная опытными преподавателями, способствовала успешному обучению более 5000 учеников. Пользователи также имеют возможность участвовать в мероприятиях, читать отзывы и использовать обширные обучающие материалы для самостоятельного обучения. Это приложение создано для того, чтобы предоставить всесторонний и уникальный вокальный опыт в рамках инновационного подхода к обучению.

Фа Соль предоставляет увлекательный вокальный опыт, отображающей атмосферу мира вокального искусства. Интуитивно-понятный интерфейс обеспечивает легкость в использовании. Приложение обладает многочисленными настройками и опциями для персонализации опыта пользователя. Дизайн, представленный на рисунке 1.5, демонстрирует стиль и привлекательность интерфейса.

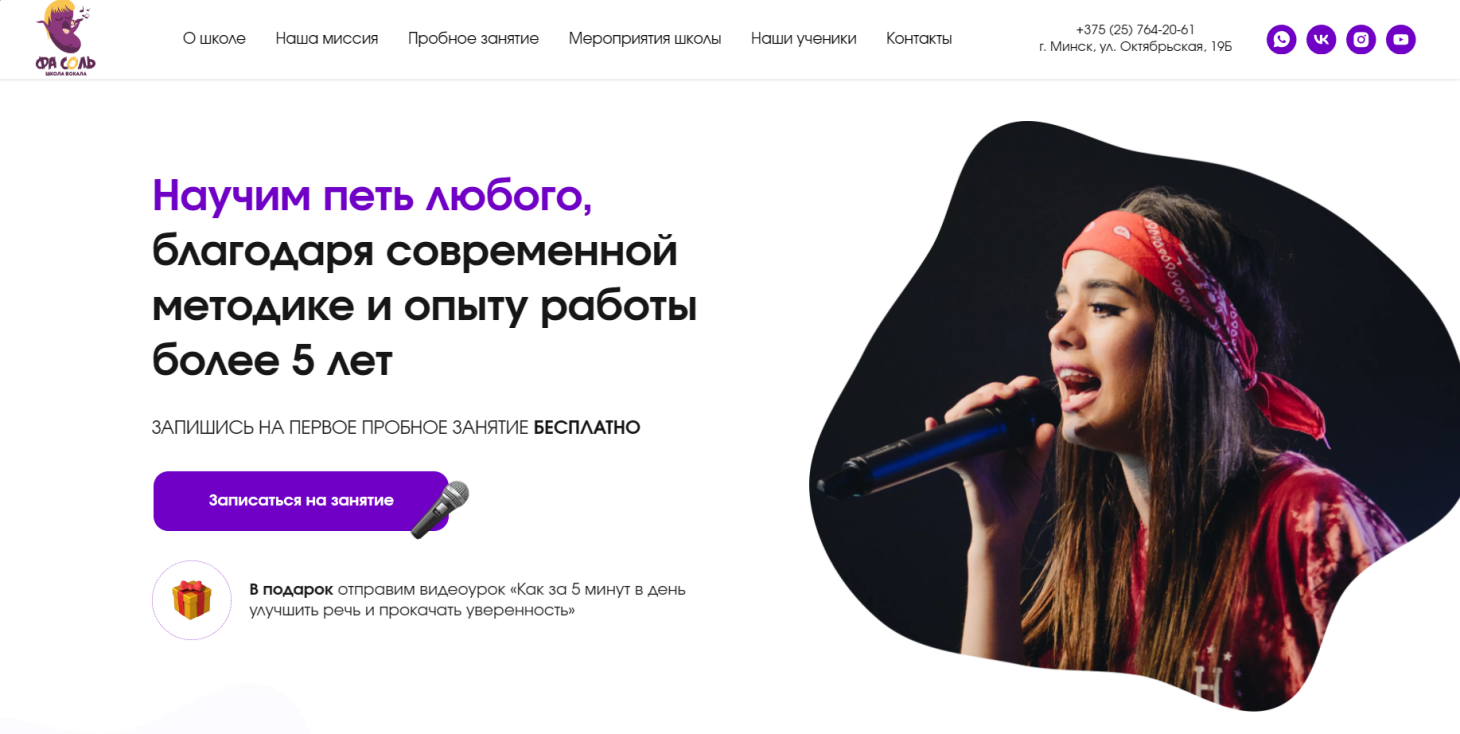


Рисунок 1.5 — Окно просмотра.

При просмотре возможности вокальных объектов, они выводятся в удобном окне (рисунок 1.6), которое может быть размещено в удобном месте.

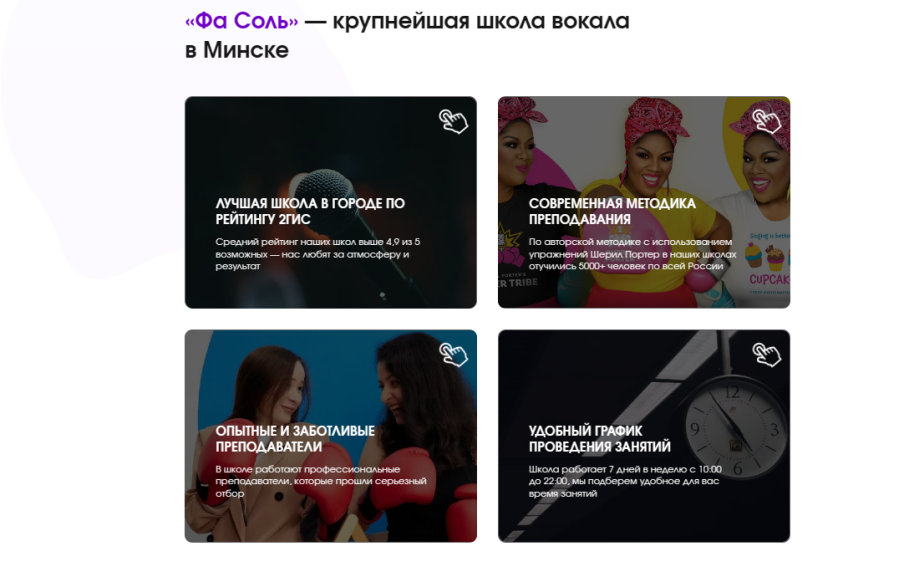


Рисунок 1.6 — Окно просмотра возможностей школы

Краткие характеристики также доступны в углу экрана, обеспечивая легкость сбора данных для бесплатного урока (рисунок 1.7).

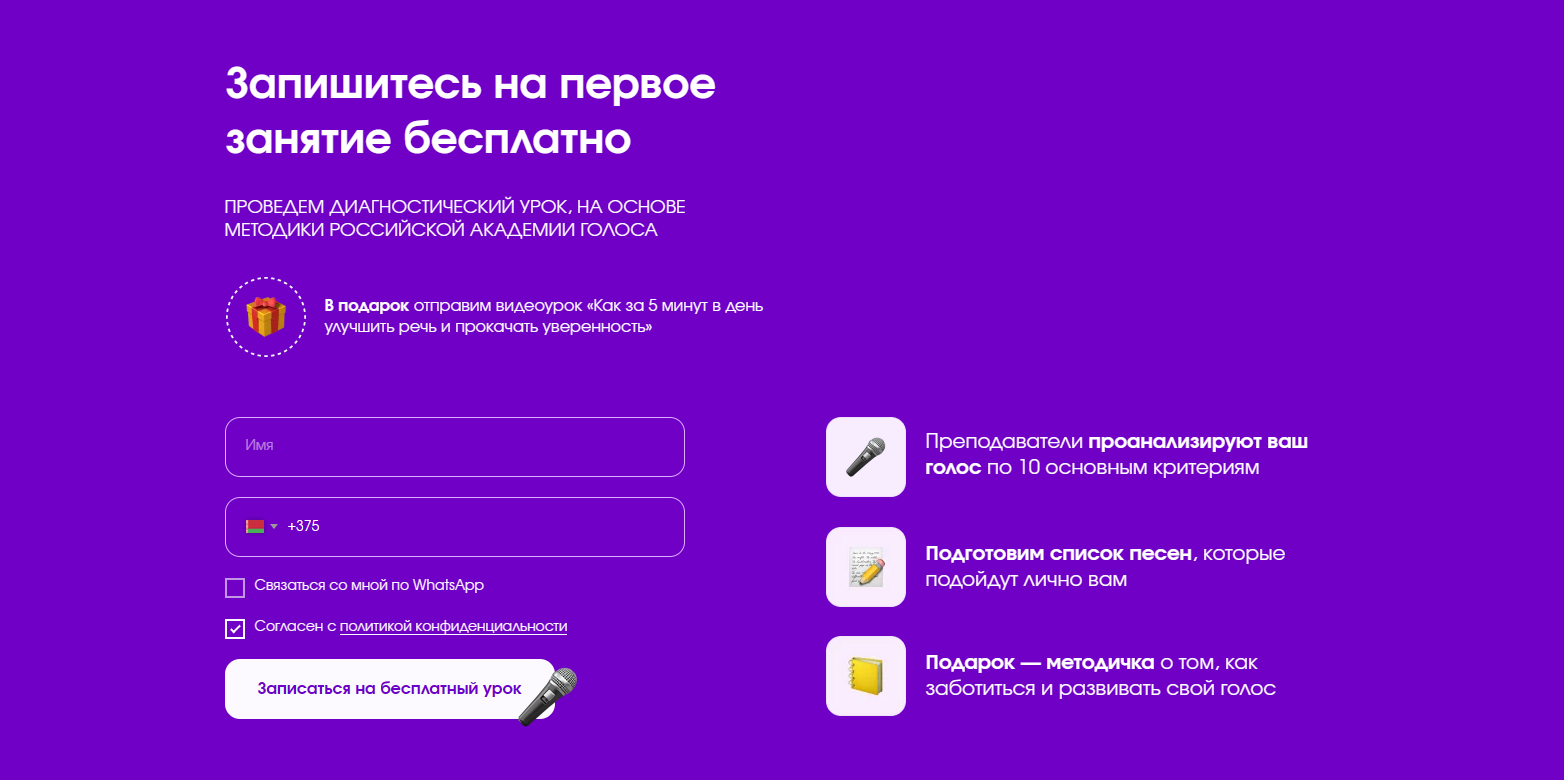


Рисунок 1.7 — Окно сбора данных.

Из недостатков можно выделить слишком простую структуру. Сайт состоит только из одной страницы, которая разделена на информативные блоки, и довольно простым навигационным меню. Не присутствуют отзывы от клиентов, что внушает недоверие и нежелание пользоваться данным ресурсом.

## **Cult Academy**

Приложение Cult Academy [3] предоставляет пользователям возможность погружения в виртуальный мир вокального искусства. Пользователь может свободно перемещаться по виртуальной студии, наблюдая за уроками, встречами с преподавателями, и другими вокальными мероприятиями. Cult Academy также предоставляет подробную информацию о каждом уроке, включая расписание, стоимость и описание содержания. Пользователь имеет возможность контролировать темп своего вокального развития, а также получает доступ к обширным настройкам и опциям для персонализации своего виртуального опыта. Поддержка модификаций позволяет добавлять новые элементы и улучшать визуальное оформление, создавая более увлекательное виртуальное вокальное пространство.

Интерфейс веб-сайта также отличается интуитивной понятностью и простотой использования для всех пользователей. Главное окно (рисунок 1.9) позволяет сразу перейти на этап записи на урок, и выбрать подходящую программу.

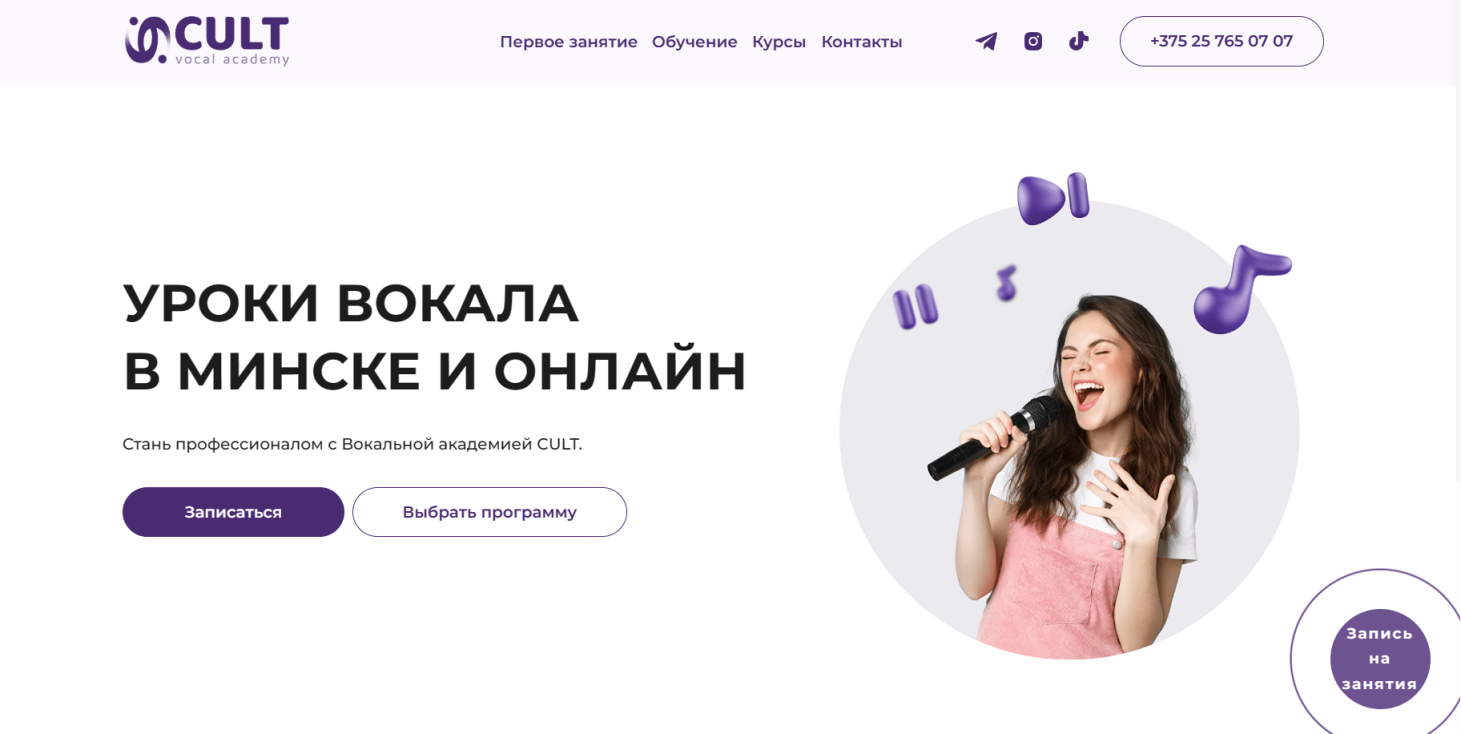


Рисунок 1.9 — Дизайн Cult Academy.

