МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_\_\_\_\_\_Павленко Г.Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема: Разработка мобильного приложения управления личной библиотекой

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИСп 22-1  Родионова Александра Денисовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2025

Аннотация

Данный курсовой проект посвящен разработке приложения CatClap, интегрированного в Telegram, которое предоставляет пользователям возможность взаимодействовать с виртуальными питомцами, использовать внутриигровую валюту и участвовать в таблице лидеров. Актуальность проекта обусловлена популярностью мессенджеров как платформ для приложений и ростом интереса к интерактивным играм с элементами монетизации.

Для реализации приложения использованы фреймворк Vue.js для разработки фронтенда, Flask для серверной части и база данных MongoDB для хранения информации о пользователях, питомцах, транзакциях и таблице лидеров. Telegram Web Apps API обеспечивает интеграцию мини-приложения в Telegram, что делает его доступным и удобным для пользователей.

Приложение предоставляет пользователям базового котика, с которым можно взаимодействовать через кнопки. Реализованы функции покупки внутриигровой валюты, приобретения новых окрасов для питомца, добавления друзей и просмотра таблицы лидеров. В работе рассмотрены этапы проектирования архитектуры приложения, создание пользовательского интерфейса, реализация бизнес-логики и интеграция с Telegram Web Apps API.

Ключевые слова: Telegram, Vue.js, Flask, MongoDB, приложение, виртуальный питомец, монетизация.  
Объём работы: 71 страница.  
Приложение: CatClap (Telegram-приложение).  
Среда разработки: Visual Studio Code.  
GitHub: https://github.com/ARodidoll/Coursework.git

Оглавление

[Введение 4](#_Toc196694309)

[1. Описание предметной области 6](#_Toc196694310)

[1.1. Аналоги разрабатываемого приложения 6](#_Toc196694311)

[1.2 Техническое задание 8](#_Toc196694312)

[1.3 Описание структуры программы 11](#_Toc196694313)

[2. Описание разработки приложения 16](#_Toc196694314)

[2.1 Обоснование средств разработки 16](#_Toc196694315)

[2.2 Разработка интерфейса 20](#_Toc196694316)

[2.3 Разработка логики работы приложения. Схема взаимодействия компонентов 32](#_Toc196694317)

[2.4 Описание переменных, компонентов, классов и подпрограмм 33](#_Toc196694318)

[3. Тестирование и установка приложения 34](#_Toc196694319)

[Заключение 39](#_Toc196694320)

[Список использованной литературы 40](#_Toc196694321)

[Приложение А 41](#_Toc196694322)

[Листинг программы 41](#_Toc196694323)

[Приложение Б 67](#_Toc196694324)

[Руководство пользователя 67](#_Toc196694325)

# Введение

В современном мире мессенджеры, такие как Telegram, становятся не только средством общения, но и полноценной платформой для создания приложений, которые предоставляют пользователям уникальный функционал. Telegram Web Apps API открывает новые возможности для разработчиков, позволяя создавать мини-приложения, которые запускаются прямо внутри мессенджера. Это делает приложения более доступными, удобными и привлекательными для широкой аудитории.

Одной из актуальных задач является разработка приложений, которые сочетают в себе элементы развлечения, монетизации и социального взаимодействия. Такие приложения позволяют пользователям не только получать удовольствие от игрового процесса, но и взаимодействовать друг с другом, соревноваться и делиться своими достижениями. Виртуальные питомцы, как элемент игрового процесса, остаются популярной темой, так как они создают эмоциональную связь между пользователем и приложением.

Целью данного курсового проекта является разработка приложения CatClap, которое предоставляет пользователям возможность взаимодействовать с виртуальными питомцами, использовать внутриигровую валюту для улучшений и участвовать в таблице лидеров. Приложение ориентировано на создание увлекательного пользовательского опыта, который включает управление питомцем, покупку улучшений, добавление друзей и участие в рейтинговых соревнованиях.

Для реализации проекта используются современные технологии. Фронтенд приложения разработан на Vue.js, что обеспечивает модульность, гибкость и удобство разработки пользовательского интерфейса. Серверная часть построена на Flask, который предоставляет REST API для обработки запросов от фронтенда. Для хранения данных используется MongoDB, что позволяет эффективно управлять информацией о пользователях, питомцах, транзакциях и таблице лидеров. Telegram Web Apps API обеспечивает интеграцию приложения с Telegram, что делает его доступным для пользователей без необходимости установки дополнительных программ.

В рамках проекта были поставлены задачи по проектированию архитектуры приложения, созданию пользовательского интерфейса, реализации бизнес-логики и интеграции с Telegram. Также проведено тестирование приложения для обеспечения его стабильной работы и защиты данных пользователей. Приложение CatClap демонстрирует возможности современных технологий для создания интерактивных решений, которые сочетают в себе развлекательные и социальные элементы.

# 1. Описание предметной области

## 1.1. Аналоги разрабатываемого приложения

На рынке существует множество Telegram-ботов и приложений, которые предоставляют схожий функционал. Однако большинство из них ориентированы на выполнение отдельных задач, тогда как CatClap объединяет несколько направлений в одном продукте. Рассмотрим основные аналоги:

1. Telegram-боты для развлечений:

Боты, такие как @gamebot, предоставляют пользователям мини-игры и развлекательный контент.



Рисунок 1 – Логотип @gamebot



Рисунок 2 – Интерфейс @gamebot

Однако такие боты не включают функционал управления питомцами или мероприятий.

1. Боты с монетизацией:

Telegram-боты, такие как @donate, предоставляют возможность сбора донатов.



Рисунок 3 – Логотип @donate

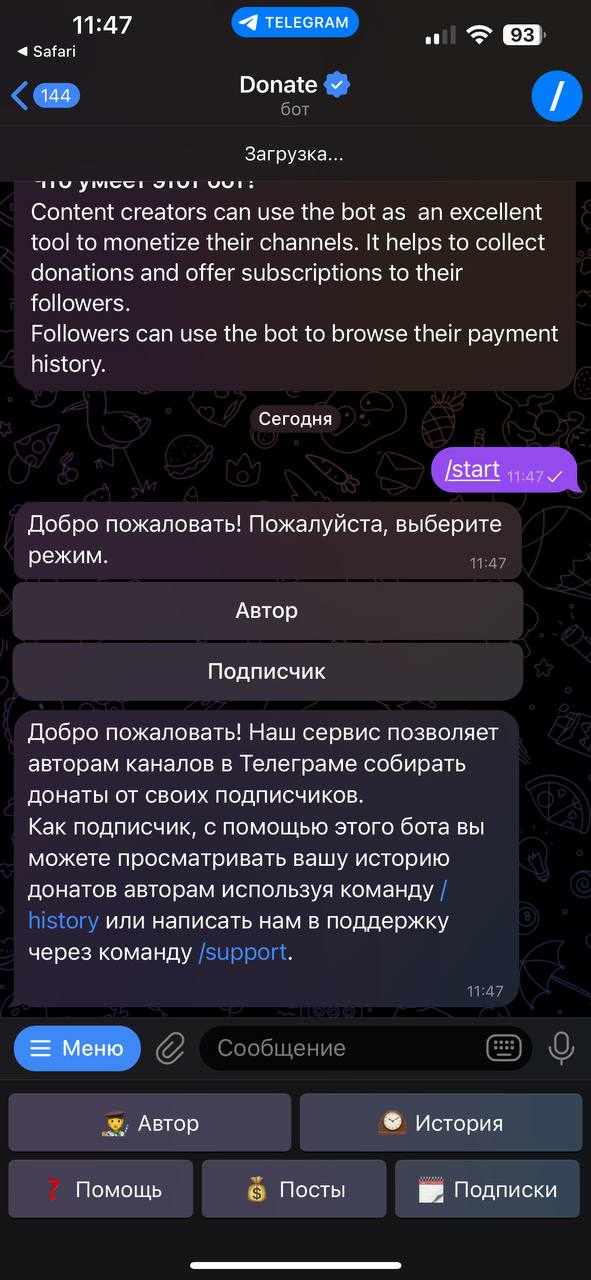


Рисунок 4 – Интерфейс @donate

Однако такие боты не предлагают развлекательного контента или взаимодействия с питомцами.

# 1.2 Техническое задание

Приложение CatClap представляет собой Telegram Web App, которое позволяет пользователям взаимодействовать с виртуальными питомцами, покупать внутриигровые улучшения, добавлять друзей и участвовать в таблице лидеров. Приложение ориентировано на создание увлекательного пользовательского опыта, сочетающего элементы развлечения и социального взаимодействия.

Приложение должно предоставлять пользователям базового котика при первом входе. Котик обладает уникальными характеристиками, такими как окрас и настроение. Взаимодействие с котиком осуществляется через кнопки, такие как "Погладить", "Покормить" или "Поиграть". Эти действия изменяют настроение котика, которое может быть, например, "Счастливым" или "Грустным". В зависимости от настроения котик издает соответствующие звуки. Пользователь может наблюдать за изменением состояния питомца в реальном времени.

