МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет компьютерных наук

Кафедра информационных систем

Мобильное приложение для обмена кулинарными рецептами "YumYard"

Курсовая работа

09.03.02 Информационные системы и технологии

Информационные системы и сетевые технологии

6 семестр 2022/2023 учебного года

Зав. кафедрой _____ к. т. н., доцент Д.Н. Борисов

Обучающийся _____ ст. 3 курса оч. отд. А. С. Пальчикова

Обучающийся _____ ст. 3 курса оч. отд. В. Д. Симонов

Обучающийся _____ ст. 3 курса оч. отд. Е. А. Терёшкин

Руководитель _____ ст. преп. В.С. Тарасов

Воронеж 2023

Содержание

1 Введение

1.1 Обзор сферы мобильных приложений для кулинарии.

Мобильные приложения для кулинарии стали значительной частью повседневной жизни многих пользователей. Они не только помогают в поиске новых рецептов, но и способствуют обучению кулинарным навыкам и улучшению пищевых привычек. Рынок мобильных приложений для кулинарии активно развивается, предлагая широкий спектр функциональностей, от простых сборников рецептов до комплексных помощников в планировании питания и покупок продуктов.

1.2 Значение и актуальность приложений для обмена рецептами.

Приложения для обмена кулинарными рецептами обладают большим социальным и культурным значением. Они способствуют обмену кулинарными традициями и инновациями между пользователями из различных уголков мира, тем самым обогащая пищевую культуру и расширяя кулинарные горизонты. В условиях глобализации и интереса к здоровому образу жизни, актуальность таких приложений только увеличивается, поскольку они предоставляют инструменты для разнообразия диеты и внедрения полезных пищевых привычек.

1.3 Цели и задачи работы.

Основная цель данной курсовой работы - разработка мобильного приложения "YumYard", которое предоставляет платформу для обмена кулинарными рецептами. Это приложение должно стать удобным инструментом для пользователей, заинтересованных в кулинарии, позволяя им не только делиться своими рецептами, но и находить новые, а также комментировать и оценивать рецепты других пользователей.

Для достижения этой цели были поставлены следующие задачи:

— Исследование существующего рынка мобильных приложений для кулинарии и анализ конкурентов.

— Определение функциональных требований и технических спецификаций на основе потребностей целевой аудитории.

— Проектирование архитектуры и интерфейса пользователя, обеспечивающих удобство использования и эстетическую привлекательность.

— Реализация основных функций приложения, таких как регистрация, добавление и редактирование рецептов, их оценка и комментирование.

— Обеспечение безопасности пользовательских данных и защита от несанкционированного доступа.

2 Разработка пользовательского интерфейса

2.1 Дизайн интерфейса

В разработке мобильного приложения "YumYard" особое внимание уделяется дизайну интерфейса, который является ключевым фактором в обеспечении удобства и привлекательности приложения для пользователя. Процесс создания дизайна включает несколько этапов:

— **Исследование пользователей и сбор требований:** начальный этап включает изучение целевой аудитории и сбор основных требований к функциональности и внешнему виду приложения. Это помогает определить, какие элементы интерфейса наиболее важны и как они должны быть организованы для обеспечения наилучшего пользовательского опыта.

— **Скетчинг и прототипирование:** на этом этапе дизайнеры создают первичные наброски интерфейса и разрабатывают прототипы. Прототипы могут быть как низкоуровневыми (бумажными или цифровыми макетами), так и высокоуровневыми интерактивными моделями, которые позволяют оценить интерактивность и поток пользовательских задач.

— **Разработка макетов:** после утверждения прототипов создаются детализированные макеты интерфейса с использованием инструментов дизайна, таких как Adobe XD, Sketch или Figma. В этом процессе уточняются цветовая схема, шрифты и расположение элементов управления.

— **Адаптивный дизайн:** учитывается необходимость адаптации интерфейса под различные размеры экранов и ориентации устройств. Дизайн должен обеспечивать эффективное

отображение на широком спектре устройств, от смартфонов до планшетов.

2.2 UX/UI тестирование

Тестирование пользовательского интерфейса (UI) и пользовательского опыта (UX) критически важно для определения соответствия разработанного продукта ожиданиям и потребностям конечных пользователей. Тестирование проводится на нескольких этапах разработки:

- **Юзабилити-тестирование:** оценивает удобство и интуитивность интерфейса. Тесты могут проводиться в лабораторных условиях с участием реальных пользователей. Пользователи выполняют типичные задачи, позволяя идентифицировать любые проблемы с навигацией, доступностью функций или пониманием контента.

- **A/B тестирование:** используется для сравнения двух версий страниц или интерфейсов, чтобы определить, какая из них лучше с точки зрения показателей вовлеченности и конверсии.

- **Тестирование на соответствие стандартам доступности:** обеспечивает, чтобы приложение было доступно для пользователей с ограниченными возможностями, включая проверку контрастности цветов, поддержку экранных читалок и навигацию с клавиатуры.

Результаты этих тестов влияют на итерации дизайна, помогая дизайнерам и разработчикам улучшить продукт перед его окончательным запуском.

3 Обзор предметной области

3.1 История развития и состояние рынка кулинарных приложений.

Рынок мобильных приложений для кулинарии активно развивается уже более десяти лет. Первые приложения были простыми сборниками рецептов, позволяющими пользователям только просматривать и читать кулинарные рецепты. Со временем, по мере развития технологий и увеличения запросов пользователей, функциональность приложений расширилась. Современные приложения предлагают не только базы данных рецептов, но и интеграцию с социальными сетями, возможности планирования покупок, интеграцию с умными кухонными устройствами, а также личные рекомендации на основе диетических предпочтений и здоровья пользователя.

3.2 Анализ аналогов "YumYard" и их функциональности

На рынке существует несколько популярных приложений, которые можно считать аналогами "YumYard", например, "Allrecipes", "Yummly" и "Epicurious". Эти приложения предлагают пользователям возможность просматривать рецепты, добавлять их в избранное, а также делиться своими собственными рецептами. Однако "YumYard" отличается интеграцией с социальными функциями, такими как оценки, комментарии и обширные возможности по настройке пользовательского интерфейса. К тому же, "YumYard" планируется оснастить продвинутыми инструментами анализа предпочтений пользователей, что позволит предлагать им индивидуализированные рецепты.

3.3 Преимущества и недостатки существующих решений

Одним из основных преимуществ существующих приложений является их способность предоставлять пользователю обширный выбор рецептов с детализированными инструкциями и питательной информацией. Многие из них также интегрированы с функциями планирования покупок, что позволяет легко создавать список покупок на основе выбранных рецептов. Однако,

существующие решения часто страдают от перегруженности интерфейса и недостаточной персонализации, что может отталкивать пользователей, желающих более гибкой и персонализированной функциональности. К тому же, многие приложения не обладают достаточными мерами защиты личных данных, что является важным аспектом в свете растущих требований к конфиденциальности.

"YumYard" стремится устранить эти недостатки, предлагая удобный, настраиваемый интерфейс и улучшенные меры защиты персональных данных, а также более глубокую интеграцию с социальными функциями для обеспечения дополнительной стоимости и удовлетворения потребностей активных пользователей социальных кулинарных сообществ.

4 Техническое задание

4.1 Обзор общих требований к приложению и его функциональных характеристик.

Согласно техническому заданию, мобильное приложение "YumYard" должно предоставлять пользователям платформу для обмена кулинарными рецептами. Приложение позволит пользователям хранить, добавлять, просматривать, редактировать рецепты, а также комментировать и оценивать их. Оно также должно поддерживать функции регистрации и входа для различения между анонимными и зарегистрированными пользователями.

Основные функции приложения включают:

- **Регистрация и аутентификация:** позволяет пользователям создавать учетные записи и входить в систему для доступа к персональным и дополнительным функциям.
- **Добавление и управление рецептами:** пользователи могут добавлять свои рецепты, включая текстовые описания и фотографии, а также редактировать и удалять их.
- **Поиск и фильтрация:** возможность поиска рецептов по различным параметрам, таким как ингредиенты или категории.
- **Интерактивные функции:** возможность оценивать рецепты, добавлять их в избранное и комментировать.
- **Личный профиль:** просмотр и редактирование личной информации и управление своими рецептами.

4.2 Описание архитектуры и использование технологий (Django, React Native, REST API).

Приложение "YumYard" будет использовать модернизированную клиент-серверную архитектуру с использованием следующих технологий:

Backend (серверная часть):

— **Django:** выбор Django как backend-фреймворка обусловлен его мощными возможностями для быстрой разработки надежных веб-приложений. Django предоставляет встроенную административную панель, что упрощает управление содержимым и пользователями.

— **Django REST Framework:** для создания API будет использоваться Django REST Framework, который облегчает создание веб-API и обеспечивает гибкость в обработке запросов и ответов.

Frontend (клиентская часть):

— **React Native:** использование React Native позволит разрабатывать нативные мобильные приложения для iOS и Android с использованием JavaScript, что существенно ускоряет разработку и поддержку приложений на обеих платформах.

— **Взаимодействие с сервером:** коммуникация между клиентом и сервером будет осуществляться через REST API, что позволяет поддерживать стандарты и упрощает интеграцию с другими системами и сервисами.

Безопасность и защита данных:

— **Аутентификация и авторизация:** безопасность будет обеспечиваться с помощью системы аутентификации и авторизации, которые контролируют доступ к функционалу приложения.