Вкладка преподаватели (рисунок 1.11) выполнена в минималистичном стиле, чтобы обеспечить максимальную ясность и удобство использования для пользователя.

Во вкладке "Тарифы" (рисунок 1.10) представлена подробная информация о доступных курсах и уроках, помогая пользователям выбрать наиболее подходящий вариант.

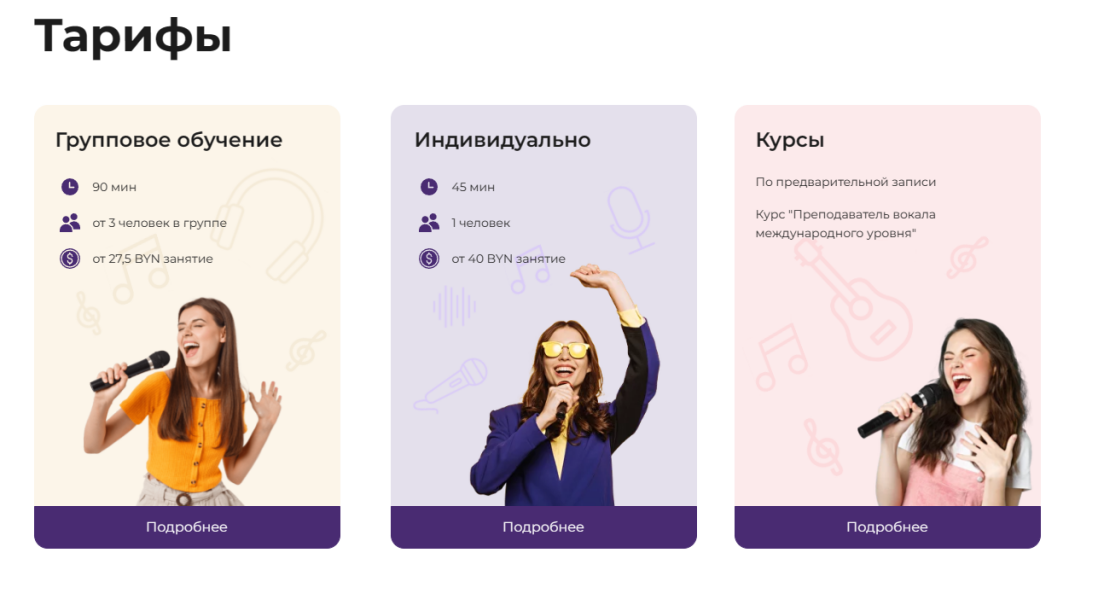


Рисунок 1.10 — Вкладка тарифы.

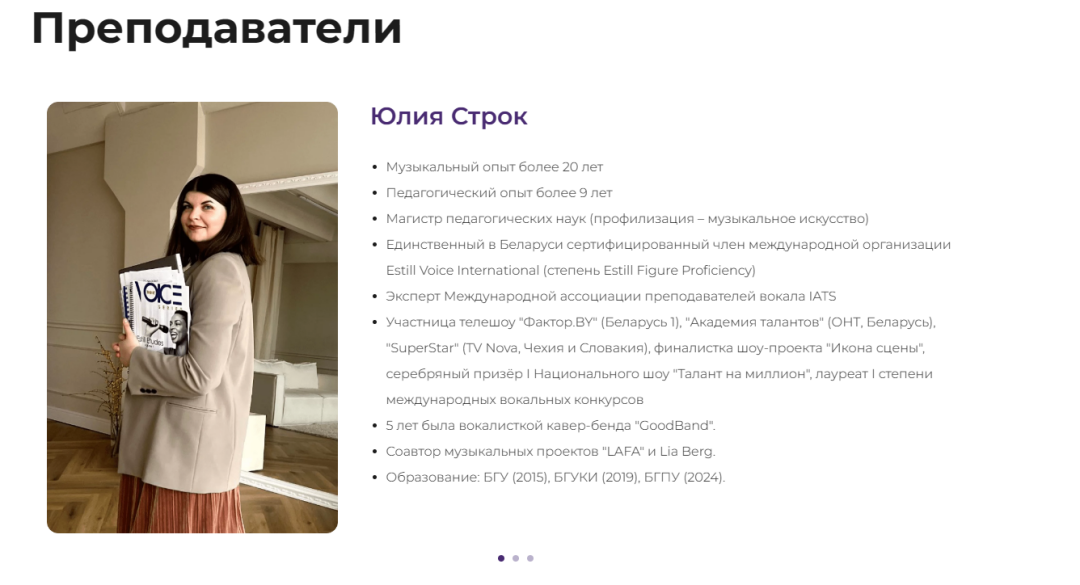


Рисунок 11 — Вкладка «Преподаватели»

Сайт не даёт возможность создать аккаунт для дальнейшей работы с сайтом студии вокала. Всё происходит через звонок администратору.

## **1.4 Вывод**

В ходе проведения аналитического обзора аналогов студий вокала были рассмотрены и проанализированы основные характеристики и подходы ведущих студий, работающих в данной сфере. Исследование показало, что успешные вокальные студии обладают рядом общих черт, таких как высококвалифицированные преподаватели, гибкая система обучения, разнообразие предлагаемых программ, современное техническое оснащение, удобный и понятный интерфейс.

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

## **2.1 Теоретический анализ и моделирование предметной области**

Теоретический анализ предметной области играет важную роль в разработке приложения, поскольку он помогает полностью понять и учесть основные аспекты и требования предметной области.

Приложение “Студия вокала” реализует три основные категории функциональности. Первая категория включает в себя функции, которые доступны только авторизированным пользователям. Вторая категория включает в себя функции, которые доступны только администраторам. Все категории функциональности, а именно функциональность для администратора и авторизированного клиента , будут подробно описаны в следующих подразделах.

Для реализации приложения “Студия вокала” была разработана следующая модель предметной области:

* авторизированный пользователь: это пользователь, который успешно прошел процесс аутентификации и имеет действительную учетную запись;
* администратор: это пользователь с особыми правами, который имеет доступ к различным функциям приложения, связанными с управлением системой;
* преподаватель: это высококвалифицированный преподаватель, обладающий знаниями и опытом в области вокала, который предоставляет услуги для клиентов;
* тариф (абонемент занятий): это услуга, предоставляемая квалифицированным преподавателем, с целью развить способности и приобрести новые навыки в области вокала.

## **2.2 Описание функциональности ПС**

Функционал приложения играет важную роль в обеспечении качественного пользовательского опыта и успешной реализации его основной цели.

Как говорилось ранее весь предоставляемый функционал приложением “Студия вокала” можно разделить на две категории. Первой категорией является функционал, который предоставляет ряд возможностей и инструментов, направленных на поддержку и улучшение навыков вокала для клиентов:

* регистрация и профиль: клиенты могут создать свой учетную запись;
* просмотр и выбор преподавателей: клиенты имеют возможность просматривать профили преподавателей, ознакомиться с их квалификацией, специализацией;
* выбор тарифа: клиенты могут выбрать желаемый тариф, а также просмотреть подробную информацию об желаемом тарифе;
* просмотр и добавление отзывов: клиент всегда может просмотреть добавленные другими пользователями приложения отзывы, а также оставить свой.

Второй категорией является функционал, реализованный для администратора. Данный функционал включает в себя:

* управление преподавателями: администратор имеет возможность добавлять новых преподавателей в систему, редактировать их профили и удалять;
* управление тарифами: администратор может добавлять новые тарифы, а также их редактировать. Эта функция позволяет администратору гибко управлять списком доступных тарифов, обновлять их описание, параметры и стоимость в соответствии с текущими требованиями и потребностями студии.

Для более подробного ознакомления с функциональностью приложения, в Приложении А представлена диаграмма вариантов использования.

## **2.3** **Спецификация функциональных требований**

В программном средстве “Студия вокала” при запуске необходимо реализовать регистрацию или авторизацию для дальнейшего полного использования приложения, но можно воспользоваться режимом Гостя, что ограничивает функционал. Для авторизации параметрами являются уникальное имя и пароль пользователя, которые содержаться в базе данных. Для регистрации необходимо ввести почту, куда будут приходить письма-уведомления, имя и пароль. Введенные данные проходят валидацию: логин должен быть уникальным, почта должна иметь общепринятый формат, пароль иметь длину не менее 6 символов. Если данные прошли валидацию успешно, то они заносятся в базу данных, а пользователь получает полный доступ к приложению.

При входе, пользователя встречает окно с приветствием, через которое возможно перейти к окну просмотру преподавателей, тарифов, отзывов, информации о студии и своей учетной записи. Перейдя в свою учетную запись, пользователь может изменить информацию о себе.

Администратор также может войти через окно логирования, после чего ему предоставляется особый функционал. На странице с информацией о преподавателях он может добавить нового преподавателя нажав по кнопке “добавить”, которая вызывает дополнительное окно для ввода данных о перподавателе: ФИО, описание, должность, опыт, изображение. Также администратор может изменять или удалять преподавателей.

На странице о тарифах администратор может добавить тарифы, кликнув по кнопке “добавить”, что вызывает новое окно-форму для регистрации тарифов. Также может изменить или же вовсе удалить тариф, кликнув по окну “удалить/изменить”

## **2.4 Вывод**

Анализ требований к программному средству и разработка функциональных требований позволили сформировать четкое представление о необходимых характеристиках и функциональности, которые обеспечат успешную реализацию проекта.

# **3. Проектирование программного средства**

## **3.1 Проектирование базы данных**

Данное приложение использует базу данных с названием "MyStudio" и работает на платформе Microsoft SQL Server 2020. Microsoft SQL Server 2020 является реляционной системой управления базами данных (СУБД), разработанной компанией Microsoft. Это мощное и надежное решение для хранения, управления и обработки данных, которое обеспечивает эффективное взаимодействие с базой данных для данной вокальной студии. Логическая схема базы данных представлена на рисунке 3.1.

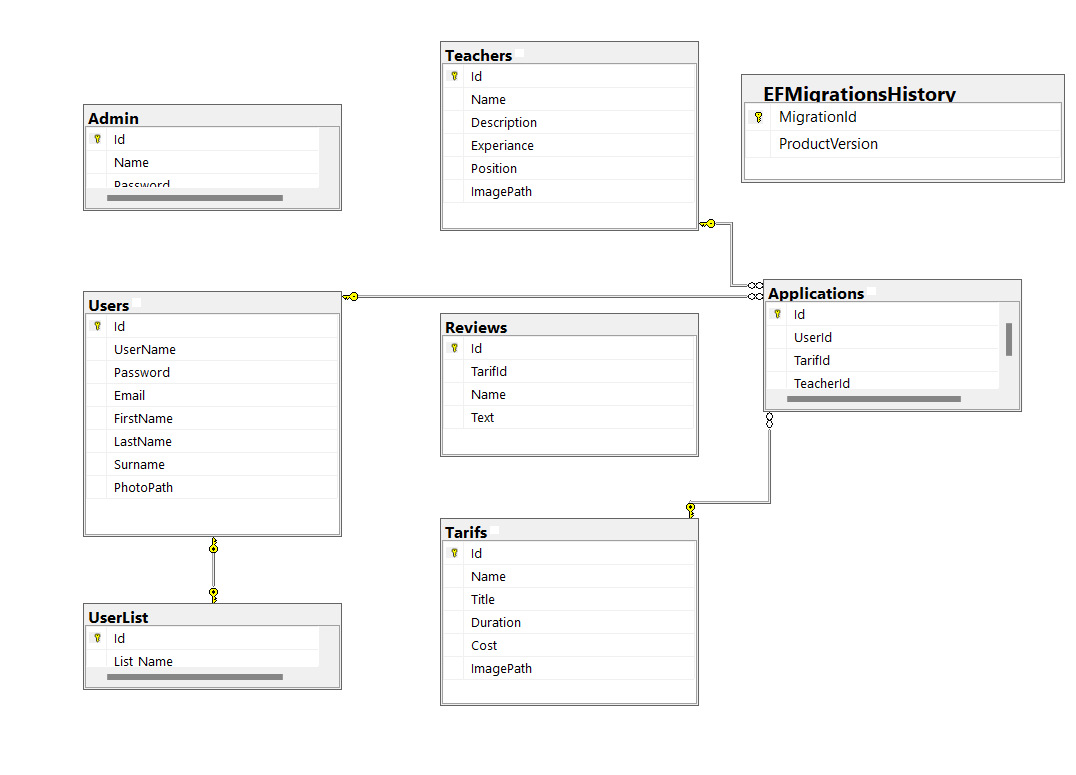


Рисунок 3.1 - Логическая схема данных

Данная база данных содержит в себе 8 таблиц: Users, UserList, Tarifs, Reviews, Teachers, Admin, Applications, EFMigrationHistory.

