Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра информационных систем и технологий

**ИНТЕРНЕТ-СЕРВИС**

**Магазин видеоигр**

**КОНЦЕПЦИЯ**

Разработал: Берёзка Вадим Сергеевич

ФИТ, 2 курс, группа 3

Проверил: Заянковский Дмитрий Владимирович

Минск 2024

Содержание

[Введение 3](#_Toc179325405)

[Постановка задачи и анализ аналогичных решений 4](#_Toc179325406)

[Функциональные возможности сервиса 7](#_Toc179325407)

[Соглашение об уровне услуг 12](#_Toc179325408)

[Логическая и физическая схема базы данных 15](#_Toc179325409)

[Архитектура сервиса 20](#_Toc179325410)

[Процесс создания сервиса 24](#_Toc179325411)

[Источники 26](#_Toc179325413)

# Введение

Проект «Интернет-магазин видеоигр» направлен на создание платформы, где пользователи смогут удобно и быстро приобретать видеоигры различных жанров и платформ.

С увеличением популярности видеоигр и их доступности через интернет, возникает необходимость в удобных и безопасных каналах для их приобретения. Актуальность проекта обусловлена ростом числа геймеров и необходимостью предложить им качественный сервис для покупки игр.

Перечень аналогичных решений:

* ­[Steam](https://store.steampowered.com/?l=russian) – популярная платформа для цифровой дистрибуции игр;
* [Epic Games Store](https://store.epicgames.com/ru/) – магазин, предлагающий эксклюзивные игры и акции.

Целевая аудитория проекта включает в себя:

* геймеров всех возрастов, ищущих новые игры;
* родителей, желающих приобрести игры для детей.

Основная цель проекта – создать удобный и интуитивно понятный интернет-магазин для продажи видеоигр, который обеспечит высокий уровень обслуживания пользователей и увеличит продажи за счет широкого ассортимента и привлекательных предложений.

Задачи для достижения цели:

* разработка функционального интерфейса для пользователей;
* обеспечение безопасных методов оплаты;
* реализация мобильной версии магазина;
* реализация десктопной версии магазина;
* создание базы данных игр с описанием и изображениями.

Роли пользователей:

* покупатель – пользователь, который ищет и покупает игры;
* администратор – управляет контентом магазина и обрабатывает заказы;
* разработчик – отвечает за техническое обеспечение и поддержку платформы.

Краткое описание архитектуры программного обеспечения

Frontend: интерфейс пользователя, реализованный с использованием React, React-Redux.

Для сетевых запросов будет использоваться Firebase.

# Постановка задачи и анализ аналогичных решений

*Аналог №1 - Steam (https://store.steampowered.com):*

В сервисе Steam реализован обширный каталог видеоигр, включая AAA-игры, инди-проекты и DLC. Пользователи могут искать игры по жанрам, популярности и оценкам, а также получать доступ к различным скидкам и акциям. Подобный функционал может быть использован в интернет-магазине видеоигр для улучшения пользовательского опыта при покупке видеоигр.

Возможности, которые можно использовать в собственном проекте:

1. Обширный каталог игр - создание разнообразного каталога, включающего AAA-игры, инди-проекты и DLC; возможность фильтрации по жанрам, платформам и ценам для упрощения поиска.

2. Поиск по критериям - внедрение многоуровневой системы поиска, позволяющей пользователям находить игры по популярности, рейтингу и другим критериям.

3. Скидки и акции - регулярные акции и специальные предложения, которые помогут увеличить продажи и привлекут больше пользователей.

4. Социальные функции - возможность общения между игроками, обсуждения игр и оценки контента через отзывы и рейтинги.

5. Поддержка различных языков - многоязычная поддержка для привлечения международной аудитории.

Возможности, которые могут потребовать доработки:

1. Персонализированные рекомендации - система, основанная на анализе предпочтений пользователей, может потребовать доработки алгоритмов для более точного подбора игр.

2. Система лояльности и накопительных баллов - внедрение системы бонусов и скидок для постоянных клиентов потребует анализа и разработки соответствующих алгоритмов.

*Аналог №2 - GOG.com (https://www.gog.com):*

Платформа GOG.com предлагает DRM-free игры, что позволяет пользователям скачивать и сохранять их без ограничений. Поиск осуществляется по жанрам, популярности и новизне, а также есть возможность находить классические игры. Этот подход может быть адаптирован в интернет-магазине видеоигр для продажи игр с максимальной свободой использования.

Возможности, которые можно использовать в собственном проекте:

1. DRM-free игры - предоставление пользователям возможности скачивать и сохранять игры без ограничений, что повышает уровень удовлетворенности.

2. Классические игры - создание раздела для классических игр, что привлечет пользователей, ностальгирующих по старым титулам.

3. Система пожеланий - возможность добавления игр в список желаемого для удобства пользователей.

4. Поддержка сообщества - функции для взаимодействия пользователей друг с другом, такие как форумы и обсуждения.

Возможности, которые могут потребовать доработки:

1. Интеграция с мобильными приложениями - разработка мобильного приложения для доступа к играм и функциям магазина, что может потребовать значительных ресурсов.

2. Кроссплатформенная поддержка - обеспечение совместимости с различными устройствами и платформами может потребовать дополнительных усилий по тестированию.

3. Обратная связь и поддержка - создание системы поддержки и обратной связи для пользователей, чтобы они могли легко решать возникающие проблемы.

*Аналог №3 - Epic Games Store (https://www.epicgames.com/store):*

Epic Games Store предоставляет пользователям доступ к эксклюзивным играм и бесплатным раздачам. Сервис предлагает поиск по категориям и жанрам, а также регулярные распродажи. Этот функционал может быть интегрирован в интернет-магазин видеоигр для привлечения пользователей через эксклюзивные предложения и скидки.

