

Министерство образования Иркутской области

Государственное бюджетное профессиональное образовательное учреждение

Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

ДП.09.02.07-3.25.212.17. ПЗ

УТВЕРЖДАЮ

Зам. директора по УР, к.т.н.

_____ Е.А. Коробкова

ИНТЕРНЕТ-МАГАЗИН КОМПЬЮТЕРНЫХ ИГР

Норм контролёр:

(подпись, дата)

(Е.С. Кудрявцева)

Консультант по
экономической части:

(подпись, дата)

(А.П. Юргина)

Руководитель:

(подпись, дата)

(С.А. Удальцов)

Студент:

(подпись, дата)

(И.Г. Пермяков)

Иркутск 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Предпроектное исследование	5
1.1 Описание предметной области	5
1.2 Анализ инструментов, используемых в разработке программного продукта	6
2 Техническое задание на разработку программного продукта.....	16
3 Проектирование программного продукта	17
3.1 Архитектура программного продукта.....	17
3.2 Функциональное и структурное проектирование.....	18
3.3 Проектирование базы данных.....	22
3.4 Проектирование пользовательского интерфейса программного продукта ..	26
4 Реализация программного продукта	38
5 Отладка и тестирование программного продукта	45
6 Руководство пользователя программного продукта	49
7 Экономическая часть	57
7.1 Расчет трудоемкости разработки программного обеспечения.....	57
7.2 Затраты на оплату.....	58
7.3 Затраты на амортизацию оборудования	59
7.4 Расчет затрат на электроэнергию	60
7.5 Затраты на материалы, израсходованные при проведении разработки и прочие.....	61
7.6 Расчет затрат на разработку программного обеспечения	61
7.7 Расчет цены.....	62
ЗАКЛЮЧЕНИЕ	63
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	65
Приложение А Техническое задание	67
Приложение Б – Листинг кода шаблона страниц	71
Приложение В – Листинг кода контроллера поиска	75

					ДП. 09.02.07-3.25.212.17. ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Пермяков И.Г.			ИНТЕРНЕТ-МАГАЗИН КОМПЬЮТЕРНЫХ ИГР пояснительная записка	Лит.	Лист	Листов
Провер.		Удальцов С.А.					2	75
Реценз.						ГБПОУИО «ИАТ» ВЕБ-21-2		
Н. Контр.		Кудрявцева Е.С.						
Утверд.		Коробкова Е.А.						

ВВЕДЕНИЕ

Интернет-магазин компьютерных игр представляет собой специализированную платформу, предназначенную для приобретения цифровых копий игр через интернет. Такие магазины предоставляют пользователям доступ к обширному ассортименту игр, охватывающему различные жанры, платформы и ценовые категории. Пользователи могут выбирать игры, основываясь на своих предпочтениях. Интернет-магазины компьютерных игр стали неотъемлемой частью современной игровой индустрии, обеспечивая высокий уровень удобства, скорости и безопасности при совершении покупок.

В современном мире компьютерные игры стали популярным видом развлечения. С развитием цифровых технологий и увеличением доступности интернета рынок компьютерных игр продолжает активно расти. Это создает спрос на удобные и функциональные платформы для покупки и продажи игр. Интернет-магазины компьютерных игр предоставляют пользователям возможность быстро и безопасно приобретать лицензионные продукты, а разработчикам и издателям — эффективно распространять свои игры. Однако, несмотря на высокую конкуренцию, многие существующие решения имеют ограниченный функционал или недостаточно удобны для пользователей. Это делает разработку нового интернет-магазина компьютерных игр актуальной задачей, которая может удовлетворить потребности как пользователей, так и бизнеса.

Целью дипломного проекта является разработка интернет-магазина компьютерных игр с широким функционалом, включающим удобный каталог-магазин, систему покупки игр, система уровня, библиотека хранения своих купленных игр и профиль пользователя.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) провести анализ предметной области;

					ДП.09.02.07-3.25.212.17. ПЗ	Лист
Изм.	Лис	№ докум.	Подпись	Дата		3

- 2) разработать техническое задание;
- 3) провести анализ программных средств разработки;
- 4) разработать и протестировать основные модули интернет-магазина;
- 5) разработать программную документацию.

Необходимыми методами исследования для решения задач являются:

- 1) формирование требований к интернет-магазину на основе анализа предметной области;
- 2) сравнительный анализ современных технологий и инструментов для веб-разработки;
- 3) применение инструментов автоматизированного тестирования;
- 4) создание технической документации, включающей описание пользовательских инструкций.

Интернет-магазин компьютерных игр будет предоставлять пользователям информацию о товарах, ценах и других важных данных, необходимых для принятия решений о покупке. Магазин будет включать в себя такие ключевые компоненты, как каталог игр, управление аккаунтом. Это позволит не только удовлетворить потребности пользователей, но и повысить эффективность работы администраторов.

					ДП.09.02.07-3.25.212.17. ПЗ	Лист
Изм.	Лис	№ докум.	Подпись	Дата		4

1 Предпроектное исследование

1.1 Описание предметной области

Интернет-магазин компьютерных игр предоставляет пользователям возможность приобретать лицензионный продукт, без опасения того, что можно словить вирус. Магазин предлагает широкий ассортимент игр различных жанров. Ключевыми объектами взаимодействия в интернет-магазине являются:

- пользователи (клиенты): основные участники системы, которые осуществляют покупку игр. Пользователи взаимодействуют с каталогом игр, добавляют товары в корзину, оформляют заказы и используют личный кабинет для управления своими покупками;

- администраторы: сотрудники, которые управляют контентом магазина. Они добавляют новые игры в каталог, обновляют информацию о существующих играх, настраивают цены;

- игры: основной товар магазина. Каждая игра имеет название, описание, цену, системные требования, жанр и другие атрибуты, которые помогают пользователям сделать выбор;

- техническая поддержка: обеспечивает помощь пользователям в решении проблем, связанных с покупкой и активацией игр. Пользователи могут обращаться в техническую поддержку через форму обратной связи, а администраторы оперативно реагируют на запросы.

В интернет-магазине компьютерных игр происходят такие процессы как:

- поиск и выбор игры: пользователь просматривает каталог игр, используя сортировку или фильтры по жанрам и ценам. После выбора пользователь добавляет игру в корзину;

- оформление заказа: пользователь оформляет заказ путем указания способа оплаты и подтверждения покупки, после чего получает ключ для активации игры;

					ДП.09.02.07-3.25.212.17. ПЗ	Лист
Изм.	Лис	№ докум.	Подпись	Дата		5

– управление контентом: администраторы добавляют новые игры в каталог, обновляют информацию о существующих играх, настраивают цены и скидки;

– техническая поддержка: пользователи могут обратиться в техническую поддержку через форму обратной связи, чтобы получить помощь в решении проблем, связанных с покупкой или активацией игр. Администраторы обрабатывают запросы и предоставляют необходимую помощь.

Все объекты связаны между собой через процессы:

– пользователи взаимодействуют с каталогом игр, оформляют заказы и используют личный кабинет для управления своими покупками;

– администраторы управляют контентом магазина, обрабатывают заказы и обеспечивают техническую поддержку;

Автоматизация интернет-магазина компьютерных игр является актуальной задачей, так как она охватывает ключевые процессы, такие как оформление заказов, управление контентом и техническая поддержка. Это позволяет:

1) упростить и ускорить процесс покупки игр для пользователей: автоматизация оформления заказов делает процесс мгновенным и удобным;

Таким образом, автоматизация интернет-магазина компьютерных игр является важным элементом, который позволяет не только удовлетворить потребности пользователей, но и оптимизировать внутренние процессы. Автоматизация позволяет минимизировать ручной труд, ускорить процессы оформления заказов и управления контентом.

1.2 Анализ инструментов, используемых в разработке программного продукта

Разработка интернет-магазина компьютерных игр требует использования современных инструментов и технологий, которые обеспечат высокую

производительность, надежность и удобство разработки. Для создания интернет-магазина были определены следующие инструменты:

- Draw.io: используется для создания диаграмм, таких как контекстная и диаграмма декомпозиций.
- Figma – это мощный инструмент для проектирования пользовательских интерфейсов, который позволяет создавать прототипы и макеты с высокой точностью. Использовался для создания прототипов интернет-магазина и его макетов, что позволило заранее определить визуальное оформление и удобство взаимодействия.
- MySQL – это популярная система управления базами данных, которая обеспечивает надежное хранение и обработку данных. Использовался для хранения информации.
- PhpMyAdmin – это инструмент для проектирования, администрирования и визуализации баз данных. Применялся для проектирования базы данных и создания ER-модели.
- HTML – это стандартный язык разметки для создания структуры веб-страниц. Использовался для создания структуры страниц интернет-магазина, включая заголовки, тексты, формы и другие элементы.
- CSS – это язык стилей, используемый для оформления веб-страниц. Применялся для стилизации страниц интернет-магазина, включая задание цветов, шрифтов, отступов и других визуальных параметров.
- JavaScript – это скриптовый язык программирования, используемый для создания интерактивности на страницах. Применялся для добавления динамики.
- PHP – это популярный язык программирования для разработки серверной части веб-приложений. Использовался в качестве языка для разработки серверной части, обеспечивая обработку данных на стороне сервера и взаимодействие с базой данных.

– Visual Studio Code – это мощная и легковесная среда разработки с поддержкой множества языков программирования и расширений. Применялся в качестве среды разработки для написания и отладки кода.

– Bootstrap – это популярный CSS-фреймворк, который предоставляет готовые компоненты и стили для создания адаптивных веб-интерфейсов. Использовался для упрощения и ускорения разработки интерфейса, что позволило быстро создавать современные и адаптивные страницы.

– Laravel – это современный PHP-фреймворк, который упрощает разработку сложных веб-приложений благодаря встроенным функциям для маршрутизации, работы с базой данных и аутентификации. Использовался в качестве фреймворка для разработки серверной части интернет-магазина, обеспечивая высокую производительность и удобство разработки.

– PHPUnit – это фреймворк для автоматизированного тестирования PHP-кода, который позволяет проверять корректность работы отдельных модулей и функций. Применялся для тестирования серверной части интернет-магазина, что повысило надежность и качество кода.

Информацию, содержащуюся в интернет-магазине, необходимо хранить, изменять, структурировать и использовать. Это реализуется благодаря базе данных. Были рассмотрены следующие варианты систем управления базами данных:

- 1) MySQL.
- 2) PostgreSQL.
- 3) SQLite.

MySQL – это одна из самых популярных реляционных систем управления базами данных с открытым исходным кодом. Она широко используется в веб-разработке благодаря своей простоте, производительности и надежности. Особенности являются высокая производительность при работе с большими объемами данных, широкая поддержка со стороны сообщества, обширная документация, высокая скорость выполнения запросов.

PostgreSQL – это объектно-реляционная система управления базами данных с открытым исходным кодом, которая поддерживает расширенные функции и стандарты SQL. Особенности PostgreSQL являются расширенные возможности для работы с JSON, XML и геопространственными данными, поддержка пользовательских типов данных и функций, активное сообщество и регулярные обновления, сложная настройка и администрирование.

SQLite – это встраиваемая реляционная система управления базами данных, которая хранит всю базу данных в одном файле. Идеально подходит для небольших проектов. Особенности SQLite: не требует установки и настройки сервера, занимает меньше места на дисковом пространстве, простота интеграции в приложения, ограниченная производительность при работе с большими объемами данных.

Для наглядности сравнения вариантов реализации базы данных была составлена таблица 1.

Таблица 1 – Сравнение средств реализации базы данных

Название	MySQL	PostgreSQL	SQLite
Простота использования	+	-	+
Высокая производительность	+	+	-
Большое количество типов данных	+	+	-
Популярность	+	-	+
Отказоустойчивость	-	+	-

Таким образом для интернет-магазина компьютерных игр была выбрана MySQL, так как она обеспечивает высокую производительность, простоту использования, поддерживает большое количество типов данных.

Для разработки серверной части интернет-магазина необходимо выбрать язык программирования, который обеспечит высокую производительность и простоту разработки. Были рассмотрены следующие варианты языков программирования:

- 1) PHP.
- 2) Python.
- 3) Node.js.

PHP – это серверный язык программирования, который изначально был создан для веб-разработки. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов. Особенности языка являются простота изучения и использования, высокая производительность, обширная документация.

Python – это высокоуровневый язык программирования общего назначения, который используется для веб-разработки, анализа данных, машинного обучения, автоматизации и многого другого. Особенности Python: простой и понятный синтаксис, который делает код легко читаемым, подходит для широкого круга задач, высокая производительность. Однако в серверной разработке язык используется только благодаря фреймворку. Недостатком также считается высокое потребление памяти.

Node.js – это среда выполнения JavaScript на стороне сервера, которая позволяет разрабатывать высокопроизводительные веб-приложения с использованием одного языка как для клиентской, так и для серверной части. Особенностью является то, что он основан на событийно-ориентированной архитектуре, что делает его идеальным для работы с реальным временем (например, чаты, стриминги), а также высокая производительность благодаря асинхронной модели выполнения.

Для наглядности сравнения языков программирования была составлена таблица 2.

Таблица 2 – Сравнение языков программирования

Название	PHP	Python	Node.js
Простота использования	+	+	-

Высокая производительность	+	-	+
Более активное сообщество	+	+	+
Объектно-ориентированные возможности	+	+	+

Таким образом для разработки интернет-магазина был выбран PHP, так как он обеспечивает высокую производительность и простоту использования.

Для разработки интернет-магазина компьютерных игр важно выбрать удобную и функциональную среду разработки, которая обеспечит высокую производительность и удобство работы. Были рассмотрены следующие варианты сред разработки:

- 1) Visual Studio Code.
- 2) PhpStorm.
- 3) Atom.

Visual Studio Code – это мощный редактор кода с открытым исходным кодом, разработанный Microsoft. Поддерживает множество языков программирования и фреймворков, включая PHP, JavaScript, Python и другие. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, автодополнение и средства для рефакторинга.

PhpStorm – это профессиональная среда для разработки на PHP, разработанная JetBrains. Она предоставляет мощные инструменты для работы с PHP, Laravel, Symfony и другими фреймворками. Особенностью PhpStorm является функционал, который пользователи могут расширить с помощью установки плагинов или написания собственных.

Atom – это бесплатный текстовый редактор с открытым исходным кодом, разработанный GitHub. Он поддерживает множество языков программирования и может быть расширен с помощью плагинов. Atom прост в использовании, однако менее производителен, чем другие среды.

Для наглядности сравнения сред разработки была составлена таблица 3.

Таблица 3 – Сравнение сред разработки

Название	Visual Studio Code	PhpStorm	Atom
Интеграция с Git	+	+	+
Подсказки по коду	+	+	+
Производительность	+	+	-
Распространяется бесплатно	+	-	+
Поддержка фреймворков	+	+	-

Таким образом в качестве среды разработки был выбран Visual Studio Code благодаря поддержке фреймворков, высокой производительности и возможности использовать среду бесплатно.

Для разработки пользовательского интерфейса интернет-магазина компьютерных игр важно выбрать инструмент, который обеспечит быстрое создание адаптивных и современных веб-страниц. Были рассмотрены следующие варианты CSS-фреймворков:

- 1) Bootstrap.
- 2) Tailwind CSS.

Bootstrap – это один из самых популярных CSS-фреймворков, предназначенный для быстрого создания адаптивного интерфейса. Содержит множество готовых шаблонов: карточки, формы, кнопки, модальные окна и другие компоненты веб-интерфейса. Для создания макетов используется адаптивная сетка, что позволяет интерфейсу корректно отображаться на всех устройствах.

Tailwind CSS – это ориентированный на утилиты CSS-фреймворк, который предоставляет низкоуровневые классы для создания интерфейса. В отличие от Bootstrap, Tailwind не предлагает готовых компонентов, а вместо этого позволяет создавать уникальные дизайны. Особенностью Tailwind является его поддержка адаптивности, а также гибкость, позволяющая создавать

уникальные дизайны. Однако у Tailwind CSS меньше документации и ресурсов по сравнению с Bootstrap.

Для наглядности сравнения вариантов реализации пользовательского интерфейса была составлена таблица 4.

Таблица 4 – Сравнение средств реализации пользовательского интерфейса

Название	Bootstrap	Tailwind CSS	Foundation
Простота использования	+	-	-
Поддержка сообщества	+	+	-
Адаптивность	+	+	+
Поддержка компонентов	+	-	+

Таким образом в качестве фреймворка для создания пользовательского интерфейса был выбран Bootstrap, так как он прост в использовании, имеет обширную документацию и набор готовых компонентов.

Для разработки серверной части интернет-магазина компьютерных игр важно выбрать фреймворк, который обеспечит высокую производительность, удобство разработки и поддержку современных технологий. Были рассмотрены следующие варианты фреймворков:

- 1) Laravel.
- 2) Symfony.
- 3) CodeIgniter.

Laravel – это современный PHP-фреймворк, который предоставляет мощные инструменты для веб-разработки. Он известен своей простотой и широким набором встроенных функций. Имеет встроенную поддержку регистрации, авторизации, входа и управления пользователями, гибкую систему маршрутов для обработки HTTP-запросов, а также управление структурой базы данных через код.

Symfony – это мощный и гибкий PHP-фреймворк, который используется для создания сложных веб-приложений. Он известен своей модульностью и

расширяемостью. Состоит из набора независимых компонентов, которые можно использовать отдельно. Содержит инструменты для работы с базой данных через объекты, а также гибкую систему маршрутов. В отличие от Laravel требует больше времени для настройки и изучения.