Для монетизации в приложении реализована внутриигровая валюта — CatCoin и CatClap. Пользователи могут приобретать валюту за реальные деньги через встроенные платежные системы Telegram. Виртуальная валюта используется для покупки новых окрасов для питомца. Окрасы разделены по ценовым категориям, что позволяет пользователям выбирать улучшения в зависимости от их предпочтений и возможностей. Магазин предоставляет пользователю удобный интерфейс для выбора и покупки окрасов.

Приложение также предоставляет возможность добавлять других пользователей в список друзей. Пользователь может отправить запрос на добавление в друзья, просматривать список друзей и их профили. Это способствует социальному взаимодействию между пользователями. В отдельном разделе приложения отображается таблица лидеров, где пользователи могут видеть, кто заработал больше всего CatCoin и CatClap. Таблица обновляется в реальном времени, что стимулирует пользователей к активности.

Приложение должно быть устойчивым к сбоям и корректно обрабатывать ошибки, такие как некорректный ввод данных, без потери информации. Все данные пользователей, включая информацию о котиках, внутриигровой валюте и друзьях, должны быть защищены от несанкционированного доступа. Время отклика сервера на запросы не должно превышать двух секунд, а приложение должно поддерживать одновременную работу более 1000 пользователей без снижения производительности.

Для обеспечения качества работы приложения необходимо провести юнит-тестирование всех функций, протестировать взаимодействие между серверной частью и фронтендом, а также проверить производительность при высокой нагрузке. Безопасность данных пользователей также должна быть протестирована. Приложение будет сопровождаться регулярными обновлениями, исправлением ошибок и добавлением новых функций. Для пользователей и разработчиков будет создана документация, описывающая API, функции и возможности приложения.

Разработка приложения включает несколько этапов. На подготовительном этапе проводится сбор и анализ требований, а также определение целевой аудитории. На этапе проектирования создаются макеты взаимодействия пользователя с приложением и описывается структура базы данных. На этапе разработки реализуется функциональность приложения, включая REST API для взаимодействия с фронтендом. После этого проводится тестирование, включающее функциональное, нагрузочное и интеграционное тестирование, а также исправление выявленных ошибок. На этапе сопровождения приложение будет регулярно обновляться и поддерживаться.

Программа состоит из нескольких модулей. Модуль управления пользователями отвечает за регистрацию и авторизацию через Telegram, а также за хранение данных о пользователях, таких как Telegram ID, баланс внутриигровой валюты и список друзей. Модуль управления котиком обрабатывает взаимодействия с питомцем, изменяет его настроение и воспроизводит звуки. Модуль монетизации позволяет пользователям покупать внутриигровую валюту через Telegram Payments и использовать её для приобретения новых окрасов. Модуль таблицы лидеров подсчитывает и отображает лидеров по количеству CatCoin и CatClap, обновляя данные в реальном времени. Все данные хранятся в базе данных MongoDB, которая обеспечивает быстрый доступ и защиту информации. Взаимодействие с пользователями осуществляется через Telegram Web App, который предоставляет удобный интерфейс с кнопками и мультимедиа.

# 1.3 Описание структуры программы

Приложение CatClap реализовано основе архитектуры клиент-сервер. Telegram-бот выступает в роли клиента, который взаимодействует с серверной частью через REST API. Серверная часть обрабатывает запросы, выполняет бизнес-логику и взаимодействует с базой данных MongoDB. Основные компоненты программы включают Telegram-бот, серверную часть и базу данных.

* Приложение (Telegram Web App)

Реализовано с использованием фреймворка Vue.js и интегрировано в Telegram через Web Apps API. Оно предоставляет пользователям удобный и интерактивный интерфейс для взаимодействия с приложением. Основные функции мини-приложения включают отображение информации о питомце, покупку новых окрасов, управление друзьями и просмотр таблицы лидеров

Состоит из нескольких компонентов. Компонент AppHeader.vue отображает заголовок приложения и элементы навигации, такие как кнопки для перехода в магазин, таблицу лидеров или профиль пользователя. Компонент CatSection.vue отвечает за отображение текущего состояния питомца, включая его окрас и настроение. Магазин реализован в компоненте CatStore.vue, где пользователи могут выбирать и приобретать окрасы за внутриигровую валюту. Компонент Leaderboard.vue отображает таблицу лидеров, где пользователи могут видеть топ участников по количеству CatCoin и CatClap. Для управления списком друзей используется компонент Friend.vue, который позволяет добавлять и удалять друзей, а также просматривать их профили.

Отправляет запросы к серверу через REST API для выполнения различных действий, таких как изменение настроения питомца, покупка окраса или добавление друга. Сервер возвращает данные, которые отображаются пользователю в виде обновленного интерфейса.

* Серверная часть

Серверная часть приложения CatClap реализована с использованием фреймворка Flask. Она предоставляет REST API для обработки запросов от Telegram-бота. Серверная часть отвечает за выполнение бизнес-логики, взаимодействие с базой данных MongoDB и возврат результатов Telegram-боту.

Модуль управления пользователями обрабатывает регистрацию пользователей при первом взаимодействии с ботом. Telegram ID пользователя используется как уникальный идентификатор, который сохраняется в базе данных. Этот модуль также предоставляет функции для управления списком друзей, добавления новых друзей и получения списка друзей пользователя.

Модуль управления котиком отвечает за взаимодействие пользователя с виртуальным питомцем. При первом входе пользователя создается базовый котик с уникальными характеристиками, такими как окрас и настроение. Взаимодействия пользователя с котиком, такие как "Погладить", "Покормить" или "Поиграть", изменяют настроение котика. Этот модуль также возвращает звуки, соответствующие текущему настроению питомца.

Модуль монетизации реализует функционал покупки внутриигровой валюты (CatCoin и CatClap) через Telegram Payments. Пользователи могут использовать валюту для приобретения новых окрасов для котика. Модуль обрабатывает запросы на покупку валюты, обновляет баланс пользователя и сохраняет историю транзакций в базе данных.

Модуль таблицы лидеров подсчитывает и возвращает данные о лидерах по количеству CatCoin и CatClap. Таблица лидеров обновляется в реальном времени и доступна всем пользователям. Этот модуль сортирует данные пользователей по количеству валюты и возвращает топ-10 лидеров.

Серверная часть взаимодействует с базой данных MongoDB, где хранятся данные о пользователях, котиках, транзакциях и таблице лидеров. Она обеспечивает высокую производительность, устойчивость к сбоям и защиту данных от несанкционированного доступа.

* База данных

Для хранения данных в приложении CatClap используется база данных MongoDB. Она обеспечивает надежное хранение информации о пользователях, их котиках, внутриигровой валюте, транзакциях и таблице лидеров. Структура базы данных организована в виде коллекций, каждая из которых отвечает за определенный тип данных.

Коллекция users хранит информацию о пользователях. В этой коллекции сохраняются уникальный Telegram ID пользователя, баланс внутриигровой валюты (CatCoin и CatClap), а также список друзей. Эти данные используются для идентификации пользователя, управления его балансом и взаимодействия с другими пользователями.

Коллекция cats содержит данные о виртуальных питомцах. Для каждого котика сохраняются его владелец (ссылка на Telegram ID пользователя), текущий окрас и настроение. Эта информация обновляется при взаимодействии пользователя с котиком, например, при изменении настроения или покупке нового окраса.

Коллекция transactions используется для хранения истории покупок. В ней фиксируются данные о транзакциях, включая Telegram ID пользователя, тип покупки (например, CatCoin или CatClap), количество приобретенной валюты или стоимость окраса, а также временная метка операции. Эта информация позволяет отслеживать действия пользователей и анализировать их активность.

Коллекция leaderboard отвечает за хранение данных для таблицы лидеров. В этой коллекции сохраняются Telegram ID пользователей, их текущий баланс CatCoin и CatClap. Эти данные используются для формирования таблицы лидеров, которая обновляется в реальном времени и отображает топ-10 пользователей с наибольшим количеством валюты.

База данных MongoDB обеспечивает быстрый доступ к данным, их надежное хранение и защиту от несанкционированного доступа. Она играет ключевую роль в работе приложения, поддерживая все основные функции, такие как управление пользователями, взаимодействие с котиками, обработка покупок и отображение таблицы лидеров.

* Взаимодействие компонентов

Приложение CatClap построено на основе архитектуры клиент-сервер, где Telegram-бот выступает в роли клиента, а серверная часть обрабатывает запросы и взаимодействует с базой данных MongoDB. Взаимодействие между компонентами организовано следующим образом:

Пользователь взаимодействует с приложением через Telegram-бота, отправляя команды или нажимая на интерактивные кнопки. Например, пользователь может отправить команду /start для регистрации или нажать кнопку "Погладить", чтобы взаимодействовать с котиком. Telegram-бот принимает эти действия и формирует соответствующий запрос к серверу через REST API.