— **Защита данных:** приложение будет использовать шифрование данных и безопасное хранение чувствительной информации, включая данные пользователей и рецепты. Django предоставляет мощные инструменты для защиты от CSRF и XSS атак, а также поддерживает безопасную работу с сессиями и cookies.

В целом, выбор этих технологий направлен на создание надежного, безопасного и легко масштабируемого приложения, которое может быть адаптировано к изменяющимся требованиям и условиям рынка.

4.2.1 Технологический обзор: Django, React Native, REST API

Django

— **Архитектура: Django** — это высокоуровневый веб-фреймворк на Python, который следует архитектурному паттерну "Model-View-Template" (MVT). Это позволяет разработчикам быстро создавать безопасные и поддерживаемые веб-приложения, разделяя логику приложения на отдельные компоненты.

— **Принципы работы:** Django управляет всем циклом запроса-ответа: обработкой запросов, взаимодействием с базой данных через ORM для моделей, выполнением бизнес-логики в представлениях, и формированием ответов с помощью шаблонов

React Native

— **Архитектура:** React Native позволяет разработчикам использовать React — библиотеку JavaScript для создания пользовательских интерфейсов — для разработки нативных мобильных приложений для iOS и Android. Код JavaScript выполняется в отдельном потоке, обеспечивая взаимодействие с нативным API через асинхронный мост.

— **Принципы работы:** React Native использует компонентный подход, что позволяет создавать переиспользуемые блоки интерфейса, что улучшает эффективность разработки и поддерживаемость кода.

REST API

— **Архитектура:** REST (Representational State Transfer) — это стиль архитектуры программного обеспечения для систем, основанных на сети, включая веб. RESTful API позволяет взаимодействовать с веб-ресурсами через HTTP с использованием методов, таких как GET, POST, PUT и DELETE.

— **Принципы работы:** REST API взаимодействует с веб-приложениями, отправляя запросы на сервер и получая ответы. Эти API должны быть без состояний, что означает, что каждый запрос должен содержать всю необходимую информацию для его обработки.

4.2.2 Безопасность мобильных приложений

Шифрование

— **Описание:** шифрование — это процесс кодирования информации таким образом, что только уполномоченные стороны могут ее дешифровать. В мобильных приложениях данные пользователя, отправляемые на сервер или хранимые локально, должны быть зашифрованы для предотвращения несанкционированного доступа.

— **Технологии:** используются такие стандарты, как AES (Advanced Encryption Standard) для шифрования данных и SSL/TLS для защиты данных, передаваемых по сети.

Аутентификация и авторизация

— **Описание:** аутентификация проверяет, является ли пользователь тем, за кого он себя выдает, в то время как авторизация определяет, к каким ресурсам у пользователя есть доступ.

— **Технологии:** используется простая схема HTTP-аутентификации на основе токенов, для управления сессиями пользователей после их аутентификации.

Защита от уязвимостей

— **Описание:** мобильные приложения подвержены различным уязвимостям, таким как SQL-инъекции, XSS (межсайтовый скриптинг), и межсайтовая подделка запросов (CSRF).

— **Меры предосторожности:** регулярное обновление зависимостей, использование проверенных библиотек, проведение пентестирования и код-ревью, а также обучение разработчиков основам безопасности.

5 Функционал приложения

5.1 Описание основных функций приложения: регистрация, добавление и редактирование рецептов, комментирование, оценки.

Регистрация и аутентификация: пользователи могут создавать свои учетные записи в приложении "YumYard", предоставляя базовую информацию, такую как имя пользователя, электронная почта и пароль. Аутентификация будет обеспечиваться через стандартные механизмы входа с подтверждением по электронной почте для усиления безопасности учетных записей.

Добавление и редактирование рецептов: зарегистрированные пользователи могут добавлять новые рецепты, включая описание, ингредиенты, шаги приготовления и фотографии. Пользователи также могут редактировать и удалять свои рецепты. Для удобства, рецепты могут быть сохранены в черновиках до публикации.

Комментирование и оценки: приложение позволяет пользователям оставлять комментарии к рецептам и оценивать их по пятибалльной шкале. Эти социальные функции способствуют взаимодействию внутри сообщества и помогают другим пользователям выбирать рецепты на основе отзывов и рейтингов.

5.2 Описание интерфейса пользователя и навигации по приложению

Главная страница: отображает популярные и новые рецепты, а также предоставляет доступ к функции поиска. Главная страница служит входной точкой для исследования контента приложения.

Меню навигации (Bottom menu): включает в себя элементы для быстрого доступа к главной странице, поиску рецептов, избранному и личному профилю. Для неавторизованных пользователей предоставляется ограниченный доступ с предложением зарегистрироваться или войти.

Страница рецепта: показывает полную информацию о рецепте, включая ингредиенты, шаги приготовления, фотографии, комментарии и рейтинги. Также предоставляются кнопки для добавления в избранное и публикации комментариев.

5.3 Описание мер по обеспечению безопасности и защиты данных

Защита данных: все данные пользователей и рецепты хранятся в защищенной базе данных. Доступ к данным строго регулируется и ограничен соответствующими правами доступа.

Аутентификация и сессии: приложение использует защищенные методы аутентификации и управления сессиями. Сессии пользователя защищены токенами, которые предотвращают перехват и подделку сессий.

Эти меры по обеспечению безопасности и защиты данных позволяют пользователю чувствовать себя уверенно, используя приложение для обмена кулинарными рецептами, и способствуют созданию доверительной и безопасной среды для всех пользователей.

6 Реализация

6.1 Frontend разработка

6.1.1 Импорт всего необходимого для навигации и экранов в приложении

— React хуки: useEffect, useState

— React Navigation: NavigationContainer, useNavigationContainerRef, createStackNavigator

— Типы: TypeRootStackParamList

— Экраны и навигаторы: PrivateNavigator, Auth, Start, Settings, Splash, Account

— Кастомный хук: useAuth

Пример кода, использованный в разработке:

```
import { useEffect, useState } from 'react'
import {
    NavigationContainer,
    useNavigationContainerRef
} from '@react-navigation/native'
import { PrivateNavigator } from
'./PrivateNavigator'
import { createStackNavigator } from '@react-
navigation/stack'
```

```
    import { TypeRootStackParamList } from
'./navigation.types'

    import { Auth, Start } from 'screens/index'

    import { useAuth } from 'hooks/useAuth'

    import { Settings } from
'screens/Settings/Settings'

    import { Splash } from 'screens/Splash/Splash'

    import { Account } from 'screens/Account/Account'
```

6.1.2 Отслеживание текущего маршрута и обновление состояний при изменениях

— Аутентификация: const { isAuth } = useAuth()

— Состояние и навигация: const [_, setCurrentRoute] = useState(undefined) и const navRef = useNavigationContainerRef()

— Обновление текущего маршрута

Пример кода, использованный в разработке:

```
    export const Navigation = () => {

        const { isAuth } = useAuth()

        const [_, setCurrentRoute] = useState<string |
undefined>(undefined)

        const Stack =
createStackNavigator<TypeRootStackParamList>()
```

```
        const navRef = useNavigationContainerRef()

        useEffect(() => {

        setCurrentRoute(navRef.getCurrentRoute()?.name)

            const listener =
navRef.addListener('state', () =>

        setCurrentRoute(navRef.getCurrentRoute()?.name)

            )

            return () => {

                navRef.removeListener('state',
listener)

            }

        }, [navRef])
```

6.1.3 Отображение экранов в зависимости от статуса авторизации пользователя

В этом примере кода в зависимости от статуса авторизации пользователя (isAuth), решается какие экраны будут отображены: если пользователь не авторизован, то отображаются экраны Splash, Start и Auth, а если авторизован - экраны PrivateNavigator, Settings и Account. Каждый экран определен как Stack.Screen внутри StackNavigator, который управляет стеком экранов.

```
    return (

            <NavigationContainer ref={navRef}>

                <Stack.Navigator
```

```
                    screenOptions={{ headerShown:
false }}
                    initialRouteName='Splash'
                >
                    {!isAuth ? (
                        <>
                            <Stack.Screen
name='Splash' component={Splash} />
                            <Stack.Screen name='Start'
component={Start} />
                            <Stack.Screen name='Auth'
component={Auth} />
                        </>
                    ) : (
                        <>
                            <Stack.Screen
    name='PrivateNavigator'
    component={PrivateNavigator}
                            />
```

```
                                <Stack.Screen
name='Settings' component={Settings} />



                                <Stack.Screen
name='Account' component={Account} />

                            </>

                        ) }

                    </Stack.Navigator>

                </NavigationContainer>

            )

        }
```

6.2 Backend разработка

6.2.1 Основные модели в базе данных

Основными моделями в базе данный являются:

— User;

— UserProfile;

— Recipe;

— Comment;

— Category.

Данный код описывает модель профиля пользователя:

```
    class UserProfile(models.Model):

        user = models.OneToOneField(User,
on_delete=models.CASCADE, related_name='profile')

        avatar =
models.ImageField(upload_to='avatars/', null=True,
blank=True)

        followers = models.ManyToManyField(User,
related_name='following', blank=True)

        def __str__(self):

            return self.user.username

        @property

        def followers_count(self):

            return self.followers.count()
```

6.2.2 Адрес Backend

По данным URL Frontend отправляет запросы и тем самым связывается с Backend.