CodeIgniter – это легковесный PHP-фреймворк, который предоставляет базовые инструменты для создания веб-приложений. Он известен своей простотой и минимализмом. CodeIgniter предоставляет только базовые функции, что делает его легковесным и быстрым. Обладает простой системой маршрутов для обработки HTTP-запросов. Требуется меньше времени для настройки в отличие от Symfony. Отсутствует встроенная поддержка аутентификации, для нее требуется установка дополнительных библиотек.

Для наглядности сравнения вариантов реализации серверной части была составлена таблица 5.

Таблица 5 – Сравнение средств реализации серверной части

Название	Laravel	Symfony	CodeIgniter
Простота использования	+	-	+
Производительность	+	+	+
Поддержка аутентификации	+	+	-
Поддержка REST API	+	+	+
Поддержка сообщества	+	+	-

Таким образом, для разработки серверной части интернет-магазина компьютерных игр был выбран PHP-фреймворк Laravel, так как он обеспечивает высокую производительность, простоту использования, отличную поддержку сообщества и широкий набор встроенных функций.

Подводя итог, для разработки интернет-магазина компьютерных игр было решено использовать следующие средства:

- 1) для создания диаграмм использовались CASE-средства Draw.io;

- 2) для наглядного составления структуры базы данных использовался инструмент для визуального проектирования баз данных – MySQL Workbench;
- 3) для разработки дизайна интернет-магазина использовался онлайн-сервис для разработки дизайна и прототипа – Figma;
- 4) для разработки интерфейса использовался язык разметки HTML, язык стилей CSS, а также JavaScript;
- 5) в качестве фреймворка для разработки интерфейса использовался Bootstrap;
- 6) на этапе разработки программного продукта использовался язык программирования PHP;
- 7) в качестве фреймворка для разработки программного продукта использовался Laravel;
- 8) на этапе разработки программного продукта использовался редактор кода Visual Studio Code;
- 9) для структурирования, чтения, изменения и удаления информации использовалась система управления базами данных MySQL.

2 Техническое задание на разработку программного продукта

В начале разработки создавалось техническое задание, в котором указывались основные требования.

Для создания технического задания использовался стандарт ГОСТ 34.602-2020.

Согласно ГОСТ 34.602-2020 техническое задание должно включать следующие разделы:

- общие сведения;
- назначение и цели создания интернет-магазина;
- требования к интернет-магазину в целом;
- требования к документированию;
- состав и содержание работ по созданию интернет-магазина;

Техническое задание на разработку интернет-магазина представлено в приложении А.

					ДП.09.02.07-3.25.212.17. ПЗ	Лист
Изм.	Лис	№ докум.	Подпись	Дата		16

3 Проектирование программного продукта

3.1 Архитектура программного продукта

Архитектура программного продукта – это структурированное решение, которое определяет, как компоненты системы взаимодействуют друг с другом для выполнения задач, соответствующих техническим и функциональным требованиям. Для интернет-магазина компьютерных игр была разработана архитектура, которая обеспечивает высокую производительность, безопасность, управляемость и интеграцию с внешними системами.

Для интернет-магазина была выбрана клиент-серверная архитектура с использованием шаблона Model-View-Controller (MVC) на стороне сервера. Этот подход позволяет разделить логику на три компонента:

- модель (Model): отвечает за работу с данными;
- представление (View): отвечает за отображение данных пользователю;
- контроллер (Controller): управляет взаимодействием между моделью и представлением.

Архитектура интернет-магазина включает в себя следующие компоненты:

- клиентская часть;
- серверная часть;
- внешние сервисы.

На рисунке 1 представлена схема архитектуры интернет-магазина компьютерных игр.



Рисунок 1 – Схема архитектуры

Из представленной схемы можно увидеть следующее взаимодействие:

1) На стороне клиента:

- Отправляются запросы на сервер (получение списка игр, оформление заказа);

- Принимаются и отображаются данные;

- Обрабатываются вводимые данные пользователями (например, из форм).

2) На стороне сервера:

- Обрабатываются запросы от клиента;

- Взаимодействие с базой данных для получения и обновления информации;

- Генерируются ответы и отправляются клиенту (например, HTML-страницы);

- Интеграция с внешними сервисами.

Клиент-серверная архитектура позволяет разделить ответственность между клиентской и серверной частями, что упрощает разработку и поддержку. Model-View-Controller, в свою очередь, обеспечивает четкое разделение логики, что повышает читаемость и поддерживаемость кода.

Таким образом, разработанная архитектура интернет-магазина компьютерных игр обеспечивает высокую производительность, безопасность и управляемость. Она позволяет эффективно разделить обязанности между клиентской и серверной частями, а также интегрироваться с внешними сервисами.

3.2 Функциональное и структурное проектирование

Для анализа и описания интернет-магазина компьютерных игр были разработаны диаграммы, отражающие определенные аспекты функционирования интернет-магазина. На рисунке 2 представлена диаграмма прецедентов.

					ДП.09.02.07-3.25.212.17. ПЗ	Лист
Изм.	Лис	№ докум.	Подпись	Дата		18

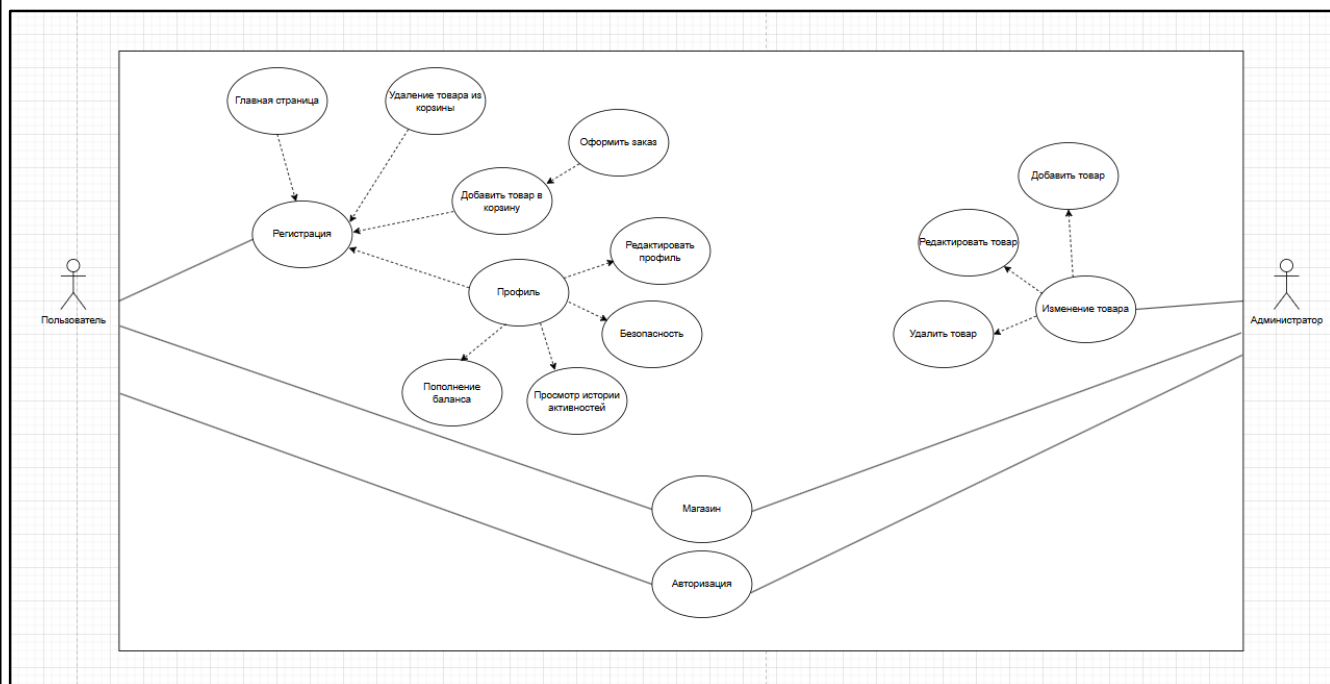


Рисунок 2 – Диаграмма прецедентов

Диаграмма прецедентов – это диаграмма, позволяющая визуализировать различные типы ролей в системе и то, как эти роли взаимодействуют с системой. На данной диаграмме представлены роли пользователей и то, какие действия они могут выполнять. Любой посетитель интернет-магазина может просматривать товары, создавать аккаунт, входить в аккаунт. Зарегистрированные пользователи могут оформлять заказы, добавлять товары в корзину, просматривать свою историю заказов. Администратор также может просматривать товары, входить в аккаунт и изменять товары (удалять, добавлять, редактировать).

Таким образом, диаграмма прецедентов демонстрирует взаимодействие основных объектов интернет-магазина компьютерных игр.

На рисунке 3 представлена диаграмма деятельности, которая показывает, как поток управления переходит от одной деятельности к другой, при этом внимание фиксируется на результате деятельности.

На диаграмме можно увидеть, как администратор входит в аккаунт, выбирает действие над товарами и выполняет их. Конечный результат интернет-магазин отображает в виде каталога. Далее пользователь может просмотреть этот каталог, выбрать товар и выбрать действие над этим товаром: добавить в корзину и оформить заказ. Оба этих действия запрашивают от пользователя вход в учетную запись или её создание. Если пользователь зарегистрировался или авторизовался, интернет-магазин отображает карточку товара при выборе действия «добавить в корзину». С этих страниц пользователь может оформить заказ.

Таким образом диаграмма деятельности интернет-магазина показывает как поток управления переходит от одной деятельности к другой.

На рисунке 4 представлена схема навигации, которая дает представление о том, как страницы интернет-магазина будут связаны друг с другом и как пользователь должен перемещаться между этими страницами.

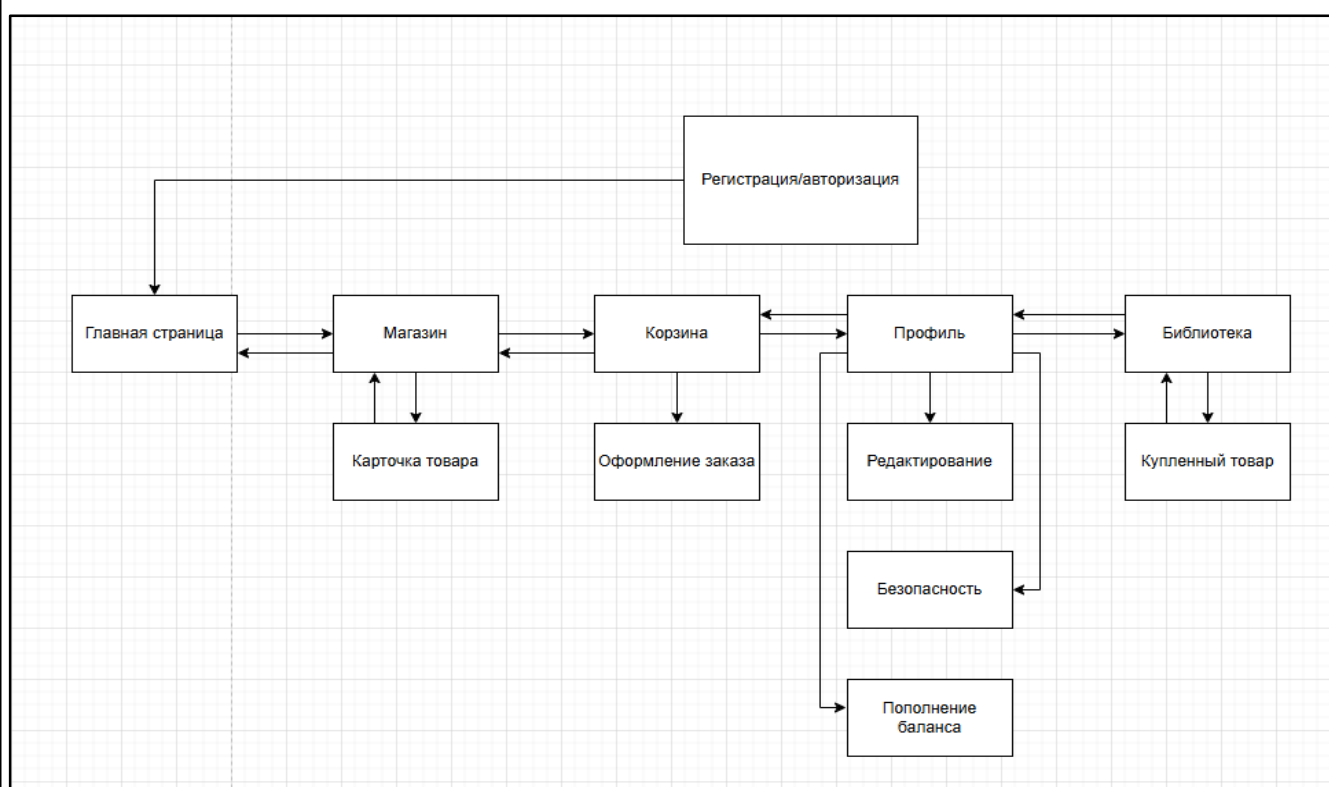


Рисунок 4 – Схема навигации

Зайдя на сайт, пользователю необходимо зарегистрироваться, чтобы попасть на главную страницу. Если пользователь уже зарегистрирован, он может перейти в свой профиль, содержащий такие разделы, как «Настройки», «Моя библиотека», «Последняя активность», «Пополнение баланса». Через раздел «Настройки» пользователь может перейти на страницу редактирования профиля и смены пароля. Через раздел «Моя библиотека» пользователь может перейти на страницу купленного товара.

Таким образом, схема навигации демонстрирует как интернет-магазин обладает интуитивно понятной структурой, которая обеспечивает удобство использования.

3.3 Проектирование базы данных

ER-модель – это модель данных, позволяющая описывать концептуальные схемы предметной области.

На рисунке 5 представлена ER-модель базы данных в третьей нормальной форме.

Нормальная форма – требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц). База данных соответствует третьей нормальной форме. Это означает, что:

- все таблицы имеют уникальные первичные ключи, а данные в полях атомарные;
- не ключевые поля зависят от всего первичного ключа, а не от его части;
- не ключевые поля зависят только от первичного ключа, а не от других не ключевых полей.

База данных содержит 6 таблиц. Таблица «users» создана для хранения информации о пользователях, таблица «cart_items» хранит информацию о

корзине, таблица «library_items» хранит информацию о купленных товарах, таблица «games» хранит информацию о самих играх, таблица «activities» создана для реализации повышения уровня профиля, после оформления заказа, таблица «sessions» содержит информацию о зарегистрированных сессиях пользователей,

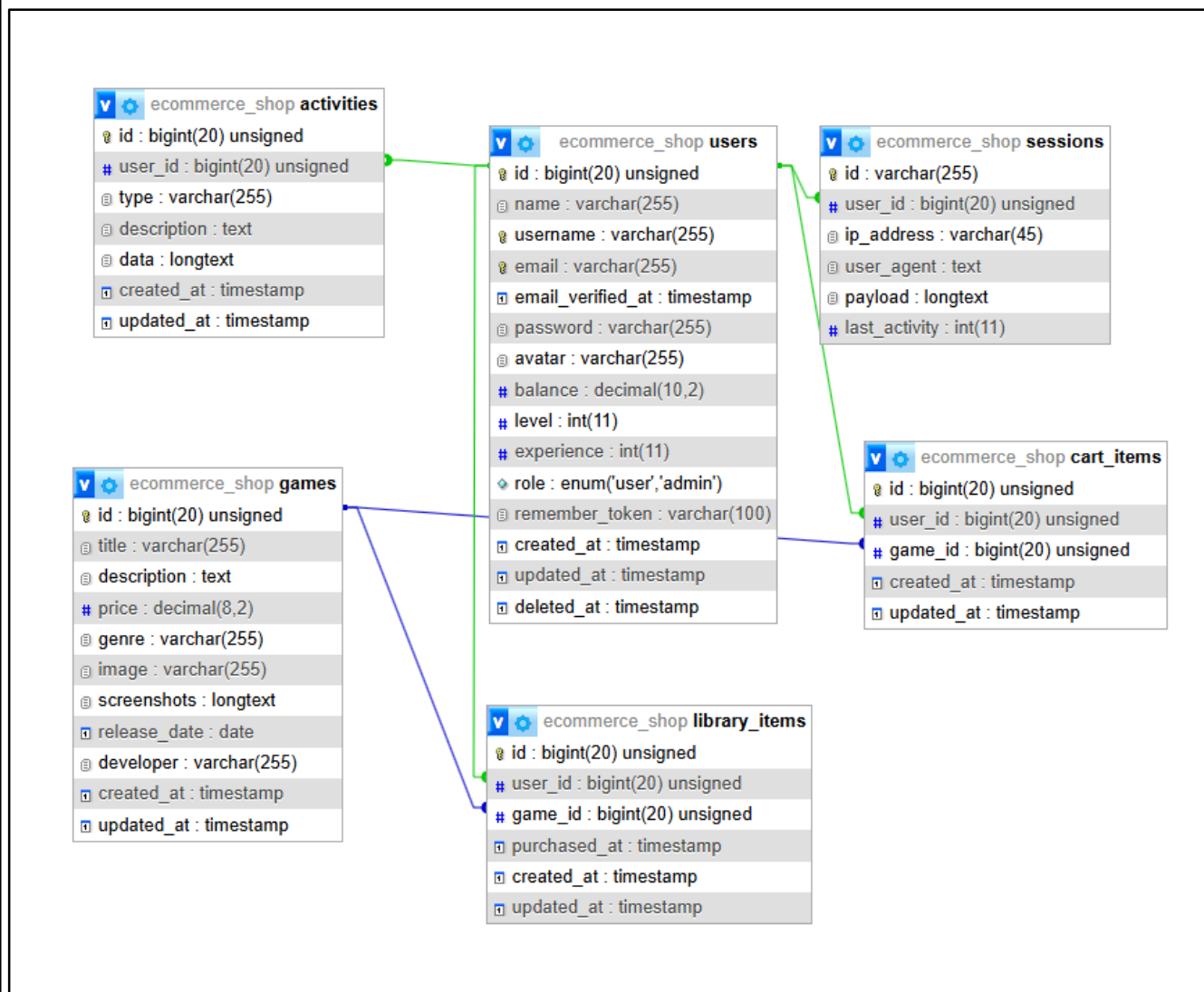


Рисунок 5 – ER-модель базы данных в третьей нормальной форме

Подробное описание базы данных представлено в таблицах 6-11.