Возможности, которые можно использовать в собственном проекте:

1. Эксклюзивные предложения - партнерство с разработчиками для предоставления уникальных игр, которые недоступны на других платформах.

2. Бесплатные игры - регулярные раздачи игр, что может значительно увеличить приток пользователей и их вовлеченность.

3. Актуальные распродажи - проведение сезонных и тематических распродаж, чтобы привлечь внимание к каталогу и увеличить объем продаж.

Возможности, которые могут потребовать доработки:

1. Улучшенный интерфейс - интуитивно понятная навигация и поиск по категориям и жанрам, чтобы пользователи могли легко находить интересующие их игры.

2. Интеграция с игровыми клиентами - возможность загрузки и управления играми через собственный клиент, что улучшает пользовательский опыт.

*Аналог №4 - Ubisoft Store (https://store.ubi.com):*

Ubisoft Store предлагает пользователям доступ к играм и контенту от одного из крупнейших разработчиков видеоигр. Сервис позволяет искать игры по жанрам, платформам и акциям, а также предлагает подписку Ubisoft+ для доступа к большому количеству игр. Этот подход может быть использован в интернет-магазине видеигр для создания уникального предложения для поклонников игр Ubisoft и повышения лояльности клиентов.

Возможности, которые можно использовать в собственном проекте:

1. Подписочная модель - внедрение подписки, подобной Ubisoft+, для предоставления доступа к обширной библиотеке игр за фиксированную плату.

2. Геймификация - внедрение элементов геймификации, таких как достижения и награды за выполнение определённых действий в магазине, что стимулирует пользователей к активному поведению.

Возможности, которые могут потребовать доработки:

1. Интеграция с Ubisoft Connect - связывание магазина с учетными записями Ubisoft для улучшения пользовательского опыта и управления играми.

2. Социальные функции - интеграция социальных возможностей, таких как возможность делиться достижениями и приглашать друзей, что может повысить вовлеченность.

*Сценарий использования «Интернет-магазин видеоигр»*

Любой пользователь интернета может подключиться к Интернет-магазину видеоигр, введя правильный URI в адресной строке браузера или запустив клиентское приложение. По умолчанию он соединяется с сервисом как пользователь, имеющий роль Guest (гость/пользователь по умолчанию).

В режиме Guest пользователю доступны основные функции, такие как просмотр каталога игр, ознакомление с «Соглашением об использовании сервиса» и возможность регистрации. Пользователь может:

- Просматривать описания игр и их системные требования.

- Изучать отзывы других игроков.

- Следить за текущими акциями и распродажами.

После регистрации пользователь автоматически переключается в режим User (зарегистрированный пользователь). Зарегистрированному пользователю доступны все функции, доступные роли Guest, а также дополнительные возможности, позволяющие ему:

- Добавлять игры в собственный список желаемого (Wishlist).

- Совершать покупки игр и контента через встроенную систему оплаты.

- Участвовать в обсуждениях на форумах и оставлять отзывы о купленных играх.

- Получать доступ к функциям социальной сети, включая добавление друзей и общение в чате.

Для пользователей, имеющих роль Developer или Publisher (разработчик или издатель), доступны возможности управления своими играми, включая:

- Публикацию новых игр и обновлений на платформе.

- Доступ к аналитике и статистике продаж для оценки эффективности своих игр.

Кроме того, пользователям доступны функции сообщества, которые включают:

- Участие в группах и мероприятиях.

- Доступ к пользовательским модификациям (модам) и контенту, созданному другими пользователями.

- Взаимодействие с системой достижений и получением наград за выполнение определенных условий в играх.

# Функциональные возможности сервиса

1.1. Пользовательские роли

В нашем интернет-магазине видеоигр предусмотрены несколько пользовательских ролей, каждая из которых имеет свои уникальные функции и возможности. Ниже представлены основные роли, доступные на платформе:

1. Гость (Guest)

Пользователи, не зарегистрированные на сайте, получают доступ к базовым функциям. Гости могут просматривать каталог игр, изучать описания продуктов и ознакомиться с акциями. Эта роль позволяет пользователям ознакомиться с сервисом и его предложениями без необходимости создания учетной записи.

2. Зарегистрированный пользователь (User)

После регистрации пользователи переходят в статус "Зарегистрированный пользователь". Эта роль предоставляет доступ ко всем функциям для гостей, а также дополнительные возможности, такие как:

- создание и управление списком желаемого (Wishlist);

- совершение покупок с использованием безопасных методов оплаты;

- участие в обсуждениях на форумах и возможность оставлять отзывы о купленных играх.

3. Разработчик (Developer)

Разработчики имеют доступ к специальному интерфейсу для управления своими играми на платформе. К основным функциям этой роли относятся:

- публикация новых игр и обновлений;

- настройка цен и акций для своих продуктов;

- доступ к аналитическим данным о продажах и пользователях, что позволяет оценивать эффективность игр.

4. Администратор (Admin)

Администраторы имеют полный доступ к функционалу платформы и отвечают за ее бесперебойную работу. К их обязанностям относятся:

- мониторинг активности пользователей и управление учетными записями;

- поддержка пользователей, решение проблем и управление запросами на возврат;

- обновление контента на сайте, включая добавление новых игр и акций.

5. Модератор (Moderator)

Модераторы занимаются поддержанием порядка в сообществе пользователей. Их функции включают:

- мониторинг обсуждений на форумах и социальных платформах;

- удаление нежелательного или оскорбительного контента;

- обеспечение соблюдения правил сообщества и поддержание дружелюбной атмосферы.

