

## 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.

Целью курсовой работы является разработка мобильного приложения «Cardly» для обмена коллекционными карточками и управления персональными коллекциями.

Задачи:

- Провести анализ предметной области и обзор аналогичных решений;
- Разработать мобильное приложение «Cardly», включающее функционал для создания, обмена и управления коллекционными карточками.

## **1 Постановка задачи**

### **1.1 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.

### **1.2 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.

## **1.3 Требования к приложению**

### **1.3.1 Требования к приложению в целом**

Данное приложение должно удовлетворять следующим основным требованиям:

- приложение должно корректно работать на устройствах, работающих на операционной системе Android 8.0 и новее;
- реализовывать все поставленные задачи.

### **1.3.2 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:

- авторизоваться/зарегистрироваться в приложении;
- сбросить пароль
- осуществлять поиск авторизованных пользователей;
- просматривать профили других пользователей;
- просматривать инвентарь пользователя;
- просматривать каждую полученную карту пользователя отдельно;
- просматривать магазин;
- просматривать раздел «Наборы»;
- просматривать раздел «Монеты»;
- просматривать каждое предложение в обоих разделах по отдельности и его составляющие характеристики;
- просматривать список актуальных новостей-обновлений;
- просматривать каждую новость отдельно;
- просматривать список коллекций;
- просматривать содержимое каждой коллекции карт
- просматривать каждую карточку коллекции отдельно и информацию о ней;
- просматривать список активных обменов авторизованных пользователей;
- применять сортировки для отображения списка обменов;
- осуществлять поиск предложений обменов для интересующей его

карты по ее названию;

Авторизованный пользователь должен обладать возможностью:

- всеми возможностями неавторизованного пользователя;
- осуществлять быстрый обмен;
- осуществлять обычный обмен;
- отправлять жалобы;
- создавать обмены;
- откликаться на предложения обмена;
- создавать уникальные карточки;
- просмотреть список доступных квестов;
- пользователь должен иметь возможность отслеживать статус выполненных квестов;
- получать награду в виде набора или монет за выполненные квесты;
- применять сортировки к списку собранных карт;
- просматривать каждую карту отдельно;
- разобрать карту, то есть обменять ее на некоторое количество монет, которое изначально заложено для каждой карты;
- добавить карту в избранные;
- выставить карту на обмен выбрав один из способов: быстрый обмен или обычный обмен;
- покупать наборы карточек за монеты;
- пополнять баланс монет по выбранному предложению путем доната;
- менять свой аватар;
- просматривать список избранных карт;
- просматривать избранные достижения;
- просматривать весь список достижений, с возможностью добавить достижение в избранное;
- просматривать статистику полученных карт и собранных коллекций;
- заходить в настройки и иметь возможность включать/выключать уведомления, показывать/скрывать свой инвентарь от других пользователей, включать/выключать автоотклонение входящих обменов;

Администратор должен обладать возможностью:

- заблокировать пользователя, что запрещает вход в систему и выполнение любых действий;
- снимать блокировку, позволяя пользователю вернуться в систему;
- обладать доступом к детальной информации о пользователе (инвентарь, история обменов, достижения) для принятия обоснованных решений;
- отменять выполнение достижения или квеста в случае установления нарушений правил;
- отменять обмены пользователя в случае установления нарушений правил;
- удалять карточки из инвентаря пользователя;

- обладать доступом к полному списку жалоб, содержащих информацию о причине заявки;
- взять заявку в работу, чтобы пометить её как обрабатываемую, что предотвращает одновременную работу нескольких администраторов над одной заявкой;
- принимать/отклонять жалобу на пользователя и блокировать или нет данного пользователя;
- написать публикацию новости;
- запланировать публикацию новости;
- создания, редактирования, удаления карточек, коллекций, наборов, квестов и достижений;
- просматривать метрики, помогающие анализировать, управлять и оптимизировать ключевые аспекты работы приложения: активность пользователей, популярность карт, эффективность модерации и техническая стабильность;

### **1.3.3 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".

### **1.3.4 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.

## **1.4 Задачи, решаемые в процессе разработки**

Были поставлены следующие задачи:

- анализ предметной области;
- обзор аналогов;
- постановка задачи;

- создание репозитория GitHub и доски в 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.



## **2 Анализ предметной области**

### **2.1 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.

## **2.2 Обзор аналогов**

На русскоязычном рынке не существует приложения для организации деятельности книжных клубов, поэтому клубы в основном ведут свою деятельность в социальных сетях в виде групп и сообществ.

На зарубежном рынке есть несколько приложений для автоматизации деятельности книжных клубов. Далее мы рассмотрим их достоинства и недостатки, чтобы разработать функционал приложения, основываясь на том, чего не хватает пользователю в существующих решениях.

## 2.2.1 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.

### Торговая площадка

Продавайте предметы участникам сообщества или приобретайте их, используя средства из кошелька Steam.

Активные лоты (0)

История сделок







Продать предмет

Войдите в систему, чтобы получить возможность просмотра, редактирования и удаления лотов Торговой площадки сообщества.

Популярные

Новые

Недавно проданные









НАЗВАНИЕ	КОЛ-ВО	ЦЕНА
 <b>Кейс «Грёзы и кошмары»</b> Counter-Strike 2	80,852	От \$2.07 USD
 <b>Кейс «Киловатт»</b> Counter-Strike 2	85,500	От \$0.87 USD
 <b>Кейс «Революция»</b> Counter-Strike 2	108,181	От \$0.66 USD
 <b>Кейс «Разлом»</b> Counter-Strike 2	228,388	От \$0.49 USD
 <b>Гремучий кейс</b> Counter-Strike 2	209,309	От \$0.35 USD
 <b>Гремучий кейс</b> Counter-Strike 2		

**Прочтите требования к безопасности** для использования Торговой площадки

Поиск предметов

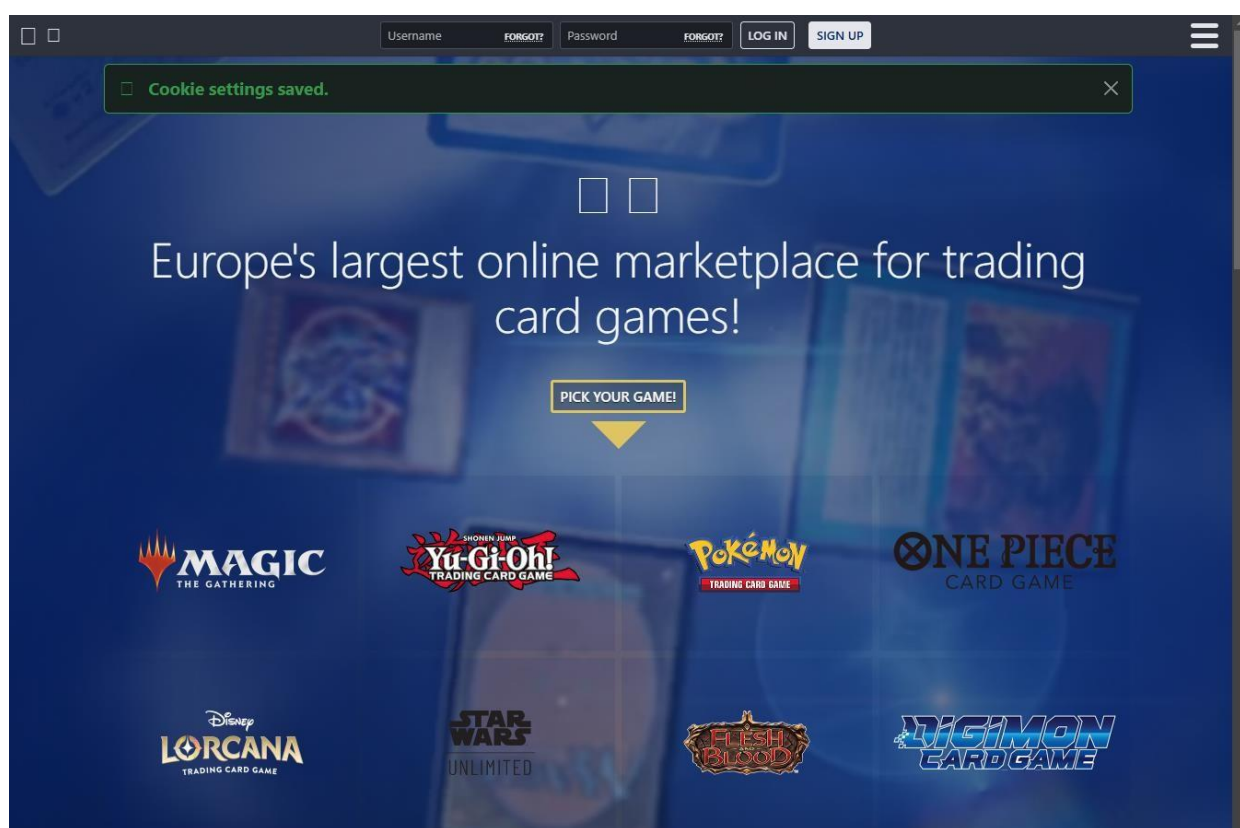
Дополнительные настройки поиска...

