Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

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

Тема Программное средство «Платформа по продаже битов»

Исполнитель

студент 3 курса группы 7 Бернович Никита Сергеевич

(Ф.И.О.)

Руководитель работы Курилец А.В.

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

Курсовой проект защищен с оценкой

Председатель

(подпись)

Минск 2024

**Содержание**

[ВВЕДЕНИЕ 2](#_Toc185291833)

[1. Аналитический обзор литературы и формирование требований 3](#_Toc185291834)

[1.1. Анализ прототипов 3](#_Toc185291835)

[1.1.1. BeatStars 3](#_Toc185291836)

[1.1.2. SoundClick 4](#_Toc185291837)

[1.2. Требования к проекту 5](#_Toc185291838)

[2. Анализ требований к программному средству и разработка функциональных требований 6](#_Toc185291839)

[2.1. Описание средств разработки 6](#_Toc185291840)

[2.2. Спецификация функциональных требований к программному средству 8](#_Toc185291841)

[2.3. Спецификация функциональных требований 8](#_Toc185291842)

[3. Проектирование программного средства 9](#_Toc185291843)

[3.1. Общая структура 9](#_Toc185291844)

[3.2. Модель базы данных 10](#_Toc185291845)

[3.3. Проектирование архитектуры проекта 14](#_Toc185291846)

[3.4. Проектирование последовательностей проекта 14](#_Toc185291847)

[4. Реализация программного средства 15](#_Toc185291848)

[4.1. Основные классы программного средства 15](#_Toc185291849)

[4.2. Описание классов и методов программного средства 15](#_Toc185291850)

[4.2.1. Выполнение входа 15](#_Toc185291851)

[4.2.2. Регистрация 16](#_Toc185291852)

[4.2.3. Просмотр битов 16](#_Toc185291853)

[4.2.4. Выполнение добавления бита 17](#_Toc185291854)

[4.2.5. Оценка бита 17](#_Toc185291855)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 18](#_Toc185291856)

[5.1. Тестирование авторизации и регистрации 18](#_Toc185291857)

[5.2. Тестирование добавления бита 19](#_Toc185291858)

[5.3. Тестирование пагинации 20](#_Toc185291859)

[6. Руководство по установке и использованию 22](#_Toc185291860)

[Заключение 26](#_Toc185291861)

[Список литературы 27](#_Toc185291862)

[ПРИЛОЖЕНИЕ А 28](#_Toc185291863)

[ПРИЛОЖЕНИЕ Б 29](#_Toc185291864)

[ПРИЛОЖЕНИЕ В 30](#_Toc185291865)

## **ВВЕДЕНИЕ**

Курсовой проект посвящен созданию приложения на WPF с использованием базы данных PostgreSQL для управления продажами битов. В современном музыкальном мире многие продюсеры, диджеи и музыканты нуждаются в удобной платформе для хранения, управления и продажи своих произведений. Основной целью данного проекта является разработка надежного и эффективного приложения для продажи битов, обеспеченного функциональной базой данных.

В рамках проекта будет реализовано приложение на WPF, которое взаимодействует с базой данных PostgreSQL для обеспечения надежного хранения информации о битах, продюсерах, жанрах, ценах и других ключевых атрибутах. Приложение позволит продюсерам загружать свои биты, устанавливать цены и описания, а пользователям — просматривать, покупать и оценивать биты. Важными функциями платформы также являются регистрация и авторизация пользователей, поиск и удаление битов, а также получение процента с продаж для модератора.

В системе будут предусмотрены две роли: модератор и пользователь. Модератор сможет управлять контентом, удалять биты и получать процент с продаж, а пользователь — покупать, продавать и оценивать биты. Приложение также будет включать функционал для регистрации и авторизации пользователей, обеспечивая безопасный доступ к системе. Интуитивный поиск по различным критериям упростит пользователям выбор и покупку нужных битов.

Разработка базы данных PostgreSQL в сочетании с приложением на WPF создаст удобную и эффективную систему для управления продажами битов, удовлетворяющую потребности как продюсеров, так и пользователей. При этом особое внимание будет уделено оптимизации производительности, чтобы обеспечить быструю обработку операций и удобство использования приложения.

1. Аналитический обзор литературы и формирование требований

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

1. Анализ прототипов

В ходе курсового проекта были изучены цели и задачи, поставленные перед нами. Были также рассмотрены примеры аналогичных решений, и произведен анализ их достоинств и недостатков. Основываясь на этом анализе, мы сформулировали требования к разрабатываемому программному средству, учитывая как позитивные аспекты, так и проблемы, выявленные в альтернативных решениях.

1. BeatStars

Сайт «BeatStars» — это популярная онлайн-платформа для продажи и покупки битов и музыкальных инструменталов. «BeatStars» предоставляет продюсерам и музыкантам возможность размещать свои биты для продажи, а артистам и исполнителям — находить и покупать подходящую музыку для своих проектов. Сайт объединяет большую аудиторию продюсеров, композиторов и исполнителей со всего мира.

Интерфейс «BeatStars» удобен и интуитивно понятен. На главной странице пользователи могут найти разделы, такие как «Биты», «Акаунты продюсеров», «Жанры», «Тренды» и другие категории, что позволяет легко ориентироваться на платформе. Также есть специальные разделы для новых релизов и популярного контента, помогающие пользователям оставаться в курсе самых свежих и популярных битов.

Функционал «BeatStars» обширен и включает все необходимое для управления продажами и покупками битов. Продюсеры могут загружать свои треки, устанавливать цену, добавлять описание и ключевые теги для удобного поиска. Также есть возможность устанавливать разные лицензии для покупки, например, аренда или покупка с полной передачей прав.

Артисты могут прослушивать демо-версии, использовать различные фильтры по жанрам, настроению, типу лицензии и цене, чтобы быстрее находить нужные биты. Сайт «BeatStars» поддерживает различные формы оплаты и предлагает продюсерам возможность получать доход через прямую продажу. Также есть программы подписки и рекламные инструменты, которые позволяют продюсерам расширять охват и улучшать видимость своих битов.

Платформа также включает социальные элементы — пользователи могут подписываться друг на друга, лайкать и комментировать работы, что делает «BeatStars» не просто магазином, но и сообществом для обмена творчеством.

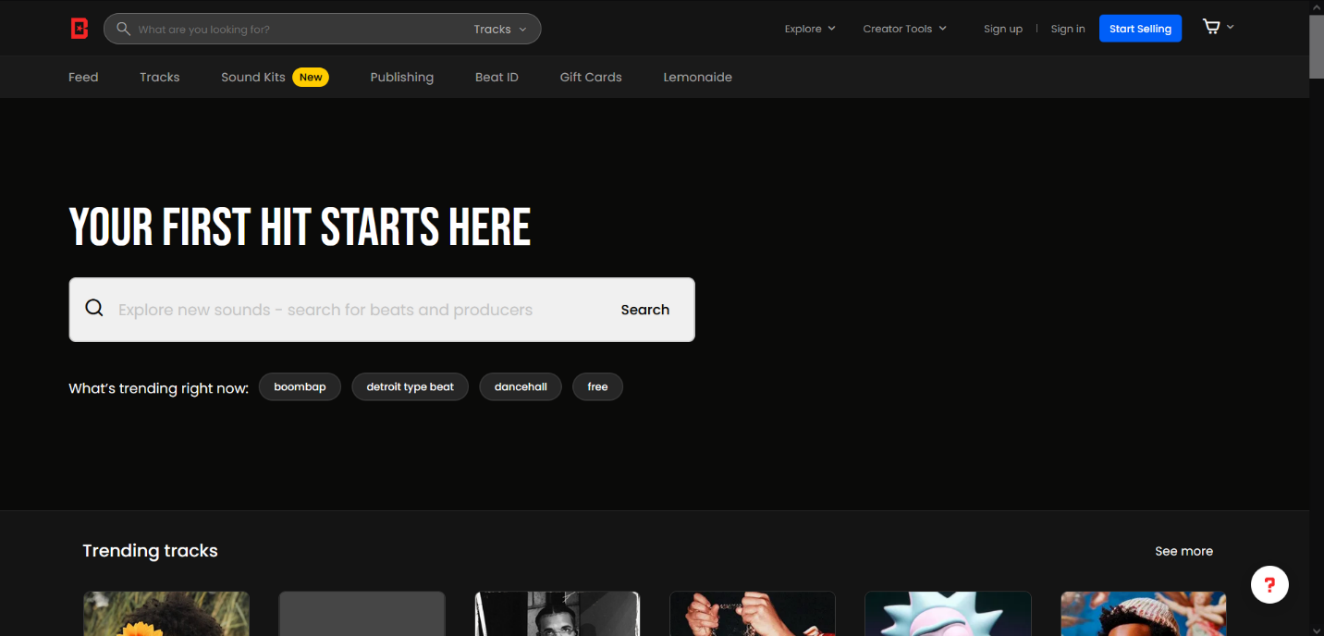


Рисунок 1.1 − Интерфейс «Beatstars»

В целом, сайт «BeatStars» является одним из лучших аналогов по функционалу и интерфейсу для продажи музыкального контента. Он предлагает широкий набор инструментов для продюсеров и музыкантов, а также интуитивный интерфейс и удобные функции для покупателей, что делает процесс покупки и продажи битов простым и эффективным.

1. SoundClick

Сайт «SoundClick» — это онлайн-платформа для размещения, продажи и покупки музыки, ориентированная на независимых артистов, продюсеров и музыкантов. «SoundClick» предоставляет возможности как для продаж музыкальных треков, так и для их бесплатного прослушивания, что делает платформу популярной среди творческих людей, желающих продвигать свое искусство и зарабатывать на нем. «SoundClick» активно используется музыкантами для создания персональных страниц, на которых они могут выкладывать свои треки, общаться с фанатами и продавать музыку.

Интерфейс сайта «SoundClick» простой и функциональный. Главная страница предлагает категории, такие как «Музыка», «Чарты», «Артисты» и «Сообщества», что облегчает навигацию по платформе. Пользователи могут искать музыку по жанрам, рейтингу, популярности и другим параметрам. У каждого артиста есть собственная страница, где можно разместить музыку, описание, фотографии и контактные данные, а также настроить параметры продаж.

Функционал «SoundClick» включает инструменты для монетизации, которые позволяют музыкантам продавать свои треки, устанавливая цену и условия лицензирования, такие как аренда и продажа прав. «SoundClick» также предлагает возможность стриминга музыки и загрузки треков в высоком качестве. Платформа поддерживает различные формы оплаты и предоставляет статистику по прослушиваниям и продажам, что помогает артистам отслеживать успех своих композиций и взаимодействие с аудиторией.

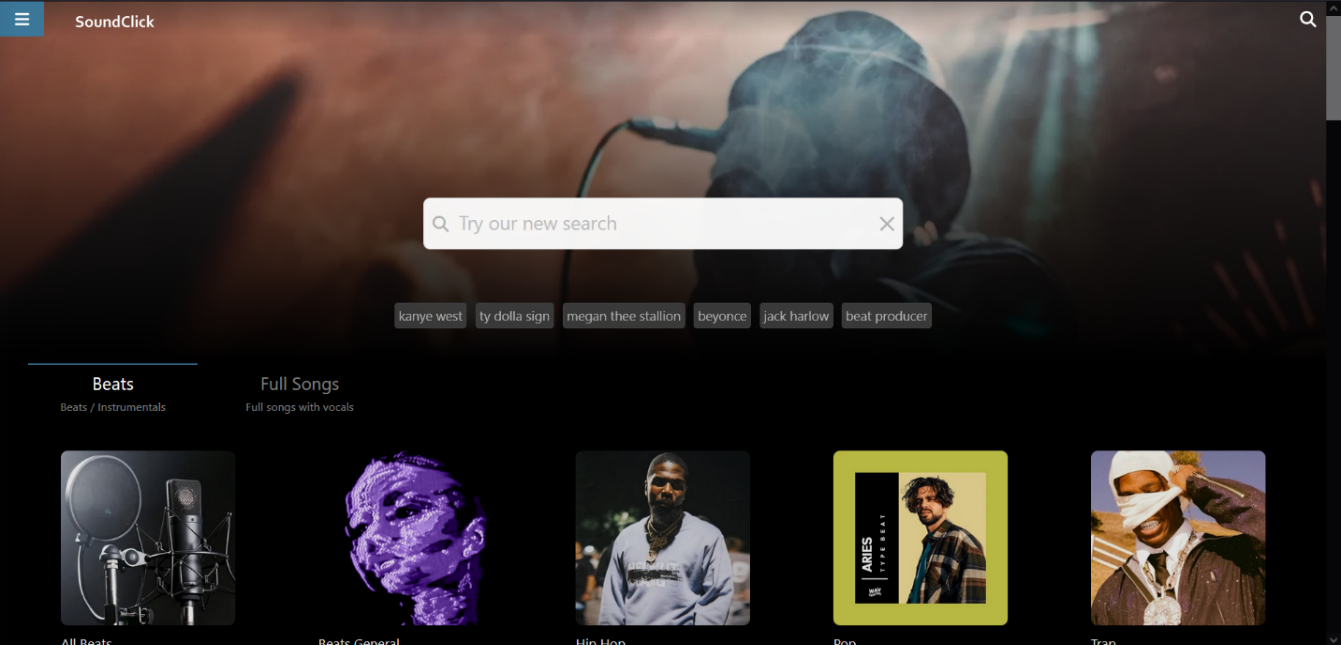


Рисунок 1.2 – Сайт «SoundClick»

В целом, сайт «SoundClick» является удобной платформой для независимых музыкантов, предоставляющей функционал для размещения, продвижения и продажи музыки. Платформа сочетает простоту интерфейса с полезными инструментами для монетизации и продвижения, что делает «SoundClick» эффективным и доступным решением для артистов, стремящихся к самовыражению и коммерческому успеху.

