МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

ЗАЩИТА ПРОЕКТА

по дисциплине
Системная и программная инженерия

Тема:
Разработка веб-приложения «Маркетплейс для жителей Африки»

Выполнили
Студент группы ИМБО-02-22 — Ким Кирилл Сергеевич
Макаров Арсений Сергеевич
Ломакин Дмитрий Владимирович
Смирнов Дмитрий Михайлович
Чахнин Михаил Анатольевич

Проверила — ассистент кафедры МОСИТ
Золотухина Мария Александровна

Состав команды и роли

Таблица 1 — Состав команды

Роль	ФИО	Обязанности
Руководитель	Ким К.С.	Управление проектом, контроль сроков
Аналитик	Ломакин Д.В.	Сбор требований, диаграммы (UML, BPMN, DFD)
Разработчик/Дизайнер	Чахнин М.А.	Реализация функционала, интерфейс
Технический писатель	Смирнов Д.М.	Документация
Тестировщик	Макаров А.С.	Тест-кейсы, нагрузочное тестирование

Описание проекта

Цель: Создание удобного маркетплейса для жителей Африки, объединяющего продавцов и покупателей.

Основные функции:

- Поиск и фильтрация товаров.
- Сравнение товаров.
- Оформление заказов с оплатой и доставкой.
- Личный кабинет для покупателей и продавцов.
- Аналитика продаж для продавцов.

Целевая аудитория:

- Покупатели (например, Абаси, 22 года, Каир).
- Продавцы (малый и средний бизнес).
- Администраторы платформы.

Цели и требования

Функциональные требования:

- Поиск товаров (фильтры, сортировка).
- Корзина и оформление заказа.
- Личный кабинет (история заказов, настройки).
- Управление товарами для продавцов.

Нефункциональные требования:

- Производительность: время отклика ≤ 500 мс.
- Безопасность: шифрование AES-256, защита от DDoS.
- Локализация: поддержка русского, английского, африканских языков.

Архитектура системы

1. Backend (Django) – основной серверный компонент, отвечающий за обработку бизнес-логики, взаимодействие с базой данных и предоставление API.
2. База данных (PostgreSQL) – реляционная база данных, обеспечивающая надежное хранение данных.
3. Контейнеризация (Docker) – используется для обеспечения гибкости развертывания и изоляции зависимостей.
4. Обратный прокси-сервер (Nginx) – используется для маршрутизации запросов и балансировки нагрузки.
5. Система кэширования (Redis) – применяется для ускорения работы системы за счет хранения часто запрашиваемых данных в памяти.

Выбор методологии

Для организации процесса разработки был выбран подход Agile. Он обеспечивает гибкое реагирование на изменения требований и активное взаимодействие между членами команды и заказчиком, это наиболее необходимо при адаптации отечественных продуктов под иные рынки и страны. Благодаря этому минимизируются риски, а итерационный подход позволяет быстро вносить улучшения в продукт. Agile позволяет выпускать рабочие версии проекта на каждом этапе, адаптируя план по мере необходимости. Для нашего проекта, написанного на Django и использующего PostgreSQL и Docker, Agile обеспечивает оптимальную гибкость и управляемость.

Иные методологии были отклонены по следующим причинам. Waterfall требует полного определения требований на начальном этапе, не допускает изменений в процессе и выдаёт конечный результат только по завершению всех этапов. Это делает её неэффективной в условиях постоянных изменений требований и ограниченного времени. Итерационные, инкрементные и спиральные модели, обладают большей гибкостью по сравнению с Waterfall, но предполагают более жёсткую структуру, меньшую степень вовлечения заказчика и не обеспечивают такого уровня адаптивности, как Agile.

Наиболее значимые риски проекта

Таблица 2 — Описание рисков

№	Риск	Важность (1-10)	Последствия (1-10)	Важность (В*П)
1	Изменение требований клиента	4	6	24
2	Недостаточная коммуникация внутри команды	2	7	14
3	Неполнота или плохая документация	7	5	35
4	Технические проблемы с сервером	9	9	81

Наиболее значимые риски проекта

Таблица 3 — Матрица рисков

Уровень угрозы \ Вероятность	Несущественные (1–2)	Низкие (3–4)	Средние (5–6)	Существенные (7–8)	Катастрофические (9–10)
Весьма вероятно (9–10)	–	–	–	–	4
Вероятно (7–8)	–	–	–	–	–
Возможно (5–6)	–	–	–	3	–
Маловероятно (3–4)	–	–	1	–	–
Крайне маловероятно (1–2)	–	–	–	2	–

Наиболее значимые риски проекта

Таблица 4 — План реагирования

№	Риск	Стратегия реагирования	Основной план реагирования	Запасной план
1	Изменение требований клиента	Минимизация	Проводить регулярные встречи с клиентом, уточнять и документировать изменения требований, согласовывать изменения с командой.	Заложить резерв времени для переработки и корректировки проекта.
2	Недостаточная коммуникация внутри команды	Минимизация	Организовать ежедневные/еженедельные планерки, использовать современные системы обмена сообщениями (Slack, Microsoft Teams) и вести протоколы встреч.	Провести внеплановую сессию обмена информацией и назначить ответственного за коммуникацию.
3	Неполнота или плохая документация	Минимизация	Внедрить систему ведения документации (Confluence, Wiki), регулярно обновлять и проводить ревизию документации.	Назначить ответственного за документацию и провести внеплановое обновление ключевых разделов.