Рисунок 3.2 иллюстрирует структуру таблицы “Users”, которая содержит информацию о клиенте.

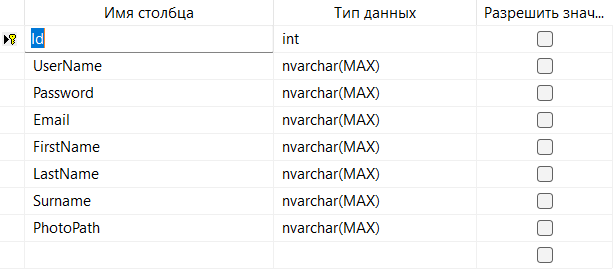


Рисунок 3.2 - Структура таблицы “Users”

Таблица Users состоит из 7 столбцов, где поле “Id” является первичным ключом. “Id” – id клиента, поле UserName хранит имя пользователя, Surname – фамилию, LastName – отчество, Password-пароль, PhotoPath – изображение, а поле Email хранит в себе почту клиента, на которую будут отправляться уведомления.

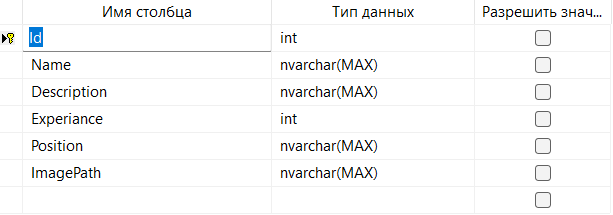


Рисунок 3.3 - Структура таблицы “Teachesrs”

Таблица на рисунке 3.3 иллюстрирует структуру таблицы “Teachers”, которая содержит информацию о преподавателе. В данной таблице поле “Id” является первичным ключом, а также хранит в себе id преподавателя. “Name” содержит ФИО преподавателя, “Description” – информация о преподавателе, “Experiance” – опыт преподавателя, “Position” – содержит квалификацию перподавателя, “imagePath” – фотография преподавателя.

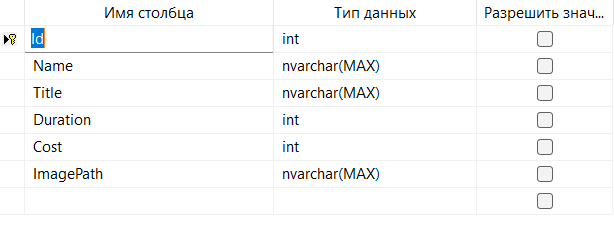


Рисунок 3.4 - Структура таблицы “Tarifs”

Таблица на рисунке 3.4 иллюстрирует структуру таблицы “Tarifs”, которая содержит информацию о тарифе. В данной таблице поле “Id” является первичным ключом, а также хранит в себе id тарифа. “Name” содержит название тарифа, “Title” – описание тарифа, “Duration” – продолжительность урока, “Cost” – содержит стоимость тарифа, “imagePath” – фотография тарифа.

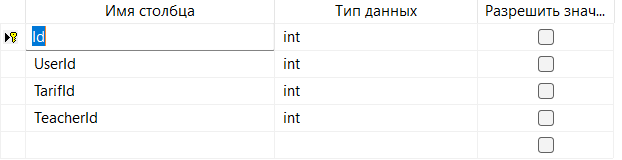


Рисунок 3.5 - Структура таблицы “Application”

Таблица на рисунке 3.5 иллюстрирует структуру таблицы “Application”, которая содержит информацию о заявке. В данной таблице поле “Id” является первичным ключом, а также хранит в себе id заявки. “UserId” содержит id Клиента, “TarifId” – id тарифа, “TeacherId” – id Преподавателя.

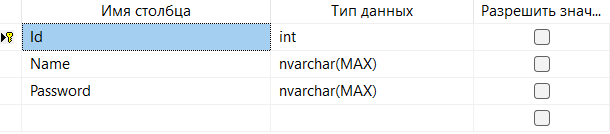


Рисунок 3.6 - Структура таблицы “Admin”

Таблица на рисунке 3.6 иллюстрирует структуру таблицы “Admin”, которая содержит информацию о администраторе. В данной таблице поле “Id” является первичным ключом, а также хранит в себе id администраторa. “Name” содержит имя администратора, “Password” – пароль администратора.

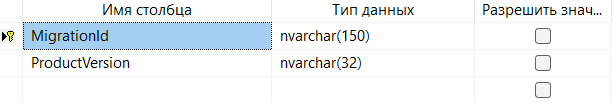


Рисунок 3.7 - Структура таблицы “EFMigrationHistory”

Таблица на рисунке 3.7 иллюстрирует структуру таблицы “EFMigrationHistory”, которая содержит информацию о миграции. В данной таблице поле “Migration” является первичным ключом, а также хранит в себе id миграции. “ProductVersion” содержит версию миграции.



Рисунок 3.8 - Структура таблицы “Reviews”

Таблица на рисунке 3.8 иллюстрирует структуру таблицы “Reviews”, которая содержит информацию об отзывах. В данной таблице поле “Id” является первичным ключом, а также хранит в себе id отзыва. “TarigId” содержит id тарифа, к которму будет прикреплен отзыв,“Name” содержит имя отправителя отзыва, “Text” – текст отзыва.

## **3.2 Проектирование архитектуры проекта**

Данная диаграмма вариантов использования наглядно демонстрирует функциональность приложения "Студия вокала" для различных ролей пользователей, таких как администратор и авторизированный пользователь. Она позволяет легко понять, какие функции доступны каждому из этих пользователей и как они могут взаимодействовать с приложением.

На рисунке 3.9 изображена диаграмма последовательности для авторизации пользователя. Она наглядно иллюстрирует процесс авторизации пользователя и вывода результата. Она демонстрирует последовательность шагов, которые происходят при попытке пользователя войти в систему. (ПРИЛОЖЕНИЕ Б)

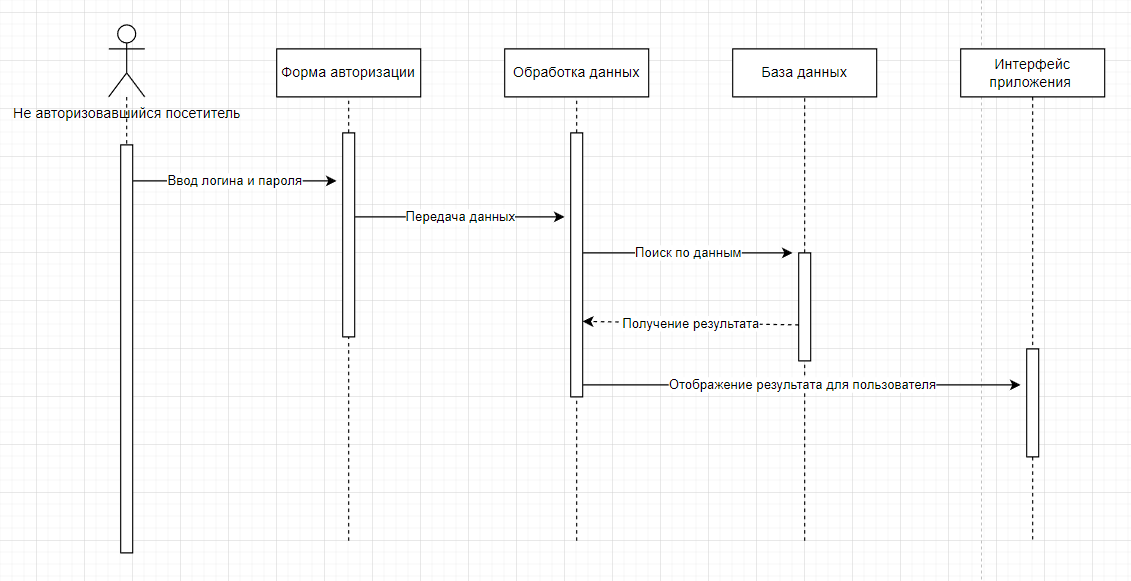


Рисунок 3.9 - Диаграмма последовательности для авторизации пользователя

На рисунке 3.11 изображена диаграмма компонентов приложения.

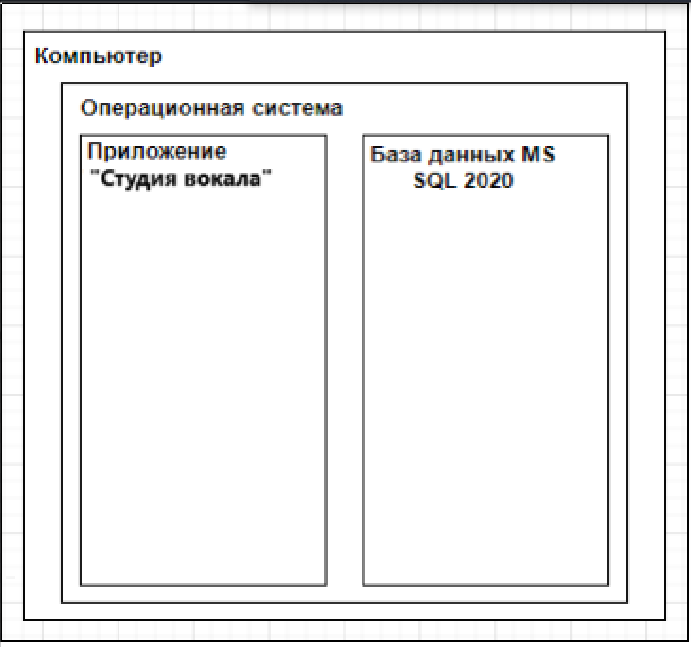


Рисунок 3.11 - Диаграмма компонентов приложения

Данная диаграмма компонентов наглядно демонстрирует компоненты, которые будут использованы для дальнейшей работы и функционировании приложения. Она предоставляет обзор основных компонентов, которые взаимодействуют между собой для обеспечения работоспособности приложения.

## **3.3 Вывод**

В процессе проектирования программного средства были разработаны ключевые архитектурные решения и определены основные компоненты системы, что позволило создать структурированный и последовательный план разработки.

# **4. Реализация программного средства**

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

## **4.1 Обобщённая структура приложения**

Для создания высококачественного приложения необходимо построить диаграмму классов и определить обобщенную структуру. Этот этап разработки играет важную роль, так как позволяет получить ясное представление о компонентах приложения и их взаимосвязи.

Диаграммой классов представлена на рисунке 4.1.

Для детального рассмотрения диаграмма классов также представлена в ПРИЛОЖЕНИИ В.

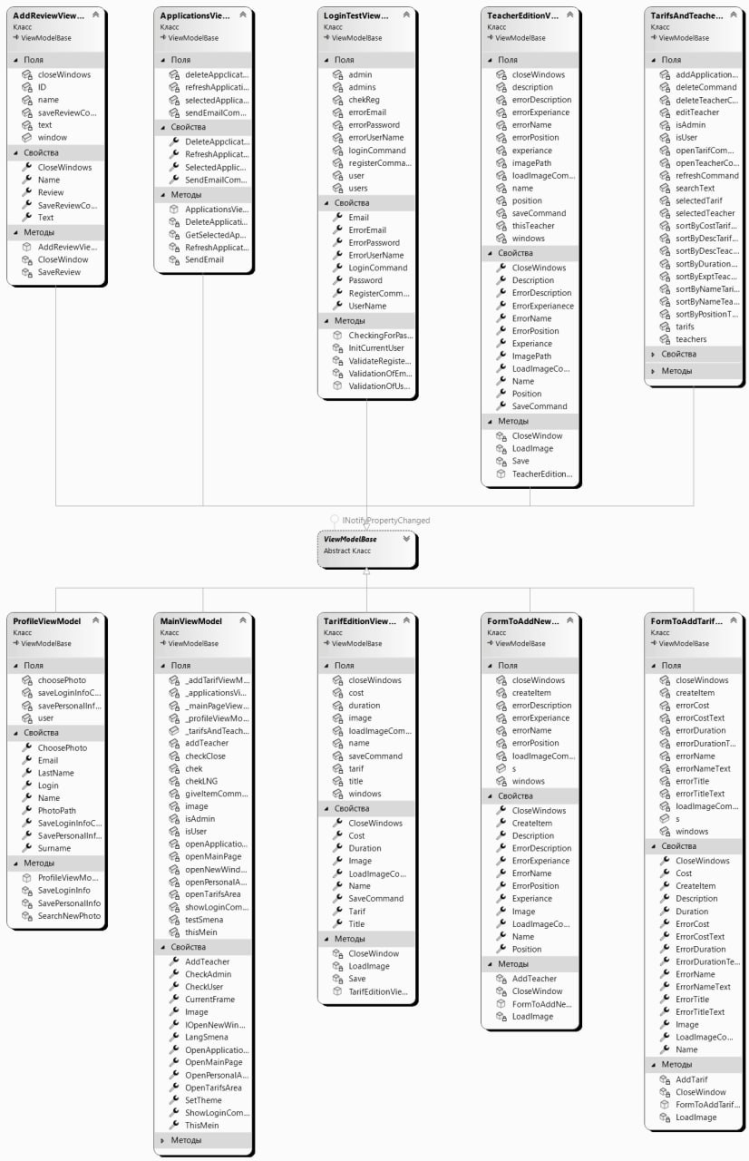


Рисунок 4.1 - Диаграмма классов

Данная диаграмма классов наглядно демонстрирует обобщенную структуру приложения. Она представляет собой визуальное отображение классов, их свойств и взаимосвязей, которые существуют внутри приложения.

