|  |  |  |  |
| --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН  СЕВЕРО-КАЗАХСТАНСКИЙ УНИВЕРСИТЕТ ИМ. М. КОЗЫБАЕВА | | | |
|  | | | |
|  | | |  |
| СЕМЕСТРОВЫЙ ПРОЕКТ  На тему: «Репозиторий ретро-игр» | | | |
| Выполнили студенты  группы АПО-20-2 |  | Симонов Д.А.  Сочилович Д.А.  Тулинов А.Е.  Толстов В.Ю. | |
| Доктор PHD, доцент кафедры  ИКТ |  | Астапенко Н.В. | |
|  |  |  | |

Петропавловск, 2023

ЗАДАНИЕ ПО ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

СЕМЕСТРОВЫЙ ПРОЕКТ по дисциплине «Программная инженерия» Задание: Командная разработка программного средства.

В процессе работы команда:

* + - 1. Осваивает работу с coda.io;
      2. Определяет тему проекта исходя из социальной или бизнес актуальности;
      3. Планирует процесс разработки, распределяет работы;

1. Выбирает модель и методы разработки из методологий Agile;
2. Разрабатывает UserStory и архитектуру ПС;
3. Реализует проект с помощью систем No/Low code / Реализует проект на основе курса Aframe.

|  |  |
| --- | --- |
| № | Ф.И.О студентов |
| 1 | Толстов В.Ю.  Тулинов А. Е.  Сочилович Д. А.  Симонов Д. А. |

Порядок защиты: На защиту семестрового проекта предоставляется

«сшитая» пояснительная записка и ссылка на github, где размещены:

* видео-презентация проекта (видеозапись экрана, где демонстрируется проект);
* электронный вариант пояснительной записки;
* архив проекта или ссылка на проект.

Весь процесс разработки документируется в пояснительной записке, оформленной в соответствии с «Методическими указаниями по курсовому и дипломному проектированию».

Семестровый проект должен состоять из следующих структурных элементов:

* титульный лист;
* задание по выполнению курсовой работы (проекта);
* содержание (1 лист); - введение (2-3 листа);
* основная часть (15-25 листов):
* аналитическая часть (7-10 листов);
* проектная часть (8-15 листов);
* заключение (1-2 листа);
* список использованной литературы (1-2 листа, не менее 15 источников).

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc134166552)

[1 Аналитическая часть 6](#_Toc134166553)

[1.1 Методология разработки Scrum 6](#_Toc134166554)

[1.3 Технология Bravo Studio 8](#_Toc134166555)

[1.4 Распределение обязанностей 10](#_Toc134166556)

[1.5 Анализ аналогов 12](#_Toc134166557)

[2 Проектная часть 15](#_Toc134166558)

[2.1 Работа с Coda.io 15](#_Toc134166559)

[2.3 Разработка базы данных 23](#_Toc134166560)

[2.3 Архитектура приложения 26](#_Toc134166561)

[2.1.1 Разработка функции «каталог игр» 27](#_Toc134166562)

[2.1.2 Разработка функции «поиск» 28](#_Toc134166563)

[2.1.3 Разработка функции «оплата» 29](#_Toc134166564)

[2.1.4 Разработка функции «добавление в избранное» 29](#_Toc134166565)

[2.1.5 Разработка функции «скачивание» 30](#_Toc134166566)

[2.1.6 Разработка функции «авторизация» 31](#_Toc134166567)

[2.2 Функционал приложения 32](#_Toc134166568)

[ЗАКЛЮЧЕНИЕ 39](#_Toc134166569)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 40](#_Toc134166570)

# ВВЕДЕНИЕ

В настоящее время игры настолько востребованы, что их различные варианты появляются на различных платформах. Несмотря на разнообразие современных видеоигр, всегда есть поклонники ретро стиля, люди, которые хотят поностальгировать и непосредственно коллекционеры картриджей. В связи с этим было принято решение разработать мобильное приложение, которое позволит ознакомиться с необходимой информацией об игре, получить образ для эмулятора или же заказать непосредственно картридж.

*Целью* разработки проекта является создание мобильного приложения, которое будет представлять собой репозиторий ретро-игр для их поиска, просмотра и покупки.

Главными *задачами* являются:

* разработка функционала приложения, которое позволит пользователям удобно и быстро находить и скачивать интересующие их игры;
* разработка дизайна приложения;
* создание базы данных игр и консолей;
* интеграция платежной системы для процесса покупки игр;

В целом, проект должен предоставить пользователям удобный и надежный инструмент для покупки и наслаждения ретро-играми.

*Актуальность* приложения заключается в том, что игры старого поколения предоставляют историческую ценность для игровой индустрии, и постепенно забываются так как, нет сервисов предоставляющих упорядоченную информацию об играх старого поколения, а картриджи для приставок в большинстве своем можно найти либо у других людей, либо на аукционах.

*Объектом исследования* является мобильное приложение, которое предоставляет репозиторий ретро-игр.

*Предметом исследования* является процесс разработки мобильного приложения, включая использование различных инструментов и технологий, таких как Bravo Studio, Airtable, Firebase и Figma.

Для создания мобильного приложения были использованы различные инструменты и технологии, такие как Bravo Studio, Airtable, Firebase и Figma. Bravo Studio использовался для разработки макетов и создания интерфейса приложения. Airtable использовался для управления базой данных, включая информацию о названии игры, дате выпуска, жанре и других характеристиках. Firebase использовался для регистрации и аутентификации пользователей. Figma позволяет создать примерные прототипы дизайна и размещения элементов.

Одним из основных преимуществ использования Bravo Studio является возможность создания дизайна и интерфейса приложения без необходимости написания кода. Это позволяет сократить время разработки и сосредоточиться на других аспектах проекта. Кроме того, Bravo Studio интегрируется с другими инструментами, такими как Figma, что облегчает процесс разработки.

Airtable был использован для создания базы данных ретро-игр. Он предоставляет интуитивно понятный интерфейс для создания и управления данными, что облегчает работу с информацией о различных играх.

Firebase использовался для регистрации и аутентификации пользователей, что обеспечивает безопасность приложения и защиту от несанкционированного доступа.

Figma использовался для разработки дизайна и прототипирования приложения. Этот инструмент позволяет создавать профессиональные дизайны интерфейса, которые можно легко прототипировать и исправлять.

В целом, использование этих инструментов и технологий позволило создать мобильное приложение, которое предоставляет репозиторий ретро-игр с удобным интерфейсом, безопасностью и хорошо организованной базой данных.

# 1 Аналитическая часть

# 1.1 Методология разработки Scrum

Scrum – это методология гибкой разработки программного обеспечения, которая призвана увеличить производительность команды и качество продукта, используя итеративный и инкрементальный подход к разработке. Основными принципами Scrum являются:

* Работа в итерациях (спринтах): команды работают в коротких, фиксированных по времени итерациях, обычно продолжительностью от 1 до 4 недель, в течение которых они создают инкремент продукта, который готов к использованию.
* Командная работа: команда разработки в Scrum состоит из небольшой группы людей, которые работают вместе для достижения общей цели. Команды самоорганизующиеся, они принимают решения вместе и делятся ответственностью за результат.
* Backlog продукта: это список требований к продукту, который составляет владелец продукта. Он описывает то, что нужно сделать, чтобы продукт был готов к выходу на рынок. Backlog продукта постоянно обновляется, чтобы отразить изменения в требованиях к продукту.
* Backlog спринта: это список задач, которые команда планирует выполнить в течение спринта. Задачи выбираются из Backlog продукта, и они должны быть выполнены командой в течение спринта.
* Daily Scrum: это ежедневное собрание команды, которое проходит на 15 минут и на котором каждый член команды отвечает на три вопроса: что он сделал вчера, что он собирается сделать сегодня и какие препятствия у него возникли.
* Sprint Review: это обзор продукта, который проводится в конце каждого спринта. Команда представляет результат своей работы за спринт, владелец продукта дает обратную связь, и команда обсуждает, что они могут улучшить в следующем спринте.
* Sprint Retrospective: это собрание, которое проводится в конце каждого спринта и на котором команда обсуждает, что они сделали хорошо, что можно улучшить и как они могут улучшить свой процесс разработки в целом.

В качестве методологии разработки мобильного приложения был выбран Scrum, потому что она позволяет нам более эффективно работать в команде и достигать лучших результатов. С помощью Scrum возможно более гибко управлять процессом разработки. Кроме того, методология Scrum способствует более прозрачной и открытой коммуникации внутри команды, что помогает достигать лучших результатов и повышать качество продукта. Подробный план как была использована методология Scrum для проекта:

1. Определены роли в команде:

* Владелец продукта (Толстов Вадим): определены требования, такие как список ретро-игр, функциональность поиска, пользовательский интерфейс, интеграция с платежными системами и другие особенности.
* Scrum-мастер: обеспечено соблюдение Scrum-практик и помощь команде оставаться сфокусированной на целях спринта.
* Разработчики: сформирована команда специалистов по разработке мобильных приложений, дизайну, тестированию и интеграции.

1. Создан бэклог продукта:

* Были формулированы требования и функции приложения, такие как:

1. анализ аналогов;
2. разработка концепции приложения;
3. разработка базы данных;
4. разработка дизайна;
5. разработка каталога;
6. подключение авторизации;
7. разработка поиска;
8. подключение функции скачивания образа;
9. разработка системы избранного;
10. подключение оплаты.

* Эти требования были разделены на мелкие задачи и приоритизированы.

1. Проведено планирование спринта:

* Определена продолжительность спринта (например, 2 недели).
* Выбраны задачи из бэклога продукта для текущего спринта:

1. анализ аналогов;
2. разработка концепции приложения;
3. разработка базы данных.