1. Требования к проекту

Обзор вышеперечисленных известных аналогов позволяет проанализировать все преимущества и недостатки альтернативных возможностей и позволяет сформулировать список требований, предъявляемых к программному средству, разрабатываемому в данном курсовом проекте. Программное средство должно обеспечивать возможность выполнения перечисленных ниже функций:

* поддержать работу с базой данных;
* возможность пользователю зарегистрироваться или войти в существующую учетную запись;
* возможность авторизированным пользователям добавлять, удалять, изменять биты;
* возможность покупки и прожажи битов;
* возможность оценки битов;
* возможность искать биты.

В ходе анализа прототипов программных средств, таких как «BeatStars» и «SoundClick», были изучены их основные функции, интерфейсы и возможности для пользователей.

* «BeatStars» отличается широким функционалом, включая продажу и покупку музыкальных битов, поддержку лицензий, удобную навигацию и социальные элементы, что делает его одной из ведущих платформ в этой сфере.
* «SoundClick» предлагает простой и функциональный интерфейс, монетизацию, инструменты продвижения и возможность для независимых музыкантов создавать персонализированные страницы, что делает его эффективным решением для творческих людей.

На основе проведенного анализа были выявлены сильные стороны и недостатки платформ, которые использованы для формирования требований к разрабатываемому программному средству.

1. Анализ требований к программному средству и разработка функциональных требований

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

1. Описание средств разработки

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET 8.0;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология ADO.NET;
* PostgreSQL.

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 использованиям таких технологий, как WinForms или WPF [1].

1. Программная платформа .NET Framework 4.7.2

Платформа .NET Framework — это созданная в 2002 году технология Microsoft, которая поддерживает создание и выполнение веб-служб и приложений Windows, обеспечивающая согласованную объектно-ориентированной среду программирования для локального сохранения и выполнения объектного кода, локального выполнения кода, распределенного в Интернете, либо для его удаленного выполнения. Основой платформы является общеязыковая среда Common Language Runtime, выполняющая код на разных поддерживаемых языках программирования [2].

1. 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 [3].

1. 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 [4].

1. 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.

1. Технология ADO.NET

Для взаимодействия с базой данных в приложении WPF используется технология ADO.NET, которая предоставляет доступ к данным в реляционных базах через набор компонентов .NET. ADO.NET является частью .NET Framework и позволяет работать с данными, выполняя различные операции, такие как подключение к базе данных, выполнение SQL-запросов и обработка результатов.

Основные объекты ADO.NET включают:

* Connection — объект, который устанавливает подключение к базе данных, поддерживая работу с разными типами источников данных.
* Command — объект для выполнения SQL-запросов и команд, таких как SELECT, INSERT, UPDATE, DELETE.
* DataReader — объект для быстрого и эффективного считывания данных из базы, позволяющий получить только для чтения и только последовательный доступ к данным.
* DataSet и DataTable — объекты, которые позволяют кэшировать данные в памяти, работать с ними как с локальной копией и выполнять фильтрацию, сортировку и редактирование без постоянного обращения к базе данных.

ADO.NET поддерживает как подключенный, так и отключенный подходы к управлению данными. В подключенном подходе данные считываются и обрабатываются в реальном времени, что подходит для небольших операций. В отключенном подходе, используемом с объектами DataSet и DataTable, данные кэшируются в памяти и могут обрабатываться независимо от подключения к базе.

В отличие от технологий высокого уровня, таких как Entity Framework, ADO.NET позволяет разрабатывать точные и производительные запросы, что дает больше контроля над выполнением операций и подходит для приложений с требованиями к высокой производительности и оптимизации работы с базой данных.

1. PostgreSQL

Для организации баз данных PostgreSQL также использует реляционную модель, в которой данные хранятся в виде таблиц, состоящих из строк и столбцов. Каждая строка представляет собой отдельный объект, а столбцы — его атрибуты. Для взаимодействия с базой данных используется язык SQL (Structured Query Language), позволяющий клиентам (например, внешним программам) отправлять запросы к базе данных. PostgreSQL интерпретирует, выполняет SQL-запросы и возвращает результат клиенту. Основной используемый язык запросов — стандартный SQL, но PostgreSQL также предлагает поддержку расширений, таких как PL/pgSQL (процедурный язык PostgreSQL), который добавляет возможности процедурного программирования к стандартным SQL-запросам.

1. Спецификация функциональных требований к программному средству

Программное средство должно предоставлять следующие функциональные возможности:

Для пользователя:

* регистрация;
* авторизация;
* просмотр битов;
* поиск битов;
* покупка, продажа бита;
* оценка битов.

Для администратора:

* авторизация;
* удаления битов;
* получение процента с продаж.

1. Спецификация функциональных требований

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

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

Пользователь может просматривать биты, искать биты, продавать биты, покупать биты и оставлять оценки. Все сведения пользователе должны автоматически обновляться в базе данных.

Администратор обладает теми же возможностями, что и пользователь, а также обладает возможностями удалять биты, и получать прибыль с продаж.

В разделе проанализированы технологии и инструменты, применяемые для разработки программного средства, а также сформулированы основные функциональные требования.

Определены ключевые возможности приложения для пользователей и администраторов, включая регистрацию, авторизацию и взаимодействие с данными. Также обозначена необходимость создания базы данных для обеспечения функциональности системы.

Этот этап позволил структурировать требования и подготовить основу для реализации программного продукта.

1. Проектирование программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

1. Общая структура

Программное средство имеет следующую структуру, представленную на рисунке 3.1.

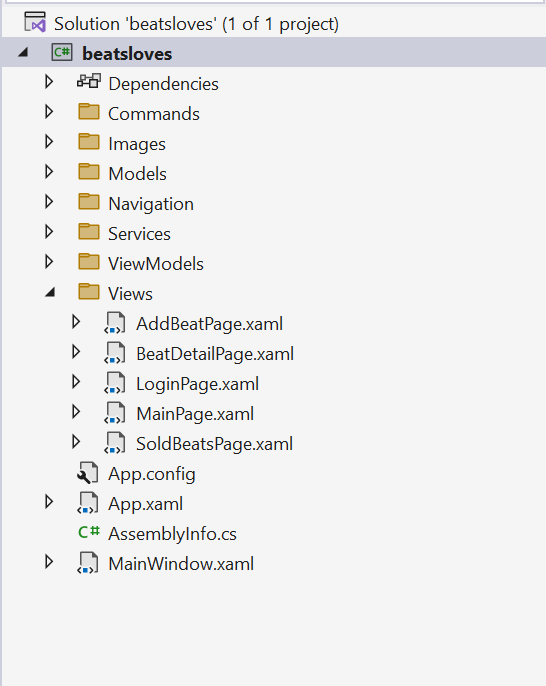


Рисунок 3.1 – Структура проекта

Описание структуры основных папок и файлов проекта представлено в таблице 3.1.

Таблица 3.1 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя файла | Содержание |
| App.config | Файл с параметрами приложения. |
| Папка Models | Данные и состояние приложения. |
| Папка Resources | Все изображения. |
| Папка ViewModels | Классы модели представления |
| Папка Views | Визуальное представление окон приложения |
| Папка Navigation | Класс для навигации |
| Папка Services | Сторонние сервисы |
| Папка Commands | Класс для реализации команд |
| App.xaml | Глобальные ресурсы |

На основании таблицы можно понять общую структуру приложения.

1. Модель базы данных

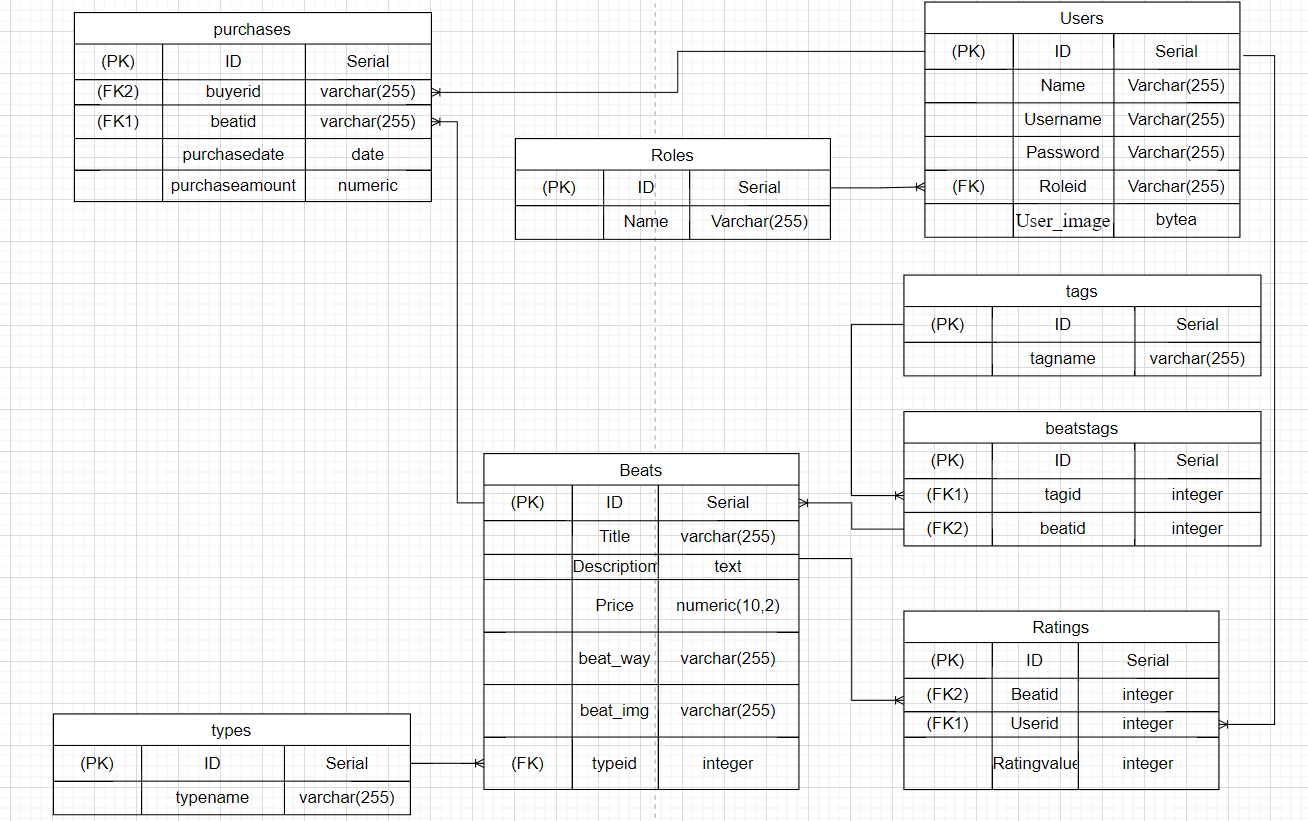
Для реализации поставленной задачи была создана база данных. Для ее создания использовалась система управления реляционными базами данных PostgreSQL. База данных состоит из таблиц, представленных на рисунке 3.2. Скрипт для создания базы данных представлен в приложении В

Рисунок 3.2 – База данных

На рисунке 3.3 проиллюстрирована структура таблицы «Beats». Таблица «Beats» предназначена для хранения информации о всех битах. В ней 8 столбцов:

* ID – ключевое поле таблицы, содержащее уникальный идентификатор бита;
* title – заголовок бита;
* description – описание бита;
* price – цена бита;
* typeid – уникальный идентификатор типа бита;
* beatway – путь к биту;
* beatimg – изображение бита;
* userid – уникальный идентификатор пользователя;

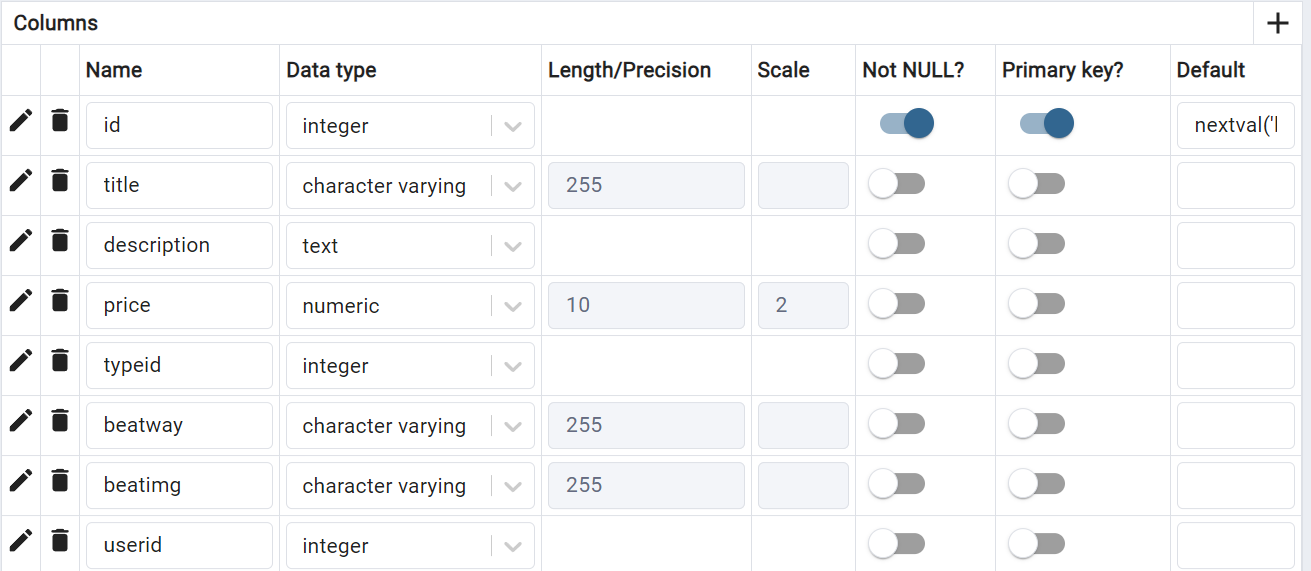


Рисунок 3.3 – Структура таблицы «Beats»

На рисунке 3.4 изображена структура таблицы «Users». Таблица «Users» предназначена для хранения основной информации о пользователях и содержит в себе 6 столбцов, а именно:

* ID – ключевое поле таблицы, содержащее уникальный идентификатор для каждого пользователя;
* username – имя пользователя;
* password – пароль пользователя;
* roleid – уникальный идентификатор для роли;
* balance ­– баланс пользователя;
* user\_image – содержит изображение пользователя

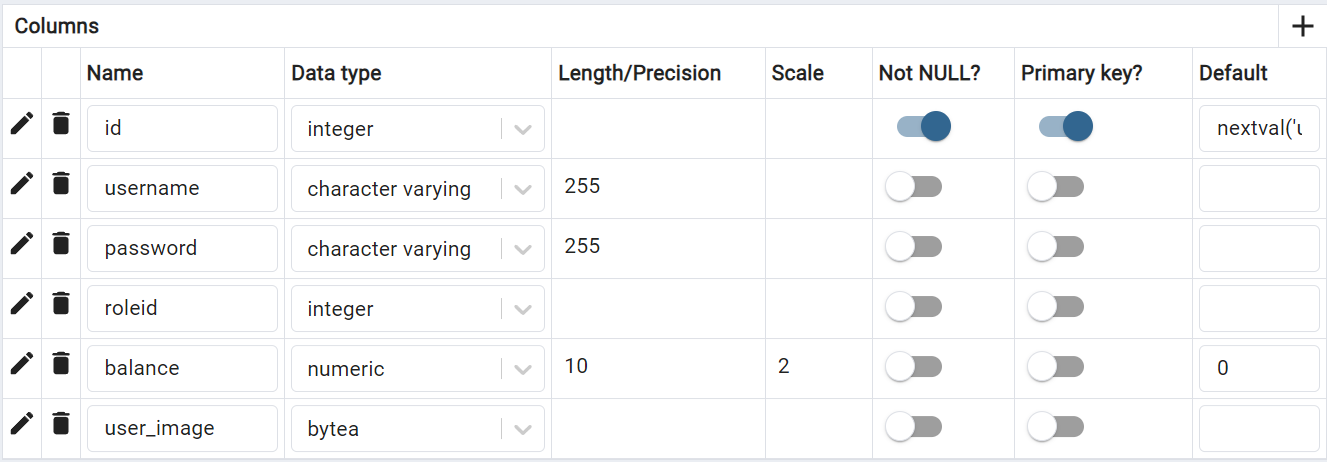


Рисунок 3.4 – Структура таблицы «Users»

На рисунке 3.5 показана структура таблицы «Roles». Таблица «Roles» предназначена для ролей. В ней 2 столбца:

* ID– идентификатор роли;
* name – название роли;

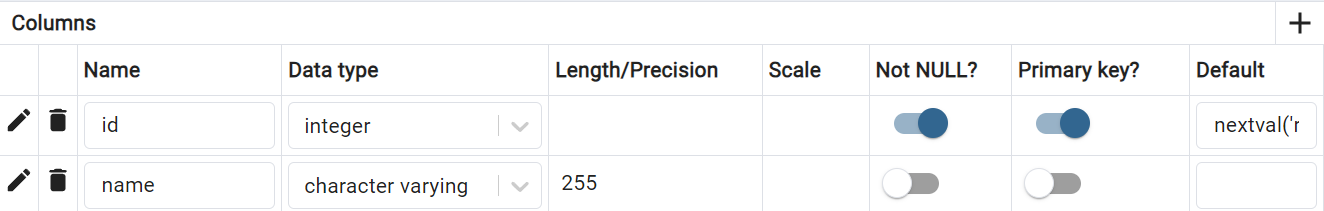


Рисунок 3.5 – Структура таблицы « Roles»

На рисунке 3.6 показана структура таблицы «Purchases». Таблица «Purchases» предназначена для хранения всех покупок, которые были произведены. В ней 5 столбца:

* ID – уникальный идентификатор покупки;
* buyerid – уникальный идентификатор покупателя;
* beatid – уникальный идентификатор бита;
* purchasedate –дата покупки.
* Purchaseamount – сумма покупки.

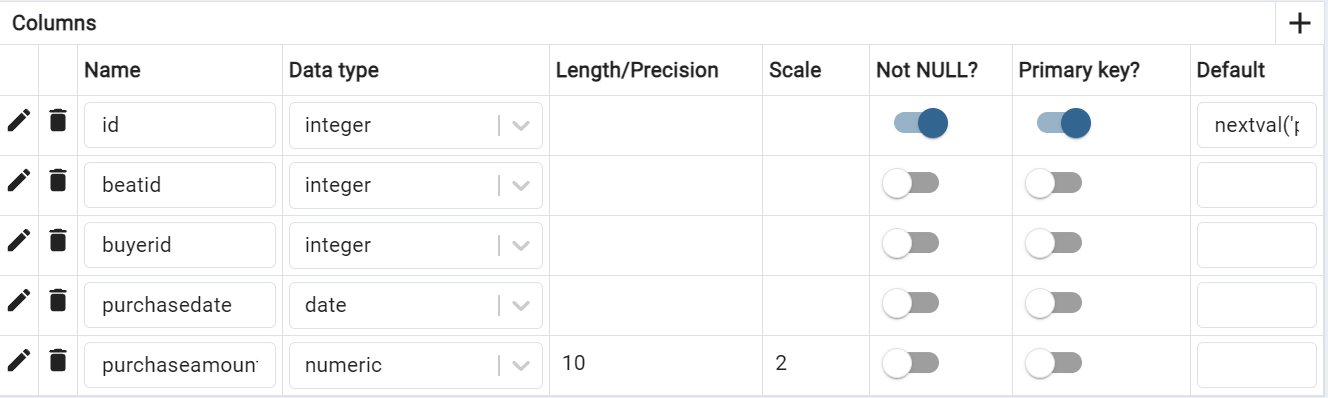


Рисунок 3.6 – Структура таблицы «Purchases»

На рисунке 3.7 показана структура таблицы «Tags». Таблица «Tags» предназначена для хранения всех тегов. В ней 2 столбца:

* ID – уникальный идентификатор тега;
* tagname – название тега;

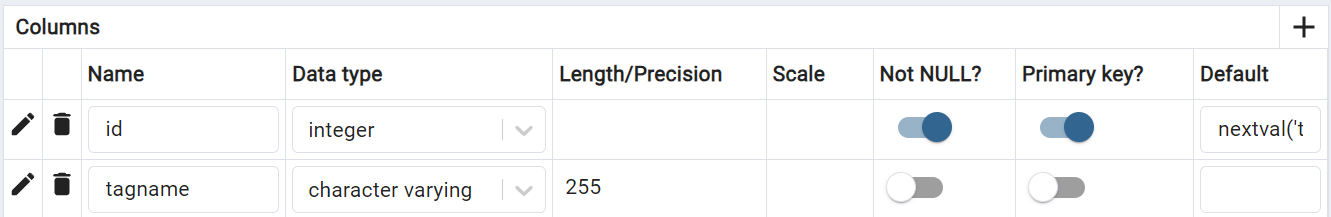


Рисунок 3.7 – Структура таблицы «Tags»

На рисунке 3.8 показана структура таблицы «Beatstags». Таблица «Beatstags» предназначена для связи тегов с битами. В ней 3 столбца:

* ID – уникальный идентификатор связи;
* beatid – уникальный идентификатор бита;
* tagid – уникальный идентификатор тега;

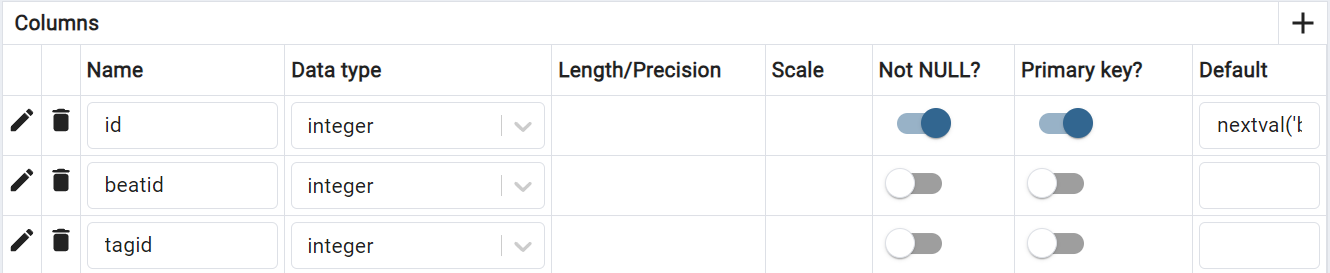


Рисунок 3.8 – Структура таблицы «Beatstags»

На рисунке 3.9 показана структура таблицы «Types». Таблица «Types» предназначена для хранения типов битов. В ней 2 столбца:

* ID – уникальный идентификатор типа;
* typename – имя типа;

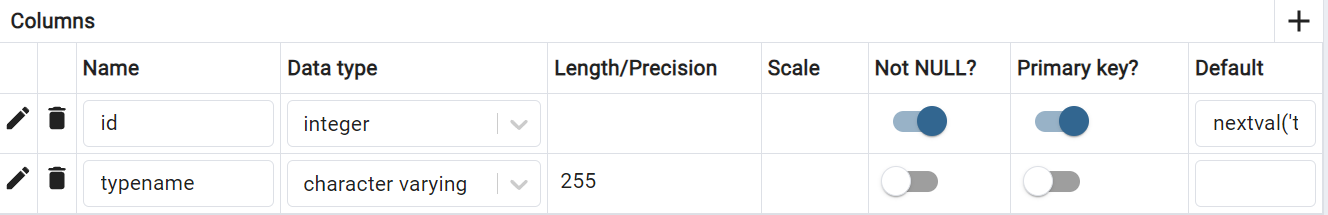


Рисунок 3.9 – Структура таблицы «Types»