Описание стека технологий

1. Backend (Django): Надежный и мощный веб-фреймворк на языке Python, предоставляющий встроенные инструменты для разработки REST API (Django REST Framework), удобную ORM и высокий уровень безопасности.
2. База данных (PostgreSQL): Высокопроизводительная реляционная база данных с поддержкой ACID, хорошо масштабируемая и поддерживающая сложные запросы.
3. Кэширование (Redis): Ключевая NoSQL база данных, используемая для кэширования и хранения сессий пользователей.
4. Развертывание (Docker + AWS): Позволяет изолировать зависимости, упрощает развертывание и обеспечивает консистентность окружения.
5. Обратный прокси-сервер (Nginx): Позволяет изолировать зависимости, упрощает развертывание и обеспечивает консистентность окружения.

Диаграммы процессов проекта

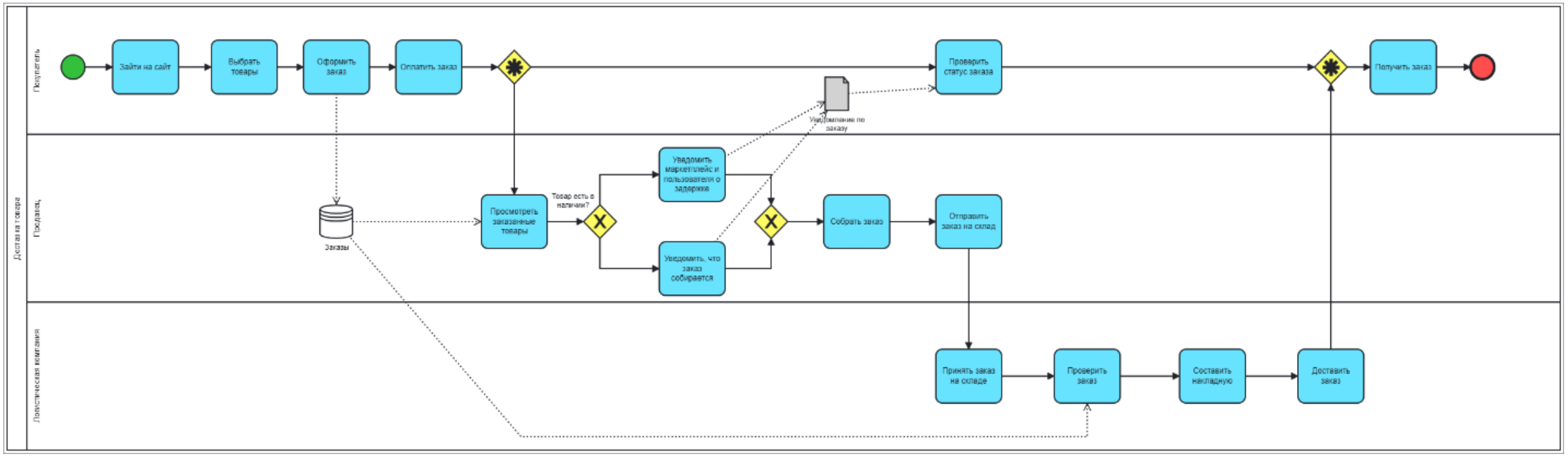


Рисунок 1 — BPMN диаграмма

Диаграммы процессов проекта