Пример кода:

```
    urlpatterns = [

        path('v1/users/', views.UserAPIList.as_view()),

        path('v1/users/<int:pk>/',
views.UserAPIDetail.as_view()),
```

```
        path('v1/recipe/',
views.RecipeAPIList.as_view()),

        path('v1/recipe/<int:pk>',
views.RecipeAPIUpdate.as_view()),

        path('v1/recipedelete/<int:pk>',
views.RecipeAPIDelete.as_view()),

        path('profile/<str:username>/follow/',
views.FollowUserView.as_view(), name='follow-user'),

        path('profile/<str:username>/unfollow/',
views.UnfollowUserView.as_view(), name='unfollow-
user'),

        path('profile/<int:pk>/',
views.UserProfileDetailView.as_view(), name='profile-
detail'),

        path('profile/update/',
views.UserProfileUpdateView.as_view(), name='profile-
update'),

    ]
```

6.2.3 Основные представления.

UserAPIList – отвечает за вывод списка пользователей

RecipeAPIList – вывод списка рецептов

UserProfileDetailView – отображение профиля пользователя

```
    class UserAPIList(generics.ListAPIView):
        queryset = User.objects.all()
        serializer_class = serializers.UserSerializer


    class RecipeAPIList(generics.ListCreateAPIView):
        queryset = Recipe.objects.all()
        serializer_class = RecipeSerializer
        permission_classes =
[permissions.IsAuthenticatedOrReadOnly]
        def perform_create(self, serializer):
            serializer.save(user=self.request.user)


    class
UserProfileDetailView(generics.RetrieveAPIView):
        queryset = UserProfile.objects.all()
        serializer_class = UserProfileSerializer
        permission_classes = [permissions.AllowAny]
```

7 Перспективы развития приложения

7.1 Анализ пользовательской активности и возможности сбора обратной связи.

Для постоянного улучшения качества и функциональности приложения "YumYard" предусмотрено внедрение системы аналитики, которая позволит отслеживать активность пользователей: частоту использования приложения, наиболее популярные рецепты, активность в комментариях и рейтингах. Эта информация будет использоваться для адаптации контента и функциональных возможностей под интересы и предпочтения пользователей.

Внедрение инструментов для сбора обратной связи, таких как опросы и формы обратной связи, поможет получить ценное мнение пользователей о приложении. Это позволит оперативно реагировать на потребности пользователей, улучшать интерфейс и устранять возможные недостатки в работе приложения.

7.2 Исследование возможностей расширения функционала и улучшения пользовательского опыта.

Развитие технологий и изменение пользовательских требований предоставляют множество возможностей для расширения функционала приложения "YumYard". В планы развития входит интеграция с внешними API для доступа к дополнительным базам данных рецептов, ингредиентов и пищевых продуктов. Также предполагается разработка персонализированных рекомендаций на основе истории поиска и предпочтений пользователей, что сделает использование приложения более удобным и личностно ориентированным.

Улучшение пользовательского опыта также будет включать оптимизацию интерфейса и навигации, чтобы сделать использование приложения более интуитивным и доступным для широкого круга

пользователей, включая тех, кто может иметь ограниченные технические навыки.

7.3 Оценка потенциальных направлений монетизации приложения.

Монетизация приложения "YumYard" может быть реализована через несколько каналов:

- **Реклама:** внедрение таргетированной рекламы продуктов питания и кухонной утвари, что может быть интересно аудитории приложения.

- **Премиум подписки:** введение платных подписок, предоставляющих доступ к эксклюзивным рецептам, дополнительным функциям управления рецептами и отсутствию рекламы.

- **Партнерские программы:** сотрудничество с продовольственными магазинами и кулинарными школами для предложения специальных акций и скидок пользователям приложения.

Эти направления не только помогут в финансировании дальнейшего развития и поддержки приложения, но и усилят его привлекательность для пользователей, предлагая им дополнительную ценность.

8 Приложение

8.1 Диаграммы, использованные в разработке.

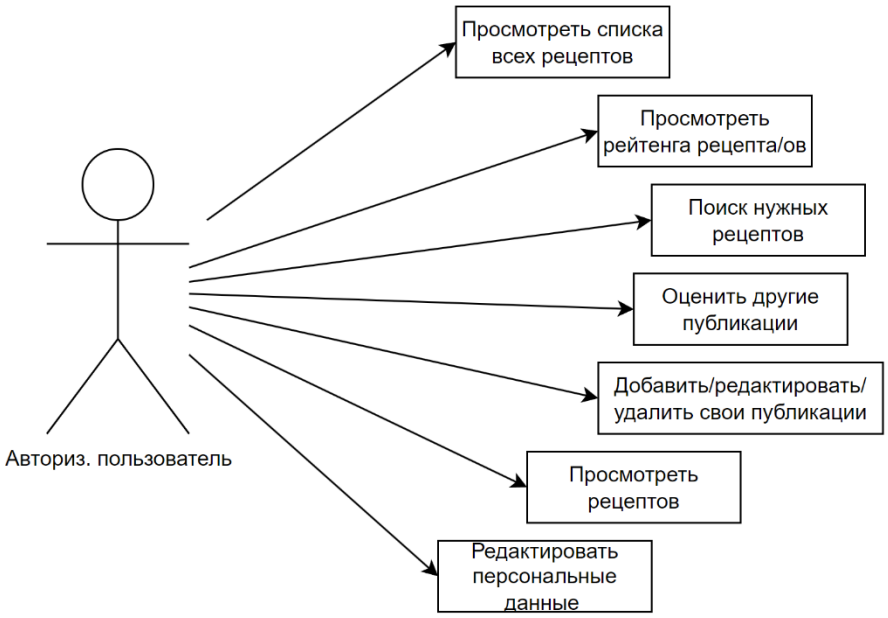

Рисунок 1 - Диаграмма прецедентов. Авторизованный пользователь.

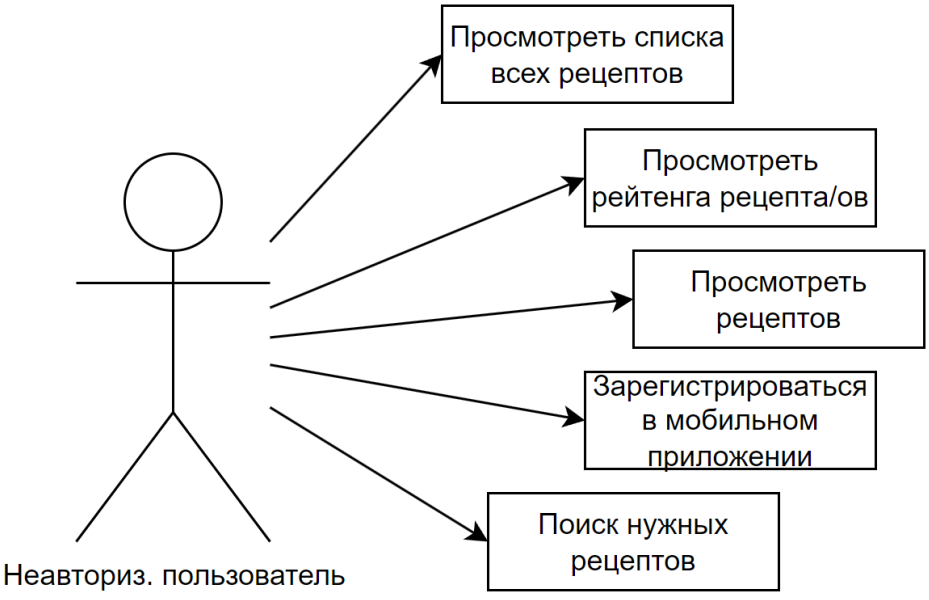

Рисунок 2 - Диаграмма прецедентов. Неавторизованный пользователь.

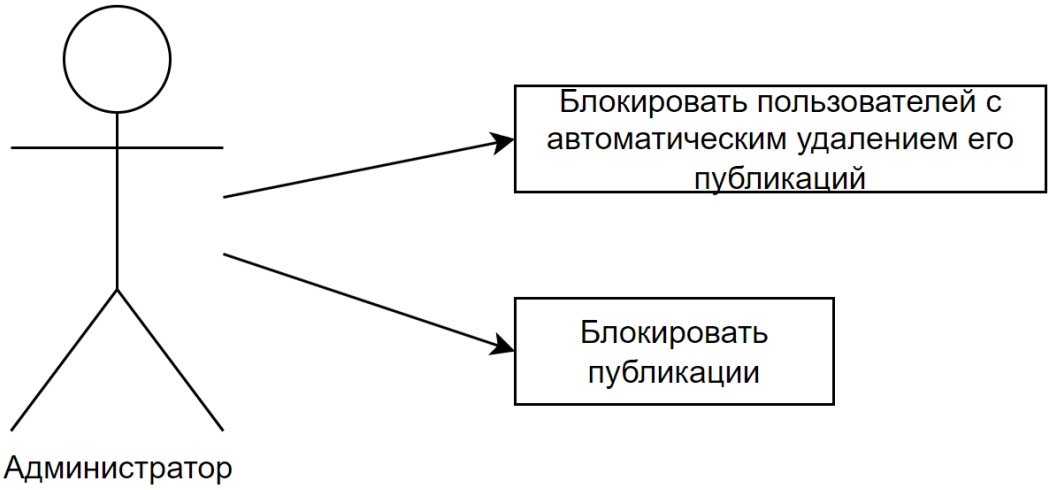

Рисунок 3 - Диаграмма прецедентов. Администратор.