1.2. Функциональное наполнение пользовательских ролей



Рисунок 1 - Диаграмма вариантов использования для гостя и пользователя

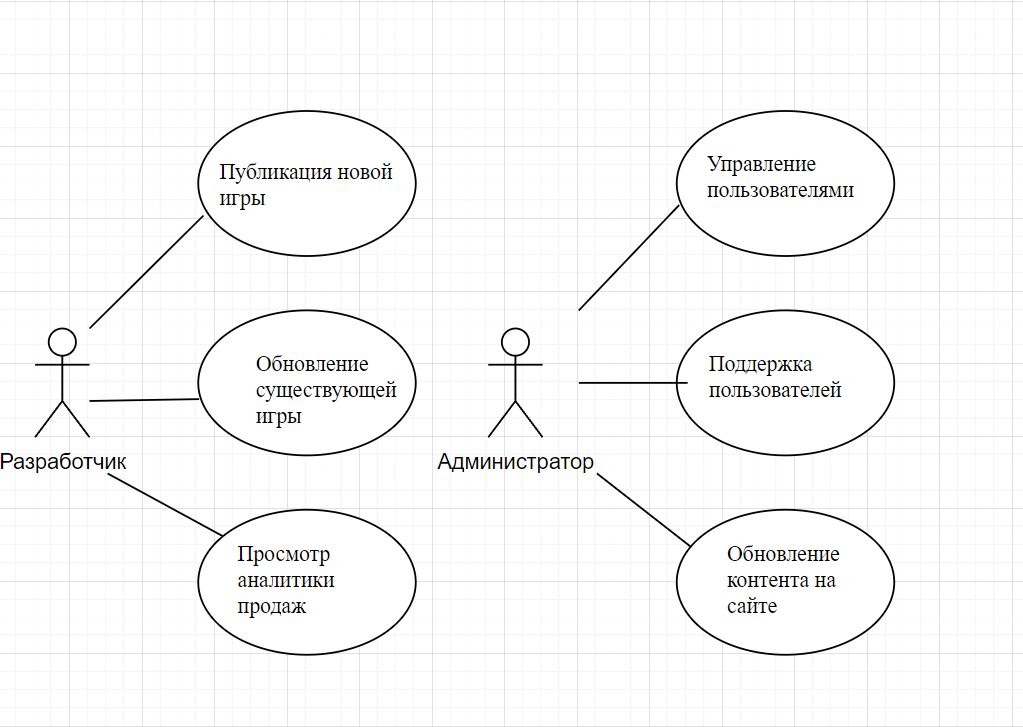


Рисунок 2 - Диаграмма вариантов использования для разработчика и администратора

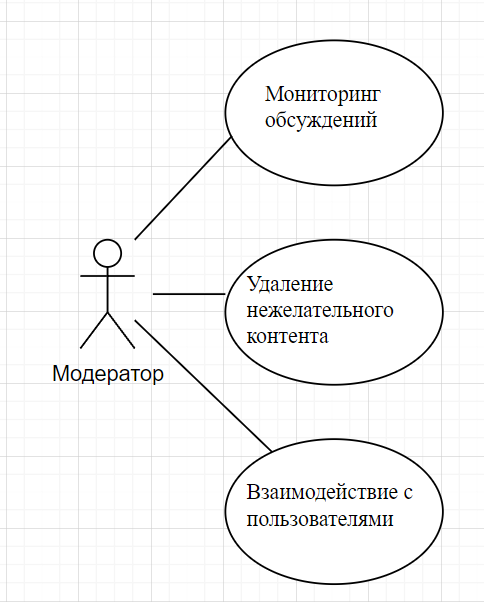


Рисунок 3 - Диаграмма вариантов использования для модератора

1. Гость (Guest)

|  |  |
| --- | --- |
| Название прецедента | Описание прецедента |
| Просмотр каталога игр | Гость может просматривать доступные игры, фильтровать по категориям и видеть акции. |
| Ознакомление с описаниями | Гость может кликнуть на игру для просмотра её описания, системных требований и отзывов. |
| Регистрация на сайте | Гость может создать учетную запись для получения дополнительных функций. |

Подробное пояснение прецедентов:

- просмотр каталога игр: гость получает доступ к полному списку игр, который может быть отсортирован по жанрам, популярности и новизне;

- ознакомление с описаниями: при нажатии на игру открывается страница с детальным описанием, скриншотами и видео, что помогает пользователю принять решение о покупке;

- регистрация на сайте: гость может заполнить форму регистрации, указав личные данные, что откроет доступ к более широким возможностям.

2. Зарегистрированный пользователь (User)

|  |  |
| --- | --- |
| Название прецедента | Описание прецедента |
| Добавление в список желаемого | Пользователь может добавлять игры в свой личный список желаемого для дальнейшего просмотра. |
| Оформление покупки | Пользователь может выбрать игру и пройти процесс оформления заказа, включая выбор способа оплаты. |
| Оставление отзыва | Пользователь может написать отзыв о купленной игре, который будет виден другим пользователям. |
| Участие в форумах | Пользователь может участвовать в обсуждениях на форумах, задавать вопросы и получать ответы. |

Подробное пояснение прецедентов:

- добавление в список желаемого: пользователь может легко добавлять игры в свой список желаемого, что упрощает поиск интересующих товаров;

- оформление покупки: процесс включает выбор игры, добавление в корзину, выбор метода оплаты и подтверждение заказа;

- оставление отзыва: после завершения игры пользователь может оставить отзыв, делясь впечатлениями и помогая другим в выборе;

- участие в форумах: пользователь может задавать вопросы, делиться опытом и получать советы от других игроков.

