

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc199959365)

[1 Постановка задачи 7](#_Toc199959366)

[1.1 Цели создания приложения 7](#_Toc199959367)

[1.2 Задачи приложения 7](#_Toc199959368)

[1.3 Требования к приложению 7](#_Toc199959369)

[1.3.1 Требования к приложению в целом 7](#_Toc199959370)

[1.3.2 Требования к функциям (задачам), выполняемым приложением 8](#_Toc199959371)

[1.3.3 Требования к оформлению и верстке страниц 10](#_Toc199959372)

[1.3.4 Требования к защите информации 10](#_Toc199959373)

[1.4 Задачи, решаемые в процессе разработки 10](#_Toc199959374)

[2 Анализ предметной области 12](#_Toc199959375)

[2.1 Глоссарий 12](#_Toc199959376)

[2.2 Обзор аналогов 13](#_Toc199959377)

[2.2.1 Steam Community Market 13](#_Toc199959378)

[2.2.2 Cardmarket 15](#_Toc199959379)

[2.2.3 Case Battle 17](#_Toc199959380)

[2.2.4 Pokemon TCG 18](#_Toc199959381)

[2.3 Сравнительная таблица аналогов 20](#_Toc199959382)

[3 Моделирование системы 22](#_Toc199959383)

[3.1 Диаграмма прецедентов (Use Case) 22](#_Toc199959384)

[3.2 Диаграмма активностей (Activity diagram) 22](#_Toc199959385)

[3.3 ER-диаграмма 22](#_Toc199959386)

[3.4 Диаграмма классов (Class diagram) 22](#_Toc199959387)

[3.5 Диаграмма развертывания (Deployment diagram) 23](#_Toc199959388)

[3.6 Диаграмма последовательности (Sequence diagram) 23](#_Toc199959389)

[4 Реализация 24](#_Toc199959390)

[4.1 Средства реализации 24](#_Toc199959391)

[4.2 Архитектура клиентской части 25](#_Toc199959392)

[4.2.1 Компонент представления (View) 26](#_Toc199959393)

[4.2.2 Компонент данных (Model) 27](#_Toc199959394)

[4.2.3 Компонент модели представления (ViewModel) 28](#_Toc199959395)

[4.3 Архитектура серверной части 29](#_Toc199959396)

[4.3.1 Слой контроллеров (Controller) 29](#_Toc199959397)

[4.3.2 Слой сервисов (Service) 30](#_Toc199959398)

[4.3.3 Слой моделей (Model) 30](#_Toc199959399)

[4.3.4 Слой репозиториев (Repository) 31](#_Toc199959400)

[4.3.5 Слой DTO (Data Transfer Object) 31](#_Toc199959401)

[4.3.6 Слой конфигурации и безопасности (Config, Security) 32](#_Toc199959402)

[4.4 Реализация интерфейса 32](#_Toc199959403)

[4.4.1 Экран загрузки 32](#_Toc199959404)

[4.4.2 Экран авторизации/регистрации 33](#_Toc199959405)

[4.4.3 Экран «Главная» 35](#_Toc199959406)

[4.4.4 Экран «создания карточки» 37](#_Toc199959407)

[4.4.5 Экран «Инвентарь» 37](#_Toc199959408)

[4.4.6 Экран «Магазин» 38](#_Toc199959409)

[4.4.7 Экран «Обменник» 39](#_Toc199959410)

[4.4.8 Экран «Профиль» 42](#_Toc199959411)

[4.4.9 Экран «Квесты» 42](#_Toc199959412)

[4.4.10 Экран «Новости» 43](#_Toc199959413)

[4.4.11 Экран «Принятия обмена» 45](#_Toc199959414)

[4.4.12 Экран «Создания обмена» 47](#_Toc199959415)

[5 Тестирование 49](#_Toc199959416)

[6 Аналитика 52](#_Toc199959417)

[6.1 Преимущества AppMetrica 52](#_Toc199959418)

[6.2 Воронки 52](#_Toc199959419)

[6.3 Статистика 53](#_Toc199959420)

[ЗАКЛЮЧЕНИЕ 55](#_Toc199959421)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 56](#_Toc199959422)

[ПРИЛОЖЕНИЕ А 57](#_Toc199959423)

[ПРИЛОЖЕНИЕ Б 57](#_Toc199959424)

[ПРИЛОЖЕНИЕ В 59](#_Toc199959425)

[ПРИЛОЖЕНИЕ Г 59](#_Toc199959426)

[ПРИЛОЖЕНИЕ Д 68](#_Toc199959427)

[ПРИЛОЖЕНИЕ Е 69](#_Toc199959428)

# ВВЕДЕНИЕ

Коллекционирование является одним из популярных видов досуга, объединяющим людей всех возрастов и интересов. Среди коллекционных предметов особое место занимают карточки, которые привлекают внимание своей уникальной эстетикой, разнообразием тематик и возможностью обмена. С ростом технологического прогресса коллекционирование и обмен картами всё чаще переходят в цифровую сферу, что открывает широкие возможности для упрощения коммуникации между коллекционерами. Однако на данный момент многие платформы для обмена картами либо не предоставляют полного набора необходимых инструментов, либо имеют сложную структуру, что затрудняет поиск, обмен и управление коллекциями.

Текущие решения не всегда отвечают потребностям пользователей в простой и автоматизированной системе, которая бы не только упростила обмен карточками, но и предоставляла возможности для создания уникальных экземпляров, организации персональных коллекций и мониторинга актуальных предложений. Отсутствие упорядоченных механизмов для оперативного и безопасного обмена информацией, а также интеграции передовых технологий, например, искусственного интеллекта, создает барьеры для пользователей и организаторов подобных платформ. В связи с этим, разработка мобильного приложения, которое обеспечит комплексный набор функций в едином удобном доступе, приобретает особую актуальность.

Актуальность данной работы обусловлена необходимостью создания современной платформы, которая упростит процесс обмена коллекционными карточками, повысит удобство управления коллекциями и обеспечит автоматизацию ключевых процессов взаимодействия пользователей. Приложение «Cardly» призвано решить эти задачи, предоставляя пользователям интуитивно понятный инструмент для поиска, создания и обмена карточками, а также для формирования уникального пользовательского опыта.

1. Постановка задачи
   1. Цели создания приложения

Целями создания приложения являются:

* Разработка платформы для обмена коллекционными карточками между пользователями, упрощающей процесс поиска, обмена и заключения сделок.
* Предоставление возможности пополнять свою коллекцию путем создания коллекционных карточек с уникальным дизайном и тематикой.
* Реализация функционала для формирования и управления персональными коллекциями карт, включая возможность отслеживания недостающих экземпляров.
  1. Задачи приложения

Приложение позволяет решать следующие задачи:

* Просматривать наборы коллекционных карточек и взаимодействовать с ними.
* Просматривать профили пользователей и совершать с ними обмен;
* Совершать быстрые и обычные обмены.
* Коллекционировать карточки.
* Создавать уникальные карточки с помощью ИИ.
* Выполнять квесты.
* Осуществлять редактирование данных своего аккаунта после регистрации или авторизации.
  1. Требования к приложению
     1. Требования к приложению в целом

Данное приложение должно удовлетворять следующим требованиям:

* Приложение должно корректно работать на устройствах, работающих на операционной системе Android 8.0 и новее.
* Реализовывать все поставленные задачи.
  + 1. Требования к функциям (задачам), выполняемым приложением

Функциональные требования (functional requirements) определяют, каким должно быть поведение продукта в тех или иных условиях. Они определяют, что разработчики должны создать, чтобы пользователи смогли выполнить свои задачи (пользовательские требования) в рамках бизнес- требований. Функциональные требования описываются в форме традиционных утверждений со словами «должен» или «должна» [1].

Разрабатываемое приложение должно соответствовать следующим функциональным требованиям:

Неавторизованный пользователь должен обладать возможностью:

* Авторизация и регистрация в приложении, включая восстановление пароля.
* Поиск и просмотр профилей других авторизованных пользователей.
* Просмотр пользовательского инвентаря, включая отдельные полученные карты.
* Просмотр содержимого магазина.
* Ознакомление с актуальными новостями и обновлениями.
* Просмотр коллекций карт.
* Просмотр активных обменов от авторизованных пользователей.

Авторизованный пользователь должен обладать возможностью:

* Использование всех функций, доступных неавторизованному пользователю.
* Создание и отклик на обмены, включая быстрый и обычный формат, а также выставление карт на обмен.
* Отправка жалоб.
* Создание уникальных карточек с помощью нейронной сети.
* Просмотр списка доступных квестов, отслеживание их выполнения и получение наград.
* Работа с коллекцией карт: сортировка, просмотр каждой карты, добавление в избранное, разбор на монеты.
* Покупка наборов карточек за монеты.
* Пополнение баланса монет через донат.
* Настройка профиля, включая смену аватара.
* Просмотр списка избранных карт и достижений.
* Просмотр статистики по полученным картам и собранным коллекциям.
* Доступ к настройкам: управление уведомлениями, отображением инвентаря для других пользователей и автоотклонением входящих обменов.

Администратор должен обладать возможностью:

* Блокировка и разблокировка пользователей.
* Доступ к детальной информации о пользователях (инвентарь, история обменов, достижения) для принятия решений.
* Отмена достижений, квестов и обменов при нарушении правил.
* Удаление карточек из инвентаря пользователя.
* Работа с жалобами: просмотр всех заявок с указанием причин, взятие жалоб в обработку, принятие или отклонение, принятие решений о блокировке.
* Управление контентом: создание, редактирование и удаление карточек, коллекций, наборов, квестов и достижений.
* Публикация и планирование новостей.
* Просмотр и анализ метрик для оценки пользовательской активности, популярности контента, эффективности модерации и технической стабильности приложения.
  + 1. Требования к оформлению и верстке страниц

Оформление и верстка страниц должны удовлетворять следующим требованиям:

* Приложение должно быть оформлено в едином стиле.
* Приложение должно быть разработано в одной цветовой палитре с использованием ограниченного набора шрифтов.
* Цветовая палитра должна быть контрастной.
* Необходимо корректное и одинаковое отображение страниц на экранах различного размера с диагональю от 5.5" до 6.9".
  + 1. Требования к защите информации

Для аутентификации пользователей необходимо использовать JSON web token, обеспечивающий компактный и защищенный контейнер для данных. Даже если злоумышленник получит этот токен, то через заданное количество времени (от 2 до 10 минут) этот токен станет недействительным.

* 1. Задачи, решаемые в процессе разработки

Были поставлены следующие задачи:

* Анализ предметной области.
* Обзор аналогов.
* Постановка задачи.
* Создание репозитория GitHub и доски в таск-трекере Jira.
* Разработка требований: к приложению в общем, к функциям, к структуре, к программному обеспечению, к оформлению и верстке страниц, к защите информации.
* Создание диаграмм: use case, активностей, последовательностей, ER, классов.
* Разработка дизайна приложения.
* Написание технического задания в соответствии с ГОСТ 34.602 – 2020.
* Реализация интерфейса приложения.
* Реализация серверной части приложения.
* Развертывание приложения.
* Написание курсовой работы.

1. Анализ предметной области
   1. Глоссарий

**Frontend** – Презентационная часть информационной или программной системы, ее пользовательский интерфейс и связанные с ним компоненты.

**Backend** – Логика работы сайта, внутренняя часть продукта, которая находится на сервере и скрыта от пользователя.

**GitHub** – Веб-сервис для хостинга IT-проектов и их совместной разработки.

**PostgreSQL** – Реляционная база данных с открытым кодом.

**Фреймворк** – Программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

**API** – список способов и правил, по которым различные программы общаются между собой и обмениваются данными.

**Квест** – небольшое задание после выполнения которого пользователь получает награду в виде монет или набора.

**Набор** – комплект, состоящий из нескольких карточек. Приобретается в магазине за местную валюту или выдается после выполнения квеста. После открытия набора из него выпадает определенное количество карт, которые помещаются в инвентарь пользователя.

**Избранные карты** – карты, которые пользователь пометил особым знаком звездочки, после чего они отражаются на витрине его профиля.

**Событие** – важное действие или изменение в приложении, требующее внимания пользователя.

**Монеты** – местная валюта, за которую пользователь может покупать наборы карточек, создавать уникальные карточки с помощью искусственного интеллекта на выбранную тематику.

**Обменник** – страница, на которой отображен весь список активных обменов пользователей.

**Активный обмен** – статус обмена, который говорит о том, что обмен находится в режиме ожидания ответа от одного из пользователей.

**Инвентарь** – страница, на которой отображаются все карты, имеющиеся у данного пользователя.

**Карта** – является предметом обмена в сервисе. Состоит из названия, изображения и текстового описания. Каждая карта обладает редкостью.

**Редкость** – параметр, определяющий ценность карты и вероятность ее получения. Карта может обладать одним из пяти типов редкости: обычная, редкая, эпическая, легендарная, уникальная. (Тип редкости «уникальная» устанавливается только для карт, сгенерированных при помощи ИИ)

**Коллекция** – набор карт, объединенных общей тематикой.

**Достижение** – параметр, отражающий прогресс пользователя. Состоит из названия, изображения и описания (в описании указывается условие для получения достижения). Достижение может быть «Получено» или «Не получено».

**Id** – уникальный идентификатор пользователя.

**ИИ** – Искусственный интеллект.

* 1. Обзор аналогов

После исследования рынка было выявлено, что платформы, предназначенной исключительно для коллекционирования и обмена карточками, не существует в российском сегменте рынка. И в ходе работы будут рассматриваться зарубежные сервисы по схожей тематике.

Ниже представлен анализ каждого из конкурентов.

* + 1. Steam Community Market

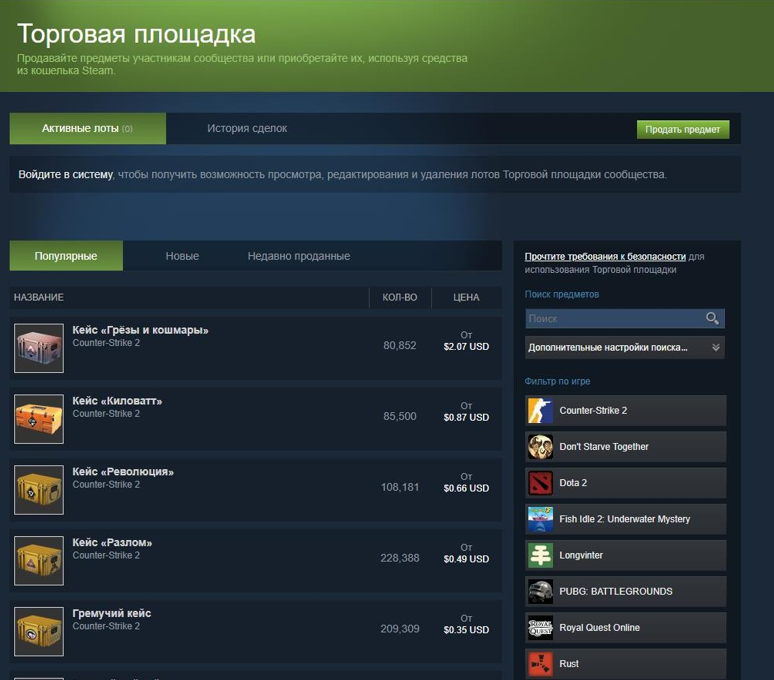
Это виртуальная торговая платформа, разработанная компанией Valve, интегрированная в экосистему Steam. Она позволяет пользователям покупать и продавать внутриигровые предметы, такие как скины, оружие, коллекционные карточки, фоны, эмодзи и другие цифровые товары, связанные с играми на платформе Steam, за средства Steam Wallet. Платформа была запущена в мае 2013 года вместе с введением коллекционных карточек Steam и изначально поддерживала игру Team Fortress 2, а позже была расширена для других игр.