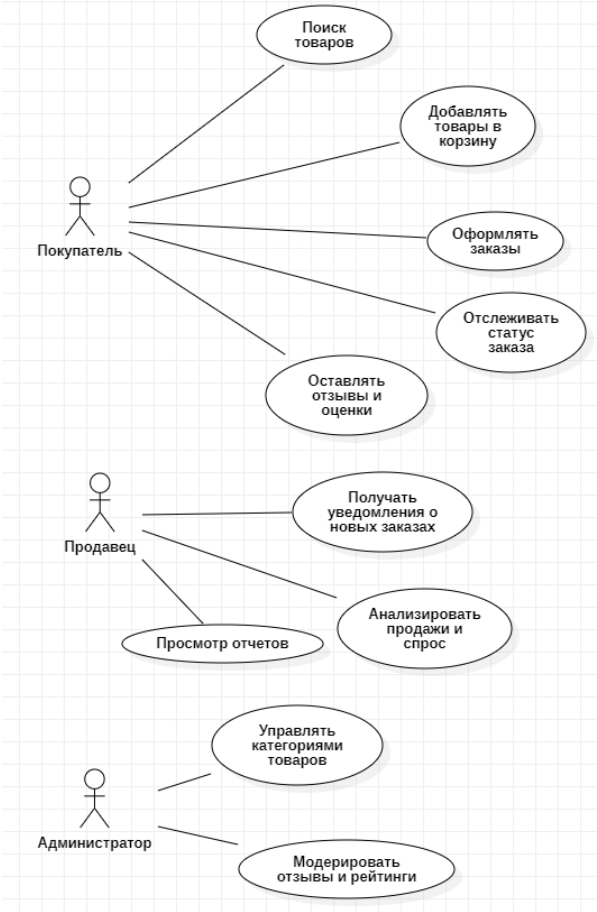


Рисунок 2 — Use Case диаграмма

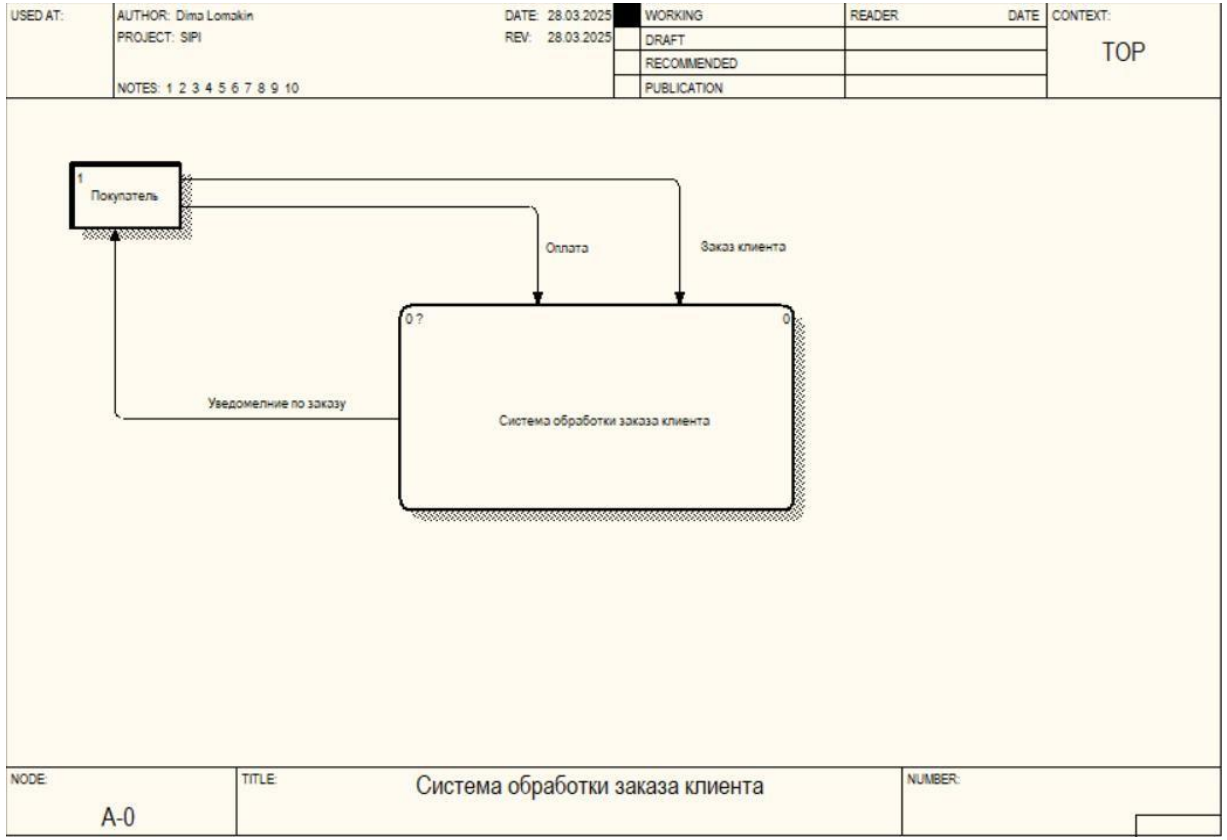


Рисунок 3 — Верхний уровень DFD диаграммы

Диаграммы процессов проекта

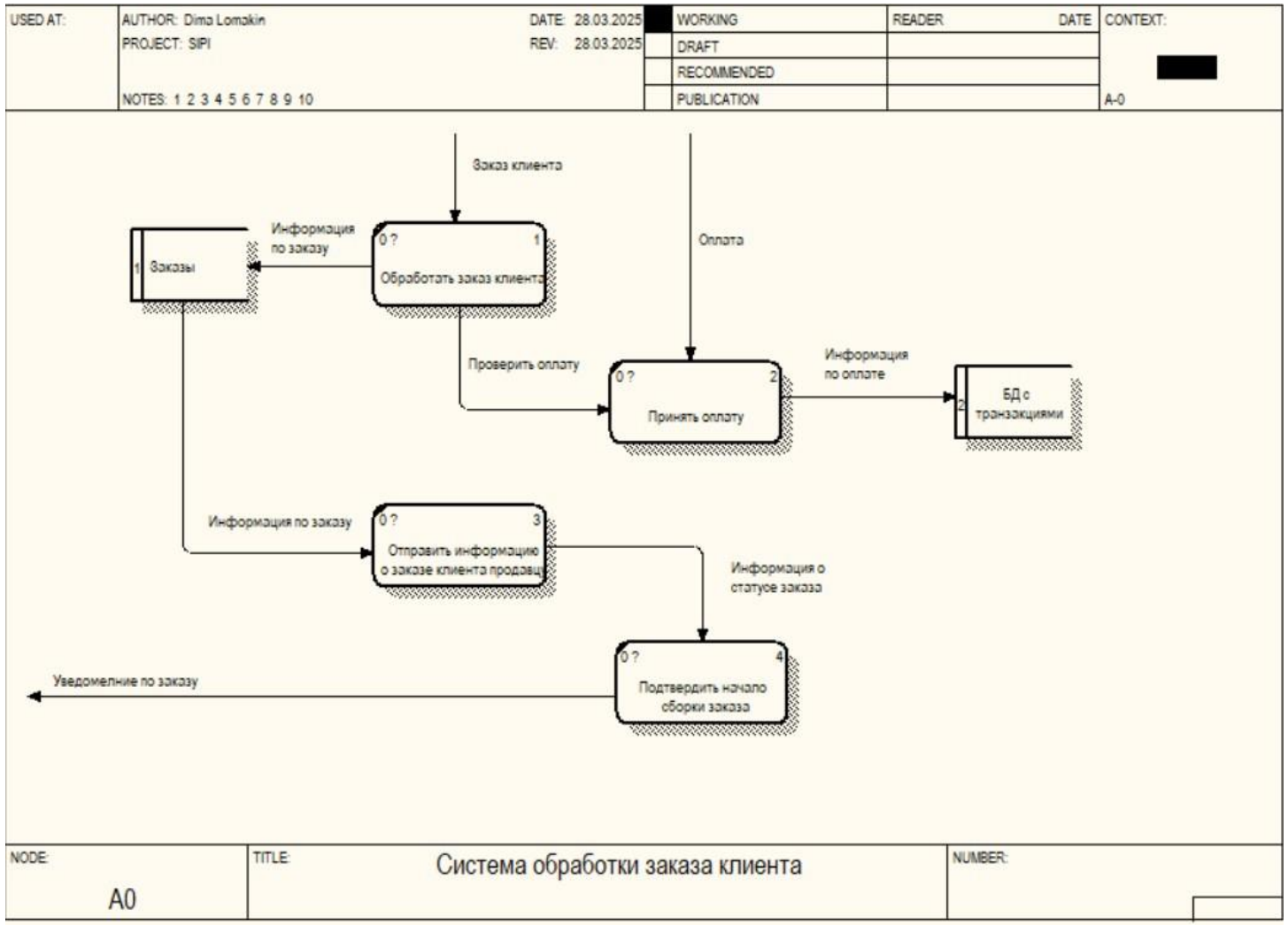


Рисунок 4 — Декомпозиция верхнего уровеня DFD диаграммы

Описание функционала приложения

Основные функции:

- Поиск по категориям и фильтрам.
- Корзина с изменением количества товаров.
- Отслеживание статуса заказа