* Было проверено соответствие объема работ возможностям команды.
* Были организованы ежедневные стендапы:

1. Сбор команды каждый день для обсуждения прогресса, возникающих проблем и планов на день.
2. Обсуждение задач, таких как создание базы данных, дизайн пользовательского интерфейса или интеграция с платежными системами.
3. Проведен обзор спринта:

* В конце спринта демонстрировались результаты работы друг другу, например, пользовательский интерфейс, список доступных игр или функциональность поиска.
* Получена обратная связь друг от друга и других заинтересованных сторон, чтобы узнать, какие аспекты приложения работают хорошо и что может потребовать доработки или изменений.
* Внесена полученная обратная связь в бэклог продукта и приоритизированы новые/измененные требования.

1. Организована ретроспектива спринта:

* После обзора спринта команда собиралась для анализа прошедшего спринта.
* Обсуждалось, что сработало хорошо, что не сработало и что можно улучшить в будущих спринтах. Например, рассматривались проблемы с коммуникацией, процессами разработки или тестирования.
* Определялись действия для улучшения процессов и повышения эффективности команды на следующих спринтах.

# 1.3 Технология Bravo Studio

Bravo Studio – это платформа для создания мобильных приложений без необходимости знания кода. Она позволяет людям создавать и опубликовывать приложения на iOS и Android устройствах, используя простой и интуитивно понятный интерфейс.

Платформа имеет редактор, который позволяет пользователям создавать и настраивать макеты приложений, добавлять различные элементы дизайна, такие как кнопки, текст, изображения, видео и многое другое. Пользователи могут также настраивать функциональность приложения, добавлять различные модули, такие как формы, списки, карты и другие, а также создавать и настраивать базы данных для хранения информации.

Bravo Studio поддерживает интеграцию с различными сервисами, такими как Google Maps, Instagram, Twitter, Facebook и другие. Это позволяет пользователям добавлять функциональность, связанную с этими сервисами, в свои приложения.

После того, как приложение создано, пользователи могут опубликовать его в App Store и Google Play Store, где оно становится доступным для скачивания и использования.

Bravo Studio предоставляет различные тарифные планы, в зависимости от потребностей пользователей. Они варьируются от бесплатного плана, который позволяет создавать одно приложение, до более дорогих планов, которые предоставляют более широкие возможности и дополнительную поддержку.

Кроме того, Bravo Studio предоставляет ресурсы и учебные материалы для помощи пользователям в создании своих приложений. Это включает в себя документацию, видеоуроки и сообщество пользователей, где можно задавать вопросы и общаться с другими разработчиками.

Bravo Studio также позволяет пользователям работать в режиме реального времени, что означает, что изменения в приложении могут быть видны непосредственно в редакторе. Это позволяет быстро тестировать различные версии приложений и делать корректировки до публикации в магазинах приложений.

При разработке проекта Bravo Studio используется в первую очередь для создания макета приложения. Есть возможность выбрать из множества готовых элементов дизайна, таких как кнопки, поля для ввода текста, изображения и другие, и настроить их в соответствии с нашими потребностями.

Bravo Studio был выбран для разработки проекта по нескольким причинам.

Во-первых, Bravo Studio предоставляет интуитивно понятный интерфейс, который позволяет людям без технических знаний и опыта в программировании создавать мобильные приложения.

Во-вторых, Bravo Studio позволяет работать в режиме реального времени, что позволяет быстро тестировать и корректировать приложение.

В-третьих, Bravo Studio предоставляет пользовательскую аналитику и данные, которые помогают отслеживать успех приложения и определять области для улучшения.

В целом, Bravo Studio предоставляет мощный инструмент для создания мобильных приложений, который позволяет создавать высококачественные приложения без необходимости вручную писать код.

Для создания макетов, а также оформления дизайна использовалась среда Figma. Это инструмент для создания макетов и прототипов веб-приложений и мобильных приложений, который позволяет дизайнерам и разработчикам работать в режиме реального времени над одним и тем же проектом. Figma имеет широкий набор функций, включая возможность создания векторных изображений, добавления элементов интерфейса и создания интерактивных прототипов.

Figma используется для создания макетов и прототипов приложения вместе с Bravo Studio. С помощью Figma можно создавать дизайн интерфейса приложения, добавлять элементы и определять стиль, цвет и шрифты. Также Figma предоставляет возможность создания прототипов приложения, чтобы проверить работоспособность и дизайн приложения перед его разработкой в Bravo Studio.

С помощью Figma можно легко создавать макеты и прототипы приложения в режиме реального времени, что позволяет всей команде сотрудничать и работать вместе над проектом. В Figma имеется возможность создания отдельных страниц и проектов, чтобы разделять дизайн на разные компоненты, такие как логин, профиль пользователя и главная страница приложения.

После создания макетов и прототипов приложения в Figma, их можно экспортировать в форматы, совместимые с Bravo Studio. Это позволяет импортировать дизайн в Bravo Studio и использовать его в разработке приложения.

В качестве базы данных в разработке проекта используется Airtable API. Airtable – это облачная база данных, которая позволяет создавать, хранить и управлять данными любой сложности. Она предоставляет пользовательский интерфейс для создания таблиц, которые можно использовать для хранения данных, таких как контакты, задачи, проекты, продукты и многое другое. Кроме того, Airtable позволяет настраивать поля таблицы и взаимодействовать с другими инструментами, такими как Bravo Studio. Данные из Airtable можно интегрировать в мобильные приложения, созданные в Bravo Studio.

При совместном использовании Airtable и Bravo Studio можно управлять данными и быстро обновлять их в приложении, при этом не нужно вручную писать код для интеграции данных. Это значительно упрощает и ускоряет процесс разработки, так как можно сосредоточиться на разработке функциональности приложения, а не на написании кода для работы с данными.

Кроме того, Airtable предоставляет различные инструменты для управления данными, такие как фильтры, сортировки и формулы, которые можно использовать для настройки отображения и манипулирования данными в приложении.

В целом, Airtable и Bravo Studio предоставляют мощный инструментарий для создания мобильных приложений, которые быстро и эффективно работают с данными. Можно использовать Airtable для управления и хранения данных, а затем интегрировать эти данные в приложение, созданное в Bravo Studio, чтобы получить такое приложение, которое быстро обновляется с новыми данными и удовлетворяет требованиям пользователей.

Для добавления в приложения функциональности входа и аутентификации пользователя была использована Firebase. Firebase – это платформа для разработки мобильных и веб-приложений, которая предоставляет широкий набор инструментов для создания приложений, включая инструменты для хранения, аналитики, облачных функций и многое другое. Firebase также предоставляет набор инструментов для регистрации и аутентификации пользователей, которые можно использовать для создания безопасных и защищенных приложений.

С помощью API, предоставляемым Firebase, можно добавить функционал регистрации и аутентификации в мобильные приложения, созданные в Bravo Studio. Есть возможность настроить Firebase для подключения к приложению в Bravo Studio, используя соответствующий ключ API. В целом, Firebase является незаменимым инструментом для разработки проекта в Bravo Studio.

# 1.4 Распределение обязанностей

Распределение ответственности в команде разработки проекта играет важную роль в достижении успеха. Оно позволяет определить роли и обязанности каждого члена команды, установить ясные цели и ожидания, а также обеспечить более эффективную и продуктивную работу.

Распределение ответственности позволяет четко определить кто отвечает за конкретные аспекты проекта, что позволяет лучше планировать и контролировать процесс работы. Это также помогает избежать дублирования работ, что может снижать эффективность команды и затягивать сроки.

Кроме того, распределение ответственности способствует улучшению коммуникации в команде. Каждый член команды понимает, кто за что отвечает, и может обратиться к соответствующему человеку при возникновении вопросов. Это также позволяет установить более четкие и реалистичные сроки выполнения задач, что способствует более точному планированию и сокращению задержек. Распределение ответственностей помогает оптимизировать процесс работы, повышает качество и эффективность проекта, а также улучшает коммуникацию и сотрудничество в команде.

Команда разработки приложения состоит из четырех разработчиков. Согласно методологии Scrum, она предполагает следующие роли и их ответственности:

Scrum Master отвечает за:

* организацию ежедневных стендапов;
* управление бэклогом продукта и планирование спринтов;
* координацию работы команды и поддержку коммуникации между разработчиками;
* решение конфликтов и проблем внутри команды.

Роль Scrum Master закреплена за Толстов В.Ю.

Product Owner отвечает за:

* определение целей и задач приложения;
* работу с пользователями для определения требований и пожеланий;
* составление бэклога продукта и планирование релизов;
* расставление приоритетов задач разработки.

Роль Product Owner закреплена за Сочилович Д. А.

Frontend Developer отвечает за:

* разработку пользовательского интерфейса;
* работу с дизайнером для определения функционала и дизайна приложения;
* настройку блоков и тестирование функционала.

Роль Frontend Developer закреплена за Тулинов А. Е.

Backend Developer отвечает за:

* разработку серверной части приложения;
* работу с базами данных и API;
* настройку блоков и тестирование функционала;
* обеспечение безопасности и надежности работы сервера.

Роль Backend Developer выполняют Симонов Д. А. и Толстов В.Ю.

Матрица ответственности согласно методологии Scrum изображена на таблице 1.1.

Таблица 1.1

Матрица ответственности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Обязанности/Разработчики | Scrum Master | Product Owner | Frontend Developer | Backend Developer |
| Организация стендапов | + |  |  |  |
| Управление бэклогом продукта | + |  |  |  |
| Координация работы команды | + |  |  |  |
| Решение конфликтов | + |  |  |  |
| Определение целей и задач |  | + |  |  |
| Разработка интерфейса |  |  | + |  |
| Настройка блоков |  |  | + | + |
| Тестирование функционала |  |  | + | + |
| Разработка серверной части |  |  |  | + |
| Работа с базами данных |  |  |  | + |
| Обеспечение безопасности |  |  |  | + |

Как видно из матрицы ответственности, каждый член команды имеет свои задачи и обязанности, необходимые для успешного завершения проекта. Все роли в команде имеют равную значимость и взаимодействуют друг с другом для создания приложения высокого качества.