Таблица 6 – Описание полей таблицы users

Поле	Тип данных	Описание
id	INT	Уникальный идентификатор пользователя (PK)
name	VARCHAR(255)	Имя пользователя
username	VARCHAR(255)	Никнейм пользователя
email	VARCHAR(255)	Почта
password	VARCHAR(255)	Пароль
avatar	VARCHAR(255)	Аватар пользователя
balance	DECIMAL(10,2)	Баланс
level	INT(11)	Уровень
experience	INT(11)	Опыт
role	ENUM	Роль

Таблица 7 – Описание полей таблицы cart_items

Поле	Тип данных	Описание
id	INT	Уникальный идентификатор записи в корзине (PK)
user_id	INT	Идентификатор пользователя
game_id	INT	Идентификатор игры

Таблица 8 – Описание полей таблицы library_items

Поле	Тип данных	Описание
id	INT	Уникальный идентификатор библиотеки (PK)
user_id	BIGINT(20)	Идентификатор пользователя
game_id	BIGINT(20)	Идентификатор игры

Таблица 9 – Описание полей таблицы games

Поле	Тип данных	Описание
id	ENUM('new', 'discount', 'none')	Тэг товара
title	DECIMAL(5,2)	Размер скидки
description	TEXT	описание
price	DECIMAL(8,2)	цена
genre	VARCHAR(255)	жанр
image	VARCHAR(255)	изображение
screenshots	LONGTEXT	скриншоты
release_date	DATE	дата выпуска
developer	VARCHAR(255)	разработчик

Таблица 10 – Описание полей таблицы activities

Поле	Тип данных	Описание
id	INT	Уникальный идентификатор заказа (PK)
user_id	DECIMAL(10,2)	Итоговая сумма заказа
type	VARCHAR(255)	Тип
description	TEXT	Описание
data	LONGTEXT	Дата

Таблица 11 – Описание полей таблицы sessions

Поле	Тип данных	Описание
id	VARCHAR(255)	Идентификатор сессии
user_id	BIGINT(20)	Идентификатор пользователя
ip_adress	VARCHAR(45)	Айпи адрес пользователя
user_agent	TEXT	Пользовательский агент

payload	LONGTEXT	Нагрузка
last_activity	INT(11)	Последняя активность

3.4 Проектирование пользовательского интерфейса программного продукта

Проектирование пользовательского интерфейса представляет собой важный этап создания программного продукта, направленный на обеспечение удобного и эффективного взаимодействия пользователя с программным продуктом. Интерфейс включает в себя совокупность визуальных и функциональных элементов, которые позволяют пользователю получать необходимую информацию. На начальном этапе проектирования осуществляется создание прототипа, который формируется на основе анализа требований к программному продукту. Для разработки прототипа применяется специализированный инструмент Figma.

На рисунке 6 представлен прототип главной страницы интернет-магазина.

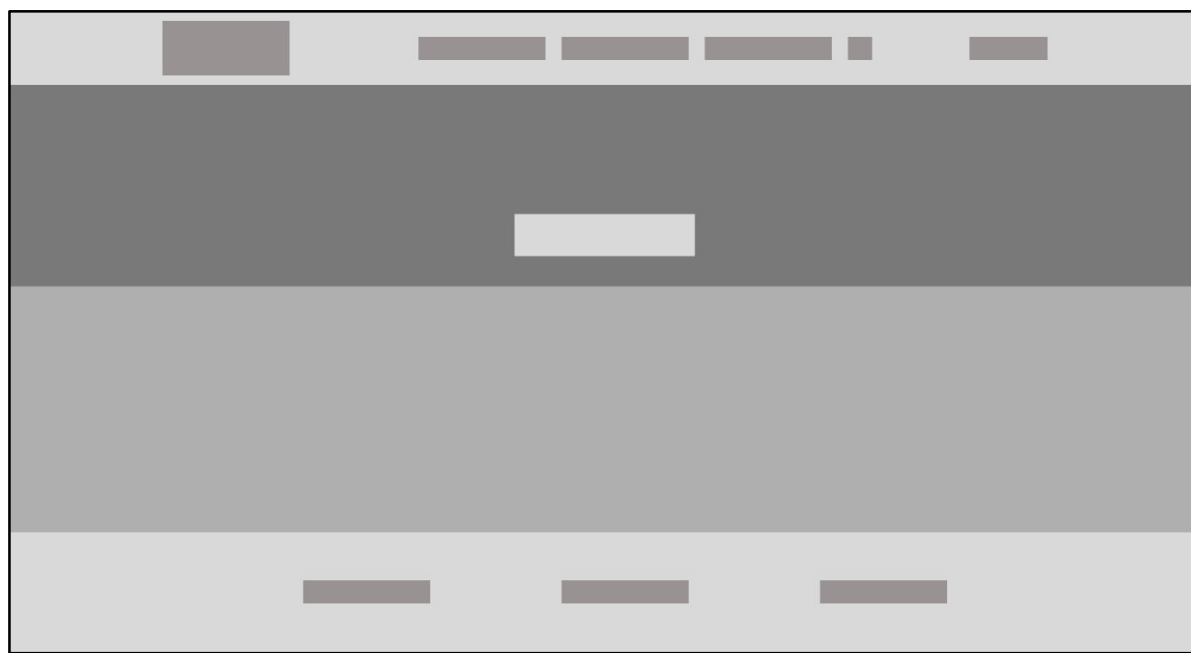


Рисунок 6 – Прототип главной страницы

В верхней части страницы расположен логотип интернет-магазина, меню навигации и профиль.

Под шапкой расположена приветствующая секция. Под секцией расположен контейнер где представлены причины выбора моего магазина. В футере находятся контактные данные.

На рисунке 7 представлен прототип страницы магазина.

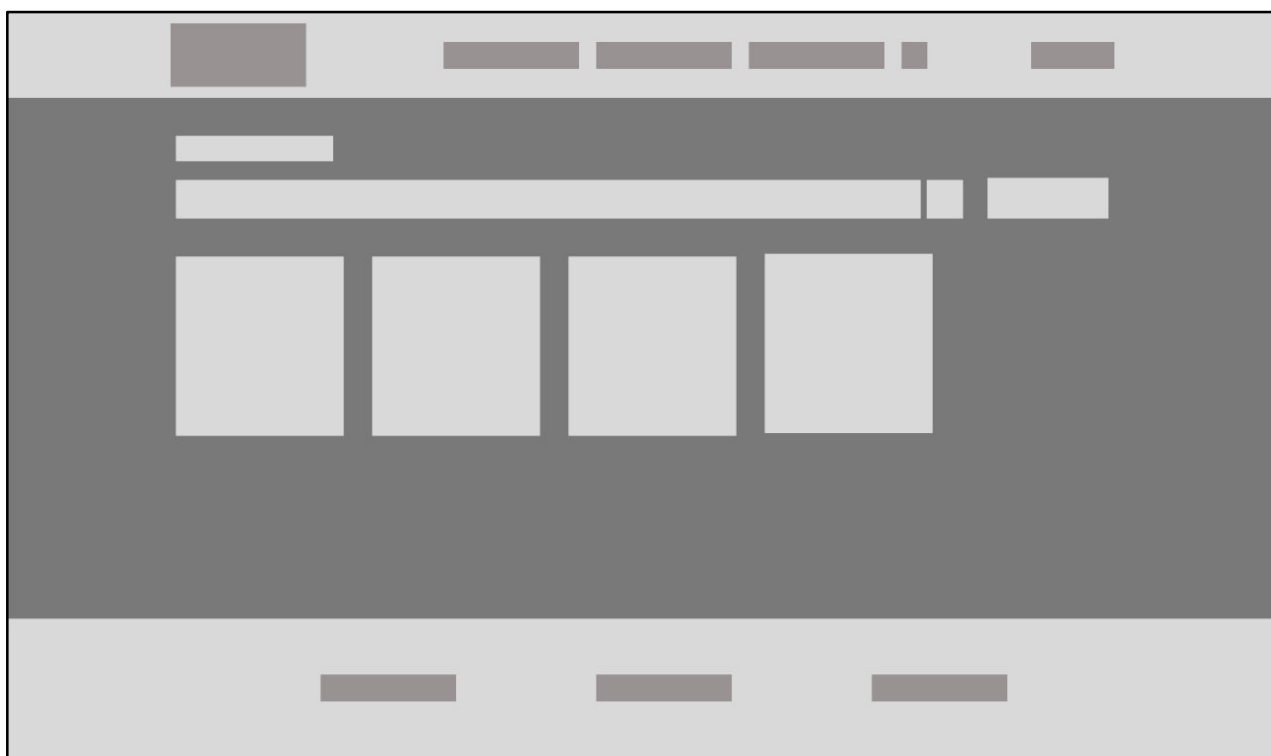


Рисунок 7 – Страница всех товаров

В верхней части страницы располагаются шапка и навигационное меню. На данной странице пользователь видит каталог всех товаров.

На рисунке 8 представлен прототип страницы корзины.



Рисунок 8 – Страница корзины

В верхней части страницы находятся шапка и навигационное меню. Ниже находится сама корзина.

В нижней части страницы находится подвал.

На рисунке 9 представлен прототип страницы карточки товара.



Рисунок 9 – Карточка товара

В верхней части страницы находятся шапка и навигационное меню. Под ними располагается изображение товара. Справа от изображения находится информация о деталях товара, таких как издатель, год выпуска и жанр. Там же расположена цена товара и кнопка для добавления товара в корзину, под этой же кнопкой расположено описание товара.

Под изображением находятся небольшие скриншоты самого товара.

В нижней части страницы находится подвал.

На рисунке 10 представлен прототип страницы оформления заказа.



Рисунок 10 – Страница оформления заказа

В верхней части страницы находятся шапка и навигационное меню. Под ними располагается информация о заказе и сам товар. Под товаром, справа находится кнопка «Оформить заказ»

В нижней части находится подвал.

На рисунке 11 представлен прототип страницы библиотеки.

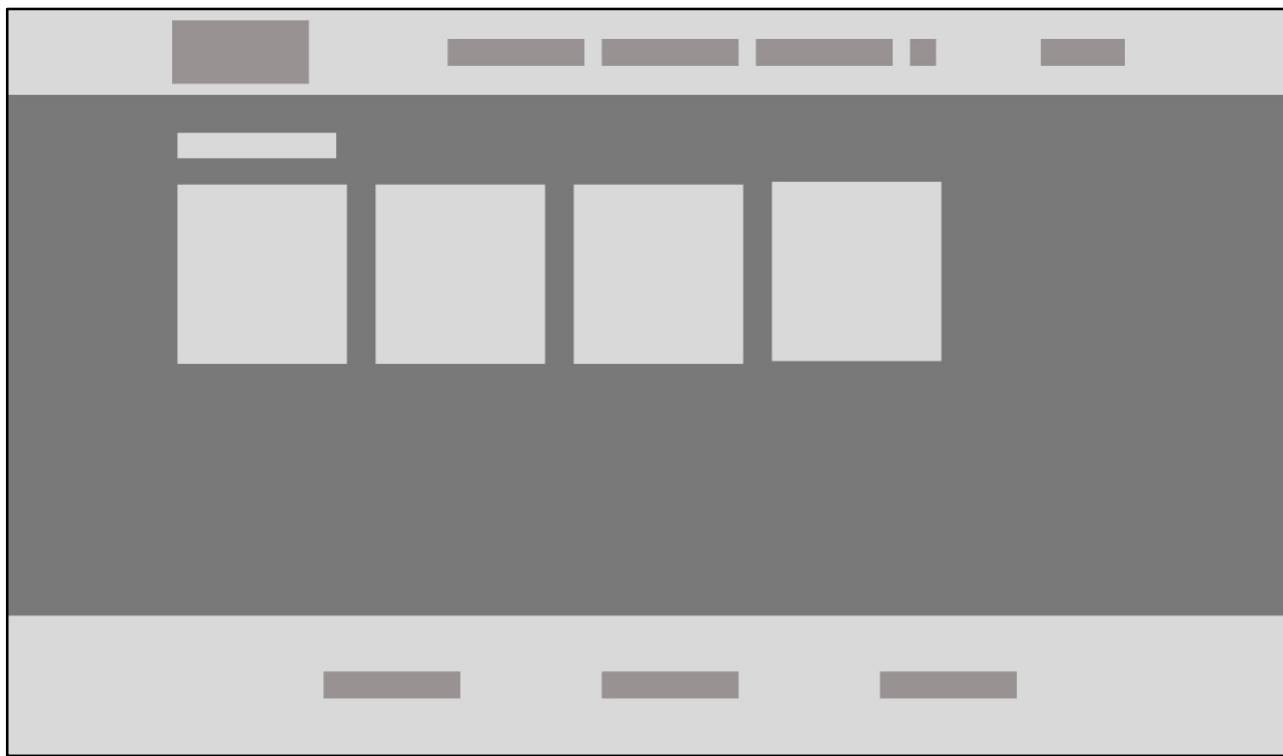


Рисунок 11 – Прототип страницы библиотеки

В верхней части страницы находятся шапка и навигационное меню. Ниже сама библиотека, где будут храниться купленные игры.

Ниже расположен подвал интернет-магазина.

Для интернет-магазина компьютерных игр были выбраны определенные цвета и шрифты, обеспечивающие единый стиль. Цветовая гамма и выбранные шрифты интернет-магазина представлены на рисунке 12.



Рисунок 12 – Цветовая гамма и шрифты

Цветовая гамма интернет-магазина компьютерных игр была разработана с целью создания привлекательного и удобного для восприятия интерфейса. Основные цвета, выбранные для дизайна, включают:

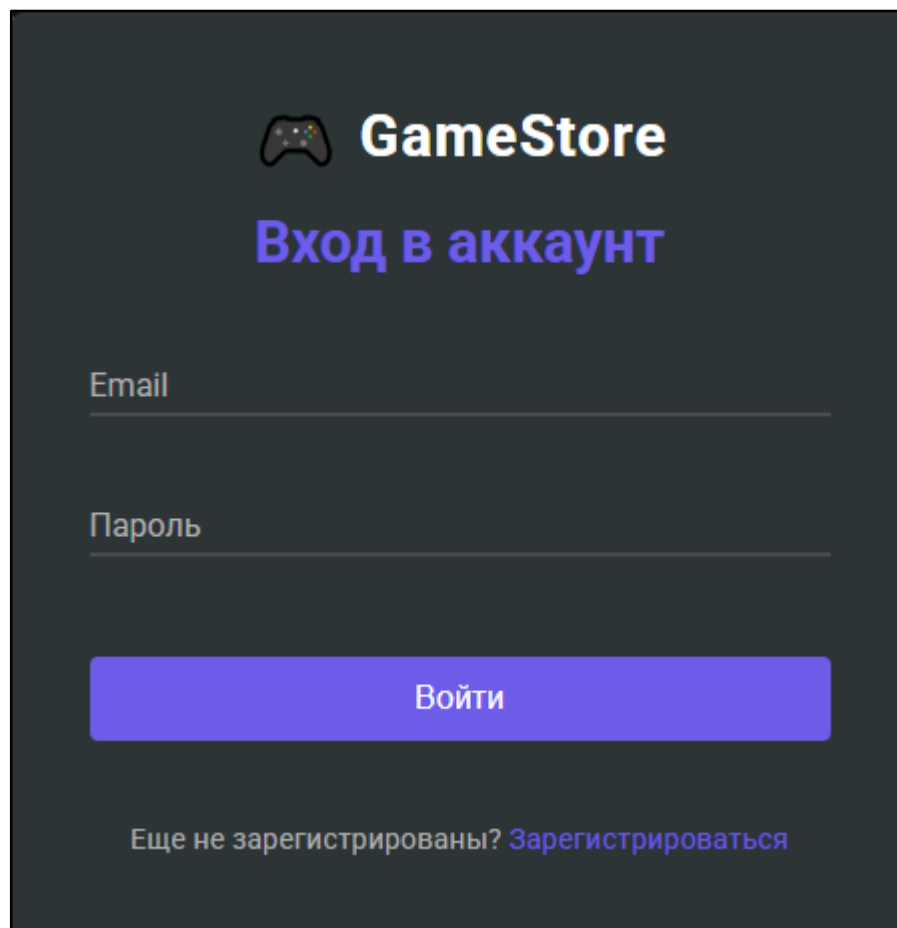
- #1A1A2E и #5649C0 были выбраны для стилизации всего интернет-магазина, так как эти цвета приятно сочетаются друг с другом и приятно глазу. Такие цвета ассоциируются с энергией, которая придает силу для новых начинаний в сфере киберспорта. Для заднего фона интернет-магазина выбрано изображение абстракции, которая даст вам полностью углубиться в виртуальный мир.

Для текстового оформления интернет-магазина были выбраны шрифты семейства Roboto sans-serif:

- Roboto sans-serif был выбран в качестве основного для текстового контента благодаря своей нейтральности, понятности и высокой читаемости. Его универсальность позволяет использовать шрифт для длинных текстов, таких как описания игр, не вызывая усталости глаз;

Таким образом, выбор цветовой палитры и шрифтов был сделан с учетом потребностей целевой аудитории и задач интернет-магазина, что обеспечивает эстетическую привлекательность, функциональность и удобство взаимодействия пользователей с интернет-магазином.