Фильтр по игре

-  Counter-Strike 2
-  Don't Starve Together
-  Dota 2
-  Fish Idle 2: Underwater Mystery
-  Longvinter
-  PUBG: BATTLEGROUNDS
-  Royal Quest Online
-  Rust

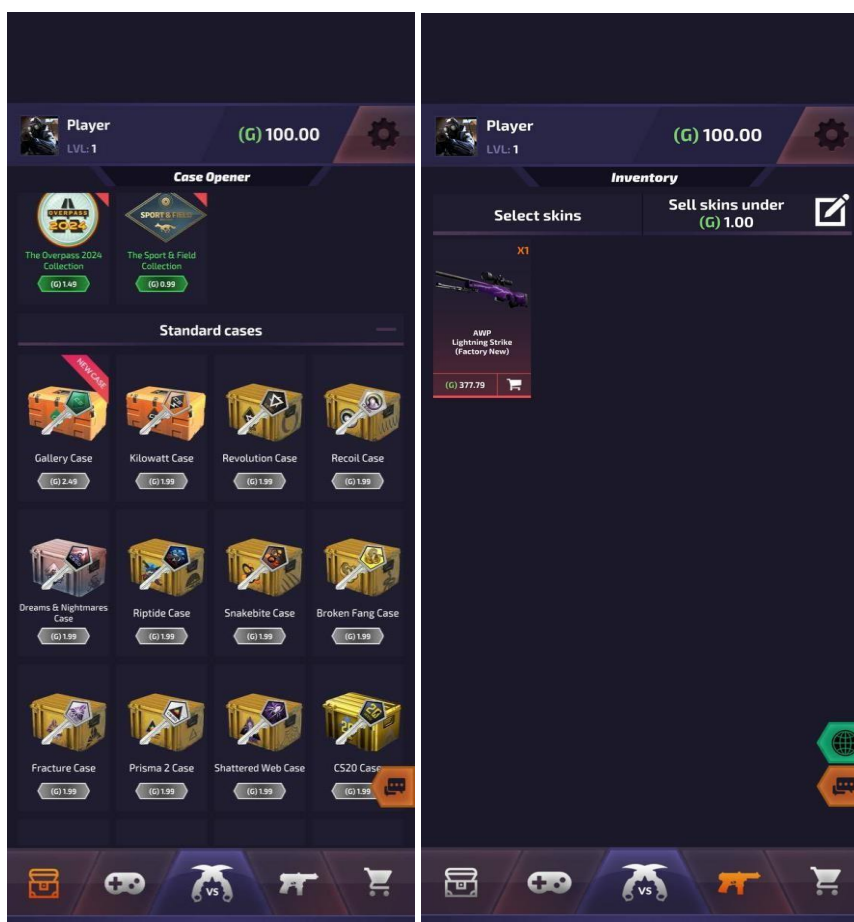
## 2.2.2 Cardmarket

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.



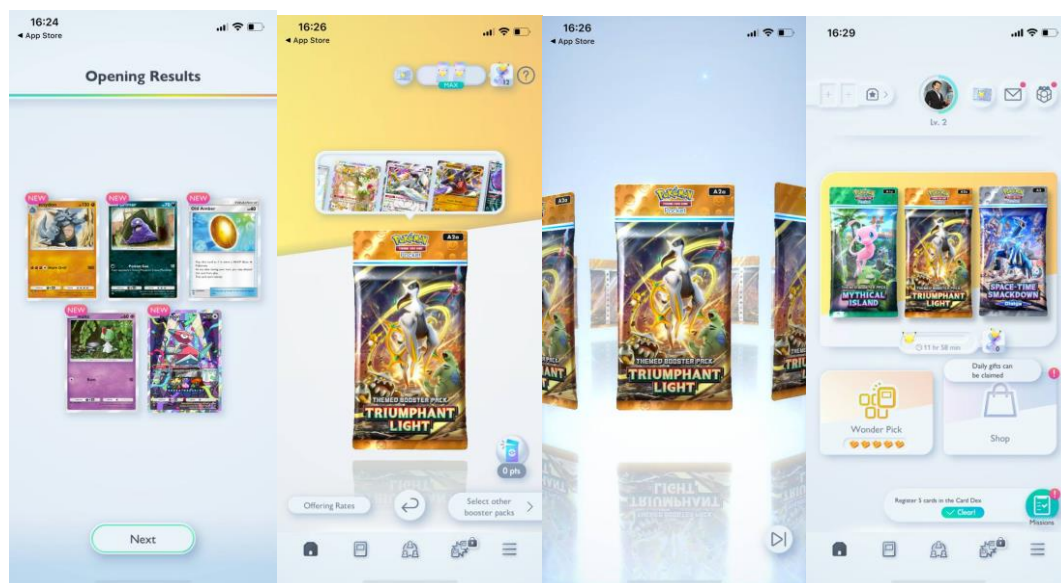
### 2.2.3 Case Battle

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.



## 2.2.4 Pokemon TCGP

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.