Для более чёткого представления и детализации структуры пользовательского интерфейса в приложении необходимо составить таблицу с названиями XAML-файлами и их кратким содержанием. Данные файлы представлены в таблице 4.1.

Таблица 4.1 – Описание XAML-файлов

|  |  |
| --- | --- |
| Имя файла | Содержание |
| lang-ENG.xaml | Словарь ресурсов для смены языка в приложении на английский. |
| lang-RU.xaml | Словарь ресурсов для смены языка в приложении на русский. |
| dark.xaml | Словарь ресурсов для смены темы в приложении на темную. |
| light.xaml | Словарь ресурсов для смены темы в приложении на светлую. |
| LoginView.xaml | Страница с формой авторизации\ регистрации нового аккаунта. |
| ProfilePage.xaml | Страница профиля текущего клиента с возможностью изменения личных данных. |
| AddReview.xaml | Форма для создания отзыва. |
| MainPage.xaml | Страница с ознакомительной информацией о вольной студии. |
| ApplicationView.xaml | Страница выбранных услуг (заявки). |
| CurrentTarifView.xaml | Страница выбранного тарифа. |
| CurrentTeacherView.xaml | Страница выбранного преподавателя. |
| TarifsAndTeacher.xaml | Страница со список всех существующих преподавателей и тарифов в приложении. |
| ReviewsPage.xaml | Страница со всеми отзывами оставленных от клиентов. |
| MainView.xaml | Страница с навигационной панелью. |
| Generic.xaml | Словарь ресурсов с набором стилей для элементов приложения. |
| EdittarifView.xaml | Страница редактирования тарифа. |
| TeacherEditionView.xaml | Страница редактирования преподавателя. |
| FormToAddNewTarif.xaml | Страница добавления тарифа. |
| FormToAddNewTeacher.xaml | Страница добавления преподавателя. |

Каждый из этих файлов является частью приложения и содержит разметку для соответствующей страницы или окна. Они играют важную роль в определении структуры и внешнего вида пользовательского интерфейса приложения.

## **4.2 Реализация архитектуры MVVM**

Архитектура MVVM (Model-View-ViewModel) является важным и широко применяемым подходом при разработке приложений на платформе WPF.

Для реализации паттерна MVVM файлы программы были распределены по соответствующим директориям и реализовали соответствующие функции.

При использовании подхода DataBase First в разработке приложения, классы-модели были автоматически сгенерированы самим приложением на основе существующей базы данных. Разберём все модели:

* Admin.cs – содержит информацию об администраторе;
* Application.cs – содержит необходимую информацию для оформления заявки;
* Review.cs – содержит информацию об отзыве;
* Tarif.cs – содержит информацию о тарифе;
* Teacher.cs – содержит информацию о преподавателе;
* UserList.cs – содержит информацию о клиентах;
* Users.cs – содержит информацию о клиенте.

Для реализации архитектуры MVVM (Model-View-ViewModel) необходимо создать классы, которые будут выступать в роли ViewModel.

ViewModel представляет прослойку между моделью (Model) и представлением (View). Его основная задача - предоставить данные и логику, необходимую для отображения и взаимодействия с моделью в пользовательском интерфейсе.

Данные классы находятся в папке MVVM, рассмотрим их поподробнее:

* AddReviewViewModel.cs – содержит необходимые команды для добавления отзыва;
* ApplicationsViewModel.cs – содержит необходимые команды для оформления заявки;
* CurrentTarifViewModel.cs – содержит необходимые команды для отображения и работы с конкретным тарифом;
* CurrentTeacherViewModel.cs – содержит необходимые команды для отображения и работы с конкретным преподавателем;
* FormToAddNewTeacherViewModel.cs – содержит необходимые команды для добавления преподавателя;
* FormToAddTarifViewModel.cs – содержит необходимые команды для добавления тарифов;
* LoginTestViewModel.cs – содержит команды необходимые для авторизации пользователя;
* MainViewModel.cs – содержит команды необходимые для навигационного меню, а также команды для управления стилевым оформлением приложения, а также для смены языка в приложении;
* ProfileViewModel.cs – содержит команды необходимые для изменения личной информации пользователя;
* TarifEditionViewModel.cs – содержит команды для изменения и удаления тарифа;
* TarifsAndTeachersViewModel.cs – содержит команды, необходимые для отображения и управления тарифами и преподавателями;
* TeacherEditionViewModel.cs – содержит команды для изменения и удаления тарифа;
* ViewModelBase.cs – содержит основные функции и команды для работы с данными в приложении.

Для реализации архитектуры MVVM (Model-View-ViewModel) необходимо создать страницы, которые будут выступать в роли представления (View).

Представление (View) отображает пользовательский интерфейс и взаимодействует с пользователем. В контексте MVVM, представление является пассивным и не содержит бизнес-логики. Его основная задача - отображать данные, полученные из ViewModel, и передавать пользовательские действия обратно в ViewModel для обработки.

Данные страницы были описаны выше в таблице 4.1.

## **4.3 Реализация паттерна Repository и UnitOfWork**

Реализация паттерна Repository и UnitOfWork является важной в разработке приложений, особенно при работе с базами данных. Эти паттерны помогают организовать доступ к данным и обеспечить эффективное управление транзакциями.

Паттерн Repository реализуется для всех классов-моделей в приложении, чтобы абстрагироваться от работы с различными видами баз данных.

Рассмотрим текущие классы-репозитории в приложении:

* UsersRepository.cs – данный класс реализует методы для работы с моделью Users;
* ApplicationRepository.cs – данный класс реализует методы для работы с моделью Applications;
* TarifsRepository.cs – данный класс реализует методы для работы с моделью Tarif;
* TeacherRepository.cs – данный класс реализует методы для работы с моделью Teacher;
* ReviewRepository - данный класс реализует методы для работы с моделью Review.

Далее рассмотрим реализацию паттерна UnitOfWork в приложении "Студия вокала". Данный паттерн тесно взаимодействует с классами-репозиториями и служит общей точкой для доступа к репозиториям, а также объединяет все изменения в единую транзакцию.

Класс реализующий паттерн UnitOfWork называется “UnitOfWork” и содержит все вышеописанные классы-репозитории.

4.4 Реализация хеширования паролей

Хранение паролей в базе данных является небезопасным, поскольку злоумышленники могут получить несанкционированный доступ у учетным записям пользователей. Поэтому было принято решение использовать алгоритм SaltedHash для хеширования паролей.

Salted hash – это алгоритм хеширования паролей, который добавляет к паролю случайно сгенерированную уникальную «соль» (salt) перед хешированием. Соль является дополнительным значением, добавляемым к паролю, чтобы защитить его от атак перебора.

Класс SaltedHash реализует данный алгоритм. Структура класса представлена в ПРИЛОЖЕНИИ Г.

4.5 Реализация отправки писем

ApplicationsViewModel.cs демонстрирует, как можно эффективно использовать возможности .NET для взаимодействия с пользователями посредством электронной почты, обеспечивая высокую степень автоматизации и безопасности в приложении. Листинг класса приведен в ПРИЛОЖЕНИИ Г

Основное преимущество этого метода заключается в его способности быстро и безопасно отправлять уведомления, что существенно улучшает пользовательский опыт и автоматизирует процесс коммуникации. Структура метода приведена ниже в листинге 4.2.

|  |
| --- |
| private void SendEmail()  {  try  {  SmtpClient client = new SmtpClient("smtp.mail.ru", 587);  client.EnableSsl = true;  client.Timeout = 20000;  client.DeliveryMethod = SmtpDeliveryMethod.Network;  client.UseDefaultCredentials = false;  client.Credentials = new NetworkCredential("trikoza\_tatjana@mail.ru", "17DtZq8DCuun0FzidzV9");  MailMessage mailMessage = new MailMessage();  mailMessage.From = new MailAddress("trikoza\_tatjana@mail.ru");  mailMessage.To.Add(CurrentUser.Email); // Получатель  mailMessage.Subject = "Студия Вокала 'Chocolate voice' ";  var selectedApplication = GetSelectedApplication();  string messageBody = "Ваша заявка успешно рассмотрена!\nBы записаны на:\n";  messageBody += $"Тариф: {selectedApplication.Tarif.Name}\n";  messageBody += $"Преподаватель: {selectedApplication.Teacher.Name}";    mailMessage.Body = messageBody;    client.Send(mailMessage);  MessageBox.Show("Вы записаны на тариф!\nУведомление о занятии придет вам на почту");  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при отправке письма: {ex.Message}");  }  } |

Листинг 4.2 – Структура метода SendEmail.

## **4.6 Реализация паттерна Command**

Паттерн Command является важным в разработке приложений, поскольку он обеспечивает отделение логики действия от элементов пользовательского интерфейса и позволяет управлять выполнением операций в ответ на пользовательские действия.

Класс реализующий паттерн Command называется “DelegateCommand” и содержит следующие методы:

1. CanExecute – данный метод проверяет, может ли данная команда выполниться прямо сейчас.
2. Execute – данный метод определяет метод который будет вызываться при срабатывании команды.

## **4.7 Вывод**

В процессе реализации программного средства была создана обобщённая структура приложения, внедрена архитектура MVVM, а также реализованы паттерны Repository, UnitOfWork и Command. Эти шаги обеспечили модульность, тестируемость и масштабируемость системы.

# **5. Тестирование работоспособности и анализ полученных результатов**

Данное программное средство тестировалось вручную, были выполнены все технические моменты, предусмотренные реализацией, а также были предприняты попытки нарушить работу приложения.

В курсовом проекте задействуется обработка исключений, таким образом, что пользователь будет уведомлен о неудачном выполнении операции. Присутствуют различные всплывающие окна, выводящие сообщение об ошибке, также присутствует, а также различные изображения.

В момент регистрации возможна ситуация, когда пользователь вводит уже существующее имя или же просто некорректно вводит электронную почту, либо пароль. На рисунке 5.1, 5.2, 5.3 продемонстрирована обработка данных исключений.

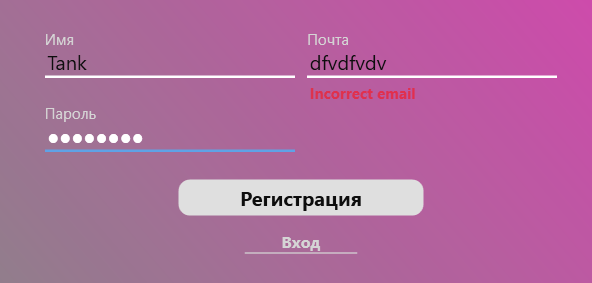


Рисунок 5.1 - Неверно введена почта

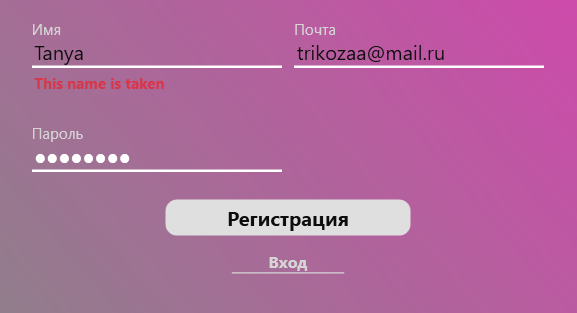


Рисунок 5.2 - Имя уже зарезервировано.

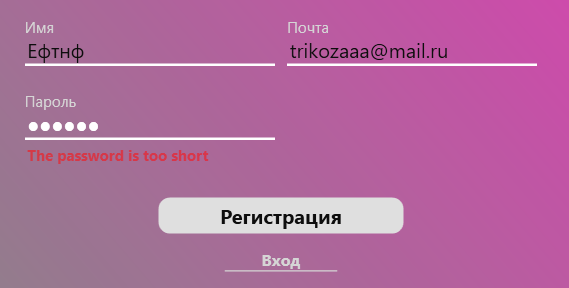


Рисунок 5.3 - Неверно введен пароль.

При регистрации нового пользователя его данные заносятся в таблицу User в базе данных.

Когда пользователь вводит свои учетные данные и нажимает кнопку "Войти" или аналогичное действие, приложение проверяет введенные данные на соответствие с данными, хранящимися в базе данных. Если соответствующая запись не найдена, возникает ситуация ошибки авторизации.

В такой ситуации приложение отражает неверно введенные данные.

Возможная ситуация продемонстрирована на рисунке 5.4.