Для обеспечения удобства использования интернет-магазина были разработаны следующие элементы интерфейса: поля ввода, кнопки, меню. На рисунке 13 представлена форма авторизации, состоящая из двух полей ввода и кнопки войти.



The login form for GameStore features a dark gray background. At the top center is a game controller icon followed by the text "GameStore" in white. Below this, the heading "Вход в аккаунт" is displayed in a vibrant blue. The form includes two input fields: "Email" and "Пароль", both with light gray placeholder text. A prominent blue button labeled "Войти" is positioned below the password field. At the bottom, a link "Зарегистрироваться" is provided in blue, preceded by the text "Еще не зарегистрированы?" in light gray.

Рисунок 13 – Элементы формы авторизации

На рисунке 14 представлен логотип и меню навигации, которое выполнено в соответствии с выбранными цветовой гаммой и типографикой.

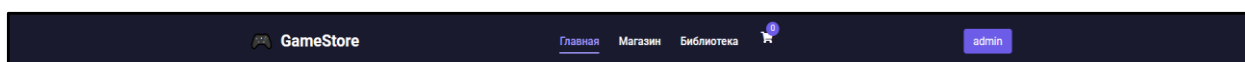


Рисунок 14 – Меню навигации

На рисунке 15 представлена панель пополнения своего аккаунта, для оформления заказов.

Пополнение баланса

Номер карты

Сумма (руб.)

Пополнить

Рисунок 15 – Панель пополнения

На рисунке 16 представлена панель информации о товаре.

Grand Theft Auto VI

Rockstar Games

Дата выхода: 15.03.2025

Экшен

В корзину

Описание

Новая часть легендарной серии игр про криминальный мир.

Рисунок 16 – Панель информации товара

					ДП.09.02.07-3.25.212.17. ПЗ	Лист
Изм.	Лис	№ докум.	Подпись	Дата		34

На рисунке 17 представлен профиль пользователя с его настройками, библиотекой и активностью.

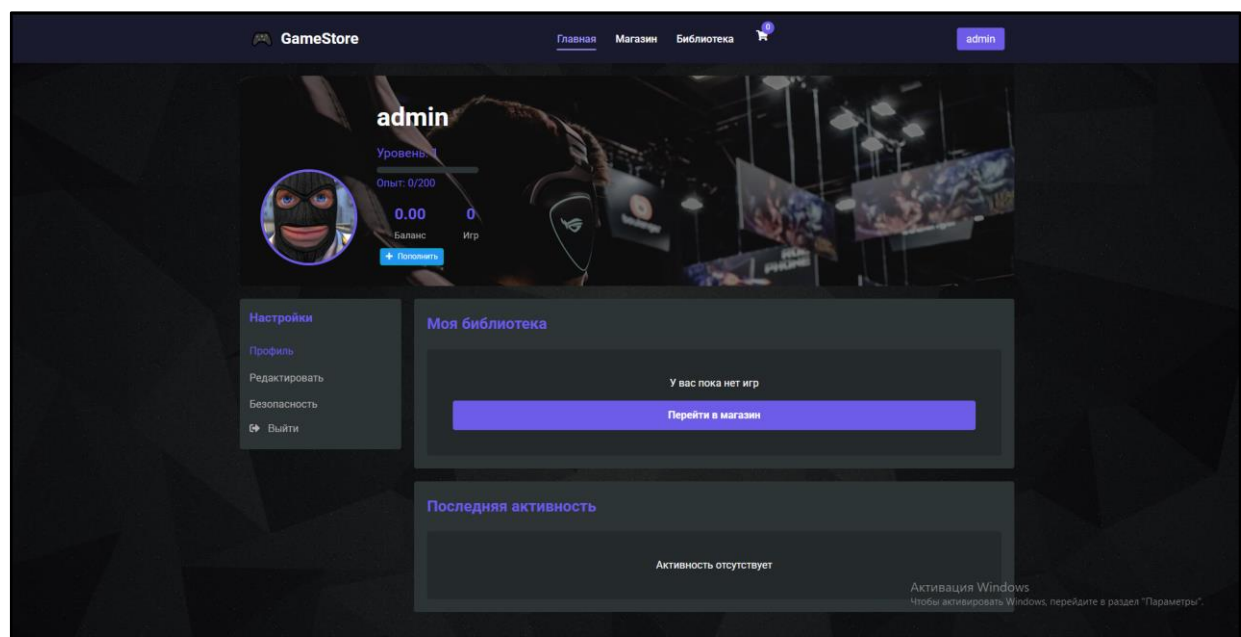


Рисунок 17 – Страница профиля

На рисунке 18 представлены страница магазина товаров, на ней можно закинуть в корзину сам товар, отфильтровать товары по жанру, либо поиску.

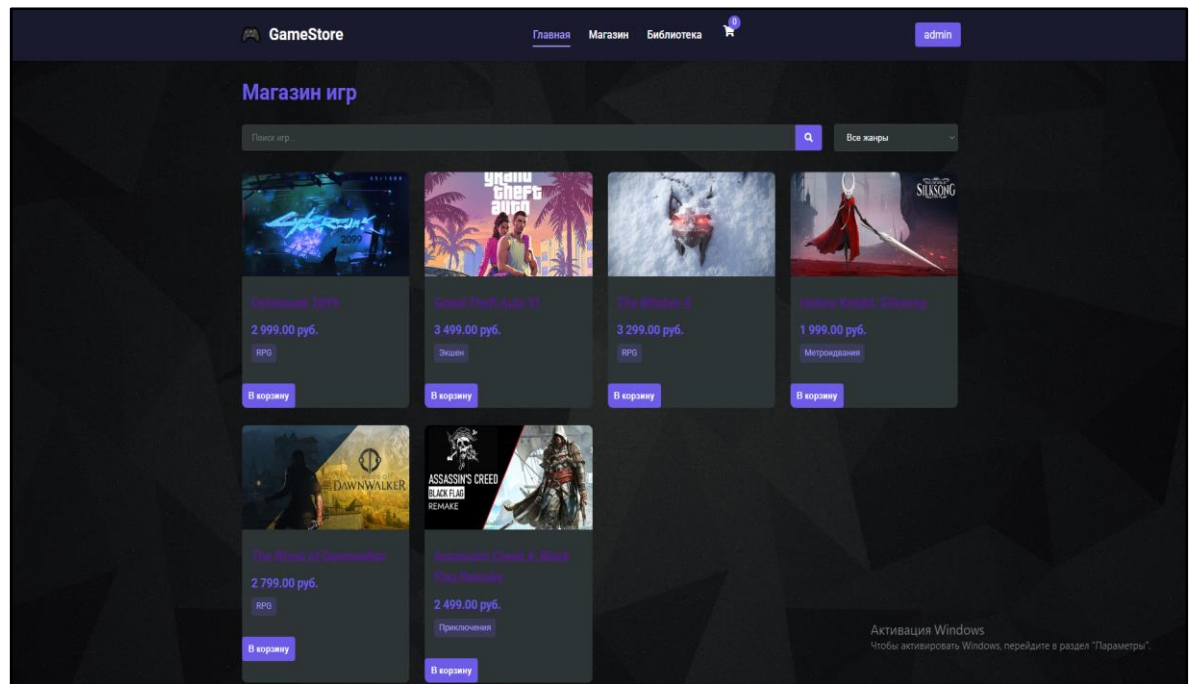


Рисунок 18 – Страница магазина

Разработанные элементы интерфейса обеспечивают удобное взаимодействие пользователя с интернет-магазином, создают привлекательный пользовательский интерфейс.

При разработке дизайн-макетов интернет-магазина компьютерных игр учитывался ряд ключевых аспектов, направленных на создание визуально привлекательного интерфейса. Основное внимание уделялось потребностям целевой аудитории.

На рисунке 19 представлен дизайн-макет главной страницы.

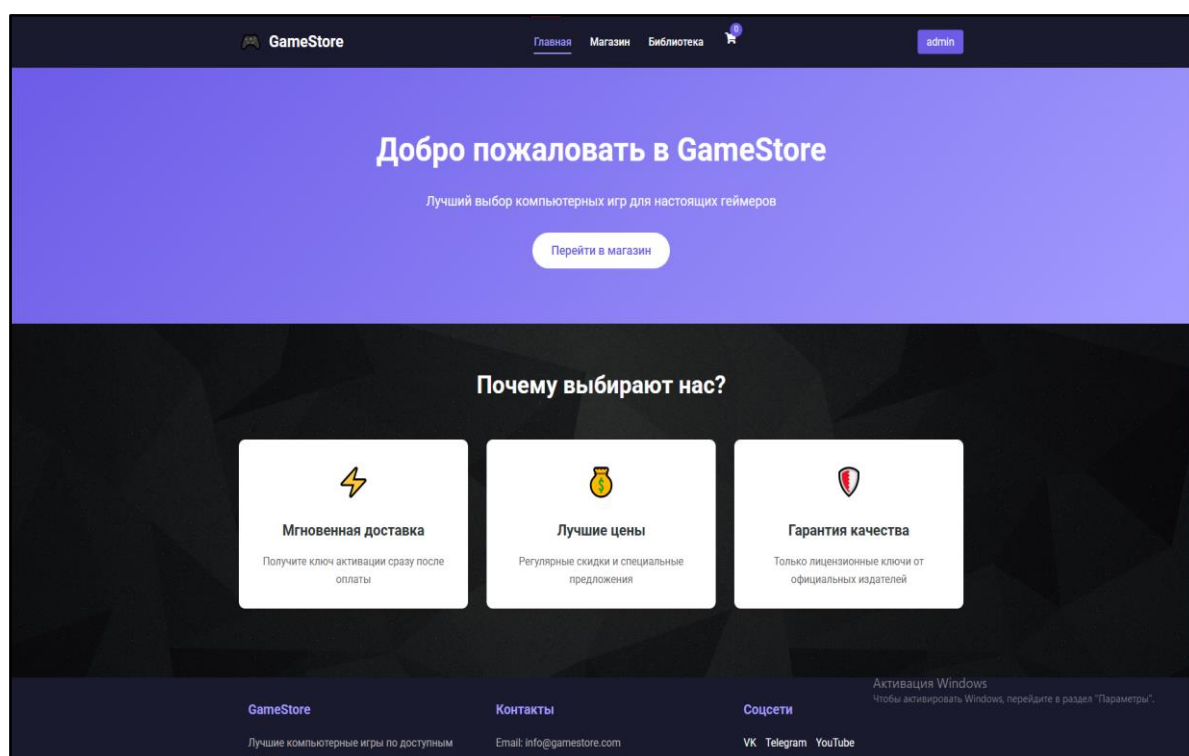


Рисунок 19 – Дизайн-макет главной страницы

На данном макете представлен дизайн главной страницы интернет-магазина, соответствующий его общему стилю.

На рисунке 20 представлен дизайн-макет страницы профиля.

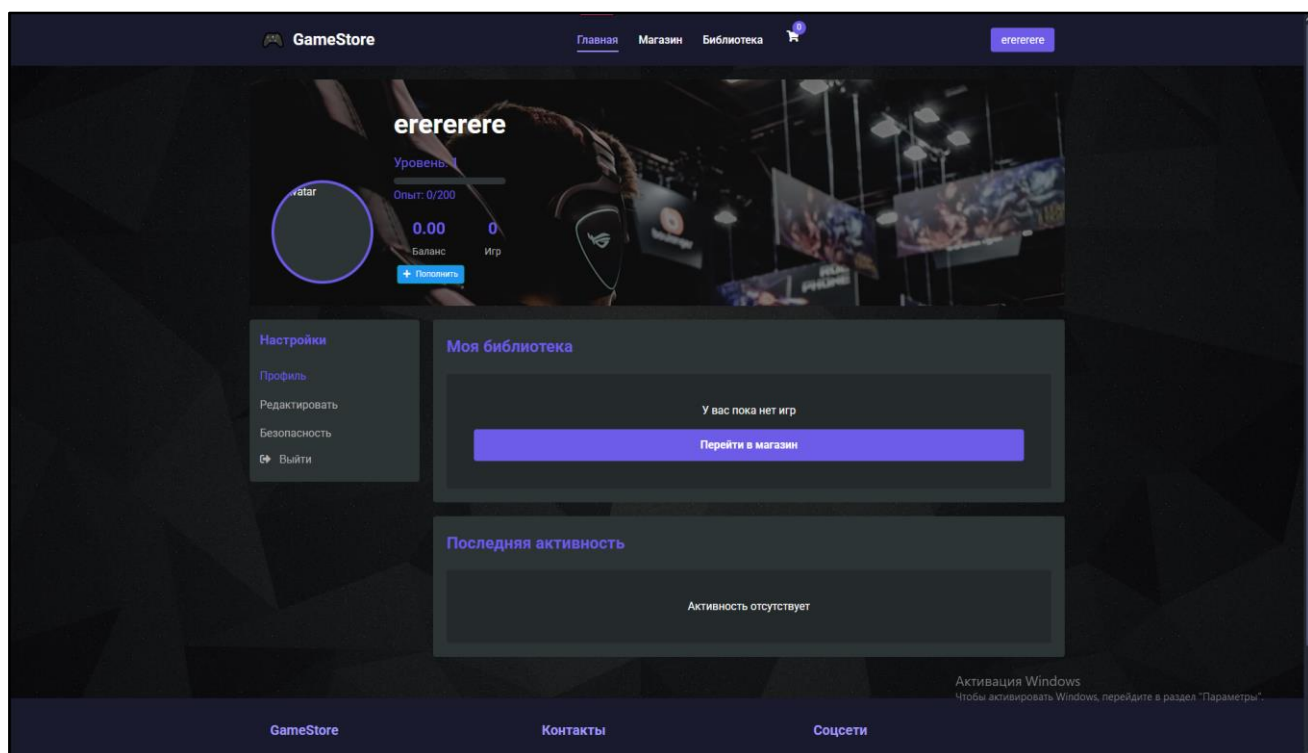


Рисунок 20 – Дизайн-макет страницы профиля

На данном макете располагаются все данные и настройки пользователя, соответствующие стилистике интернет-магазина.

Дизайн-макеты интернет-магазина компьютерных игр были разработаны с учетом потребностей целевой аудитории. Макеты отражают визуальную иерархию: заголовки, текст, изображения и кнопки выделяются с помощью размеров, шрифтов и цветовых акцентов, что помогает пользователям быстро находить ключевую информацию. Также макеты отражают уникальный стиль интернет-магазина, включая логотип, цветовую гамму и шрифты.

4 Реализация программного продукта

Клиентская часть реализована благодаря синтаксису blade. С помощью этого синтаксиса повторяющиеся элементы кода можно записать в один шаблон, который потом будет использоваться во множествах представлений. Часть кода такого шаблона представлена на рисунке 21. Полный код представлен в приложении Б.

					ДП.09.02.07-3.25.212.17. ПЗ	Лист
Изм.	Лис	№ докум.	Подпись	Дата		38

```

app.blade.php ×
resources > views > layouts > app.blade.php > html > body.site-container > header.header > div.container > div.header_inner > nav.nav > a.cart-link
1 <!DOCTYPE html>
2 <html lang="ru">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>GameStore - Интернет-магазин компьютерных игр</title>
8   <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700&display=swap" rel="stylesheet">
9   <link href="/css/app.css" rel="stylesheet">
10  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
11 </head>
12
13 <body class="site-container">
14   <header class="header">
15     <div class="container">
16       <div class="header_inner">
17         <a href="/" class="logo">
18           <span class="logo_icon"></span>
19           <span class="logo_text">GameStore</span>
20         </a>
21
22         <nav class="nav">
23           <a href="/" class="nav_link active">Главная</a>
24           <a href="/shop" class="nav_link">Магазин</a>
25           <a href="/library" class="nav_link">Библиотека</a>
26           <a href="{{ route('cart.index') }}" class="cart-link">
27             <i class="fas fa-shopping-cart"></i>
28             @auth
29             <span class="cart-count">{{ auth()->user()->cartItems()->count() }}</span>
30             @endauth
31           </a>
32         </nav>
33
34         <div class="user-menu">
35           <a href="{{ route('profile') }}" class="user-menu_link">
36             <span>{{ Auth::user()->username }}</span>
37           </a>
38         </div>
39       </div>
40     </div>
41   </header>
42
43   <main class="site-content">
44     @yield(section: 'content')
45   </main>
46
47   <footer class="footer">
48     <div class="container">
49       <div class="footer_inner">
50         <div class="footer_col">
51           <h3 class="footer_title">GameStore</h3>
52           <p class="footer_text">Лучшие компьютерные игры по доступным ценам</p>
53         </div>
54         <div class="footer_col">
55           <h3 class="footer_title">Контакты</h3>
56           <p class="footer_text">Email: info@gamestore.com</p>
57           <p class="footer_text">Телефон: +7 (123) 456-78-90</p>
58         </div>
59         <div class="footer_col">
60           <h3 class="footer_title">Соцсети</h3>

```

Рисунок 21 – Часть кода основного шаблона страниц

Реализация базы данных выполнена с помощью миграций. Для начала нужно подключение к базе данных, а затем выполнение миграций. На рисунке 22 представлено подключение к базе данных.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=ecommerce_shop
DB_USERNAME=root
DB_PASSWORD=
```

Рисунок 22 – Подключение к базе данных

Настройки подключения содержат следующие параметры:

- 1) DB_CONNECTION – выбор сервера для базы данных;
- 2) DB_HOST – хост, на котором запущен сервер mysql;
- 3) DB_PORT – номер порта, на котором запущен сервер mysql;
- 4) DB_DATABASE – имя базы данных, к которой идет подключение;
- 5) DB_USERNAME – имя пользователя mysql;
- 6) DB_PASSWORD – пароль пользователя mysql.

На рисунке 23 представлена миграция таблицы пользователей.