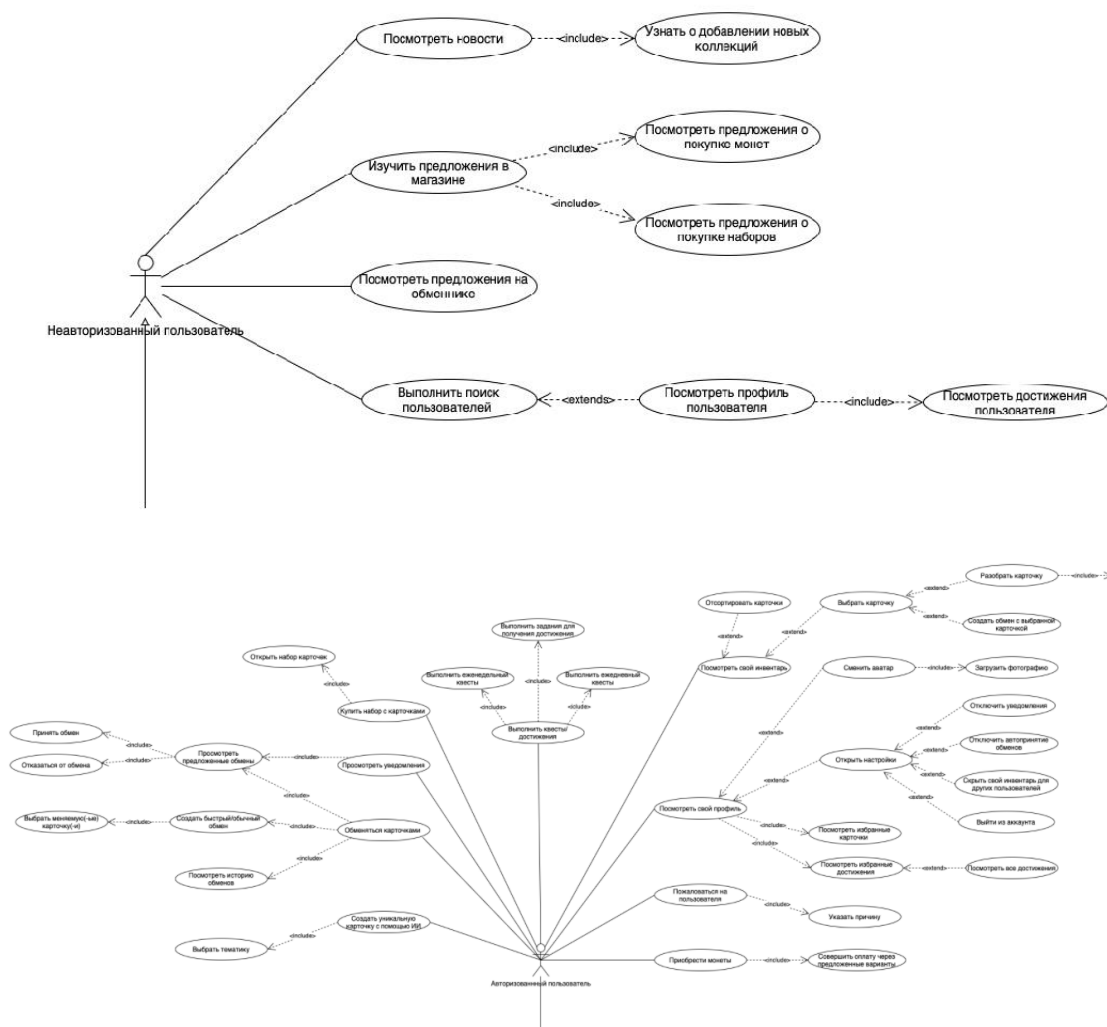


## 3 Моделирование системы

### 3.1 Диаграмма прецедентов (Use Case)

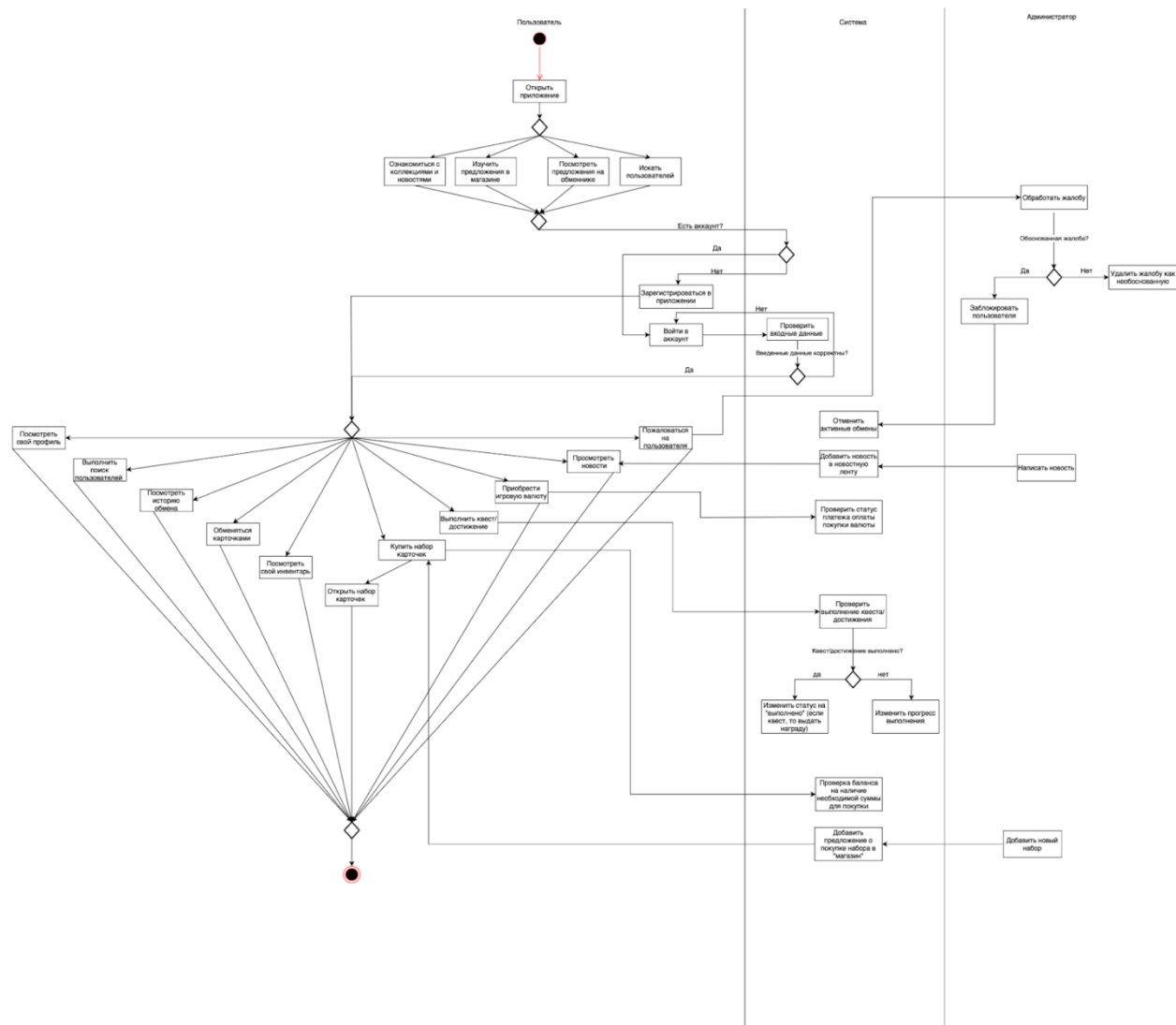
Диаграмма прецедентов – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

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





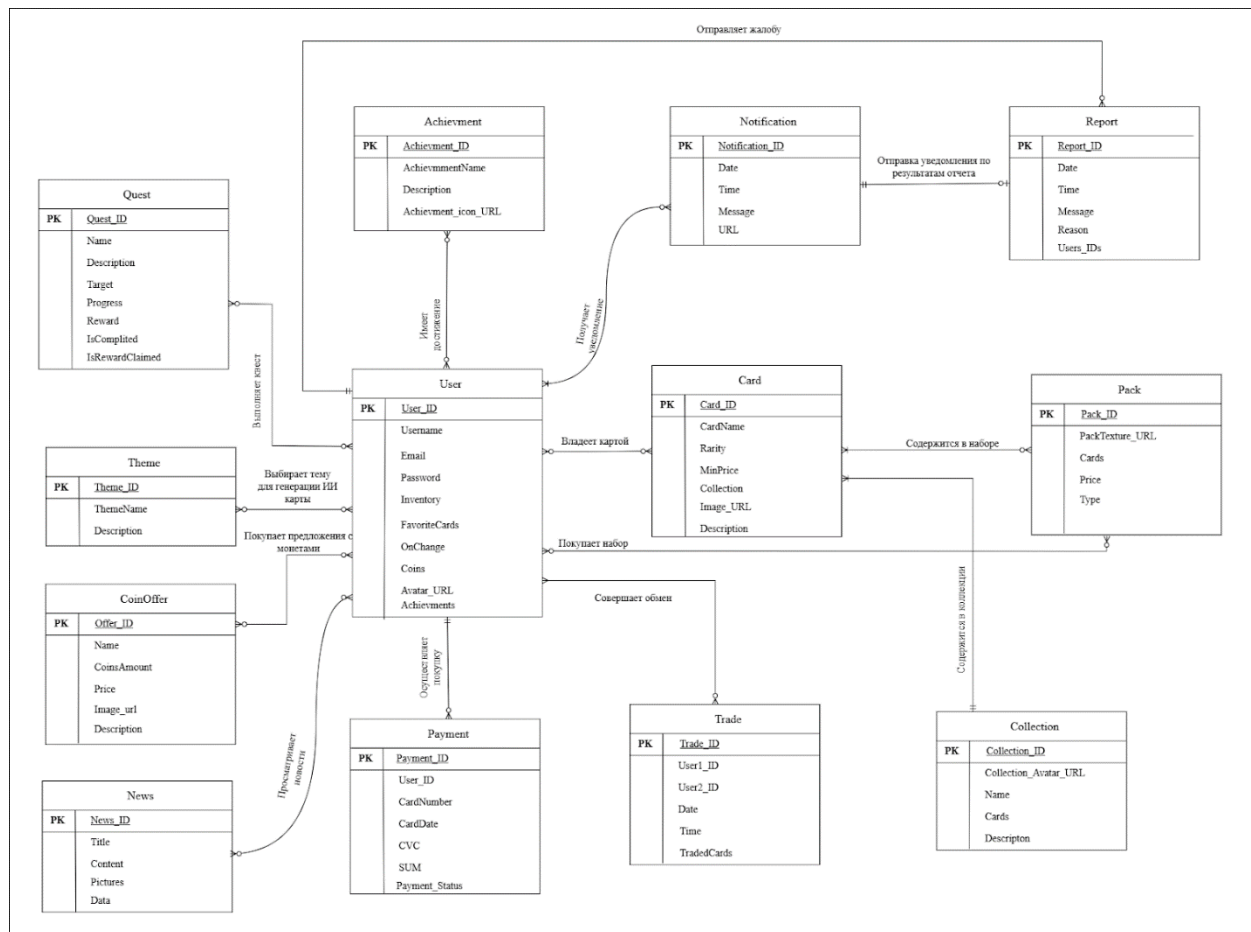




### 3.3 ER-диаграмма

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

На рисунке представлена ER-диаграмма для разрабатываемого приложения Cardly.

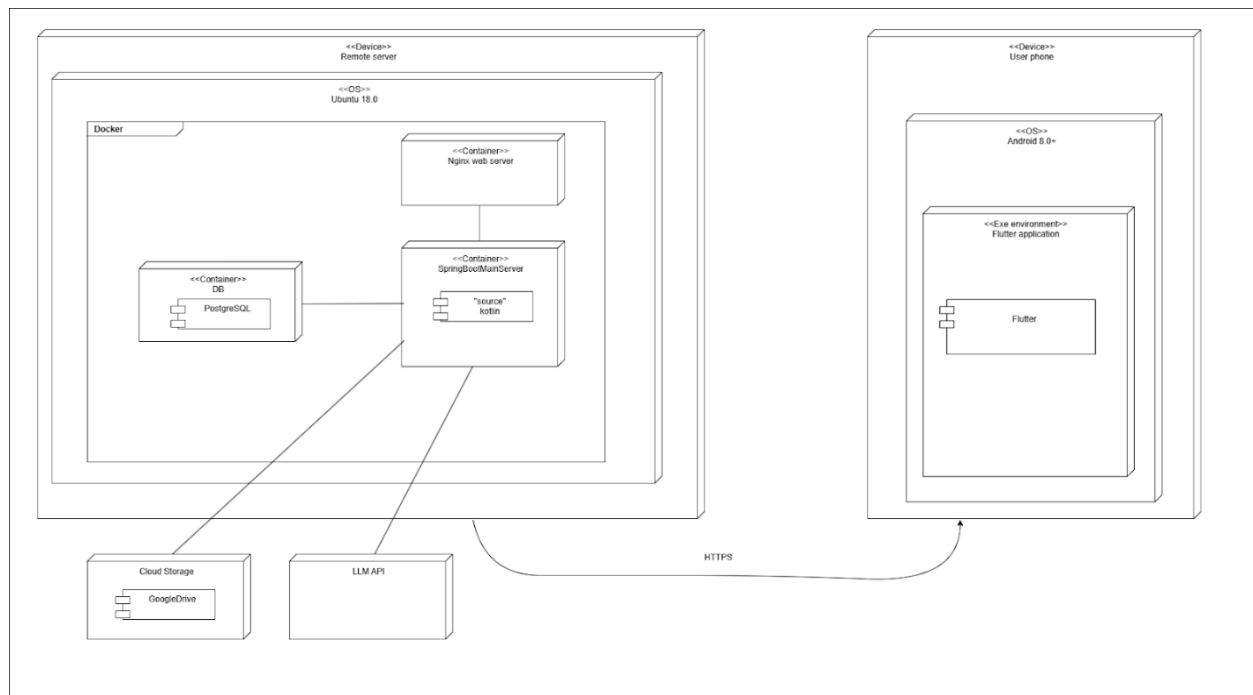


### 3.4 Диаграмма классов (Class diagram)

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.