Преимущества:

* Интеграция с экосистемой Steam. Покупка, продажа и обмен карт осуществляется прямо в клиенте Steam, без необходимости использовать сторонние сайты.
* Автоматизированные и мгновенные транзакции. Все покупки и продажи проходят моментально, без ручного подтверждения другой стороной. Это снижает риск мошенничества.
* История цен и графики. У каждого предмета есть история цен и график спроса/предложения, что помогает принимать обоснованные решения при покупке или продаже.

Недостатки:

* Привязка к внутренней валюте Steam. Все сделки проходят в Steam-валюте. Средства, полученные с продаж, нельзя вывести на реальный счёт — их можно тратить только в Steam.
* Комиссии Valve. С каждой продажи удерживается комиссия (обычно 15%), что снижает доход с предметов.
* Ограничения по странам и валютам. Некоторые предметы недоступны или имеют разные цены в зависимости от региона, что затрудняет глобальный трейд.
* Нет персонализированных коллекционных возможностей. Хотя можно собирать значки и эмблемы, платформа не ориентирована на глубокую коллекционную механику: нет галерей, аукционов, редких предметов вне маркетплейса.



1. Steam Community Market
   * 1. Cardmarket

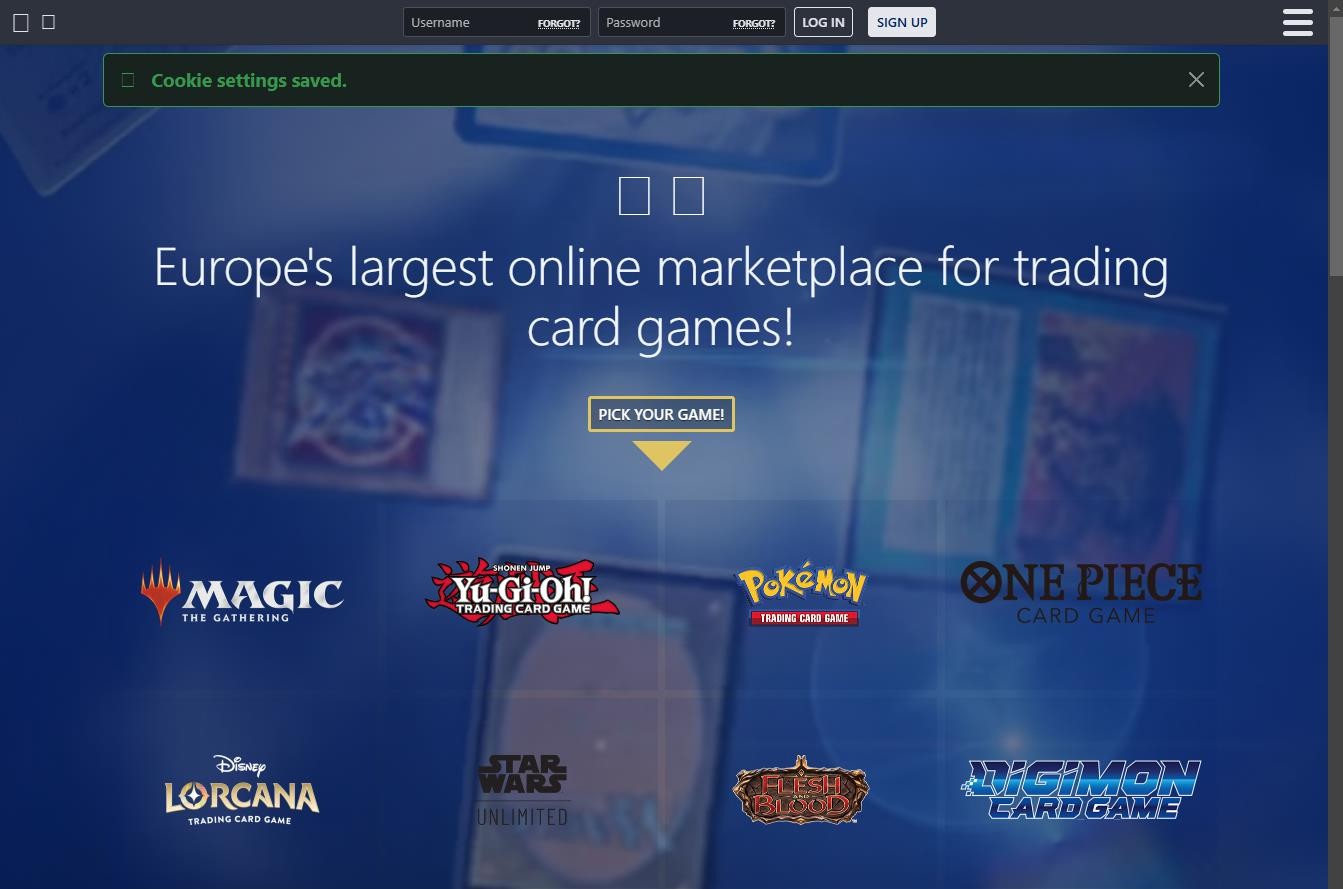
Ведущая онлайн-площадка, в первую очередь ориентированная на торговлю карточными играми, с сильным акцентом на Pokémon, Magic: The Gathering, Yu-Gi-Oh! и другие коллекционные карточные игры. Расположенная в Европе, она служит платформой, на которой пользователи могут покупать и продавать отдельные карты, бустеры, запечатанные продукты и аксессуары. Пользователи могут выставлять свои товары на продажу, устанавливать цены и участвовать в прямых транзакциях с другими коллекционерами или игроками, что делает ее площадкой одноранговой торговли.

Преимущества:

* Широкий ассортимент карточек. Платформа предлагает обширный выбор карточек различных коллекций, что позволяет пользователям находить нужные им позиции.
* Надежные продавцы. Многие пользователи отмечают высокую надежность продавцов на платформе, что способствует безопасным сделкам.
* Конкурентные цены. Цены на карточки зачастую являются конкурентоспособными, что привлекает как покупателей, так и продавцов.

Недостатки:

* Ограниченная география доставки: Платформа может иметь ограничения по доставке в определенные регионы, что стоит учитывать при покупке.
* Возможные задержки в поддержке: Некоторые пользователи сообщают о задержках в ответах службы поддержки, что может вызывать неудобства при решении проблем.
* Комиссии за транзакции: Платформа может взимать комиссии за определенные операции, что следует учитывать при планировании сделок.



1. Cardmarket
   * 1. Case Battle

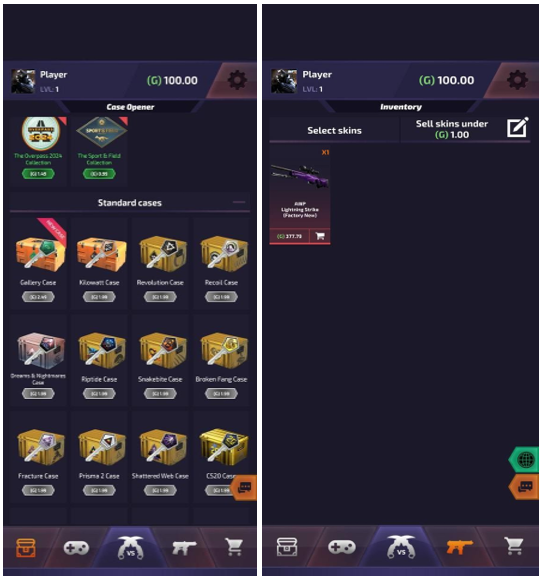
Это мобильное приложение, доступное в App Store, которое относится к категории симуляторов, ориентированных на концепцию «открытия кейсов», популярную механику в таких играх, как Counter-Strike: Global Offensive (CS:GO, теперь CS2). В Case Battle игроки участвуют в открытии виртуальных кейсов, которые представляют собой случайные ящики с добычей, содержащие игровые предметы, такие как скины оружия, ножи или другие предметы коллекционирования.

Преимущества:

* Разнообразие кейсов и режимов. Платформа предлагает широкий выбор кейсов с различными скинами, а также уникальный режим битвы, где игроки соревнуются за призы.
* Бонусы и акции. Регулярные акции, бонусы для новых и постоянных пользователей позволяют получить дополнительные возможности для открытия кейсов.

Недостатки:

* Отсутствие лицензии и правовой защищенности. Платформа работает без официальной лицензии, что может вызывать опасения по поводу правовой защищенности пользователей.
* Ограниченность тематик и ассортимента. Платформа ориентирована в основном на тематику CS:GO и аналогичных игровых предметов, что ограничивает интерес пользователей, предпочитающих коллекционирование в других игровых или культурных направлениях.



1. Case Battle
   * 1. Pokemon TCG

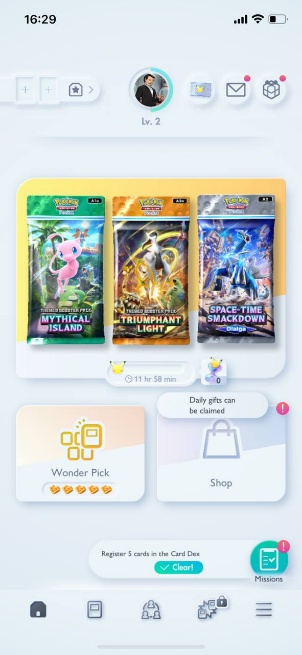
Мобильное приложение, разработанное Creatures Inc. и DeNA Co., Ltd., выпущенное для устройств iOS и Android. Запущенное в октябре 2024 года, оно предлагает цифровую версию Pokémon Trading Card Game, позволяя игрокам собирать, обмениваться и сражаться с помощью цифровых карт покемонов. Основные функции включают открытие двух бесплатных бустеров ежедневно, сбор карт с ностальгическими и эксклюзивными иллюстрациями, обмен определенными картами с друзьями и участие в случайных сражениях с упрощенными колодами из 20 карт. Приложение представляет «иммерсивные карты» с трехмерными иллюстрациями и поддерживает такие функции, как рейтинговые матчи, доски для демонстрации коллекций и механику Wonder Pick для получения карт из чужих бустеров.

Преимущества:

* Разнообразие коллекционных функций. Платформа предлагает уникальные функции, такие как "Wonder Pick", позволяющая получать случайные карты из коллекций других игроков, а также возможность настраивать внешний вид коллекции и карточек.
* Интеграция с экосистемой Pokémon. Платформа тесно связана с другими продуктами Pokémon, что обеспечивает знакомство и комфорт для поклонников франшизы.

Недостатки:

* Ограниченные возможности обмена. Система обмена картами имеет множество ограничений, включая необходимость использования специальных валют и энергии, что делает процесс обмена сложным и неэффективным.
* Отсутствие полноценной коллекционной системы. Некоторые пользователи отмечают недостаток функций для глубокого коллекционирования, таких как возможность сохранять нераспакованные бустеры или создавать подробные галереи карт.
* Ограниченность тематик. Платформа в основном сосредоточена на популярных наборах и персонажах, что может ограничивать интерес коллекционеров, ищущих более разнообразные или редкие карты.



1. Pokemon TCG
   1. **Сравнительная таблица аналогов**

С целью выявления конкурентных преимуществ и направлений дальнейшего развития проекта Cardly был проведён анализ конкурентов. В качестве критериев сравнения, отражающих реальные потребности рынка, были выбраны следующие характеристики: совершение быстрого обмена, доступ на территории РФ, широкий выбор тематик карт, мини игры, интеграция с ИИ.

Результаты сравнительного анализа Cardly и основных конкурентов (Steam Community Market, Case Battle, Cardmarket, Pokemon TCGP) представлены в таблице 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Сервис | Быстрый обмен | Доступно в России | Широкий выбор тематик карт | Интеграция с ИИ | Мини игры |
| Cardly | + | + | + | + | - |
| Cardmarket | + | - | + | - | - |
| Case Battle | - | + | - | - | + |
| Pokemon TCGP | - | - | - | - | + |
| Steam Community Market | + | + | + | - | - |

Таблица 1 – Сравнительная таблица аналогов

1. Моделирование системы
   1. Диаграмма прецедентов (Use Case)

Диаграмма прецедентов – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

На рисунках представлены диаграмма прецедентов для разрабатываемого приложения Cardly. На ней представлены 3 основные роли – неавторизованный пользователь, авторизованный пользователь и администратор (см. приложение Б).

* 1. Диаграмма активностей (Activity diagram)

Диаграмма активностей – UML-диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний.

На рисунке представлена диаграмма активностей для разрабатываемого приложения Cardly (см. приложение В).

* 1. ER-диаграмма

ER-диаграмма – это модель данных, визуализирующая связь между «сущностями» внутри системы.

На рисунке представлена ER-диаграмма для разрабатываемого приложения Cardly (см. приложение Д).

* 1. Диаграмма классов (Class diagram)

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними.

На рисунке представлена диаграмма классов для разрабатываемого приложения Cardly (см. приложение Е).

* 1. Диаграмма развертывания (Deployment diagram)

Диаграммы развертывания обычно используются для визуализации физического аппаратного и программного обеспечения системы. Она моделирует физическое развертывание артефактов на узлах.

На рисунке представлена диаграмма развертывания для разрабатываемого приложения Cardly (см. приложение А).

* 1. Диаграмма последовательности (Sequence diagram)

Диаграмма последовательности – UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента.

На рисунках представлены диаграммы последовательности для разрабатываемого приложения Cardly для авторизации, авторизованного пользователя (3 диаграммы), неавторизированного пользователя (2 диаграммы) и администратора (2 диаграммы) соответственно (см. приложение Г).

1. Реализация
   1. Средства реализации

Для реализации серверной части приложения будут использоваться следующие средства:

* Фреймворк Spring с модулем Spring Boot 3.4.4.

Выбор такого решения основан на наличии большого количества модулей, предоставляющих простой интерфейс для разработчика и позволяющих существенно сократить время разработки, а также возможностями фреймворка по работе с различными моделями взаимодействия элементов системы.

* Язык программирования Kotlin.

Одним из ключевых достоинств Kotlin является его высокая надежность, обеспечиваемая строгой статической типизацией. Это позволяет более эффективно работать со сложными структурами данных и минимизировать ошибки на этапе компиляции. компиляции.

* СУБД PostgreSQL.

Данная СУБД является свободно распространяемой и предоставляет функционал аналогичный платным конкурентам. PostgreSQL хорошо справляется с увеличением объема данных и числа пользователей, что важно для приложения, ведь со временем число пользователей будет увеличиваться. Также эта БД легко интегрируется с различными фреймворками и библиотеками.

* Docker.

Контейниризатор позволит быстрее и надежнее масштабировать приложения в рамках системы, упаковывая их в отдельные блоки.