Серверная часть обрабатывает запросы, выполняя бизнес-логику, соответствующую действию пользователя. Например, при регистрации пользователя сервер проверяет наличие его Telegram ID в базе данных и, если пользователь новый, создает запись в коллекции users. Если запрос связан с взаимодействием с котиком, сервер обновляет данные о настроении питомца в коллекции cats и возвращает результат.

База данных MongoDB используется для хранения и управления данными. Сервер взаимодействует с коллекциями базы данных для выполнения операций чтения, записи, обновления или удаления данных. Например, при покупке окраса сервер обновляет баланс пользователя в коллекции users и сохраняет информацию о транзакции в коллекции transactions.

После обработки запроса сервер возвращает результат Telegram-боту. Это может быть текстовое сообщение, данные о текущем состоянии котика, обновленный баланс пользователя или таблица лидеров. Telegram-бот отображает результат пользователю в удобной форме, используя текстовые сообщения, кнопки или медиафайлы.

Такое взаимодействие компонентов обеспечивает гибкость и масштабируемость приложения, позволяя легко добавлять новые функции и поддерживать стабильную работу при высокой нагрузке.

# 2. Описание разработки приложения

## 2.1 Обоснование средств разработки

Для реализации приложения CatClap был выбран язык программирования Python. Python является одним из самых популярных языков благодаря своей простоте, читаемости и широкому набору библиотек. Он идеально подходит для разработки серверной части приложений, особенно в сочетании с фреймворком Flask. Flask был выбран как основной инструмент для создания REST API, так как он легковесен, гибок и предоставляет все необходимые возможности для построения серверной архитектуры.



Рисунок 5 – Логотип Python

Для работы с базой данных используется MongoDB. Это документно-ориентированная база данных, которая позволяет хранить данные в формате JSON-подобных документов. MongoDB была выбрана из-за её гибкости, масштабируемости и высокой производительности при работе с большими объемами данных. Для взаимодействия с MongoDB в Python используется библиотека pymongo, которая предоставляет удобный интерфейс для выполнения операций чтения, записи, обновления и удаления данных.



Рисунок 6 – Логотип MongoDB

Для тестирования REST API использовался инструмент Postman. Он позволяет проверять корректность работы серверной части, отправляя запросы и анализируя ответы. Для разработки пользовательского интерфейса Telegram-бота использовались встроенные возможности Telegram, такие как команды и кнопки.



Рисунок 7 – Логотип Postman

Telegram-бот реализован с использованием библиотеки python-telegram-bot. Эта библиотека предоставляет удобный интерфейс для работы с Telegram Bot API, позволяя обрабатывать команды, отправлять сообщения, кнопки и медиафайлы. Она была выбрана из-за своей популярности, активного сообщества и обширной документации.

Для разработки макета дизайна интерфейса и общего вида программы будет использована программа Figma (логотип представлен на рисунке 10), которая позволяет быстро создавать визуальный макет приложения, путем размещение простых фигур, а также использования внутренних инструментов приложения.



Рисунок 8 – Логотип Figma

Также Figma позволяет подключать сторонние плагины, разработанные комьюнити дизайнеров, что позволяет упростить процесс разработки визуальных макетов в десятки раз, а следовательно, и понижает порог входа в профессиональное использование приложения. Некоторые из них представлены на рисунках 11 и 12.

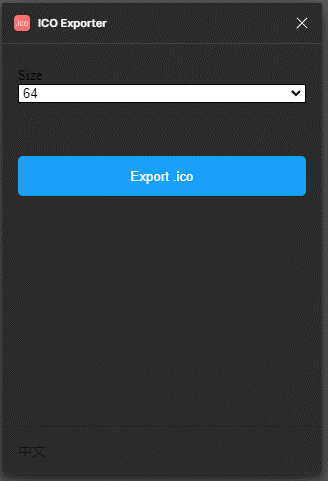


Рисунок 9 – Плагин в Figma ICO Exporter

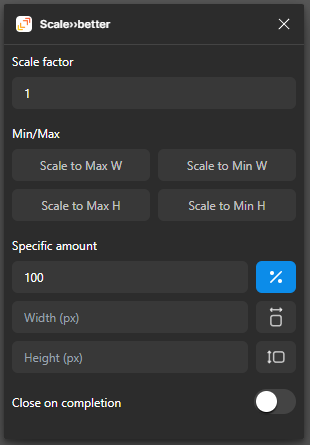


Рисунок 10 – Плагин Figma Scale better

## 2.2 Разработка интерфейса

Интерфейс программы был начат с разработки макета в Figma. Макет представлен на рисунках 11-.16

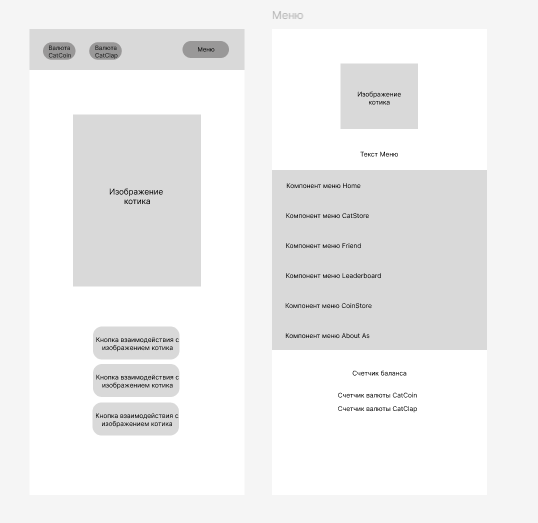


Рисунок 11 - Макет интерфейса

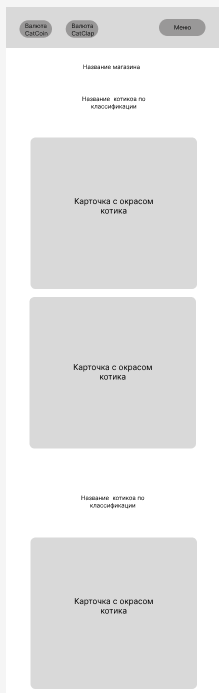


Рисунок 12 - Макет интерфейса

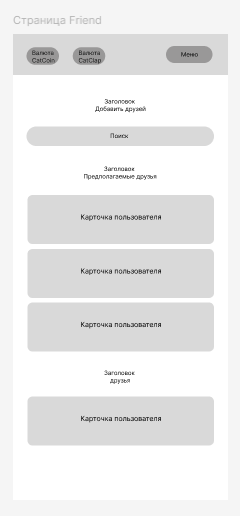


Рисунок 13 - Макет интерфейса



Рисунок 14 - Макет интерфейса

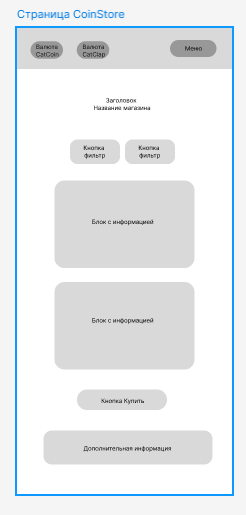


Рисунок 15 - Макет интерфейса

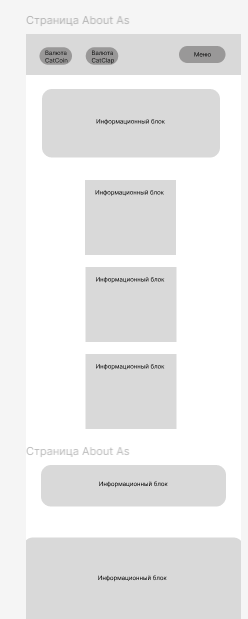
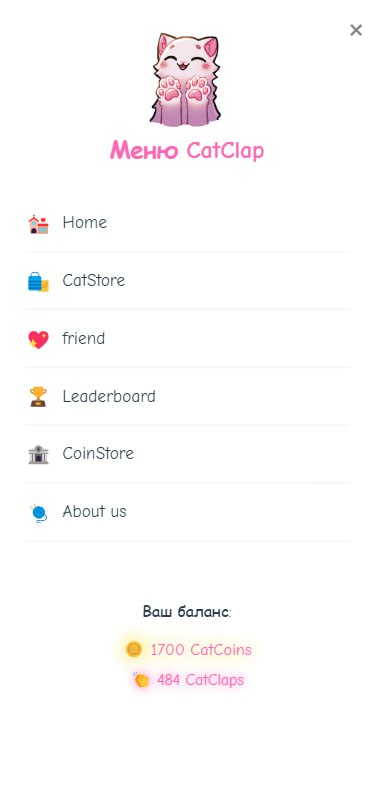


Рисунок 16 - Макет интерфейса

Интерфейс приложения CatClap реализован в виде мини-приложения (мини-аппа) Telegram с использованием фреймворка Vue.js. Telegram Web Apps API позволяет запускать мини-приложение прямо внутри Telegram, предоставляя пользователям удобный и интерактивный интерфейс. Весь фронтенд построен на Vue.js, что обеспечивает модульность, гибкость и возможность повторного использования компонентов.