# 1.5 Анализ аналогов

Конкурентный анализ является важным этапом в разработке приложения, поскольку он помогает определить конкурентное преимущество вашего продукта. Примеры подобных приложений:

1. https://www.old-games.ru/

* Сайт имеет большую коллекцию игр для различных платформ, включая консоли и компьютеры.
* Предоставляет пользовательский интерфейс, который удобен в использовании, с возможностью быстрого поиска и фильтрации игр.
* Сайт имеет сообщество, где пользователи могут делиться опытом, обсуждать игры и помогать друг другу с проблемами.
* Есть возможность скачать игры бесплатно, без регистрации.
* Сайт не имеет рекламы, что создает приятный пользовательский опыт.

2. http://free-iso.org/

* Сайт специализируется на играх для консолей PlayStation и Xbox.
* Сайт имеет большую библиотеку игр, но качество некоторых файлов может быть низким.
* Скачивание игр доступно бесплатно, без регистрации.
* Не имеет пользовательского интерфейса и возможности быстрого поиска и фильтрации игр.
* Есть много рекламы, что может быть раздражающим для пользователей.

3. http://emuplanet.ru/

* Сайт имеет большую библиотеку игр для различных платформ, включая консоли и компьютеры.
* Скачивание игр доступно бесплатно, без регистрации.
* Сайт имеет простой пользовательский интерфейс, но не предоставляет возможности быстрого поиска и фильтрации игр.
* Есть много рекламы, что может быть раздражающим для пользователей.
* Некоторые файлы игр могут быть повреждены или содержать вирусы.

4. https://www.emu-land.net/

* Сайт имеет большую коллекцию игр для различных платформ, включая консоли и компьютеры.
* Сайт имеет пользовательский интерфейс, который удобен в использовании, с возможностью быстрого поиска и фильтрации игр.
* Скачивание игр доступно бесплатно, но требует регистрации.
* Сайт имеет форум, где пользователи могут общаться и обмениваться опытом.
* Есть много рекламы, что может быть раздражающим для пользователей.

5. http://oldconsoles.ru/

* Сайт специализируется на играх для старых консолей, таких как NES, Sega, Atari и других.
* Сайт имеет пользовательский интерфейс, который удобен в использовании, с возможностью быстрого поиска и фильтрации игр.
* Скачивание игр доступно бесплатно, без регистрации.
* Есть сообщество, где пользователи могут общаться и делиться опытом.
* Сайт имеет немного рекламы, что создает приятный пользовательский опыт.

Из анализа можно сделать следующие выводы:

Сайты old-games.ru и emu-land.net имеют наиболее разнообразную библиотеку игр для различных платформ, а также предоставляют пользовательский интерфейс, который удобен в использовании, с возможностью быстрого поиска и фильтрации игр.

Сайт free-iso.org специализируется на играх для консолей PlayStation и Xbox, но не имеет пользовательского интерфейса и возможности быстрого поиска и фильтрации игр. Кроме того, качество некоторых файлов может быть низким, а многочисленная реклама может раздражать пользователей.

Сайт emuplanet.ru также имеет большую библиотеку игр для различных платформ, но не предоставляет возможности быстрого поиска и фильтрации игр. Кроме того, на сайте много рекламы, и некоторые файлы игр могут быть повреждены или содержать вирусы.

Сайт oldconsoles.ru специализируется на играх для старых консолей и имеет пользовательский интерфейс, который удобен в использовании, с возможностью быстрого поиска и фильтрации игр. Кроме того, сайт имеет сообщество, где пользователи могут общаться и делиться опытом, и немного рекламы, что создает приятный пользовательский опыт.

Исходя из вышесказанного, для того чтобы приложение было конкурентоспособным, может потребоваться увеличить количество игр в библиотеке и обеспечить высокое качество игр. Также следует добавить функциональность сохранения игры и быстрого продолжения с места, где пользователь остановился. Приложение имеет преимущество в том, что оно является бесплатным и может привлечь большее количество пользователей.

# 2 Проектная часть

# 2.1 Работа с Coda.io

Coda.io – это онлайн-сервис, который объединяет функциональность электронных таблиц, баз данных, документов и других инструментов в единое рабочее пространство. Он позволяет создавать документы, проекты и таблицы с уникальными функциями, интегрировать их с другими приложениями и сервисами, а также работать над ними совместно с командой в режиме реального времени.

Основные преимущества Coda.io:

* Гибкость и универсальность: Coda.io позволяет создавать документы любой сложности, включая таблицы, базы данных, справочники и другие типы документов. Пользователь может создавать любые нужные функции, используя формулы, а также расширять функциональность Coda.io с помощью сторонних интеграций.
* Коллаборация в режиме реального времени: Coda.io позволяет работать с документами в режиме реального времени с другими участниками команды. Пользователь может видеть изменения, вносимые другими участниками, и обсуждать их встроенными инструментами комментирования и обсуждения.
* Удобство использования: Coda.io предоставляет простой и интуитивно понятный пользовательский интерфейс, что делает его доступным для использования для любого уровня навыков пользователей. Он также имеет множество готовых шаблонов, которые могут быть использованы для быстрого создания документов и таблиц.
* Интеграции: Coda.io может интегрироваться с множеством сторонних приложений и сервисов, включая Google Sheets, Slack, Zapier, GitHub и многие другие. Это позволяет автоматизировать множество задач и создавать более эффективные рабочие процессы.
* Сохранение истории изменений: Coda.io автоматически сохраняет историю изменений, что позволяет пользователям отслеживать, кто и когда внес изменения в документ или таблицу.
* Безопасность: Coda.io предоставляет высокий уровень безопасности для хранения и обработки данных. Он использует шифрование данных в покое и в передаче, а также предоставляет возможность управлять уровнем доступа к документам и таблицам.

В целом, Coda.io – это мощный инструмент для создания и управления документами и таблицами, который позволяет пользователям создавать настраиваемые рабочие процессы и работать с командой в режиме реального времени.

В разработке проекта Coda.io используется для управления задачами, сроками и коммуникацией внутри команды. Какие функции Coda.io использовались во время проекта:

* Создание таблицы задач: в Coda.io можно создать таблицу, которая содержит все задачи, необходимые для завершения проекта, с указанием приоритетов и дат завершения.
* Назначение задач: можно назначать задачи для каждого члена команды, используя соответствующую колонку в таблице задач. Каждый член команды может видеть, какие задачи ему были назначены, и установить себе напоминания о сроках выполнения.
* Слежение за прогрессом: можно использовать столбец «Статус» в таблице задач, чтобы отслеживать прогресс выполнения задачи. Это позволяет всей команде видеть, насколько близко проект находится к завершению, а также выявлять возможные задержки.
* Создание таблицы коммуникации: кроме того, можно создать таблицу для коммуникации с возможностью обмениваться идеями, задавать вопросы и обсуждать проблемы, связанные с проектом.
* Использование шаблонов: Coda.io предоставляет готовые шаблоны для создания документов, которые могут использоваться в проекте.

Coda.io был выбран, потому что это мощный инструмент, который предоставляет широкие возможности для управления проектами, коммуникации внутри команды и интеграции с другими приложениями. Все необходимые функции доступны в одном месте, что упрощает и ускоряет работу над проектом.

Кроме того, Coda.io предоставляет готовые шаблоны и инструменты для создания собственных шаблонов, что позволяет быстро создавать документы, таблицы и другие элементы проекта, не тратя много времени на настройку.

В «coda.io» был создан документ «RetroPlay» созданный для отображения основных целей и задач проекта. Документ «RetroPlay» представлен в соответствии с рисунком 2.1

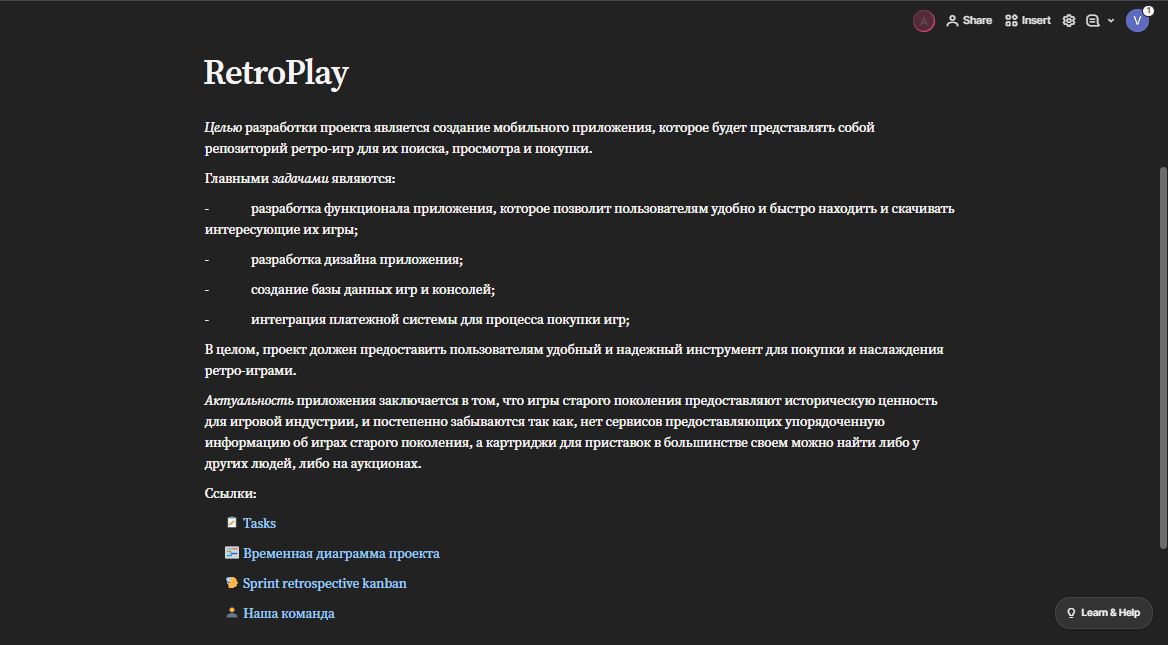


Рисунок 2.1 Документ «RetroPlay»

В «coda.io» был создан документ «Команда», который представляет собой инструмент для организации информации о членах команды в рамках проекта или организации. В этом документе можно создавать и хранить профили участников команды.