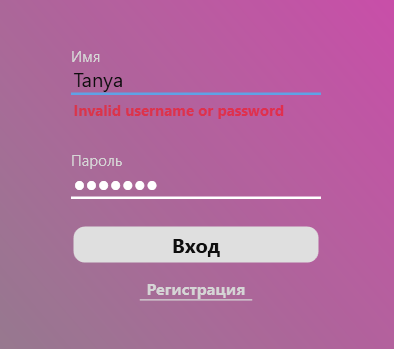
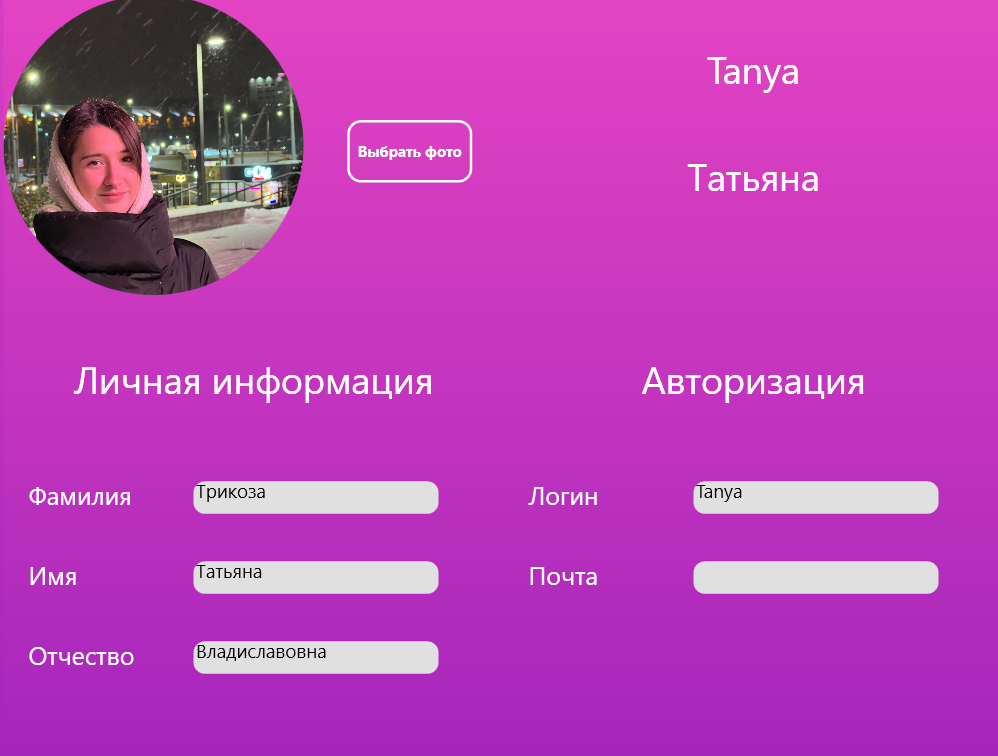


Рисунок 5.4 - Неверно введены данные.

В случае, когда пользователь пытается оформить заявку и не заполнил некоторые обязательные поля (почта), приложение обнаружит эту неполадку и выведет соответствующее сообщение.

Ситуация изображена на рисунке 5.5



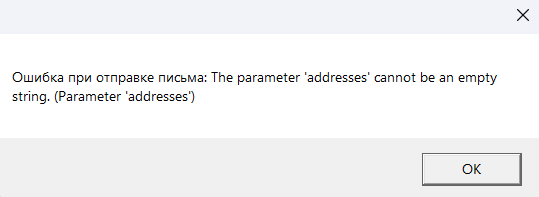


Рисунок 5.5 - При оформлении заявки, пользователь не заполнил поля в профиле

Подобные случаи могут случиться и у администратора, поэтому рассмотрим несколько из них.

Когда администратор добавляет новый тариф он вводит данные в поля формы тарифа и эти данные не проходят проверку на валидность, приложение может визуально отобразить ошибку, добавляя красную границу вокруг неверно введенных данных, изображено на рисунке 5.6.



Рисунок 5.6 - Введённая информация в поля карточки тарифа не валидна.

Та же ситуация возникает при изменении тарифа или преподавателя, изображено на рисунке 5.7

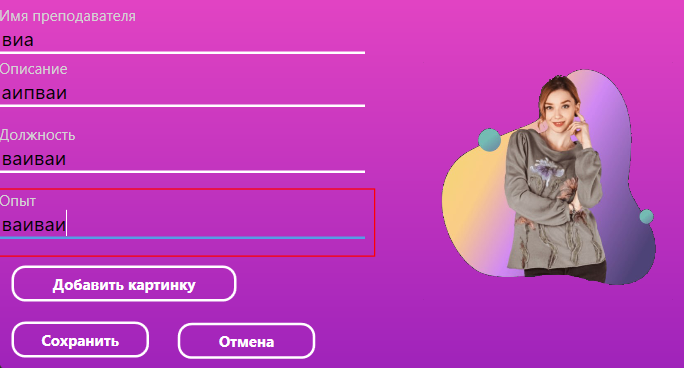


Рисунок 5.7 - При изменении преподавателя данные не проходят валидность.

## **5.1 Вывод**

Тестирование работоспособности и анализ полученных результатов позволили убедиться в правильной работе программного средства, выявить и устранить ошибки, а также подтвердить соответствие системы требованиям и ожиданиям пользователей.

# **6. Руководство по использованию**

Окно регистрации содержит поля для ввода учетных данных, таких как электронная имя, почта и пароль. Пользователь может использовать эти поля для авторизации, если у него уже есть учетная запись. После ввода данных пользователь нажимает на кнопку "Войти".

Форма регистрации представлена на рисунке 6.1.

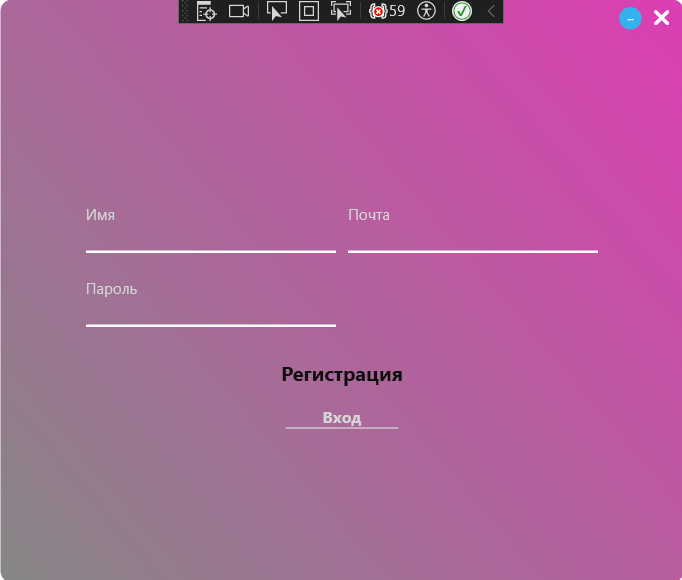


Рисунок 6.1 - Окно регистрации.

Если у пользователя имеется личный аккаунт в приложении, ему всегда доступна авторизация.

При выборе опции авторизации, пользователю предлагается ввести свое имя и пароль в соответствующие поля. Пароль вводится в отдельное поле в зашифрованном виде для обеспечения безопасности.

При вводе данных пользователь может проверить являются ли они корректными, а именно : ничего не будет отображаться в случае корректности; будет отображаться текст красного цвета, что обозначает противоположный результат.

После заполнения полей логина и пароля пользователь нажимает на кнопку "Войти", чтобы произвести авторизацию в приложении.

Форма авторизации представлена на рисунке 6.2.

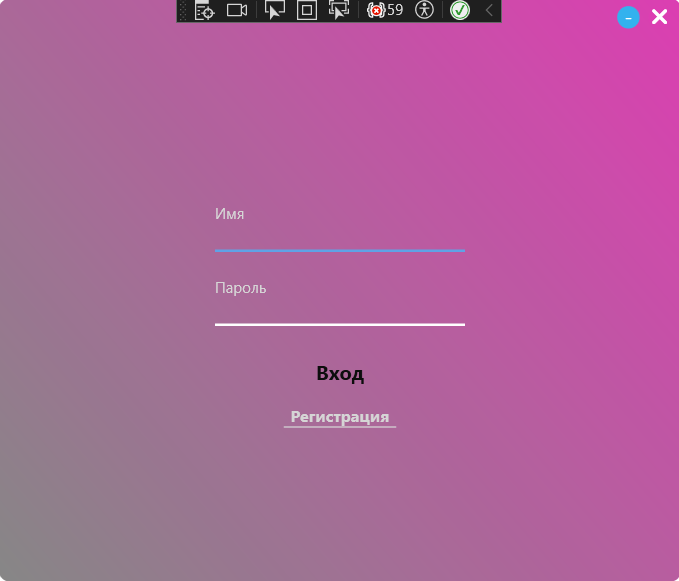


Рисунок 6.2 - Окно авторизации

Данные формы общие как для администратора так и для авторизированного пользователя.

## **6.1 Руководство по использованию администраторам**

После входа в приложения администратора встречает приветствующая страница, а также навигационное меню, расположенное вертикальном положении.

Данная страница изображена на рисунке 6.3.

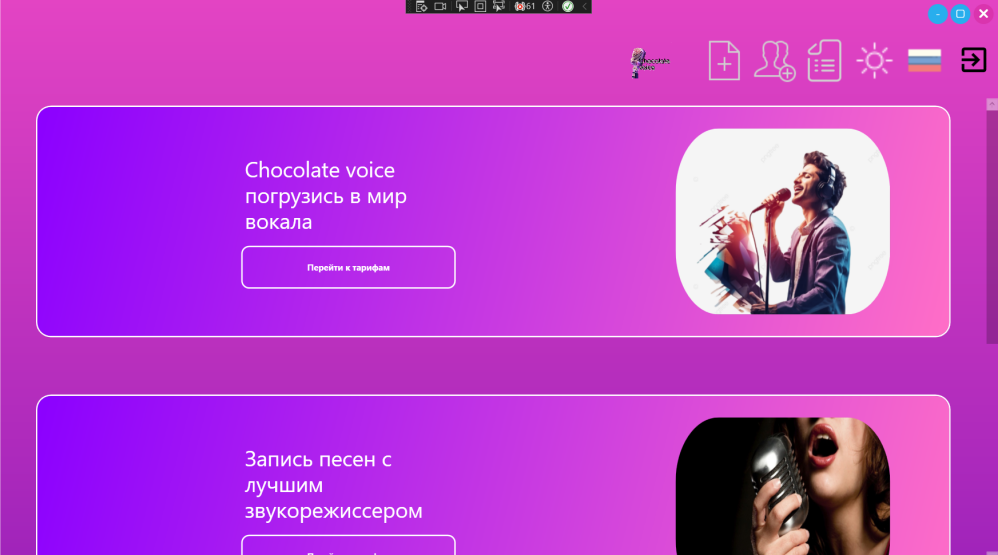


Рисунок 6.3 - Приветствующая страница

Нажав на кнопку “Документ” мы переходим к общему перечню тарифов и преподавателей, которых мы можем отсортировать по возможным перечням (опыт, должность, имя). Так же для администратора имеется дополнительная кнопка “добавить”, с помощью которого можно добавить нового преподавателя в базу данных. Представлено на рисунке 6.4.

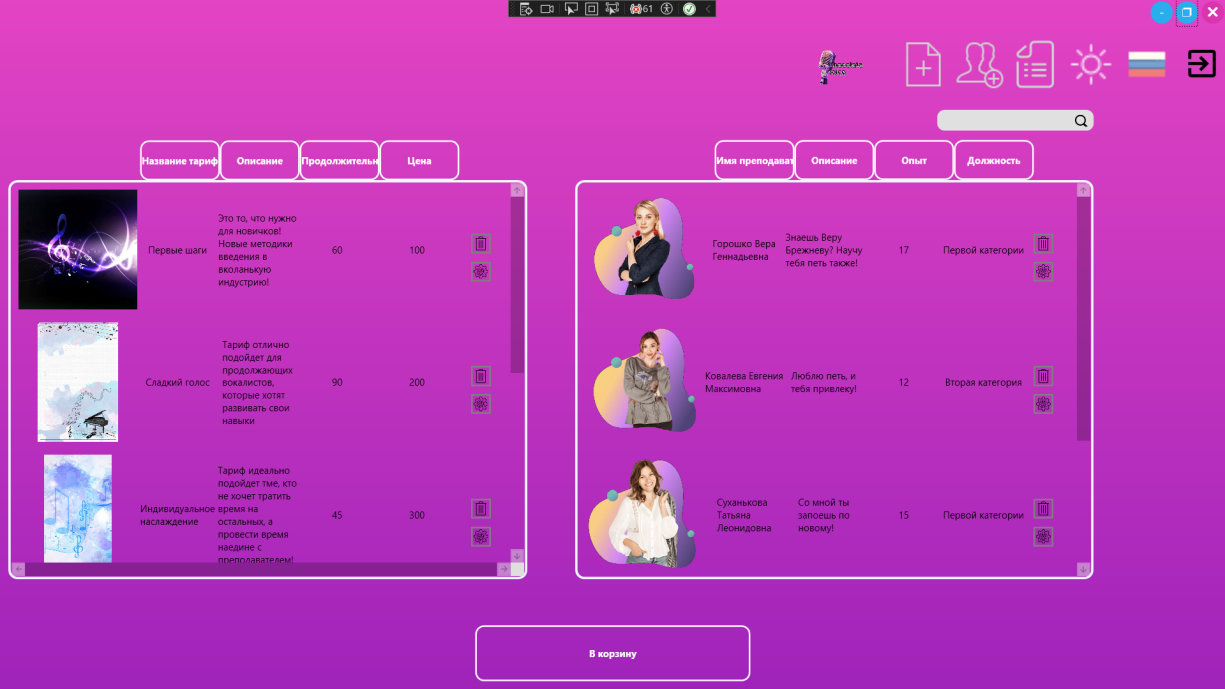


Рисунок 6.4 - Страница со списком преподавателей

При нажатии на кнопку “добавить”, администратор попадает в новое окно, с помощью которого может добавить преподавателя в базу данных. Данная форма представлена на рисунке 6.5.

