Касьянов Артём, НБИбд-01-22, 1032217607

ТЗ + UML

Оглавление

1. ТЗ 2
1.1. Проект Looks like Avito 2
1.2. Основание для разработки 2
1.3. Цель и задачи проекта 2
1.4. Функциональные требования 2
1.5. Нефункциональные требования 3
1.6. Программные средства 4
1.7. Стадии доработки 4
1.8. Ограничения 4
1.9. Контроль и приёмка 4
2. Диаграмма классов 5
3. Диаграмма компонентов 10
4. Диаграмма развертывания 12
5. Диаграмма композитной структуры 14
6. Диаграмма объектов 16
7. Диаграмма пакетов 19
8. Диаграмма временная 22
9. Диаграмма обзоров взаимодействия 24
10. Диаграмма коммуникаций 27
11. Диаграмма состояний 29
12. Диаграмма последовательностей 31
13. Диаграмма активности 34

1. ТЗ

1.1. Проект Looks like Avito

Информационно-аналитическая система размещения, просмотра и управления объявлениями, включающая функции отзывов, рекомендаций и административного контроля.

1.2. Основание для разработки

Данный проект реализуется в рамках курса **"Управление информационно-аналитическими системами"** с целью практического освоения принципов построения ИА-систем, их архитектуры, управления пользователями и данными, на базе объектной модели (UML).

1.3. Цель и задачи проекта

Цель:
Разработка и моделирование учебной ИА-системы, позволяющей пользователям размещать и находить объявления, оставлять отзывы, получать рекомендации и взаимодействовать через интерфейс, а администраторам — управлять содержимым и пользователями.

Задачи:

- Реализация интерфейса для взаимодействия пользователей;
- Обеспечение регистрации и управления сессиями;
- Размещение и поиск товаров;
- Интеграция системы отзывов и модерации;
- Построение системы персонализированных рекомендаций;
- Административный контроль за пользователями и объявлениями.

1.4. Функциональные требования

Система должна включать следующие компоненты и функции, соответствующие структуре диаграммы классов:

Интерфейс (Interface)

- Просмотр объявлений;
- Поиск товаров;
- Личный кабинет;
- Оставление отзывов;
- Получение рекомендаций.

Пользователи (User, Admin)

- Регистрация, авторизация и завершение сессии;
- Обновление личных данных;
- Для администратора:
    - Управление пользователями;
    - Управление товарами;
    - Просмотр отчётов;
    - Модерация отзывов.

Профили (UserProfileController)

- Просмотр, редактирование и удаление профилей.

Сессии (SessionRepository)

- Хранение и управление активными сессиями.

Объявления (Product, ProductRepository)

- Добавление, обновление и удаление товаров;
- Поиск товаров по заданным критериям;
- Загрузка фотографий и подробностей.

Отзывы (Review, ReviewService, ReviewController, SellerReviewRepository)

- Оставление, редактирование и удаление отзывов;
- Проверка и модерация отзывов;
- Получение отзывов по товарам и продавцам;
- Фильтрация по рейтингу.

Рекомендации (RecommendationAnalysis, RecommendationsModule)

- Анализ поведения пользователя;
- Генерация персональных рекомендаций;
- Отслеживание взаимодействий.

1.5. Нефункциональные требования

- Отзывчивый и понятный пользовательский интерфейс;
- Время ответа при стандартной нагрузке — не более 2–3 секунд;
- Базовая защита от несанкционированного доступа;
- Масштабируемость и возможность хранения больших массивов данных;
- Поддержка клиент-серверной архитектуры.

1.6. Программные средства

Языки и технологии:

- Backend: Python (Flask / FastAPI)
- Frontend: React / HTML + CSS
- База данных: PostgreSQL / SQLite
- Инструменты анализа: Pandas, Plotly
- Визуализация классов: PlantUML

1.7. Стадии доработки

1. Анализ требований и проектирование (включая диаграммы классов);
2. Реализация пользовательских сценариев;
3. Разработка логики отзывов и рекомендаций;
4. Интеграция административных функций;
5. Тестирование системы и документирование;
6. Подготовка отчётных материалов.

1.8. Ограничения

- Использование тестовых данных или парсинг открытых площадок;
- Отсутствие интеграции с реальной платёжной системой;
- Все функции реализуются в рамках учебного проекта.

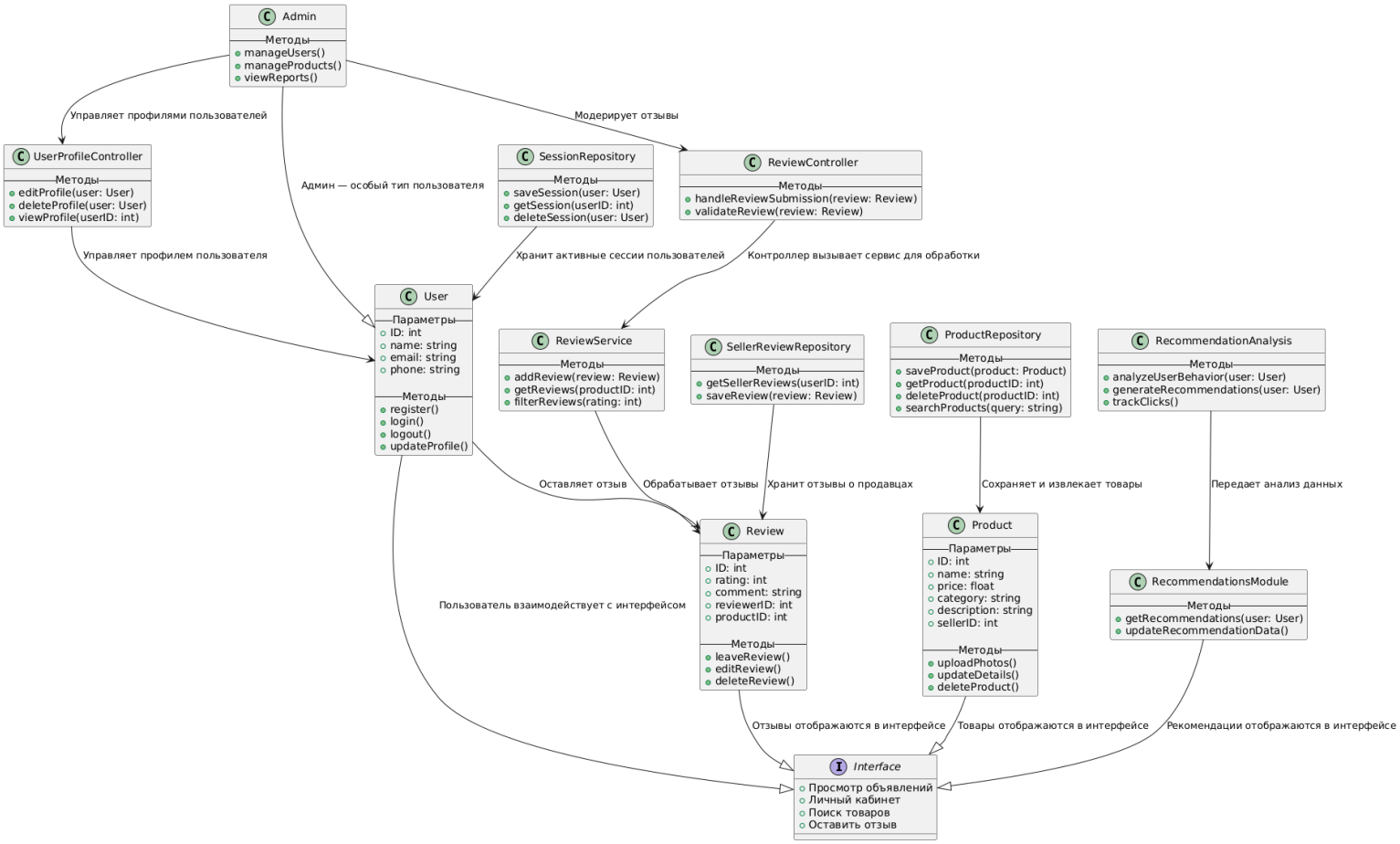
1.9. Контроль и приёмка

Система считается завершённой после реализации следующих компонентов:

- Функциональность всех описанных классов;
- Демонстрация работы через пользовательский интерфейс;
- Презентация отчёта с диаграммой классов и кратким описанием компонентов.