В документе «Команда» можно создавать профили участников команды с указанием их имени, контактных данных, ролей и ответственностей. Это позволяет централизованно хранить информацию о членах команды и обеспечивать доступ к актуальным данным.

Кроме того, в документе предусмотрены возможности для описания навыков, компетенций и опыта каждого участника команды. Это помогает оценить квалификацию и способности каждого участника и обеспечивает более эффективное распределение задач и ролей внутри команды.

Документ «Команда» представлен в соответствии с рисунком 2.2

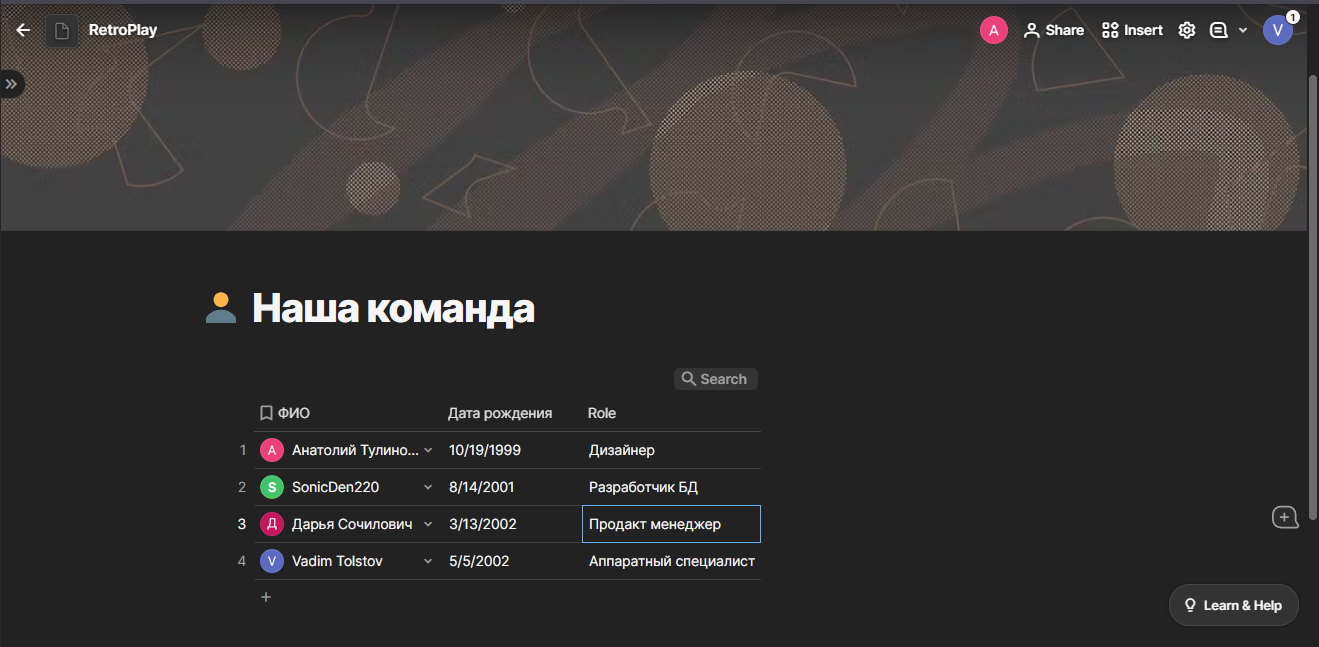


Рисунок 2.2 Документ «Команда»

Спринт-ретроспектива (sprint retrospective) - это формат совещания, который проводится в конце каждого спринта в Agile-разработке для анализа произведенной работы, идентификации проблем и поиска путей улучшения процесса разработки в будущем.

В Coda.io спринт-ретроспектива может быть организована с использованием таблицы или шаблона, который уже содержит стандартные разделы и инструменты, такие как:

* Список пройденных задач. Здесь каждый участник команды может внести информацию о выполненных им задачах в рамках спринта. Это поможет увидеть, какие задачи были выполнены успешно, а какие вызвали проблемы.
* Оценка работы команды. В этом разделе можно провести общую оценку работы команды за прошедший спринт. Каждый участник может выразить свое мнение о том, что было сделано хорошо, а что можно улучшить.
* Идентификация проблем. В этом разделе каждый участник команды может описать проблемы, с которыми столкнулся в процессе работы над задачами. Это могут быть технические проблемы, связанные с кодом или инфраструктурой, или организационные проблемы, связанные с процессами работы команды.
* Поиск путей улучшения. В этом разделе команда может обсудить возможные способы улучшения процесса разработки в будущем на основе выявленных проблем.

Кроме того, в Coda.io можно добавить различные виджеты, такие как диаграммы или графики, чтобы визуализировать данные и упростить анализ информации. Также можно установить напоминания о проведении спринт-ретроспективы и настроить доступ к документу для участников команды. Спринт ретроперспектива представлена в соответствии с рисунком 2.3

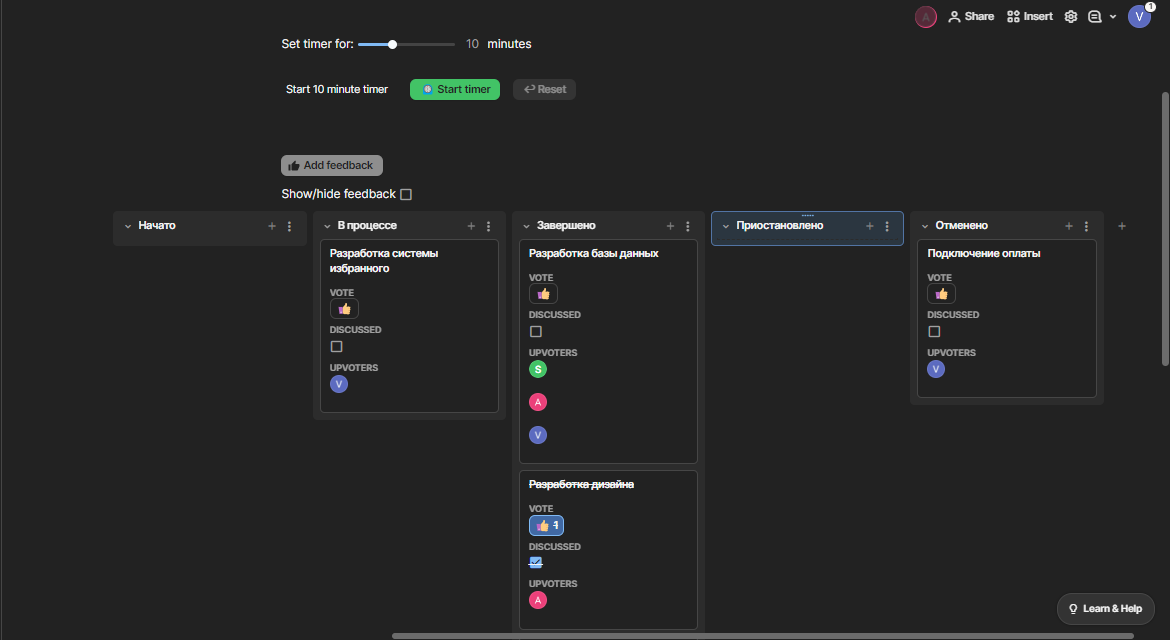


Рисунок 2.3 Спринт ретроперспектива

Временная диаграмма проекта (или Gantt-диаграмма) в Coda.io - это таблица, которая позволяет визуально отобразить хронологию задач и событий проекта в виде горизонтальных полос и меток на временной шкале.

Для создания временной диаграммы проекта в Coda.io нужно создать новую таблицу и добавить необходимые колонки, такие как "Название задачи", "Начало", "Конец", "Продолжительность", "Ответственный" и т.д. Затем необходимо заполнить эти колонки данными о задачах и событиях проекта.

Когда таблица заполнена данными, можно построить Gantt-диаграмму, выбрав соответствующую опцию в меню "Вид". После этого появится графическое представление задач и событий проекта на временной шкале.

На временной диаграмме проекта в Coda.io можно производить различные операции, такие как изменение продолжительности задач, перемещение задач во времени, изменение ответственных за задачи и т.д. Также можно добавлять новые задачи и события, удалять или редактировать существующие.

В целом, временная диаграмма проекта в Coda.io является мощным инструментом для визуализации и управления ходом проекта, который позволяет легко отслеживать прогресс и соблюдать сроки выполнения задач. Временная диаграмма проекта «RetroPlay» представлена в соответствии с рисунком 2.4

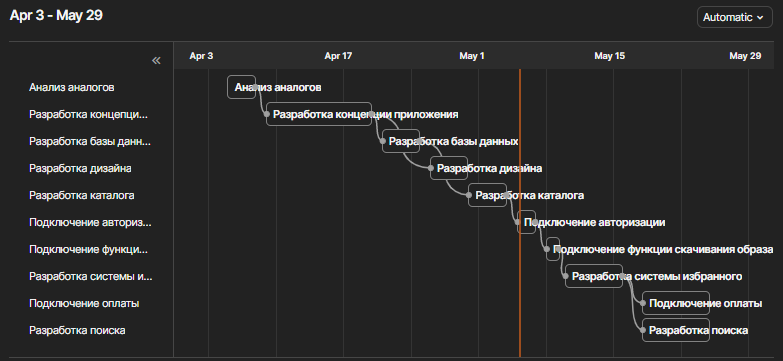


Рисунок 2.4 Временная диаграмма

Документ задачи в Coda.io - это инструмент для отслеживания и управления задачами в проекте. Он обычно содержит список задач и информацию, связанную с каждой задачей, такую как ее статус, приоритет, ответственные лица, дедлайн, описание и т.д.