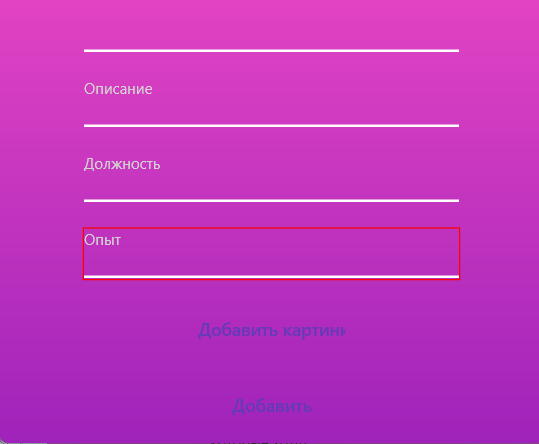


Рисунок 6.5 - Страница добавления преподавателя

При клике на кнопку “настроить”, администратору открывается окно для редактирования. Также присутствует кнопка “удалить”, которая удалит текущего преподавателя из базы данных.

Окно редактирования преподавателя представлено на рисунке 6.6.

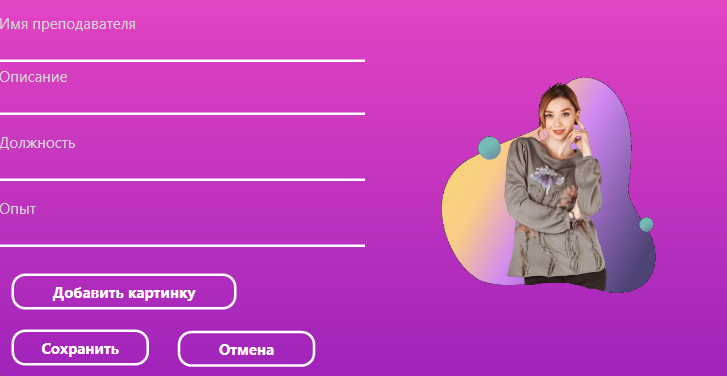


Рисунок 6.6 - Страница редактирования преподавателя

Далее нажав на кнопку “Документ”, мы также попадаем на страницу, где выдается перечень преподавателей и тарифов, которых так же можно сортировать и добавлять новые в базу данных, представлено на рисунке 6.7.

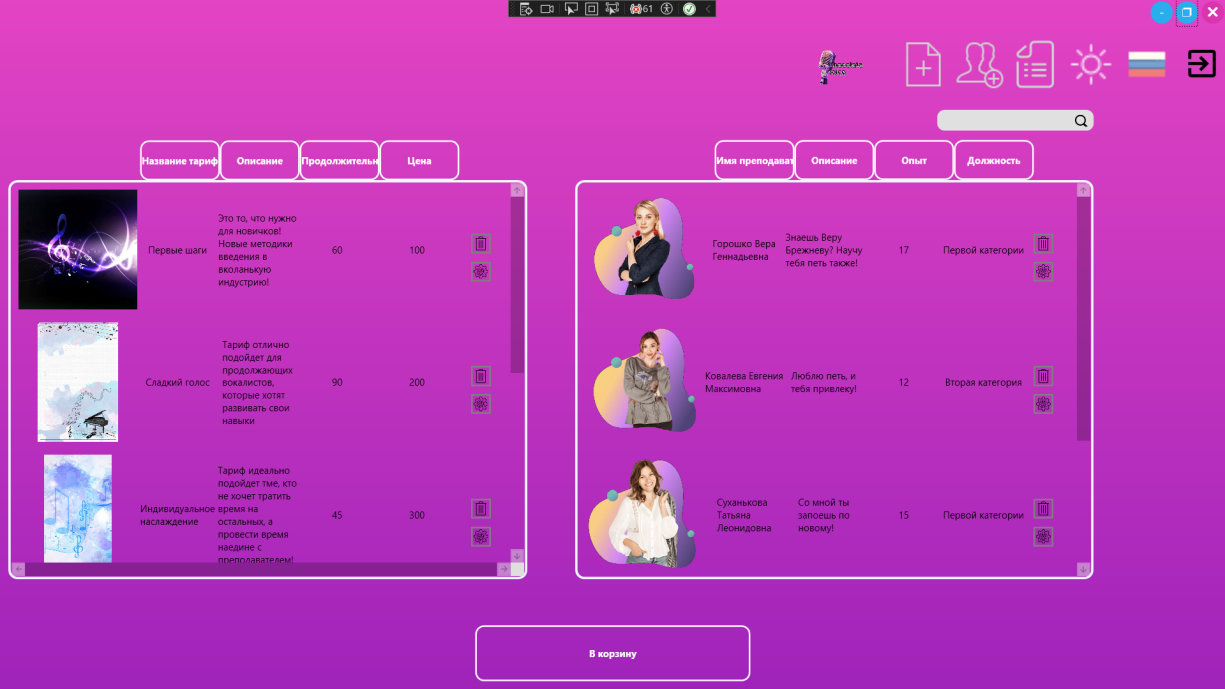


Рисунок 6.7 - Страница с перечнем тарифов и преподавателей

Нажав на кнопку “добавить”, администратор переходит в новое окно, в которой находится форма для добавления нового тарифа. В данной форме необходимо заполнить обязательные поля, для дальнейшего добавлении в базу данных, представлено на рисунке 6.8.

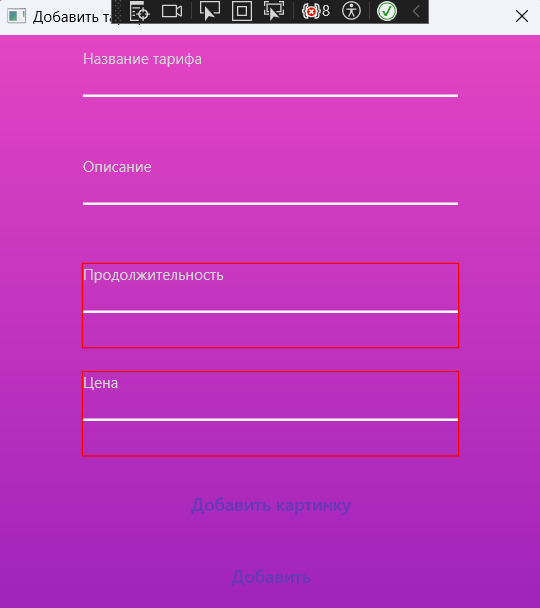


Рисунок 6.8 - Форма для добавления тарифа

Для изменение текущего тарифа, администратор должен нажать на на кнопку “настроить”. Администратору вновь открывается форма для редактирования. Так же на форме доступна кнопка “удалить”, которая удалит тариф из базы данных.

Данная форма представлена на рисунке 6.9.

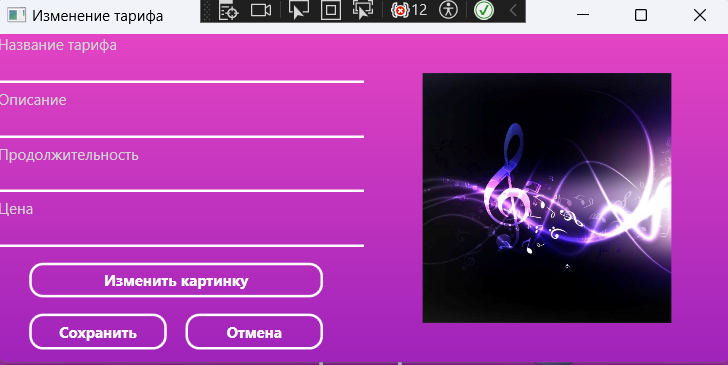


Рисунок 6.9 - Форма для изменения тарифа

## **6.2 Руководство по использованию пользователем**

После успешной авторизации, пользователя встречает приветствующая страница с навигационным меню. Данное меню обеспечивает пользователю быстрый доступ к различным функциям приложения, позволяя ему легко перемещаться по различным страницам, что представлено на рисунке 6.11.

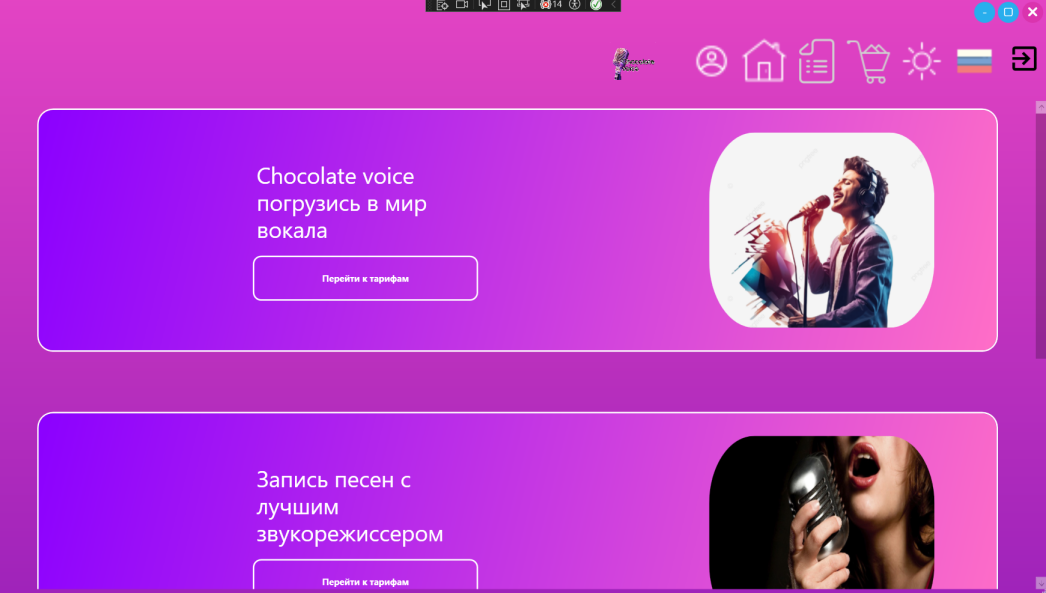


Рисунок 6.10 - Приветствующая страница

Если пользователь решит нажать на изображение аккаунта, то ему откроется новая страница с возможностью изменения личных данных, а также изображения профиля.

Данная страница представлена на рисунке 6.12.

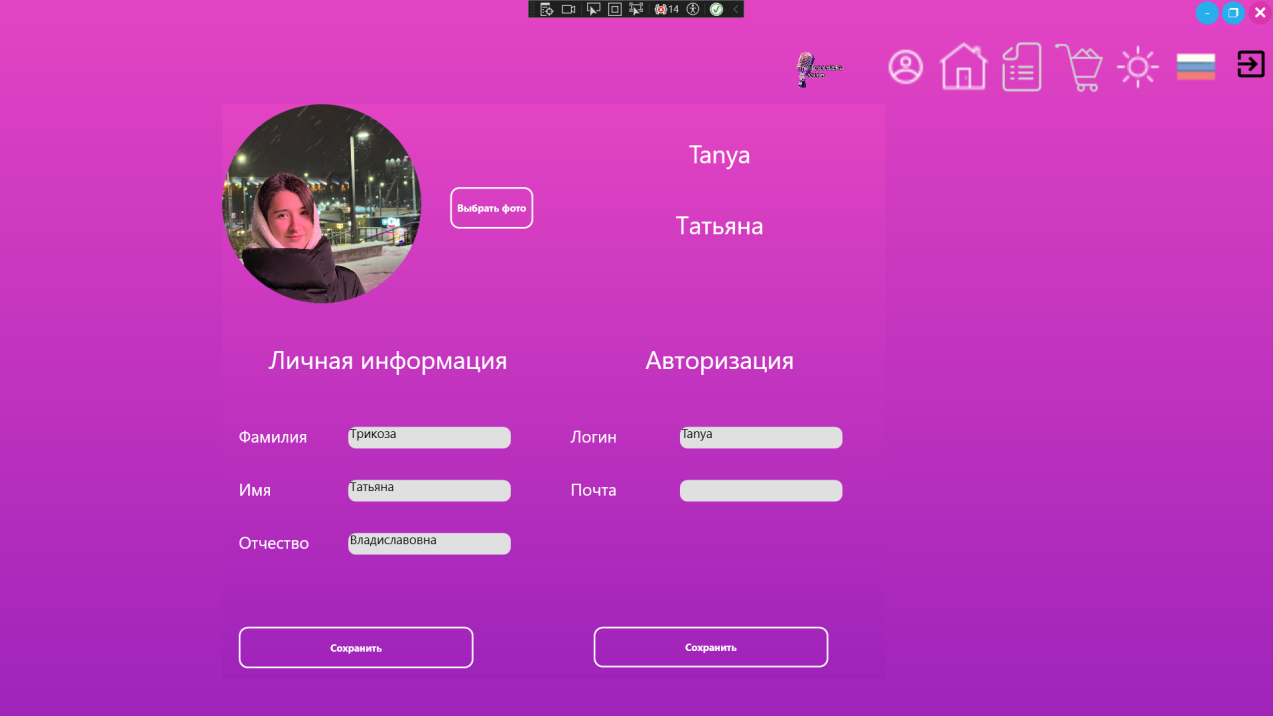


Рисунок 6.11 - Профиль пользователя

Нажав на кнопку “документ” мы переходим к общему перечню тарифов и преподавателей, которых мы можем отсортировать по различным перечням.

Как видно, кнопки для администратора скрылись.

Представлено на рисунке 6.12.