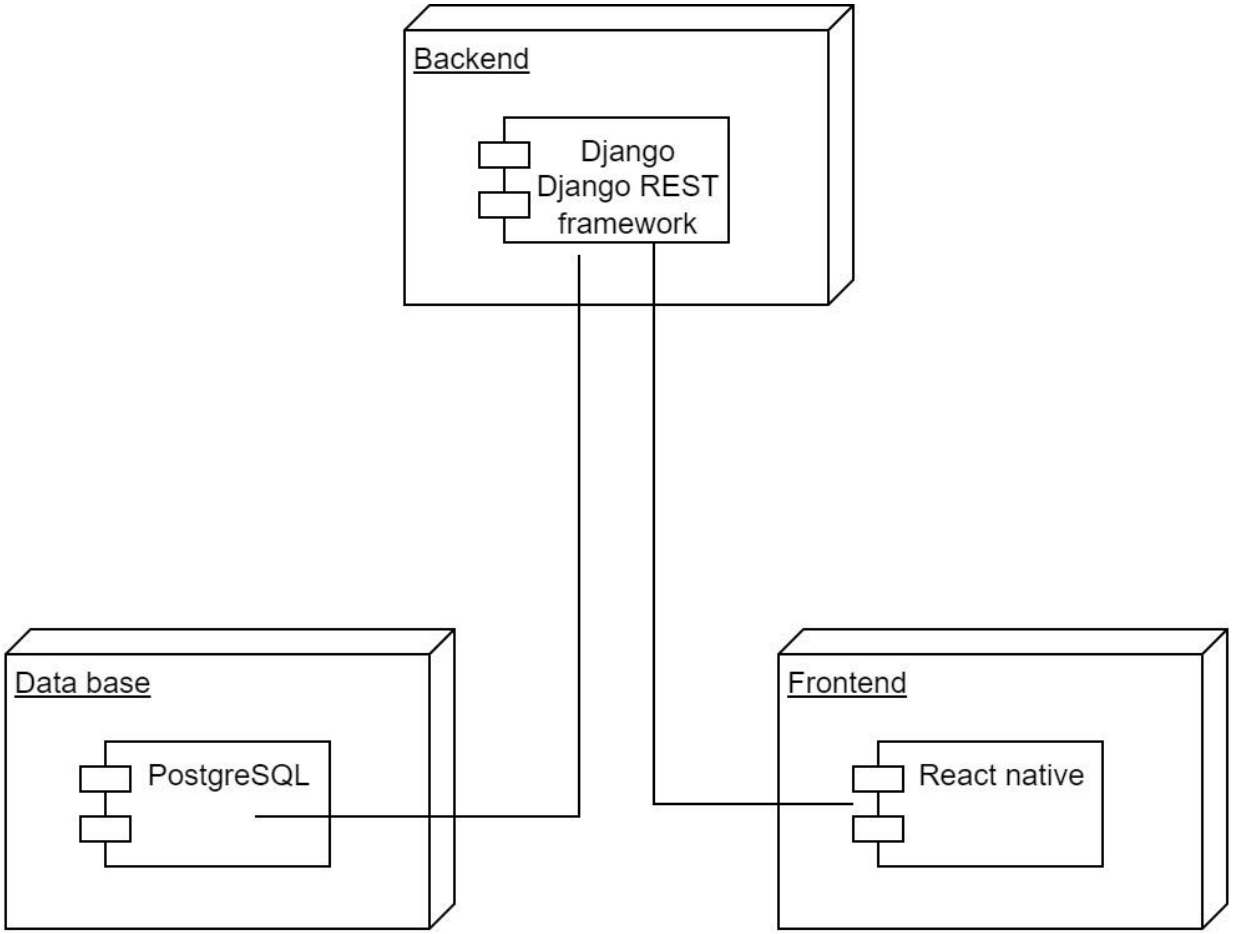

Рисунок 4 - Диаграмма развёртывания.

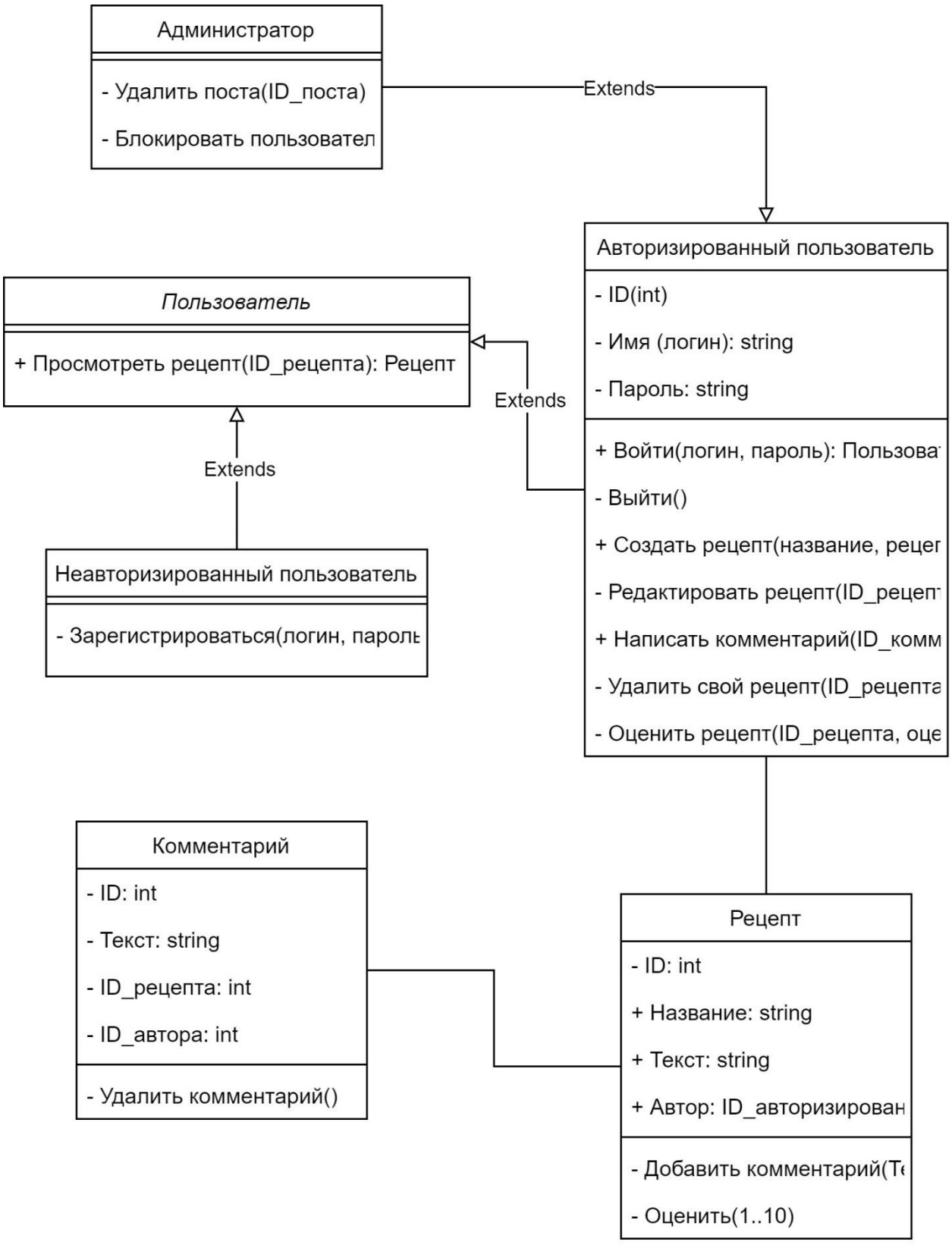

Рисунок 5 - Диаграмма классов.