* Инструмент для создания документации API Swagger.
* Google Drive.

Предоставляет бесплатное облачное хранилище размером 15 ГБ. Впоследствии при увеличении масштабов приложения может быть выполнен переход на Firebase cloud storage.

Также будет использована нейросеть для генерации изображений. Сгенерированные изображения могут иметь размер до 1024 пикселей с каждой из сторон. Нейросеть поддерживает запросы на русском языке. Пользователь не сможет вводить текст для запроса самостоятельно, а будет выбирать тему для генерации из списка предложенных. Этот механизм позволит реализовать защиту от генерации непристойного контента и позволит достичь высокого результата в соответствии сгенерированного изображения заявленной тематике (не менее 95%).

Для реализации клиентской части приложения будут использоваться следующие средства:

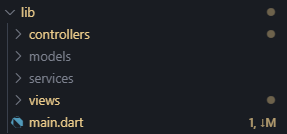
* Фреймворк Flutter version 3.29.0 on channel stable.
* Язык программирования Dart version 3.7.0.
* Android sdk 34.
* CI/CD.

Flutter обладает следующими преимуществами:

* Кроссплатформенность.
* Быстрота и лёгкость проектирования мобильных приложений.
* Понятная и полная документация.
  1. Архитектура клиентской части

Приложение реализовано в соответствии с подходом MVVM (Model – View – ViewModel).

Паттерн проектирования Model-View-ViewModel представляет собой способ организации частей приложения таким образом, чтобы классы, отвечающие за визуальное представление (View), были обособлены от классов работы с данными (Model). При этом связь между данными и представлением обеспечивается через промежуточный слой модели представления (ViewModel).

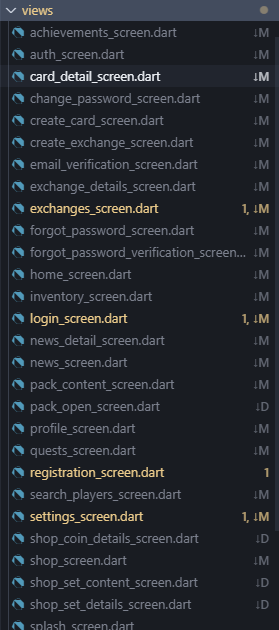


1. Реализация шаблона MVVM

Здесь файл main.dart, содержащий функцию main – точка входа в приложение. В нём производится первоначальная конфигурация приложения, устанавливается базовый URL для запросов на сервер.

* + 1. Компонент представления (View)

Слой screen содержит классы отображения визуальных элементов. Каждый экран, реализуемый в классе с постфиксом «screen» на рисунке показана структура папки.

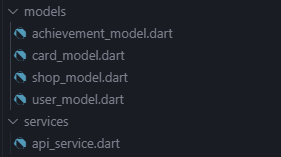


1. Реализация компонента View
   * 1. Компонент данных (Model)

Взаимодействие с данными в приложении реализуется в слоях services и models.

Слой models содержит классы, представляющие собой сущности для всех JSON-моделей, которые отправляются на сервер или приходят с сервера. Эти классы отвечают за преобразование данных из JSON в объекты и обратно, что позволяет удобно работать с данными внутри приложения.

Слой services реализует работу с внешними API и сервисами. Здесь размещаются классы, отвечающие за отправку HTTP-запросов (GET, POST, PUT, DELETE) и обработку ответов от сервера. Такой подход позволяет централизовать сетевое взаимодействие и повторно использовать сервисы в различных частях приложения.



1. Пример реализации слоя models
   * 1. Компонент модели представления (ViewModel)

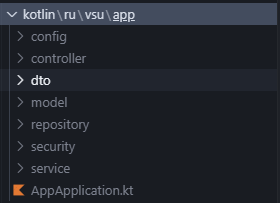
Слой controllers выполняет роль ViewModel. Здесь реализуются классы, которые управляют состоянием приложения, бизнес-логикой и взаимодействием между слоями данных и представления. Контроллеры наследуются от ChangeNotifier и используются совместно с Provider для реактивного обновления UI при изменении состояния.

В контроллерах реализуются методы для авторизации, регистрации, восстановления пароля и других бизнес-процессов. Они получают данные из сервисов, преобразуют их в нужный для отображения вид и уведомляют представления об изменениях. Такой подход обеспечивает чистое разделение ответственности и упрощает тестирование бизнес-логики.

****

1. Пример слоя controllers
   1. Архитектура серверной части

Серверная часть приложения реализована с использованием подхода многослойной архитектуры (Layered Architecture). Такой подход обеспечивает разделение ответственности между слоями. Файл AppApplication.kt содержит точку входа в приложение и отвечает за инициализацию Spring Boot-контекста, настройку компонентов и запуск сервера.



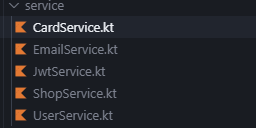
1. Пример архитектуры серверной части
   * 1. Слой контроллеров (Controller)

Пакет controller содержит классы-контроллеры, реализующие REST API приложения. Каждый контроллер отвечает за обработку HTTP-запросов, маршрутизацию и возврат ответов клиенту. Например, реализованы контроллеры для аутентификации (AuthController.kt), управления карточками (CardController.kt), работы с магазином (ShopController.kt). Контроллеры принимают и валидируют входные данные, вызывают соответствующие сервисы и формируют ответы в виде DTO.



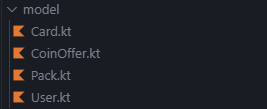
1. Пример содержания пакета controller
   * 1. Слой сервисов (Service)

Пакет service реализует бизнес-логику приложения. Здесь размещаются сервисы, отвечающие за обработку данных, выполнение основных операций, взаимодействие с репозиториями и сторонними сервисами (например, отправка email, работа с JWT). Каждый сервис инкапсулирует бизнес-правила и обеспечивает повторное использование логики в разных частяхприложения.



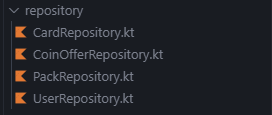
1. Пример содержания пакета service
   * 1. Слой моделей (Model)

Пакет model содержит классы-сущности, отражающие структуру данных, используемых в приложении. Эти классы соответствуют таблицам базы данных и описывают основные объекты предметной области, такие как пользователь (User.kt), карточка (Card.kt), предложение монет (CoinOffer.kt), набор (Pack.kt). Модели используются для хранения и передачи данных между слоями приложения.



1. Пример содержания пакета model
   * 1. Слой репозиториев (Repository)

Пакет repository содержит интерфейсы и классы для доступа к данным. Репозитории инкапсулируют логику взаимодействия с базой данных, предоставляя методы для поиска, сохранения, обновления и удаления сущностей. Такой подход позволяет легко изменять способ хранения данных и упрощает тестирование бизнес-логики.



1. Пример содержания пакета repository
   * 1. Слой DTO (Data Transfer Object)

Пакет dto содержит классы-объекты передачи данных, используемые для обмена информацией между клиентом и сервером, а также между слоями приложения. DTO позволяют отделить внутренние модели данных от внешних представлений, обеспечивая безопасность и гибкость API.

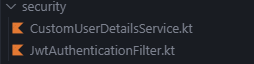


1. Пример содержания пакета DTO
   * 1. Слой конфигурации и безопасности (Config, Security)

Пакеты config и security содержат классы для настройки приложения (например, почтовый сервис, OpenAPI, безопасность). Здесь реализуются фильтры, настройки JWT-аутентификации, сервисы для работы с пользователями и другие компоненты инфраструктуры.



1. Пример содержания пакета config



1. Пример содержания пакета security
   1. Реализация интерфейса
      1. Экран загрузки

При каждом открытии мобильного приложения, пока не завершится полная инициализация клиентской части, пользователь видит загрузочный экран (см. рисунок 17).

Этот экран должен демонстрировать логотип приложения и может быть оформлен в фирменных цветах. Время отображения загрузочного экрана ограничено несколькими секундами, после чего происходит переход к основному интерфейсу приложения.



1. Экран загрузки
   * 1. Экран авторизации/регистрации

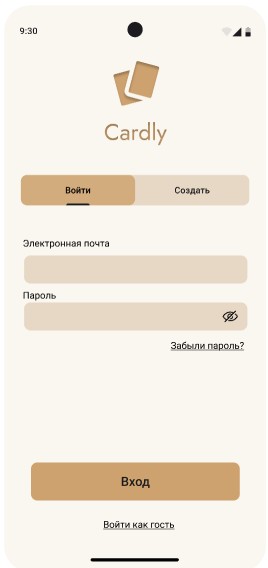
Для доступа к функциям мобильного приложения, требующим подтверждения личности, пользователи должны пройти авторизацию или регистрацию.

Вкладка «Войти» - предназначена для входа ранее зарегистрированного пользователя (см. рисунок 18). Содержит следующие элементы:

* Форма для заполнения полей личными данными: электронная почта, пароль.
* Кнопка «Вход».
* Кнопка «Забыли пароль?».
* Переключатель «Вход» - «Создать».
* Кнопка «Войти как гость».

При нажатии на кнопку «Войти как гость» пользователь может войти в приложение с ограниченным доступом к функционалу.

При успешной авторизации пользователь перенаправляется на главный экран.

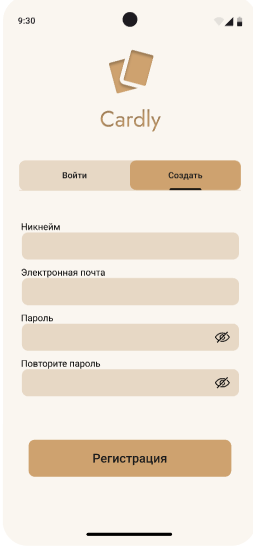
****

1. Экран входа

Вкладка «Регистрация» - предназначена для первичной регистрации новых пользователей (см. рисунок 19) содержит следующие элементы:

* форма для заполнения полей личными данными: никнейм, электронная почта, пароль, повторение пароля.
* Переключатель «Вход» - «Создать».
* Кнопка «Регистрация».
* Кнопки просмотра пароля.

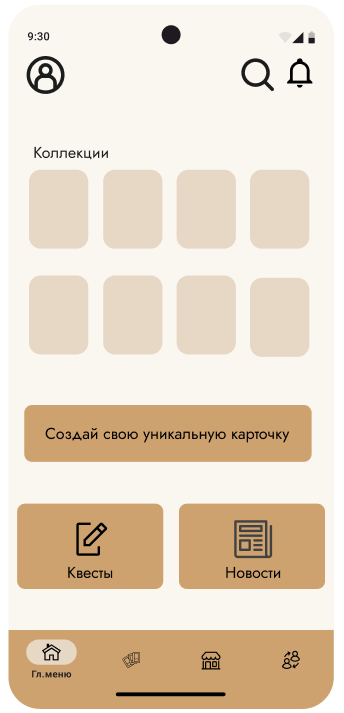
После успешного создания учетной записи, пользователь автоматически переносится на экран подтверждения электронной почты.



1. Экран регистрации
   * 1. Экран «Главная»

На этот экран пользователь попадает после «экрана загрузки», соответственно он доступен как для авторизованного пользователя, так и для неавторизованного (см. рисунок 20). Он также доступен через нижнюю панель навигации. На экране имеются следующее элементы:

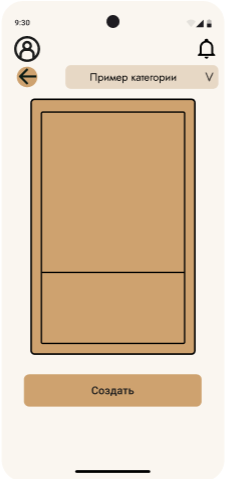
* Панель навигации, включающая в себя разделы: главное меню, карты, магазин, обменник.
* Список «Коллекции» с отображением обложек всех возможных коллекций.
* Верхняя панель, включающая в себя иконки: профиль, поиск, уведомления.
* Кнопки: создать свою уникальную карточку, квесты, новости. После нажатия на определенную кнопку пользователю открывается другой экран или новое окно с функционалом данного раздела.



1. Экран «Главная»
   * 1. Экран «создания карточки»

Экран создания уникальной карточки с помощью искусственного интеллекта (см. рисунок 21) содержит следующие элементы:

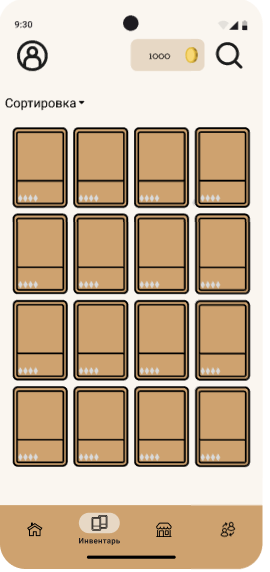
* Верхняя панель: значки «Профиль» и «Уведомления».
* Кнопка «Назад» в виде стрелочки возвращает на предыдущий экран.
* Кнопка выбора категории – раскрывает всплывающее окно выбора категории. После выбора одной из них закрывается.
* Пустое поле карточки, где после сгенерируется новая.
* Кнопка «Создать».



1. Экран создания карточки
   * 1. Экран «Инвентарь»

На экране «Инвентарь» отображаются все имеющиеся карты пользователя (см. рисунок 22), которые он может сортировать с помощью фильтров, просматривать каждую карточку отдельно, выбрав ее. Экран содержит следующие элементы:

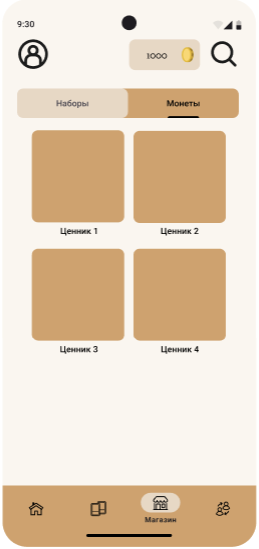
* Верхняя панель: значки «Профиль» и «Поиск».
* Кнопка «Сортировка» с выпадающим списком параметров.
* Список карт.
* Панель навигации.

****

1. Экран «Инвентарь»
   * 1. Экран «Магазин»

Экран «Магазин» (см. рисунок 23) отображает предложения покупок за монеты и реальные деньги пользователя.

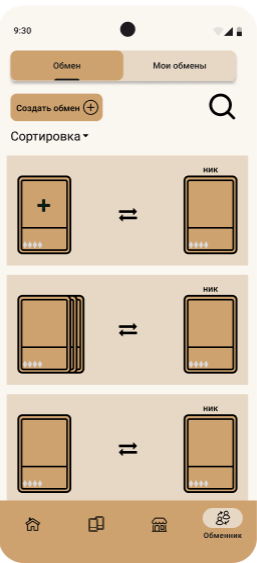
* Раздел «Наборы» представляет собой предложения покупки наборов карт за монеты.
* Раздел «Монеты» представляет собой предложения покупки местной валюты за реальные деньги.



1. Экраны «Наборы» и «Монеты» в «Магазине»
   * 1. Экран «Обменник»

Экран «Обменник» предназначен для обзора и взаимодействия с предложениями обмена от других пользователей (см. рисунок 24). Раздел «Обмен» предназначен для отображения списка всех активных обменов пользователей. Содержит элементы:

* Кнопку «Создать обмен».
* Кнопку «Сортировка» с выпадающим списком параметров.
* Кнопку «Поиск».
* Список активных обменов авторизованных пользователей.