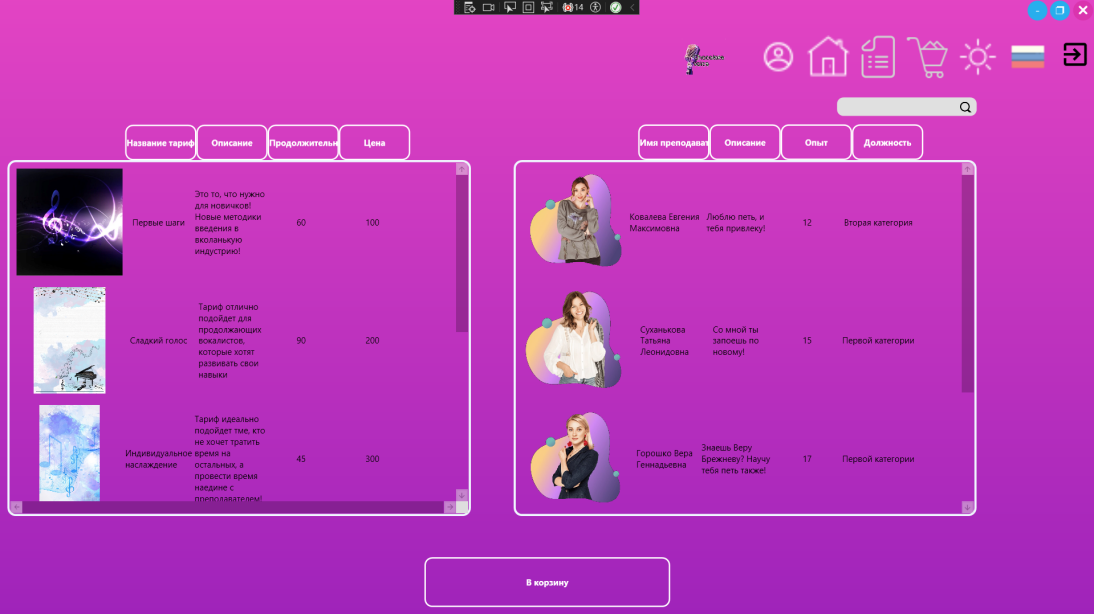


Рисунок 6.12 - Страница с перечнем тарифов и преподавателей

При двойном клике на изображение преподавателя из списка, пользователю открывается окно с заполненными данными для более детального ознакомления.

Профиль преподавателя представлен на рисунке 6.14.

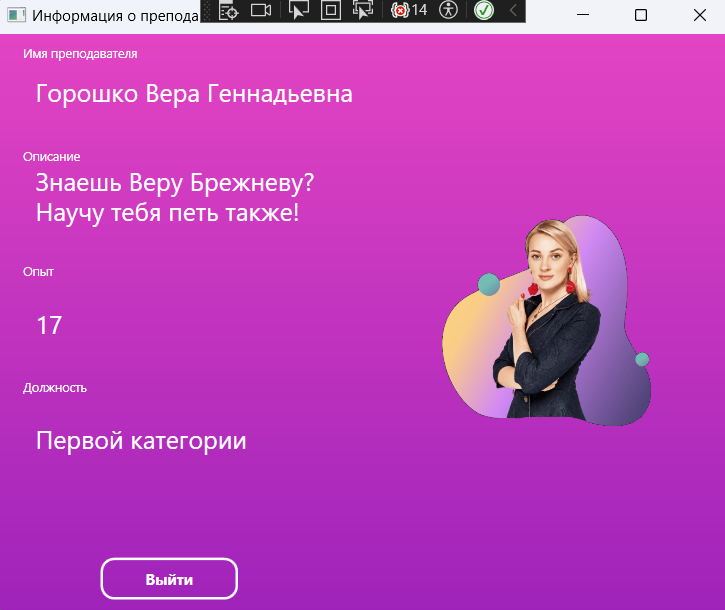


Рисунок 6.14 - Профиль преподавателя

При двойном клике на изображение тарифа из списка (рис. 6.15), пользователю открывается окно с заполненными данными для более детального ознакомления.

Пользователь также имеет возможность просмотреть отзывы и нажать на кнопку "Оставить отзыв", которая предоставит ему возможность анонимно оставить отзыв о текущем тарифе. (рисунок 6.16)

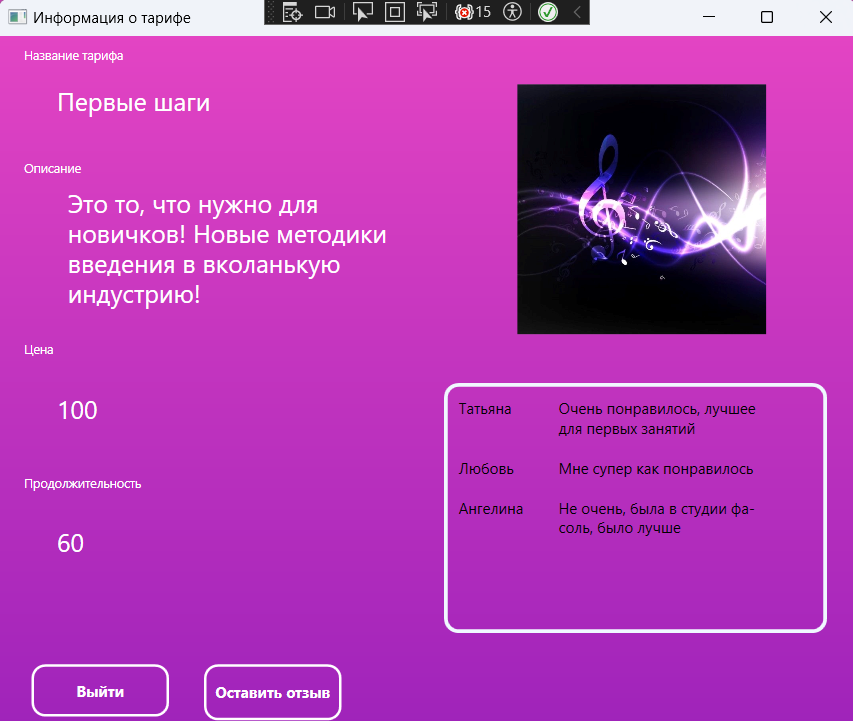


Рисунок 6.15 - Информация о тарифе.

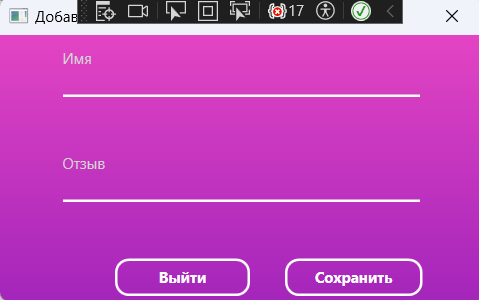


Рисунок 6.16 - Форма отзыва пользователя.

Далее пользователь может выбрать тариф и преподавателя, просто кликнув на её карточку и кликнуть на кнопку «в корзину». После данного действия приложение предоставит старицу для оформления заявки.

Данная форма представлена на рисунке 6.15.

Далее необходимо кликнуть один раз на выбранный тариф и нажать соответствующую ему кнопку «Оформить».

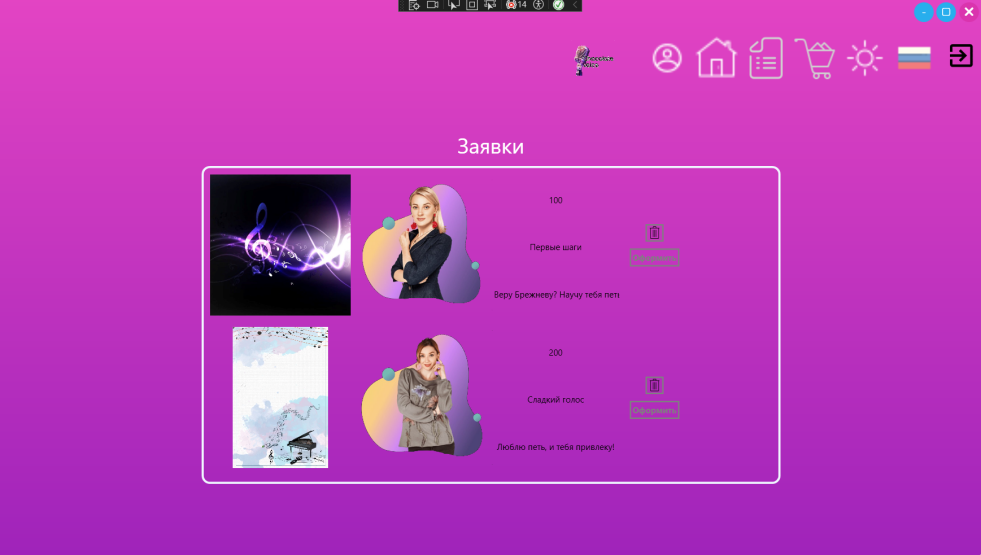


Рисунок 6.17- Оформление выбранной заявки

После успешного оформления заявки пользователь может просмотреть успешность записи в своей почте. (рисунок 6.18, 6.19 )

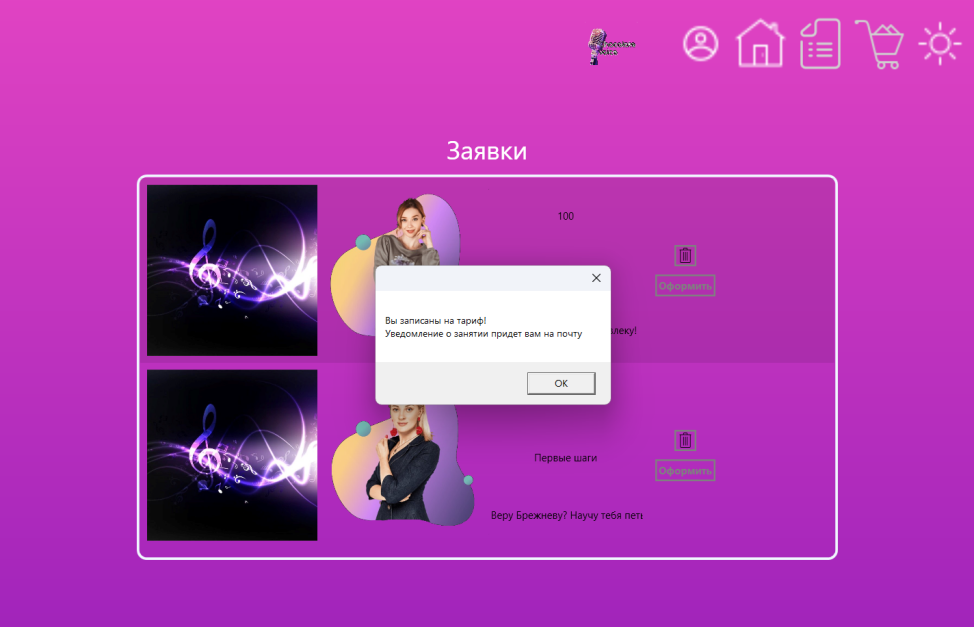


Рисунок 6.18 - Успешное оформление завяки

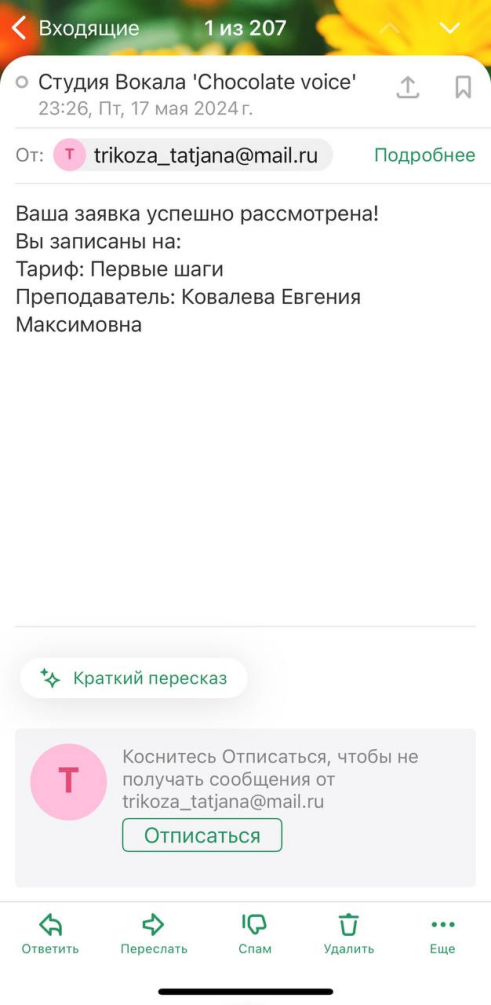


Рисунок 6.19 - Письмо об информации о записи на тариф.

## **6.3 Вывод**

Подготовка и внедрение руководств по использованию для администраторов и пользователей являются ключевыми этапами в обеспечении успешного запуска и эксплуатации программного средства. Эти руководства позволяют пользователям различных уровней уверенно и эффективно использовать все возможности системы, что способствует её стабильной работе и удовлетворенности всех участников процесса.

**ЗАКЛЮЧЕНИЕ**

В итоге выполнения данного курсового проекта было разработано программное средство “Студия вокала”. При разработке были выполнены все пункты из указанного списка предполагаемого основного функционала приложения.

В программном средстве были реализованы следующие функции:

* регистрация пользователя в системе;
* вход пользователя в систему;
* вход администратора в систему;
* просмотр данных о преподавателях, тарифах и отзывах;
* добавление данных о преподавателях и тарифах;
* удаление данных о преподавателях и тарифах;
* редактирование страниц преподавателей и тарифов;
* оформление заявки на выбранный тариф;
* добавление отзывов;
* изменение информации в личном кабинете;
* изменение языка приложения;
* изменение темы приложения.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно.