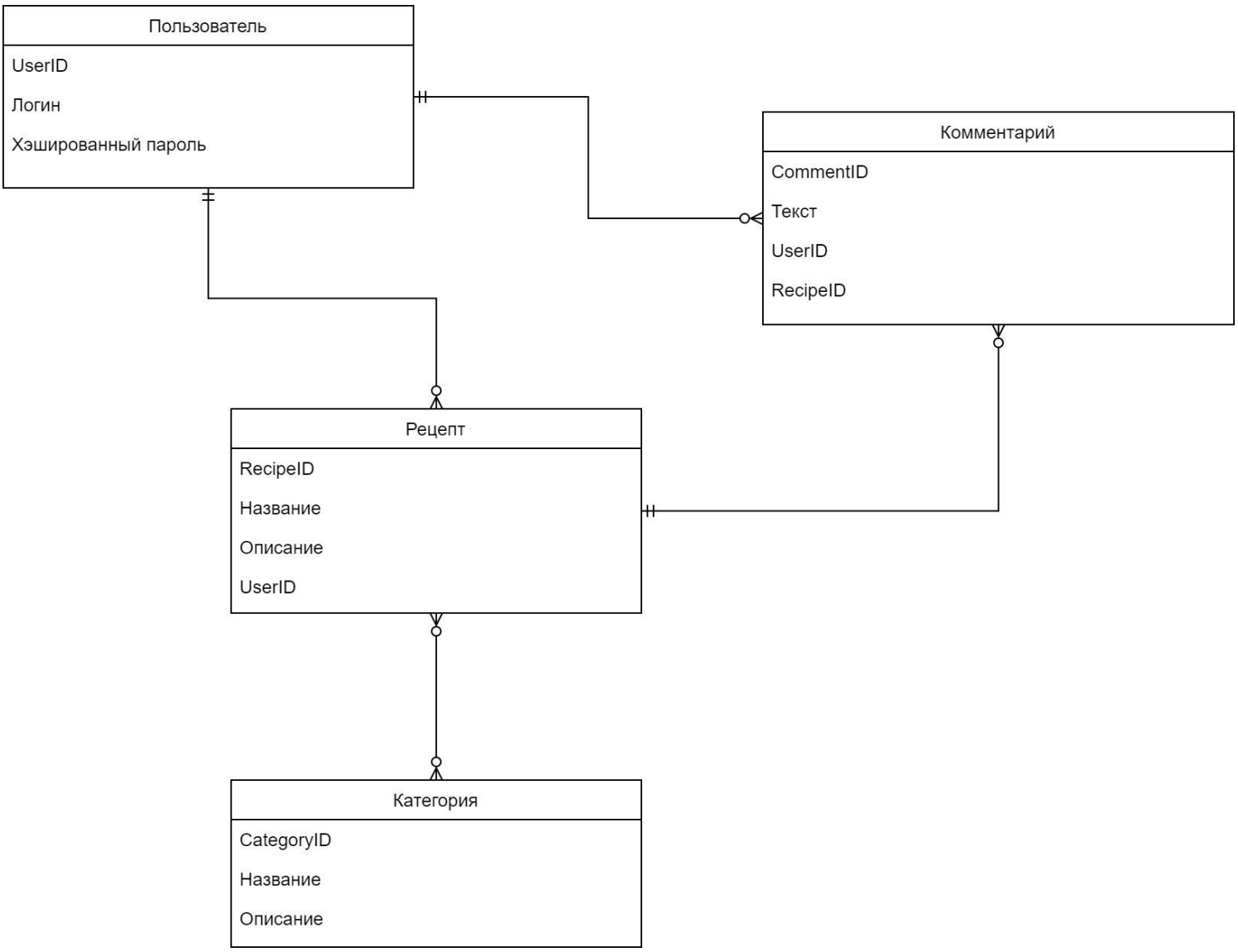

Рисунок 6 - ER диаграмма.

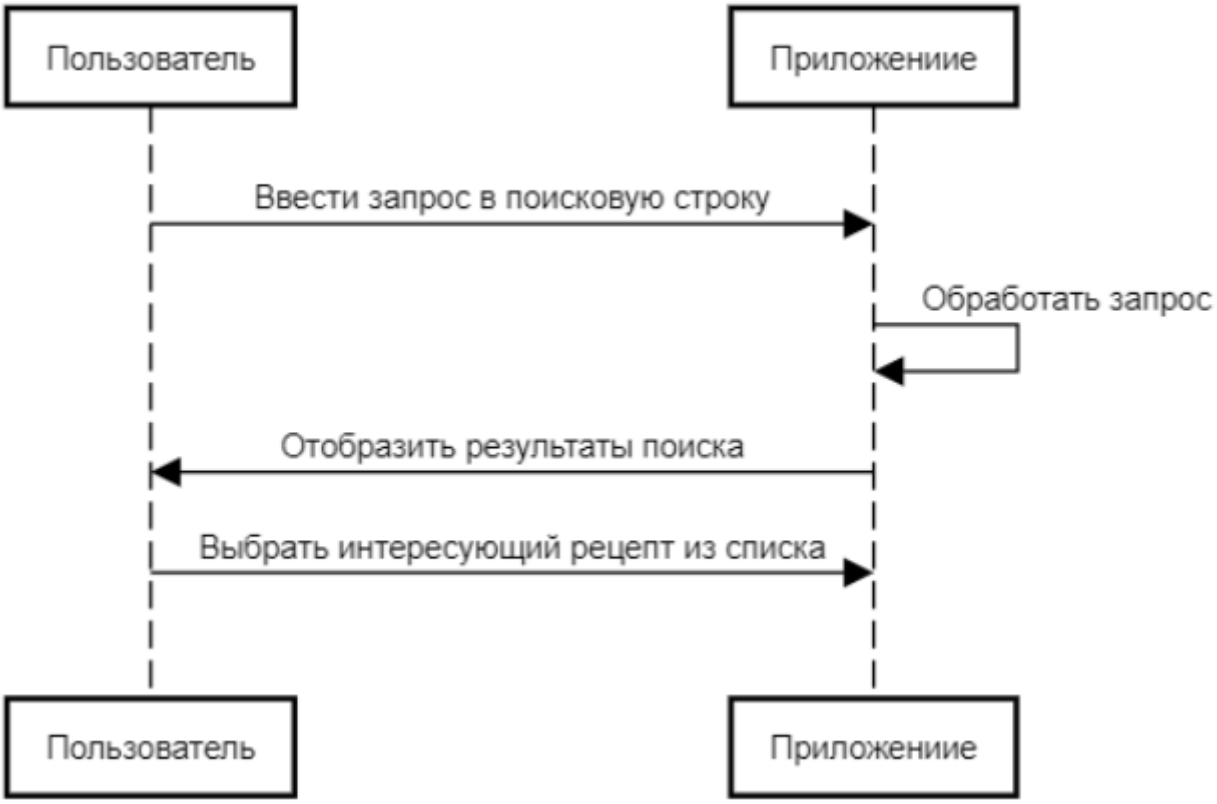

Рисунок 7 - Диаграмма последовательностей. Процесс поиска рецептов.

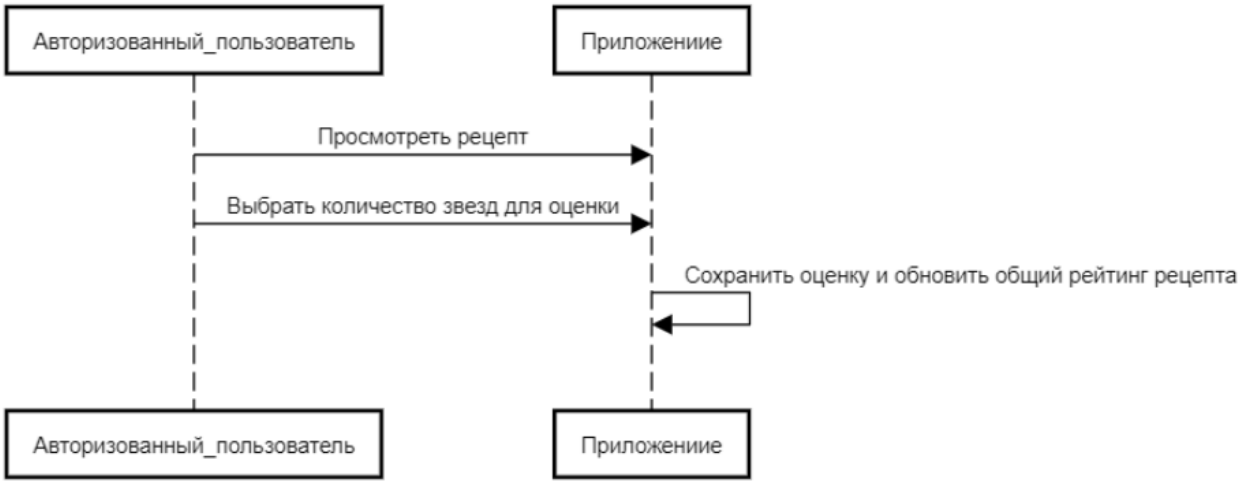

Рисунок 8 - Диаграмма последовательностей. Процесс оценки рецепта.

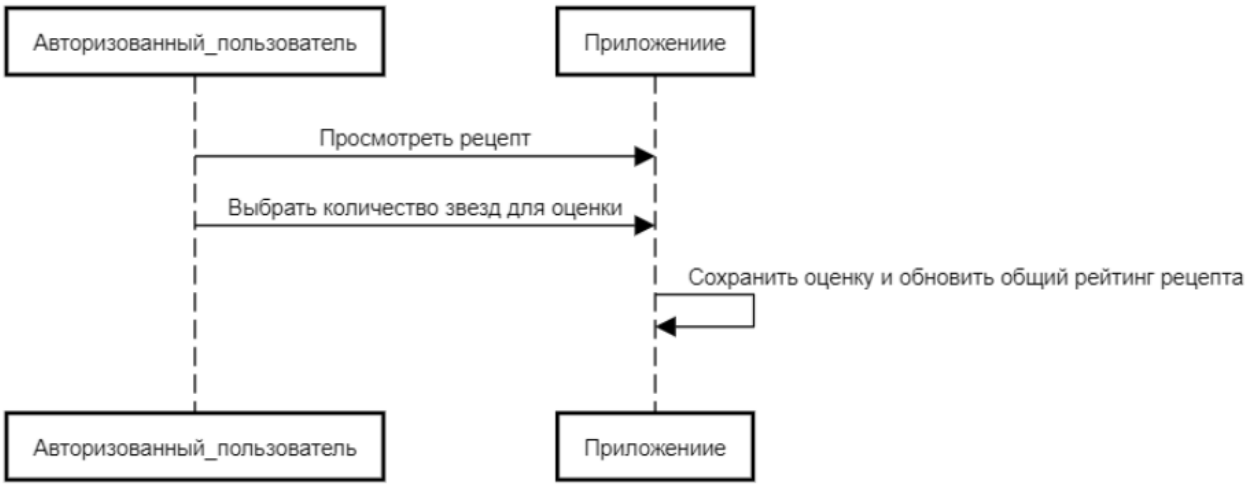

Рисунок 9 - Диаграмма последовательностей. Процесс изменения персональных данных пользователя.