****

1. Экран «Обменник» раздел «Обмены»

Раздел «Мои обмены» отображает список обменов, предложенных пользователю или созданных им ранее, а также их статус. Содержит элементы:

Список обменов, предложенных пользователю ранее или уже завершенных. Если обмен имеет статус «Ожидает подтверждения», то при тапе на него возможны следующие варианты:

* Отменить предложение обмена, если обмен был выставлен данным пользователем (см. рисунок 25).
* Нажать на кнопку «Принять» - для подтверждения входящего предложения обмена или нажать на кнопку «Отклонить» для отклонения входящего предложения обмена (см. рисунок 26).



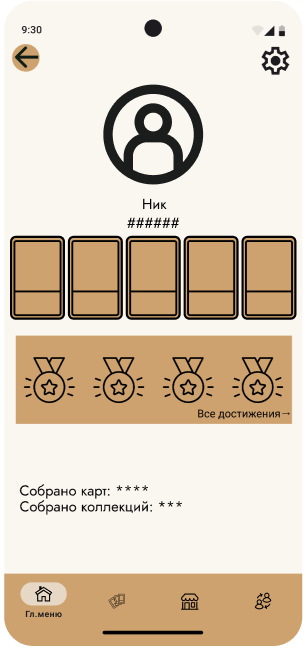
1. Отмена исходящего предложения обмена

****

1. Отклик на входящее предложение обмена
   * 1. Экран «Профиль»

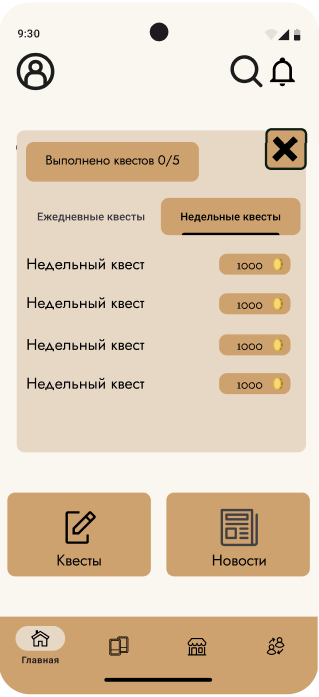
Настраиваемая страница профиля (см. рисунок 27) позволяет менять аватар, список избранных карт, список достижений, статистику собранных карт и коллекций. Содержит следующие элементы:

* Аватар пользователя.
* Никнейм пользователя.
* ID пользователя.
* Список избранных карт.
* Список избранных достижений.
* Статистика с параметрами собранных карт и коллекций.
* Кнопка настройки.
* Кнопка «Назад» в виде стрелочки



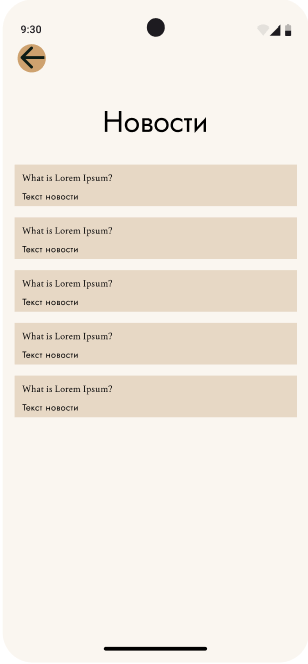
1. Экран «Профиль» пользователя
   * 1. Экран «Квесты»

Всплывающий экран «Квесты» имеет два раздела: «Ежедневные квесты» и «Недельные» (см. рисунок 28). После выполнения заданий, пользователь может отследить свой прогресс в верхней части экрана и получить награду. Ниже представлен переключатель между разделами и список самих заданий.

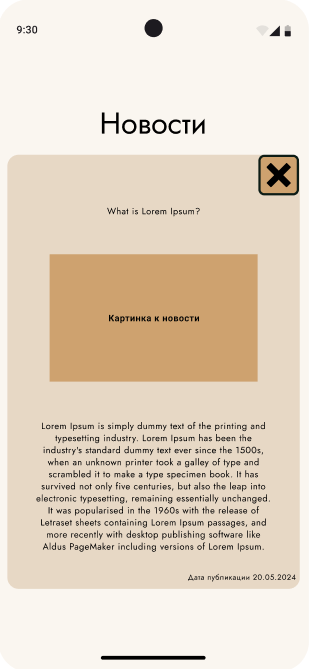
 

1. Экраны «Ежедневных» и «Недельных» квестов
   * 1. Экран «Новости»

Экран «Новости» представляет собой список актуальных новостей о вышедших обновлениях (см. рисунок 29), где каждое окно с новостью можно развернуть и просмотреть ее подробности (см. рисунок 30).



1. Экран списка новостей



1. Экран с развернутой новостью
   * 1. Экран «Принятия обмена»

В «Обменнике» пользователь может откликаться на два вида обменов:

* Пользователь, выставивший предложение обмена, указал список желаемых карт.
* Пользователь, выставивший предложение обмена, не указал список желаемых карт.

При отклике на 1 вариант обмена (см. рисунок 31) окно состоит из следующих элементов:

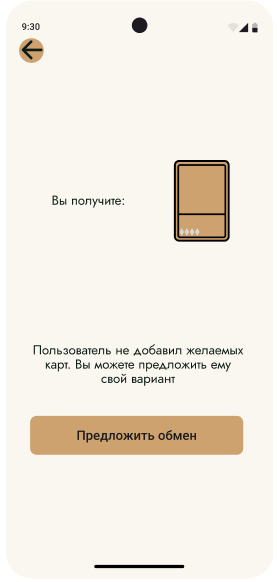
* Кнопка стрелочка «Назад».
* Карточка, которую пользователь может получить.
* Список карт, если обмен быстрый.
* Кнопка «Обменяться картой из списка» для совершения быстрого обмена.
* Кнопка «Предложить другую карту», если пользователь не хочет предоставлять на обмен карты из списка желаемого другого пользователя.

****

1. Экран отклика на первый вариант обмена

При отклике на второй вариант обмена (см. рисунок 32) окно состоит из следующих элементов:

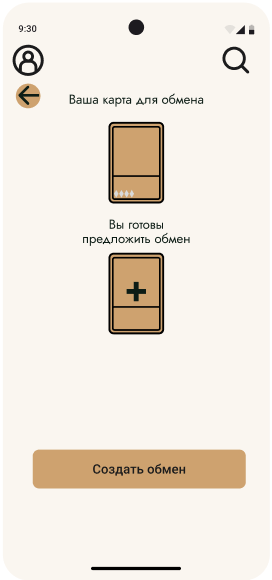
* Кнопка стрелочка «Назад».
* Карточка, которую пользователь может получить.
* Кнопка «Предложить обмен». После нажатия на эту кнопку пользователь перейдет в свой инвентарь и сможет выбрать для обмена одну из имеющихся карт.

****

1. Экран принятия обычного обмена
   * 1. Экран «Создания обмена»

При создании обмена (см. рисунок 33) открывается страница, состоящая из следующих элементов:

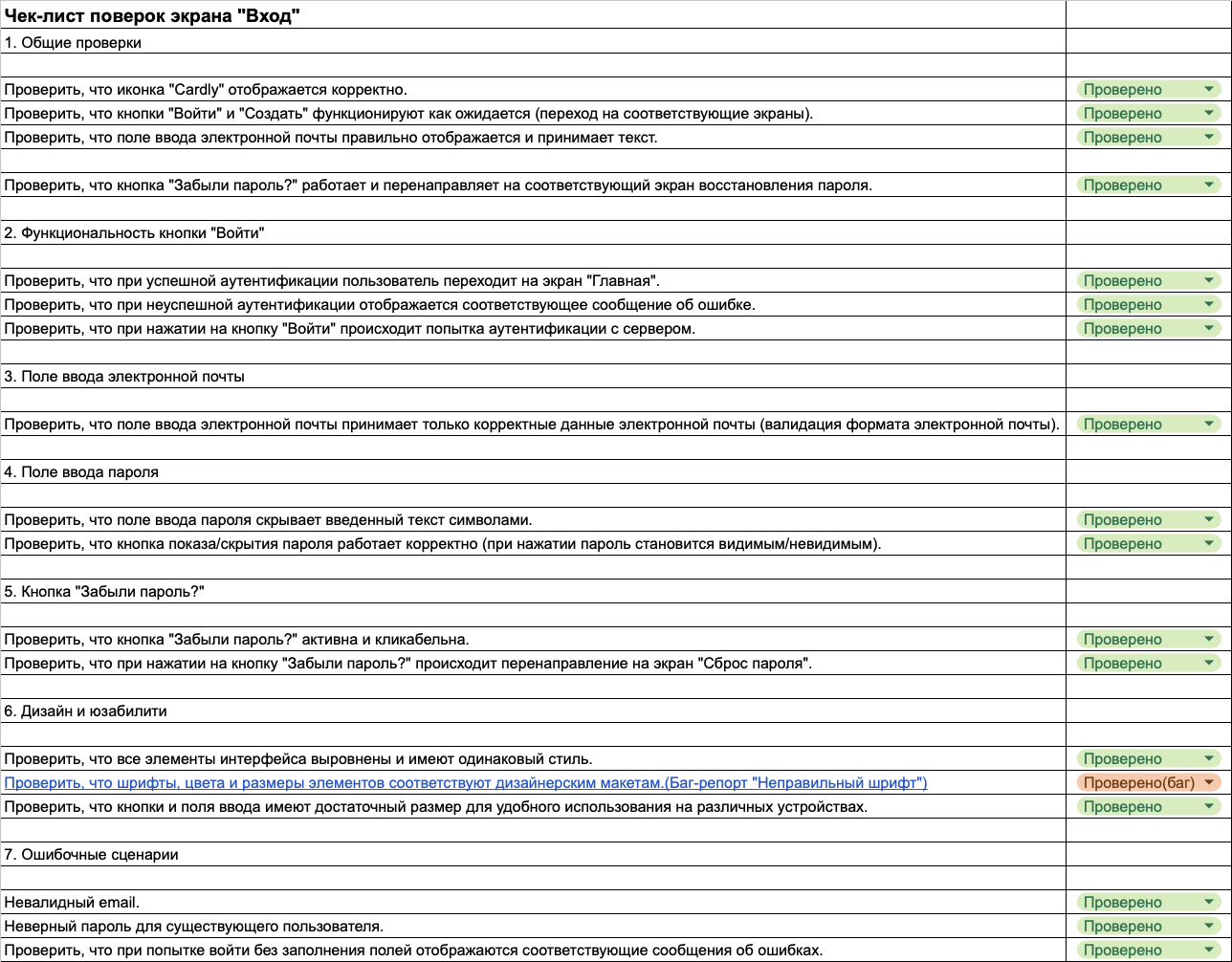
* Кнопка стрелочка «Назад».
* Окно для выбора карты, выставляемой на обмен.
* Окно выбора карты или списка карт для создания быстрого обмена.
* Кнопка «Создать обмен».

****

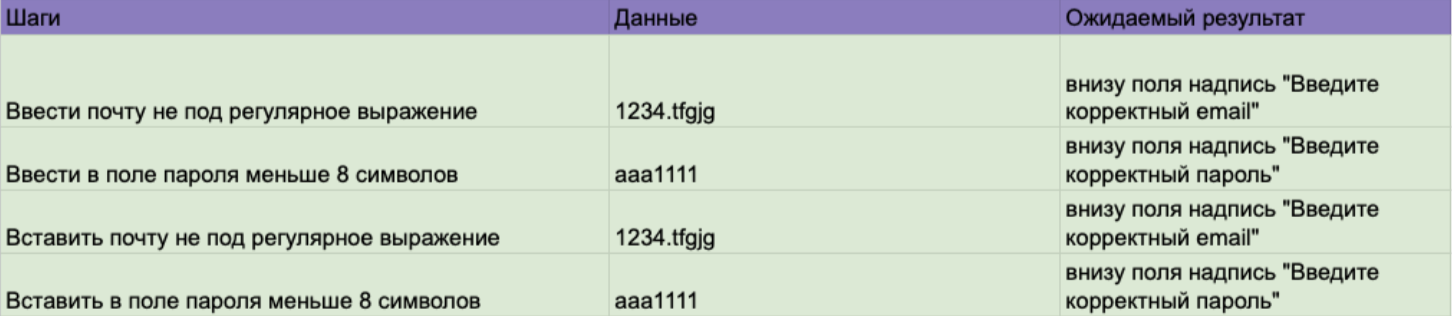
1. Страница создания обмена
2. Тестирование

Тестирование программного обеспечения представляет собой процесс проверки соответствия фактического поведения программы ожидаемым результатам на основе заранее подготовленного набора тестов, сформированного определённым образом. В рамках тестирования приложения были проведены функциональное, регрессионное и smoke-тестирование, а также применялись чек-листы для структурированной проверки второстепенных функций и элементов интерфейса. Функциональное тестирование выполнялось для подтверждения корректной работы ключевых функций приложения в соответствии с разработанными тест-кейсами. Регрессионное тестирование проводилось выборочно с целью проверки отсутствия новых ошибок после внесения изменений в программный код. Smoke-тестирование использовалось для оперативной проверки основных сценариев работы приложения после каждой новой сборки, что позволяло быстро оценить стабильность базовой функциональности. Чек-листы, в свою очередь, обеспечивали систематизацию проверки второстепенных функций, таких как корректность отображения текстов, кнопок и адаптивности интерфейса, что значительно ускоряло процесс тестирования.

На начальном этапе тестирования были разработаны тест-кейсы и чек-листы, охватывающие ключевые функциональные сценарии и элементы интерфейса приложения. Тест-кейсы составлялись вручную и фиксировались в Google Таблицах, что обеспечивало удобство совместной работы команды и оперативное обновление данных. Каждый тест-кейс включал последовательность шагов выполнения, входные данные или параметры для тестирования, а также описание ожидаемого результата, отражающее корректное поведение системы. Чек-листы, в отличие от тест-кейсов, представляли собой упрощённые списки задач, не требующие детального описания шагов, но позволяющие быстро оценить состояние второстепенных функций (см. рисунок 34).

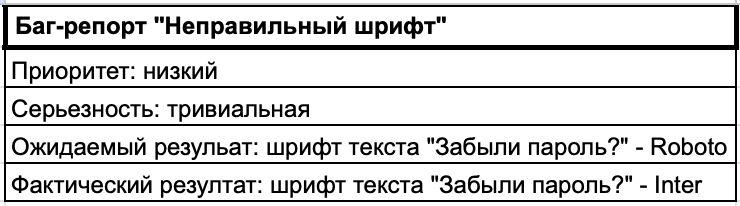


1. Пример чек-листа



1. Пример тест-кейса для одной из функций приложения

На следующем этапе осуществлялось непосредственное выполнение тестов. Тестирование проводилось вручную на устройствах с операционной системой Android, а также на устройствах с различными размерами экранов для проверки адаптивности интерфейса и кроссплатформенной совместимости. Тест-кейсы выполнялись в соответствии с заранее подготовленным планом, тогда как чек-листы использовались для дополнительной проверки второстепенных функций, что повышало полноту тестирования. При выявлении несоответствия фактического результата ожидаемому тест-кейс или пункт чек-листа отмечался как неуспешный, после чего создавался баг-репорт с подробным описанием проблемы, шагами воспроизведения и, при необходимости, скриншотами (см. рисунок 36). Все баг-репорты фиксировались в Google Таблицах, что обеспечивало удобное отслеживание дефектов и координацию работы команды.