На рисунке 3.10 показана структура таблицы «Ratings». Таблица «Ratings» предназначена для хранения оценок битов. В ней 4 столбца:

* ID – уникальный идентификатор типа;
* beatid – уникальный идентификатор бита;
* userid – уникальный идентификатор пользователя;
* ratingvalue – значение оценки;

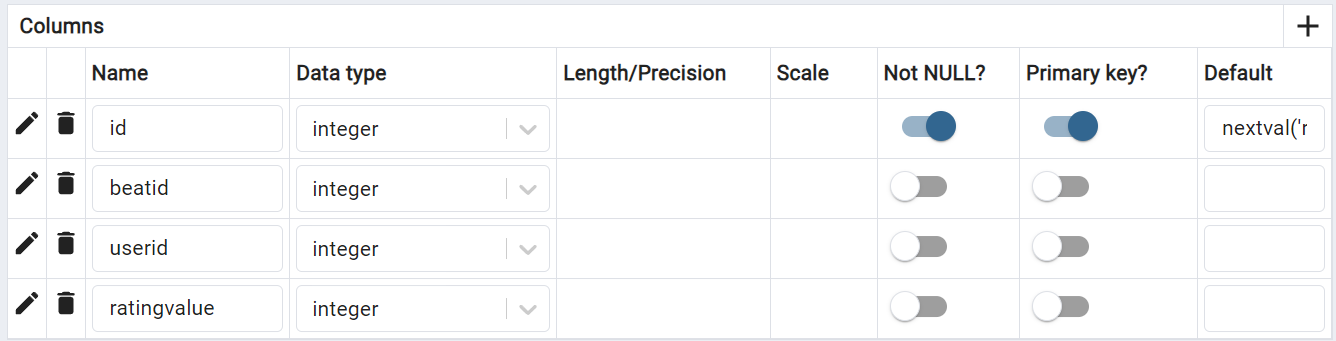


Рисунок 3.10 – Структура таблицы «Ratings»

В качестве системы управления базой данных была выбрана PostgreSQL, которая обеспечивает надежность, масштабируемость и высокую производительность. Модель базы данных представлена в виде набора таблиц, отражающих ключевые сущности и их взаимосвязи.

1. Проектирование архитектуры 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 взаимодействии с системой, и определяет набор действий, выполняемых этой системой.

Диаграмма использования представлена в приложении А.

1. 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 Б.

В данном разделе описаны основные этапы проектирования программного средства, включая структуру проекта, проектирование базы данных и архитектуры, а также моделирование взаимодействия компонентов системы.

* Определена структура проекта с разделением на папки и файлы, обеспечивающая удобство разработки и поддержки.
* Разработана модель базы данных, которая включает ключевые таблицы и их взаимосвязи, реализующие функциональность системы.
* Проектирование архитектуры выполнено с использованием диаграмм, таких как диаграмма использования и диаграмма последовательностей, которые позволяют визуализировать взаимодействие пользователей и системы, а также работу её компонентов.

1. Реализация программного средства
2. Основные классы программного средства

Программное средство разработано с учетом реализации всех необходимых функций для обеспечения работы пользователей и администраторов. Для пользователей предусмотрены возможности регистрации и авторизации, управления битами (просмотр, поиск, добавление, изменение, удаление, покупка и продажа), а также взаимодействия с профилем, включая добавление изображения. Администраторы имеют доступ к функционалу авторизации, управлению битами, а также получению прибыли с продаж.

Каждая из функций реализована через отдельные классы и методы, обеспечивающие их выполнение. Далее подробно рассмотрены все функциональные возможности, а также описание их реализации и структуры.

1. Описание классов и методов программного средства
2. Выполнение входа

Функционал для входа пользователей в систему реализован в классе LoginViewModel. Основной задачей данного класса является обработка пользовательских данных для входа, проверка их корректности и аутентификация через базу данных.

Ключевым методом является OnLogin(), который вызывается при нажатии на кнопку «Войти».

Проверка введённых данных. Проверяются введённые логин и пароль, а также выполняется их валидация с использованием базы данных.

Аутентификация пользователя. Введённый логин и пароль передаются в базу данных, где вызывается функция authenticate\_user. Если пользователь успешно аутентифицирован, происходит получение данных текущей сессии, включая UserID, RoleID, баланс и изображение пользователя. Если изображение отсутствует, используется placeholder-изображение.

Перенаправление. В случае успешной аутентификации пользователь перенаправляется на главную страницу. В случае ошибки появляется сообщение с соответствующим уведомлением.

Листинг реализации класса LoginViewModel представлен в приложении В.

1. Регистрация

Для регистрации новых пользователей реализован метод OnRegister(), также расположенный в классе LoginViewModel.

Валидация пользовательских данных. Проверяется длина логина и пароля, их непустое состояние и соответствие пароля заданным требованиям (наличие букв и цифр).

Добавление в базу данных. Используется функция register\_user, выполняемая через SQL-запрос. Если логин или другие данные уже зарегистрированы, пользователь получает уведомление об ошибке.

Уведомление о результате. При успешной регистрации выводится сообщение «Успешно», после чего пользователь может перейти к авторизации. Листинг реализации класса LoginViewModel представлен в приложении В.

1. Просмотр битов

После успешной авторизации пользователь перенаправляется на основную страницу программного средства, представленную классом MainViewModel, позволяющую перейти на страницу каталога битов.

При загрузке основной вкладки метод LoadBeats() обращается к базе данных для получения списка музыкальных битов с их характеристиками. Полученные данные добавляются в коллекцию Beats, отображаемую пользователю.

Для выполнения поиска пользователь вводит запрос в поле SearchQuery, после чего метод SearchBeats() отправляет запрос к хранимой процедуре SearchBeats в базе данных. Метод возвращает отфильтрованные биты, соответствующие запросу, и обновляет коллекцию Beats.

Метод LoadMyBeats() запрашивает список битов, созданных текущим пользователем (используется идентификатор из Session.UserID). Данные добавляются в коллекцию Beats.

Метод LoadPurchases() отображает биты, приобретенные текущим пользователем, используя хранимую процедуру GetMyBeatsPurchases

Листинг реализации класса MainViewModel представлен в приложении В.

1. Выполнение добавления бита

Пользователь может добавить новый бит или отредактировать существующий с помощью интерфейса приложения. По завершении ввода данных и нажатии на кнопку «Сохранить» вызывается метод AddOrEditBeat(), реализованный в классе AddBeatPageViewModel.

Если пользователь добавляет бит, то данные о нём, включая название, описание, цену, тип лицензии, путь к файлу, изображение и теги, отправляются в базу данных с помощью хранимой процедуры AddBeat. В случае редактирования бита метод использует хранимую процедуру UpdateBeat, в которую также передаётся идентификатор редактируемого бита.

Листинг реализации класса AddBeatPageViewModel представлен в приложении В.

1. Оценка бита

Пользователь может оценить бит, используя интерфейс приложения. Для этого в классе BeatDetailViewModel реализован метод ExecuteRateBeat(), который позволяет установить оценку для выбранного бита. Оценка сохраняется в базе данных с помощью хранимой процедуры SetBeatRating, принимающей идентификатор бита, идентификатор пользователя и значение оценки.

После успешной отправки оценки вызывается метод ExecuteLoadAverageRating(), который обновляет среднюю оценку для текущего бита, используя процедуру CalculateAverageRatingForBeat. Средняя оценка отображается в интерфейсе, что позволяет пользователю увидеть актуальные данные

Листинг реализации класса BeatDetailViewModel представлен в приложении В.

В разделе описана реализация программного средства, включая создание ключевых классов и методов. Реализованы функции регистрации и авторизации пользователей, управления битами (добавление, редактирование, просмотр и удаление), а также система оценок битов. Взаимодействие с базой данных осуществляется через хранимые процедуры. Каждый функционал представлен отдельным классом, что обеспечивает разделение логики и упрощает поддержку системы.

1. Тестирование, проверка работоспособности и анализ полученных результатов
2. Тестирование авторизации и регистрации

В момент авторизации и регистрации возможна ситуация, когда пользователь вводит некорректные данные, например, неверный пароль, незарегистрированный логин. Такие исключения обрабатываются программным средством с помощью показа пользователю всплывающих сообщений с текстом ошибки. Примеры обработки разных видов исключительных ситуаций представлены на рисунках 5.1 – 5.3.