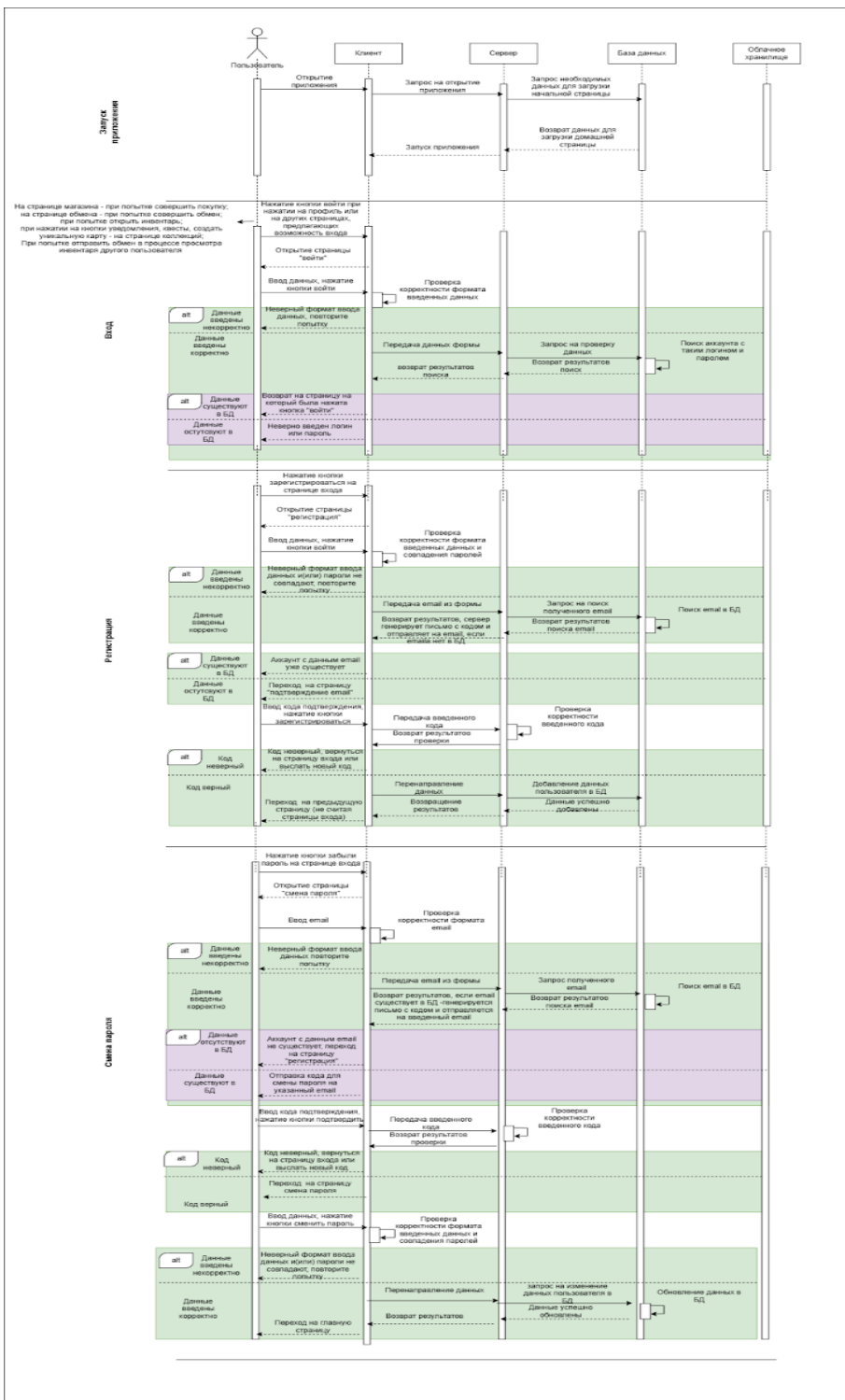


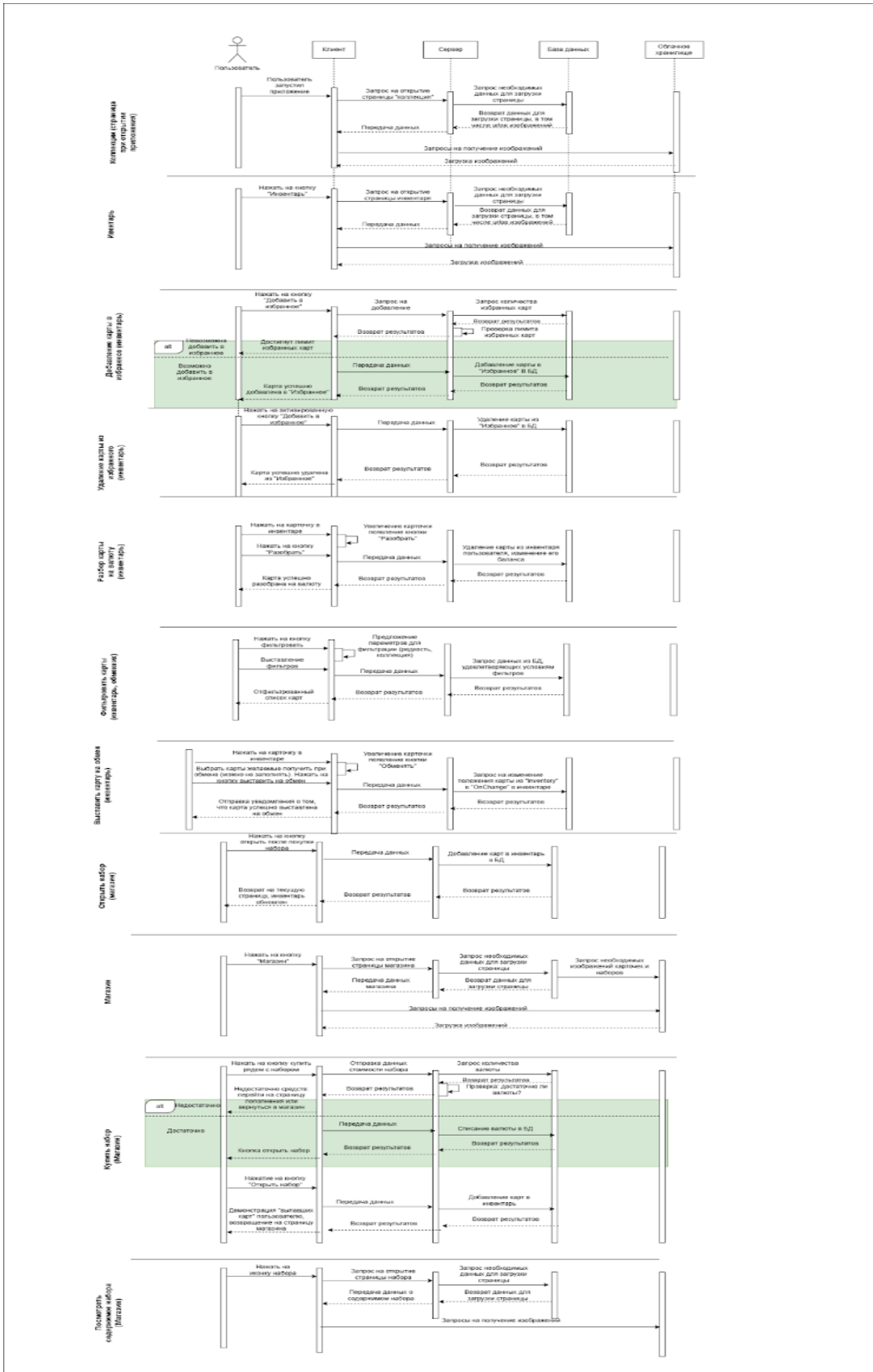


### **3.6 Диаграмма последовательности (Sequence diagram)**

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

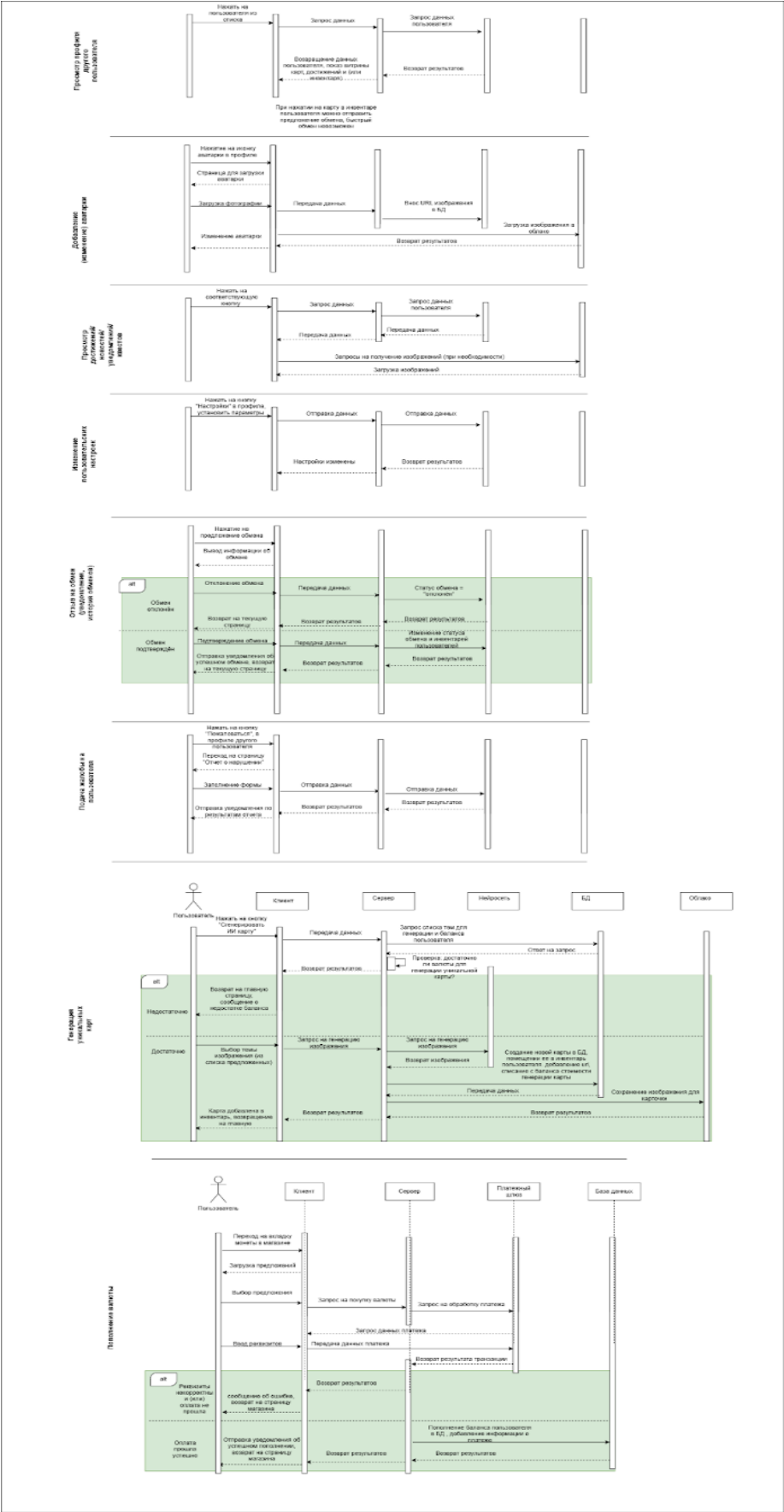
На рисунках представлены диаграммы последовательности для разрабатываемого приложения Cardly для авторизации, авторизованного пользователя (3 диаграммы), неавторизованного пользователя (2 диаграммы) и администратора (2 диаграммы) соответственно.