1. Пример баг-репорта

После устранения выявленных дефектов проводилась повторная проверка. Если изменения касались только конкретного бага, выполнялись шаги воспроизведения из соответствующего тест-кейса или чек-листа для подтверждения устранения проблемы. В случае успешной проверки задача переводилась в статус «Готово». Регрессионное тестирование проводилось выборочно в тех случаях, когда изменения затрагивали уже протестированные участки функциональности, чтобы гарантировать отсутствие новых ошибок. Использование чек-листов на этапе повторной проверки позволило сократить время на тестирование второстепенных функций, сохранив при этом высокую степень контроля качества.

1. Аналитика

Для сбора статистики по использованию приложения Cardly была выбрана система AppMetrica от Яндекса. Это решение обусловлено ориентацией проекта на российский рынок, а также удобством интеграции и широкими возможностями анализа пользовательского поведения.

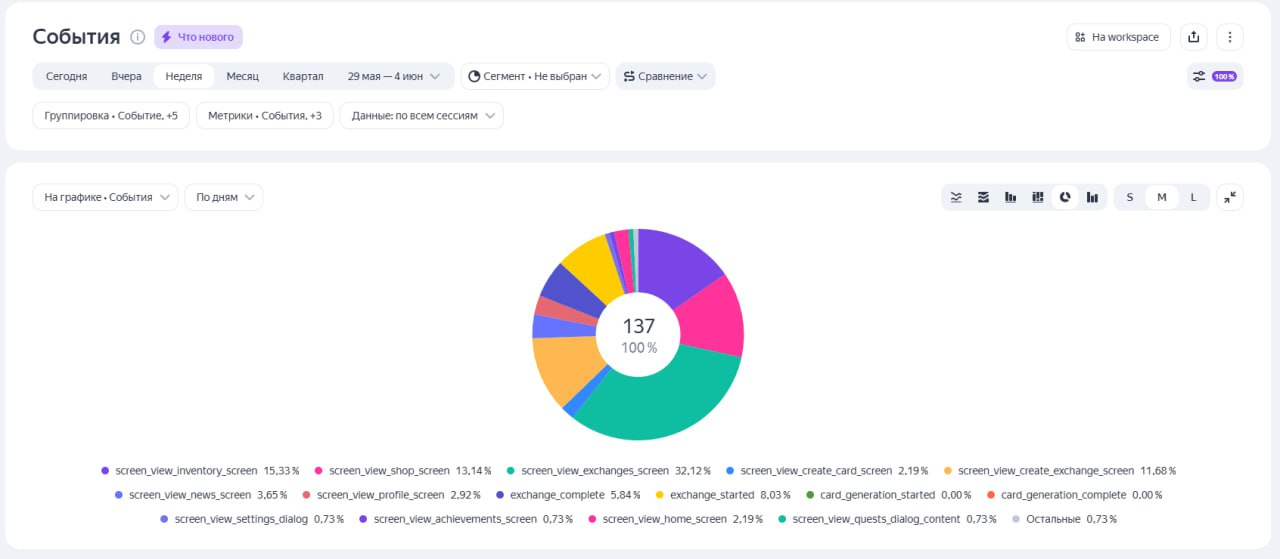
Сервис позволяет отслеживать установки приложения, действия пользователей внутри него и составлять отчеты по ключевым метрикам, не прибегая к сложной настройке. Также важным фактором при выборе стала доступность интерфейса на русском языке и отсутствие ограничений на использование в России.

* 1. Преимущества AppMetrica
* Полностью на русском языке;
* Легко интегрируется с Android-приложениями;
* Предоставляет подробную аналитику;
* Бесплатный сервис без ограничений в объёме событий или количестве пользователей;
  1. Воронки

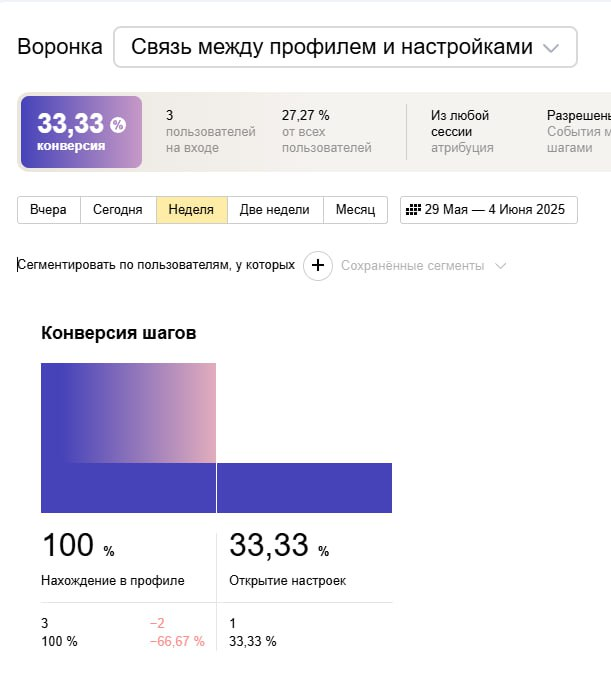
Для формирования аналитики по приложению были созданы четыре воронки:

* Регистрация.
* Покупка в магазине.
* Обмен с другим пользователем.
* Генерация уникальной карты через ИИ.
  1. Статистика

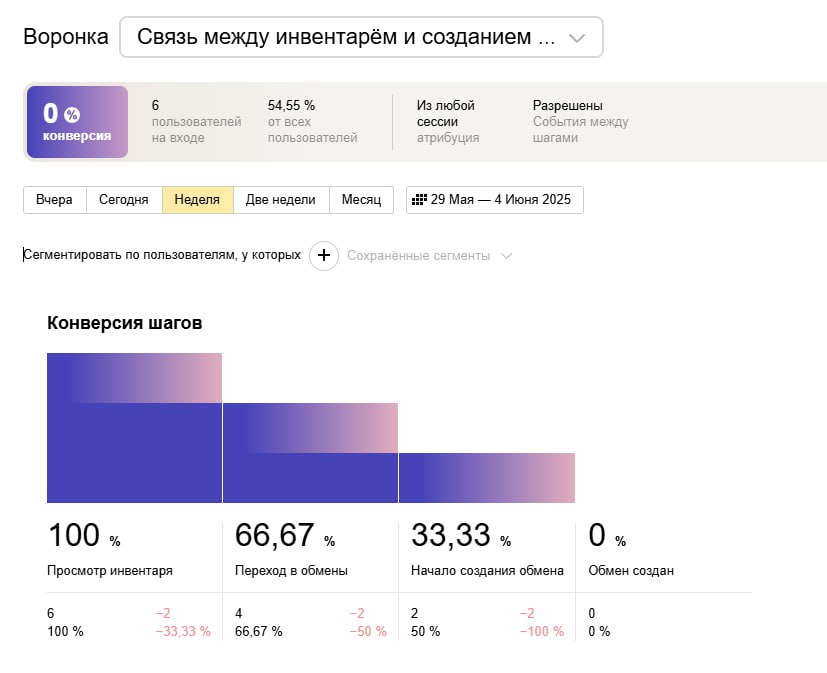
На рисунках продемонстрированы статистики по количеству событий в приложении. На рисунке 37 показано общее число заходов на разные экраны. Рисунок 38 демонстрирует одну из воронок перехода в настройки профиля. Рисунок 39 показывает воронку создания обмена через инвентарь.



1. Число событий заходов на экраны приложения



1. Пример воронки между инвентарем и обменом



1. Пример воронки создания обмена из инвентаря

# ЗАКЛЮЧЕНИЕ

Во время работы над проектом «Cardly» нам удалось пройти полный путь от идеи до полноценного мобильного приложения. Мы подробно изучили рынок коллекционных карточек, проанализировали интересы пользователей и разработали решение, которое сочетает в себе механику коллекционирования, элемент случайности и возможность взаимодействия между пользователями.

Главной целью было создать приложение, в котором пользователи могли бы открывать наборы, собирать собственные коллекции и обмениваться карточками с другими. Мы также добавили функцию генерации уникальных карточек с помощью нейросети, чтобы каждый пользователь мог создать что-то по-настоящему своё.

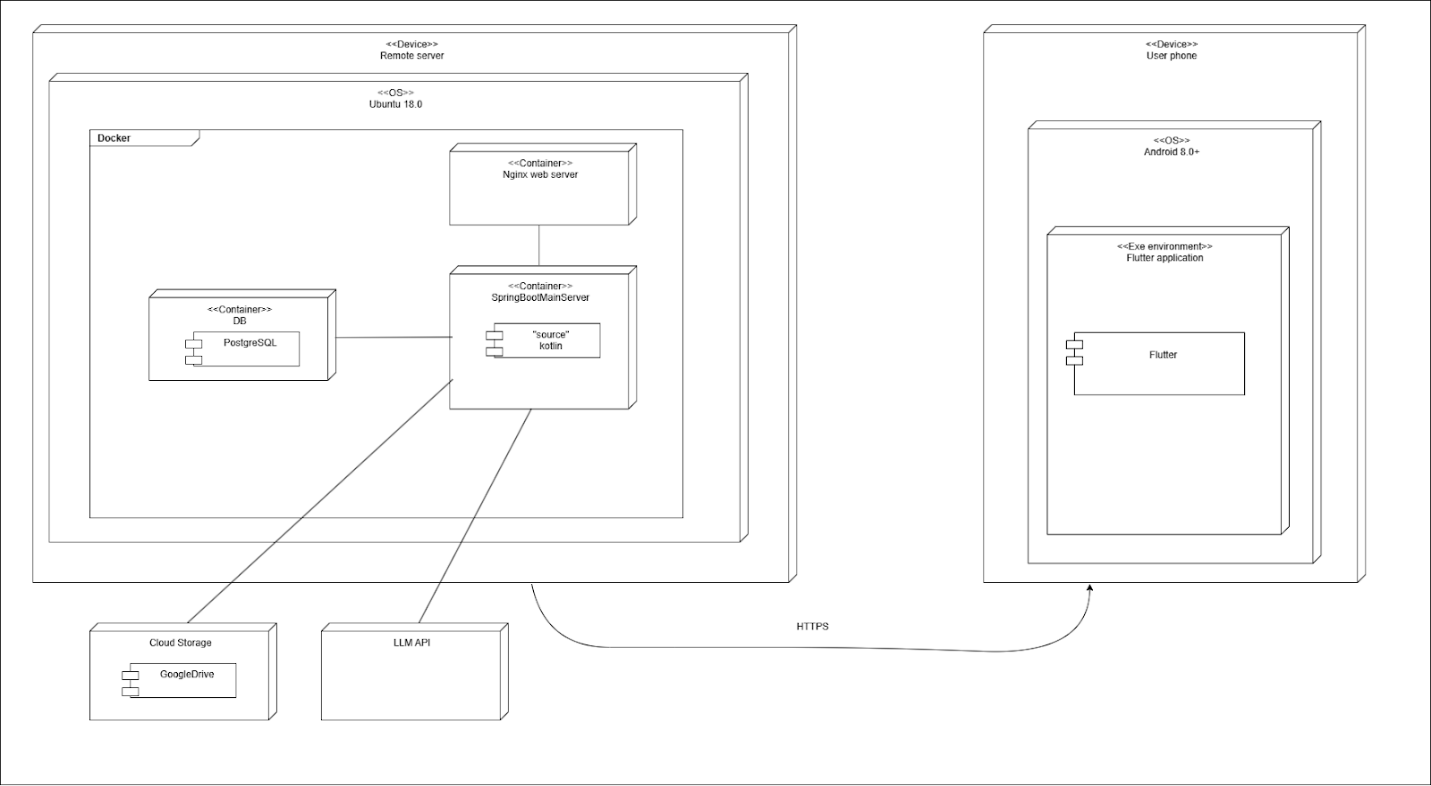
Основная функциональность приложения «Cardly»:

* Просматривать наборы коллекционных карточек и взаимодействовать с ними.
* Просматривать профили пользователей и совершать с ними обмен.
* Совершать быстрые и обычные обмены.
* Коллекционировать карточки.
* Создавать уникальные карточки с помощью ИИ.
* Выполнять квесты.
* Осуществлять редактирование данных своего аккаунта после регистрации или авторизации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

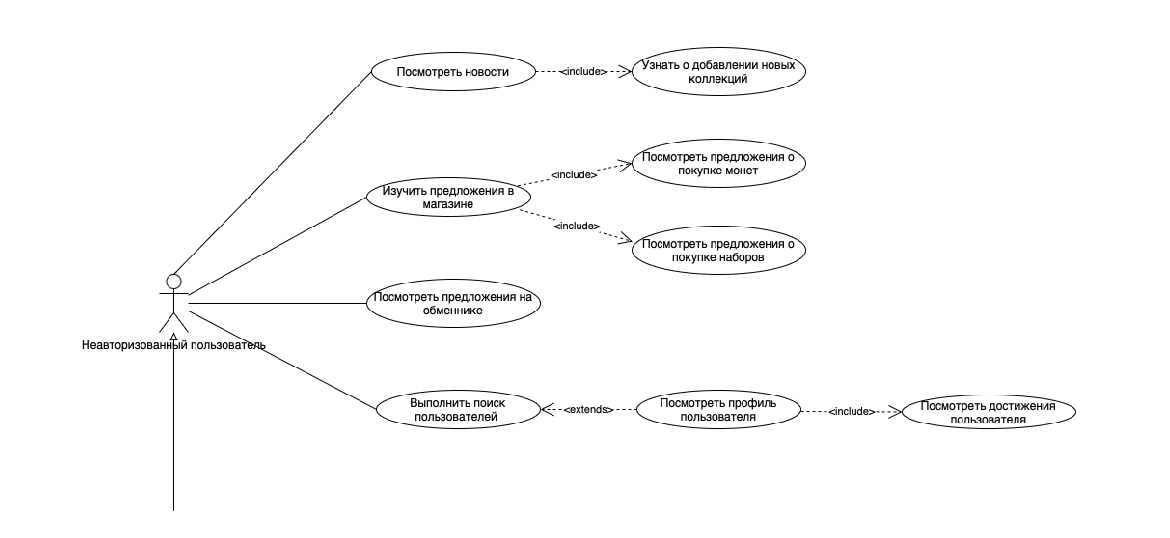
1. Вигерс, К. Разработка требований к программному обеспечению / К. Вигерс, Дж. Битти. – 3-е изд., дополненное. – Москва : Русская редакция, 2014. – 736 с. – ISBN 978-5-7502-0433-5. – Текст : непосредственный.
2. Документация Flutter [Электронный ресурс]. – Режим доступа: https://docs.flutter.dev/ – Заглавие с экрана. – (Дата обращения 05.04.2025).
3. Документация Kotlin [Электронный ресурс]. – Режим доступа: https://kotlinlang.org/docs/home.html# – Заглавие с экрана. – (Дата обращения 09.04.2025).