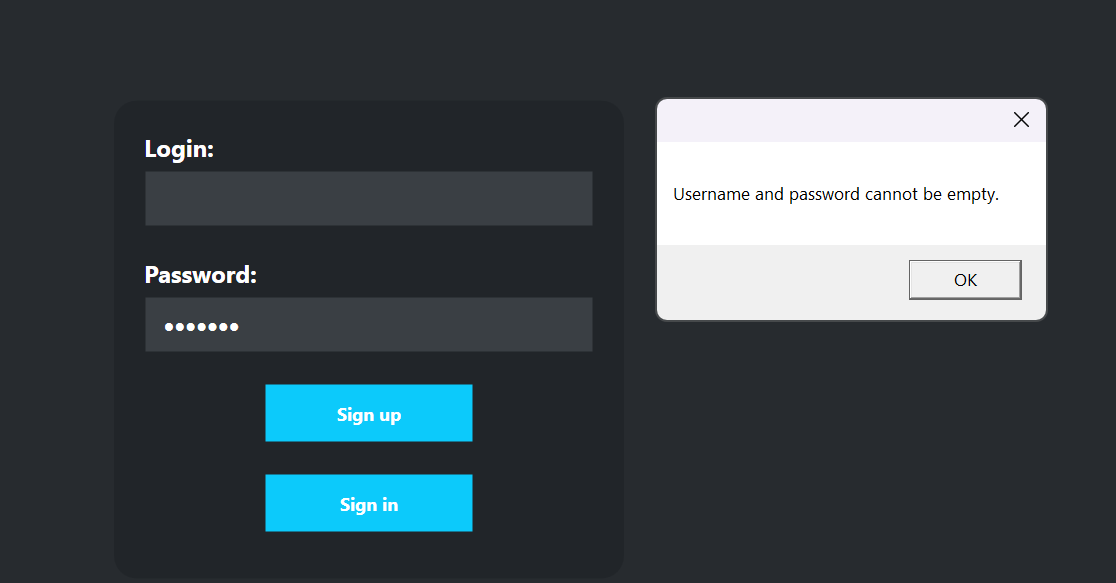


Рис 5.1 – Обработка пустого поля логина

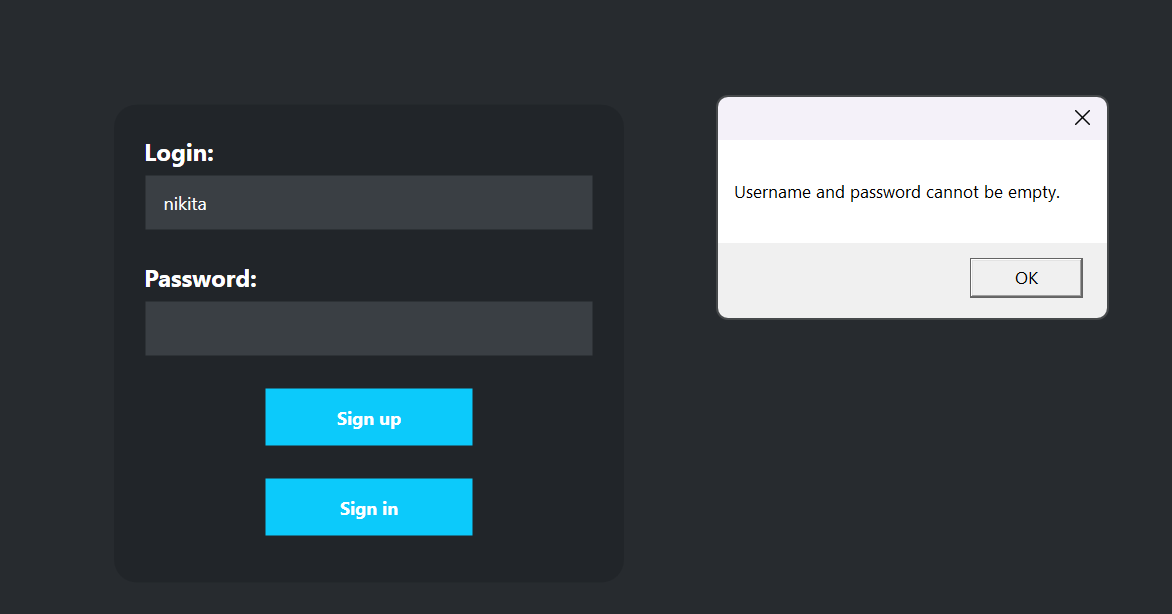


Рис 5.2 – Обработка пустого поля пароля

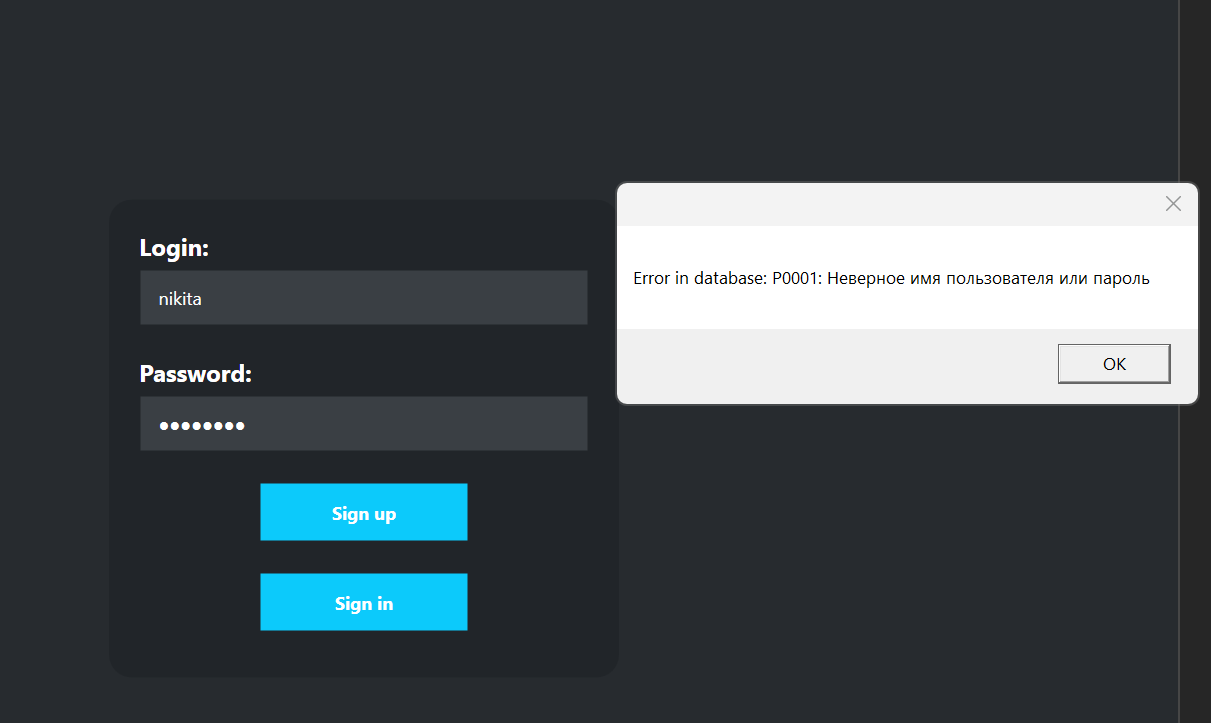


Рис 5.3 – Обработка неверного пароля или логина

Функционал авторизации и регистрации в программном средстве работает корректно и надежно. Система учитывает возможные ошибки, такие как ввод некорректных данных, незарегистрированный логин или неверный пароль, и своевременно обрабатывает их.

1. Тестирование добавления бита

Во время заполнения пользователем формы добавления бита возможны следующие исключительные ситуации.

Незаполненные обязательные поля. При отправке формы с пустыми полями, такими как Title, Description или Type ID, должно выводиться сообщение об ошибке, предупреждающее пользователя о необходимости заполнения этих полей.

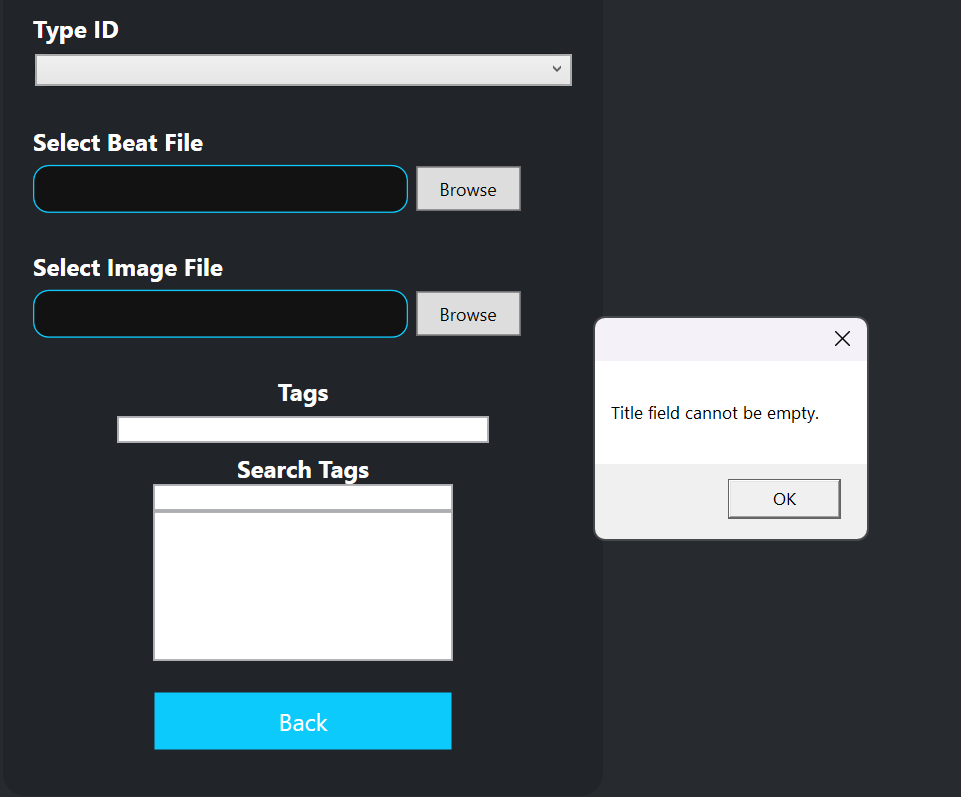


Рис 5.4 – Использование пустого заголовок

1. Тестирование пагинации

В случае, когда заканчиваются биты для вывода на следующей странице, должно выводится уведомление об этом.

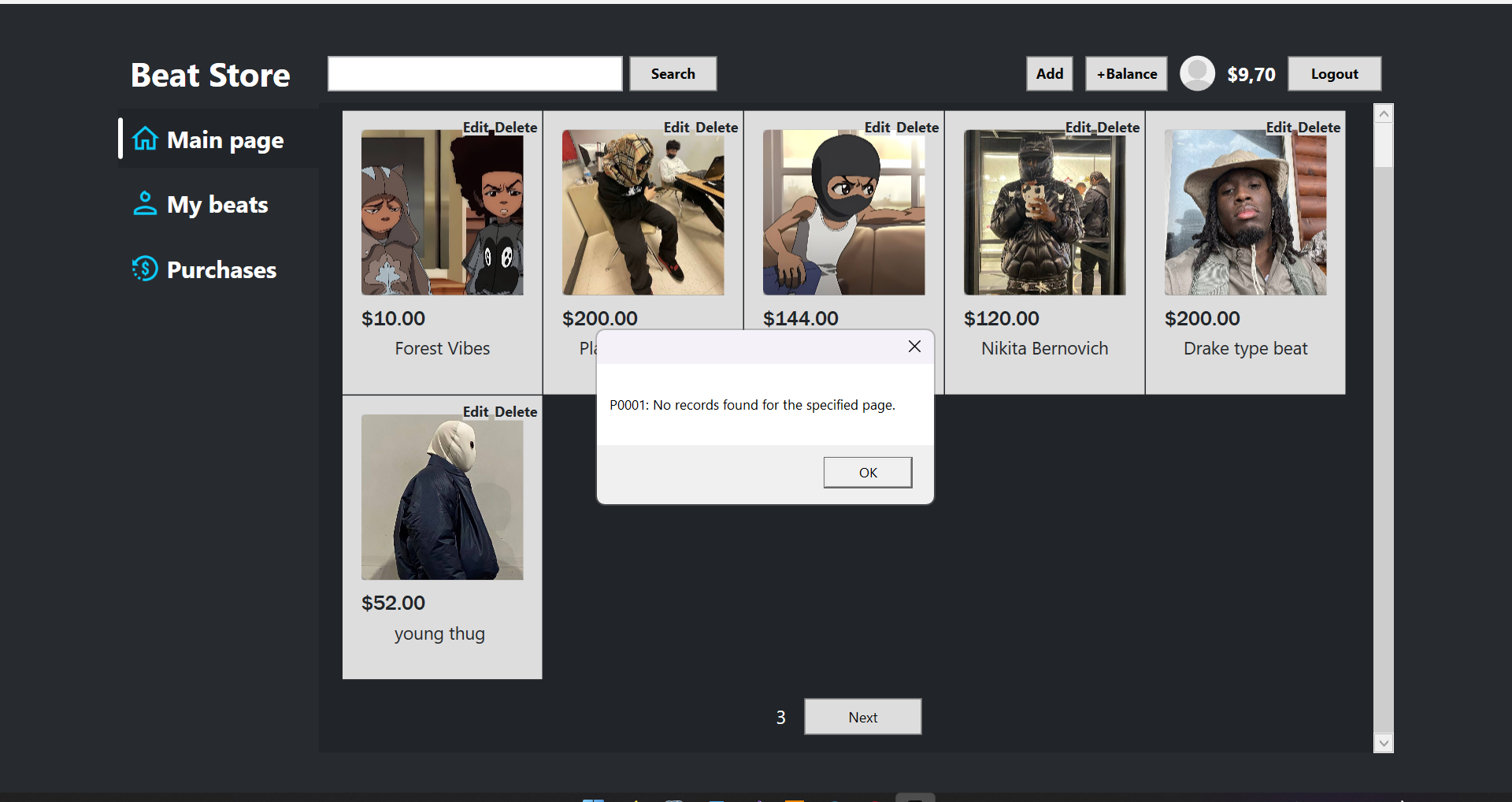


Рис 5.5 – Отсутствие битов на следующей странице

В разделе описаны результаты тестирования ключевых функциональных возможностей программного средства. Тестирование авторизации и регистрации показало, что система корректно обрабатывает ошибки, такие как пустые поля или неверные данные для логина и пароля, с помощью всплывающих сообщений. Тестирование функционала добавления бита подтвердило, что программа верно уведомляет пользователя о незаполненных обязательных полях. Также было протестировано поведение пагинации: при отсутствии битов на следующей странице выводится соответствующее уведомление. В целом, все компоненты функционируют стабильно, корректно обрабатывая возможные исключения.

1. Руководство по установке и использованию

При запуске программного средства пользователь попадает на страницу авторизации, содержащая формы входа и регистрации.

Стартовая страница позволяет зарегистрированному пользователю совершить вход в аккаунт, а не зарегистрированному – зарегистрироваться в системе, после чего пользователь может совершить вход в систему.

По умолчанию новому зарегистрированному клиенту присваивается роль «Пользователь». Роль «Администратор» при регистрации не присваивается. Окно авторизации представлено на рисунке 6.1.