Документ задачи в Coda.io может быть организован по-разному, но обычно он содержит следующие элементы:

* Список задач - это основная таблица, в которой отображаются все задачи проекта. Каждая задача представляется строкой таблицы и содержит информацию о задаче, такую как ее название, описание, статус и т.д.
* Столбцы таблицы - это колонки таблицы, которые содержат информацию о каждой задаче, такую как ее статус, приоритет, ответственные лица, дедлайн и т.д.
* Формы - это специальные инструменты, которые позволяют быстро добавлять новые задачи в список задач. Формы могут содержать поля для заполнения информации о задаче, такие как ее название, описание, ответственные лица и т.д.
* Фильтры - это инструменты, которые позволяют фильтровать список задач по определенным критериям, таким как статус, приоритет или ответственные лица. Фильтры могут помочь быстро находить нужные задачи и отслеживать их статус.
* Дашборды - это специальные страницы, которые позволяют визуализировать информацию о задачах в проекте. Дашборды могут содержать графики, диаграммы и другие элементы, которые позволяют быстро оценить текущее состояние проекта.
* Комментарии - это инструменты, которые позволяют общаться с другими участниками проекта и обсуждать детали задачи. Комментарии могут быть полезны для решения проблем и координации действий между участниками проекта.

Документ «Задачи» представлен в соответствии с рисунком 2.5

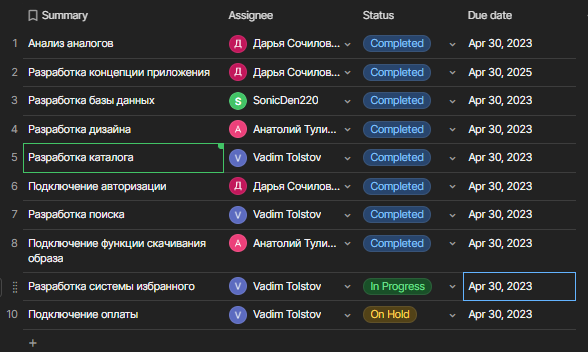


Рисунок 2.5 Документ «Задачи»

Coda.io использовалась для планирования и проектирования рабочего процесса разработки приложения, включая создание документа "Проект" с описанием целей и задач проекта, временной диаграммы проекта, документа "Команда" для управления участниками команды и их ролями, а также документа "Задачи" для распределения задач и контроля за их выполнением.

С помощью Coda.io были упрощены процессы планирования и контроля за выполнением задач, а также улучшена коммуникация и сотрудничество между членами команды. В документе "Задачи" были созданы списки задач, отсортированные по приоритетам и срокам выполнения, что позволило членам команды легко отслеживать свой прогресс и прогресс других участников команды. Кроме того, документ "Временная диаграмма" помогал планировать и отслеживать график выполнения проекта, а документ "Команда" упрощал процесс управления участниками команды и их ролями.

В целом, использование Coda.io повысило эффективность и производительность команды в процессе разработки приложения и позволило легко контролировать ход проекта и прогресс выполнения задач.

2.2 Разработка UserStory

User Story 1: Регистрация

Как пользователь, я хочу зарегистрироваться в приложении, чтобы мне стали доступны все функциональные возможности, и я мог найти игру, которую хочу, скачать ее образ, добавить в избранное, или купить картридж.

Действия пользователя:

* открыть приложение;
* нажать на кнопку «Регистрация»;
* заполнить форму регистрации;
* нажать на кнопку «Зарегистрироваться»;
* перейти в каталог.

Ожидаемый результат:

Пользователь успешно зарегистрирован в приложении и может авторизоваться для использования всех функций.

User Story 2: Авторизация

Как зарегистрированный пользователь, я хочу авторизоваться в приложении, чтобы мне стали доступны все функциональные возможности, и я мог найти игру, которую хочу, скачать ее образ, добавить в избранное, или купить картридж.

Действия пользователя:

* открыть приложение;
* ввести свою почту и пароль;
* нажать на кнопку «Войти»;
* перейти на экран каталога.

Ожидаемый результат:

Пользователь успешно авторизован в приложении и может начать использовать все функциональные возможности.

User Story 3: Выбор игры

Как авторизованный пользователь, я хочу ознакомиться с каталогом игр и информацией о них.

Действия пользователя:

* открыть приложение;
* выбрать игру в каталоге;
* просмотреть информацию.

Ожидаемый результат:

Пользователь получил необходимую ему информацию.

User Story 4: Добавление игры в избранное

Как авторизованный пользователь, я хочу добавить игру в избранное, чтобы иметь быстрый доступ к ней.

Действия пользователя:

* открыть приложение;
* выбрать игру в каталоге;
* отметить избранной на иконку;
* открыть раздел избранное.

Ожидаемый результат:

Пользователь добавил игру в избранное и может получить информацию о ней.

User Story 5: Заказ картриджа

Как авторизованный пользователь, я хочу заказать картридж игры.

Действия пользователя:

* открыть приложение;
* выбрать игру в каталоге;
* нажать на кнопку заказа картриджа;
* ввести необходимые данные в том числе номер карты;
* провести оплату.

Ожидаемый результат:

Пользователь заказал картридж и получит его в ближайшем времени.

User Story 6: Загрузка образа для эмулятора

Как авторизованный пользователь, я хочу загрузить образ игры для запуска ее на эмуляторе.

Действия пользователя:

* открыть приложение;
* выбрать игру в каталоге;
* нажать на кнопку загрузки образа.

Ожидаемый результат:

Пользователь загрузил образ для эмулятора и может запустить его у себя.

# 2.3 Разработка базы данных

Проектирование базы данных является одним из важных шагов при разработке такого приложения, как репозиторий ретро-игр. В этом процессе была определена структура данных, которая используется в системе. В качестве подхода к проектированию были применены инфологическая и даталогическая модели, а в качестве No-Code среды для реализации был выбран сервис Airtable.

Инфологическая модель отображает предметную область системы без учета ограничений, связанных с реализацией в конкретной СУБД. Инфологическая модель базы данных для приложения RetroPlay изображена на рисунке 2.6:



Рисунок 2.6 Инфологическая модель базы данных

Исходя из представленной модели, можно увидеть, что в ней имеется две сущности: видеоигра и консоль. У сущности «видеоигра» имеется множество атрибутов, таких как:

* Идентификатор (ключевое поле);
* Название игры;
* Описание игры;
* Изображение для каталога;
* Изображение игры на ее соответствующей странице;
* Дата выхода игры;
* ROM-файл для скачивания игры и последующего ее запуска на эмуляторе;
* Платформы, на которых игра выходила;
* Жанр игры;
* Количество игроков;
* Цена игры при покупке ее картриджа;
* Компания-издатель игры;
* Восприятие, оценки критиков, мнения игроков.

Сущность «консоль» включает в себя следующие атрибуты:

* Идентификатор (ключевое поле);
* Название/модель консоли;
* Изображение консоли;
* Страна-производитель;
* Кабель подключения.

В даталогической модели более конкретно отображены названия атрибутов, как они используются в системе управления базами данных Airtable с учетом типов данных. Даталогическая модель RetroPlay представлена на рисунке 2.7:

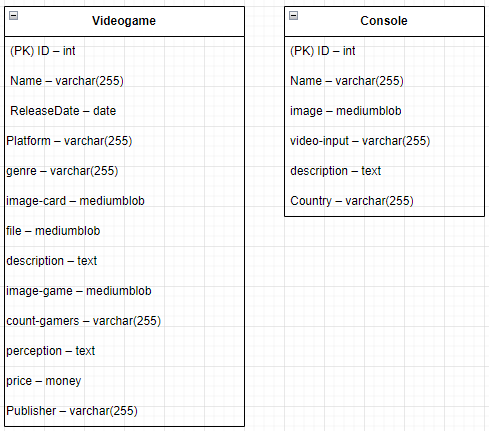


Рисунок 2.7 Даталогическая модель базы данных

Airtable предоставляет удобный интерфейс для работы с таблицами, а также совместимость с Bravo Studio. Airtable прост в использовании благодаря своему интуитивно понятному интерфейсу и гибкой структуре данных. В отличие от традиционных реляционных баз данных, где необходимо предварительно определить схему и структуру данных, в Airtable можно быстро и легко создавать таблицы, добавлять поля и заполнять их информацией. Кроме того, Airtable предоставляет широкий выбор типов полей, таких как текст, числа, даты, файлы, ссылки и т.д., что позволяет точно определить тип данных, которые необходимы для каждой конкретной таблицы и поля. В результате использование Airtable позволяет экономить время и снижать затраты на разработку БД, что особенно важно для небольших проектов и стартапов.

Часть таблицы видеоигр в интерфейсе Airtable представлена на рисунке 2.8:

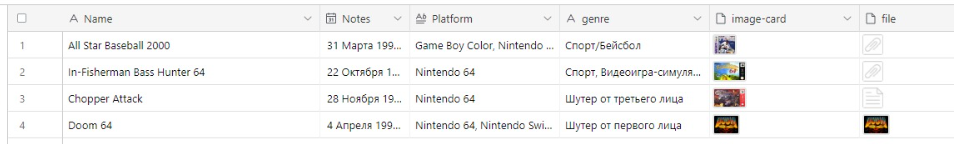


Рисунок 2.8 Интерфейс Airtable

В целом, использование Airtable для создания БД позволяет упростить процесс проектирования и управления данными, особенно из-за участия нескольких человек в проекте.

# 2.3 Архитектура приложения

Архитектура приложения – это общий план и организация всех компонентов, которые входят в его состав.

Архитектура приложения включает в себя следующие компоненты:

* Bravo Studio – платформа для создания мобильных приложений без написания кода.
* Firebase как база данных для работы с аккаунтами пользователей и аутентификации.
* Airtable как база данных для хранения информации об играх и консолях.
* Окно «Авторизация» – это первый экран, который увидит пользователь при запуске приложения.
* Окно «Регистрация» – это экран, который позволяет новым пользователям зарегистрироваться в приложении.
* Окно «Каталог игр» – это экран, на котором пользователь может просмотреть список доступных игр.
* Окно «Каталог консолей» – это экран, на котором пользователь может просмотреть список доступных игровых консолей.
* Окно «Избранное» – это экран, на котором пользователь может увидеть список игр, которые он добавил в свой список избранных.
* Окно «Оплата» – это экран, на котором пользователь может оплатить выбранные игры.
* Окно «Карточка игры» – это экран, на котором пользователь может узнать подробную информацию об определенной игре, такую как описание, оценки, отзывы и т.д.
* Окно «Карточка консоли» – это экран, на котором пользователь может узнать подробную информацию об определенной игровой консоли, такую как описание, характеристики и т.д.
* Stripe – это сервис онлайн-платежей, который позволяет пользователям оплачивать покупки в приложении.