Описание функционала приложения

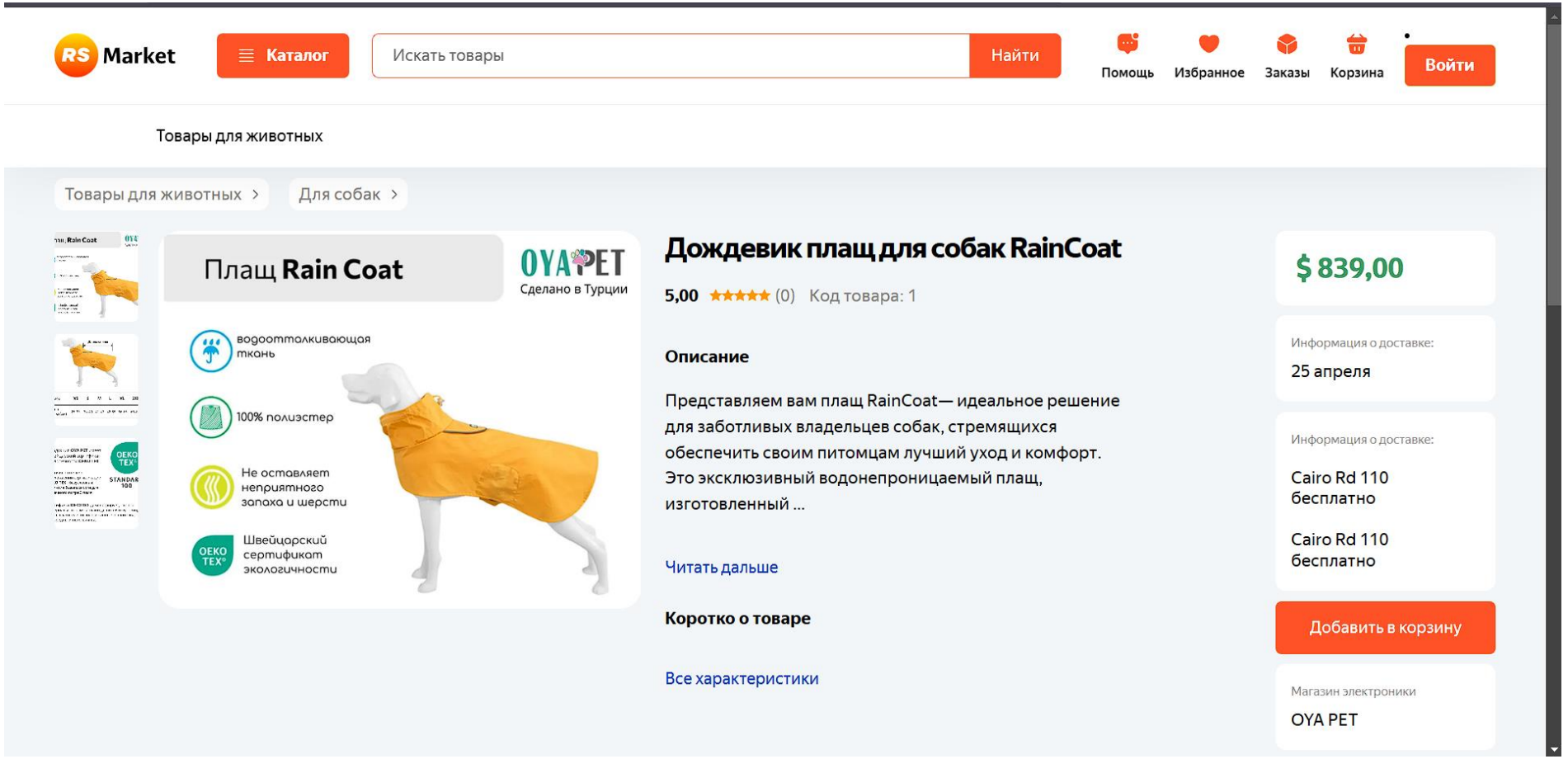


Рисунок 5 — Поиск товара

Описание функционала приложения

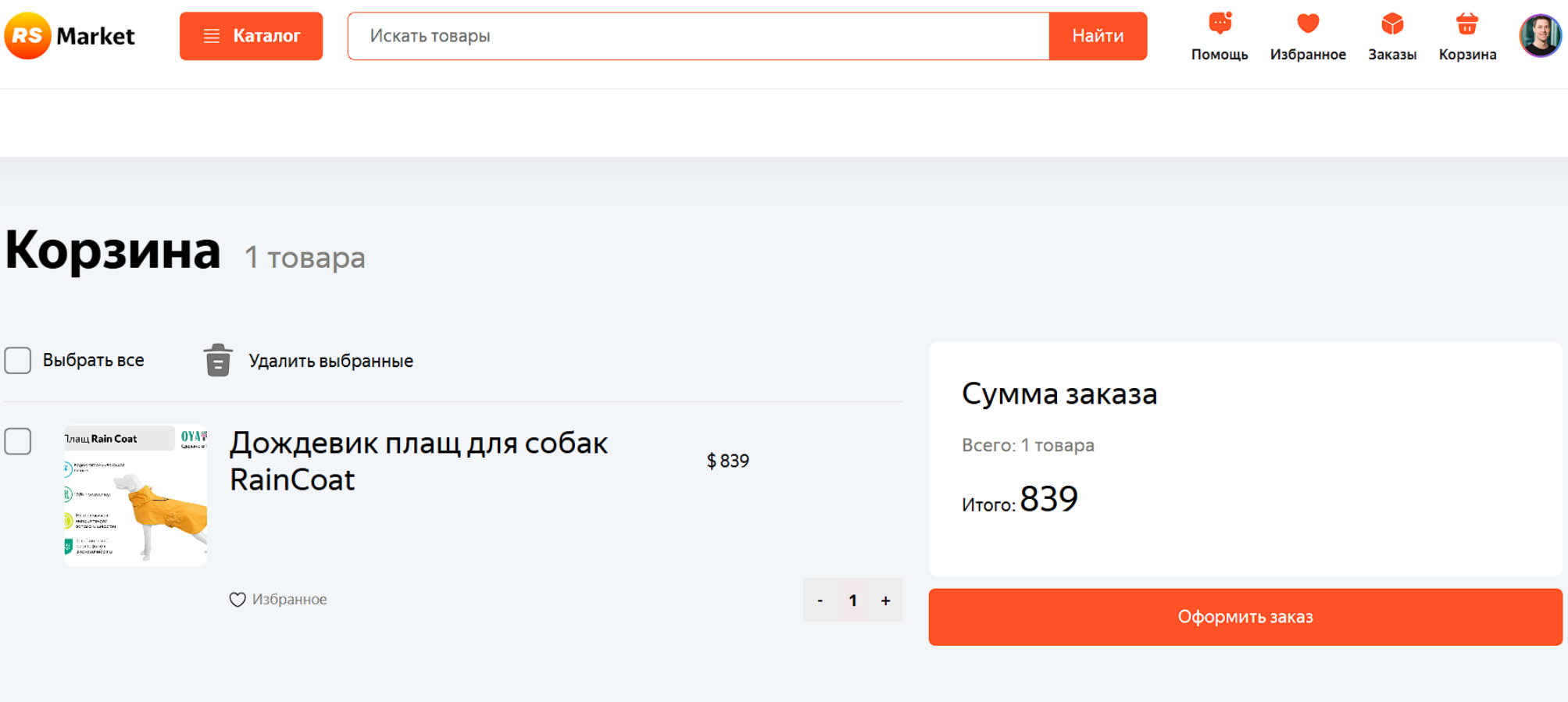


Рисунок 6 — Страница корзины

Описание функционала приложения

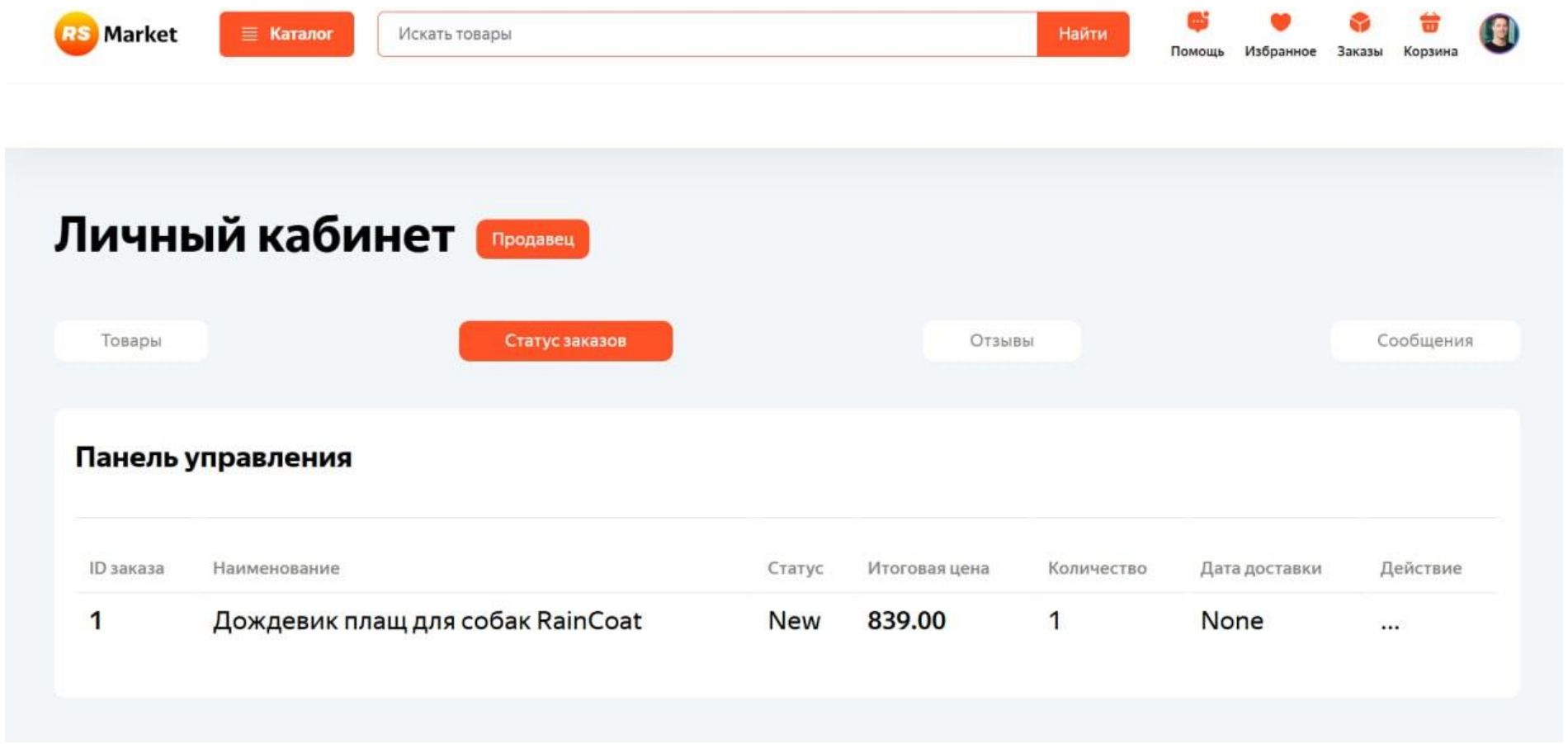


Рисунок 7 — Отслеживание товара

Тестирование

Инструменты:

- Selenium – автоматизация UI-тестов.
- JMeter – нагрузочное тестирование (10 000 RPS).
- OWASP ZAP – проверка безопасности.

Результаты:

- 100% тест-кейсов пройдены.
- Время отклика: 500 мс при пиковой нагрузке.

Развертывание

Для развёртывания веб-приложения используется платформа **Coolify**, которая обеспечивает автоматическую настройку и управление инфраструктурой.

1. Приложение и база данных упакованы в Docker-контейнеры.
2. Создан docker-compose.yml для запуска всех сервисов.
3. Репозиторий подключён к Coolify — при каждом пуше происходит автоматическое обновление.
4. Coolify разворачивает контейнеры, настраивает переменные окружения и управляет сетью.
5. Приложение доступно через автоматически настроенный домен и HTTPS.

План модернизации

В качестве модификации нашего продукта нами был добавлен сервис для развертывания нашего приложения – coolify.