# ПРИЛОЖЕНИЕ А

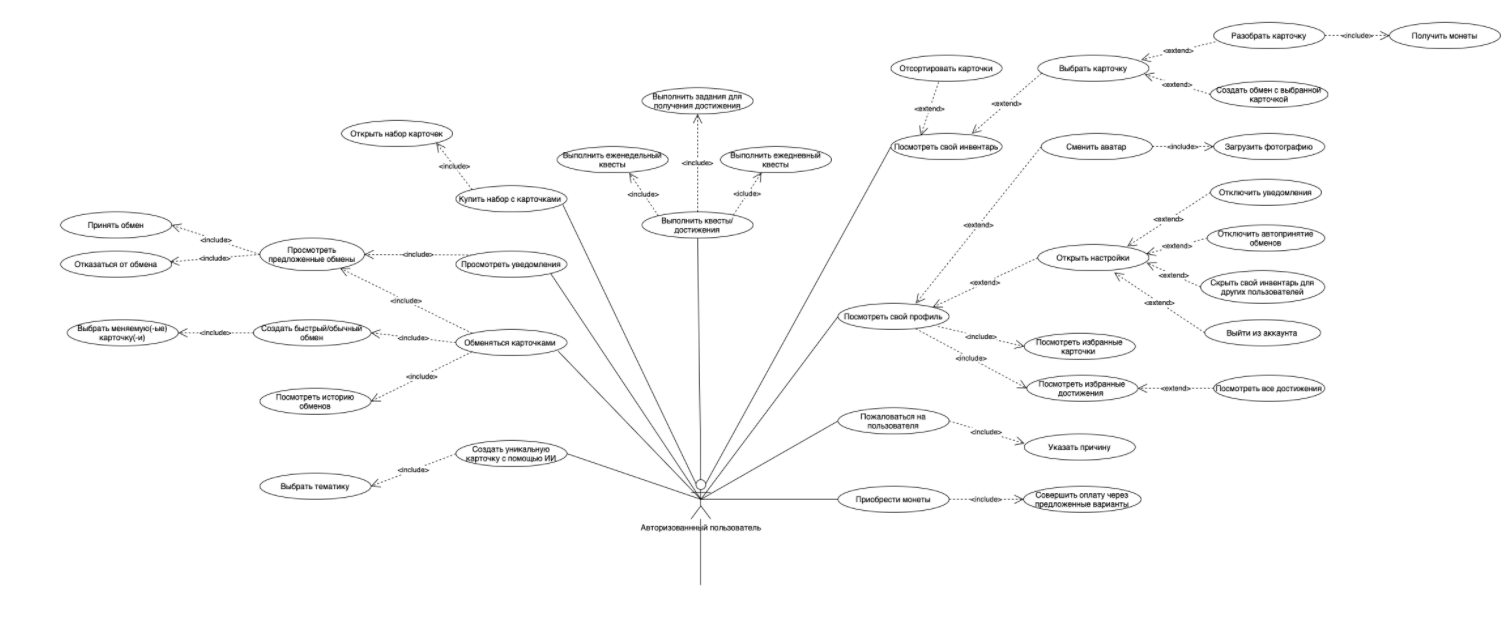


1. Deployment diagram.

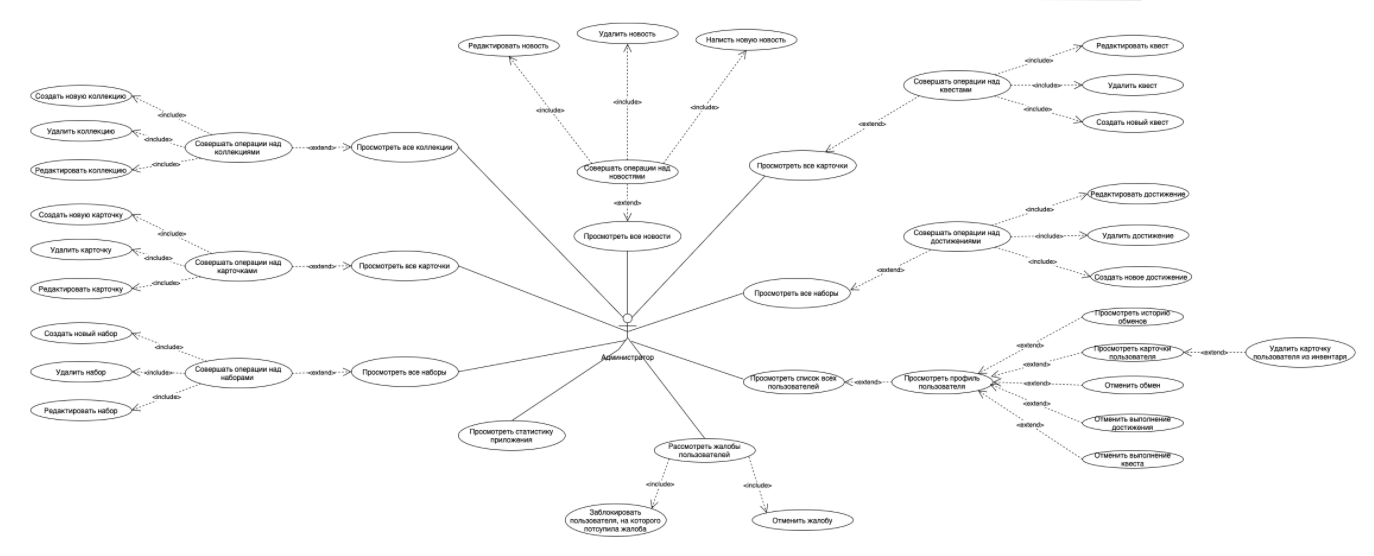
# ПРИЛОЖЕНИЕ Б



1. Use case Неавторизованный пользователь

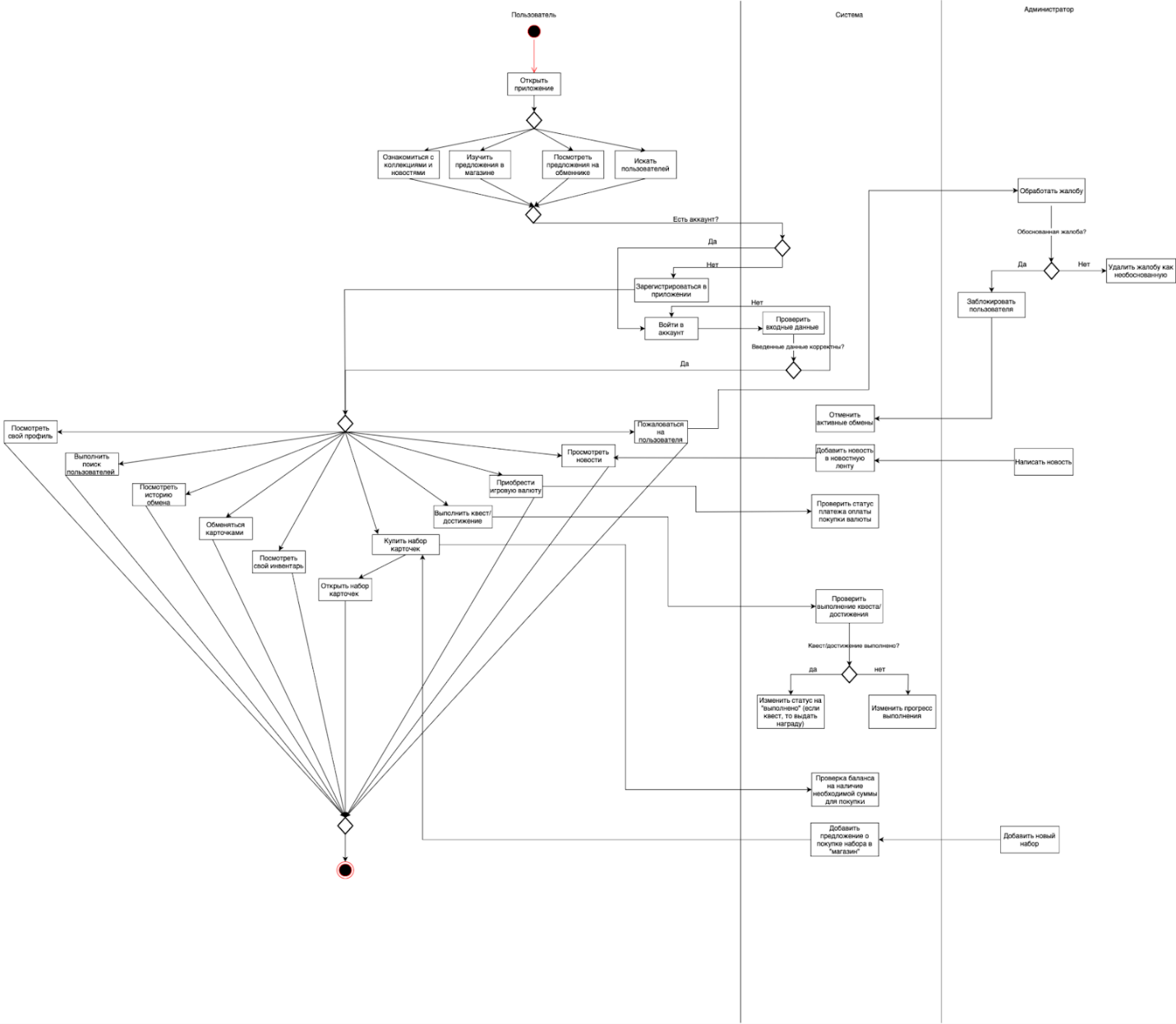


1. Use case Авторизованный пользователь



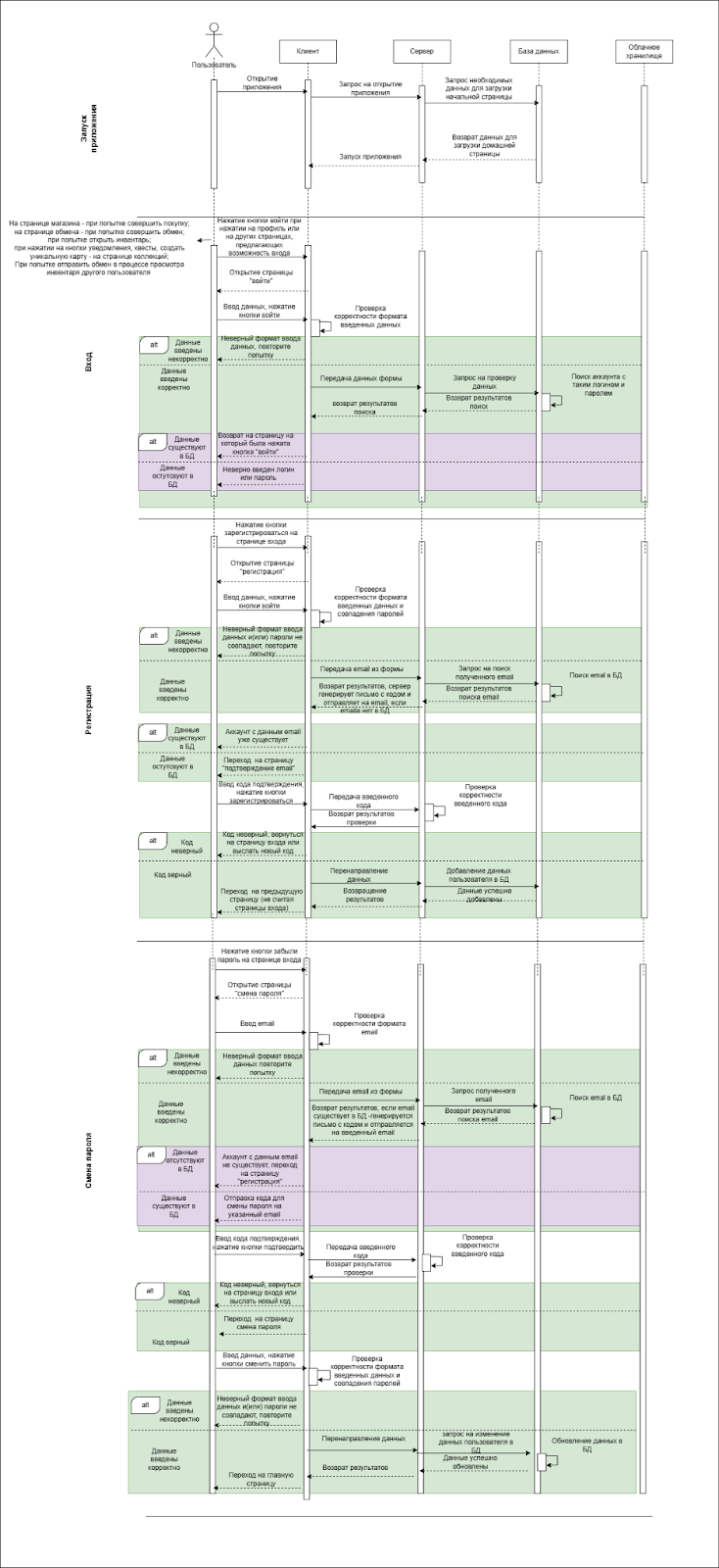
1. Use case Администратор