Основные компоненты интерфейса включают:

AppHeader.vue - компонент, отвечающий за отображение заголовка приложения. Он содержит элементы навигации, такие как кнопки для перехода в магазин, таблицу лидеров или профиль пользователя.



Рисинук 17 – Меню

CatSection.vue - компонент, который дополняет компонент CatStore.vue.

CatStore.vue - компонент магазина, где пользователи могут приобретать новые окрасы для своих питомцев. Магазин отображает доступные окрасы, их стоимость в CatCoin или CatClap.

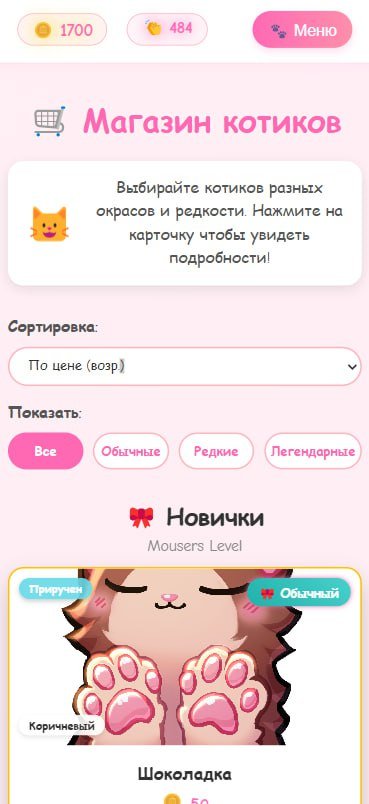


Рисунок 17 – Страница CatStore

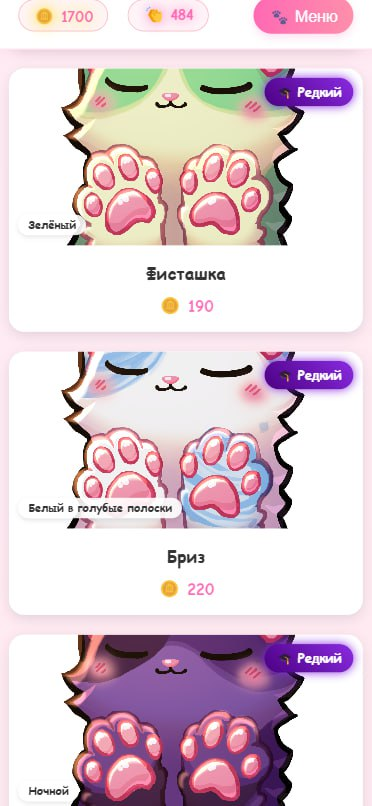


Рисунок 18 – Страница CatStore

Leaderboard.vue - компонент таблицы лидеров, который отображает топ пользователей по количеству CatCoin и CatClap. Таблица обновляется в реальном времени, предоставляя пользователям актуальную информацию о лидерах.



Рисунок 19 – Страница Leaderboard

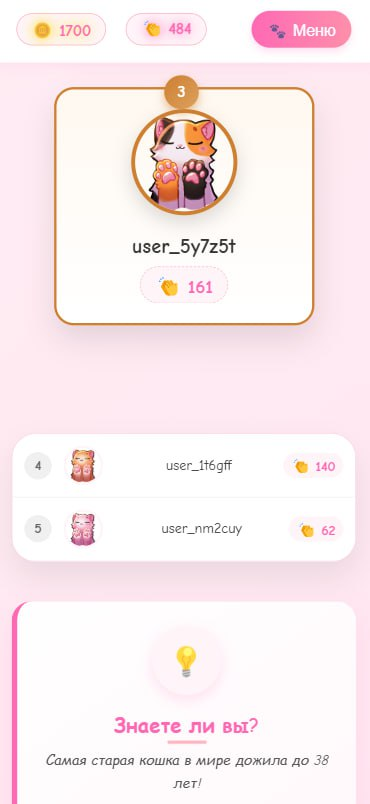


Рисунок 20 – Страница Leaderboard

Friend.vue - компонент для управления списком друзей. Пользователи могут добавлять новых друзей, удалять существующих и просматривать их профили.

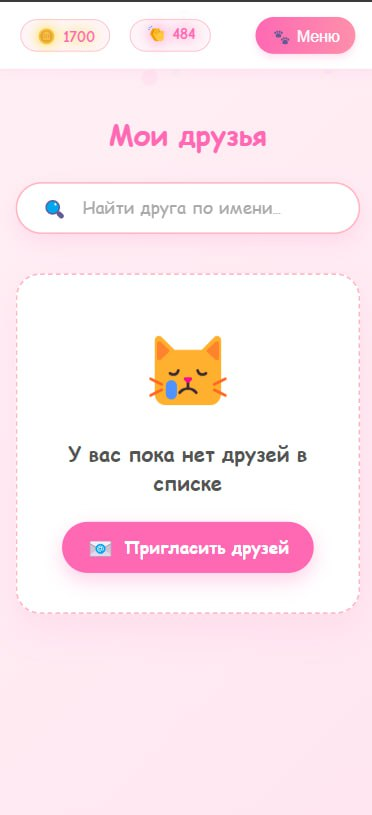


Рисунок 21 – Страница Friend

Для улучшения пользовательского опыта используются статические файлы, такие как изображения и аудиофайлы. Изображения питомцев (cat\_happy.png, cat\_sad.png и другие) отображают текущее настроение котика, а аудиофайлы (meow.mp3, happy.mp3) воспроизводятся в зависимости от действий пользователя.

Интерфейс приложения разработан таким образом, чтобы пользователи могли легко перемещаться между разделами, такими как магазин, таблица лидеров и профиль. Telegram Web Apps API обеспечивает плавную интеграцию мини-приложения с Telegram, позволяя запускать его прямо внутри чата. Это делает взаимодействие с приложением удобным и интуитивно понятным.

## 2.3 Разработка логики работы приложения. Схема взаимодействия компонентов

Логика работы приложения основана на взаимодействии между Telegram-ботом, серверной частью и базой данных. Пользователь отправляет команды или нажимает кнопки в Telegram-боте. Бот формирует запрос к серверу через REST API, передавая данные о действии пользователя. Сервер обрабатывает запрос, выполняет бизнес-логику и взаимодействует с базой данных MongoDB. После обработки сервер возвращает результат Telegram-боту, который отображает его пользователю.

Схема взаимодействия компонентов:

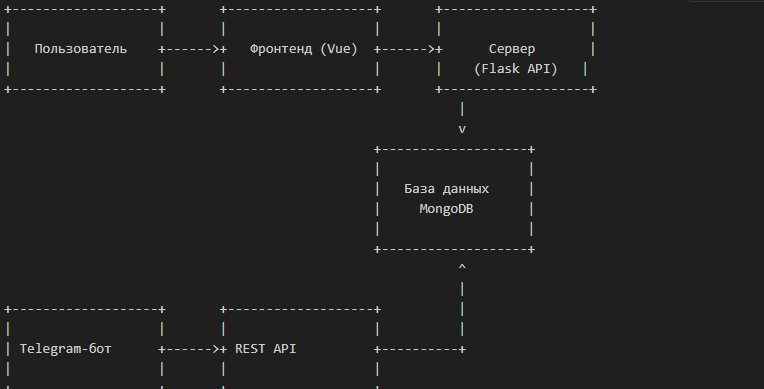


Рисунок 22 – Схема взаимодействия компонентов

## 2.4 Описание переменных, компонентов, классов и подпрограмм

В таблице ниже приведено описание ключевых переменных, компонентов, классов и подпрограмм, используемых в приложении:

Таблица 1 - Описание переменных, компонентов, классов и подпрограмм

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| user\_id | Строка | Уникальный идентификатор пользователя в Telegram. |
| cat\_id | Строка | Уникальный идентификатор питомца. |
| event\_id | Строка | Уникальный идентификатор мероприятия. |
| coins | Целое число | Количество виртуальной валюты у пользователя. |
| add\_cat() | Функция | Создает нового питомца и сохраняет его данные в базе данных. |
| get\_user\_stats() | Функция | Возвращает статистику пользователя (например, количество питомцев, донаты). |
| register\_user() | Функция | Регистрирует нового пользователя в базе данных. |
| MongoClient | Класс | Класс из библиотеки pymongo для подключения к базе данных MongoDB. |
| Cat | Класс | Модель данных для питомца. |
| Event | Класс | Модель данных для мероприятия. |
| Participant | Класс | Модель данных для участника мероприятия. |
| App.vue | Компонент Vue.js | Главный компонент фронтенда. |
| CatSection.vue | Компонент Vue.js | Компонент для отображения информации о питомцах. |
| Leaderboard.vue | Компонент Vue.js | Компонент для отображения таблицы лидеров. |
| Telegram Bot API | Внешний компонент | API для взаимодействия с Telegram-ботом. |

# 3. Тестирование и установка приложения

Для использования Telegram-приложения CatClap необходимо выполнить следующие шаги:

1. Добавление бота в Telegram:

* Найдите бота CatClap в Telegram, используя его имя пользователя (например, @CatClapBot).
* Нажмите кнопку «Start» для начала работы с ботом.