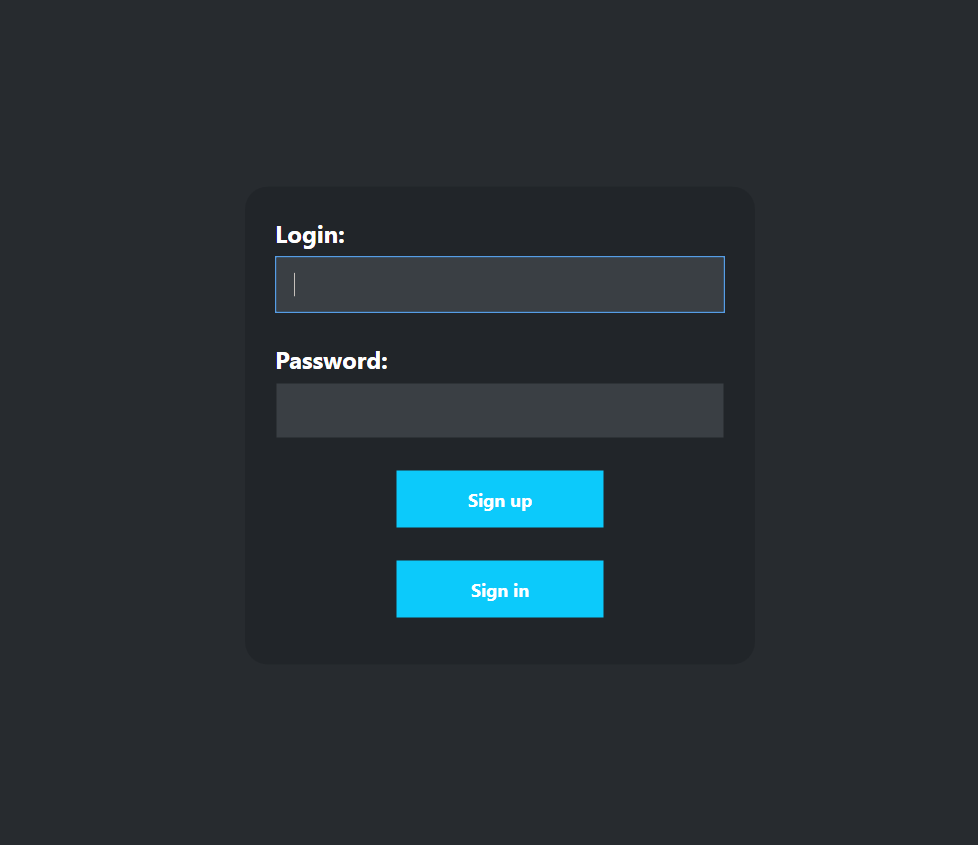


Рисунок 6.1 – Страница авторизации

После выполнения авторизации пользователь перенаправляется на основную страницу программного средства.

Пользователю доступен просмотр всех битов. Для просмотра подробной информации о бите для дальнейшей покупки необходимо нажать на бит, а после «Purchase», если понравился бит, то можно выбрать оценку бита.

Также пользователь имеет возможность выполнять поиск. Для этого он должен ввести запрос, и нажать на кнопку поиска.

Основная страница изображена на рисунке 6.3.

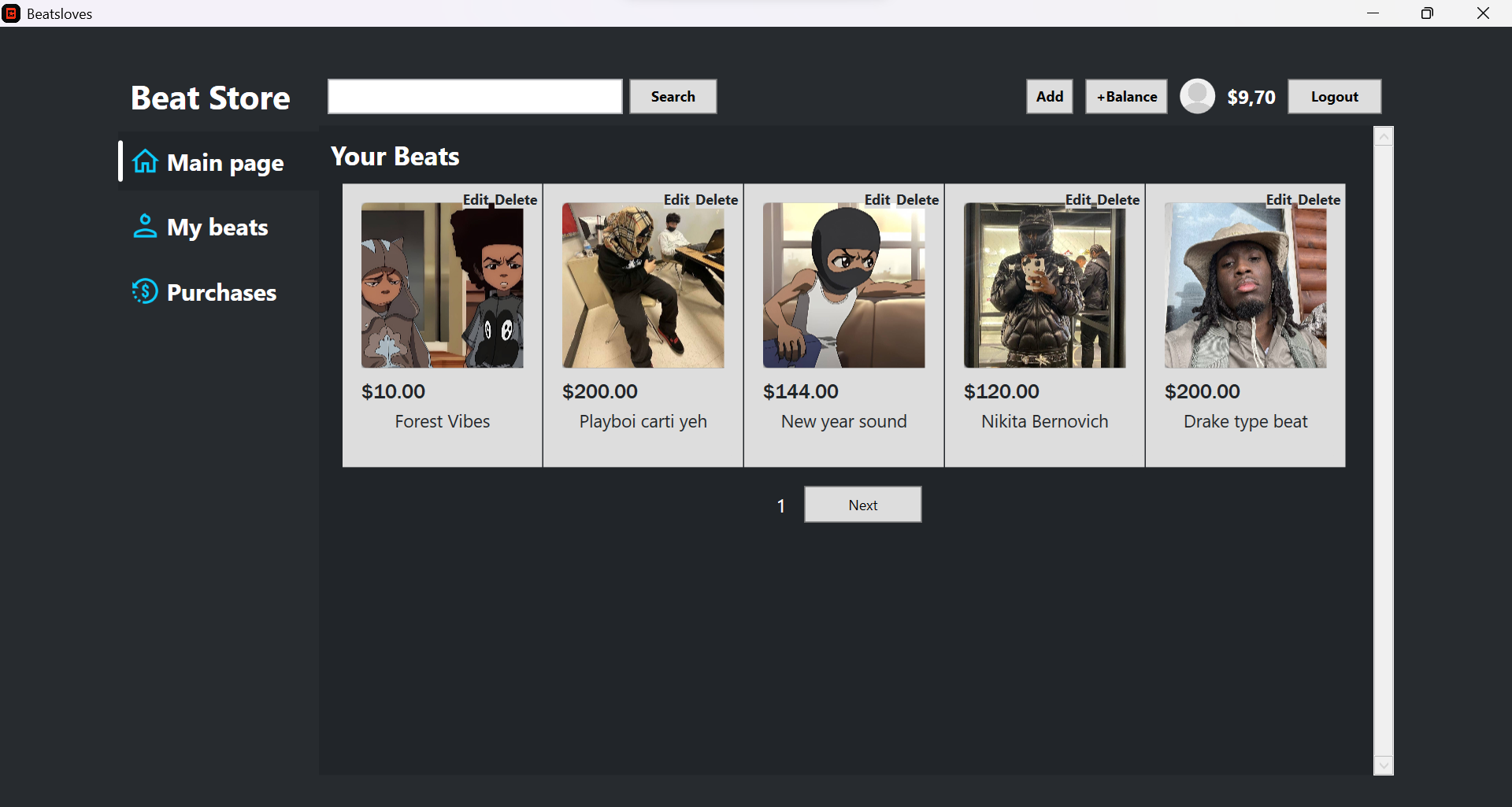


Рисунок 6.2 – Основная страница

При нажатии на кнопку «My beats» пользователь перенаправляется на страницу своих битов. При нажатии на кнопку «Purchases» пользователь перенаправляется на страницу приобретенных битов. При нажатии на бит открывается подробная информация о бите. На этой странице можно приобрести или оценить бит, а так же прослушать его. Страницы проиллюстрированы на рисунках 6.3 - 6.4.

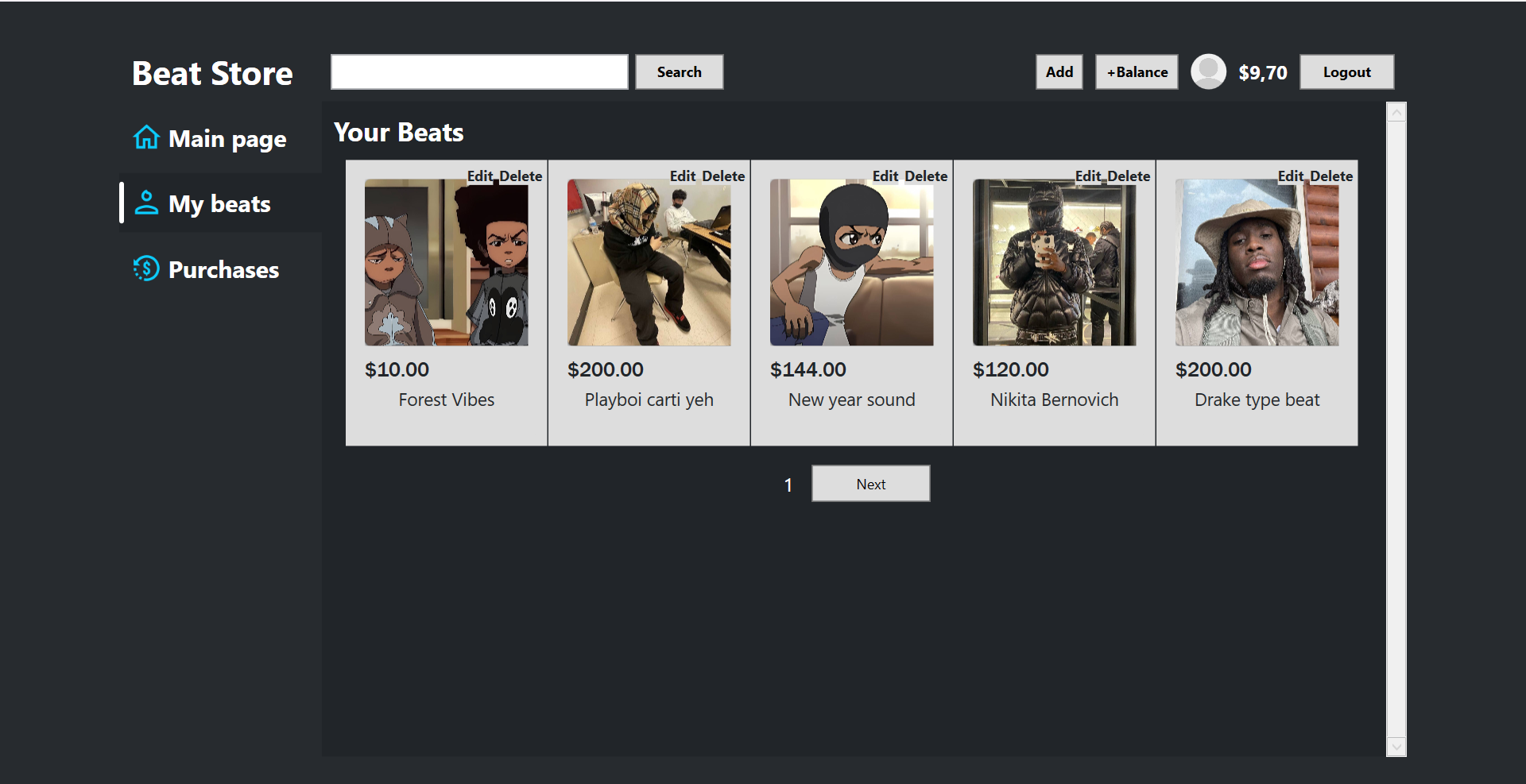


Рисунок 6.3 – Страница с моими битами

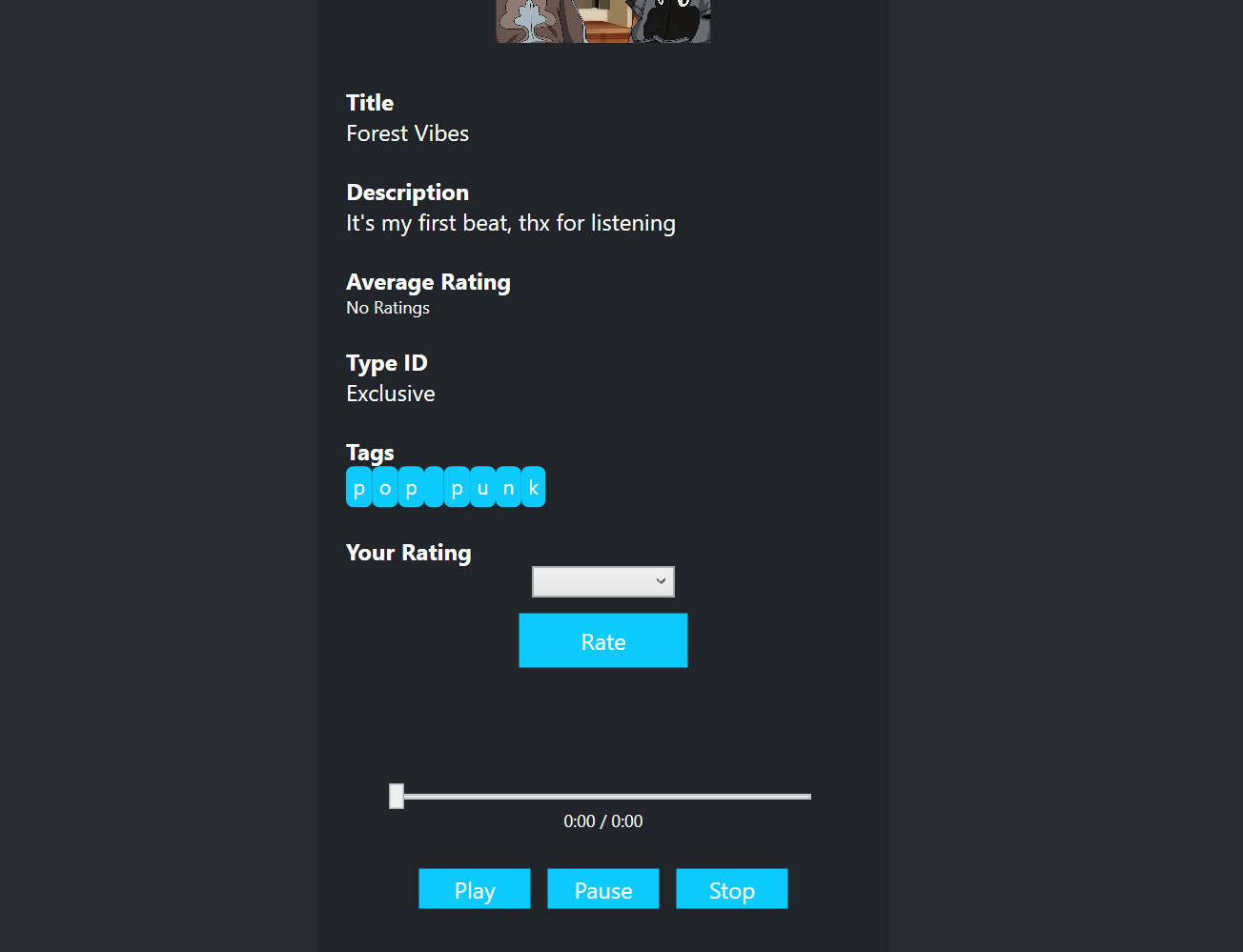


Рисунок 6.4 – Страница с подробной информацией о бите

Так же реализован поиск и пополнение баланса на главной странице, помимо этого, где нажать на изображение в верхнем правом углу, откроется профиль где показываются биты, которые у вас купили, и где можно поменять свое профильное изображение.