# ПРИЛОЖЕНИЕ В

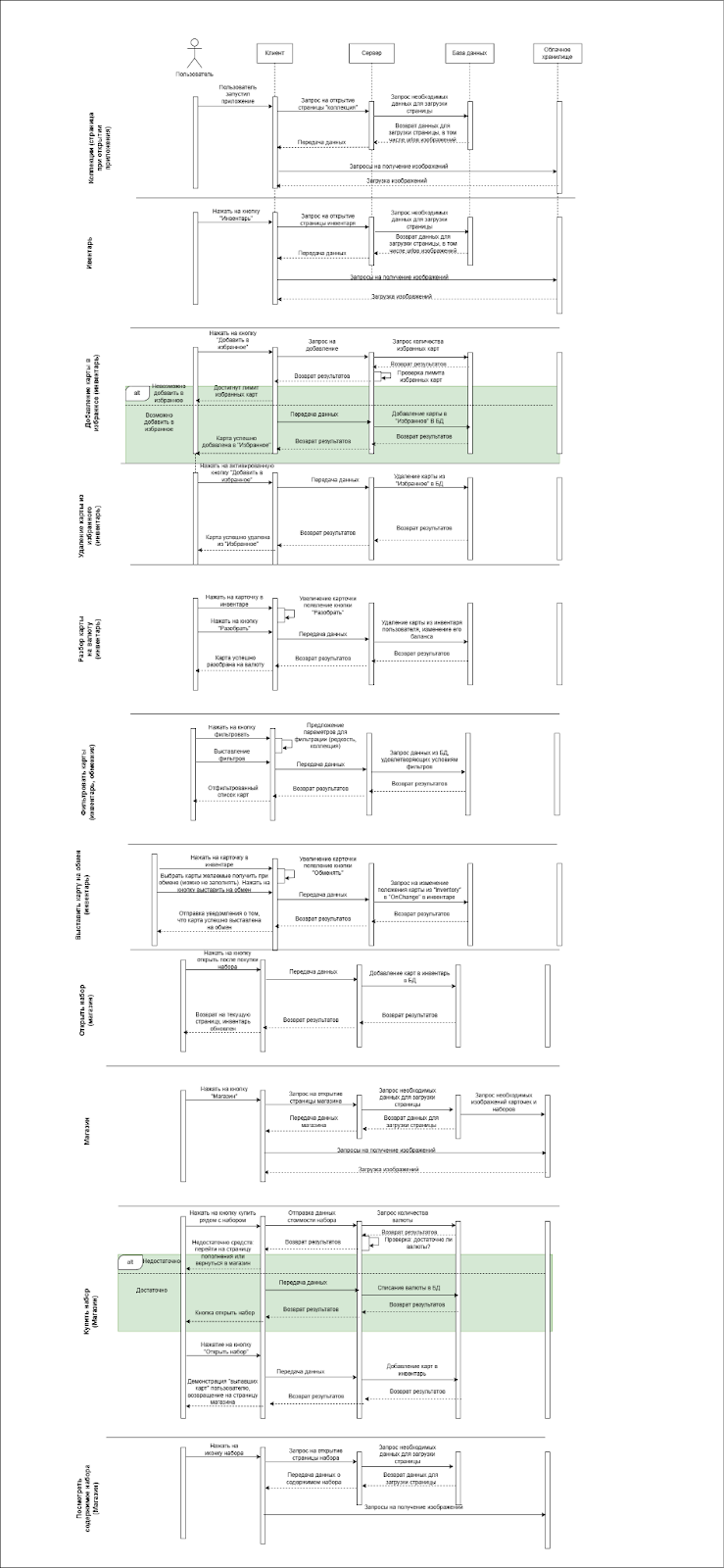


1. Activity diagram

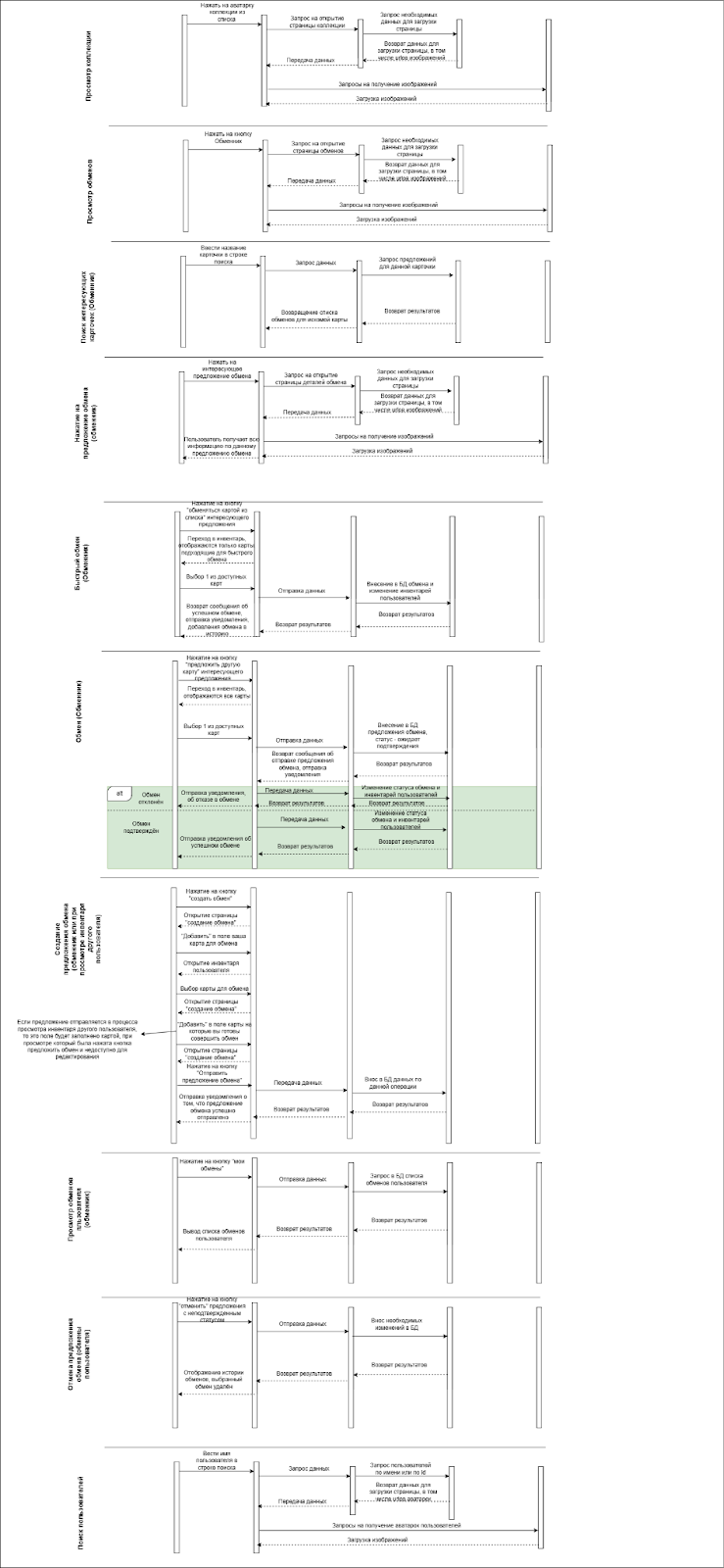
# ПРИЛОЖЕНИЕ Г



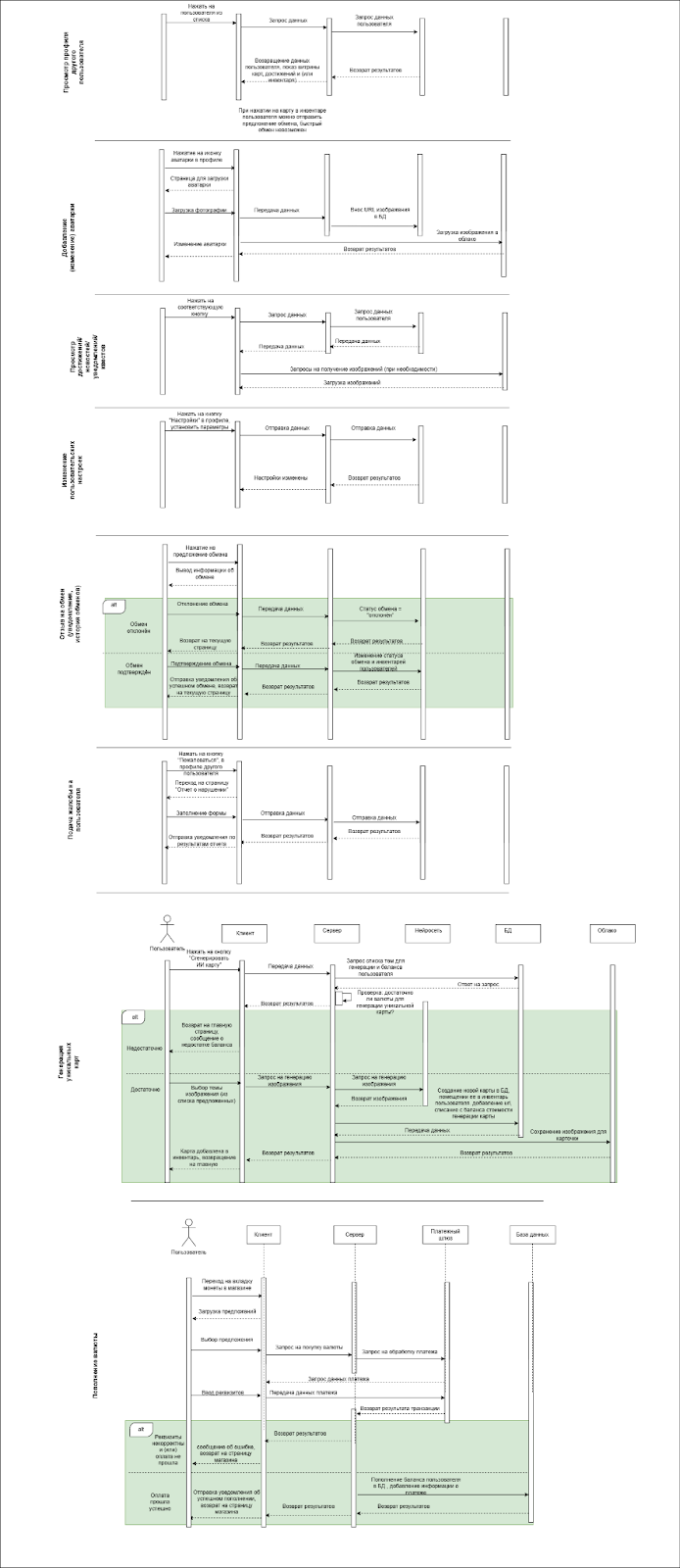
1. Authorization sequence diagram



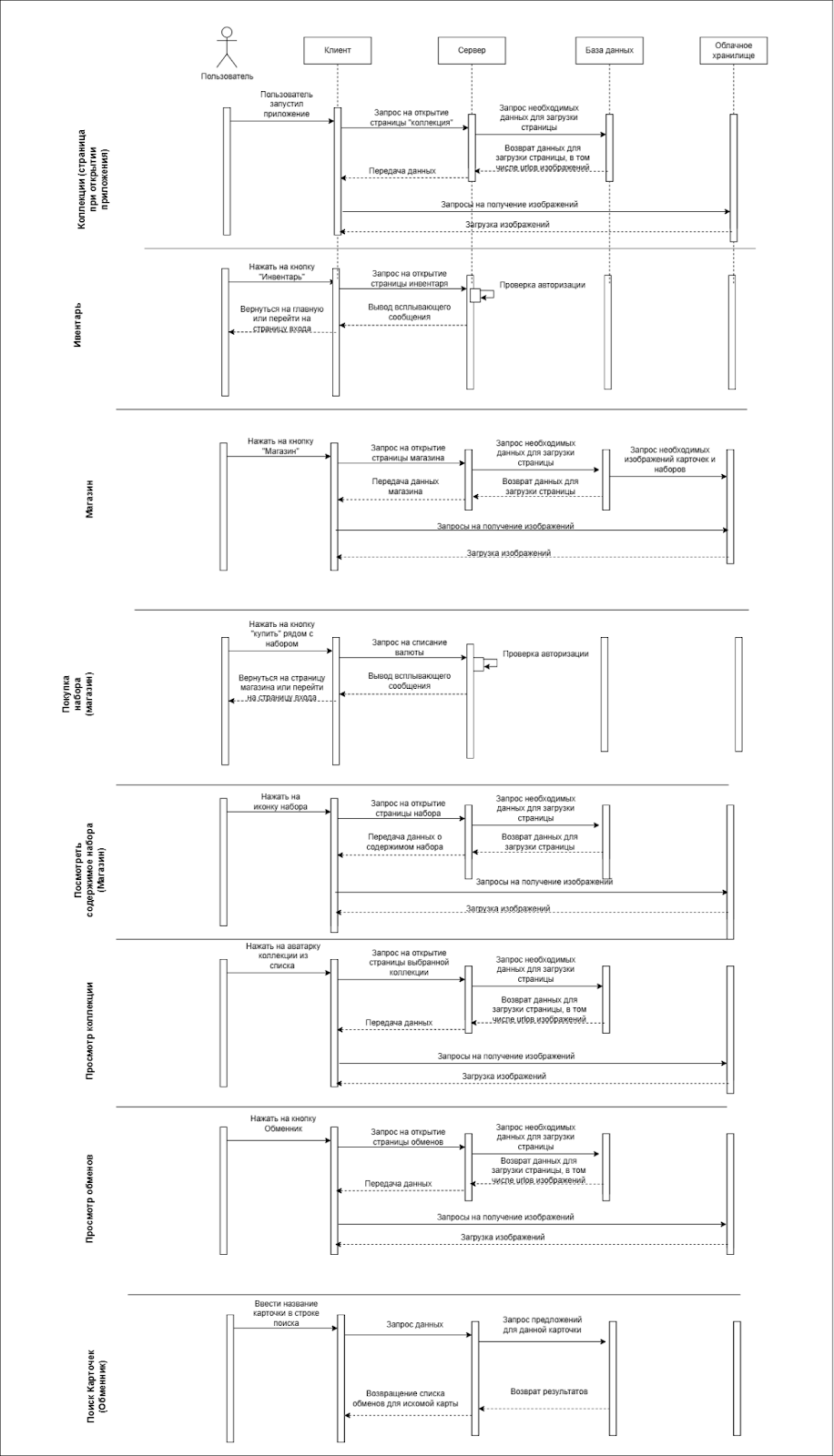
1. Authorized user sequence diagram (1)



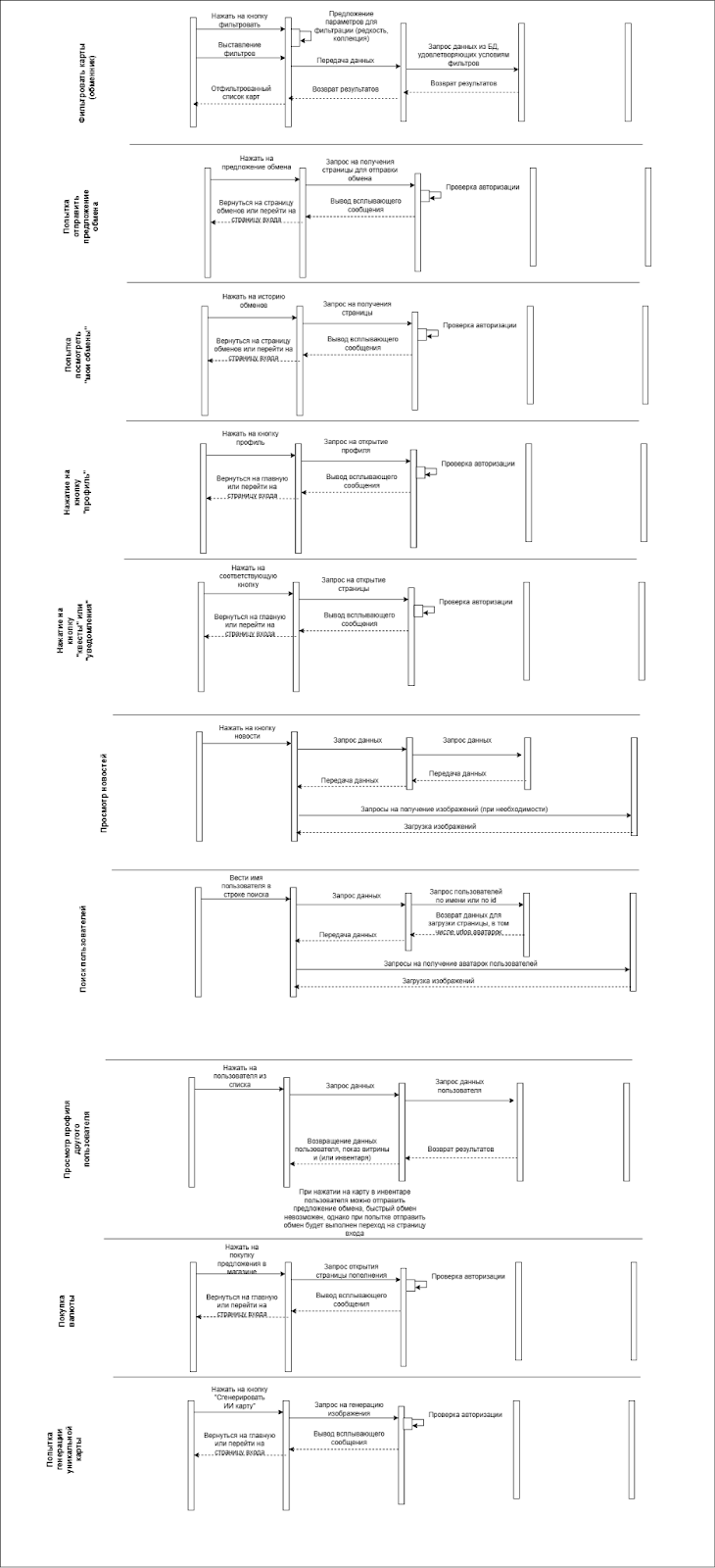
1. Authorized user sequence diagram (2)