Тестирование программы:

Таблица 2 - Тест-кейсы к программе

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  теста | Название теста | Описание | Шаги воспроизведения | Ожидаемый результат | Статус |
| 1 | Взаимодействие с котиком | Проверка изменения настроения котика при взаимодействии. | 1. Нажать кнопку "Погладить". 2. Проверить настроение котика. | Настроение котика изменяется на "Счастливый", воспроизводится соответствующий звук. | Пройден |
| 2 | Покупка окраса | Проверка покупки нового окраса за внутриигровую валюту. | 1. Перейти в магазин. 2. Выбрать окрас. 3. Нажать кнопку "Купить". | Баланс пользователя уменьшается на стоимость окраса, окрас питомца обновляется. | Пройден |
| 3 | Просмотр таблицы лидеров | Проверка отображения таблицы лидеров. | 1. Перейти в раздел "Таблица лидеров". | Отображается актуальная таблица лидеров с топ-10 пользователей по количеству CatCoin и CatClap. | Пройден |
| 4 | Покупка внутриигровой валюты | Проверка покупки CatCoin через Telegram Payments. | 1. Перейти в раздел "Баланс". 2. Выбрать количество CatCoin. 3. Подтвердить оплату. | Баланс CatCoin увеличивается на указанное количество, транзакция сохраняется в базе данных. | Пройден |
| 5 | Отображение настроения котика | Проверка корректного отображения настроения котика после взаимодействий. | 1. Нажать кнопку "Поиграть". 2. Проверить настроение котика. | Настроение котика изменяется на "Счастливый", отображается соответствующее изображение. | Пройден |
| 6 | Обновление таблицы лидеров | Проверка обновления данных в таблице лидеров после изменения баланса. | 1. Совершить покупку CatCoin. 2. Перейти в раздел "Таблица лидеров". | Таблица лидеров обновляется, пользователь отображается на соответствующей позиции. | Пройден |

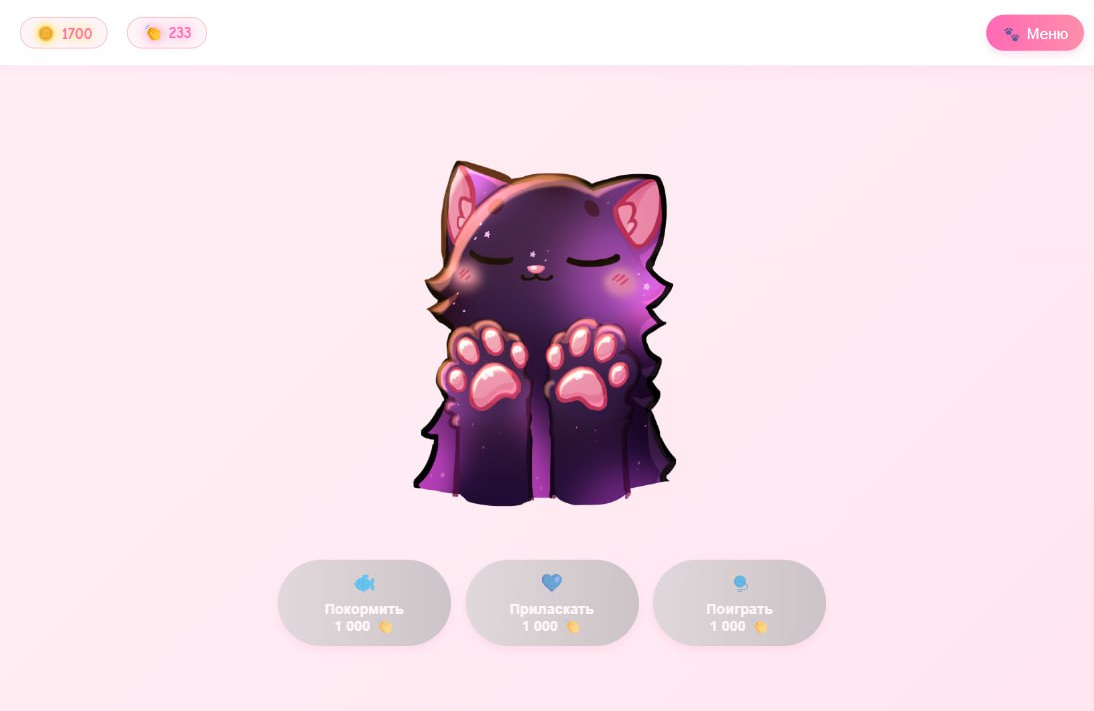


Рисунок 23 – Результат тест-кейса №1, 2, 5

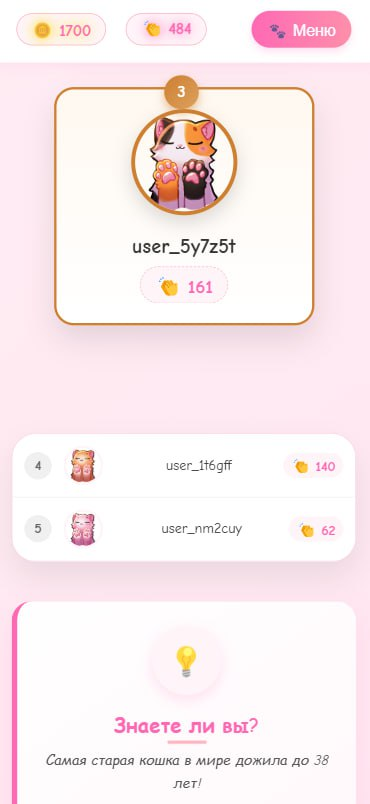


Рисунок 24 – Результат тест-кейса №3

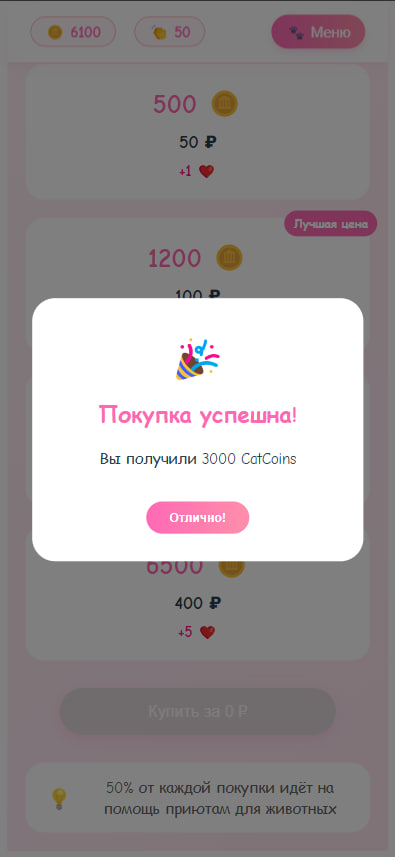


Рисунок 25 – Результат тест-кейса №4



Рисунок 26 – Результат тест-кейса №6

# Заключение

В рамках данной работы было разработано приложение CatClap, интегрированное в Telegram, которое предоставляет пользователям возможность взаимодействовать с виртуальными питомцами, использовать внутриигровую валюту и участвовать в таблице лидеров. Основной целью проекта было создание удобного, функционального и масштабируемого приложения, которое сочетает в себе элементы развлечения и социального взаимодействия.

В процессе разработки были выполнены следующие задачи: проведен анализ предметной области и аналогов, составлено техническое задание, реализованы серверная часть на Flask и фронтенд на Vue.js, а также интеграция с Telegram Web Apps API. Приложение предоставляет пользователям базового котика, с которым можно взаимодействовать через кнопки, а также функции покупки внутриигровой валюты, приобретения новых окрасов, добавления друзей и просмотра таблицы лидеров.

Приложение CatClap успешно реализует поставленные задачи, предоставляя пользователям простой и увлекательный способ взаимодействия с виртуальными питомцами. Реализация механизма монетизации позволяет использовать приложение в коммерческих целях, а таблица лидеров стимулирует пользователей к активности и социальному взаимодействию.

В дальнейшем проект может быть расширен за счет добавления новых функций, таких как дополнительные виды взаимодействия с питомцем, расширение ассортимента окрасов, улучшение аналитики и интеграция с другими платформами. Разработанное приложение демонстрирует возможности современных технологий и их применение для создания удобных и функциональных решений, ориентированных на пользователей.