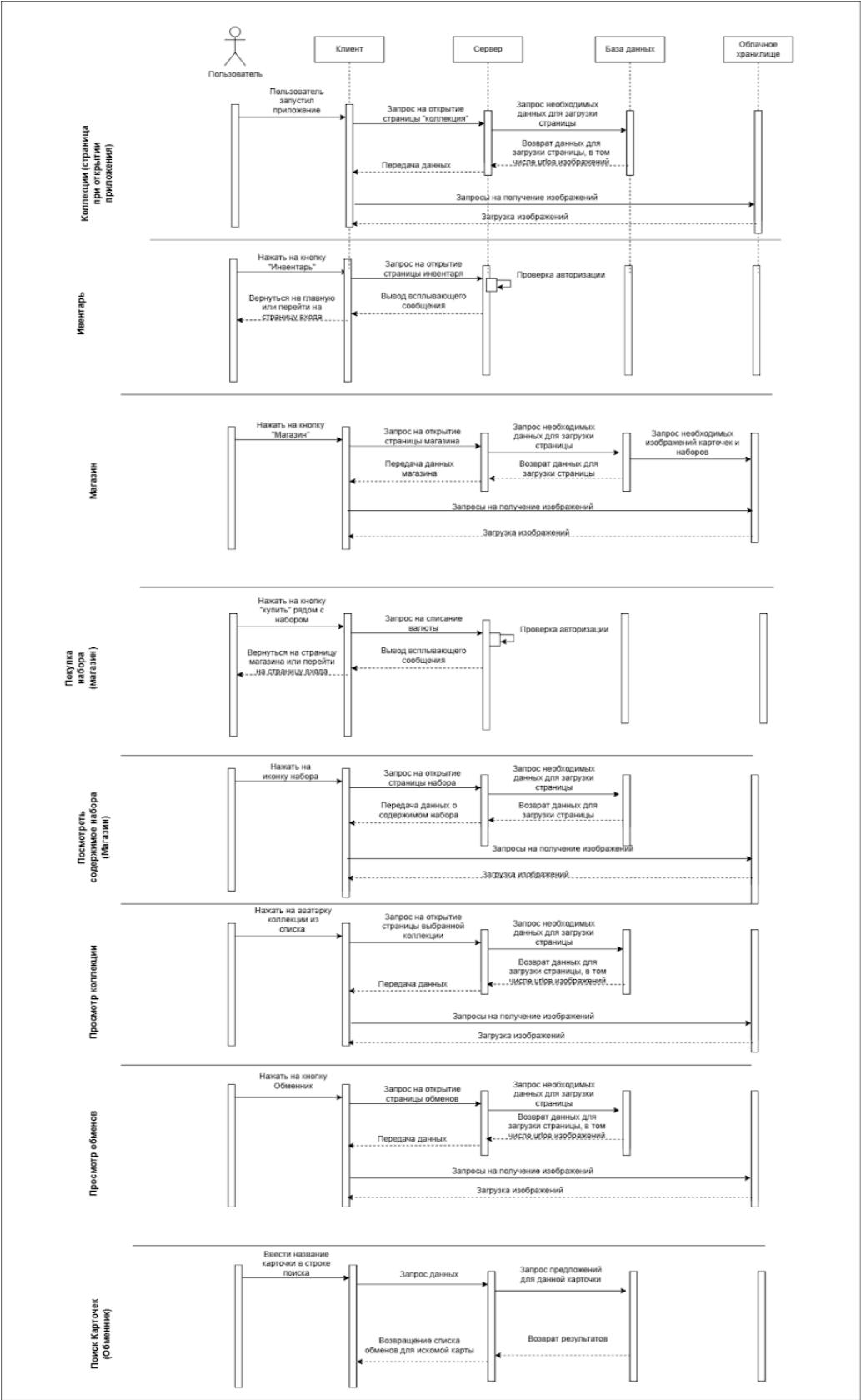


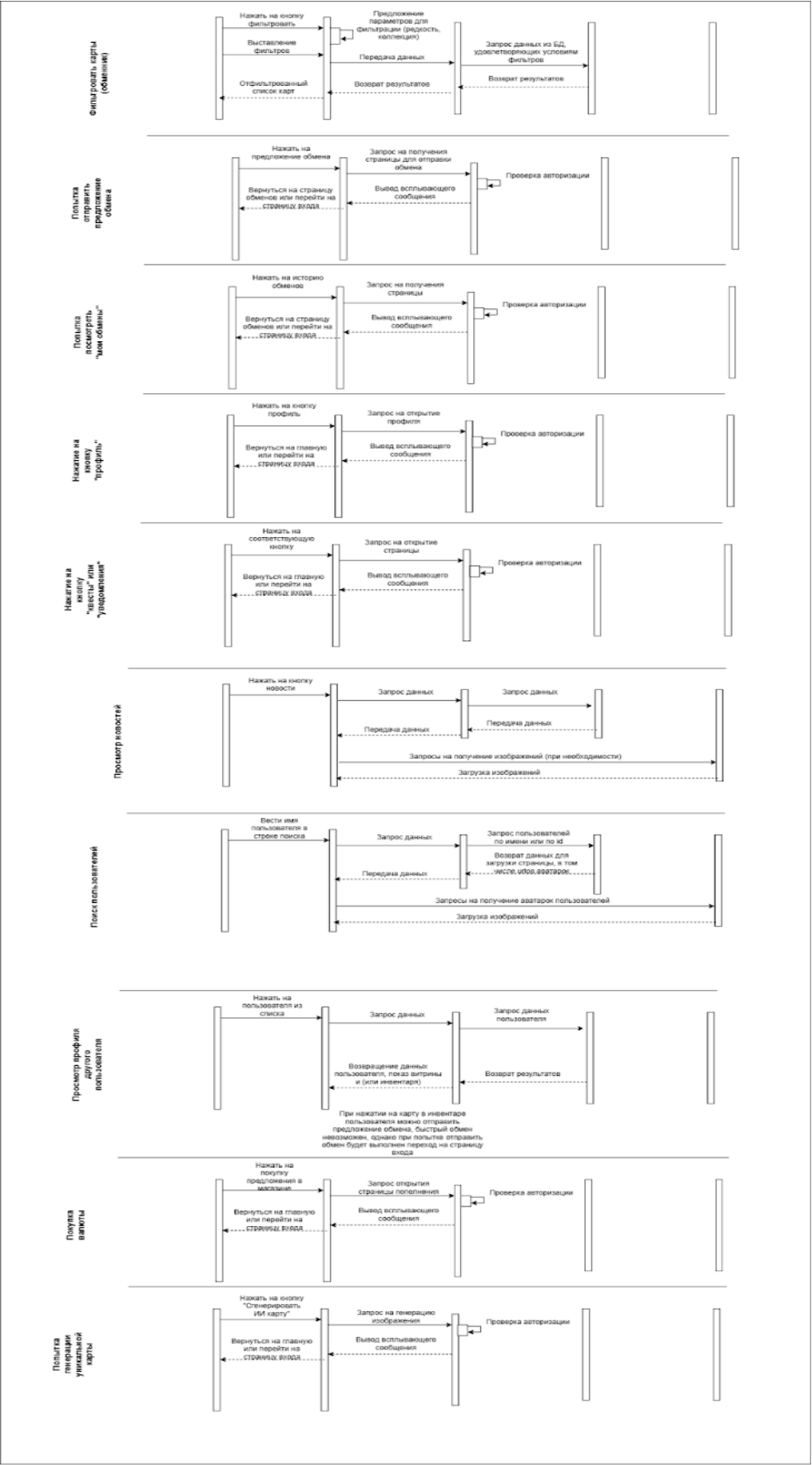


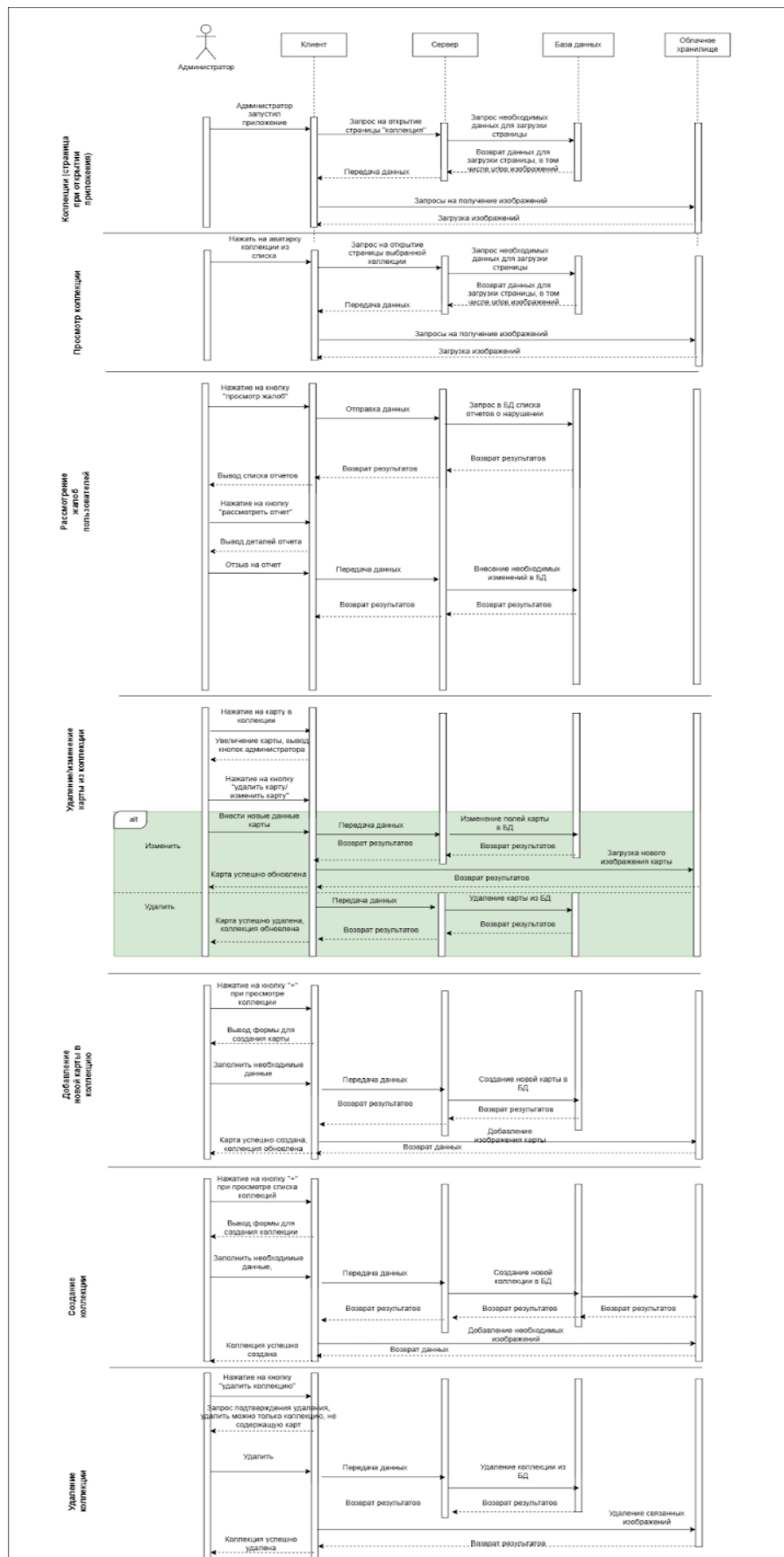


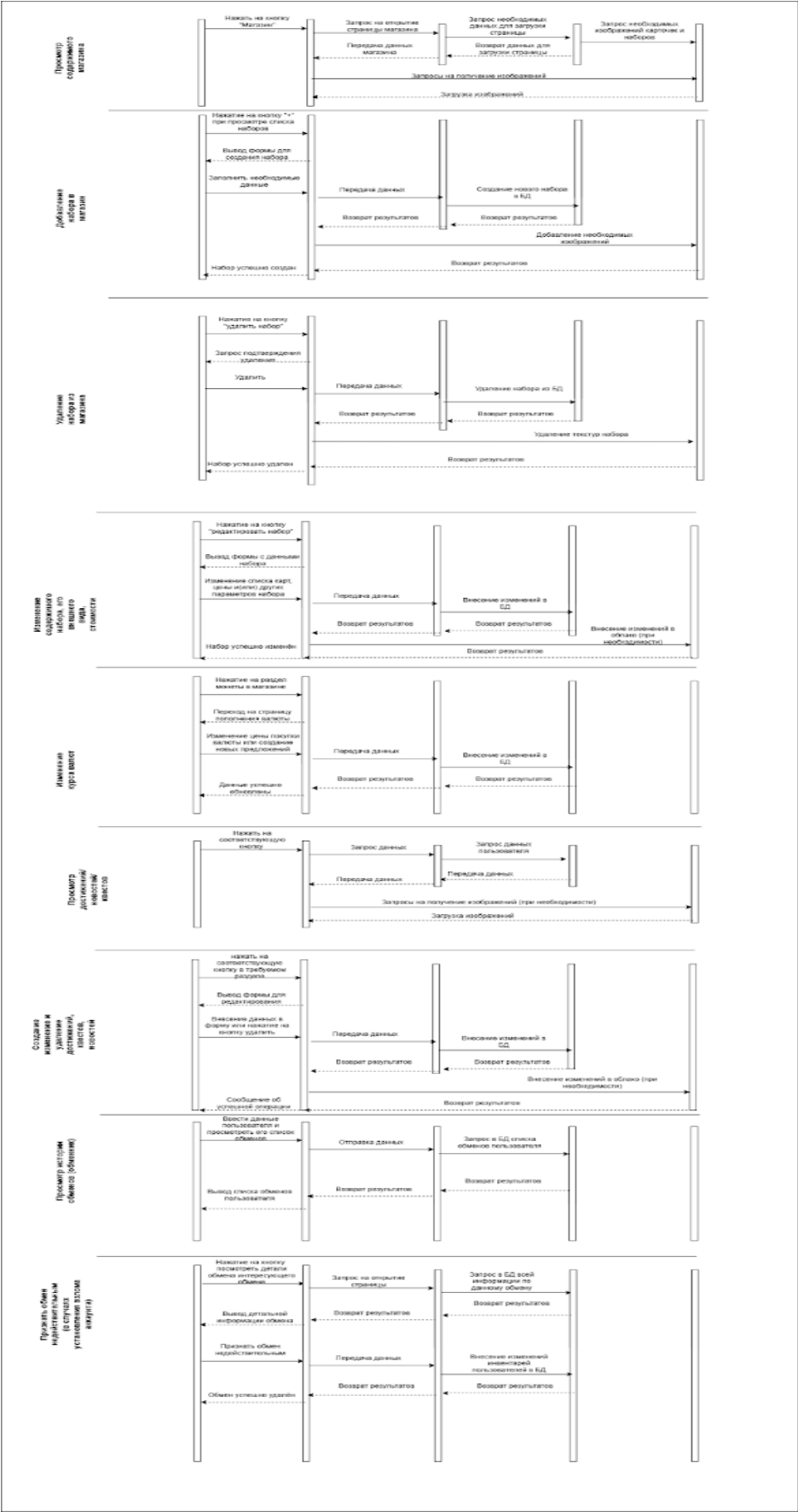












## 4 Реализация

### 4.1 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.

Контейнеризатор 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:

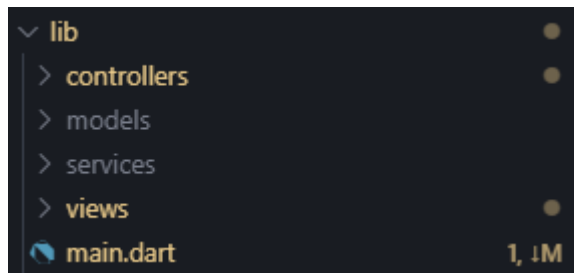
- Кроссплатформенность.