```
Schema::create('users', callback: function (Blueprint $table): void {
    $table->id();
    $table->string(column: 'name');
    $table->string(column: 'username')->unique();
    $table->string(column: 'email')->unique();
    $table->timestamp(column: 'email_verified_at')->nullable();
    $table->string(column: 'password');
    $table->string(column: 'avatar')->default(value: 'default-avatar.png')->nullable(value: false);
    $table->decimal(column: 'balance', total: 10, places: 2)->default(value: 0.00);
    $table->integer(column: 'level')->default(value: 1);
    $table->integer(column: 'experience')->default(value: 0);
    $table->enum(column: 'role', allowed: ['user', 'admin'])->default(value: 'user');
    $table->rememberToken();
    $table->timestamps();
    $table->softDeletes();
});
```

Рисунок 23 – Миграция таблицы пользователей

Миграция таблицы пользователей была создана Laravel по умолчанию.

На рисунке 24 представлена миграция таблицы «игры».

```
Schema::create(table: 'games', callback: function (Blueprint $table): void {  
    $table->id();  
    $table->string(column: 'title');  
    $table->text(column: 'description');  
    $table->decimal(column: 'price', total: 8, places: 2);  
    $table->string(column: 'genre');  
    $table->string(column: 'image');  
    $table->json(column: 'screenshots')->nullable();  
    $table->date(column: 'release_date');  
    $table->string(column: 'developer');  
    $table->timestamps();  
});
```

Рисунок 24 – Миграция «игры»

На рисунке 25 представлена миграция таблицы «корзины».

```
Schema::create(table: 'cart_items', callback: function (Blueprint $table): void {  
    $table->id();  
    $table->foreignId(column: 'user_id')->constrained();  
    $table->foreignId(column: 'game_id')->constrained();  
    $table->timestamps();  
});
```

Рисунок 25 – Миграция «корзины»

На рисунке 26 представлена миграция таблицы «библиотеки».

```
Schema::create(table: 'library_items', callback: function (Blueprint $table): void {  
    $table->id();  
    $table->foreignId(column: 'user_id')->constrained();  
    $table->foreignId(column: 'game_id')->constrained();  
    $table->timestamp(column: 'purchased_at')->useCurrent();  
    $table->timestamps();  
});
```

Рисунок 26 – Миграция «библиотеки»

На рисунке 27 представлена миграция таблицы «активности».

```
Schema::create(table: 'activities', callback: function (Blueprint $table): void {
    $table->id();
    $table->foreignId(column: 'user_id')->constrained();
    $table->string(column: 'type'); // 'purchase', 'login', etc
    $table->text(column: 'description');
    $table->json(column: 'data')->nullable();
    $table->timestamps();
});
```

Рисунок 27 – Миграция «активности»

Следующий этап реализации программного продукта – это разработка его серверной части. На рисунке 28 представлен код обновления данных пользователя.

```
public function update(Request $request): RedirectResponse
{
    $user = auth()->user();

    $request->validate(rules: [
        'name' => 'required|string|max:255',
        'username' => 'required|string|max:255|unique:users,username,' . $user->id,
        'email' => 'required|string|email|max:255|unique:users,email,' . $user->id,
        'avatar' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
    ]);

    $data = $request->only(keys: ['name', 'username', 'email']);

    if ($request->hasFile(key: 'avatar')) {
        if ($user->avatar) {
            Storage::disk(name: 'public')->delete(paths: $user->avatar);
        }
        $data['avatar'] = $request->file(key: 'avatar')->store(path: 'avatars', options: 'public');
    }

    $user->update(attributes: $data);

    return redirect()->route(route: 'profile')->with(key: 'success', value: 'Профиль успешно обновлен!');
}
```

Рисунок 28 – Код обновления данных пользователя

Обновление данных пользователя осуществляется через HTTP-запрос с методом POST, который передает данные из формы в контроллер.

На рисунке 29 представлен код списка игр и фильтрации.

```

public function index(): View
{
    $games = Game::query()
        ->when(value: request(key: 'genre'), callback: fn(Builder<Game> $q, $genre): Build... => $q->where('genre', $genre))
        ->when(value: request(key: 'search'), callback: fn(Builder<Game> $q, $search): Build... => $q->where('title', 'like', "%{$search}%"))
        ->paginate(perPage: 6);

    $genres = Game::distinct()->pluck(column: 'genre');

    return view(view: 'shop.index', data: compact(var_name: 'games', var_names: 'genres'));
}

```

Рисунок 29 – Код списка игр и фильтрации

Функция `index()` выводит список игр с пагинацией, фильтрует их по жанру и ищет по названию. Также она собирает все уникальные жанры для фильтра в выпадающем списке. Всё это передается в шаблон для отображения.

Таким образом реализована серверная часть обновления данных пользователя и список игр с пагинацией, и фильтром.

Также в интернет-магазине реализованы функции пополнения баланса профиля через номер карты(симуляция). На рисунке 30 представлена часть кода контроллера баланса. Полный код представлен в приложении В.

```

app > Http > Controllers > BalanceController.php > PHP Intelephense > BalanceController > topu
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\User;
7  use App\Models\Activity;
8  use Illuminate\Support\Facades\Auth;
9
10 2 references | 0 implementations
10 class BalanceController extends Controller
11 {
12     1 reference | 0 overrides
12     public function showTopupForm(): View
13     {
14         return view(view: 'balance.topup');
15     }
16     1 reference | 0 overrides
16     public function topup(Request $request): RedirectResponse
17     {
18         $request->validate(rules: [
19             'card_number' => 'required|string|size:16|regex:/^[0-9]+$/',
20             'amount' => 'required|numeric|min:100|max:10000',
21         ]);

```

Рисунок 30 – Часть кода контроллера баланса

5 Отладка и тестирование программного продукта

Отладка (Debugging) – это процесс поиска, анализа и устранения ошибок в программном коде, который выполняется после написания программы и перед её выпуском.

Тестирование программного обеспечения (ПО) – это процесс проверки соответствия программы заданным требованиям, выявления ошибок и оценки её качества перед выпуском.

На рисунке 31 представлен тест проверки на добавление игры в корзину

```
0 references | 0 overrides
public function test_add_game_to_cart(): void
{
    $user = User::factory()->create();
    $this->actingAs(user: $user);

    $game = Game::factory()->create();

    $response = $this->post(uri: route(name: 'cart.add', parameters: $game));

    $response->assertRedirect()
        ->assertSessionHas(key: 'success', value: 'Игра добавлена в корзину');

    $this->assertDatabaseHas(table: 'cart_items', data: [
        'user_id' => $user->id,
        'game_id' => $game->id
    ]);
}
```

Рисунок 31 – Тест на добавление игры в корзину

Сначала создается пользователь, затем создаем игру и отправляем запрос на добавление в корзину. Проверяем перенаправление и сообщение об успехе и проверяем, что игра действительно добавлена в корзину пользователя.

На рисунке 32 представлен тест на просмотр корзины с товарами внутри

```

0 references | 0 overrides
public function test_view_cart_with_items(): void
{
    $user = User::factory()->create();
    $this->actingAs(user: $user);

    $games = Game::factory()->count(3)->create();
    foreach ($games as $game) {
        $user->cartItems()->create(['game_id' => $game->id]);
    }

    $response = $this->get(uri: route(name: 'cart.index'));

    $response->assertOk();

    foreach ($games as $game) {
        $response->assertSee(value: $game->title);
    }

    $total = $games->sum('price');
    $response->assertSee(value: $total);
}

```

Рисунок 32 – Тест на просмотр корзины с товарами

Создаем пользователя и создаем игры, добавляя их в корзину, получаем страницу корзины и проверяем ответ. Проверяем, что все игры отображаются и проверяем подсчет общей суммы.

На рисунке 33 представлен тест на удаление игры из корзины

```

0 references | 0 overrides
public function test_remove_game_from_cart(): void
{
    $user = User::factory()->create();
    $this->actingAs(user: $user);

    $game = Game::factory()->create();
    $cartItem = $user->cartItems()->create(['game_id' => $game->id]);

    $response = $this->delete(uri: route(name: 'cart.remove', parameters: $cartItem));

    $response->assertRedirect()
        ->assertSessionHas(key: 'success', value: 'Игра удалена из корзины');

    $this->assertDatabaseMissing(table: 'cart_items', data: [
        'id' => $cartItem->id
    ]);
}

```

Рисунок 33 – Тест на удаление игры из корзины

Создаем пользователя и также создаем игру, и добавляем в корзину. Отправляем запрос на удаление, проверяем перенаправление и сообщение и проверяем, что игры больше нет в корзине.

На рисунке 34 представлен результат всех тестов