3. Разработчик (Developer)

|  |  |
| --- | --- |
| Название прецедента | Описание прецедента |
| Публикация новой игры | Разработчик может загрузить новую игру на платформу и заполнить описание. |
| Обновление существующей игры | Разработчик может вносить изменения в уже опубликованные игры, включая обновления контента. |
| Просмотр аналитики продаж | Разработчик может просматривать отчеты о продажах своей игры и анализировать данные пользователей. |

Подробное пояснение прецедентов:

- публикация новой игры: процесс включает загрузку файлов игры, описание, скриншоты и настройку цен;

- обновление существующей игры: разработчик может добавлять новые функции или исправления ошибок, чтобы улучшить игровой процесс;

- просмотр аналитики продаж: разработчик имеет доступ к статистике, что помогает ему понять, как игра воспринимается пользователями.

4. Администратор (Admin)

|  |  |
| --- | --- |
| Название прецедента | Описание прецедента |
| Управление пользователями | Администратор может добавлять, редактировать или удалять учетные записи пользователей. |
| Поддержка пользователей | Администратор отвечает на запросы пользователей и помогает решать проблемы. |
| Обновление контента на сайте | Администратор управляет контентом на сайте, добавляя новые игры и акции. |

Подробное пояснение прецедентов:

- управление пользователями: администратор имеет возможность контролировать доступ к сервису и управлять учетными записями для обеспечения безопасности;

- поддержка пользователей: администратор предоставляет помощь и поддержку, обеспечивая высокий уровень обслуживания клиентов;

- обновление контента на сайте: администратор отвечает за актуальность информации на сайте, что важно для привлечения клиентов.

5. Модератор (Moderator)

|  |  |
| --- | --- |
| Название прецедента | Описание прецедента |
| Мониторинг обсуждений | Модератор следит за обсуждениями на форумах, обеспечивая соблюдение правил. |
| Удаление нежелательного контента | Модератор удаляет оскорбительные сообщения или спам. |
| Взаимодействие с пользователями | Модератор отвечает на вопросы пользователей и помогает поддерживать порядок. |

Подробное пояснение прецедентов:

- мониторинг обсуждений: модератор следит за участниками форумов, чтобы поддерживать дружелюбную атмосферу и предотвращать конфликты;

- удаление нежелательного контента: модератор имеет право удалять сообщения, которые нарушают правила сообщества, что помогает сохранять качество обсуждений;

- взаимодействие с пользователями: модератор активно отвечает на вопросы и помогает пользователям, создавая поддерживающую атмосферу.

# Соглашение об уровне услуг

1. Стороны соглашения

Настоящее Соглашение об уровне услуг (SLA) заключается между:

- Интернет-магазин видеоигр (в дальнейшем «Поставщик услуг»), юридическое лицо, зарегистрированное по адресу ул. Свердлова 13а, и

- пользователями (в дальнейшем «Клиенты»), которые используют сервис Интернет-магазин видеоигр.

2. Сроки действия соглашения

Данное соглашение вступает в силу с 01.09.2024 и остается в силе до тех пор, пока Клиенты используют услуги Поставщика услуг или до момента его отмены одной из сторон с предварительным уведомлением за 30 дней.

3. Расписание работы сервиса

Сервис Интернет-магазин видеоигр работает круглосуточно, семь дней в неделю, за исключением периодов планового технического обслуживания. Поставщик услуг будет уведомлять Клиентов о времени и длительности плановых работ не менее чем за 48 часов до их начала.

4. Доступ к службам поддержки сервиса

Клиенты могут получить доступ к службе поддержки через следующие каналы:

- электронная почта: support@Vadimka.com

- телефон: +375299780571

- онлайн-чат: доступен на сайте с 02.09.2024.

Часы работы службы поддержки: с 9:00 до 18:00 по МСК, с понедельника по пятницу.

5. Процедура сообщения о дефектах сервиса и порядок исправления дефектов

Клиенты должны сообщать о любых дефектах или проблемах сервиса через службу поддержки. Поставщик услуг обязуется:

- подтвердить получение сообщения о дефекте в течение 1 рабочего часа.

- предоставить оценку времени исправления дефекта в течение 4 рабочих часов.

- исправить дефект в разумные сроки, в зависимости от его сложности и влияния на пользователей.

6. Процедура запроса на изменение сервиса и порядок ответа на запрос

Клиенты могут направлять запросы на изменение сервиса через службу поддержки. Поставщик услуг обязуется:

- подтвердить получение запроса в течение 1 рабочего часа.

- провести анализ запроса и предоставить ответ в течение 3 рабочих дней.

- при необходимости, провести обсуждение с Клиентом для уточнения деталей запроса.

7. Гарантированное время отклика на запрос к сервису

Поставщик услуг обязуется предоставить гарантированное время отклика на запросы Клиентов в течение 1 рабочего часа для срочных запросов и в течение 4 рабочих часов для обычных запросов.

8. Гарантированная доступность сервиса

Поставщик услуг гарантирует доступность сервиса на уровне 99,5% в месяц, что означает, что сервис может быть недоступен не более чем 3,6 часов в месяц по причинам, не связанным с плановым обслуживанием.

9. Описание способов оплаты

Клиенты могут оплачивать услуги Интернет-магазина видеоигр следующими способами:

- кредитные и дебетовые карты: Visa, MasterCard, American Express.

- электронные кошельки: PayPal, Яндекс.Деньги и другие.

- банковские переводы: по запросу.

Оплата производится в соответствии с тарифами, указанными на сайте в момент оформления заказа.

10. Контакты администрации сервиса

Для получения дополнительной информации или в случае вопросов, связанных с данным соглашением, Клиенты могут обращаться по следующим контактам:

- электронная почта: admin@Vadimka.com

- телефон: +375299780571

Организационное обеспечение информационной системы

Интернет-сервис Интернет-магазин видеоигр принадлежит компании BerezkaCor, зарегистрированной по адресу ул. Свердлова 13а. Компания специализируется на разработке и продаже видеоигр, а также предоставляет услуги цифровой дистрибуции через свой интернет-магазин.