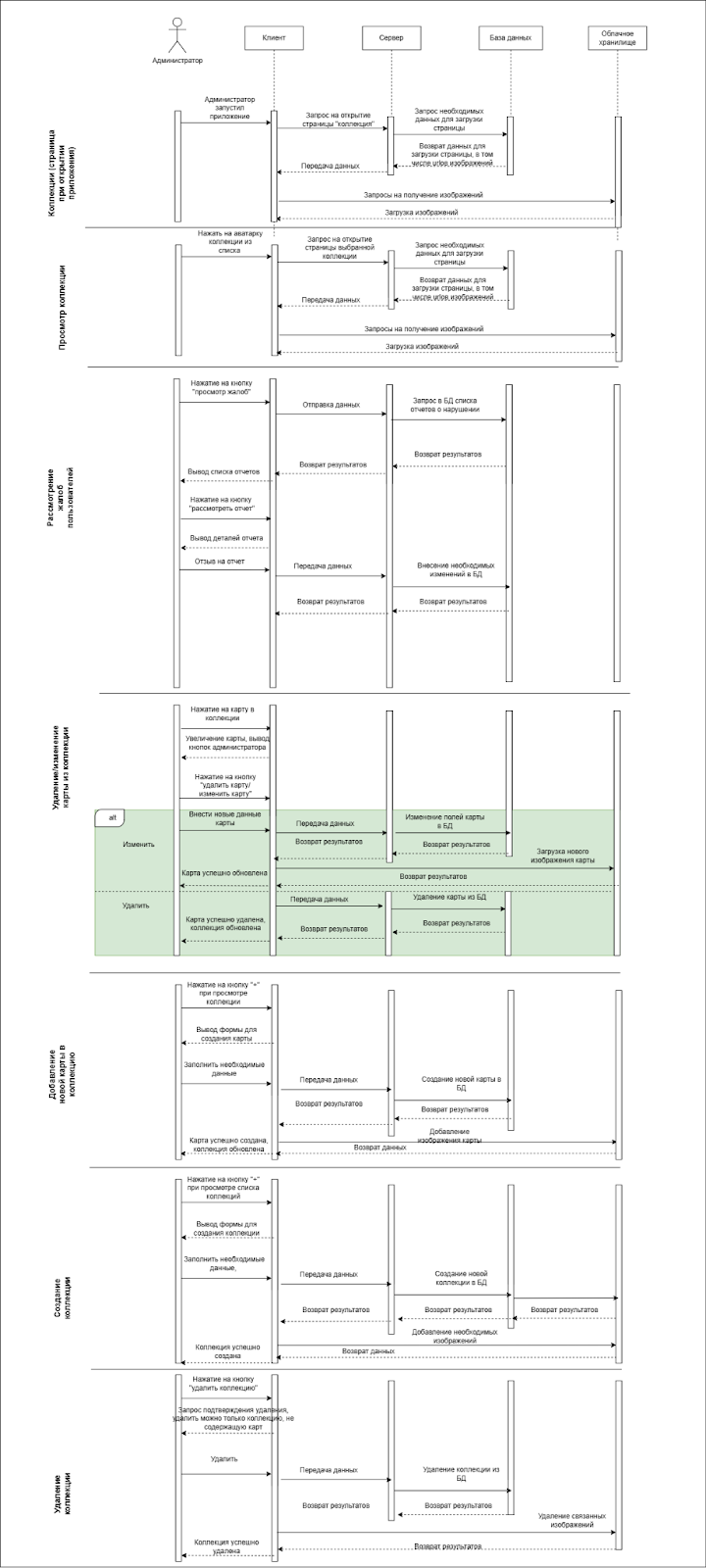
1. Authorized user sequence diagram (3)



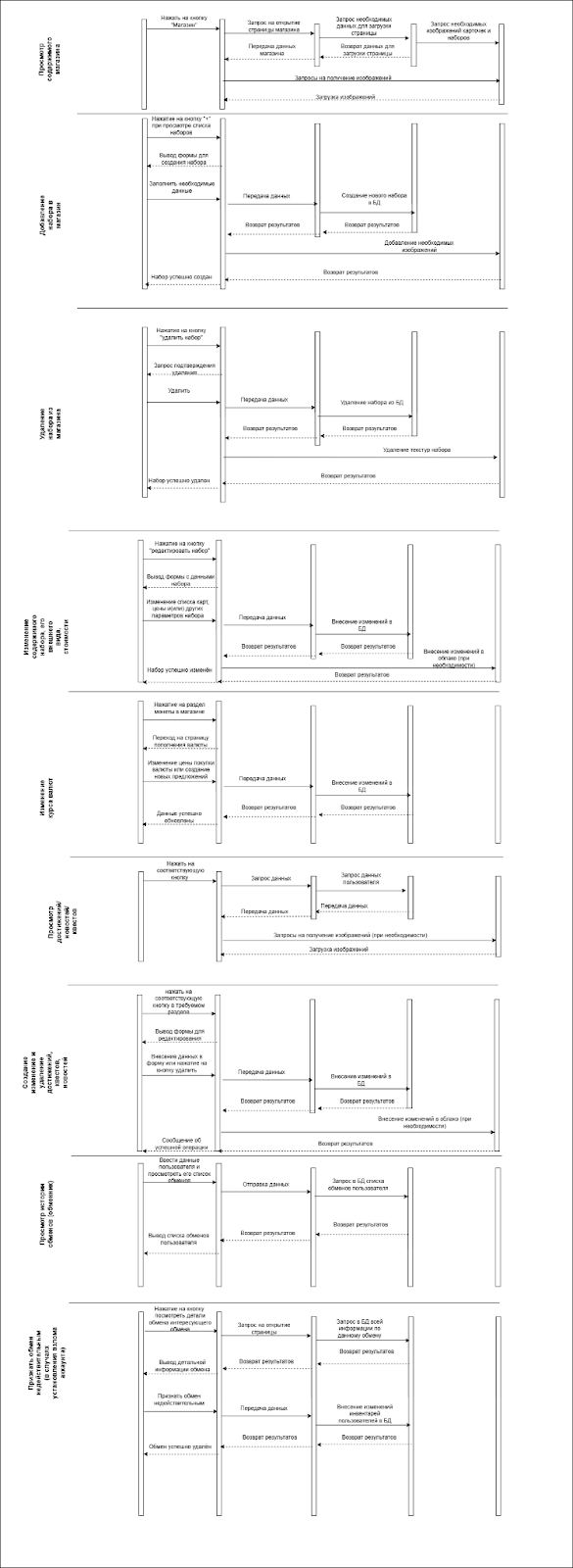
1. Unauthorized user sequence (1)



1. Unauthorized user sequence (2)

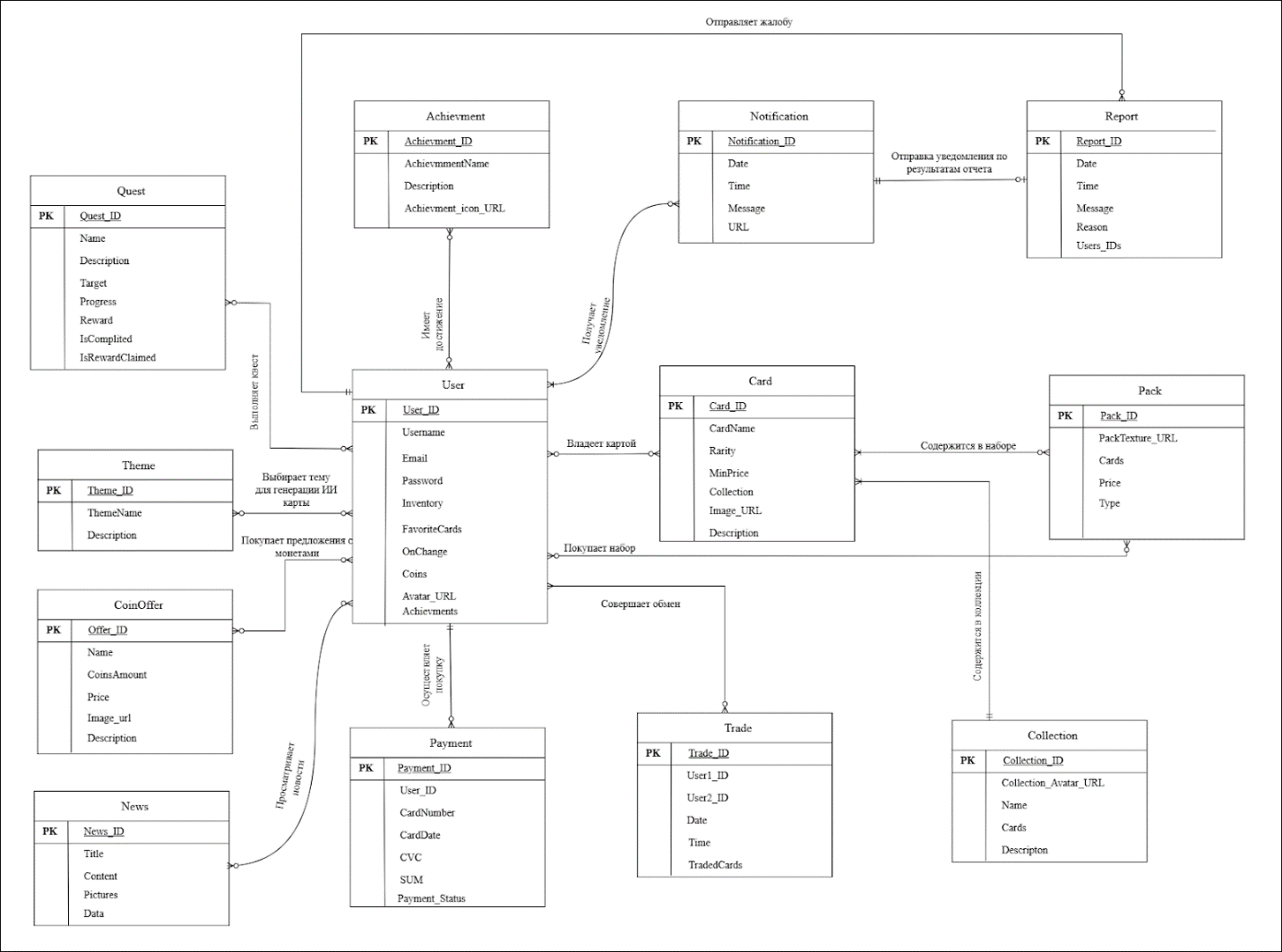


1. Admin sequence (1)



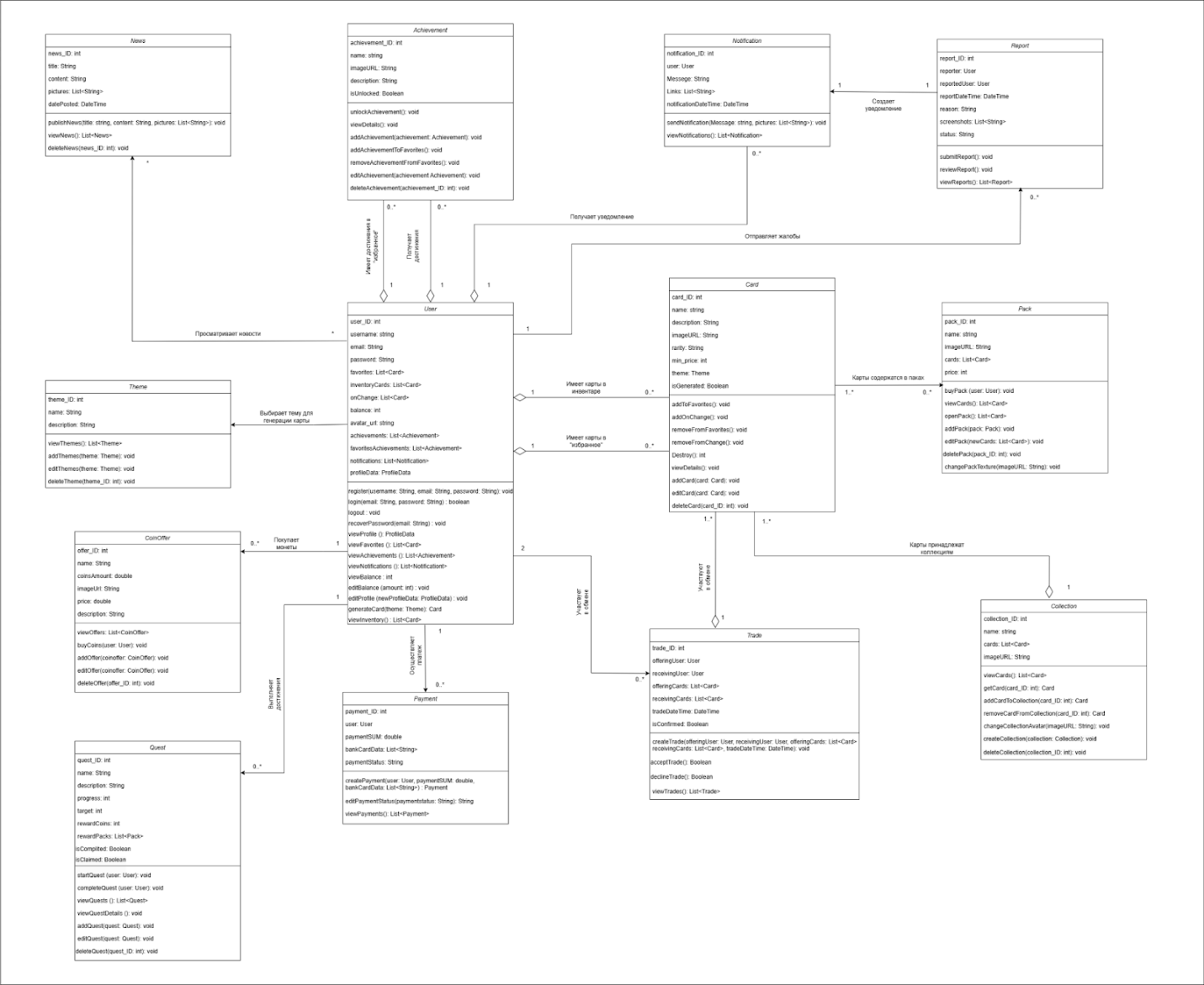
1. Admin sequence (2)

# ПРИЛОЖЕНИЕ Д



1. ER diagram

# ПРИЛОЖЕНИЕ Е



1. Class diagram