В следствие данной модификации были внесены некоторые изменения в нашу систему, а именно:

- инструменты разработки;
- матрица требований;
- ГОСТ 34.602-2020;
- автоматического тестирование системы;

Документация разработчика

```
# Django Project (Dockerized)

## 🚀 Быстрый старт

### 1. Клонируйте репозиторий

```bash
git clone [https://____.git](https://github.com/rprescott2/AfricaShop)
cd имя-папки-проекта
```

2. Запуск проекта в Docker

Для развёртывания проекта используйте команду:

```bash
docker compose up --build
```

> ⚠ Убедитесь, что Docker и Docker Compose установлены на вашей машине.

3. Первичная инициализация данных

После запуска контейнеров выполните команду инициализации данных:

```bash
docker exec -it app python -m manage seed
```
```

Рисунок 8 — README-файл часть 1

```
> Где `app` — это имя контейнера Django-приложения, указанное в `docker-compose.yml`.

Команда создаёт базовые сущности и подготавливает проект к работе.

### 4. Доступ к приложению

После запуска проект будет доступен по адресу:

📍 [http://localhost:8000](http://localhost:8000)

---

## 🛠 Стек технологий

- Django (Backend)
- PostgreSQL (База данных)
- Docker + Docker Compose (Контейнеризация)

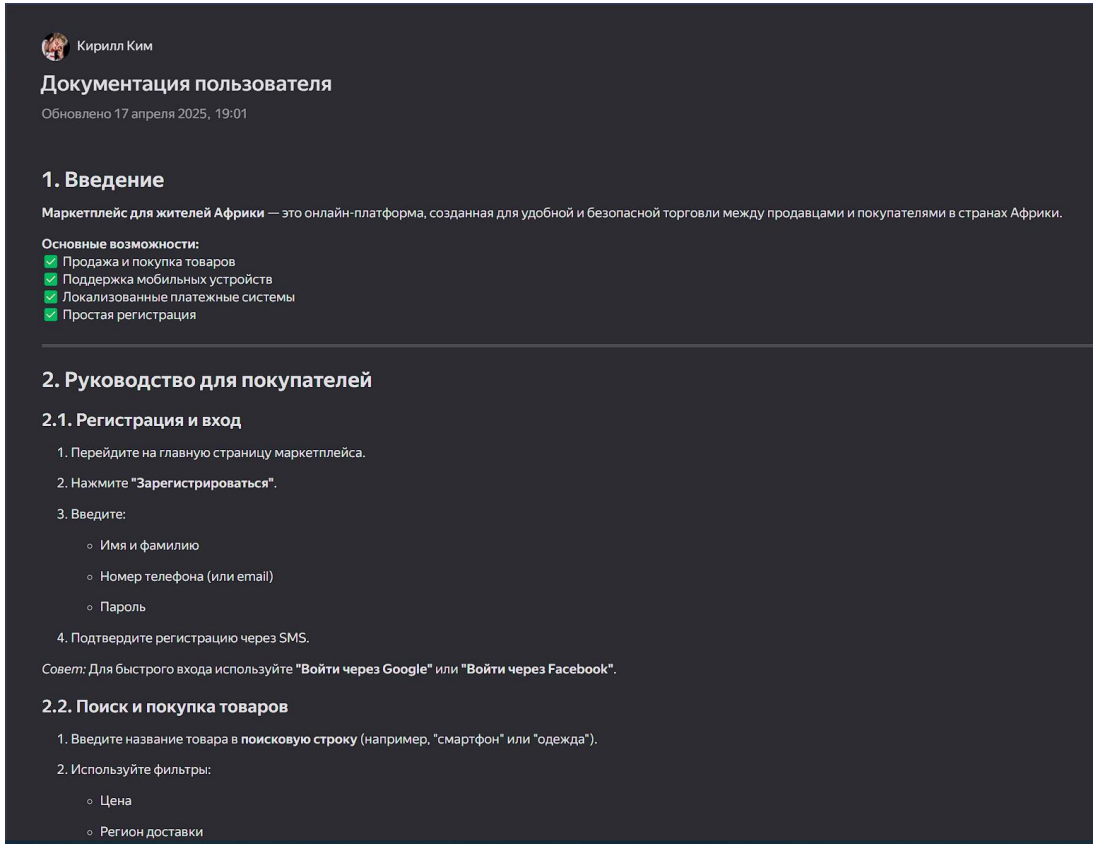
---

## 📁 Структура

```
.
├── manage.py
├── project_name/
├── app/
├── Dockerfile
├── docker-compose.yml
└── README.md
```
```

Рисунок 9 — README-файл часть 2

Документация пользователя



Кирилл Ким

Документация пользователя

Обновлено 17 апреля 2025, 19:01

1. Введение

Маркетплейс для жителей Африки — это онлайн-платформа, созданная для удобной и безопасной торговли между продавцами и покупателями в странах Африки.