Организационная структура

Для эффективного управления и поддержки интернет-сервиса Интернет-магазин видеоигр была создана организационная структура, состоящая из следующих отделов:

1. Отдел разработки

Назначение: этот отдел отвечает за проектирование, разработку и обновление функционала интернет-сервиса.

Состав:

- руководитель отдела разработки: Берёзка Вадим Сергеевич — координация работы команды, планирование задач.

- разработчики: 3-5 специалистов — разработка программного обеспечения, реализация новых функций.

- дизайнер: Берёзка Вадим Сергеевич — создание интерфейса и графического оформления сервиса.

- тестировщики: 2 специалиста — тестирование функционала, выявление и исправление ошибок.

2. Отдел поддержки пользователей

Назначение: обеспечение качественной поддержки клиентов, обработка запросов и решение проблем пользователей.

Состав:

- руководитель отдела поддержки: Берёзка Вадим Сергеевич — управление командой, анализ эффективности работы.

- специалисты по поддержке: 4-6 человек — помощь пользователям через электронную почту, телефон и онлайн-чат.

- модераторы: 2 специалиста — контроль за корректностью обсуждений на форумах, поддержание порядка в сообществе.

3. Отдел маркетинга

Назначение: разработка и реализация маркетинговых стратегий для продвижения интернет-сервиса и привлечения пользователей.

Состав:

- руководитель отдела маркетинга: Берёзка Вадим Сергеевич — планирование маркетинговых кампаний, анализ рынка.

- специалисты по контенту: 2-3 человека — создание рекламных материалов, работа с социальными сетями.

- аналитики: 1-2 специалиста — анализ данных по продажам и пользовательскому поведению.

4. Финансовый отдел

Назначение: управление финансовыми потоками компании, планирование бюджета и оценка стоимости разработки и поддержки сервиса.

Состав:

- финансовый директор: Берёзка Вадим Сергеевич — руководство финансовыми процессами, составление отчетности.

- бухгалтер: Берёзка Вадим Сергеевич — ведение бухгалтерского учета, расчет заработной платы и налогов.

- аналитик: Берёзка Вадим Сергеевич — анализ финансовых данных, составление прогнозов.

5. Отдел ИТ и безопасности

Назначение: обеспечение технической поддержки и безопасности интернет-сервиса.

Состав:

- IT-менеджер: Берёзка Вадим Сергеевич — управление IT-инфраструктурой, координация работы отдела.

- системные администраторы: 2 человека — настройка и поддержка серверов, управление сетевой инфраструктурой.

- специалисты по безопасности: 1-2 человека — мониторинг и защиту информационных систем от угроз.

Оценка стоимости разработки и поддержки сервиса

Для оценки стоимости разработки и поддержки интернет-сервиса Интернет-магазин видеоигр необходимо учитывать следующие аспекты:

- заработная плата сотрудников: оценка затрат на оплату труда всех членов команды, включая разработчиков, тестировщиков, специалистов по поддержке и менеджеров.

- технические ресурсы: затраты на серверы, хостинг, лицензии на программное обеспечение и инструменты разработки.

- маркетинговые расходы: бюджет на рекламные кампании, продвижение в социальных сетях и другие маркетинговые активности.

- операционные расходы: коммунальные услуги, аренда офисных помещений и другие текущие расходы.

# Логическая и физическая схема базы данных

Логическая схема базы данных представлена на рисунке 1.

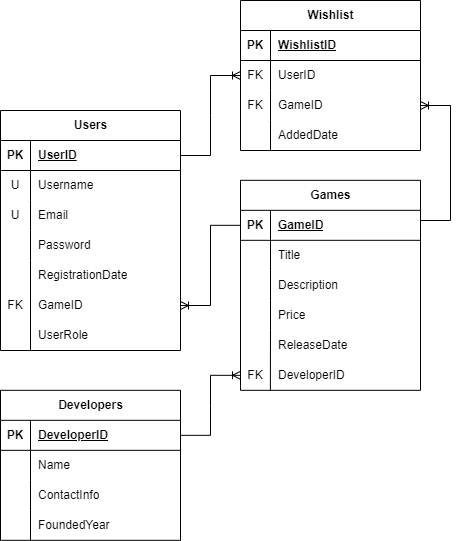


Рисунок 1 - Диаграмма логической схемы базы данных

* 1. Таблица, содержащая сведения обо всех таблицах базы данных

|  |  |
| --- | --- |
| **Имя таблицы** | **Назначение таблицы** |
| Users | Хранит информацию о пользователях интернет-сервиса |
| Games | Содержит данные об играх, представленных в интернет-магазине |
| Developers | Хранит информацию о разработчиках игр |
| Wishlist | Сохраняет данные о списках желаемого для пользователей |

* 1. Сведения обо всех полях таблицы базы данных

Таблица Users

|  |  |
| --- | --- |
| **Поле таблицы** | **Назначение поля** |
| UserID (PK) | Уникальный идентификатор пользователя |
| Username (U) | Уникальное имя пользователя |
| Email (U) | Уникальный адрес электронной почты пользователя |
| Password | Пароль пользователя, хранящийся в зашифрованном виде |
| RegistrationDate | Дата регистрации пользователя в системе |
| GameID (FK) | Уникальный идентификатор игры |
| UserRole | Роль пользователя (Guest, User, Developer, Publisher) |

Таблица Games

|  |  |
| --- | --- |
| **Поле таблицы** | **Назначение поля** |
| GameID (PK) | Уникальный идентификатор игры |
| Title | Название игры |
| Description | Описание игры |
| Price | Цена игры |
| ReleaseDate | Дата выпуска игры |
| DeveloperID (FK) | Внешний ключ, ссылающийся на разработчика игры |