```

PASS Tests\Feature\CartControllerTest
✓ add game to cart
✓ view cart with items
✓ remove game from cart
Tests: 3 passed (11 assertions)
Duration: 0.74s

```

Рисунок 34 – Результат тестов

Модульное тестирование позволяет быстро находить ошибки в отдельных компонентах и обеспечивает уверенность в их работоспособности при рефакторинге.

В таблице 12 представлены тестовые наборы в соответствии с сценариями тестирования.

Таблица 12 – Тестовые наборы в соответствии с сценариями тестирования

№	Тестируемый компонент	Сценарий тестирования	Ожидаемый результат	Методы проверки
1	Корзина	Авторизованный пользователь добавляет игру в корзину	1. Редирект на предыдущую страницу 2. Сообщение “Игра добавлена в корзину” 3. Запись в таблице cart_items	- assertRedirect() - assertSessionHas('success') - assertDatabaseHas()
2	Корзина	Просмотр содержимого корзины	1. Отображение всех добавленных игр 2. Корректный расчет общей суммы 3. Нет ошибок 500	- assertSee() - assertDontSee('0.00') - assertOk()
3	Корзина	Удаление игры из корзины	1. Редирект на страницу корзины 2. Сообщение «Игра удалена из корзины» 3. Отсутствие записи в БД	- _assertRedirect() - assertSessionHas('success') - assertDatabaseMissing()

Представленные тесты охватывают базовые сценарии, но требуют расширения для полного покрытия функционала. Табличная форма наглядно демонстрирует связь между компонентами, их поведением и методами верификации.

6 Руководство пользователя программного продукта

Для запуска интернет-магазина компьютерных игр необходимо прописать команду в терминале Visual Studio Code «npm run build», затем создать новый терминал и прописать «php artisan serve». Перейти по адресу, появившемуся в терминале Visual Studio Code, после чего откроется страница с авторизацией, прежде чем попасть на ту же главную страницу, потребуется для начала авторизоваться или зарегистрироваться. (рисунок 35).

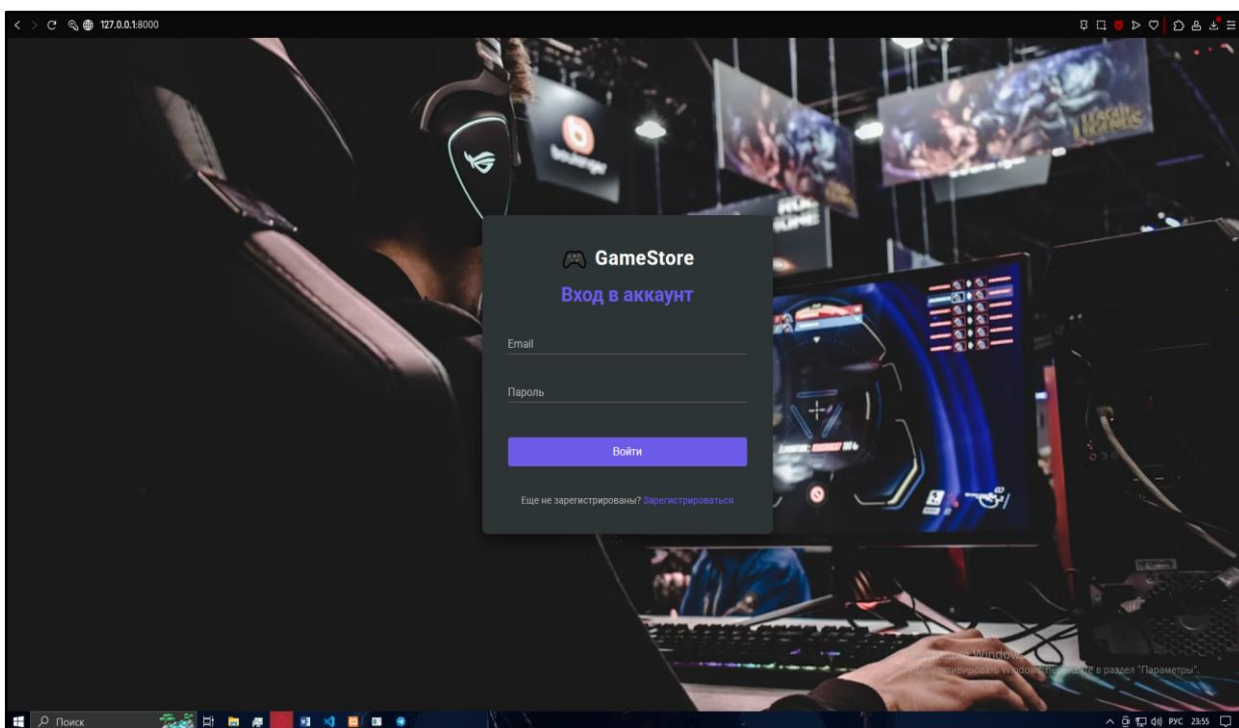


Рисунок 35 – Страница авторизации

После авторизации или регистрации пользователя, нас перебросит на главную, приветствующую страницу (рисунок 36).

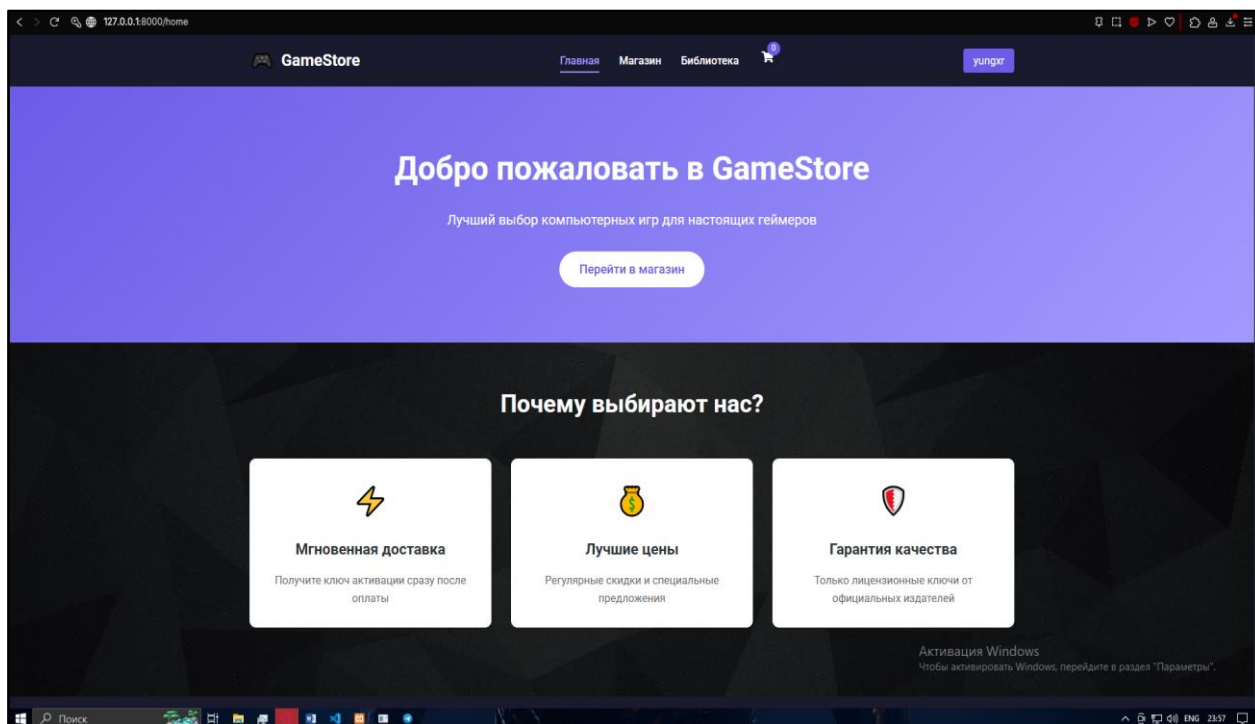


Рисунок 36 – Главная страница

С главной страницы мы можем перейти в магазин, где представлены сами товары, а именно игры, на данный момент их пока 6(рисунок 37).

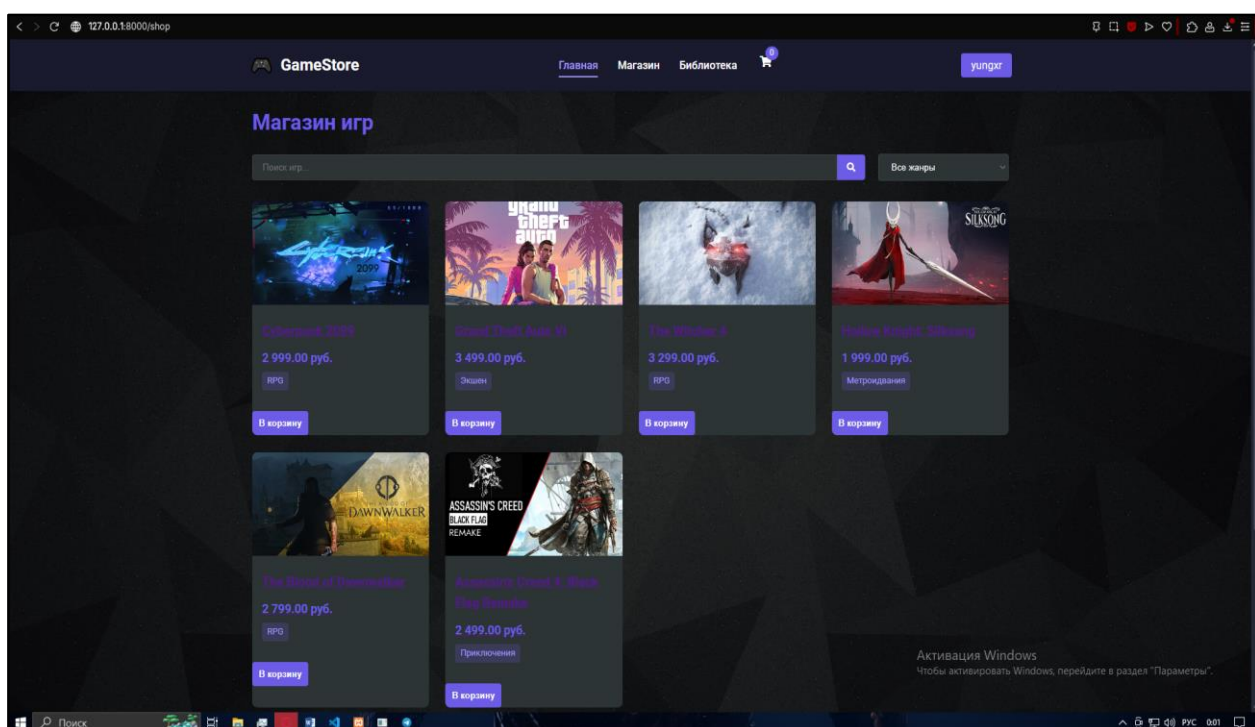


Рисунок 37 – Страница магазина

Изм.	Лис	№ докум.	Подпись	Дата

ДП.09.02.07-3.25.212.17. ПЗ

Лист

50

Всем пользователям доступен поиск. В верхней части страницы присутствует строка поиска и кнопка поиска по жанрам. Пользователи могут осуществлять обычный поиск по названию, или, нажав на кнопку поиска по жанру выбрать необходимый. Пример поиска представлен на рисунках 38-40.

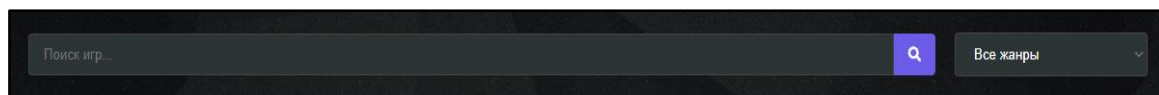


Рисунок 38 – Поиск по названию

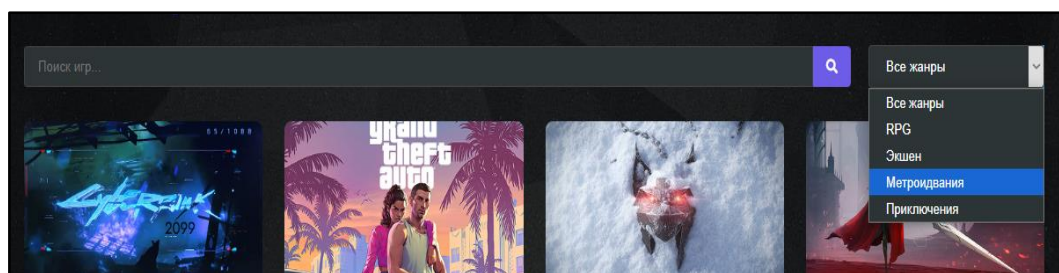


Рисунок 39 – Выбор жанра для поиска

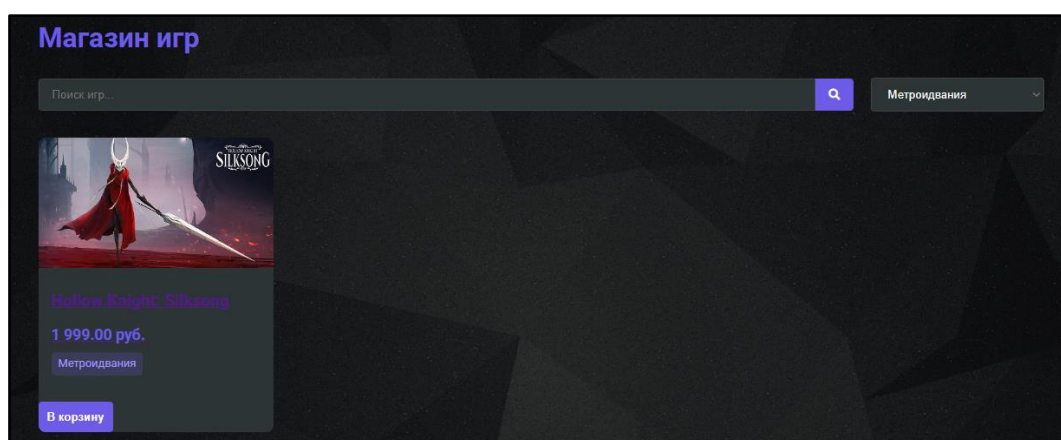


Рисунок 40 – Результаты поиска

С магазина мы можем перейти в свою библиотеку, где будут храниться наши купленные игры(рисунок 41).

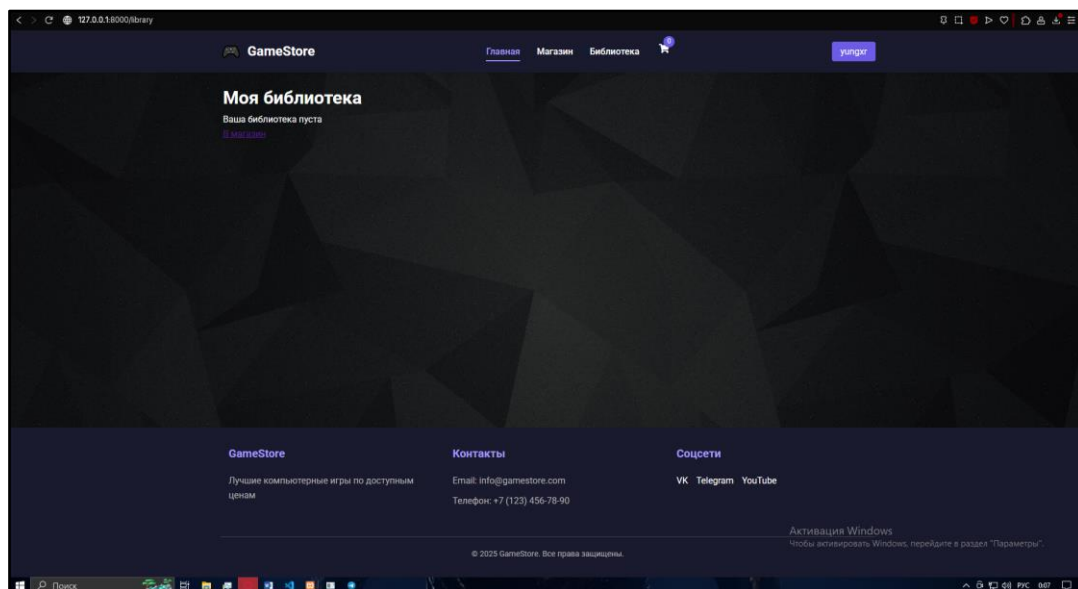


Рисунок 42 – Страница библиотеки

Страница профиля, где можно в разделе «Настройки» настроить свой профиль(добавить аватар), изменить никнейм, почту и в «Безопасность» сменить пароль(рисунок 43-45).

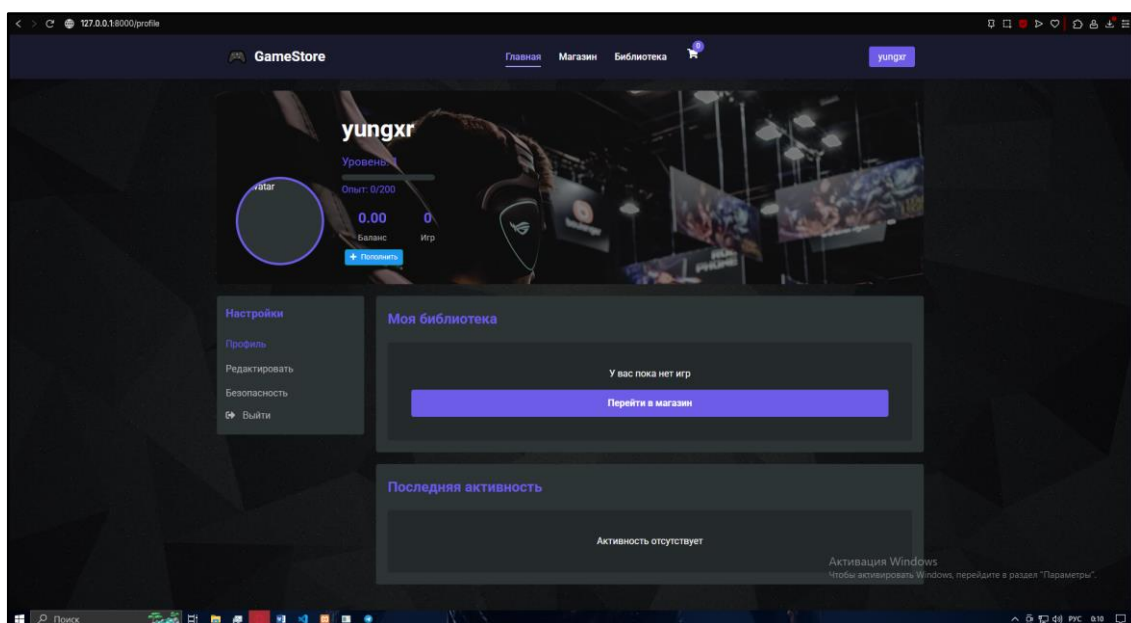


Рисунок 43 – Страница профиля

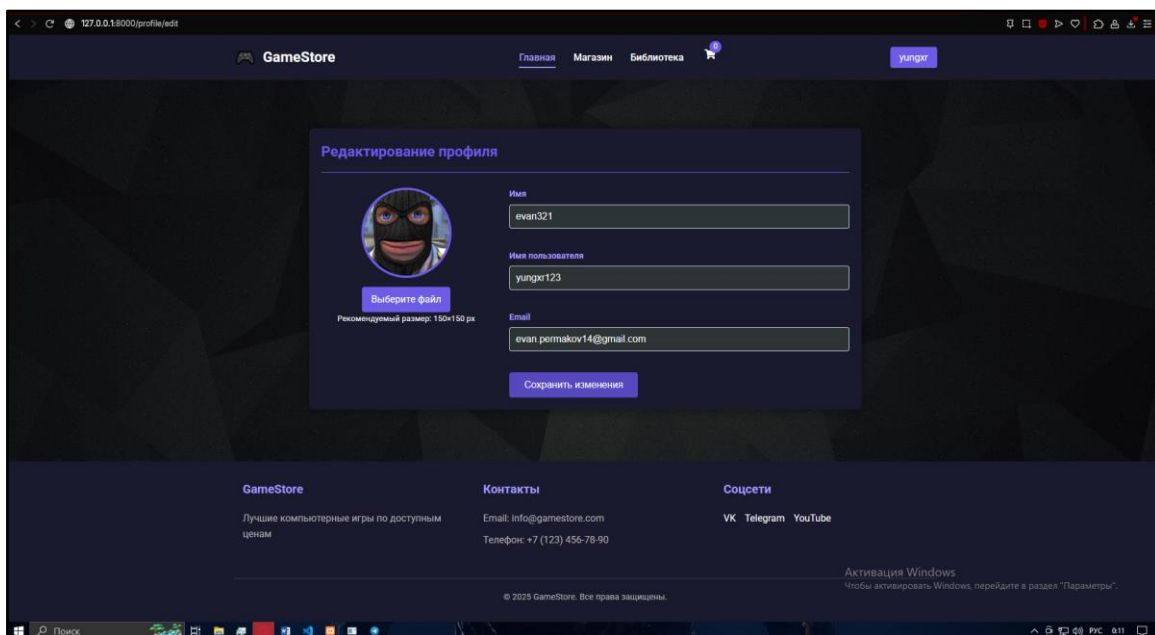


Рисунок 44 – Страница редактирования профиля

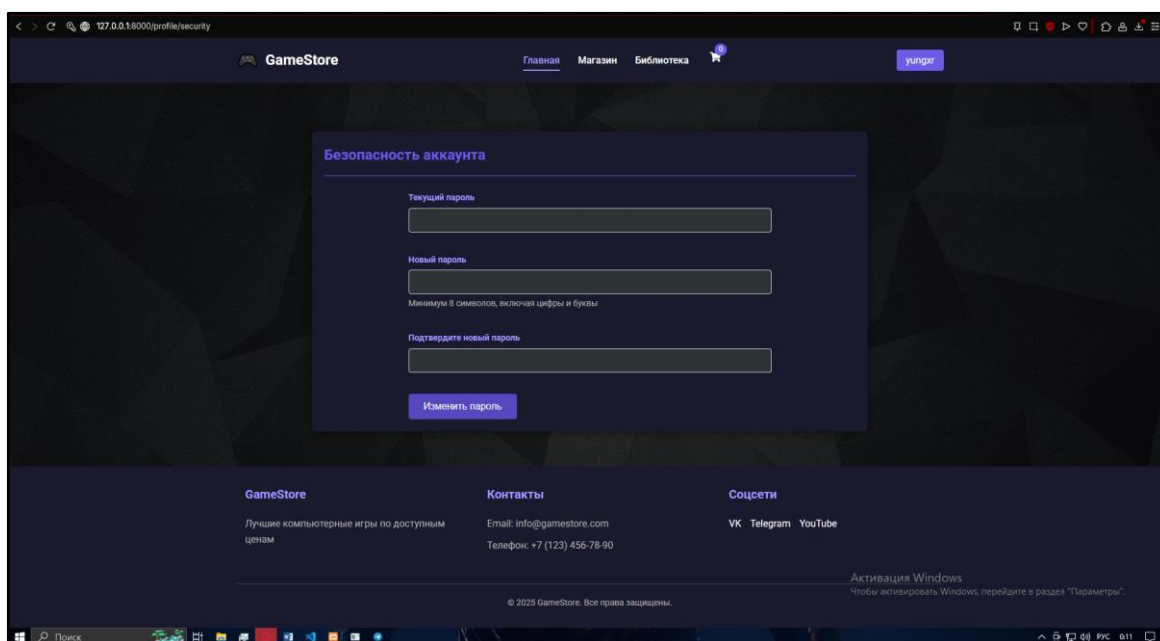


Рисунок 45 – Страница безопасности профиля

Страница корзины представлена на рисунке 46.

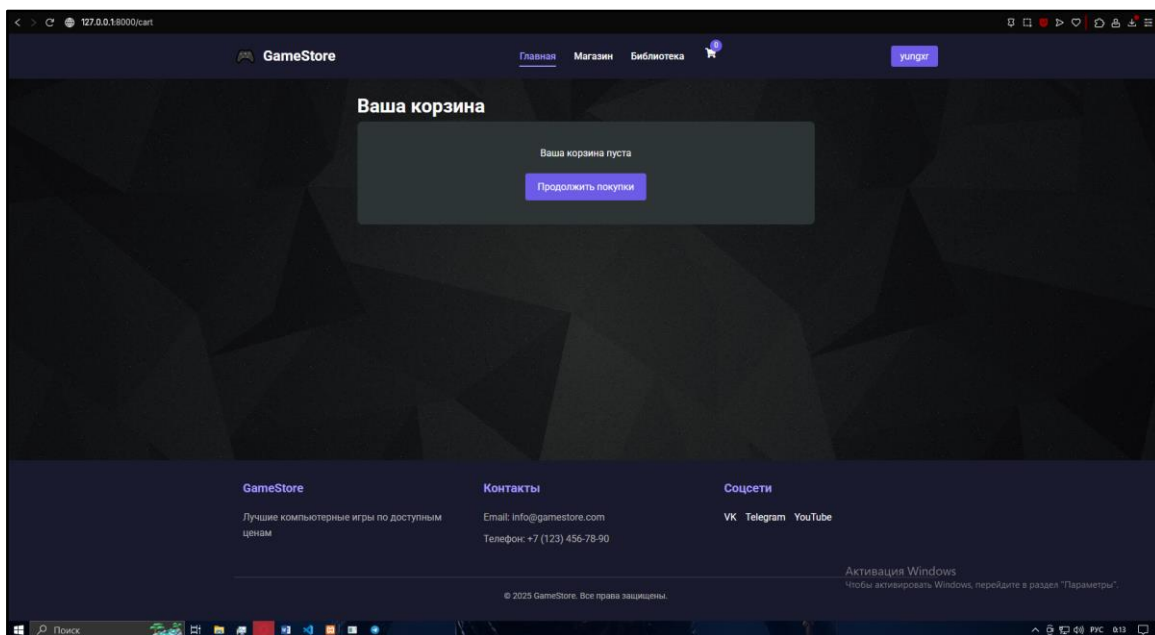


Рисунок 46 – Корзина пользователя

Для того чтобы оформить заказ пользователю нужно для начала пополнить баланс своего профиля на странице того же профиля и только тогда он сможет купить игру (рисунок 47-48).

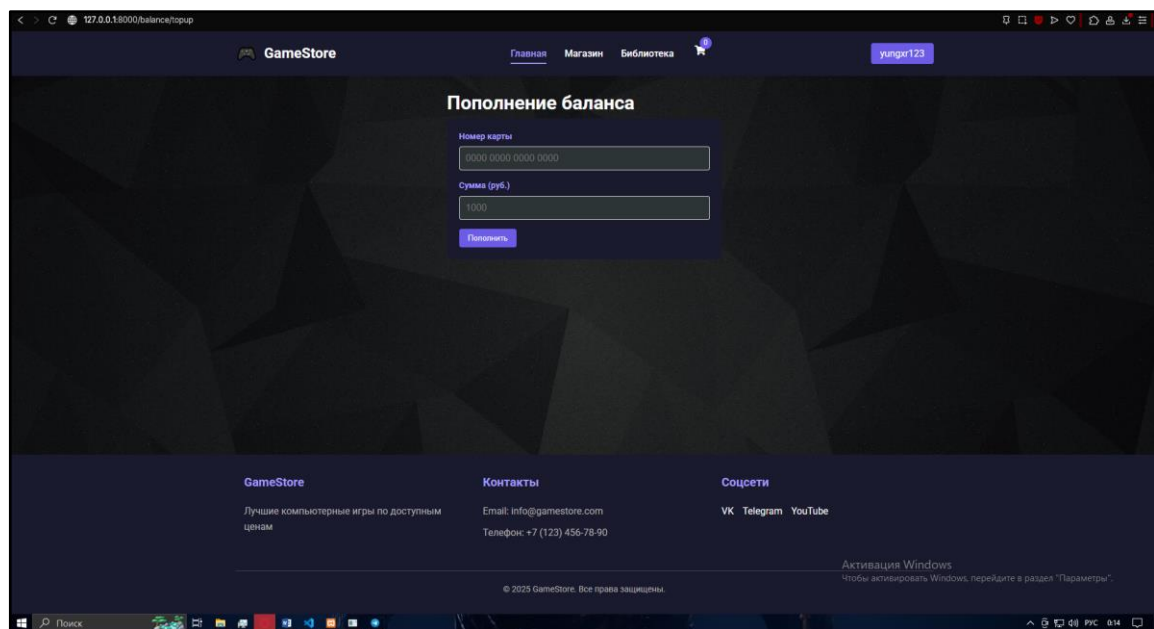


Рисунок 47 – Страница пополнения баланса

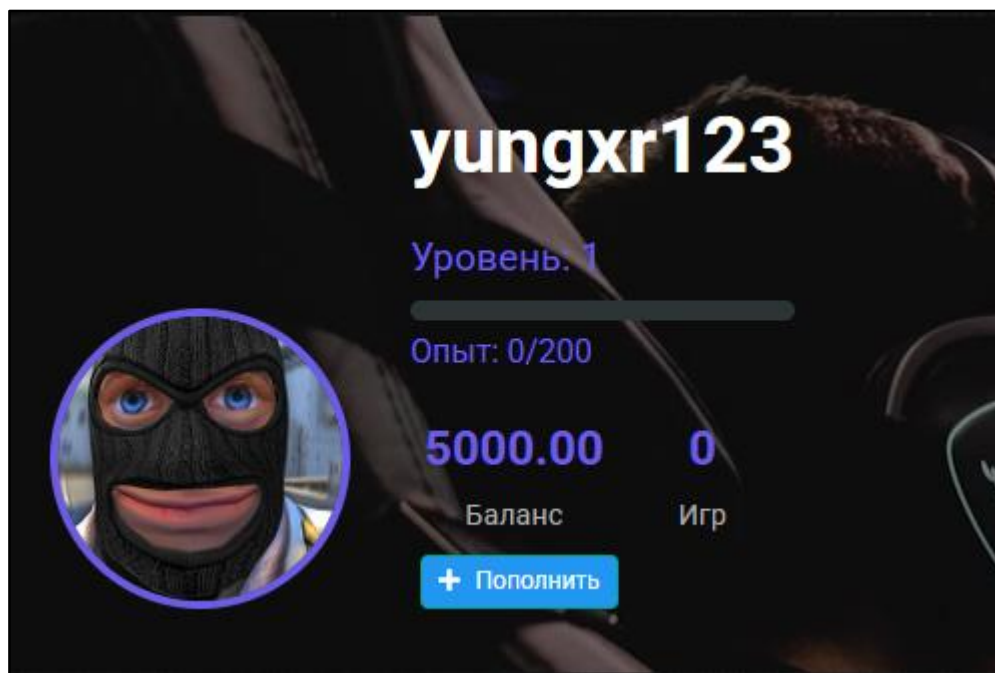


Рисунок 48 – Страница профиля с пополненным балансом

Теперь пользователь может на странице магазина положить игру в корзину, которую он хочет и оформить покупку (рисунок 49-50).

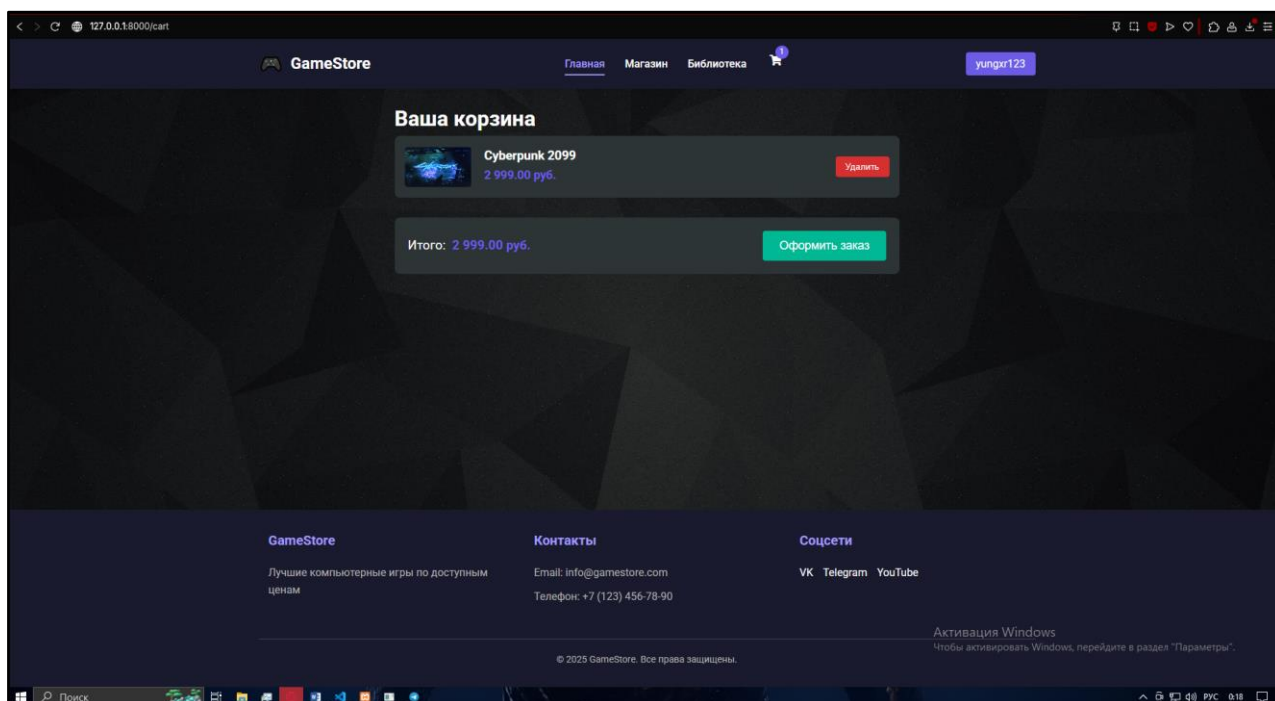


Рисунок 49 – Страница корзины с товаром в ней

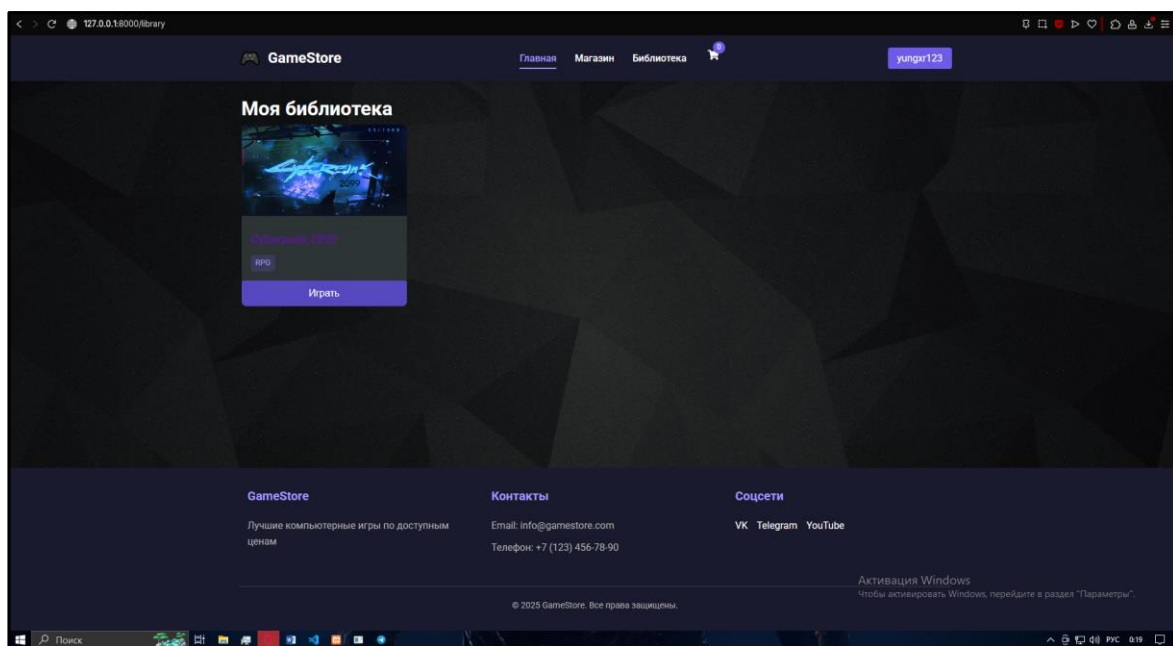


Рисунок 50 – Страница библиотеки с купленной игрой

Также на странице профиля, после оформления заказа, теперь можно увидеть саму игру в разделе библиотеки, отследить активность и повышение уровня (рисунок 51).

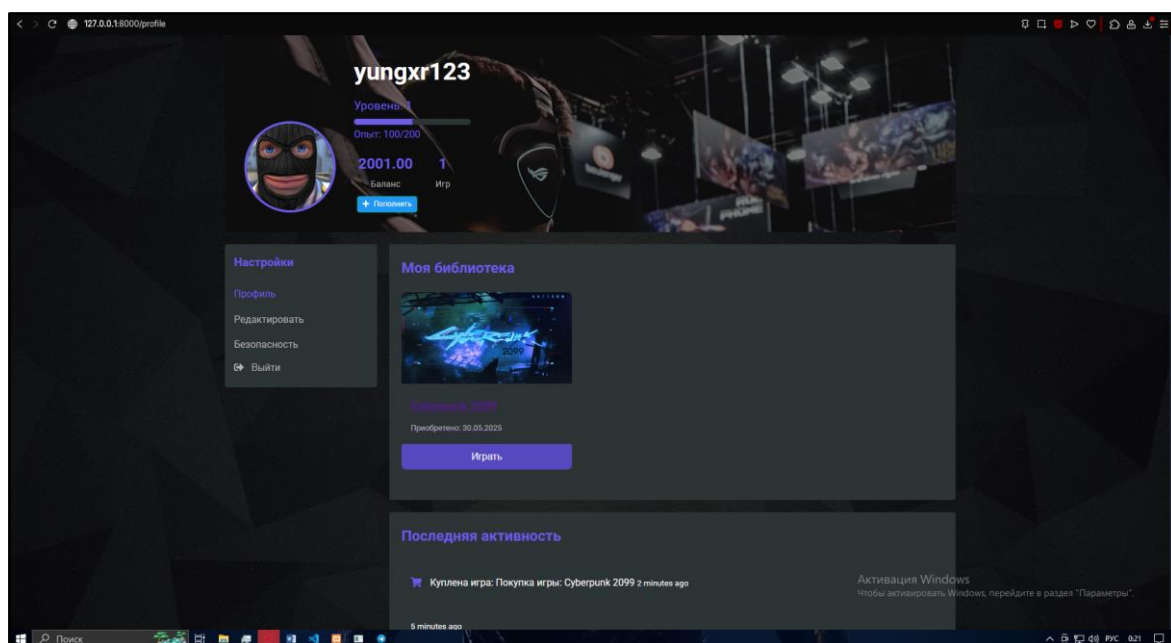


Рисунок 51 – Страница профиля после покупки игры

7 Экономическая часть

Создание интернет-магазина компьютерных игр – это сложный и многоаспектный процесс, требующий не только технической реализации, но и грамотного экономического обоснования. В условиях стремительного роста цифрового рынка и увеличения спроса на игровую продукцию онлайн-платформа становится эффективным инструментом для бизнеса, позволяющим расширить целевую аудиторию, минимизировать затраты на аренду и содержание физических торговых точек, а также обеспечить круглосуточный доступ к ассортименту для пользователей. Однако успешная реализация подобного проекта невозможна без детального анализа затрат, оценки трудоемкости разработки, а также прогнозирования потенциальной доходности.

7.1 Расчет трудоемкости разработки программного обеспечения

Трудоемкость считается ключевым показателем экономической эффективности (характеристика расхода труда). Позволяет оценить использование рабочего времени в результате производственной деятельности или при оказании услуг. Формула трудоемкости показывает количество труда, затрачиваемое на единицу продукции.

Суммарные затраты труда рассчитываются как сумма составных затрат труда по формуле (1):

$$\sum T = t_{\text{оп}} + t_{\text{ан}} + t_{\text{пр}} + t_{\text{тз}} + t_{\text{разб}} + t_{\text{т}} + t_{\text{д}} \quad (1)$$

Таблица 13 – Суммарные затраты времени

Вид работ	Часы (всего)	Машинное время
Анализ предметной области проекта и описание структуры	14	10
Анализ инструментов	12	12

Продолжение таблицы 13

Вид работ	Часы (всего)	Машинное время
Проектирование проекта и описание этапов проектирования	16	16
Создание технического задания на разработку проекта	15	10
Разработка программного продукта	60	60
Тестирование программного продукта	21	21
Подготовка документации	35	30
Итого	173	159

$$\sum T = 14 + 12 + 16 + 15 + 60 + 21 + 35 = 173$$

Из них:

Машинное время: 159 часов (разработка, тестирование).

Ручная работа: 14 часов (анализ, техническое задание, подготовка документации).

7.2 Затраты на оплату

Затраты на оплату (ЗОТ) труда разработчика ПО включают затраты на оплату труда и отчисления от фонда заработной платы.

Затраты на заработную плату определяются произведением часовой тарифной ставки разработчика и трудоемкости разработки программного продукта по следующей формуле:

Основная заработная плата рассчитывается в рублях по формуле (2):

$$З_{\text{осн}} = \sum t * TC_{\text{мес}} \quad (2)$$

Где $\sum t$ – суммарные затраты труда, вычисляемые по формуле (1), час.;

$TC_{\text{мес}}$ – часовая тарифная ставка, руб.

В расчете зарплаты учитываем модальную заработную плату по Иркутску – 80000 руб. Ставка за час – 484 руб. (при условии среднего количества рабочих часов при 40-часовой недели -165 ч.).

$$З_{\text{осн}} = 173 * 484 = 83\,732 \text{ руб.}$$

Налог на доходы физических лиц (НДФЛ-13%) — удержание производится из доходов, начисленных в пользу сотрудника.

Итоговый расчет заработной платы для разработчика ПО рассчитывается по формуле (3):

$$З_{\text{зп}} = З_{\text{осн}} - \left(\frac{З_{\text{осн}}}{100} * \text{ЕСН}\right) \quad (3)$$

$$З_{\text{зп}} = 83\,732 - 10\,885,16 = 72\,846,84 \text{ руб.}$$

7.3 Затраты на амортизацию оборудования

Амортизация — это процесс периодического переноса начальной стоимости основного средства или нематериального актива на производственные, коммерческие или общехозяйственные расходы — в зависимости от того, как этот актив используется.

Срок полезного использования согласно классификатору основных средств равен 2-3 года.

Для разработки использовался ПК изначальной стоимостью 89 000 рублей.

Расчет амортизации производится при помощи данных формул (4-7):

$$A_n = \frac{100\%}{K_k} = \frac{100\%}{3} = 33,3\% \quad (4)$$

где:

A_n – годовая норма амортизации;

K_k – срок полезного использования в соответствии с классификатором.

$$A_r = C_o * A_n = 89000 * 33,3\% = 29\,637 \quad (5)$$

где:

A_g – ежегодная сумма амортизации;

C_o – начальная стоимость оборудования.

$$A_m = \frac{A_g}{12 \text{ мес}}, = \frac{29637}{12} = 2469,75 \quad (6)$$

где:

A_m – ежемесячная сумма амортизации.

Далее необходимо сумму ежемесячной амортизации умножить на количество месяцев эксплуатации оборудования.

$$Z_{\text{амор}} = A_m * m = 2469,75 * 1 = 2469,75 \quad (7)$$

Из таблицы 1 указано, что машинное время работы на компьютере 159 часов что равно 1 месяц.

Стоимость амортизации ПК равна 2469,75 рублей.

7.4 Расчет затрат на электроэнергию

Для расчета затрат на электроэнергию первоочередное необходимо рассчитать расход электроэнергии оборудования по следующей формуле (8):

$$E = P * t \quad (8)$$

Где:

P – электрическая мощность в киловаттах;

t – время в часах.

К числу устройств, потребляющих электроэнергию, в ходе выполнения работы относится системный блок ПК (0,3 кВт), монитор (0,05 кВт), периферийные устройства (0,01 кВт). Суммарная потребляемая энергия для данных устройств составляет 0,36 кВт·ч.