2. Диаграмма классов

Фото:



Код:

```
@startuml


' ======= Интерфейс =======

interface "Interface" {

    + Просмотр объявлений

    + Личный кабинет

    + Поиск товаров

    + Оставить отзыв

}


' ======= Пользователи =======

class User {

    -- Параметры --
```

```
    + ID: int
    + name: string
    + email: string
    + phone: string

    -- Методы --
    + register()
    + login()
    + logout()
    + updateProfile()
}

class Admin {
    -- Методы --
    + manageUsers()
    + manageProducts()
    + viewReports()
}

User --|> Interface : "Пользователь взаимодействует с интерфейсом"
Admin --|> User : "Админ — особый тип пользователя"
Admin --> UserProfileController : "Управляет профилями пользователей"
Admin --> ReviewController : "Модерирует отзывы"

' ======= Управление профилем =======
class UserProfileController {
    -- Методы --
    + editProfile(user: User)
    + deleteProfile(user: User)
    + viewProfile(userID: int)
}

UserProfileController --> User : "Управляет профилем пользователя"
```

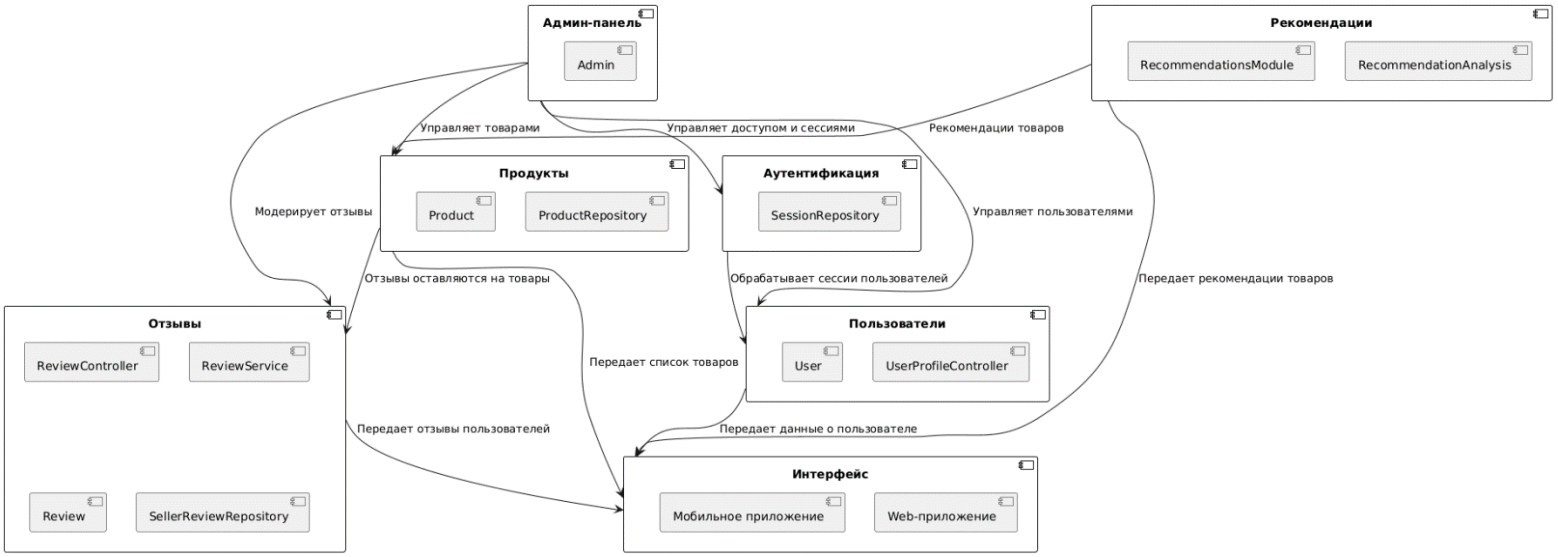
```
' ======= Сессии =======
class SessionRepository {
   -- Методы --
   + saveSession(user: User)
   + getSession(userID: int)
   + deleteSession(user: User)
}

SessionRepository --> User : "Хранит активные сессии пользователей"

' ======= Продукты =======
class Product {
   -- Параметры --
   + ID: int
   + name: string
   + price: float
   + category: string
   + description: string
   + sellerID: int

   -- Методы --
   + uploadPhotos()
   + updateDetails()
   + deleteProduct()
}

class ProductRepository {
   -- Методы --
   + saveProduct(product: Product)
   + getProduct(productID: int)
   + deleteProduct(productID: int)
   + searchProducts(query: string)
}
```

```
ProductRepository --> Product : "Сохраняет и извлекает товары"
Product --|> Interface : "Товары отображаются в интерфейсе"

' ======= Отзывы =======
class Review {
   -- Параметры --
   + ID: int
   + rating: int
   + comment: string
   + reviewerID: int
   + productID: int

   -- Методы --
   + leaveReview()
   + editReview()
   + deleteReview()
}

class ReviewService {
   -- Методы --
   + addReview(review: Review)
   + getReviews(productID: int)
   + filterReviews(rating: int)
}

class ReviewController {
   -- Методы --
   + handleReviewSubmission(review: Review)
   + validateReview(review: Review)
}

class SellerReviewRepository {
   -- Методы --
   + getSellerReviews(userID: int)
```

```
    + saveReview(review: Review)
}

ReviewService --> Review : "Обрабатывает отзывы"
ReviewController --> ReviewService : "Контроллер вызывает сервис для обработки"
SellerReviewRepository --> Review : "Хранит отзывы о продавцах"
Review --|> Interface : "Отзывы отображаются в интерфейсе"
User --> Review : "Оставляет отзыв"

' ======= Рекомендации =======
class RecommendationAnalysis {
    -- Методы --
    + analyzeUserBehavior(user: User)
    + generateRecommendations(user: User)
    + trackClicks()
}

class RecommendationsModule {
    -- Методы --
    + getRecommendations(user: User)
    + updateRecommendationData()
}

RecommendationAnalysis --> RecommendationsModule : "Передает анализ данных"
RecommendationsModule --|> Interface : "Рекомендации отображаются в интерфейсе"

@enduml
```

3. Диаграмма компонентов

Фото:



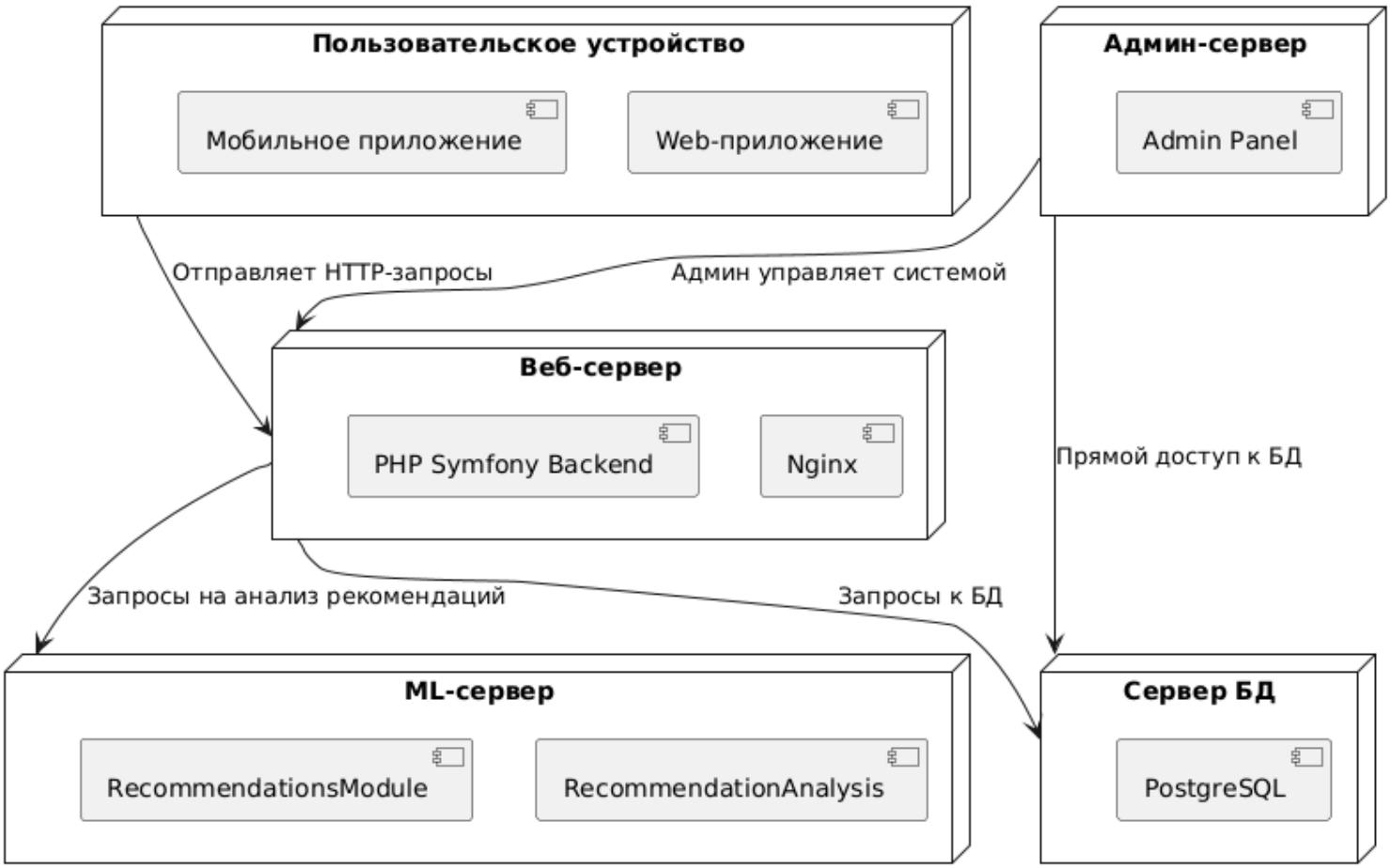
Код:

```
@startuml

' ======= Компоненты =======
component "Интерфейс" as Interface {
  [Web-приложение]
  [Мобильное приложение]
}

component "Аутентификация" as Auth {
  [SessionRepository]
}

component "Пользователи" as UserManagement {
  [UserProfileController]
  [User]
}

component "Продукты" as ProductManagement {
  [ProductRepository]
  [Product]
}

component "Отзывы" as ReviewManagement {
  [ReviewController]
  [ReviewService]
  [Review]
  [SellerReviewRepository]
}
```

```
component "Рекомендации" as Recommendations {
   [RecommendationAnalysis]
   [RecommendationsModule]
}

component "Админ-панель" as AdminPanel {
   [Admin]
}

' ======= Связи между компонентами =======
Auth --> UserManagement : "Обрабатывает сессии пользователей"
ProductManagement --> ReviewManagement : "Отзывы оставляются на товары"
Recommendations --> ProductManagement : "Рекомендации товаров"

' ======= Админ управляет всеми модулями, кроме интерфейса и рекомендаций =======
AdminPanel --> Auth : "Управляет доступом и сессиями"
AdminPanel --> UserManagement : "Управляет пользователями"
AdminPanel --> ProductManagement : "Управляет товарами"
AdminPanel --> ReviewManagement : "Модерирует отзывы"

' ======= Данные передаются в интерфейс =======
UserManagement --> Interface : "Передает данные о пользователе"
ProductManagement --> Interface : "Передает список товаров"
ReviewManagement --> Interface : "Передает отзывы пользователей"
Recommendations --> Interface : "Передает рекомендации товаров"

@enduml
```

4. Диаграмма развертывания

Фото:



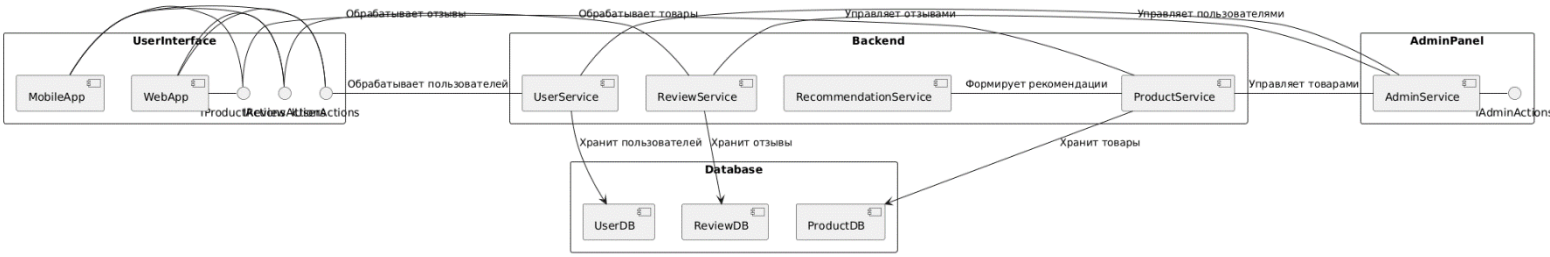
Код:

```
@startuml


' ====== Узлы (серверы и устройства) ======
node "Пользовательское устройство" as UserDevice {
   [Web-приложение]
   [Мобильное приложение]
}


node "Веб-сервер" as WebServer {
   [Nginx]
   [PHP Symfony Backend]
}
```

```
node "Сервер БД" as DatabaseServer {
   [PostgreSQL]
}

node "ML-сервер" as MLServer {
   [RecommendationAnalysis]
   [RecommendationsModule]
}

node "Админ-сервер" as AdminServer {
   [Admin Panel]
}

' ====== Связи между узлами ======
UserDevice --> WebServer : "Отправляет HTTP-запросы"
WebServer --> DatabaseServer : "Запросы к БД"
WebServer --> MLServer : "Запросы на анализ рекомендаций"
AdminServer --> WebServer : "Админ управляет системой"
AdminServer --> DatabaseServer : "Прямой доступ к БД"

@enduml
```