Схема архитектуры приложения представлена на рисунке 2.9:

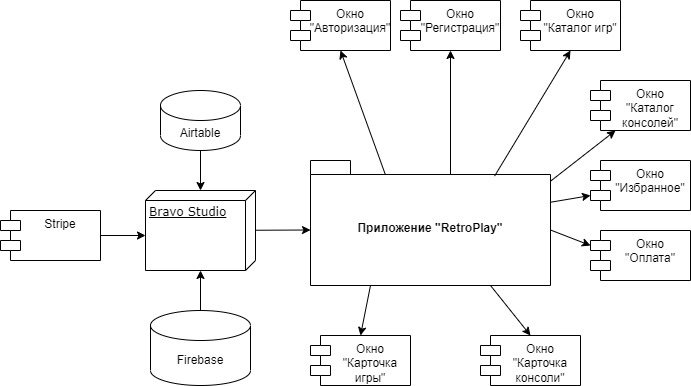


Рисунок 2.9 Архитектура приложения

Каждый компонент выполняет определенную функцию в приложении и взаимодействует с другими компонентами, чтобы создать полнофункциональное приложение.

Приложение «Репозиторий ретро-игр» включает в себе следующие функциональные блоки:

* каталог игр;
* поиск;
* добавление в избранное;
* оплата;
* скачивание;
* авторизация.

Далее будут подробно рассмотрены этапы разработки для каждого из этих блоков.

# 2.1.1 Разработка функции «каталог игр»

Процесс разработки функционального блока «каталог игр» состоит из следующих этапов:

1. Планирование:

Команда разработчиков обговаривает требования и определяет функциональные возможности, которые должны быть реализованы в каталоге игр. Затем команда обсуждает возможные способы реализации функциональных возможностей и определяет необходимые ресурсы.

1. Проектирование:

На основе обсуждений команда разработчиков создает макеты экранов для каталога игр в Figma. Дизайнер разрабатывает дизайн экранов на основе макетов, и в результате получается прототип, который команда использует в дальнейшей разработке.

1. Настройка Airtable:

Создается база данных игр в Airtable, в которую затем будут добавляться игры для отображения в каталоге. Определяются необходимые поля для каждой игры, такие как название, жанр, год выпуска, описание, обложка и т.д.

1. Разработка бэкенда:

С помощью Firebase команда разработчиков настраивает регистрацию и аутентификацию пользователей. Затем создается серверная логика, которая позволяет получать данные из базы данных Airtable и отображать их в каталоге игр. В частности, команда разработчиков настраивает функцию, которая отображает все игры, добавленные в базу данных Airtable.

1. Разработка фронтенда:

С помощью Bravo Studio команда разработчиков создает экраны для отображения игр в каталоге. Для каждой игры создается отдельный экран, на котором отображается информация о ней, такая как название, обложка, жанр, год выпуска, описание и т.д.

1. Тестирование:

После завершения разработки команда разработчиков проводит тестирование каталога игр, чтобы убедиться, что все функциональные возможности работают корректно и что пользовательский интерфейс является интуитивно понятным и удобным в использовании.

# 2.1.2 Разработка функции «поиск»

Разработка функционального блока «поиск» для нашего приложения на Bravo Studio может быть разделена на несколько этапов:

1. Планирование и проектирование функционала поиска:

* Определение параметров поиска, которые должны быть доступны для пользователя.
* Определение логики поиска и фильтрации результатов.
* Проектирование интерфейса для ввода параметров поиска и отображения результатов.

1. Реализация функционала поиска:

* Создание формы для ввода параметров поиска.
* Настройка подключения к базе данных Airtable, чтобы получить доступ к таблице с информацией об играх.
* Написание кода для выполнения запроса к базе данных с учетом выбранных пользователем параметров поиска.
* Обработка результатов запроса и отображение их на странице в удобном формате для пользователя.

1. Тестирование и отладка:

* Проведение тестового поиска и проверка корректности результатов.
* Исправление ошибок и улучшение производительности.

1. Документация:

* Создание документации для пользователей, которая объясняет, как использовать функционал поиска и какие параметры доступны.

# 2.1.3 Разработка функции «оплата»

Для разработки и реализации платёжной системы были проделаны следующие шаги:

1. Определение требований:

Сначала определяются, какие требования должны удовлетворяться, чтобы сделать блок оплаты полезным для пользователей. Например, предоставить безопасный и удобный способ оплаты, позволяющий пользователям покупать игры в нашем каталоге.

1. Дизайн:

Следующим шагом является создание дизайна блока оплаты. Можно использовать Figma для создания макетов и прототипов, чтобы определить, как будет выглядеть и работать блок оплаты.

1. Разработка на Bravo Studio:

После создания макета команда переходит к разработке блока оплаты на платформе Bravo Studio. Для этого используются инструменты Bravo Studio для создания пользовательского интерфейса, интеграции с Stripe для обработки оплаты и обработки заказов.

1. Интеграция с Stripe:

Для обработки оплаты используется Stripe, который предоставляет множество инструментов для обработки платежей.

1. Тестирование:

После разработки и интеграции блока оплаты необходимо протестировать его, чтобы убедиться, что он работает должным образом и не вызывает проблем у пользователей. Для этого можно использовать инструменты тестирования Bravo Studio и Stripe для тестирования блока оплаты.

# 2.1.4 Разработка функции «добавление в избранное»

В процессе создания функции «добавление в избранное» были проделаны следующие этапы:

1. Проектирование интерфейса

Первым этапом является проектирование интерфейса. Для этого используется Figma. В этом блоке необходимо создать кнопку для добавления в избранное, которая будет размещена на странице деталей игры. Также необходимо создать страницу «Избранное», на которой будут отображаться все игры, добавленные пользователем в избранное. На этом этапе также определяются требования к функциональности.

1. Создание базы данных

Для сохранения информации о том, какие игры были добавлены в избранное, используется Airtable. На этом этапе создается таблица, в которой будут храниться данные об играх, добавленных в избранное. Также создается связь между этой таблицей и таблицей с играми.

1. Создание компонента

На этом этапе используется Bravo Studio для создания компонента, который будет реализовывать функциональность добавления игр в избранное. Создается действие «Добавить в избранное» и связывается с кнопкой добавления в избранное. При нажатии на эту кнопку, данные об игре сохраняются в таблицу в Airtable.

1. Реализация функциональности «Избранное»

На этом этапе создается страница «Избранное» и подключается к таблице с играми в Airtable. Также создается функция, которая будет выводить список игр, добавленных в избранное.

1. Тестирование

Последний этап – тестирование функциональности добавления в избранное. Проверяется, что кнопка работает корректно и что данные о добавленных играх сохраняются в таблице в Airtable. Также проверяется, что страница «Избранное» отображает правильный список игр.

# 2.1.5 Разработка функции «скачивание»

Разработка функции «скачивание» состоит из следующих этапов:

1. Определение требований: необходимо определить требования к функциональности блока «скачивание игр». Например, какие типы игр могут быть загружены, какие форматы файлов поддерживаются, какие механизмы скачивания будут использоваться и т.д.
2. Проектирование интерфейса: на основе определенных требований необходимо разработать интерфейс блока «скачивание игр». В Figma можно создать макет экрана с кнопкой загрузки и информацией о загруженных файлах.
3. Создание базы данных: в Airtable необходимо создать базу данных для хранения информации о загруженных файлах и связать ее с приложением на Bravo Studio.
4. Создание блока: необходимо создать блок, который будет обрабатывать запросы пользователя на загрузку файлов и сохранение информации о загруженных файлах в базе данных. Для этого можно использовать Firebase Cloud Functions, которые будут запускаться при запросе на загрузку.
5. Тестирование: после написания кода необходимо протестировать его, чтобы убедиться в правильности работы блока «скачивание игр».
6. Внесение изменений: если в процессе тестирования были выявлены ошибки, необходимо внести изменения в код и повторно протестировать блок «скачивание игр».
7. Развертывание: после тестирования и внесения изменений блок «скачивание игр» можно развернуть на сервере и включить его в работу приложения.

# 2.1.6 Разработка функции «авторизация»

В разработку функции авторизации входят следующие этапы:

1. Анализ требований: перед началом разработки необходимо определить, какие данные нужны для авторизации пользователя и как они будут храниться. Также необходимо определить, какие методы авторизации будут доступны (например, через социальные сети, электронную почту или номер телефона).
2. Создание макета интерфейса: после того, как определены требования, необходимо создать макет интерфейса для авторизации. На этом этапе определяются элементы интерфейса, такие как поля для ввода логина и пароля, кнопки и т.д.
3. Разработка бизнес-логики: на этом этапе определяется, как будет проходить процесс авторизации. Например, при входе в приложение нужно проверять логин и пароль пользователя, а при использовании социальных сетей – получать данные пользователя из API социальной сети.
4. Разработка клиентской части: после разработки бизнес-логики и создания макета интерфейса необходимо начать разработку клиентской части. На этом этапе создаются все необходимые элементы интерфейса, а также взаимодействие с бизнес-логикой.
5. Разработка серверной части: для хранения информации о пользователях, их авторизации и т.д. необходимо создать серверную часть. На этом этапе определяется, какие технологии будут использоваться для хранения данных (например, база данных) и создания серверной части.
6. Тестирование и отладка: после того, как разработка функции авторизации завершена, необходимо провести тестирование и отладку приложения. На этом этапе проверяется, как функция авторизации работает на разных устройствах, в разных ситуациях и т.д.
7. Внедрение: после успешного тестирования и отладки функция авторизации внедряется в приложение.