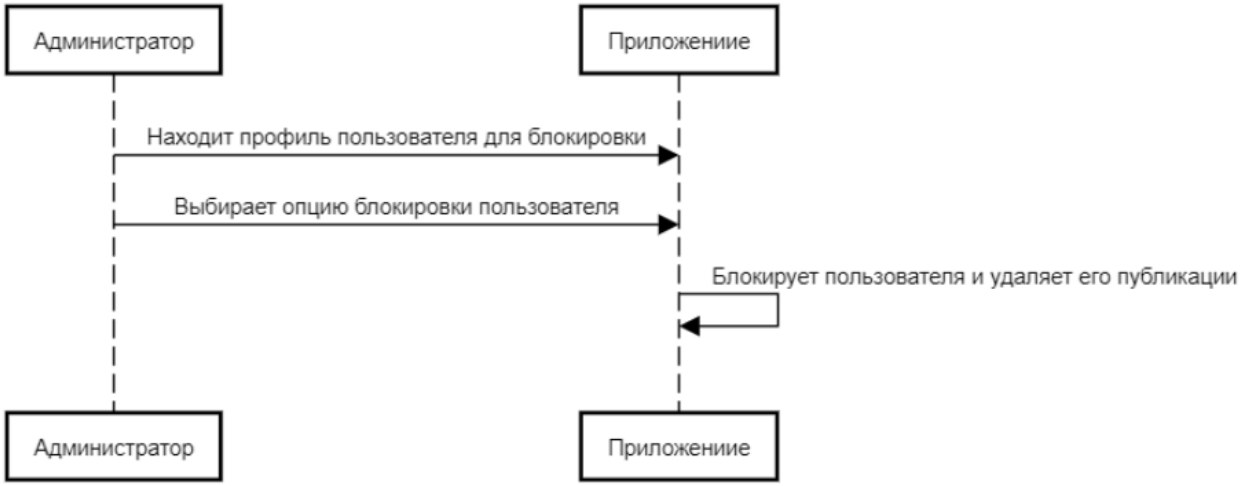

Рисунок 10 - Диаграмма последовательностей. Процесс блокировки пользователя (для администратора).

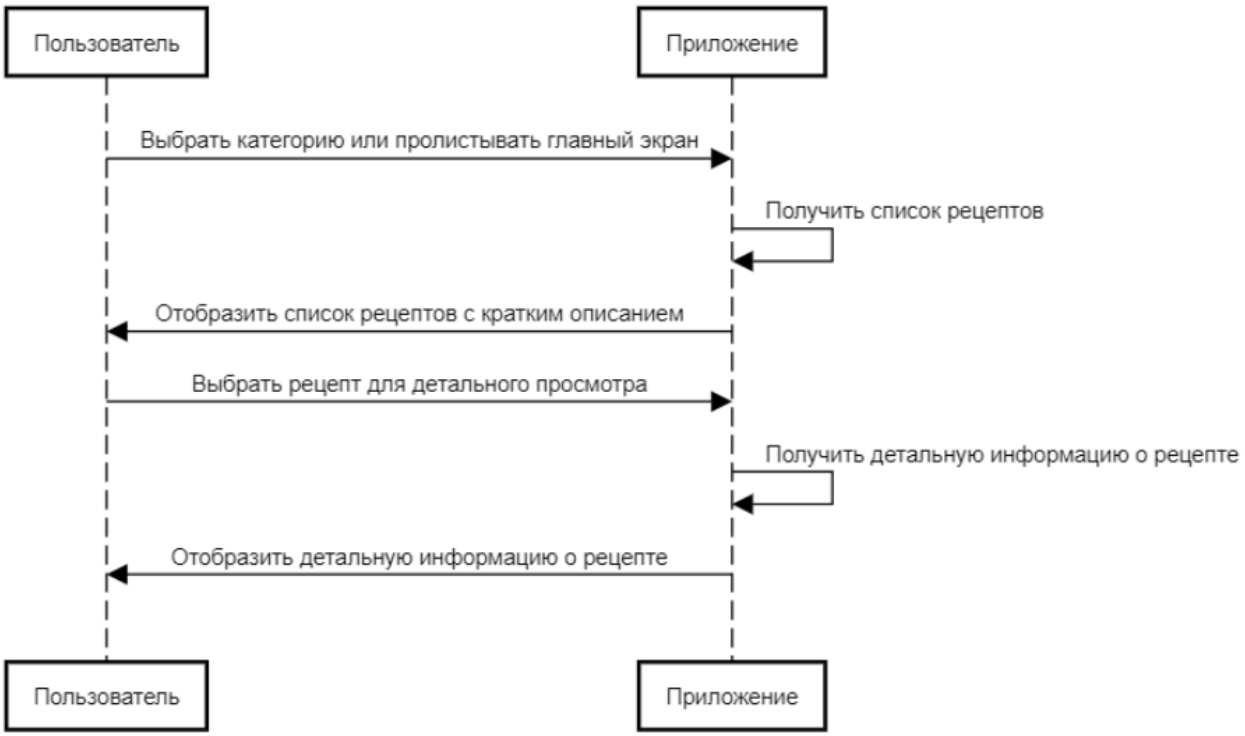

Рисунок 11 - Диаграмма последовательностей. Просмотр рецептов.

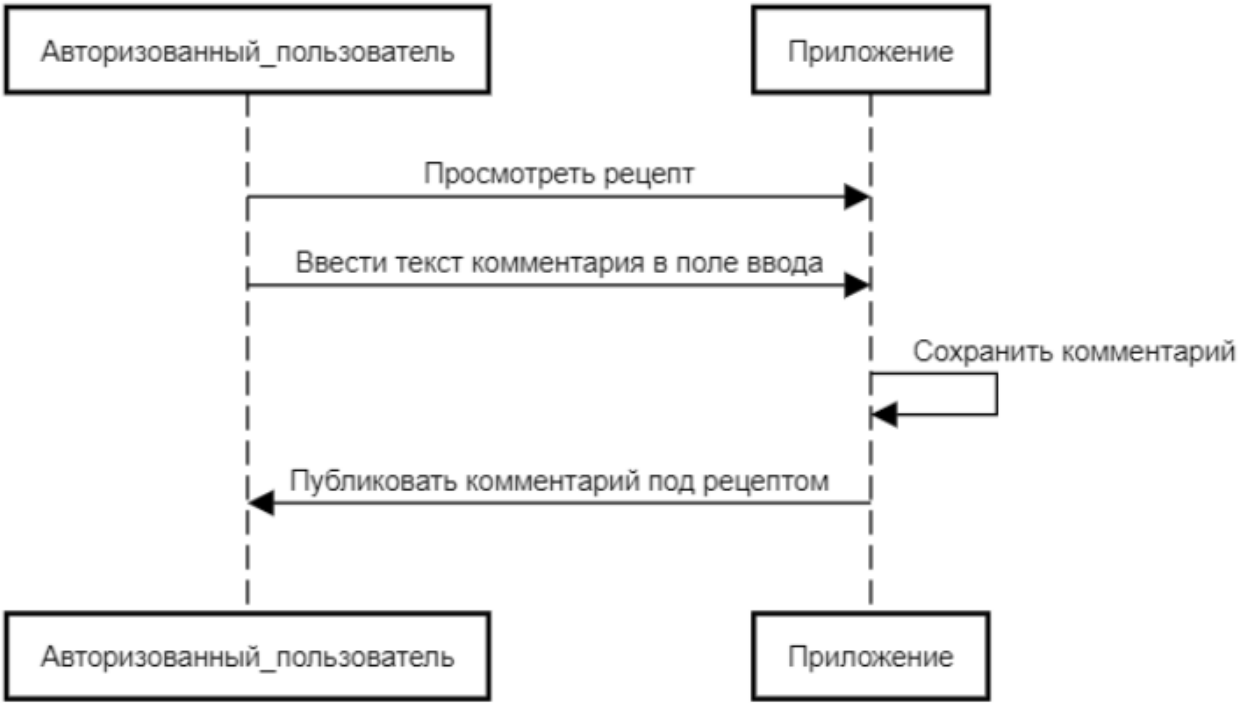

Рисунок 12 - Диаграмма последовательностей. Комментирование рецепта.