5. Диаграмма композитной структуры

Фото:



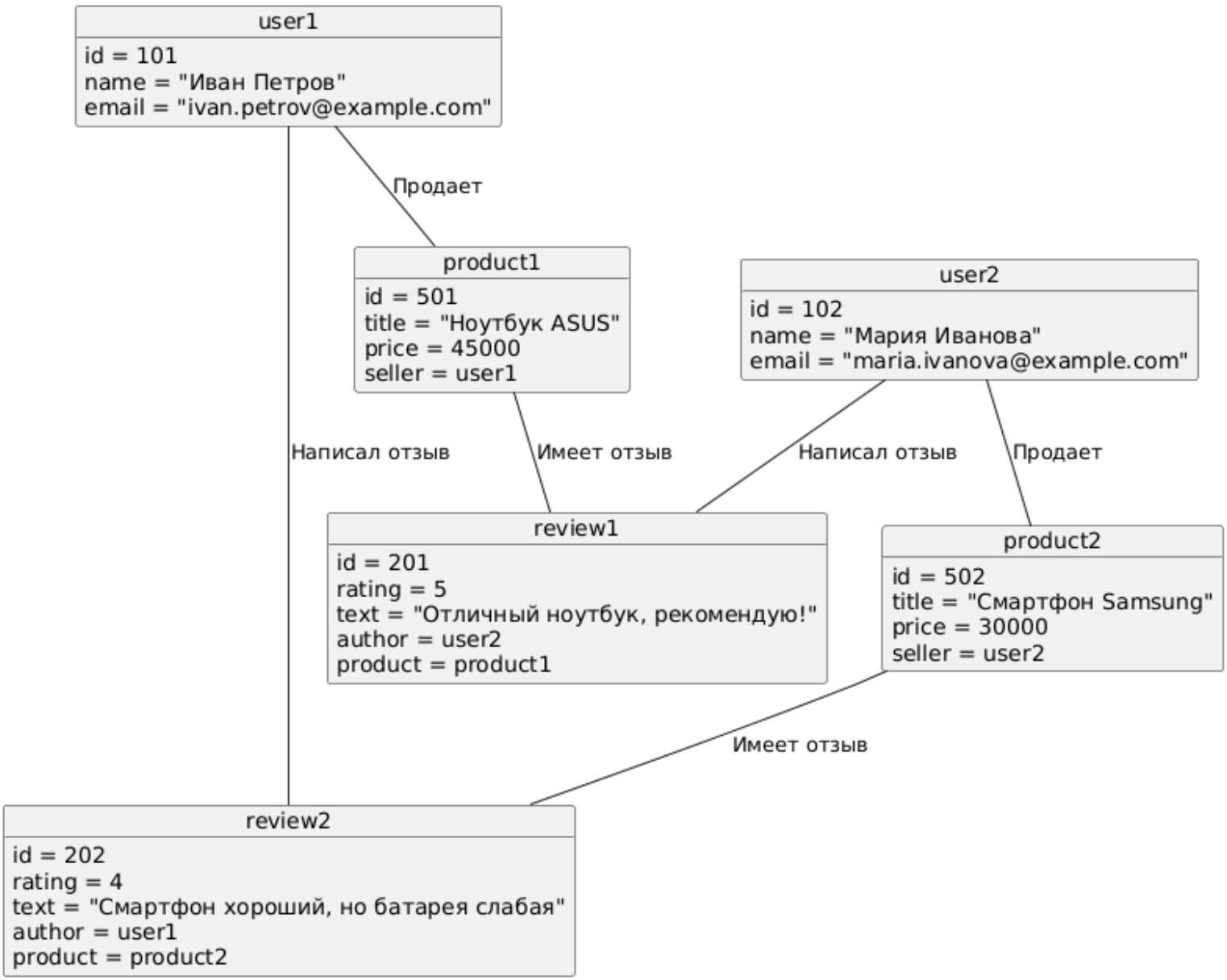
Код:

```
@startuml

' ====== Основные компоненты (классификаторы) ======
rectangle "UserInterface" {
   interface "IUserActions"
   interface "IReviewActions"
   interface "IProductActions"

   [WebApp]
   [MobileApp]

   WebApp - [IUserActions]
   MobileApp - [IUserActions]

   WebApp - [IReviewActions]
   MobileApp - [IReviewActions]

   WebApp - [IProductActions]
   MobileApp - [IProductActions]
}

rectangle "Backend" {
   component "UserService"
```

```
    component "ReviewService"
    component "ProductService"
    component "RecommendationService"

    [UserService] - [IUserActions] : "Обрабатывает пользователей"
    [ReviewService] - [IReviewActions] : "Обрабатывает отзывы"
    [ProductService] - [IProductActions] : "Обрабатывает товары"
    [RecommendationService] - [ProductService] : "Формирует рекомендации"
}

rectangle "Database" {
    [UserDB]
    [ReviewDB]
    [ProductDB]

    UserService --> UserDB : "Хранит пользователей"
    ReviewService --> ReviewDB : "Хранит отзывы"
    ProductService --> ProductDB : "Хранит товары"
}

rectangle "AdminPanel" {
    interface "IAdminActions"
    component "AdminService"

    AdminService - [IAdminActions]
    AdminService - UserService : "Управляет пользователями"
    AdminService - ReviewService : "Управляет отзывами"
    AdminService - ProductService : "Управляет товарами"
}

@enduml
```

6. Диаграмма объектов

Фото:



Код:

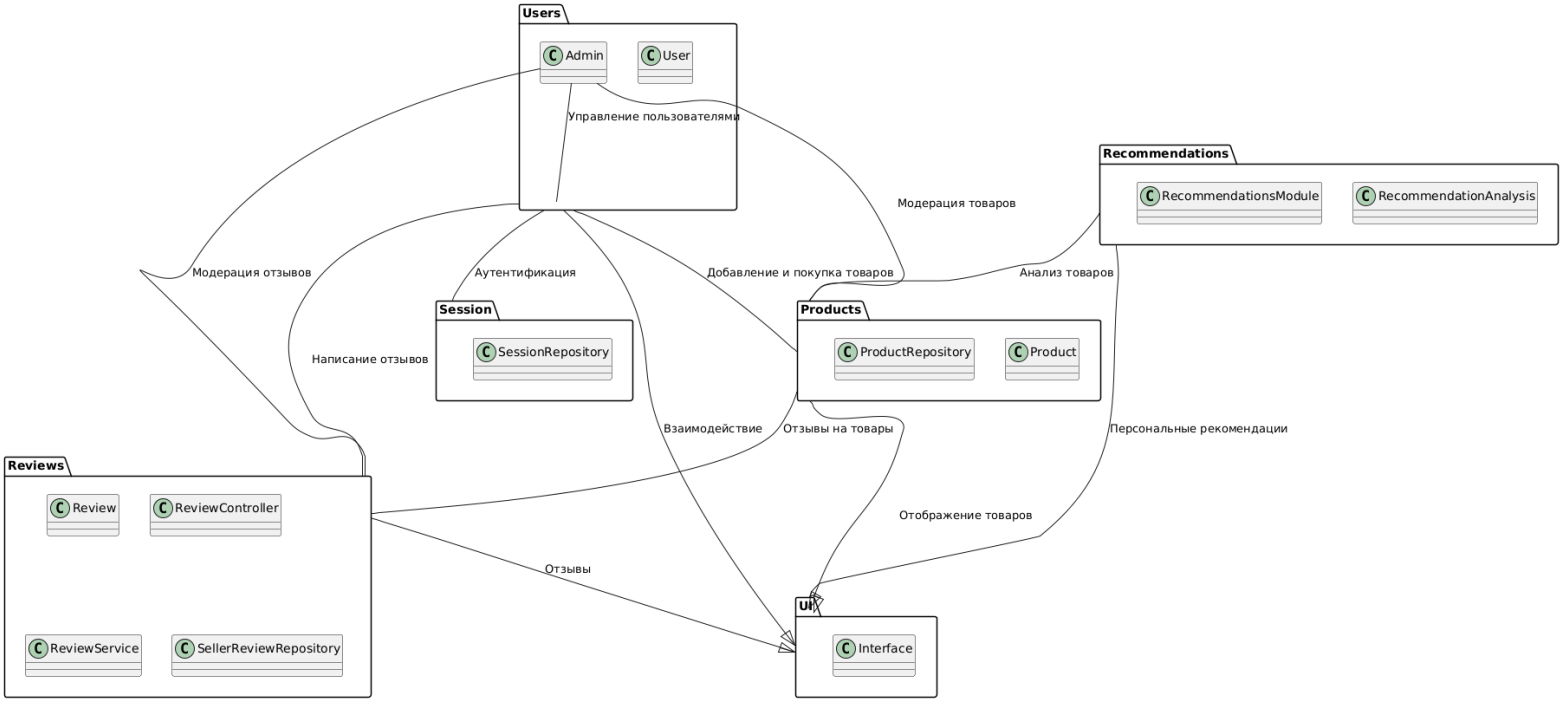
```
@startuml


' ====== Примеры объектов пользователей ======

object user1 {

  id = 101

  name = "Иван Петров"

  email = "ivan.petrov@example.com"
```

```
}

object user2 {
  id = 102
  name = "Мария Иванова"
  email = "maria.ivanova@example.com"
}

' ====== Примеры объектов товаров ======
object product1 {
  id = 501
  title = "Ноутбук ASUS"
  price = 45000
  seller = user1
}

object product2 {
  id = 502
  title = "Смартфон Samsung"
  price = 30000
  seller = user2
}

' ====== Примеры объектов отзывов ======
object review1 {
  id = 201
  rating = 5
  text = "Отличный ноутбук, рекомендую!"
  author = user2
  product = product1
}
```

```
object review2 {
  id = 202
  rating = 4
  text = "Смартфон хороший, но батарея слабая"
  author = user1
  product = product2
}

' ====== Взаимосвязи между объектами ======
user1 -- product1 : "Продает"
user2 -- product2 : "Продает"
user2 -- review1 : "Написал отзыв"
user1 -- review2 : "Написал отзыв"
product1 -- review1 : "Имеет отзыв"
product2 -- review2 : "Имеет отзыв"

@enduml
```

7. Диаграмма пакетов

Фото:



Код:

```
@startuml

' ====== Определяем пакеты ======

package "UI" {
  class Interface
}

package "Users" {
  class User
  class Admin
}

package "Products" {
  class Product
  class ProductRepository
```

```
}

package "Reviews" {
  class Review
  class ReviewController
  class ReviewService
  class SellerReviewRepository
}

package "Recommendations" {
  class RecommendationAnalysis
  class RecommendationsModule
}

package "Session" {
  class SessionRepository
}

' ====== Определяем зависимости между пакетами ======

"Users" --|> "UI" : "Взаимодействие"
"Products" --|> "UI" : "Отображение товаров"
"Reviews" --|> "UI" : "Отзывы"
"Recommendations" --|> "UI" : "Персональные рекомендации"

"Users" -- "Products" : "Добавление и покупка товаров"
"Users" -- "Reviews" : "Написание отзывов"
"Users" -- "Session" : "Аутентификация"

"Products" -- "Reviews" : "Отзывы на товары"
"Recommendations" -- "Products" : "Анализ товаров"
"Admin" -- "Users" : "Управление пользователями"
```

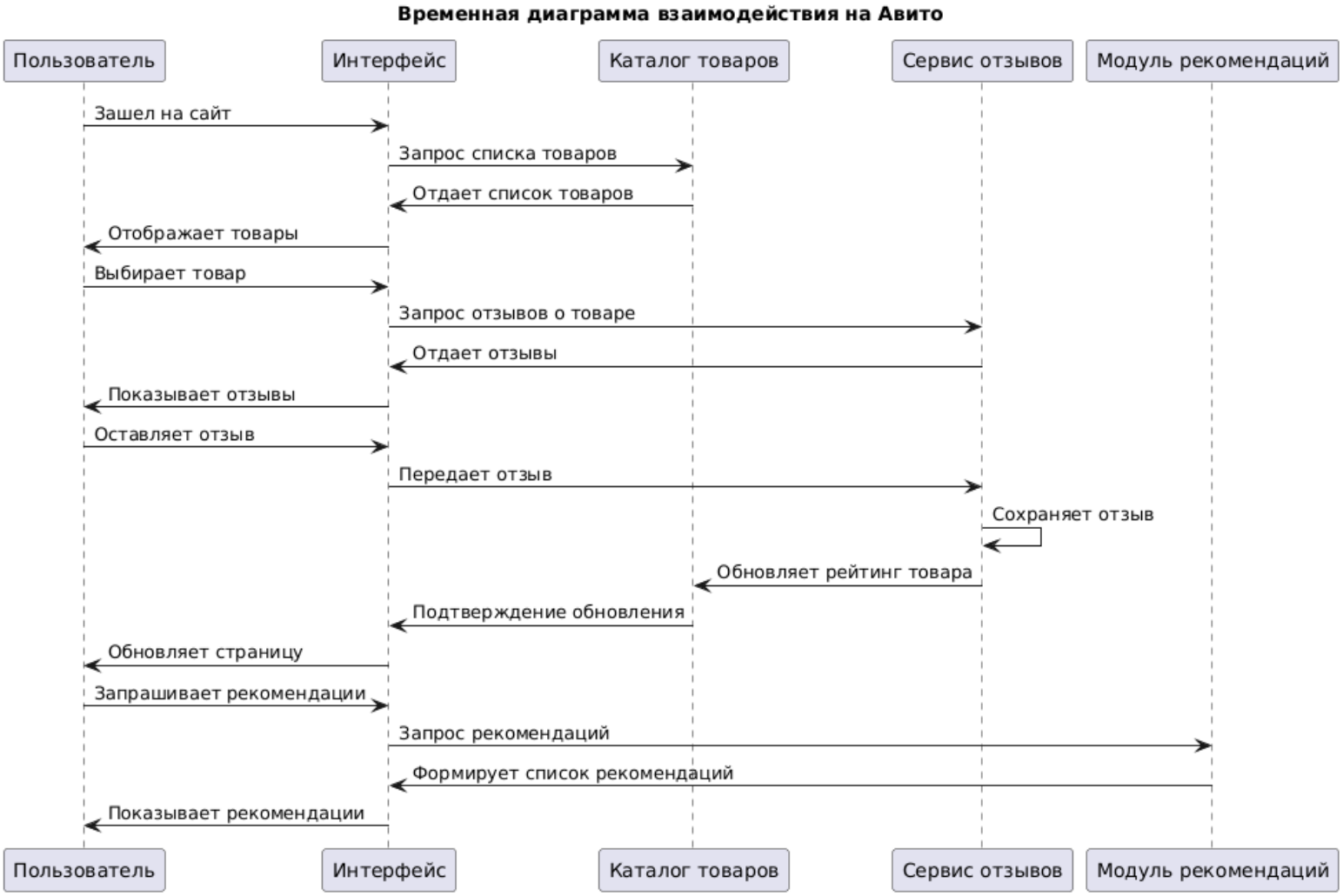
"Admin" -- "Products" : "Модерация товаров"