Таблица Developers

|  |  |
| --- | --- |
| **Поле таблицы** | **Назначение поля** |
| DeveloperID (PK) | Уникальный идентификатор разработчика |
| Name | Название компании-разработчика |
| ContactInfo | Контактная информация разработчика |
| FoundedYears | Год основания разработчика |

Таблица Wishlist

|  |  |
| --- | --- |
| **Поле таблицы** | **Назначение поля** |
| WishlistID (PK) | Уникальный идентификатор элемента списка желаемого |
| UserID (FK) | Внешний ключ, ссылающийся на пользователя |
| GameID | Уникальный идентификатор игры |
| AddedDate | Дата добавления игры в список желаемого |

1.3 Таблица, содержащая сведения обо всех связях между таблицами базы данных

|  |  |  |
| --- | --- | --- |
| **Таблица PK** | **Таблица FK** | **Описание связи** |
| Users (UserID) | Wishlist (UserID) | Один пользователь может иметь несколько игр в списке желаемого (1:N) |
| Developers (DeveloperID) | Games (DeveloperID) | Один разработчик может выпустить несколько игр (1:N) |
| Games (GameID) | Users (GameID) | Одна игра может быть у нескольких пользователей (1:N) |

Физическая схема базы данных определяет, как данные будут храниться в выбранной системе управления базами данных (СУБД). В этом случае мы будем использовать **Firebase**, облачную платформу для разработки приложений, которая предлагает простую в использовании базу данных в реальном времени.

Для реализации данной базы данных будет использоваться **Firebase Realtime Database**. Эта СУБД поддерживает хранение данных в формате JSON и предоставляет возможности синхронизации в реальном времени.

Физическая схема базы данных для Firebase описывает, как данные будут структурированы и организованы в формате JSON. Эта структура обеспечивает простоту использования, а также возможность синхронизации данных в реальном времени. Физическая схема базы данных представлена на рисунке 2.

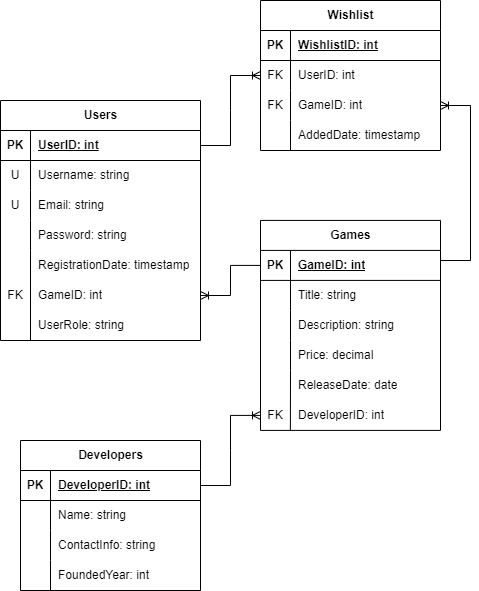


Рисунок 2 - Физическая схема базы данных

# Архитектура сервиса

Архитектура веб-приложений играет ключевую роль в обеспечении стабильной работы, масштабируемости и безопасности. Приложение, разработанное с использованием React и Firebase, представляет собой современное SPA (Single Page Application), где React отвечает за динамический пользовательский интерфейс, а Firebase предоставляет инфраструктуру для аутентификации, хранения данных и хостинга. Важно понимать, как различные компоненты и технологии взаимодействуют между собой для обеспечения надлежащего функционирования системы.

Веб-приложение можно разделить на две основные части: клиентскую и серверную. Клиентская часть полностью реализована с использованием React. React является одной из самых популярных библиотек для создания пользовательских интерфейсов. Она предоставляет возможность создания многоразовых компонентов, что упрощает поддержку и развитие системы. Серверная часть реализована на основе Firebase, облачной платформы от Google, которая предоставляет готовые решения для аутентификации, работы с базами данных, а также хостинга.

Приложение использует несколько ключевых технологий. Во-первых, React — это библиотека для создания пользовательских интерфейсов, основанная на компонентном подходе. Каждый элемент интерфейса представлен отдельным компонентом, что позволяет создавать гибкие и легко модифицируемые приложения. React использует Virtual DOM, что значительно повышает производительность. Вместо того чтобы обновлять всю страницу при каждом изменении, React обновляет только те части, которые были изменены. Это особенно важно для приложений, которые интенсивно работают с данными и требуют быстрой реакции на пользовательские действия.

Для управления состоянием приложения используется React-Redux. Redux — это библиотека, которая помогает управлять глобальным состоянием приложения. Она работает по принципу «единого источника правды», что означает, что всё состояние приложения хранится в одном месте — Redux Store. Это упрощает управление состоянием, поскольку все изменения четко отслеживаются и записываются в одном источнике. Redux работает с так называемыми действиями (Actions) и редьюсерами (Reducers). Действия описывают события, которые могут произойти в приложении, а редьюсеры обрабатывают эти события и обновляют состояние. Такой подход обеспечивает предсказуемость и надежность работы системы, так как все изменения состояния контролируются и могут быть легко воспроизведены.

Еще одной важной технологией является Redux-Thunk. Redux-Thunk — это middleware для Redux, который позволяет выполнять асинхронные действия. В контексте веб-приложений это часто используется для выполнения сетевых запросов. Например, при отправке данных на сервер или получении данных из базы данных используется Redux-Thunk для выполнения этих действий асинхронно. Без него Redux может обрабатывать только синхронные действия, но с помощью Redux-Thunk можно расширить его возможности и работать с асинхронными операциями. Например, при необходимости получить данные из Firebase Firestore (облачная база данных) или аутентифицировать пользователя через Firebase Authentication.