# **Cписок использованных источников**

1 Онлайн-ресурс студии вокала «Medley» [Электронный ресурс] – Режим доступа : <https://medley.by/> – Дата доступа: 19.05.2024.

2 Онлайн-ресурс студии вокала «Фа-Соль» [Электронный ресурс] – Режим доступа : https://schoolfasol.by/ – Дата доступа: 19.05.2024.

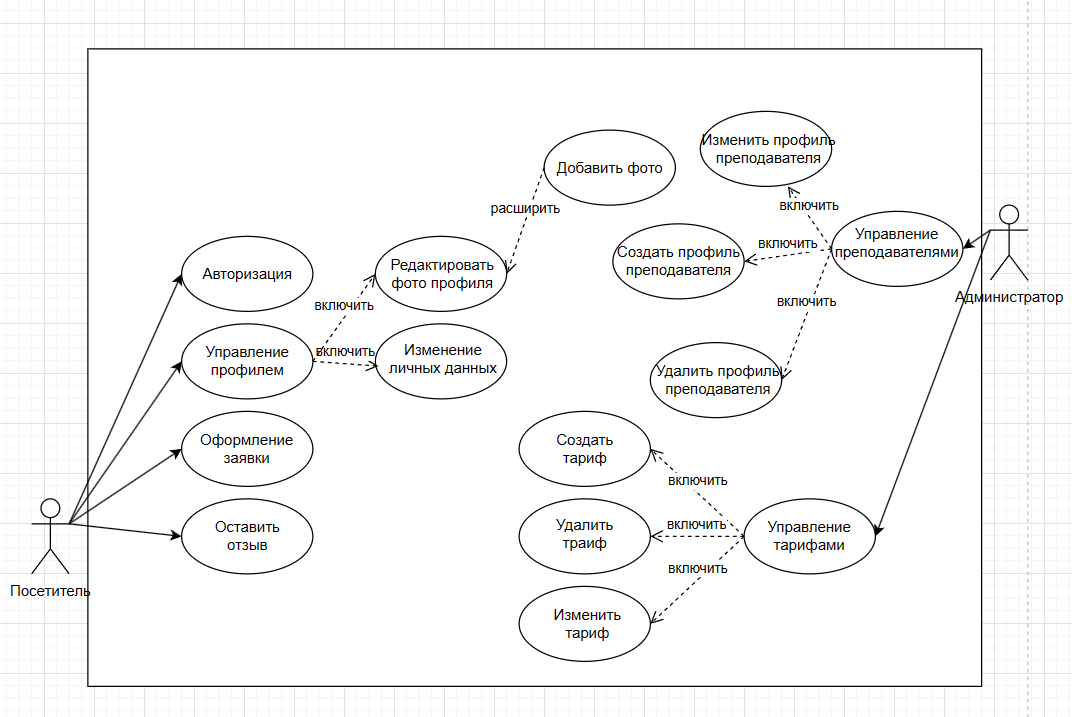
3 Онлайн-ресурс студии вокала «Cult Academy» [Электронный ресурс] – Режим доступа : https://cult.academy/ – Дата доступа: 19.05.2024.

4 Пацей Н.В. Курс лекций по языку программирования С# – Н.В. Пацей – Минск: БГТУ, 2016. – 175 с.

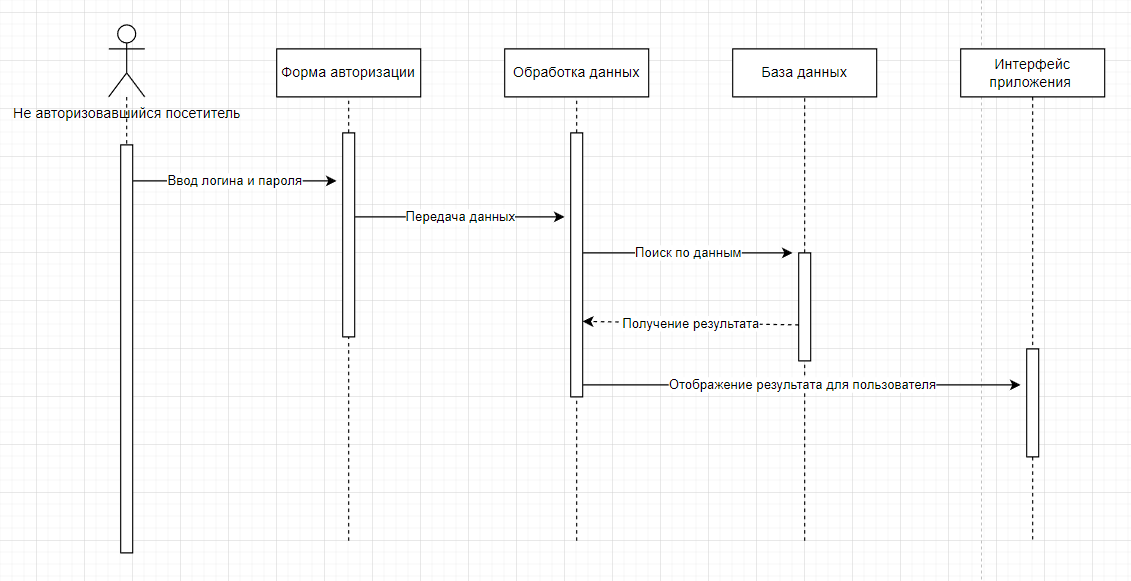
5 METANIT.COM Сайт о программировании [Электронный ресурс] – Режим доступа: <https://metanit.com>. – Дата доступа: 19.05.2024.

6 Руководство по WPF Сайт о программировании [Электронный ресурс] –Режим доступа: <https://metanit.com/sharp/wpf/#google_vignette>. – Дата доступа: 19.05.2024.

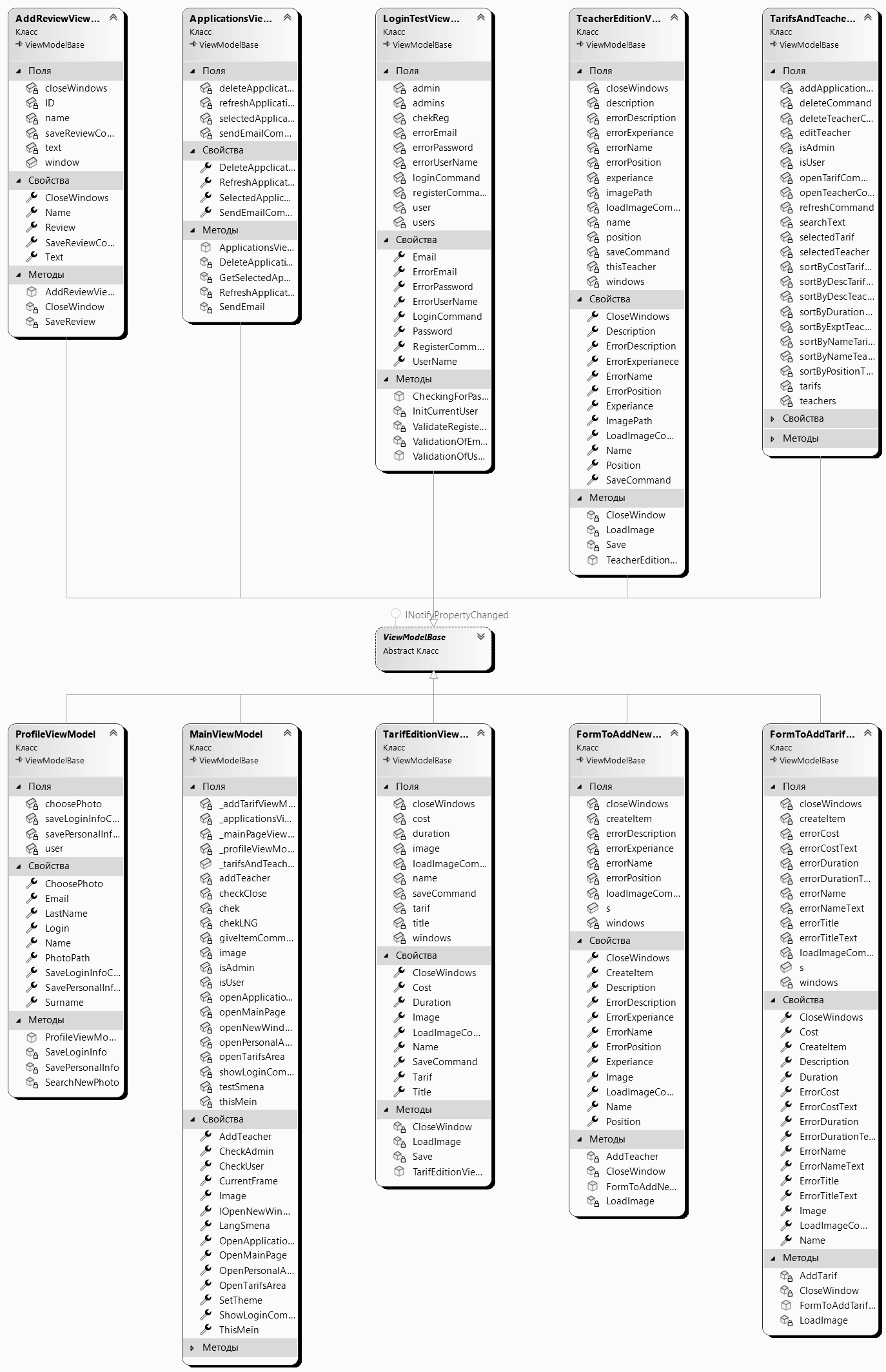
# **ПРИЛОЖЕНИЕ А**



# **ПРИЛОЖЕНИЕ Б**



# **ПРИЛОЖЕНИЕ В**



# **ПРИЛОЖЕНИЕ Г**

Листинг 1: Класс ApplicationsViewModel

|  |
| --- |
| using SampleMVVM.Commands;  using SampleMVVM.Managers;  using SampleMVVM.Model.BD;  using System;  using System.Collections.ObjectModel;  using System.Linq;  using System.Net.Mail;  using System.Net;  using System.Windows;  using System.Windows.Input;  namespace SampleMVVM.ViewModels  {  public class ApplicationsViewModel : ViewModelBase  {      private ApplicationDetails selectedApplication;  public ApplicationDetails SelectedApplication  {  get { return selectedApplication; }  set  {  selectedApplication = value;  OnPropertyChanged(nameof(SelectedApplication));  }  }  public ApplicationsViewModel()  {  RefreshAllApplications();  }  private ApplicationDetails GetSelectedApplication()  {  return SelectedApplication;  }  private DelegateCommand deleteAppclicationCommand;  public ICommand DeleteAppclicationCommand  {  get  {  if (deleteAppclicationCommand == null)  {  deleteAppclicationCommand = new DelegateCommand(DeleteApplication);  }  return deleteAppclicationCommand;  }  }  private void DeleteApplication()  {  var selectedApp = GetSelectedApplication();  if (selectedApp != null)  {  StudioManager.Instance.unitOfWork.ApplicationRepository.Delete(selectedApp.ApplicationId);  ApplicationDetailss.Remove(selectedApp);  RefreshAllApplications();  }  }  private DelegateCommand sendEmailCommand;  public ICommand SendEmailCommand  {  get  {  if (sendEmailCommand == null)  {  sendEmailCommand = new DelegateCommand(SendEmail);  }  return sendEmailCommand;  }  }  private void SendEmail()  {  try  {  SmtpClient client = new SmtpClient("smtp.mail.ru", 587);  client.EnableSsl = true;  client.Timeout = 20000;  client.DeliveryMethod = SmtpDeliveryMethod.Network;  client.UseDefaultCredentials = false;  client.Credentials = new NetworkCredential("trikoza\_tatjana@mail.ru", "17DtZq8DCuun0FzidzV9");  MailMessage mailMessage = new MailMessage();  mailMessage.From = new MailAddress("trikoza\_tatjana@mail.ru");  mailMessage.To.Add(CurrentUser.Email); // Получатель  mailMessage.Subject = "Студия Вокала 'Chocolate voice' ";  var selectedApplication = GetSelectedApplication();  string messageBody = "Ваша заявка успешно рассмотрена!\nBы записаны на:\n";  messageBody += $"Тариф: {selectedApplication.Tarif.Name}\n";  messageBody += $"Преподаватель: {selectedApplication.Teacher.Name}";    mailMessage.Body = messageBody;    client.Send(mailMessage);  MessageBox.Show("Вы записаны на тариф!\nУведомление о занятии придет вам на почту");  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при отправке письма: {ex.Message}");  }  }  private DelegateCommand refreshApplicationsCommand;  public ICommand RefreshApplicationsCommand  {  get  {  if (refreshApplicationsCommand == null)  {  refreshApplicationsCommand = new DelegateCommand(RefreshApplications);  }  return refreshApplicationsCommand;  }  }  private void RefreshApplications()  {  RefreshAllApplications();  }  }  public class ApplicationDetails  {  public int ApplicationId;  public Tarif Tarif { get; set; }  public Teacher Teacher { get; set; }  }  } |

Листинг 2: Класс PasswordHasher

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Security.Cryptography;  using System.Text;  using System.Threading.Tasks;  namespace Chocovoice.Model  {  public class PasswordHasher  {  public static string HashPassword(string password)  {  using (var sha256 = SHA256.Create())  {  byte[] hashedBytes = sha256.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));  return Convert.ToBase64String(hashedBytes);  }  }  public static bool VerifyPassword(string password, string hashedPassword)  {  return string.Equals(password, hashedPassword);  }  }  } |