- Быстрота и лёгкость проектирования мобильных приложений.
- Понятная и полная документация.

## **4.2 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).

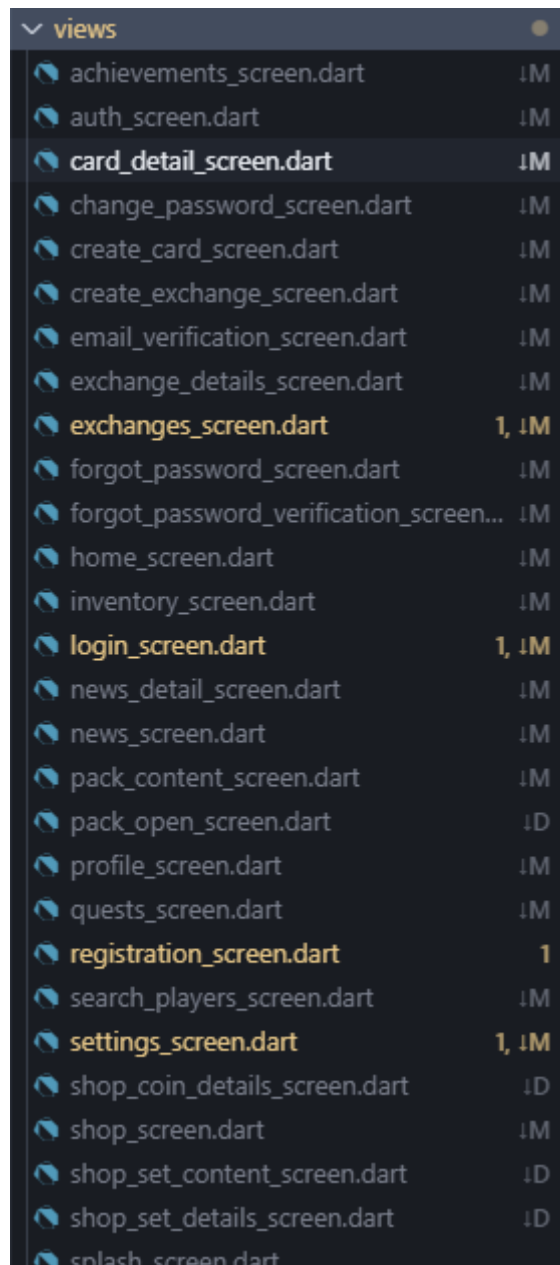


Здесь файл `main.dart`, содержащий функцию `main` – точка входа в приложение. В нём производится первоначальная конфигурация приложения, устанавливается базовый URL для запросов на сервер.