Согласно таблице 1, машинное время работы ПК равно 159 часов. Стоимость одного киловатт-часа равна 1,58 рубля для городского населения Иркутской области. Общее потребление равно 57,24 кВт·ч.

Таким образом, затраты на электроэнергию составят:

$$C_z = E * C = 57,24 * 1,58 = 90,439 \quad (9)$$

					ДП.09.02.07-3.25.212.17. ПЗ	Лист
Изм.	Лис	№ докум.	Подпись	Дата		60

7.5 Затраты на материалы, израсходованные при проведении разработки и прочие

Затраты на материалы складывается из суммы стоимости материалов, которые сводятся в таблице 21.

Таблица 14 – Затраты на материалы

Наименование	Цена за единицу (руб.)	Кол-во (шт.)	Всего (руб.)
Бумага (500 листов)	360	103	74,16
Сшивание диплома	550	1	550
Итого			624,16

7.6 Расчет затрат на разработку программного обеспечения

Затраты на разработку программного обеспечения включают в себя:

- 1) затраты на зарплату разработчика ($Z_{зп}$);
- 2) затраты на амортизацию оборудования ($Z_{аморт}$);
- 3) затраты на эксплуатацию оборудования (электроэнергия) ($Z_{экспл}$);
- 4) затраты на материалы, израсходованные при проведении разработки (бумага, картридж, и т.п.) ($Z_{мат}$);
- 5) прочие затраты (использование платного ПО и тд.).

Затраты на разработку рассчитываются путем суммы всех затрат по формуле 10.

$$Z_{разр} = Z_{зп} + Z_{аморт} + Z_{экспл} + Z_{мат} + Z_{пр} \quad (10)$$

Таблица 15 – Общие расходы

Статьи затрат	Индекс	Сумма, руб.	Удельный вес затрат, %
Заработная плата	Зп.	72846,84	98
Амортизация оборудования	Заморт.	2469,75	1

Продолжение таблицы 15

Статьи затрат	Индекс	Сумма, руб.	Удельный вес затрат, %
Затраты на электроэнергию	Зэкспл.	90,439	0
Затраты на материалы	Змат.	624,16	1
Прочие затраты	Зпр.	-	0
Итого:	Зразр.	76031,189	100

7.7 Расчет цены

Определим расчётную цену при предполагаемом (плановом) размере прибыли на уровне 12%.

$$\text{Пр} = 76\,031,189 \times 12\% = 9\,123,74 \text{ руб.}$$

$$\text{НДС} = 0,2 \times (76\,031,189 + 9\,123,74) = 17\,030,95 \text{ руб.}$$

Итоговая цена:

$$\text{Ц} = 76\,031,189 + 9\,123,74 + 17\,030,95 = 102\,185,88 \text{ руб.}$$

Итоговая стоимость программного продукта составляет 102 185,88 рублей.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта по созданию интернет-магазина компьютерных игр, была достигнута поставленная в начале работы цель – разработка интернет-магазина компьютерных игр с широким функционалом, включающим удобный магазин, систему покупки игр, библиотеку, систему уровня и профиль пользователя.

Серверная часть интернет-магазина разрабатывалась с помощью фреймворка Laravel, который позволил создать качественную и безопасную систему.

Тестирование интернет-магазина было проведено с использованием модульного тестирования. Для этого была использована мощная система тестирования, основанная на PHPUnit, которая позволила проверить корректность работы отдельных модулей и функций программного продукта.

Спроектированные диаграммы помогли понять логику взаимодействия администратора и пользователя, и в последствии реализовать ее в виде кода.

Для разработки интернет-магазина был определен и реализован следующий функционал:

- 1) авторизация и регистрация пользователей;
- 2) поиск игр с возможностью сортировки по жанру;
- 3) оформление заказа;
- 4) отображение последней активности;
- 5) управление личными данными пользователя: смена имени, email и пароля;
- 6) система повышения уровня при покупке игр
- 7) своя личная библиотека, где будут храниться купленные игры пользователем

Подводя итог, были выполнены задачи, поставленные в начале работы:

- был проведен анализ предметной области;
- было разработано техническое задание;
- был проведен анализ программных средств разработки;
- были разработаны и протестированы основные модули интернет-магазина;
- была разработана программная документация.

Все поставленные цели и задачи дипломного проекта были успешно выполнены.

					ДП.09.02.07-3.25.212.17. ПЗ	Лист
Изм.	Лис	№ докум.	Подпись	Дата		64

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 getbootstrap.com – Buttons – URL:
https://getbootstrap.com/docs/5.0/components/buttons/ (дата обращения:
03.03.2025). Текст: электронный.

2 getbootstrap.com – Input group – URL:
https://getbootstrap.com/docs/5.0/forms/input-group/ (дата обращения: 02.03.2025).
Текст: электронный.

3 getbootstrap.com – Introduction – URL:
https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата обращения:
01.03.2025). Текст: электронный.

4 getbootstrap.com – Navbar – URL:
https://getbootstrap.com/docs/5.0/components/navbar/ (дата обращения:
02.03.2025). Текст: электронный.

5 getbootstrap.com – Tables – URL:
https://getbootstrap.com/docs/5.0/content/tables/ (дата обращения: 10.03.2025).
Текст: электронный.

6 laravel.com – Controllers – URL:
https://laravel.com/docs/12.x/controllers (дата обращения: 16.03.2025). Текст:
электронный.

7 laravel.com – CSRF Protection – URL: https://laravel.com/docs/12.x/csrf
(дата обращения: 18.03.2025). Текст: электронный.

8 laravel.com – Eloquent: Relationships – URL:
https://laravel.com/docs/12.x/eloquent-relationships (дата обращения: 17.03.2025).
Текст: электронный.

9 laravel.com – File Storage – URL:
https://laravel.com/docs/12.x/filesystem (дата обращения: 09.04.2025). Текст:
электронный.

10 laravel.com – Installation – URL: <https://laravel.com/docs/12.x> (дата обращения: 16.03.2025). Текст: электронный.

11 laravel.com – Laravel Socialite – URL: <https://laravel.com/docs/12.x/socialite> (дата обращения: 11.04.2025). Текст: электронный.

12 laravel.com – Middleware – URL: <https://laravel.com/docs/12.x/middleware> (дата обращения: 10.04.2025). Текст: электронный.

13 laravel.com – Routing – URL: <https://laravel.com/docs/12.x/routing> (дата обращения: 16.03.2025). Текст: электронный.

14 laravel.com – Testing: Getting Started – URL: <https://laravel.com/docs/12.x/testing> (дата обращения: 02.05.2025). Текст: электронный.

15 laravel.com – Validation – URL: <https://laravel.com/docs/12.x/validation> (дата обращения: 20.03.2025). Текст: электронный.

Приложение А Техническое задание

Министерство образования Иркутской области

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

Техническое задание

ИНТЕРНЕТ-МАГАЗИН КОМПЬЮТЕРНЫХ ИГР

Руководитель: _____ (С.А. Удальцов)
(подпись, дата)

Студент: _____ (И.Г. Пермяков)
(подпись, дата)

Иркутск 2025

					ДП.09.02.07-3.25.212.17. ПЗ	Лист
Изм.	Лис	№ докум.	Подпись	Дата		67

1 Общие сведения

Наименование работы: интернет-магазин компьютерных игр.

Исполнитель: студент иркутского авиационного техникума группы ВЕБ-21-2, Пермяков И.Г.

Разработка интернет-магазина проходит в рамках дипломного проекта по специальности 09.02.07 Информационные системы и программирование.

Сроки разработки интернет-магазина с 19.04.2025 по 27.05.2025 года.

2 Назначение и цели создания интернет-магазина

Назначение интернет-магазина компьютерных игр заключается в автоматизации процессов управления ассортиментом. Для пользователей – просмотр магазина, поиск, фильтрация, управление корзиной, оформление заказа, управление личными данными, просмотр библиотеки, отслеживание уровня своего профиля.

3 Требования к интернет-магазину в целом

3.1 Требования к структуре и функционированию интернет-магазина

Роли и права доступа интернет-магазина:

1. пользователь (авторизованный):
 - 1.1. просмотр товаров в магазине;
 - 1.2. редактирование профиля (смена имени, никнейма профиля, почты, пароля)
 - 1.3. пополнение баланса своего профиля
 - 1.4. функции поиска и фильтрации (по названию и жанру);
 - 1.5. добавление или удаление товаров в корзину;
 - 1.6. оформление заказа корзины;
 - 1.7. просмотр недавней активности;

1.8. отслеживание своего уровня профиля

1.9. выход из аккаунта пользователя.

3.2 Требования к надежности

Для обеспечения надежности необходимо проверять корректность получаемых данных и реализовать валидность полей. Входные данные поступают в виде значений с клавиатуры. Эти значения отображаются в отдельных полях таблицы.

3.3 Требования к безопасности

Для обеспечения безопасности в интернет-магазине, необходимо реализовать разграничение прав доступа, возможность восстанавливать данные. Создать шифрование информации в базе данных.

3.4 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы

Минимальные системные требования для сервера:

- Операционная система: Windows 10;
- Процессор: Intel Core i3 2100CPU / AMD FX-6300;
- Оперативная память: 8Гб;
- SSD: 100Гб.

4 Требования к документированию

Основным документом, регламентирующим использование интернет-магазина, является руководство пользователя.

Основным документом, регламентирующим разработку интернет-магазина, является техническое задание.

					ДП.09.02.07-3.25.212.17. ПЗ	Лист
Изм.	Лис	№ докум.	Подпись	Дата		69

5 Состав и содержание работ по созданию интернет-магазина

В таблице 1 представлены плановые сроки начала и окончания работы по созданию интернет-магазина.

Таблица 1 – Плановые сроки по созданию интернет-магазина

№	Этап разработки	Срок выполнения
1	Предпроектное исследование	10.03.25

Продолжение таблицы 1

№	Этап разработки	Срок выполнения
2	Разработка технического задания	12.03.25
3	Проектирование программного продукта	15.03.25
4	Разработка программного продукта	30.04.25
5	Тестирование и отладка программного продукта	06.05.25
6	Разработка руководства пользователя	11.05.25
7	Расчет стоимости разработки программного продукта	20.05.25

Приложение Б – Листинг кода шаблона страниц