Таким образом, в процессе разработки были созданы функциональные блоки, такие как «каталог игр», «поиск», «оплата», «добавление в избранное», «скачивание игр» и «авторизация», каждый из которых был тщательно проработан и проверен на работоспособность. В ходе разработки нашего приложения мы использовали инструменты Bravo Studio, Airtable, Firebase и Figma, что значительно ускорило процесс и повысило его эффективность.

# 2.2 Функционал приложения

Страница каталога имеет в себе кнопки переключения вкладок, кнопку «избранное», а также кнопку на переход к своему профилю.

Каталог включает в себя две вкладки – «игры» и «консоли».

Каталог игр изображен на рисунке 2.10



Рисунок 2.10 Каталог игр

Если нажать на кнопку справа от игры, то она добавится в избранное.

Каталог для консолей изображен на рисунке 2.11



Рисунок 2.11 Каталог консолей

Функционал каталога консолей идентичен функционалу каталога игр.

При выборе игры или консоли в каталоге пользователь переходит на страницу соответствующего товара. Страница игры представлена на рисунке 2.12



Рисунок 2.12 Страница игры

Страница игры предоставляет общую информацию об игре, такую как количество игроков, компанию-разработчика, дату выхода, жанр, описание игры, а также оценки критиков и то, на каких платформах игра выходила. В верхней части страницы находятся промоматериалы, скриншоты, трейлеры или кадры из игры. Под этой частью находится кнопка для покупки картриджа, а в нижней части страницы находится кнопка для скачивания ROM-файла игры.

На рисунке 2.13 представлена страница о консоли.



Рисунок 2.13 Страница о консоли

На странице о консоли представлено изображение консоли, страна-производитель, а также сводка информации о самой консоли и совместимой с ней техникой, как например мониторы и кабели. Также в описании сказано, с чем в комплекте приставка поставляется (например, с какими контроллерами).

Следующая функция – оплата.

Оплата и оформление заказа представлены на рисунке 2.14

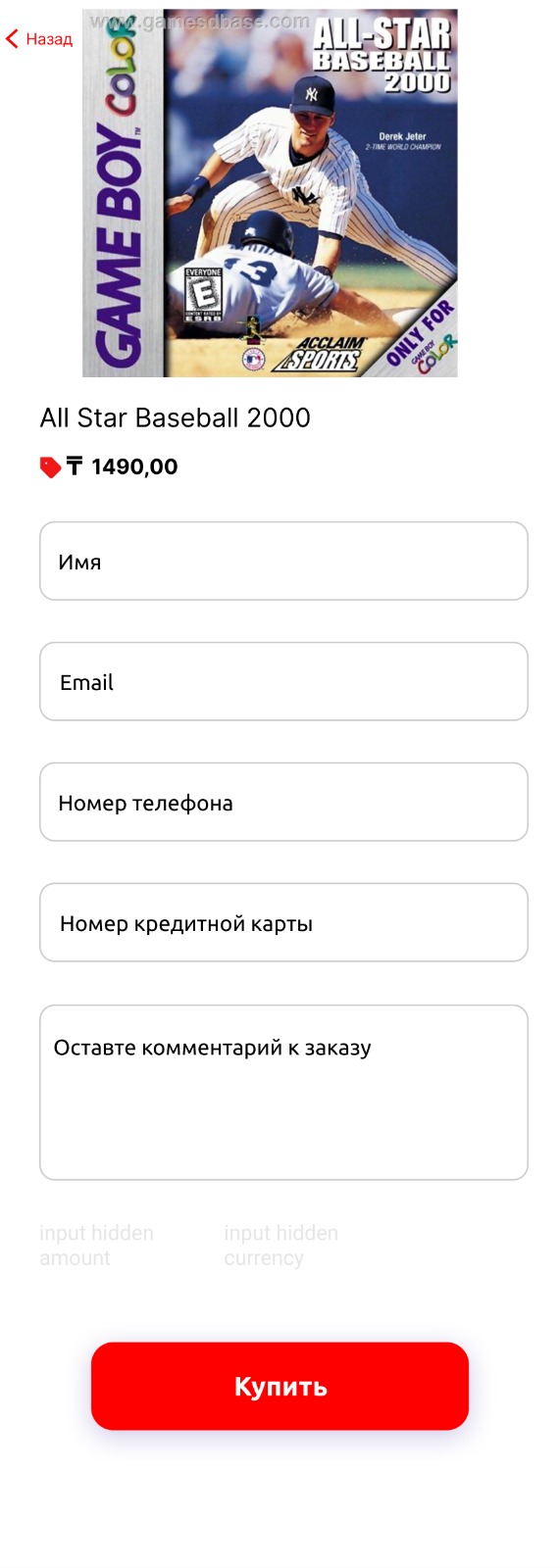


Рисунок 2.14 Меню оформления заказа

Исходя из рисунка, при оформлении заказа пользователь может оставить свои контактные данные, свой номер кредитной карты, а также дополнительно предполагается возможность оставить комментарий к заказу.

Следующая функция – добавление игры в избранное и просмотр списка избранных игр. Меню «избранное» приведено на рисунке 2.15



Рисунок 2.15 Меню «Избранное»

При добавлении игры в избранное, она отображается здесь. Игру можно удалить из избранного в любой момент при повторном нажатии на соответствующую кнопку.

Следующая функция – регистрация и авторизация. Экран регистрации представлен на рисунке 2.16.

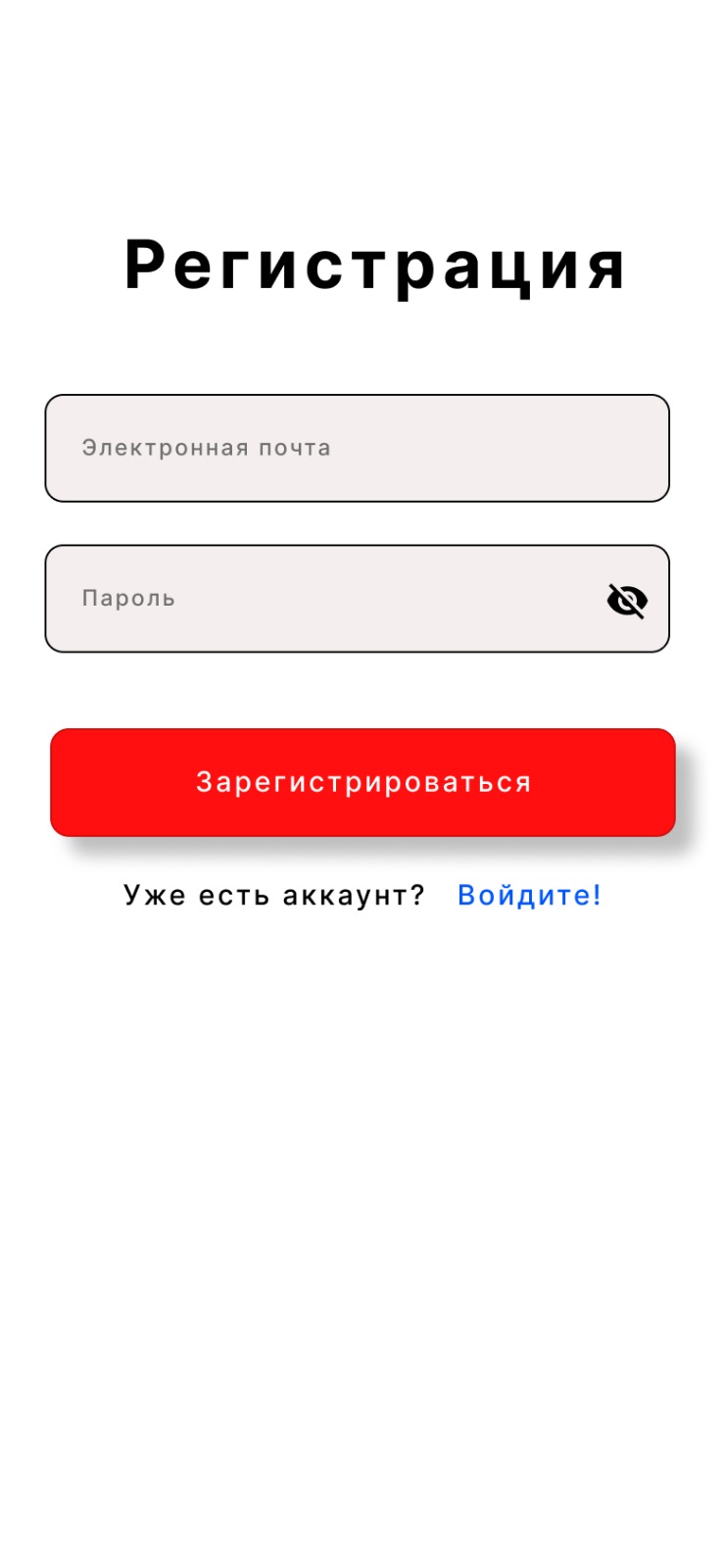


Рисунок 2.16 Регистрация

На экране регистрации пользователь может ввести свою электронную почту и пароль. Также, если у пользователя уже есть аккаунт, под кнопкой «Зарегистрироваться» есть кнопка «Войдите!». Соответственно, есть меню для входа в свой аккаунт, которое представлено на рисунке 2.17