"Admin" -- "Reviews" : "Модерация отзывов"

@enduml

8. Диаграмма временная

Фото:



Код:

@startuml

title Временная диаграмма взаимодействия на Авито

participant "Пользователь" as User

participant "Интерфейс" as UI

participant "Каталог товаров" as ProductRepo

participant "Сервис отзывов" as ReviewService

participant "Модуль рекомендаций" as RecoModule

User -> UI: Зашел на сайт

UI -> ProductRepo: Запрос списка товаров

ProductRepo -> UI: Отдает список товаров

UI -> User: Отображает товары

User -> UI: Выбирает товар

UI -> ReviewService: Запрос отзывов о товаре

ReviewService -> UI: Отдает отзывы

UI -> User: Показывает отзывы

User -> UI: Оставляет отзыв

UI -> ReviewService: Передает отзыв

ReviewService -> ReviewService: Сохраняет отзыв

ReviewService -> ProductRepo: Обновляет рейтинг товара

ProductRepo -> UI: Подтверждение обновления

UI -> User: Обновляет страницу

User -> UI: Запрашивает рекомендации

UI -> RecoModule: Запрос рекомендаций

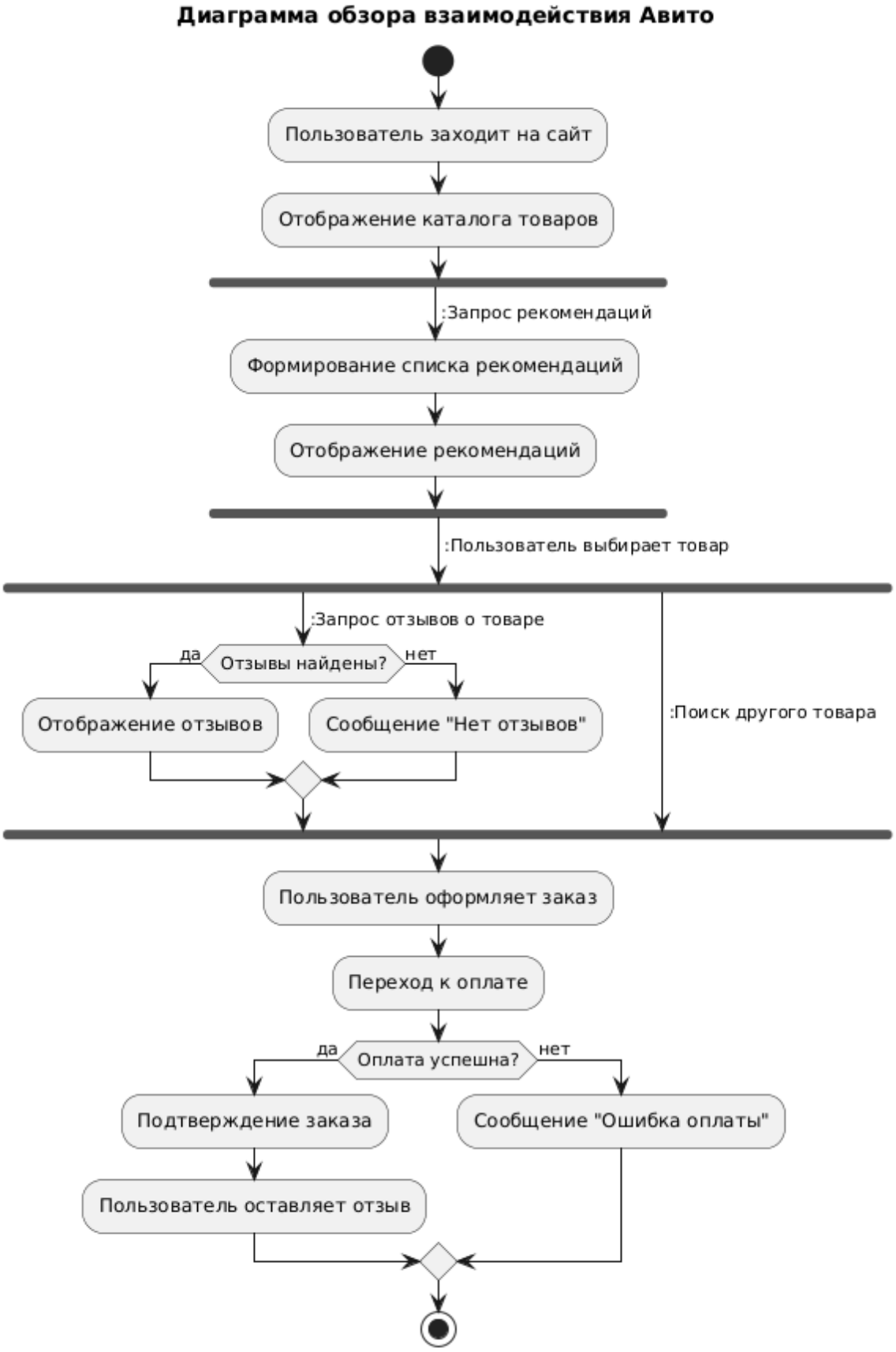
RecoModule -> UI: Формирует список рекомендаций

UI -> User: Показывает рекомендации

@enduml

9. Диаграмма обзоров взаимодействия

Фото:



Код:

```
@startuml

title Диаграмма обзора взаимодействия Авито

start

:Пользователь заходит на сайт;
:Отображение каталога товаров;

fork
   -> :Запрос рекомендаций;
   :Формирование списка рекомендаций;
   :Отображение рекомендаций;
end fork

   -> :Пользователь выбирает товар;
   fork
      -> :Запрос отзывов о товаре;
      if (Отзывы найдены?) then (да)
         :Отображение отзывов;
      else (нет)
         :Сообщение "Нет отзывов";
      endif
   fork again
      -> :Поиск другого товара;
   end fork

:Пользователь оформляет заказ;
:Переход к оплате;

if (Оплата успешна?) then (да)
```

```
   :Подтверждение заказа;
   :Пользователь оставляет отзыв;
else (нет)
   :Сообщение "Ошибка оплаты";
endif

stop

@enduml
```

10. Диаграмма коммуникаций

Фото:

Код:

```
@startuml

title Диаграмма коммуникации Авито


participant "Пользователь" as User

participant "Каталог товаров" as Catalog

participant "Система рекомендаций" as Recommender

participant "Система отзывов" as ReviewSystem

participant "Корзина" as Cart

participant "Оплата" as Payment

participant "Админ" as Admin


User -> Catalog : Просмотр списка товаров

User -> Recommender : Запрос рекомендаций

Recommender -> Catalog : Получение данных о товарах
```