Модальное окно для пополнение баланса и окно профиля изображены на рисунках 6.5-6.6.

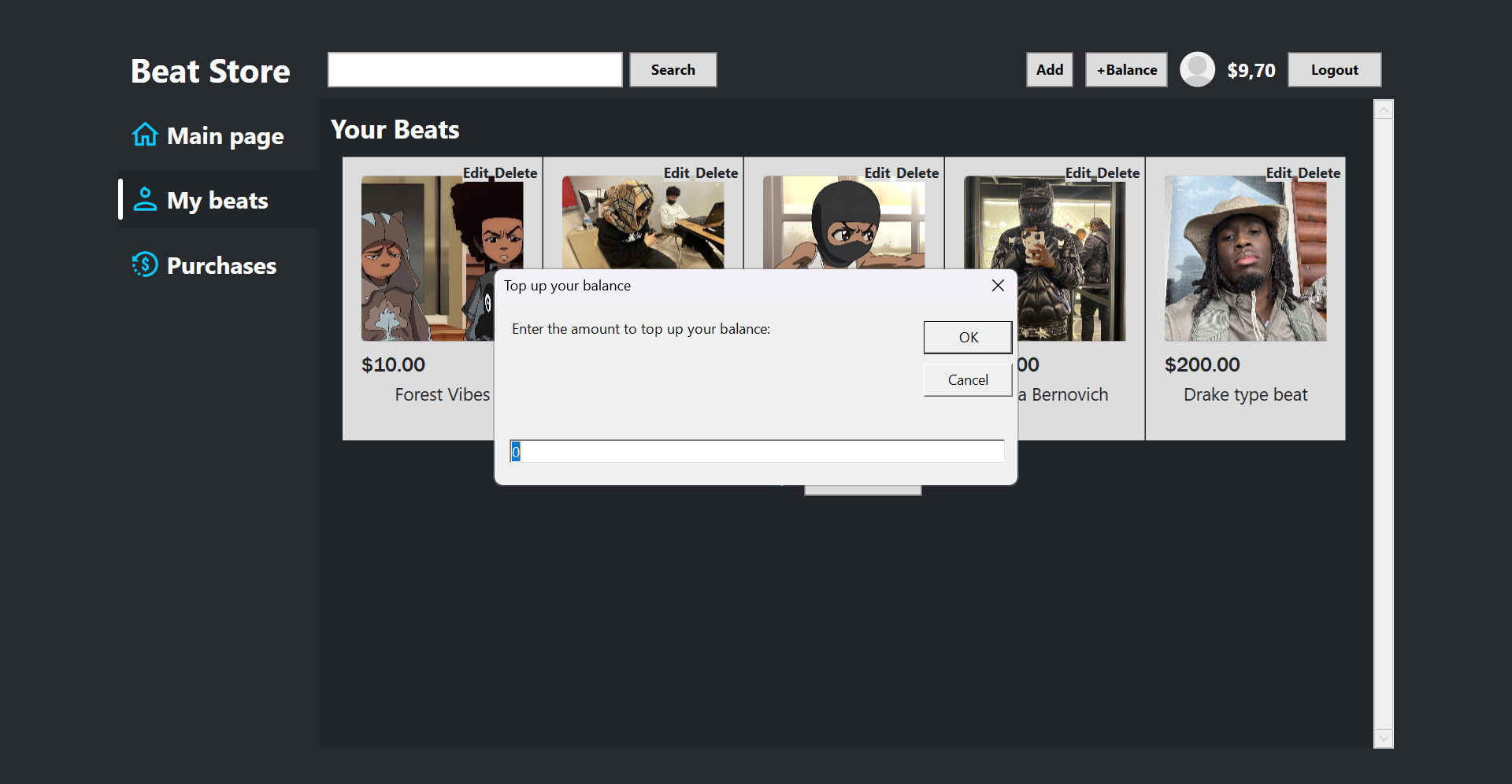


Рисунок 6.5 – Модальное окно для пополнение баланса

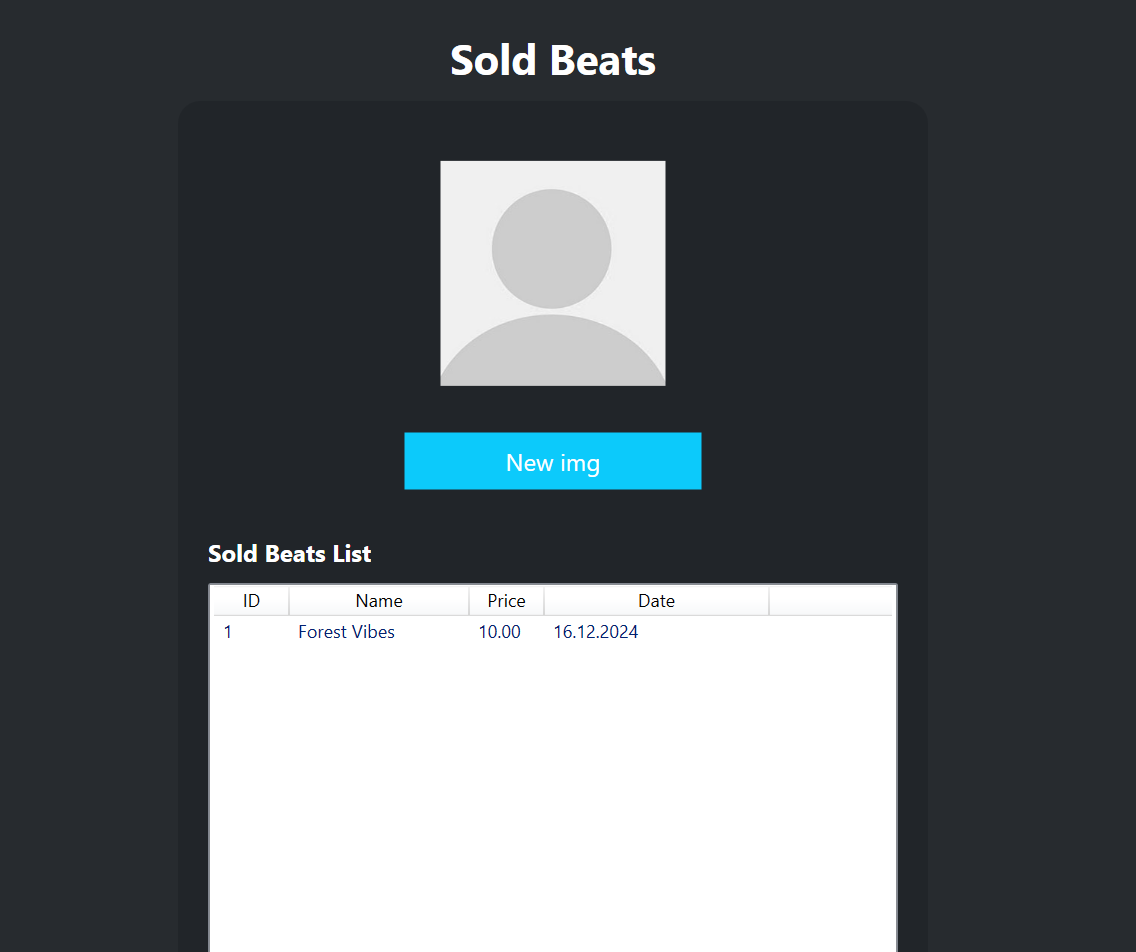


Рисунок 6.6 – Окно профиля

Руководство демонстрирует удобство использования программного средства и охватывает основные функции: авторизацию, регистрацию, просмотр и покупку битов, а также управление профилем. Реализованы простые и понятные интерфейсы для выполнения поиска, оценки, пополнения баланса и управления личными битами. Описанная последовательность действий и визуальное оформление страниц подтверждают интуитивность и функциональность пользовательского взаимодействия с программой.

# **Заключение**

В ходе выполнения курсового проекта было разработано программное средство на языка C# с использованием технологий ADO.NET, WPF.

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

* создана база данных;
* реализована возможность пользователю зарегистрироваться или войти в существующую учетную запись;
* также добавлена возможность авторизированным пользователям просмотра информации о битах;
* возможность оценки битов;
* реализована функция поиска битов.
* продажа, покупка битов.
* добавление, изменение, удаление битов.
* так же реализовано прослушивание битов, установка профильного изображения, просмотр проданных и приобретенных битов.
* добавлено пополнение баланса.

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. Microsoft Visual Studio [Электронный ресурс] – https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio – Дата доступа 23.10.2024
2. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 14.11.2024
3. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
4. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 28.11.2024
5. Работа с ADO.NET https://metanit.com/sharp/adonet/– Дата доступа 03.12.2024
6. Блинова, Е.А. Курс лекций по Базам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.

# **ПРИЛОЖЕНИЕ А**

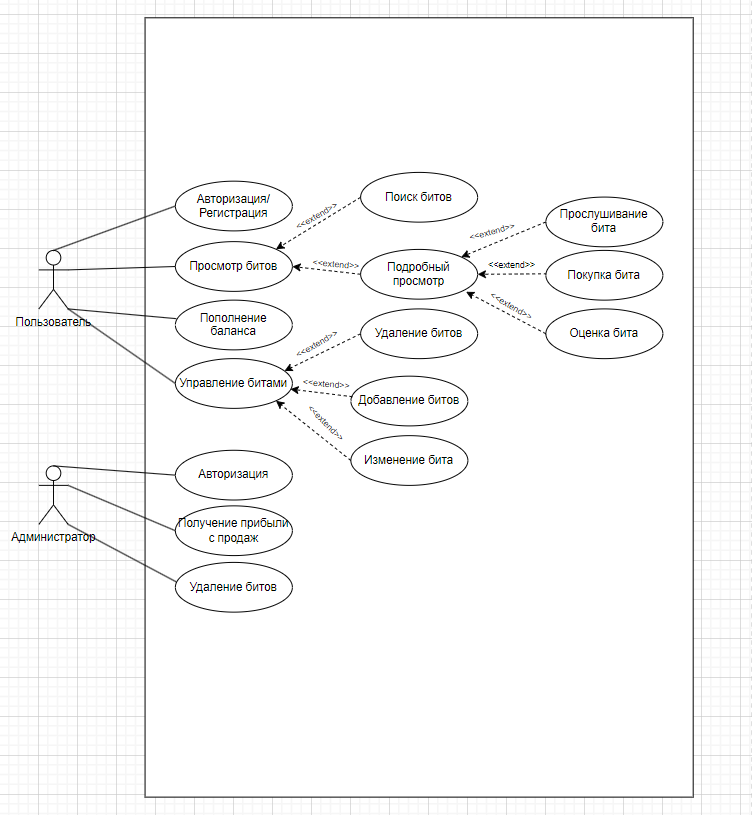
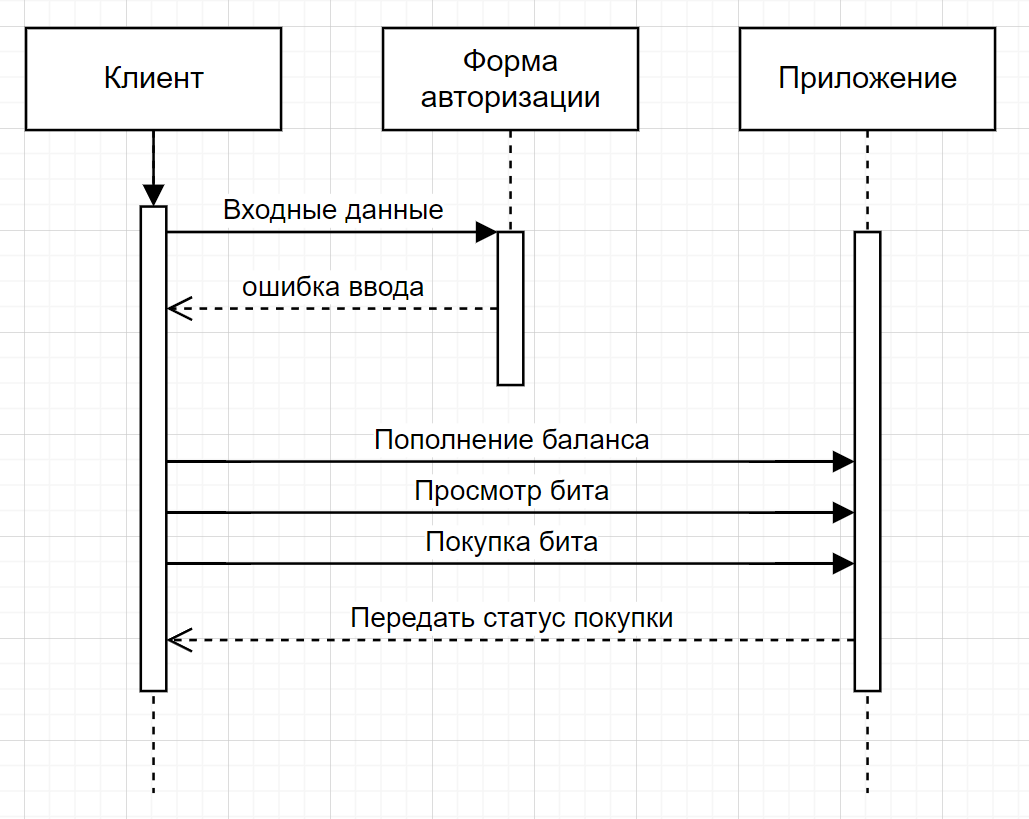