MaterialUI используется для создания пользовательского интерфейса. Это библиотека компонентов, основанная на концепции Material Design от Google. Она предоставляет набор готовых элементов интерфейса, таких как кнопки, формы, диалоги и прочие элементы, которые можно использовать в приложении. Это значительно ускоряет процесс разработки, так как разработчику не нужно создавать все элементы с нуля. К тому же, все компоненты легко настраиваются и могут быть адаптированы под нужды конкретного приложения.

Firebase выполняет несколько функций в архитектуре приложения. Во-первых, Firebase Authentication используется для аутентификации пользователей. Это может быть аутентификация через Google, Facebook, электронную почту и другие методы. Firebase Authentication предоставляет готовые API для интеграции этих функций в приложение. Это освобождает разработчиков от необходимости создания собственной системы аутентификации и обработки безопасности, что значительно экономит время разработки.

Во-вторых, Firebase Firestore используется для хранения данных. Это облачная база данных NoSQL, которая поддерживает синхронизацию данных в реальном времени. Это означает, что любые изменения в базе данных немедленно отражаются в приложении, что особенно важно для приложений с активными пользователями и большим объемом данных. Firebase Firestore также предоставляет возможность гибкого хранения и масштабирования данных. Приложение может взаимодействовать с базой данных через API или с использованием SDK Firebase.

Firebase Hosting используется для развертывания и хостинга веб-приложения. С его помощью можно легко развернуть статические файлы, такие как HTML, CSS и JavaScript, и предоставить их пользователям. Firebase Hosting автоматически масштабируется и обеспечивает быструю доставку контента благодаря интеграции с глобальной сетью доставки контента (CDN). Это означает, что пользователи по всему миру будут иметь доступ к приложению с минимальной задержкой.

Взаимодействие между клиентской и серверной частями приложения осуществляется через Firebase. Когда пользователь взаимодействует с интерфейсом приложения, данные могут быть отправлены на сервер через Firebase SDK. Например, когда пользователь вводит свои учетные данные для входа, они отправляются на сервер Firebase для проверки. Если данные корректны, Firebase возвращает токен, который используется для дальнейших действий в системе. Этот токен может быть использован для получения доступа к защищенным ресурсам или для взаимодействия с базой данных.

При работе с данными приложение взаимодействует с Firebase Firestore. Когда необходимо получить данные из базы данных, Redux-Thunk отправляет асинхронный запрос к Firebase, и полученные данные затем сохраняются в глобальном состоянии приложения (Redux Store). После этого соответствующие компоненты React автоматически обновляются, что обеспечивает мгновенное обновление интерфейса при изменении данных.

Firebase также поддерживает синхронизацию данных в реальном времени. Это означает, что если несколько пользователей работают с одними и теми же данными, все изменения сразу же отражаются у всех пользователей. Например, если один пользователь обновляет информацию в приложении, другие пользователи немедленно видят эти изменения без необходимости обновлять страницу. Это делает Firebase идеальным решением для приложений, где важна работа в реальном времени, например, для чатов, систем совместной работы или онлайн-игр.

Еще один важный аспект архитектуры — это хостинг приложения. Firebase Hosting предоставляет простой способ развернуть фронтенд приложения. Все статические файлы приложения загружаются на Firebase и автоматически доступны пользователям по всему миру. Firebase Hosting также обеспечивает высокий уровень безопасности, автоматически предоставляет SSL-сертификаты и поддерживает HTTP/2 для более быстрой передачи данных.

Такое сочетание технологий позволяет создать масштабируемое, производительное и безопасное веб-приложение. React предоставляет мощные инструменты для создания динамических интерфейсов, Redux упрощает управление состоянием, а Firebase берет на себя задачи по аутентификации, хранению данных и хостингу. Все эти технологии хорошо интегрируются друг с другом, что позволяет разрабатывать приложения с минимальными затратами на инфраструктуру и серверное обеспечение.

В заключение, архитектура приложения, основанного на React и Firebase, представляет собой мощное решение для разработки современных веб-приложений. Firebase позволяет избавиться от необходимости настройки серверной инфраструктуры, предоставляя готовые решения для аутентификации и работы с данными. React, в свою очередь, обеспечивает гибкий и легко расширяемый интерфейс, который можно адаптировать под любые нужды. Архитектура сервиса представлена на рисунке 1.