# Список использованной литературы

1. Flask. Документация по фреймворку Flask [Электронный ресурс]. – Режим доступа: [https://flask.palletsprojects.com/](vscode-file://vscode-app/c:/Users/user/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
2. MongoDB. Документация по MongoDB [Электронный ресурс]. – Режим доступа: [https://www.mongodb.com/docs/](vscode-file://vscode-app/c:/Users/user/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
3. Telegram. Telegram Bot API [Электронный ресурс]. – Режим доступа: [https://core.telegram.org/bots/api](vscode-file://vscode-app/c:/Users/user/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
4. Vue.js. Документация по Vue.js [Электронный ресурс]. – Режим доступа: [https://vuejs.org/](vscode-file://vscode-app/c:/Users/user/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
5. Python Software Foundation. Python 3 Documentation [Электронный ресурс]. – Режим доступа: [https://docs.python.org/3/](vscode-file://vscode-app/c:/Users/user/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
6. Postman. Документация по Postman [Электронный ресурс]. – Режим доступа: [https://learning.postman.com/docs/](vscode-file://vscode-app/c:/Users/user/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) (дата обращения: 24.04.2025).
7. Pymongo. Документация по библиотеке Pymongo [Электронный ресурс]. – Режим доступа: [https://pymongo.readthedocs.io/](vscode-file://vscode-app/c:/Users/user/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
8. Telegram. Руководство по разработке Telegram-ботов [Электронный ресурс]. – Режим доступа: [https://core.telegram.org/bots](vscode-file://vscode-app/c:/Users/user/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
9. MDN Web Docs. Документация по веб-технологиям [Электронный ресурс]. – Режим доступа: [https://developer.mozilla.org/](vscode-file://vscode-app/c:/Users/user/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)
10. Григорий Петров. Flask для профессионалов. – М.: Издательство «Питер», 2020. – 320 с.

# Приложение А

# Листинг программы

Leaderboard.vue

<template>

<div class="app-container">

<AppHeader :catClapCount="catClapCount" :catCoinCount="catCoinCount" />

<div class="top-container">

<div class="confetti-container">

<div

v-for="i in 20"

:key="i"

class="confetti"

:style="confettiStyle(i)"

></div>

</div>

<div class="top-header">

<div class="title-wrapper">

<h1 class="top-title">🏆 Leaderboard</h1>

<div class="title-underline"></div>

</div>

<div class="filter-box">

<button

class="filter-btn"

:class="{ active: activeFilter === 'claps' }"

@click="activeFilter = 'claps'"

>

👏 Claps

</button>

<button

class="filter-btn"

:class="{ active: activeFilter === 'coins' }"

@click="activeFilter = 'coins'"

>

🪙 Coins

</button>

</div>

</div>

<div class="top-content">

<!-- Топ 3 игрока -->

<div class="top-players">

<!-- Второе место -->

<div v-if="topPlayers[1]" class="player-card second-place">

<div class="medal silver">2</div>

<div class="ribbon silver-ribbon"></div>

<img

:src="getPlayerAvatar(topPlayers[1])"

alt="Avatar"

class="player-avatar"

:class="{ 'current-user': isCurrentUser(topPlayers[1]) }"

/>

<h3 class="player-name">{{ topPlayers[1].name }}</h3>

<div class="player-score">

<span v-if="activeFilter === 'claps'"

>👏 {{ formatNumber(topPlayers[1].claps) }}</span

>

<span v-else>🪙 {{ formatNumber(topPlayers[1].coins) }}</span>

</div>

</div>

<!-- Первое место -->

<div v-if="topPlayers[0]" class="player-card first-place">

<div class="medal gold">1</div>

<div class="ribbon gold-ribbon"></div>

<img

:src="getPlayerAvatar(topPlayers[0])"

alt="Avatar"

class="player-avatar"

:class="{ 'current-user': isCurrentUser(topPlayers[0]) }"

/>

<div class="crown">👑</div>

<div class="sparkle-container">

<span class="sparkle" v-for="n in 5" :key="n">✨</span>

</div>

<h3 class="player-name">{{ topPlayers[0].name }}</h3>

<div class="player-score">

<span v-if="activeFilter === 'claps'"

>👏 {{ formatNumber(topPlayers[0].claps) }}</span

>

<span v-else>🪙 {{ formatNumber(topPlayers[0].coins) }}</span>

</div>

</div>

<!-- Третье место -->

<div v-if="topPlayers[2]" class="player-card third-place">

<div class="medal bronze">3</div>

<div class="ribbon bronze-ribbon"></div>

<img

:src="getPlayerAvatar(topPlayers[2])"

alt="Avatar"

class="player-avatar"

:class="{ 'current-user': isCurrentUser(topPlayers[2]) }"

/>

<h3 class="player-name">{{ topPlayers[2].name }}</h3>

<div class="player-score">

<span v-if="activeFilter === 'claps'"

>👏 {{ formatNumber(topPlayers[2].claps) }}</span

>

<span v-else>🪙 {{ formatNumber(topPlayers[2].coins) }}</span>

</div>

</div>

</div>

<!-- Остальные игроки -->

<div class="players-list">

<div

v-for="(player, index) in remainingPlayers"

:key="player.id"

class="list-item"

:class="{ 'current-user-item': isCurrentUser(player) }"

>

<div class="list-rank">{{ index + 4 }}</div>

<img

:src="getPlayerAvatar(player)"

alt="Avatar"

class="list-avatar"

:class="{ 'current-user': isCurrentUser(player) }"

/>

<div class="list-name">{{ player.name }}</div>

<div class="list-score">

<span v-if="activeFilter === 'claps'"

>👏 {{ formatNumber(player.claps) }}</span

>

<span v-else>🪙 {{ formatNumber(player.coins) }}</span>

</div>

</div>

</div>

<!-- Блок интересных фактов -->

<div class="fun-fact-container">

<div class="fun-fact">

<div class="fun-fact-icon-container">

<div class="fun-fact-icon">💡</div>

</div>

<div class="fun-fact-content">

<h3>Знаете ли вы?</h3>

<p>{{ currentFact }}</p>

</div>

<button class="fun-fact-next" @click="nextFact">Еще факт</button>

</div>

</div>

</div>

</div>

</div>

</template>

<script setup>

import { ref, computed, onMounted } from "vue";

import AppHeader from "./AppHeader.vue";

import { useTelegram } from "@/composables/useTelegram";

import { api } from "@/api";

const { userId, user, isTelegram } = useTelegram();

const nextFact = () => {

const currentIndex = funFacts.indexOf(currentFact.value);

const nextIndex = (currentIndex + 1) % funFacts.length;

currentFact.value = funFacts[nextIndex];

};

const catClapCount = ref(0);

const catCoinCount = ref(0);

const selectedCat = ref(null);

const userCats = ref({}); // Объект вида { userId: catData }

const activeFilter = ref("claps"); // 'claps' или 'coins' по умолчанию

const players = ref([]); // Начинаем с пустого массива, данные загрузятся в onMounted

// Функция загрузки выбранного кота текущего пользователя

const loadSelectedCat = () => {

try {

const savedSelectedCat = localStorage.getItem("selectedCat");

if (savedSelectedCat) {

selectedCat.value = JSON.parse(savedSelectedCat);

console.log("Загружен выбранный кот для аватара:", selectedCat.value);

if (userId.value && selectedCat.value) {

userCats.value[userId.value] = selectedCat.value;

}

}

} catch (error) {

console.error("Ошибка загрузки выбранного кота:", error);

}

};

const loadAllUserCats = async () => {

try {

const response = await api.getAllUserCats();

if (response && Array.isArray(response)) {

const catsMap = {};

response.forEach((item) => {

// Проверяем валидность записи

if (typeof item !== "object" || !item.user\_id || !item.selected\_cat) {

console.warn(

item

);

return;

}

// Нормализуем ID пользователя

let normalizedUserId = item.user\_id;

if (typeof normalizedUserId === "object") {

if (normalizedUserId.$oid) normalizedUserId = normalizedUserId.$oid;

else if (normalizedUserId.\_id)

normalizedUserId = normalizedUserId.\_id;

}

normalizedUserId = String(normalizedUserId);

// Проверка на валидный ID (не "selected-cats")

if (normalizedUserId === "selected-cats") {

console.warn(

normalizedUserId

);

return;

}

catsMap[normalizedUserId] = {

id: item.selected\_cat.cat\_id,

title: item.selected\_cat.name,

images: item.selected\_cat.images || {

normal:

item.selected\_cat.images?.normal ||

"/cat\_avatars/normal/default.jpg",

happy: "/cat\_avatars/happy/default.jpg",

sad: "/cat\_avatars/sad/default.jpg",

},

color: item.selected\_cat.color,

};

});

// Обновляем состояние

userCats.value = { ...userCats.value, ...catsMap };

console.log(

"Загружены коты всех пользователей:",

Object.keys(userCats.value).length

);

}

} catch (error) {

console.error("Ошибка при загрузке котов пользователей:", error);

}

};

const getPlayerAvatar = (player) => {

if (!player || !player.id) return "/avatars/default.png";

let playerId = String(player.id);

if (playerId && userCats.value[playerId] && userCats.value[playerId].images) {

const catImages = userCats.value[playerId].images;

return catImages.normal || catImages.happy || player.avatar;

}

return player.avatar;

};

const isCurrentUser = (player) => {

return player.id === userId.value;

};

// Фильтрация и сортировка игроков

const filteredPlayers = computed(() => {

let list = [...players.value];

list = list.filter(

(player) =>

player.id !== "selected-cats" &&

typeof player.id !== "undefined" &&

player.id !== null

);

return list.sort((a, b) => {

if (activeFilter.value === "claps") {

return (b.claps || 0) - (a.claps || 0);