```
<!DOCTYPE html>
<html lang="ru">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>GameStore - Интернет-магазин компьютерных игр</title>
  <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500;700&di
splay=swap" rel="stylesheet">
  <link href="/css/app.css" rel="stylesheet">
  <link      rel="stylesheet"      href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
</head>

<body class="site-container">
  <header class="header">
    <div class="container">
      <div class="header__inner">
        <a href="/" class="logo">
          <span class="logo__icon">🎮</span>
          <span class="logo__text">GameStore</span>
        </a>

        <nav class="nav">
          <a href="/" class="nav__link active">Главная</a>
          <a href="/shop" class="nav__link">Магазин</a>
```

```

    <a href="/library" class="nav__link">Библиотека</a>
    <a href="{{ route('cart.index') }}" class="cart-link">
        <i class="fas fa-shopping-cart"></i>
        @auth
            <span class="cart-count">{{ auth()->user()->cartItems()->count()
        }}</span>
        @endauth
    </a>
</nav>

<div class="user-menu">
    <a href="{{ route('profile') }}" class="user-menu__link">
        <span>{{ Auth::user()->username }}</span>
    </a>
</div>
</div>
</div>
</header>

<main class="site-content">
    @yield('content')
</main>

<footer class="footer">
    <div class="container">
        <div class="footer__inner">
            <div class="footer__col">
                <h3 class="footer__title">GameStore</h3>

```


<p class="footer__text">Лучшие компьютерные игры по доступным ценам</p>

</div>

<div class="footer__col">

<h3 class="footer__title">Контакты</h3>

<p class="footer__text">Email: info@gamestore.com</p>

<p class="footer__text">Телефон: +7 (123) 456-78-90</p>

</div>

<div class="footer__col">

<h3 class="footer__title">Соцсети</h3>

<div class="social-links">

VK

Telegram

YouTube

</div>

</div>

</div>

<div class="footer__copyright">

© 2025 GameStore. Все права защищены.

</div>

</div>

</footer>

<script>

// Предпросмотр аватара перед загрузкой

document.getElementById('avatar').addEventListener('change', function(e) {

const file = e.target.files[0];

if (file) {

const reader = new FileReader();

reader.onload = function(e) {

```

        document.getElementById('avatar-preview').src = e.target.result;
    }
    reader.readAsDataURL(file);
}
});
</script>
</body>

</html>

```

| | | | | | | |
|------|-----|----------|---------|------|-----------------------------|------|
| | | | | | ДП.09.02.07-3.25.212.17. ПЗ | Лист |
| | | | | | | |
| Изм. | Лис | № докум. | Подпись | Дата | | 74 |

Приложение В – Листинг кода контроллера поиска

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\Product;
use App\Models\Genre;
class SearchController extends Controller
{
    public function index(Request $request) // Вывод результатов поиска
    {
        $query = $request->input('query'); // Поисковый запрос
        $sort = $request->input('sort', 'newest'); // Сортировка
        $minPrice = $request->input('min_price'); // Минимальная цена
        $maxPrice = $request->input('max_price'); // Максимальная цена
        $genreId = $request->input('genre_id'); // id жанра
        // Запрос к базе данных
        $products = Product::query()
        // Фильтр по поисковому запросу (если указан)
        ->when($query, function($q) use ($query) {
            $q->where(function($queryBuilder) use ($query) {
                // Поиск в названии или описании символов из запроса
                $queryBuilder->where('name', 'like', "%$query%")
                ->orWhere('description', 'like', "%$query%");
            });
        })
        // Фильтр по жанру (если указан)
        ->when($genreId, function($q) use ($genreId) {
            $q->whereHas('genres', function($q) use ($genreId) {
```

| | | | | | | |
|------|-----|----------|---------|------|-----------------------------|------|
| | | | | | ДП.09.02.07-3.25.212.17. ПЗ | Лист |
| | | | | | | |
| Изм. | Лис | № докум. | Подпись | Дата | | 75 |

```

$q->where('genres.id', $genreId);
});
})
// Фильтр по минимальной цене (если указана)
->when(!is_null($minPrice), function($q) use ($minPrice) {
    $q->where('price', '>=', (float)$minPrice);
})
// Фильтр по максимальной цене (если указана)
->when(!is_null($maxPrice), function($q) use ($maxPrice) {
    $q->where('price', '<=', (float)$maxPrice);
})
->with(['genres']) // Загрузка жанров
->withAvg('ratings', 'rating') // Добавление рейтинга
->withCount('ratings') // Добавление количества оценок
->orderBy($this->getSortColumn($sort), $this->getSortDirection($sort)) //
Сортировка результатов
->get();
$title = $genreId // Заголовок страницы в зависимости от поиска
? "Игры в жанре: " . Genre::find($genreId)->name
: ($query ? "Результаты поиска: '$query'" : "Все товары");
// Возвращаем представление с данными
return view('search_results', [
    'products' => $products, // Найденные товары
    'searchQuery' => $query, // Поисковый запрос
    'title' => $title, // Заголовок страницы
    'currentSort' => $sort, // Текущий способ сортировки
    'minPrice' => $minPrice, // Минимальная цена
    'maxPrice' => $maxPrice, // Максимальная цена

```

```

'genres' => Genre::withCount('products')->get(), // Все жанры с количеством
товаров
'currentGenre' => $genreId ? Genre::find($genreId) : null // Текущий жанр (если
выбран)
]);
}

protected function getSortColumn($sort) // Поле для сортировки
{
return match($sort) {
'price_asc', 'price_desc' => 'price', // Сортировка по цене
'rating' => 'ratings_avg_rating', // Сортировка по рейтингу
default => 'created_at' // Сортировка по дате (по умолчанию)
};
}

protected function getSortDirection($sort) // Направление сортировки
{
return match($sort) {
'price_asc' => 'asc', // Цена по возрастанию
'price_desc' => 'desc', // Цена по убыванию
'rating' => 'desc', // Рейтинг по убыванию
default => 'desc'
};
}

public function byGenre(Request $request, Genre $genre) // Поиск по конкретному
жанру
{
// Получение параметров сортировки и фильтрации
$sort = $request->input('sort', 'newest');
$minPrice = $request->input('min_price');

```

```

$maxPrice = $request->input('max_price');
// Получение товаров для указанного жанра и фильтров
$products = $genre->products()
->when($minPrice !== null, function($q) use ($minPrice) {
    $q->where('price', '>=', (float)$minPrice);
})
->when($maxPrice !== null, function($q) use ($maxPrice) {
    $q->where('price', '<=', (float)$maxPrice);
})
->with(['genres'])
->withAvg('ratings', 'rating')
->orderBy($this->getSortColumn($sort), $this->getSortDirection($sort))
->get();
// Возвращение представления с результатами
return view('search_results', [
    'products' => $products,
    'title' => "Игры в жанре: {$genre->name}",
    'searchQuery' => null,
    'currentSort' => $sort,
    'minPrice' => $minPrice,
    'maxPrice' => $maxPrice,
    'genres' => Genre::withCount('products')->get(),
    'currentGenre' => $genre // Передаём текущий жанр в представление
]);
}
}

```