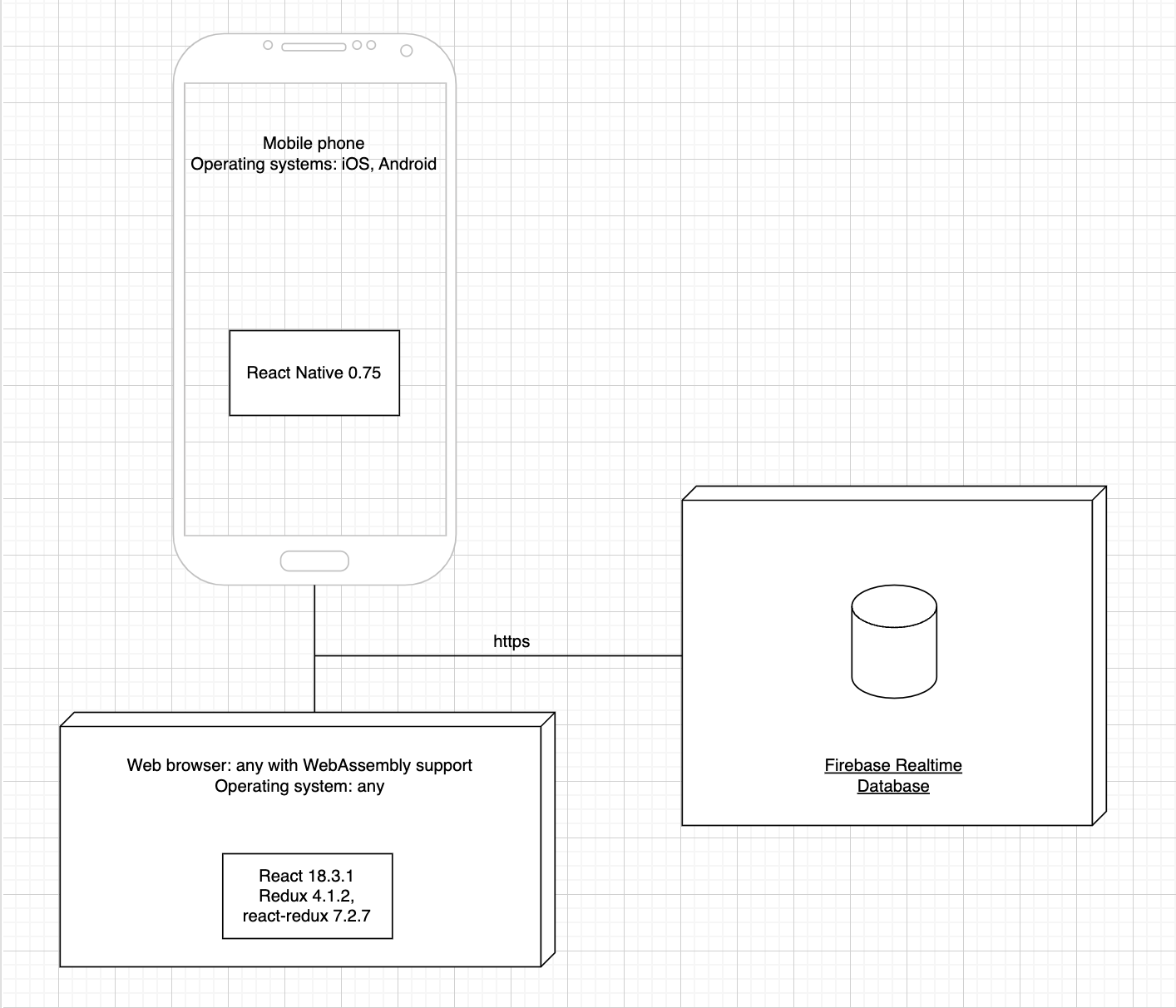


Рисунок 1 – Архитектура сервиса

# Процесс создания сервиса

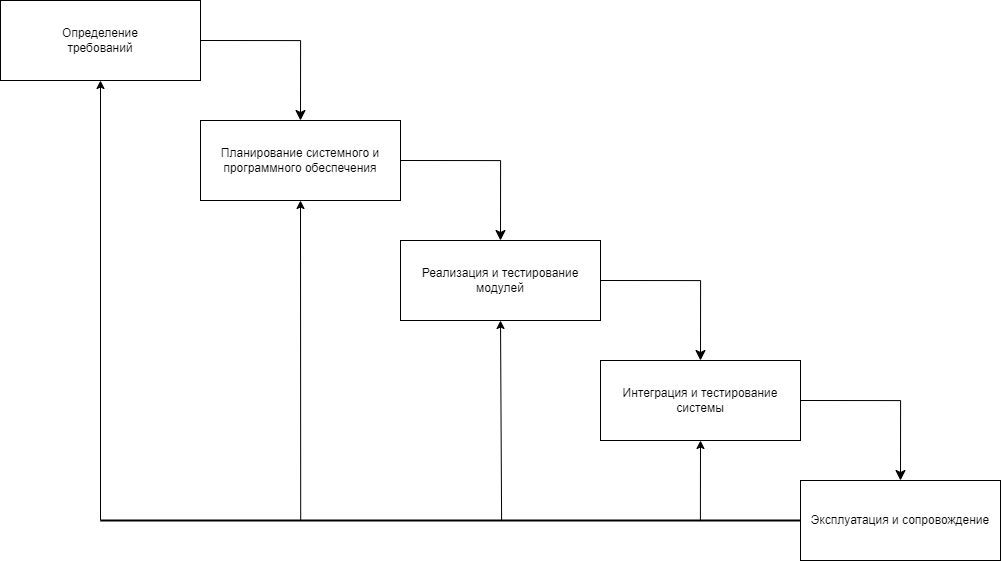


Рисунок 1 - Диаграмма последовательности, поясняющая процесс создания сервиса

1. Определение требований. Этот этап можно разделить на две категории - системный анализ (все то, что окружает конкретное программное обеспечение), и анализ требований. Документируют поведение системы, производительность, интерфейс и т.д.

2. Планирование системного и программного обеспечения. Фокусируется на основных свойствах программы, таких как структуры данных, архитектура программного обеспечения, функции интерфейса, алгоритмические и процедурные детали. Качество проекта возможно оценить. Результат документируется.

3. Реализация и тестирование модулей. В проекте Описанная система в проекте программируется в виде модулей и комплектом программ, которые тестируются отдельно. Чем детальнее проект, тем проще и более механическим может быть этап реализации.

4. Интеграция и тестирование системы. Объединяют программы и модули, тестируют всю систему, после тестирования продукт передается заказчику. Во время тестирования сосредотачиваются как на логических деталях, так и на том, отвечает ли система требованиям в отношении функциональности (проверка достоверности).

5. Эксплуатация и сопровождение - это обычно самая длинная фаза. Систему изменяют, если пользователи находят ошибки, либо окружение и рабочая среда изменяются или клиент нуждается в новой функциональности. Фаза повторяет все предыдущие этапы в рамках изменения существующей системы.

# Источники

1. reactnative.dev [Электронный ресурс] — Режим доступа: <https://reactnative.dev/> — Дата доступа: 18.09.2024
2. react.dev [Электронный ресурс] — Режим доступа: <https://react.dev> — Дата доступа: 19.09.2024
3. firebase.google.com [Электронный ресурс] — Режим доступа: <https://firebase.google.com> — Дата доступа: 20.09.2024