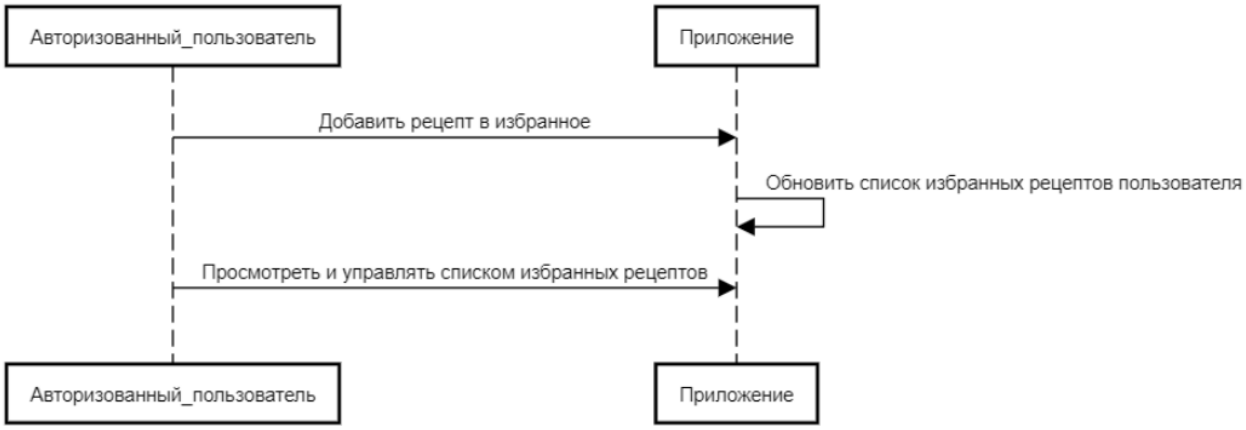

Рисунок 13 - Диаграмма последовательностей. Управление избранными рецептами.

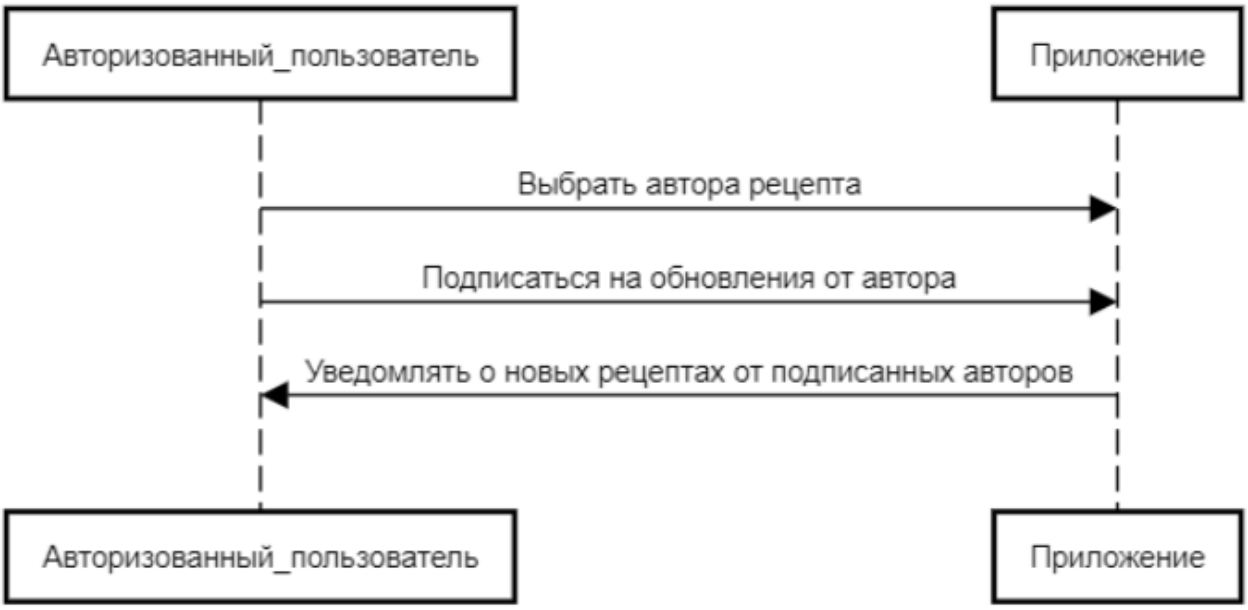

Рисунок 14 - Диаграмма последовательностей. Подписка на авторов рецептов.

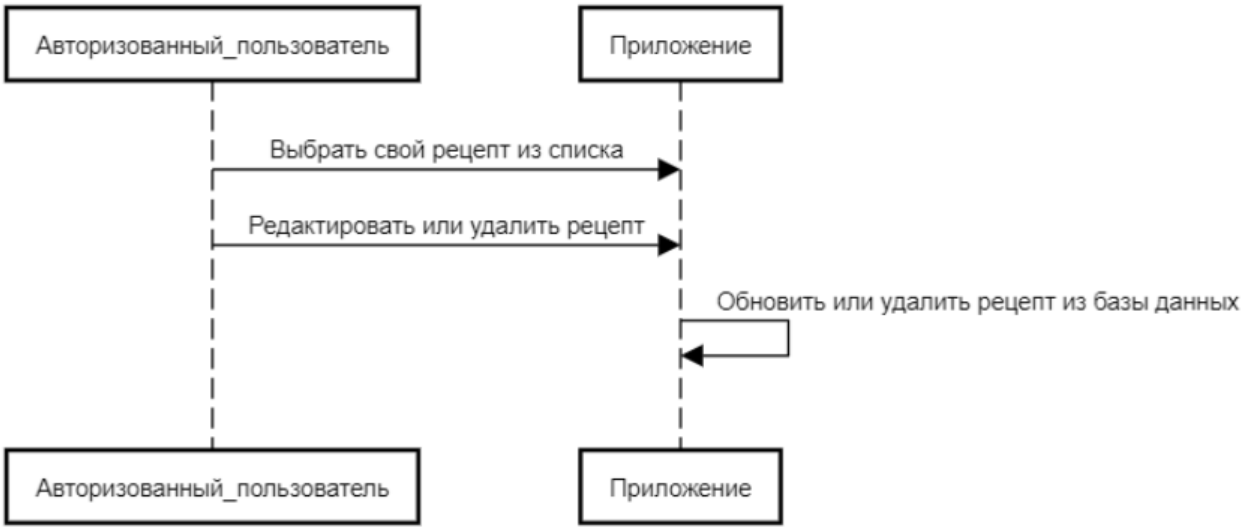

Рисунок 15 - Диаграмма последовательностей. Редактирование и удаление собственных рецептов.

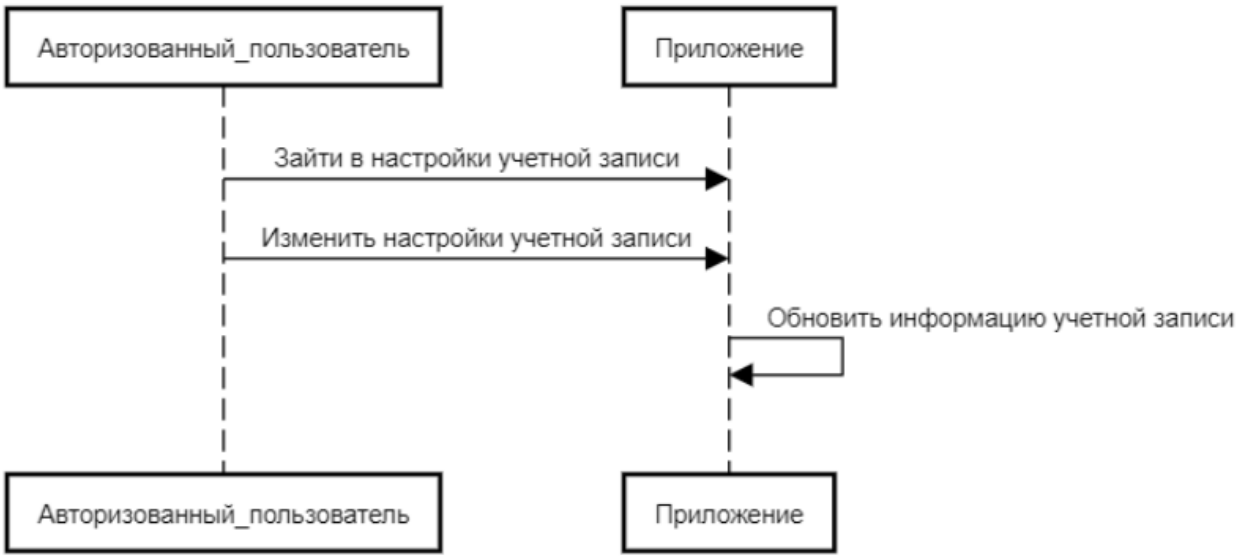

Рисунок 16 - Диаграмма последовательностей. Управление учетной записью.