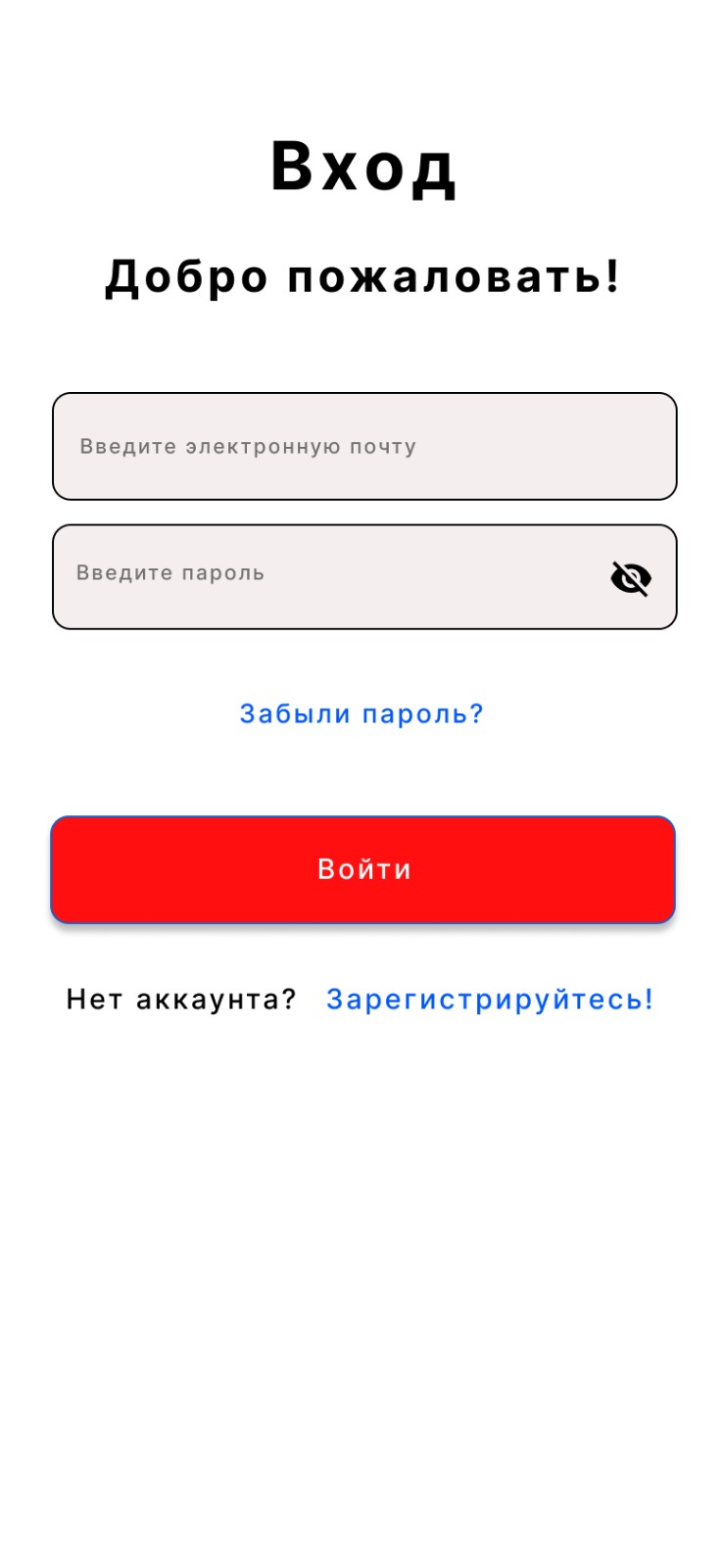


Рисунок 2.17 Меню входа.

На меню входа в аккаунт предполагается ввод своих данных – электронной почты и пароля. Помимо этого, если пользователь не помнит пароль, то есть кнопка восстановления. Также в нижней части этого меню есть перенаправление на форму регистрации.

Если пользователь попытается войти в аккаунт, когда он уже находится в системе, то будет отображено следующее оповещение, которое изображено на рисунке 2.18.

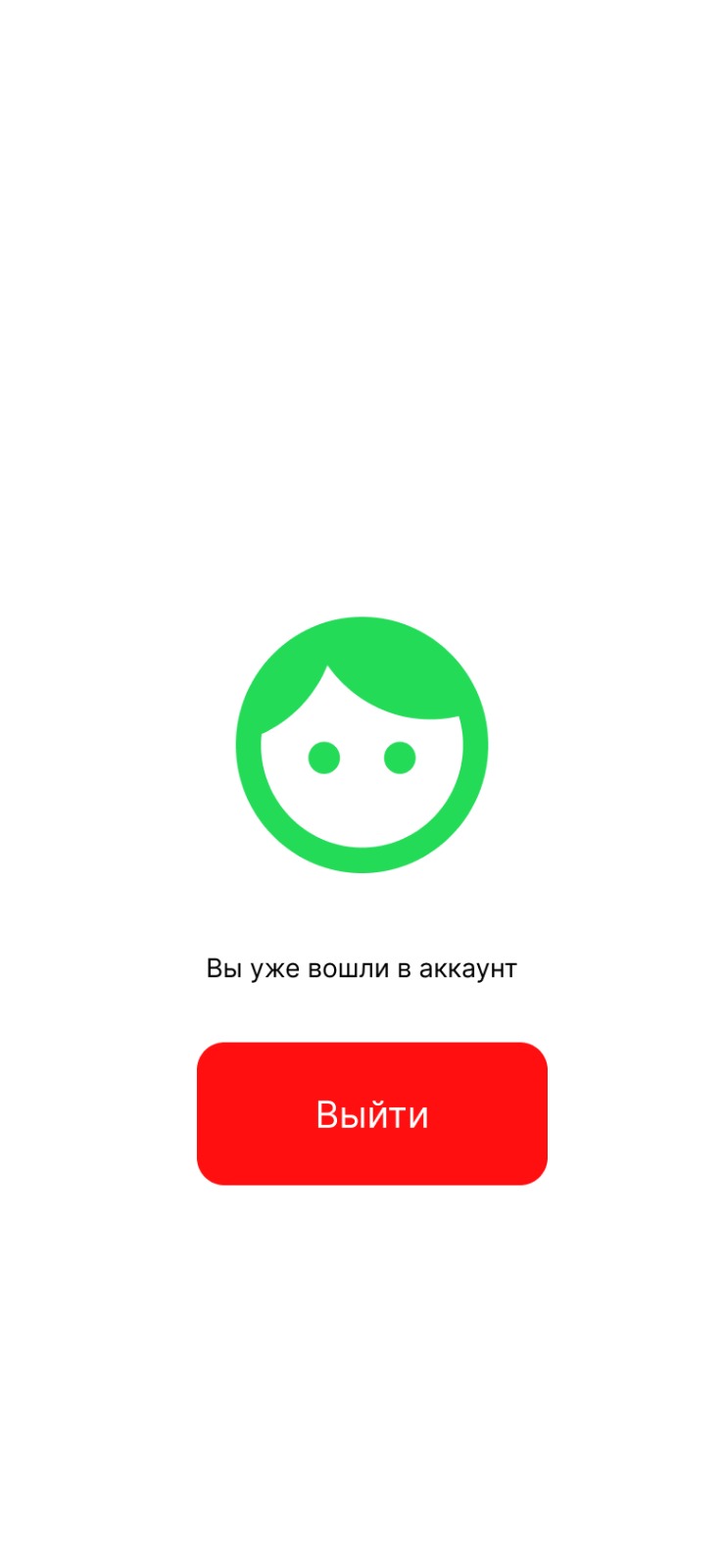


Рисунок 2.18 Оповещение о том, что пользователь в системе

# ЗАКЛЮЧЕНИЕ

В заключение можно сказать, что проект по разработке мобильного приложения «Репозиторий ретро-игр» был успешно завершен. Цели и задачи были достигнуты, которые включали создание приложения, предоставляющего удобный доступ к ретро-играм, с безопасной регистрацией и аутентификацией пользователей, а также с удобной базой данных игр и информации об игроках.

В процессе разработки были приобретены множество новых навыков и знаний. Были изучены множество различных инструментов и технологий, таких как Bravo Studio, Airtable, Firebase и Figma. Получены навыки разработки дизайна интерфейса, управления базой данных.

Итоговое приложение включает в себя такой функционал как каталог, поиск, оплата, добавление в избранное и авторизация, что позволяет пользователям наслаждаться ретро-играми.

Также стоит отметить, что проект по разработке мобильного приложения «Репозиторий ретро-игр» является актуальным и полезным в наше время, когда многие люди любят играть в ретро-игры, но не всегда могут найти удобный и безопасный способ доступа к ним. Это приложение предоставляет такой способ.

В целом, проект по разработке мобильного приложения «Репозиторий ретро-игр» был полезным опытом для всей команды. Команда научилась решать проблемы и достигать поставленных целей.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1 Ушакова, Е.В. Методические указания по выполнению дипломных проектов/работ для бакалавриата направления информационно-коммуникационные технологии: учебное пособие / Е.В. Ушакова, Л.В. Долматова. – Петропавловск, 2019. -99 с.;

2 Страница обучения Bravo Studio [Электронный ресурс]. – Режим доступа: URL: https://projects.bravostudio.app/tutorials;

3 Официальный сайт Figma [Электронный ресурс]. – Режим доступа: URL: https://www.figma.com/;

4 Начало работы с Airtable [Электронный ресурс]. – Режим доступа: URL: https://support.airtable.com/docs/getting-started-with-airtables-web-api;

5 Документация Firebase [Электронный ресурс]. – Режим доступа: URL: https://firebase.google.com/docs?hl=ru;

6 Документация Coda.io [Электронный ресурс]. – Режим доступа: URL: https://coda.io/resources/learn;

7 Пирская Л.В.: Разработка мобильных приложений в среде Android Studio. – Южный Федеральный Университет, 2019;

8 Танг Дж.: Умные мобильные проекты с TensorFlow. – ДМК Пресс, 2018;

9 Нефедов Ю., Чурсин А., Мамедова Н.: Разработка кроссплатформенных мобильных приложений – перспективные методы и стандартные практики. – Синергия, 2021;

10 Федотенко М.: Разработка мобильных приложений. Первые шаги. – Лаборатория знаний, 2019;

11 Еранссон А.: Эффективное использование потоков в операционной системе Android. – ДМК-Пресс, 2014;

12 Семенчук В.: Мобильное приложение как инструмент бизнеса. – Альпина Диджитал, 2016;

13 Колисниченко Д.: Программирование для Android. 3-е издание. – БХВ-Петербург, 2020;

14 Клифтон Я.: Проектирование пользовательского интерфейса Android. 2-е издание. – ДМК-Пресс, 2017;

15 Нормативные документы, отражающие направления деятельности СКУ им. М.Козыбаева, образовательные программы, реализуемые высшим учебным заведением, обучаемой специальностью, видами, функциями и задачами будущей профессиональной деятельности.