} else {

return (b.coins || 0) - (a.coins || 0);

}

});

});

const topPlayers = computed(() => filteredPlayers.value.slice(0, 3));

const remainingPlayers = computed(() => filteredPlayers.value.slice(3));

onMounted(async () => {

console.log("Current User ID:", userId.value);

console.log("Telegram User Data:", user.value);

console.log("Is Telegram App:", isTelegram.value);

loadSelectedCat();

await loadAllUserCats();

try {

// 1. Загрузка данных текущего пользователя (монеты, хлопки)

if (userId.value) {

const userData = await api.getUser(userId.value);

if (userData) {

catCoinCount.value = userData.coins\_count || 0;

catClapCount.value = userData.claps\_count || 0;

} else {

console.warn(`Top.vue: Пользователь с ID ${userId.value} не найден.`);

catCoinCount.value = 0;

catClapCount.value = 0;

}

} else {

// Обработка случая без userId (например, веб-версия)

catCoinCount.value = 0;

catClapCount.value = 0;

}

const leaderboardData = await api.getTopUsers();

const validPlayers = (leaderboardData || [])

.filter((p) => {

// Фильтруем записи с некорректными ID

if (!p.user\_id || p.user\_id === "selected-cats") {

console.warn("Обнаружен некорректный ID пользователя:", p.user\_id);

return false;

}

return true;

})

.map((p) => ({

id: p.user\_id, // или p.id

name: p.username || "Игрок", // или p.name

avatar: p.avatar\_url || "/avatars/default.png", // URL аватара или дефолтный

claps: p.claps\_count || 0,

coins: p.coins\_count || 0,

}));

console.log(`Загружено ${validPlayers.length} игроков для таблицы лидеров`);

players.value = validPlayers;

} catch (error) {

console.error("Ошибка загрузки данных в Top.vue:", error);

players.value = []; // Очистить список при ошибке

// Fallback на localStorage, если необходимо

const savedData = localStorage.getItem(

`catAppData\_${userId.value || "web\_fallback"}`

);

if (savedData) {

const parsed = JSON.parse(savedData);

catCoinCount.value = parsed.catCoinCount || 0;

catClapCount.value = parsed.catClapCount || 0;

} else {

catCoinCount.value = 0;

catClapCount.value = 0;

}

}

currentFact.value = funFacts[Math.floor(Math.random() \* funFacts.length)];

});

const funFacts = [

"Средний кот проводит 2/3 своей жизни во сне. Это почти 16 часов в день!",

"Кошки могут издавать около 100 различных звуков. Собаки - только около 10.",

"Самая старая кошка в мире дожила до 38 лет!",

"У кошек есть уникальный 'отпечаток носа', похожий на отпечаток пальца человека.",

"Мурлыканье кошки не всегда означает счастье; они также мурлычут, когда болеют или напуганы.",

];

const currentFact = ref(funFacts[0]);

const confettiStyle = (index) => {

const colors = ["#ff69b4", "#ffb6c1", "#ffd700", "#add8e6", "#90ee90"];

const size = Math.random() \* 8 + 4;

const left = Math.random() \* 100;

const delay = Math.random() \* 10;

const duration = Math.random() \* 5 + 5;

const color = colors[Math.floor(Math.random() \* colors.length)];

return {

left: `${left}%`,

width: `${size}px`,

height: `${size}px`,

backgroundColor: color,

animationDelay: `${delay}s`,

animationDuration: `${duration}s`,

};

};

const formatNumber = (num) => {

if (num >= 1000000) return (num / 1000000).toFixed(1) + "M";

if (num >= 1000) return (num / 1000).toFixed(1) + "K";

return num || 0; // Возвращаем 0, если num не определено

};

</script>

<style scoped>

.app-container {

min-height: 100vh;

background: linear-gradient(135deg, #fff0f5 0%, #ffe6f0 100%);

font-family: "Comic Neue", "Poppins", cursive, sans-serif;

}

.top-container {

max-width: 1200px;

margin: 0 auto;

padding: 20px;

position: relative;

overflow: hidden;

}

/\* Конфетти \*/

.confetti-container {

position: absolute;

width: 100%;

height: 100%;

top: 0;

left: 0;

pointer-events: none;

z-index: 1;

}

.confetti {

position: absolute;

top: -10px;

border-radius: 50%;

opacity: 0.7;

animation: confettiFall 10s linear infinite;

}

@keyframes confettiFall {

0% {

transform: translateY(-10px) rotate(0deg);

opacity: 0.7;

}

100% {

transform: translateY(calc(100vh)) rotate(360deg);

opacity: 0;

}

}

.top-header {

display: flex;

flex-direction: column;

align-items: center;

margin-bottom: 40px;

position: relative;

z-index: 2;

}

.title-wrapper {

position: relative;

margin-bottom: 15px;

}

.top-title {

font-size: 3.2rem;

background: linear-gradient(135deg, #ff36b3, #ff74bf, #ff36b3);

-webkit-background-clip: text;

background-clip: text;

-webkit-text-fill-color: transparent;

background-size: 200% auto;

text-align: center;

margin-bottom: 5px;

font-weight: 800;

text-shadow: 0 3px 6px rgba(255, 105, 180, 0.2);

animation: shimmer 3s ease-in-out infinite;

letter-spacing: 1px;

}

@keyframes shimmer {

0% {

background-position: 0% 50%;

}

50% {

background-position: 100% 50%;

}

100% {

background-position: 0% 50%;

}

}

.title-underline {

height: 5px;

width: 60%;

margin: 0 auto;

background: linear-gradient(to right, transparent, #ff69b4, transparent);

border-radius: 10px;

animation: pulseWidth 3s ease-in-out infinite;

}

@keyframes pulseWidth {

0%, 100% {

width: 60%;

opacity: 0.7;

}

50% {

width: 80%;

opacity: 1;

}

}

.top-subtitle {

color: #888;

font-size: 1.1rem;

margin-bottom: 20px;

font-style: italic;

}

.filter-box {

display: flex;

gap: 15px;

margin-top: 10px;

}

.filter-btn {

padding: 12px 25px;

border-radius: 30px;

border: 2px solid #ffb6c1;

background: white;

color: #666;

cursor: pointer;

transition: all 0.3s;

font-size: 1.1rem;

font-weight: 600;

}

.filter-btn:hover {

transform: translateY(-3px);

box-shadow: 0 6px 10px rgba(255, 105, 180, 0.25);

}

.filter-btn.active {

background: linear-gradient(to right, #ff69b4, #ff8fab);

color: white;

border-color: transparent;

box-shadow: 0 6px 12px rgba(255, 105, 180, 0.4);

}

.top-content {

position: relative;

z-index: 2;

}

.top-players {

display: flex;

justify-content: center;

align-items: flex-end;

margin-bottom: 50px;

height: 350px;

padding: 0 40px;

}

.player-card {

position: relative;

display: flex;

flex-direction: column;

align-items: center;

background: white;

border-radius: 20px;

padding: 25px;

box-shadow: 0 15px 30px rgba(0, 0, 0, 0.1);

transition: all 0.4s cubic-bezier(0.175, 0.885, 0.32, 1.275);

overflow: visible;

}

.player-card:hover {

transform: translateY(-15px);

box-shadow: 0 20px 40px rgba(255, 105, 180, 0.2);

}

.first-place {

height: 350px;

width: 250px;

z-index: 3;

border: 3px solid gold;

box-shadow: 0 15px 40px rgba(255, 215, 0, 0.4);

margin: 0 -15px;

background: linear-gradient(to bottom, #fffaf0, white);

}

.second-place,

.third-place {

height: 300px;

width: 200px;

z-index: 2;

}

.second-place {

border: 3px solid silver;

background: linear-gradient(to bottom, #f8f8f8, white);

}

.third-place {

border: 3px solid #cd7f32;

background: linear-gradient(to bottom, #fff8f0, white);

}

.ribbon {

position: absolute;

top: 25px;

right: -30px;

width: 100px;

height: 30px;

transform: rotate(45deg);

z-index: 5;

}

.medal {

position: absolute;

top: -20px;

left: calc(50% - 20px);

width: 40px;

height: 40px;

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

font-weight: bold;

color: white;

font-size: 1.2rem;

z-index: 10;

}

.gold {

background: linear-gradient(135deg, #ffd700, #ffcc00);

box-shadow: 0 4px 10px rgba(255, 215, 0, 0.5);

}

.silver {

background: linear-gradient(135deg, #c0c0c0, #e0e0e0);

box-shadow: 0 4px 10px rgba(192, 192, 192, 0.5);

}

.bronze {

background: linear-gradient(135deg, #cd7f32, #dda15e);

box-shadow: 0 4px 10px rgba(205, 127, 50, 0.5);

}

.crown {

position: absolute;

top: -40px;

font-size: 2.5rem;

text-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);

animation: float 3s ease-in-out infinite;

z-index: 10;