Основные возможности:
- ✅ Продажа и покупка товаров
- ✅ Поддержка мобильных устройств
- ✅ Локализованные платежные системы
- ✅ Простая регистрация

2. Руководство для покупателей

2.1. Регистрация и вход

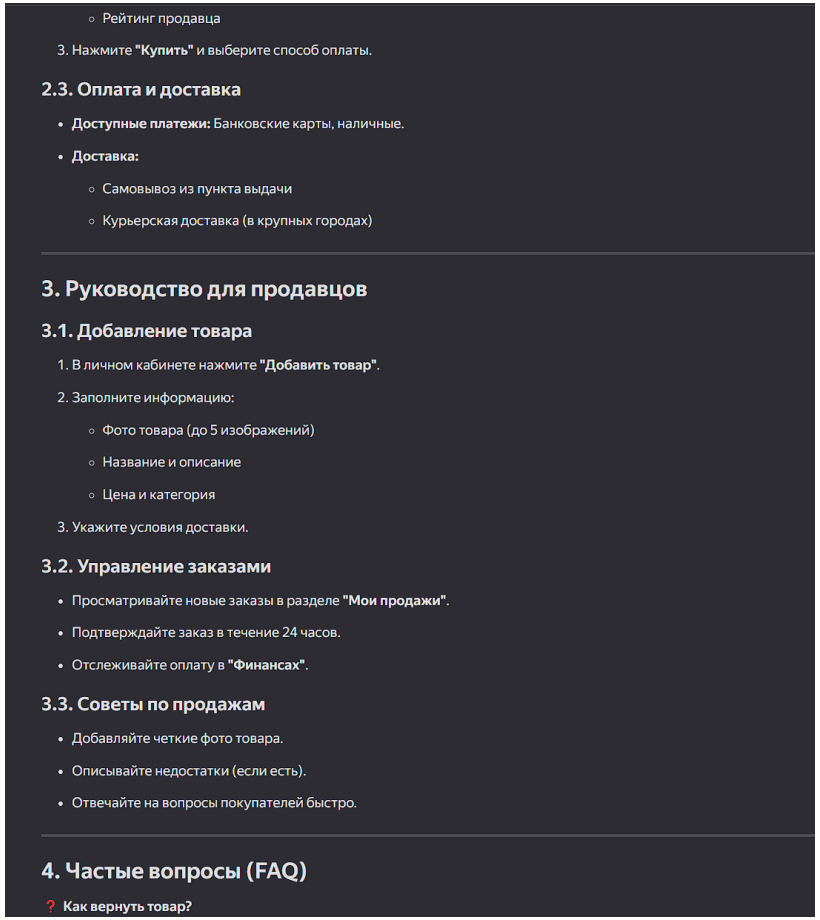
1. Перейдите на главную страницу маркетплейса.
2. Нажмите **"Зарегистрироваться"**.
3. Введите:
   - Имя и фамилию
   - Номер телефона (или email)
   - Пароль
4. Подтвердите регистрацию через SMS.

Совет: Для быстрого входа используйте **"Войти через Google"** или **"Войти через Facebook"**.

2.2. Поиск и покупка товаров

1. Введите название товара в **поисковую строку** (например, "смартфон" или "одежда").
2. Используйте фильтры:
   - Цена
   - Регион доставки

Рисунок 10 — Документация пользователя часть 1



   - Рейтинг продавца
3. Нажмите **"Купить"** и выберите способ оплаты.

2.3. Оплата и доставка

- **Доступные платежи:** Банковские карты, наличные.
- **Доставка:**
  - Самовывоз из пункта выдачи
  - Курьерская доставка (в крупных городах)

3. Руководство для продавцов

3.1. Добавление товара

1. В личном кабинете нажмите **"Добавить товар"**.
2. Заполните информацию:
   - Фото товара (до 5 изображений)
   - Название и описание
   - Цена и категория
3. Укажите условия доставки.

3.2. Управление заказами

- Просматривайте новые заказы в разделе **"Мои продажи"**.
- Подтверждайте заказ в течение 24 часов.
- Отслеживайте оплату в **"Финансах"**.

3.3. Советы по продажам

- Добавляйте четкие фото товара.
- Описывайте недостатки (если есть).
- Отвечайте на вопросы покупателей быстро.

4. Частые вопросы (FAQ)

❓ **Как вернуть товар?**

Рисунок 11 — Документация пользователя часть 2

Достоинства и недостатки приложения/системы

Основные преимущества:

- Лучшая адаптация под африканский рынок
- Оптимизация для слабого интернета
- Простота интерфейса

Недостатки:

- Ограниченное количество платежных систем

Видео

https://drive.google.com/file/d/1-nXt4s_CtA7m85Xex_IJ6zEZ3tFXW2EX/view?usp=sharing

https://drive.google.com/file/d/1Gj9CYwZJU0yaolewtPtxZwheY-vYaBJJ/view?usp=sharing

Заключение

Итоги проекта:

1. Все ключевые функции реализованы
2. Система соответствует требованиям по производительности
3. Успешно пройдены все тесты

Дальнейшее развитие:

1. Добавление мобильного приложения
2. Интеграция с местными платежными системами
3. Расширение аналитики для продавцов

Вывод:

Проект успешно решает поставленные задачи и готов к промышленной эксплуатации. Выбранные архитектурные решения и технологии доказали свою эффективность в ходе разработки и тестирования.

Спасибо за внимание!