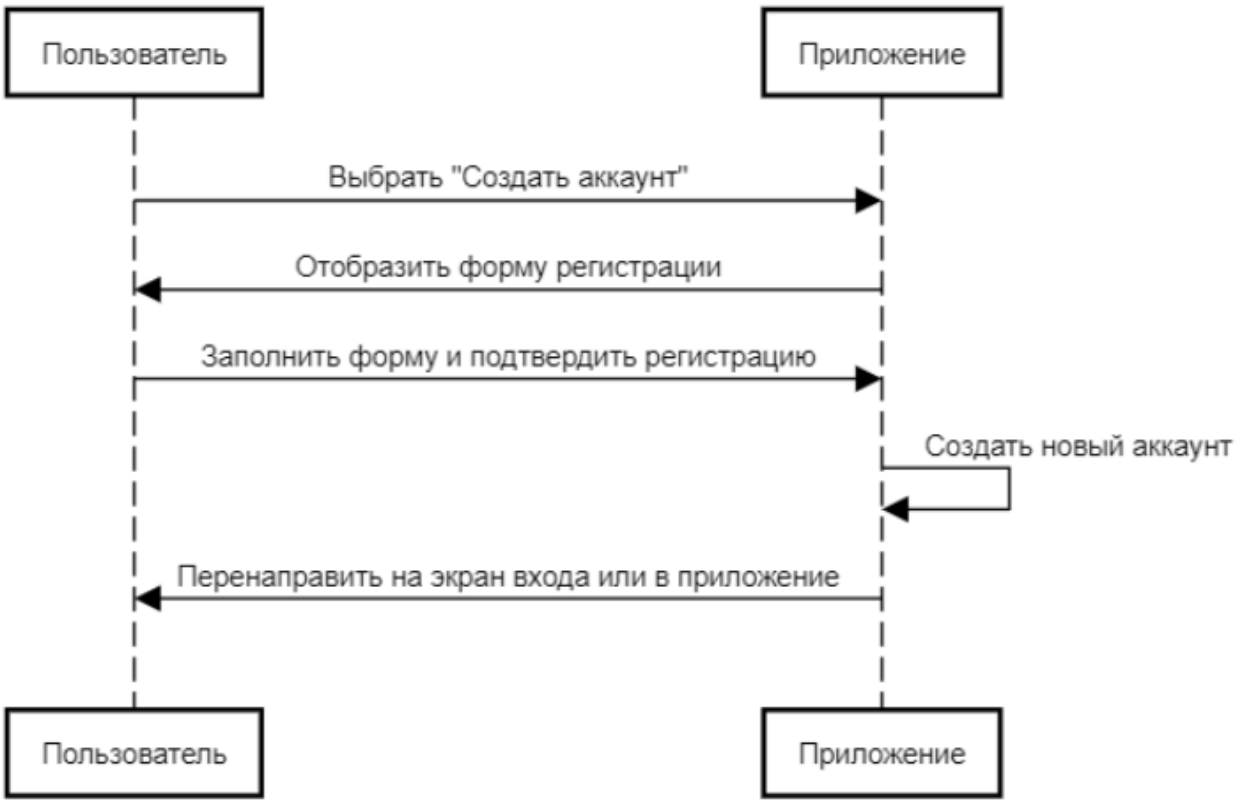

Рисунок 17 - Диаграмма последовательностей. Процесс регистрации в приложении.

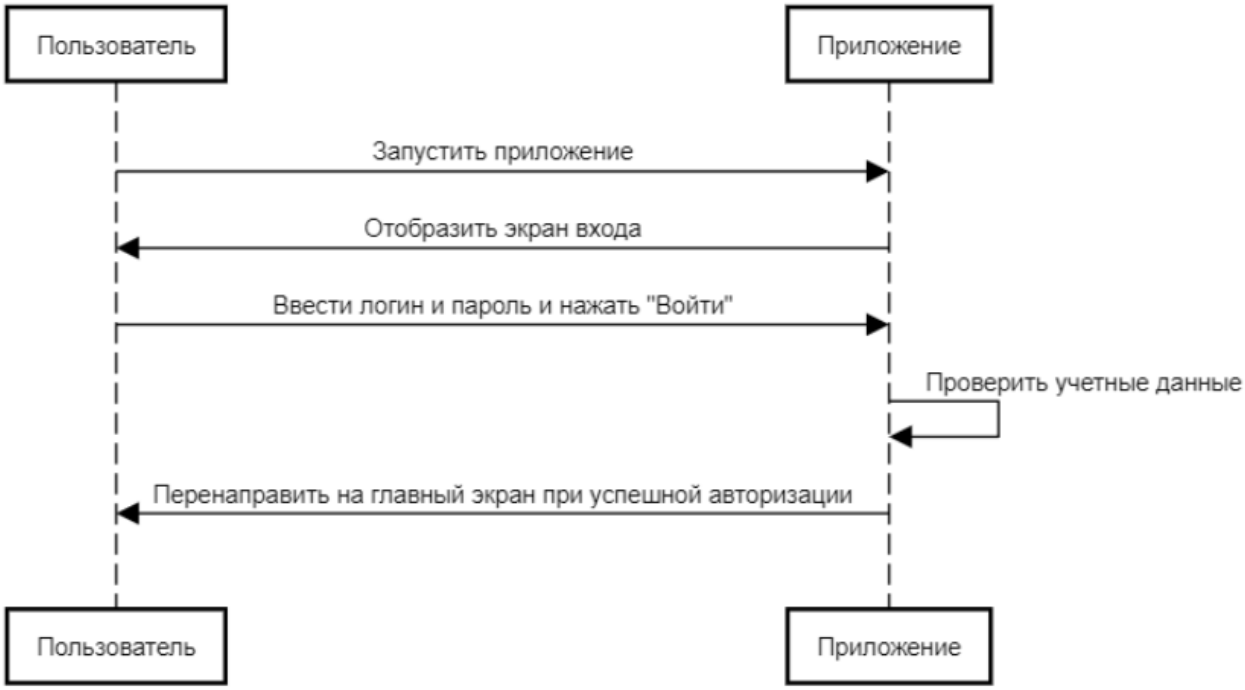

Рисунок 18 - Диаграмма последовательностей. Процесс входа в приложение.

Рисунок 19 - Диаграмма последовательностей. Восстановление забытого пароля.

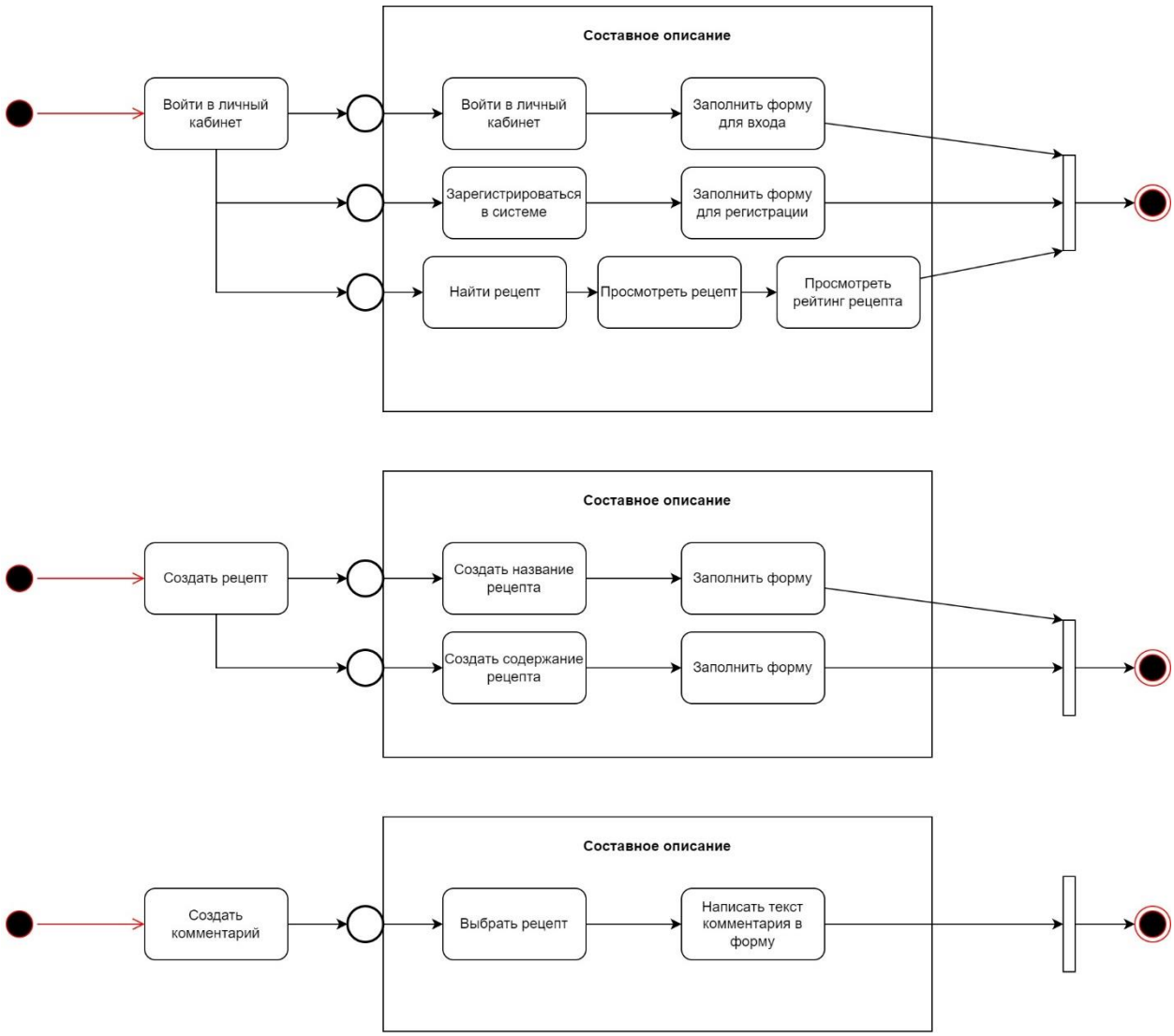

Рисунок 20 - Диаграмма состояний.