#### **4.2.1 Компонент представления (View)**

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



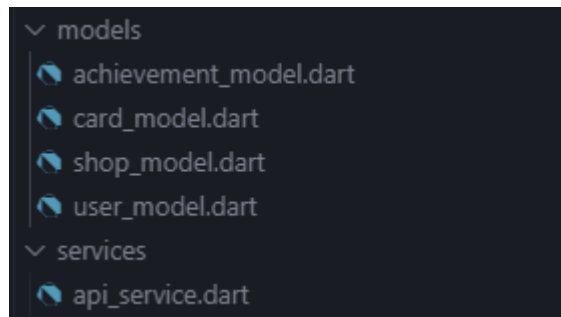


#### 4.2.2 Компонент данных (Model)

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.

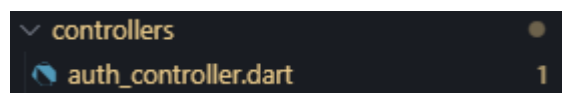
Слой `services` реализует работу с внешними API и сервисами. Здесь размещаются классы, отвечающие за отправку HTTP-запросов (GET, POST, PUT, DELETE) и обработку ответов от сервера. Такой подход позволяет централизовать сетевое взаимодействие и повторно использовать сервисы в различных частях приложения.



#### 4.2.3 Компонент модели представления (ViewModel)

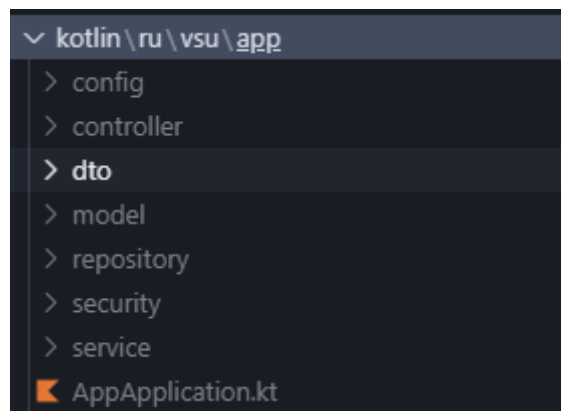
Слой `controllers` выполняет роль `ViewModel`. Здесь реализуются классы, которые управляют состоянием приложения, бизнес-логикой и взаимодействием между слоями данных и представления. Контроллеры наследуются от `ChangeNotifier` и используются совместно с `Provider` для реактивного обновления UI при изменении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



## 4.3 Архитектура серверной части

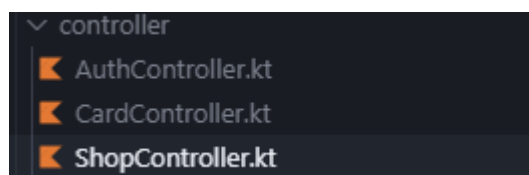
Серверная часть приложения реализована с использованием подхода многослойной архитектуры (Layered Architecture). Такой подход обеспечивает разделение ответственности между слоями. Файл `AppApplication.kt` содержит точку входа в приложение и отвечает за инициализацию Spring Boot-контекста, настройку компонентов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



### 4.3.1 Слой контроллеров (Controller)

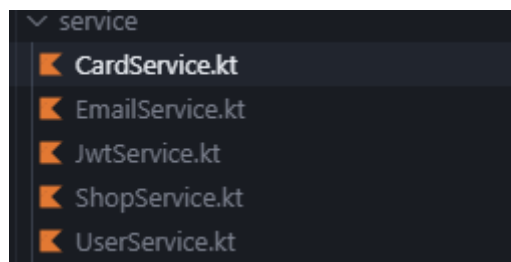
Пакет `controller` содержит классы-контроллеры, реализующие REST API приложения. Каждый контроллер отвечает за обработку HTTP-запросов, маршрутизацию и возврат ответов клиенту. Например, реализованы контроллеры для аутентификации (`AuthController.kt`), управления карточками (`CardController.kt`), работы с магазином (`ShopController.kt`).

Контроллеры принимают и валидируют входные данные, вызывают соответствующие сервисы и формируют ответы в виде DTO.



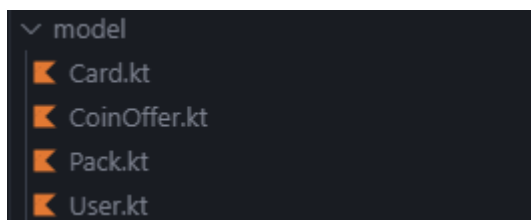
### 4.3.2 Слой сервисов (Service)

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 частях приложения.



### 4.3.3 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.



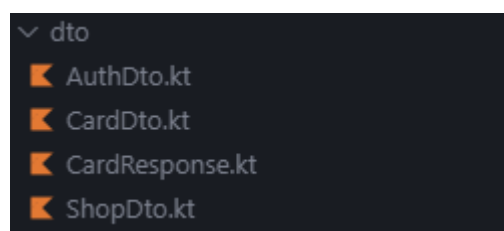
#### 4.3.4 Слой репозитория (Repository)

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.



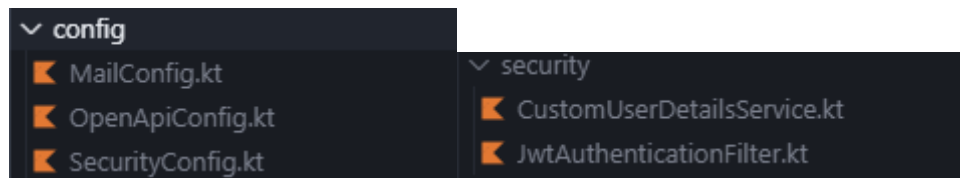
#### 4.3.5 Слой DTO (Data Transfer Object)

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.



#### 4.3.6 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.

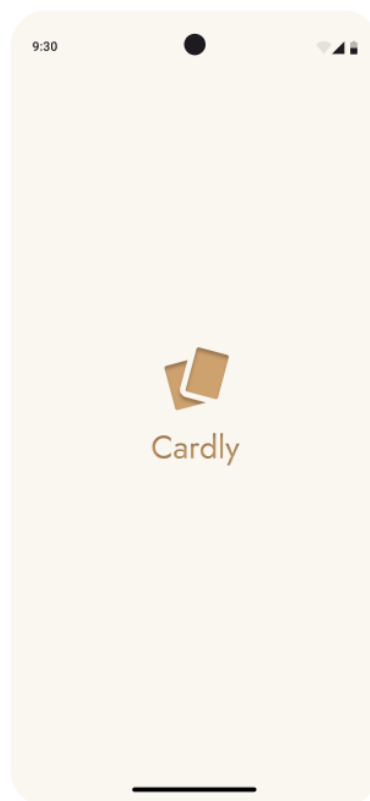


## 4.4 Реализация интерфейса

### 4.4.1 Экран загрузки

При каждом открытии мобильного приложения, пока не завершится полная инициализация клиентской части, пользователь видит загрузочный экран (см. рисунок 1).

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.



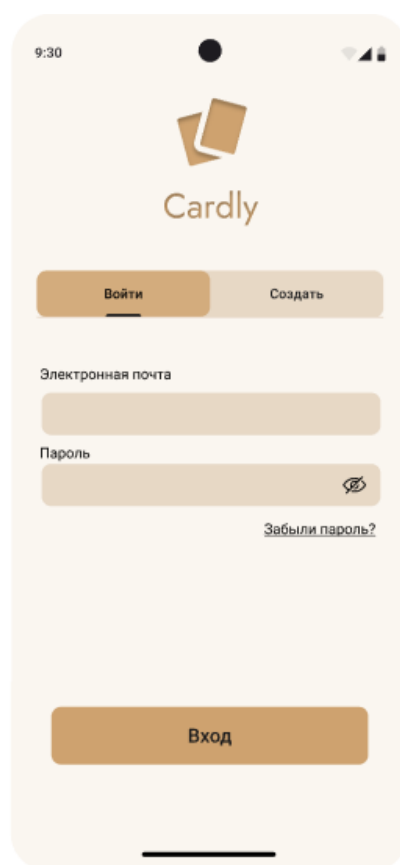
### 4.4.2 Экран авторизации/регистрации

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 2). Содержит следующие элементы:

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

После успешной авторизации пользователь перенаправляется на главный экран.



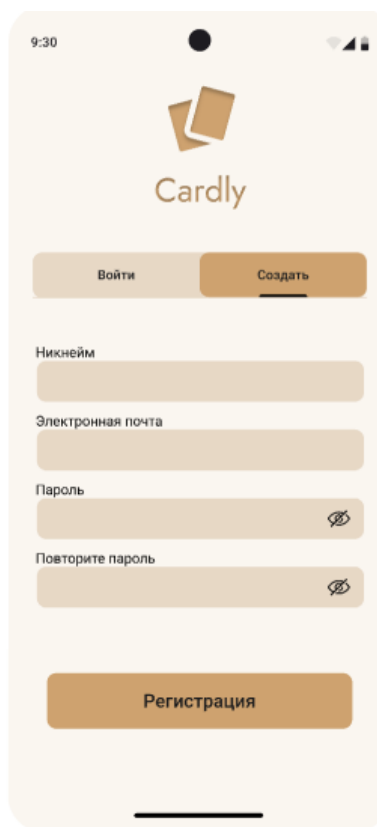
Вкладка «Регистрация» - предназначена для первичной регистрации новых пользователей (см. рисунок 3) содержит следующие элементы:

- форма для заполнения полей личными данными: никнейм,

электронная почта, пароль, повторение пароля.

- Переключатель «Вход» - «Создать».
- Кнопка «Регистрация».

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



#### 4.4.3 Экран «Главное меню»

На этот экран пользователь попадает после «экрана загрузки», соответственно он доступен как для авторизованного пользователя, так и для неавторизованного (см. рисунок 4). Он также доступен через нижнюю панель навигации. На экране имеются следующие 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.