Рисунок 1 – Диаграмма использования

# **ПРИЛОЖЕНИЕ Б**

 Рисунок 1 – Диаграмма последовательности

# **ПРИЛОЖЕНИЕ В**

Листинг класса LoginViewModel

|  |
| --- |
| using beatsloves.Commands;  using beatsloves.Views;  using System.Windows.Input;  using beatsloves.Navigation;  using System;  using System.IO;  using Npgsql;  using System.Windows;  using System.Windows.Media.Imaging;  using beatsloves.Models;  using System.Text.RegularExpressions;  using System.Configuration;  namespace beatsloves.ViewModels  {  public class LoginViewModel : BaseViewModel  {  public ICommand RegisterCommand { get; }  public ICommand LoginCommand { get; }  private readonly NavigationService \_navigationService;  public string Username { get; set; }  public string Password { get; set; }  public LoginViewModel(NavigationService navigationService)  {  \_navigationService = navigationService;  RegisterCommand = new RelayCommand(OnRegister);  LoginCommand = new RelayCommand(OnLogin);  }  private void OnRegister(object obj)  {  if (string.IsNullOrWhiteSpace(Username) || string.IsNullOrWhiteSpace(Password))  {  MessageBox.Show("Username and password cannot be empty.");  return;  }  if (Username.Length < 4 || Password.Length < 8)  {  MessageBox.Show("Username and password must be at least 4, 8 characters long.");  return;  }  if (!Regex.IsMatch(Password, @"[A-Za-z]") || !Regex.IsMatch(Password, @"\d"))  {  MessageBox.Show("Password must contain at least one letter and one digit.");  return;  }  try  {  RegisterUser(Username, Password);  }  catch (Exception ex)  {  MessageBox.Show("Registration error: " + ex.Message);  }  }  private void RegisterUser(string username, string password)  {  if (string.IsNullOrWhiteSpace(Username) || string.IsNullOrWhiteSpace(Password))  {  MessageBox.Show("Username and password cannot be empty.");  return;  }  string connectionString = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;  using (var conn = new NpgsqlConnection(connectionString))  {  conn.Open();  using (var cmd = new NpgsqlCommand("SELECT register\_user(@username, @password)", conn))  {  cmd.Parameters.AddWithValue("@username", username);  cmd.Parameters.AddWithValue("@password", password);  cmd.ExecuteNonQuery();  MessageBox.Show("Success!");  }  }  }  private void OnLogin(object obj)  {  if (string.IsNullOrWhiteSpace(Username) || string.IsNullOrWhiteSpace(Password))  {  MessageBox.Show("Username and password cannot be empty.");  return;  }  try  {  using (var conn = new NpgsqlConnection(ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString))  {  conn.Open();  using (var cmd = new NpgsqlCommand("SELECT \* FROM authenticate\_user(@username, @password)", conn))  {  cmd.Parameters.AddWithValue("@username", Username);  cmd.Parameters.AddWithValue("@password", Password);  var reader = cmd.ExecuteReader();  if (reader.Read())  {  Session.UserID = Convert.ToInt32(reader["UserID"]);  Session.RoleID = Convert.ToInt32(reader["RoleID"]);  Session.Balance = Convert.ToDecimal(reader["Balance"]);  byte[] userImage = reader["UserImage"] as byte[];  if (userImage != null && userImage.Length > 0)  {  using (var stream = new MemoryStream(userImage))  {  Session.imageSource = new BitmapImage();  Session.imageSource.BeginInit();  Session.imageSource.StreamSource = stream;  Session.imageSource.CacheOption = BitmapCacheOption.OnLoad;  Session.imageSource.EndInit();  }  }  else  {  try  {  string placeholderPath = @"..\..\..\Images\placeholder.jpg";  using (var stream = new FileStream(placeholderPath, FileMode.Open, FileAccess.Read))  {  Session.imageSource = new BitmapImage();  Session.imageSource.BeginInit();  Session.imageSource.StreamSource = stream;  Session.imageSource.CacheOption = BitmapCacheOption.OnLoad;  Session.imageSource.EndInit();  }    }  catch (Exception ex)  {  MessageBox.Show($"Error: {ex.Message}");  }  }    \_navigationService.NavigateTo(new MainPage(\_navigationService));  }  else  {  MessageBox.Show("Incorrect login or password");  }  }  }  }  catch (Exception ex)  {  MessageBox.Show("Error in database: " + ex.Message);  }  }  }  } |

Листинги методов SearchBeats и LoadBeats

|  |
| --- |
| private void SearchBeats(object obj)  {    try  {  string connectionString = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;  using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))  {  connection.Open();  using (NpgsqlCommand cmd = new NpgsqlCommand("SELECT \* FROM SearchBeats(@searchQuery, @limitPage, @offsetPage)", connection))  {  cmd.Parameters.AddWithValue("@searchQuery", string.IsNullOrEmpty(SearchQuery) ? (object)DBNull.Value : SearchQuery);  cmd.Parameters.AddWithValue("@limitPage", 5);  cmd.Parameters.AddWithValue("@offsetPage", CurrentPage);  using (NpgsqlDataReader reader = cmd.ExecuteReader())  {  while (reader.Read())  {  var beat = new Beat  {  Id = (int)reader["beat\_id"],  Title = reader["title"].ToString(),  Price = (decimal)reader["price"],  Tags = reader["tags"].ToString(),  Beatimg = reader["beat\_img"].ToString(),  Typestring = reader["type\_name"].ToString(),  Description = reader["description"].ToString(),  Beatway = reader["beat\_way"].ToString(),  UserId = (int)reader["user\_id"]  };  Beats.Add(beat);  }  }  }  }  }  catch (NpgsqlException ex)  {  MessageBox.Show(ex.Message);  if (CurrentPage > 1) { CurrentPage--; }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  if (CurrentPage > 1) { CurrentPage--; }  }  }  private void LoadBeats(object obj)  {  try  {  string connectionString = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;  using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))  {  connection.Open();  using (NpgsqlCommand cmd = new NpgsqlCommand("SELECT \* FROM GetBeatsWithTags(@limitPage, @offsetPage)", connection))  {  cmd.Parameters.AddWithValue("@limitPage", 5);  cmd.Parameters.AddWithValue("@offsetPage", CurrentPage);  using (NpgsqlDataReader reader = cmd.ExecuteReader())  {  while (reader.Read())  {  Beat beat = new Beat  {  Id = (int)reader["beat\_id"],  Title = reader["title"].ToString(),  Price = (decimal)reader["price"],  Tags = reader["tags"].ToString(),  Beatimg = reader["beat\_img"].ToString(),  Typestring = reader["type\_name"].ToString(),  Description = reader["description"].ToString(),  Beatway = reader["beat\_way"].ToString(),  UserId = (int)reader["user\_id"]  };  Beats.Add(beat);  }  }  }  }    }  catch (NpgsqlException ex)  {  MessageBox.Show(ex.Message);  if (CurrentPage > 1) { CurrentPage--; }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  if (CurrentPage>1) { CurrentPage--; }  }  } |

Листинг метода AddOrEditBeat

|  |
| --- |
| private void AddOrEditBeat(object obj)  {  if (string.IsNullOrWhiteSpace(Title))  {  MessageBox.Show("Title field cannot be empty.");  return;  }  if (string.IsNullOrWhiteSpace(Description))  {  MessageBox.Show("Description field cannot be empty.");  return;  }  if (string.IsNullOrWhiteSpace(Beatway))  {  MessageBox.Show("Beatway field cannot be empty.");  return;  }  if (SelectedLicenseType == null || SelectedLicenseType.Id <= 0)  {  MessageBox.Show("Please select a valid license type.");  return;  }  if (string.IsNullOrWhiteSpace(Beatimg))  {  MessageBox.Show("Beat image path cannot be empty.");  return;  }  if (Price <= 0)  {  MessageBox.Show("Price cannot be empty or zero.");  return;  }  string connectionString = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;  string procedureName = \_isEditMode ? "UpdateBeat" : "AddBeat";  try  {  using (var connection = new NpgsqlConnection(connectionString))  {  connection.Open();  using (var cmd = new NpgsqlCommand(procedureName, connection))  {  cmd.CommandType = CommandType.StoredProcedure;  if (\_isEditMode)  {  cmd.Parameters.AddWithValue("beat\_id", \_beatId);  }  cmd.Parameters.AddWithValue("title", Title);  cmd.Parameters.AddWithValue("description", Description);  cmd.Parameters.AddWithValue("price", Price);  cmd.Parameters.AddWithValue("typeid", SelectedLicenseType.Id);  cmd.Parameters.AddWithValue("beatway", Beatway);  cmd.Parameters.AddWithValue("beatimg", Beatimg);  cmd.Parameters.AddWithValue("tags", Tags);  cmd.Parameters.AddWithValue("user\_id", Session.UserID);  var status = cmd.Parameters.Add("status", NpgsqlTypes.NpgsqlDbType.Boolean);  status.Direction = ParameterDirection.Output;  var message = cmd.Parameters.Add("message", NpgsqlTypes.NpgsqlDbType.Text);  message.Direction = ParameterDirection.Output;  cmd.ExecuteNonQuery();  bool operationStatus = (bool)status.Value;  string operationMessage = message.Value.ToString();  if (operationStatus)  {  MessageBox.Show(\_isEditMode ? "Beat successfully updated." : "Beat successfully added.");  }  else  {  MessageBox.Show("Error: " + operationMessage);  }  }  }  }  catch (Exception ex)  {  MessageBox.Show("Error occurred while adding/updating the beat: " + ex.Message);  }  }  } |

Листинг метода ExecuteRateBeat

|  |
| --- |
| private void ExecuteRateBeat(object obj)  {  try  {    using (var conn = new NpgsqlConnection(ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString))  {  conn.Open();  using (var cmd = new NpgsqlCommand("SELECT SetBeatRating(@p\_BeatID, @p\_UserID, @p\_RatingValue)", conn))  {  cmd.Parameters.AddWithValue("@p\_BeatID", SelectedBeat.Id);  cmd.Parameters.AddWithValue("@p\_UserID", Session.UserID);  cmd.Parameters.AddWithValue("@p\_RatingValue", UserRating);  cmd.ExecuteNonQuery();    }  }    ExecuteLoadAverageRating();  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при оценке: {ex.Message}");  }  } |