Recommender -> User : Отображение рекомендованных товаров

User -> Catalog : Выбор товара

User -> ReviewSystem : Запрос отзывов

ReviewSystem -> Catalog : Получение отзывов

ReviewSystem -> User : Отображение отзывов

User -> Cart : Добавление товара в корзину

User -> Cart : Проверка содержимого корзины

User -> Payment : Оплата заказа

Payment -> User : Подтверждение покупки

Admin -> Catalog : Управление товарами

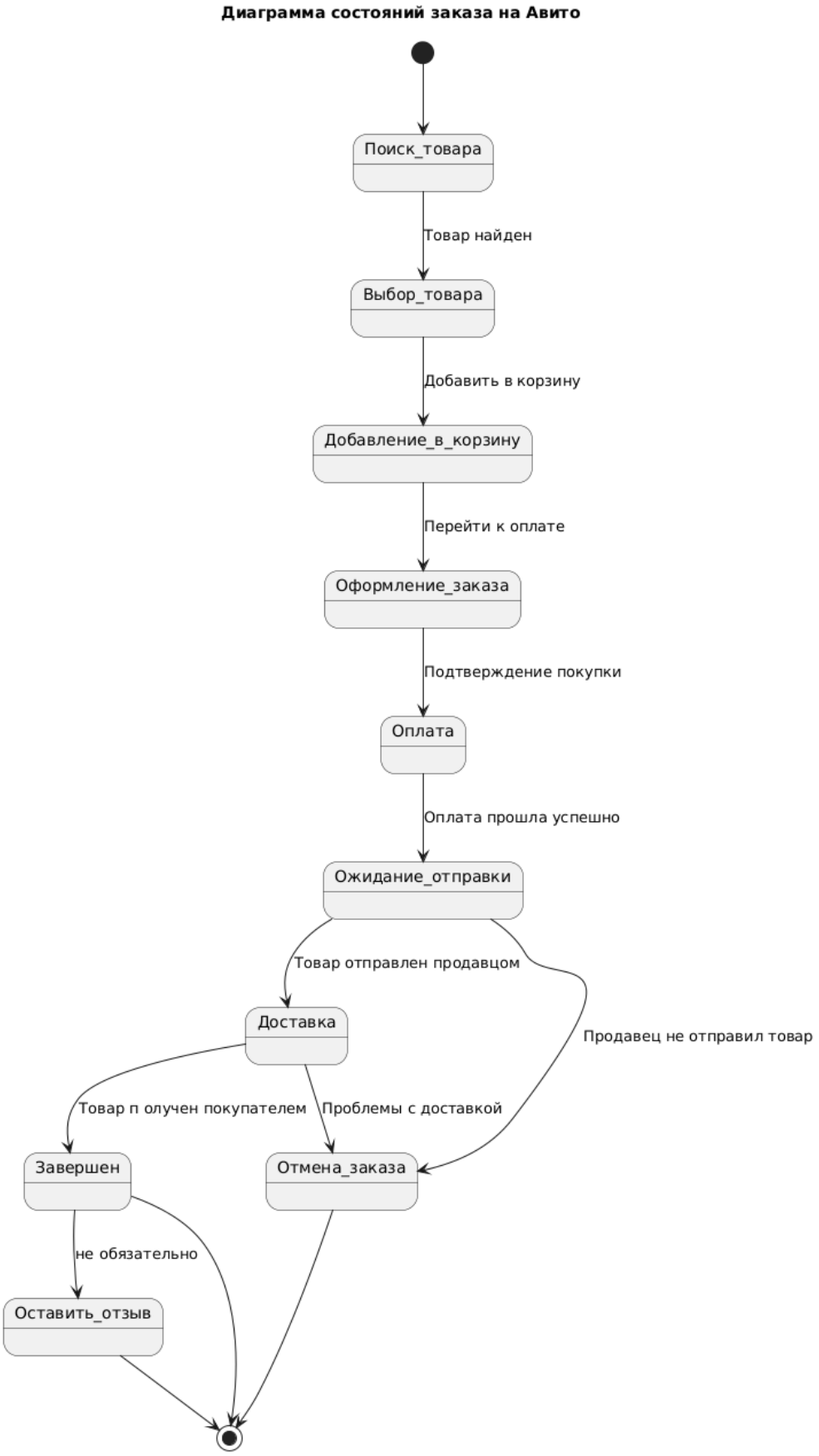
Admin -> ReviewSystem : Модерация отзывов

Admin -> Payment : Контроль транзакций

@enduml

11. Диаграмма состояний

Фото:

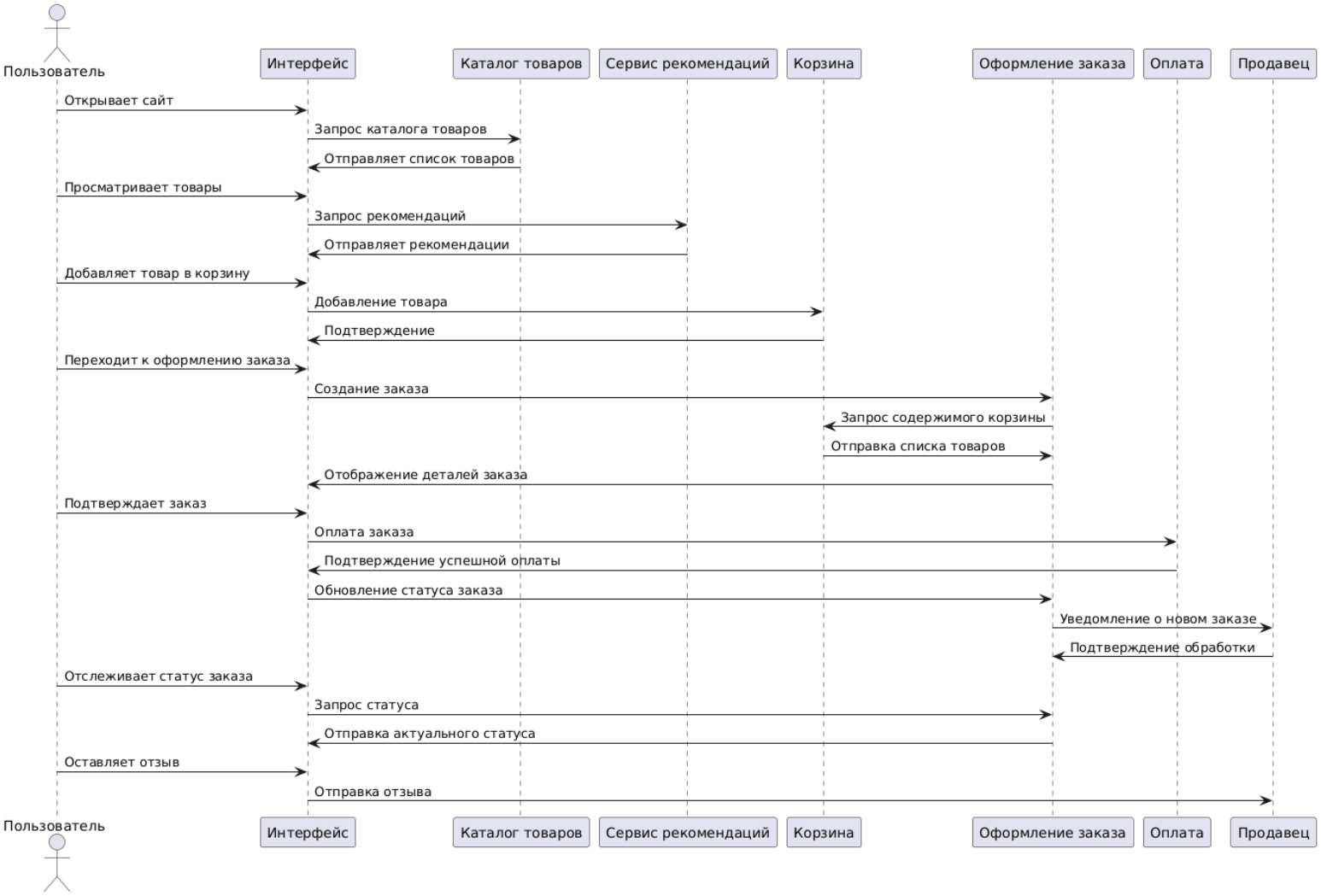


Код:

```
@startuml

title Диаграмма состояний заказа на Авито

[*] --> Поиск_товара

Поиск_товара --> Выбор_товара : Товар найден

Выбор_товара --> Добавление_в_корзину : Добавить в корзину

Добавление_в_корзину --> Оформление_заказа : Перейти к оплате

Оформление_заказа --> Оплата : Подтверждение покупки

Оплата --> Ожидание_отправки : Оплата прошла успешно

Ожидание_отправки --> Доставка : Товар отправлен продавцом

Доставка --> Завершен : Товар п олучен покупателем

Завершен --> Оставить_отзыв : не обязательно

Оставить_отзыв --> [*]

Завершен --> [*]

Ожидание_отправки --> Отмена_заказа : Продавец не отправил товар

Доставка --> Отмена_заказа : Проблемы с доставкой

Отмена_заказа --> [*]

@enduml
```

12. Диаграмма последовательностей

Фото:



Код:

```
@startuml

actor Пользователь

participant "Интерфейс" as UI

participant "Каталог товаров" as Catalog

participant "Сервис рекомендаций" as Recommender

participant "Корзина" as Cart

participant "Оформление заказа" as Order

participant "Оплата" as Payment

participant "Продавец" as Seller
```

Пользователь -> UI: Открывает сайт

UI -> Catalog: Запрос каталога товаров

Catalog -> UI: Отправляет список товаров

Пользователь -> UI: Просматривает товары

UI -> Recommender: Запрос рекомендаций

Recommender -> UI: Отправляет рекомендации

Пользователь -> UI: Добавляет товар в корзину

UI -> Cart: Добавление товара

Cart -> UI: Подтверждение

Пользователь -> UI: Переходит к оформлению заказа

UI -> Order: Создание заказа

Order -> Cart: Запрос содержимого корзины

Cart -> Order: Отправка списка товаров

Order -> UI: Отображение деталей заказа

Пользователь -> UI: Подтверждает заказ

UI -> Payment: Оплата заказа

Payment -> UI: Подтверждение успешной оплаты

UI -> Order: Обновление статуса заказа

Order -> Seller: Уведомление о новом заказе

Seller -> Order: Подтверждение обработки

Пользователь -> UI: Отслеживает статус заказа

UI -> Order: Запрос статуса

Order -> UI: Отправка актуального статуса

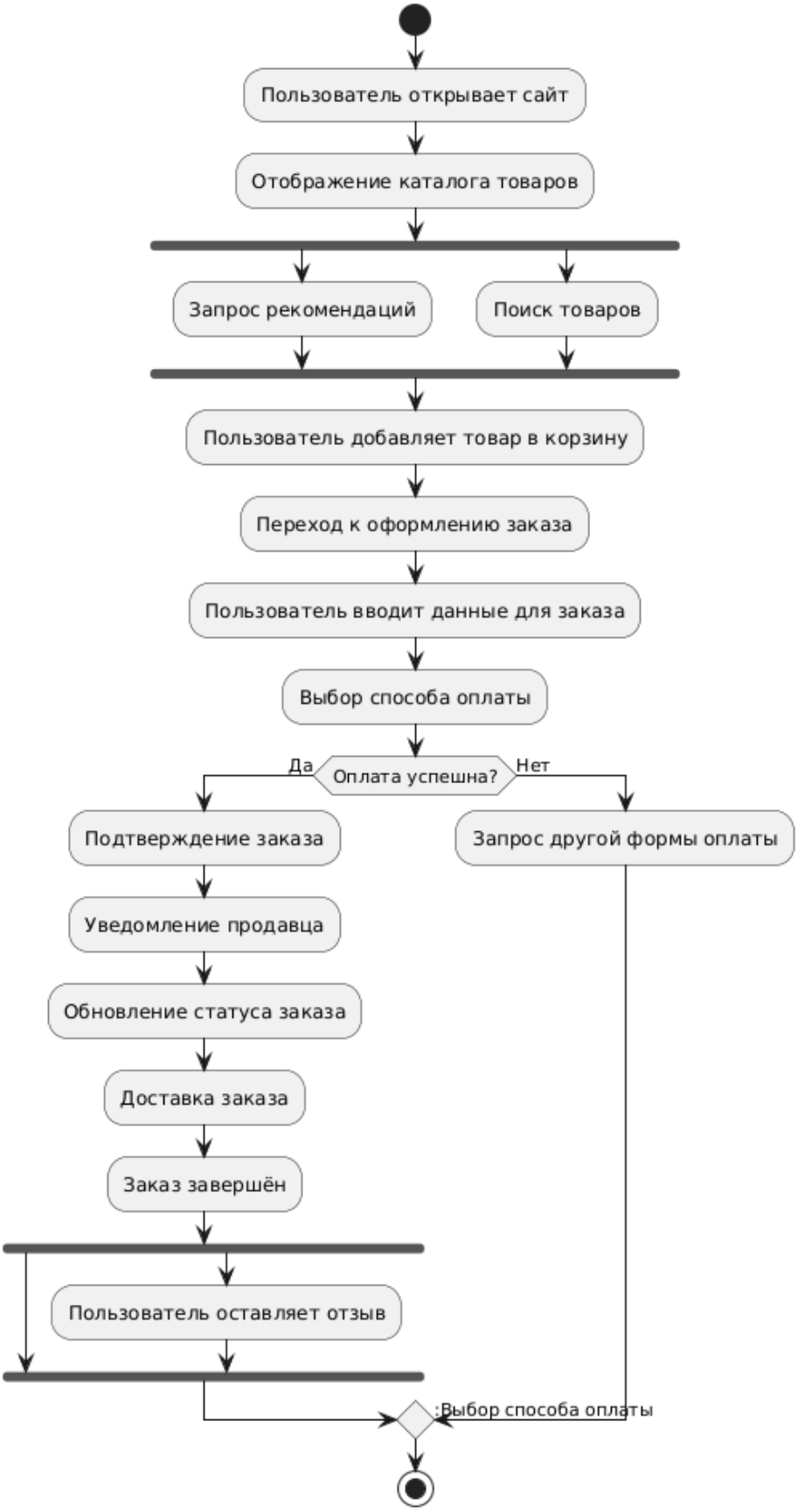
Пользователь -> UI: Оставляет отзыв

UI -> Seller: Отправка отзыва

@enduml

13. Диаграмма активности

Фото:



Код:

```
@startuml
start

:Пользователь открывает сайт;
:Отображение каталога товаров;
fork
   :Запрос рекомендаций;
fork again
   :Поиск товаров;
end fork

:Пользователь добавляет товар в корзину;
:Переход к оформлению заказа;
:Пользователь вводит данные для заказа;
:Выбор способа оплаты;

if (Оплата успешна?) then (Да)
   :Подтверждение заказа;
   :Уведомление продавца;
   :Обновление статуса заказа;
   :Доставка заказа;
   :Заказ завершён;
   fork
   fork again

      :Пользователь оставляет отзыв;
   end fork
else (Нет)
   :Запрос другой формы оплаты;
   -> :Выбор способа оплаты;
endif
```

```
stop
@enduml
```