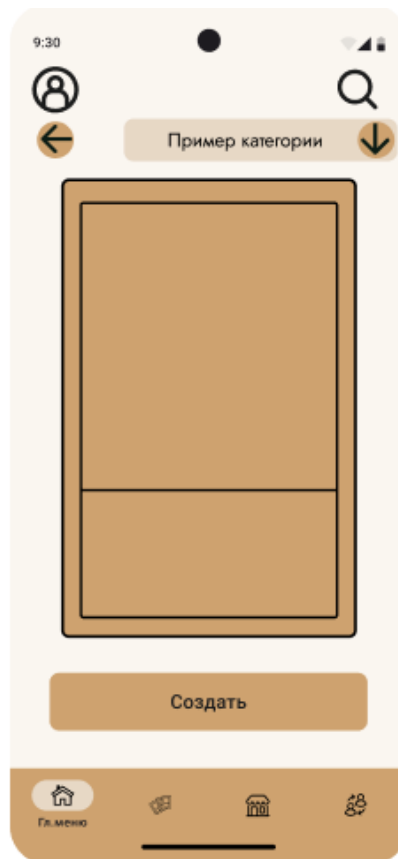
— Кнопки: создать свою уникальную карточку, квесты, новости. После нажатия на определенную кнопку пользователю открывается другой экран или новое окно с функционалом данного раздела.



#### 4.4.4 «Экран создания карточки»

Экран создания уникальной карточки с помощью искусственного интеллекта (см. рисунок 5) содержит следующие элементы:

- Верхняя панель: значки «Профиль» и «Поиск».
- 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».
- Панель навигации.



#### 4.4.5 Экран «Инвентарь»

На экране «Инвентарь» отображаются все имеющиеся карты пользователя (см. рисунок 6), 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.

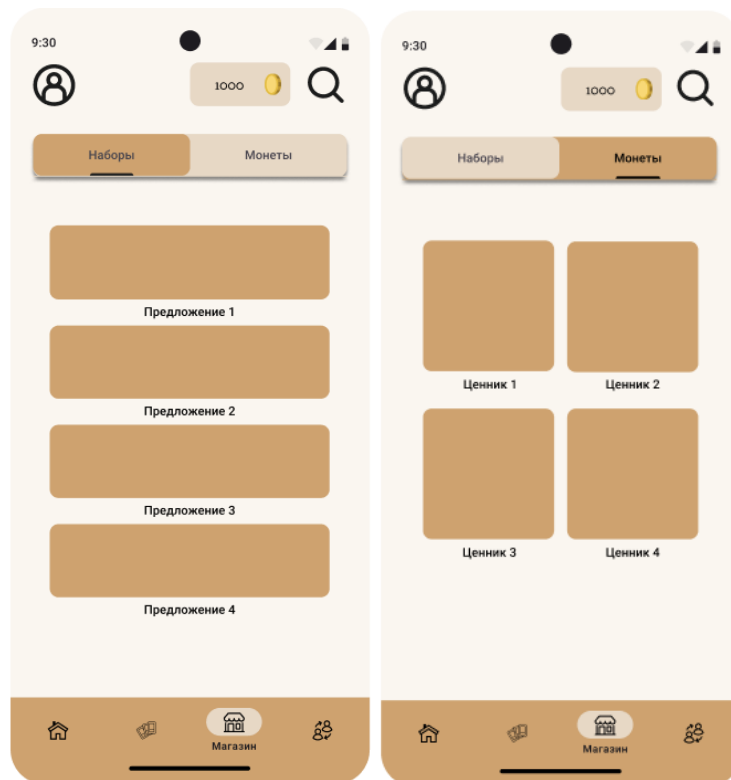


#### 4.4.6 Экран «Магазин»

Экран «Магазин» (см. рисунок 7) 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.



#### 4.4.7 Экран «Обменник»

Экран «Обменник» предназначен для обзора и взаимодействия с предложениями обмена от других пользователей (см. рисунок 8). 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.



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 (рисунок 9).

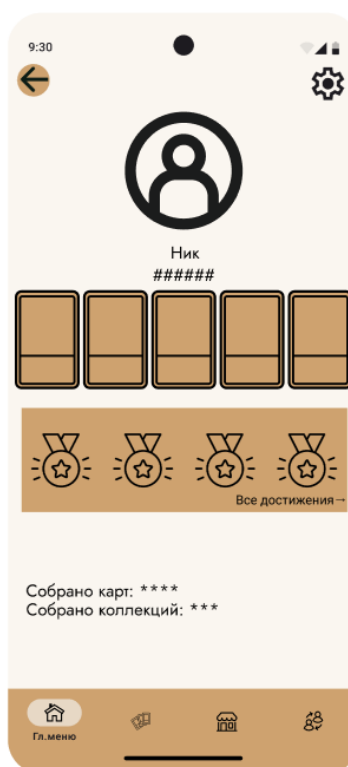
- Нажать на кнопку «Принять» - для подтверждения входящего предложения обмена или нажать на кнопку «Отклонить» для отклонения входящего предложения обмена (рисунок 10).



#### 4.4.8 Экран «Профиль»

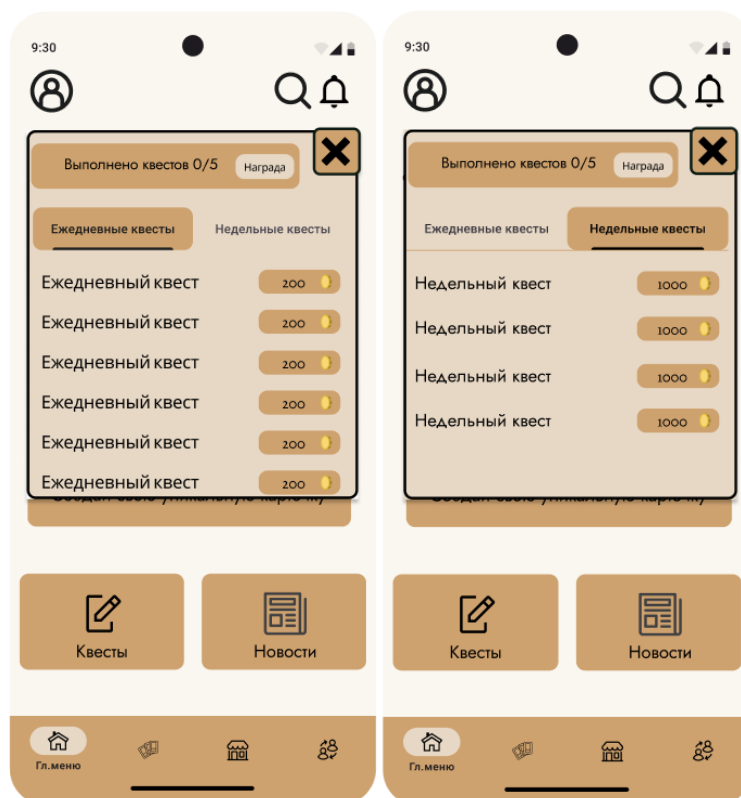
Настраиваемая страница профиля (см. рисунок 11) позволяет менять аватар, список избранных карт, список достижений, статистику собранных карт и коллекций. Содержит следующие элементы:

- 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



#### 4.4.9 Экран «Квесты»

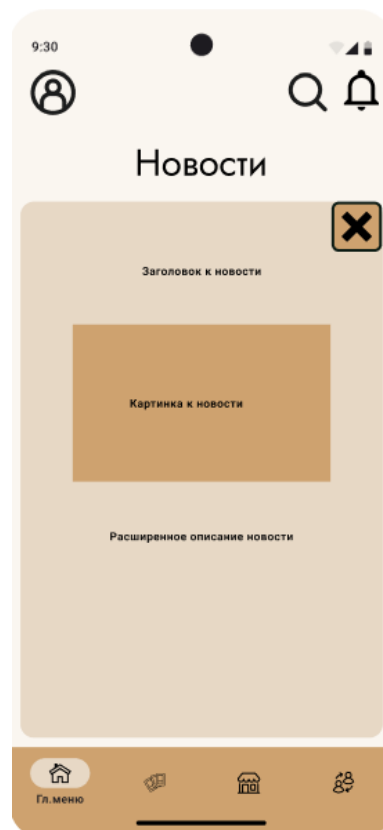
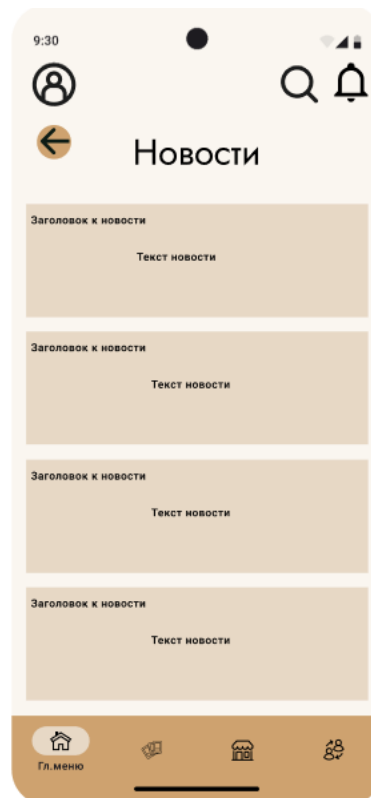
Всплывающий экран «Квесты» имеет два раздела: «Ежедневные квесты» и «Недельные» (см. рисунок 12). После выполнения заданий, пользователь может отследить свой прогресс в верхней части экрана и получить награду. Ниже представлен переключатель между разделами и список самих заданий.



#### 4.4.10 Экран «Новости»

Экран «Новости» представляет собой список актуальных новостей о вышедших обновлениях (см. рисунок 13), где каждое окно с новостью можно развернуть и просмотреть ее подробности (см. рисунок 14).





#### 4.4.11

#### 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 15) 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.



При отклике на второй вариант обмена (см. рисунок 16) 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.



#### 4.4.12 Экран «Создания обмена»

При создании обмена (см. рисунок 17) 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».

9:30



Ваша карта для обмена



Карты на которые вы готовы  
произвести обмен



Создать обмен



Обменник