}

/\* Звездочки вокруг победителя \*/

.sparkle-container {

position: absolute;

width: 100%;

height: 100%;

pointer-events: none;

}

.sparkle {

position: absolute;

font-size: 1.2rem;

animation: twinkle 3s ease-in-out infinite;

}

.sparkle:nth-child(1) {

top: 20%;

left: 10%;

animation-delay: 0.5s;

}

.sparkle:nth-child(2) {

top: 15%;

right: 10%;

animation-delay: 1s;

}

.sparkle:nth-child(3) {

bottom: 30%;

left: 5%;

animation-delay: 1.5s;

}

.sparkle:nth-child(4) {

bottom: 30%;

right: 5%;

animation-delay: 2s;

}

.sparkle:nth-child(5) {

top: 50%;

right: 8%;

animation-delay: 2.5s;

}

@keyframes twinkle {

0%,

100% {

transform: scale(1);

opacity: 1;

}

50% {

transform: scale(1.3);

opacity: 0.8;

}

}

@keyframes float {

0%,

100% {

transform: translateY(0) rotate(-5deg);

}

50% {

transform: translateY(-12px) rotate(5deg);

}

}

.player-avatar {

width: 110px;

height: 110px;

border-radius: 50%;

object-fit: cover;

border: 4px solid white;

box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15);

transition: transform 0.4s ease;

}

.player-card:hover .player-avatar {

transform: scale(1.05);

}

.first-place .player-avatar {

width: 130px;

height: 130px;

border: 5px solid gold;

}

.second-place .player-avatar {

border: 4px solid silver;

}

.third-place .player-avatar {

border: 4px solid #cd7f32;

}

.player-name {

margin: 20px 0 8px;

font-size: 1.3rem;

text-align: center;

color: #333;

font-weight: 600;

}

.first-place .player-name {

font-size: 1.6rem;

font-weight: 700;

background: linear-gradient(to right, #ff69b4, #ff8fab);

-webkit-background-clip: text;

background-clip: text;

-webkit-text-fill-color: transparent;

}

.player-score {

font-size: 1.2rem;

font-weight: bold;

color: #ff69b4;

padding: 5px 15px;

border-radius: 20px;

background: #fff5f9;

border: 1px dashed #ffb6c1;

}

.first-place .player-score {

font-size: 1.4rem;

background: #fffaf0;

border: 1px dashed gold;

}

.players-list {

background: white;

border-radius: 25px;

box-shadow: 0 15px 30px rgba(0, 0, 0, 0.08);

overflow: hidden;

border: 1px solid #ffd6e7;

margin-bottom: 40px;

}

.list-item {

display: flex;

align-items: center;

padding: 15px 25px;

border-bottom: 1px solid #f0f0f0;

transition: all 0.3s;

}

.list-item:hover {

background: #fff5f9;

transform: translateY(-3px);

box-shadow: 0 5px 15px rgba(255, 105, 180, 0.1);

z-index: 5;

position: relative;

}

.list-rank {

width: 35px;

height: 35px;

background: #f0f0f0;

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

font-weight: bold;

color: #666;

margin-right: 20px;

transition: all 0.3s;

}

.list-item:hover .list-rank {

background: #ffb6c1;

color: white;

transform: scale(1.1);

}

.list-avatar {

width: 45px;

height: 45px;

border-radius: 50%;

object-fit: cover;

margin-right: 20px;

border: 3px solid #fff5f9;

transition: all 0.3s;

и т.д.

</style>

Приложение Б

Руководство пользователя

Добро пожаловать в CatClap - приложение в Telegram, которое позволяет вам взаимодействовать с виртуальными питомцами, покупать внутриигровые улучшения и участвовать в таблице лидеров. Это руководство поможет вам освоить основные функции приложения и начать пользоваться им.

Начало работы

Для запуска приложения откройте Telegram и перейдите к приложению CatClap. Нажмите кнопку «Начать», чтобы запустить приложение. При первом входе в приложение вы автоматически регистрируетесь, и вам будет предоставлен базовый котик с уникальным окрасом и настроением.

Взаимодействие с котиком

В разделе «Home» вы можете увидеть своего питомца, его текущий окрас и настроение. Для взаимодействия с котиком используйте кнопки, такие как «Погладить», «Покормить» или «Поиграть». Каждое действие изменяет настроение котика, которое отображается в реальном времени. Например, после кормления котик становится счастливым. Также воспроизводятся звуки, соответствующие настроению питомца.

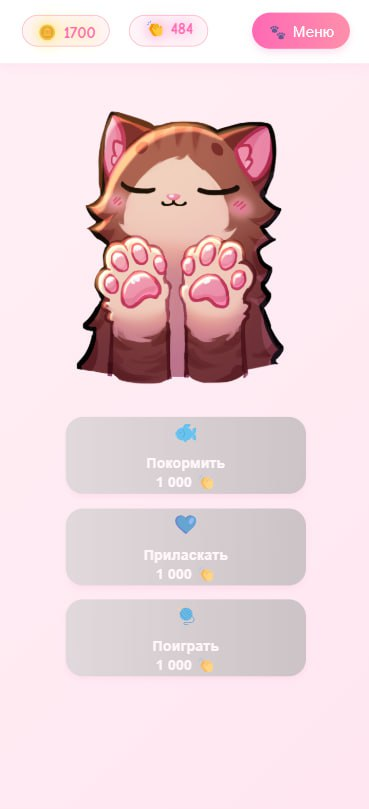


Рисунок 27 – Главный экран

Покупка новых окрасов

В разделе «CatStore» вы можете приобрести новые окрасы для своего питомца. Выберите понравившийся окрас из доступных вариантов и нажмите кнопку «Купить». Покупка осуществляется за внутриигровую валюту CatCoin или CatClap.

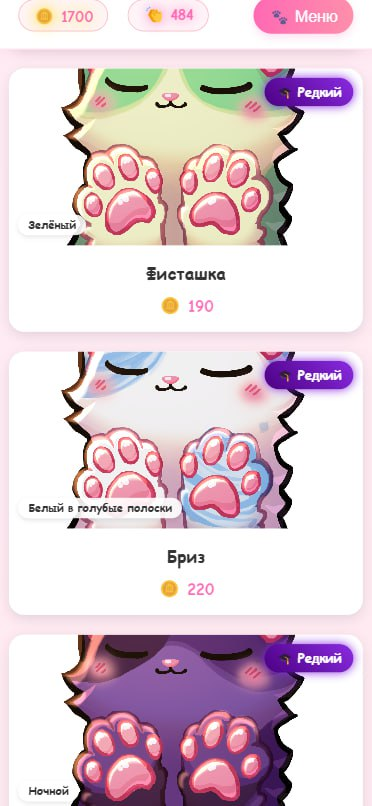


Рисунок 28 – Раздел покупки окрасов

Управление друзьями

В разделе «Friend» вы можете добавлять новых друзей, удалять их и просматривать список уже добавленных. Чтобы добавить друга, скиньте ему ссылку-приглашение, нажав на кнопку «Пригласить». После успешного добавления друг появится в вашем списке.

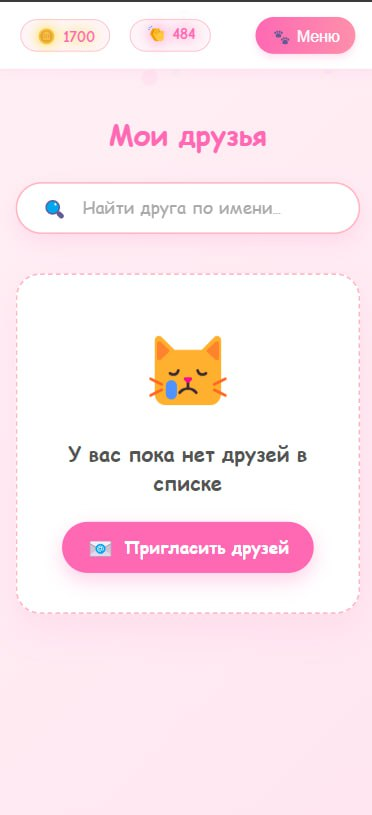


Рисунок 29 – Раздел добавления друзей

Таблица лидеров

В разделе «Leaderboard» отображаются топ-10 пользователей по количеству CatCoin и CatClap. Таблица обновляется в реальном времени, что позволяет вам следить за своим прогрессом и сравнивать свои достижения с другими пользователями.



Рисунок 30 – Раздел лидеров

Покупка внутриигровой валюты

Для пополнения баланса внутриигровой валюты перейдите в раздел «Баланс». Выберите количество CatCoin или CatClap, которое хотите приобрести, и подтвердите оплату через встроенную платежную систему Telegram. После успешной оплаты ваш баланс обновится, и вы сможете использовать валюту для покупок в магазине.



Рисунок 31 – Раздел покупки валюты

Техническая поддержка

Если у вас возникли вопросы или проблемы с приложением, вы можете обратиться в техническую поддержку через Telegram.