МИНОБРНАУКИ РОССИИ

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

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Отчет по практическим работам №9-10

по дисциплине «Системная и программная инженерия»

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

Проверила: ассистент кафедры МОСИТ
Золотухина М. А.

2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В ходе работы по организации процесса разработки веб-приложения «Маркетплейс для жителей Африки» и созданию необходимой документации.

Ключевые результаты:

1. Организация разработки

• Для проекта была выбрана гибкая методология (Agile/Scrum), так как она позволяет эффективно адаптироваться к изменениям, вовлекать заказчика в процесс и обеспечивать постепенное улучшение продукта.

• Настроена система контроля версий (Git) для командной работы.

2. Документирование проекта

• Была создана документация разработчика с помощью инструментов Doxygen или Sphinx, что упрощает поддержку кода и его дальнейшее развитие.

• Для пользователей разработана инструкция по эксплуатации в формате Wiki (FANDOM, MediaWiki или GitHub Wiki), что делает приложение более доступным и понятным для конечных пользователей.

Таким образом, применение современных подходов к разработке и документированию обеспечило основу для надежной и масштабируемой платформы, адаптированной под потребности африканского рынка.

ПРАКТИЧЕСКАЯ РАБОТА №9

Организация разработки

1. Выбор методологии управления процессом разработки

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.

Методология Lean ориентирована в первую очередь на минимизацию затрат и упрощение процессов, что делает её хорошей для предварительного тестирования гипотез и создания MVP, но недостаточно структурированной для полноценного процесса командной разработки. Применение Lean без опоры на чёткие принципы гибкой методологии может привести к чрезмерному упрощению продукта и утрате его ценности, что не будет соответствовать конечной цели проекта

2. Git-репозиторий проекта

Удалённый репозиторий проекта размещён на сервисе:

https://github.com/rprescott2/AfricaShop

Система контроля версий — Git. Используется для ведения истории изменений и совместной работы над проектом. Основной веткой разработки является main/master.

3. Используемые инструменты разработки

В процессе разработки используются следующие инструменты:

- Язык программирования: Python
- Фреймворк: Django
- СУБД: PostgreSQL
- Контейнеризация: Docker
- Система контроля версий: Git
- IDE/редактор кода: PyCharm
- Средства управления зависимостями: pip

Прочее: Docker Compose для управления многоконтейнерной архитектурой.

ПРАКТИЧЕСКАЯ РАБОТА №10

Документирование разработки ПО

1. Документация разработчика

Для разработчик была подготовлена вики-документация на платформе GitHub, которая реализована с помощью .md формата:

https://github.com/rprescott2/AfricaShop/blob/main/README.md

2. Документация пользователя

Для пользователей была подготовлена вики-документация на платформе:

https://wiki.yandex.ru/users/super-ironman00/africa-marketplace/

В документации отражены основные функции приложения, а именно:

- Установка и запуск проекта в Docker
- Регистрация и аутентификация пользователей
- Как вносить данные и где их просматривать

Документация пользователя и её авторы представлены на рисунках 10.1-10.4.

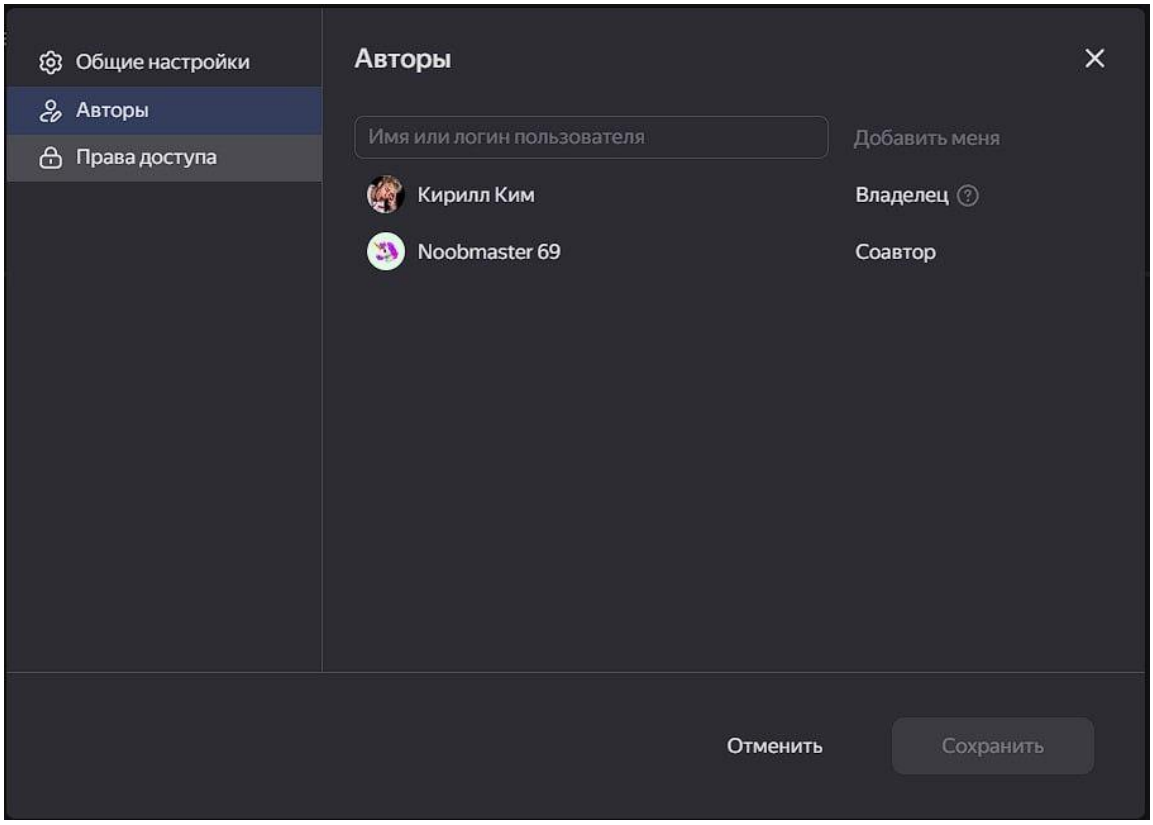
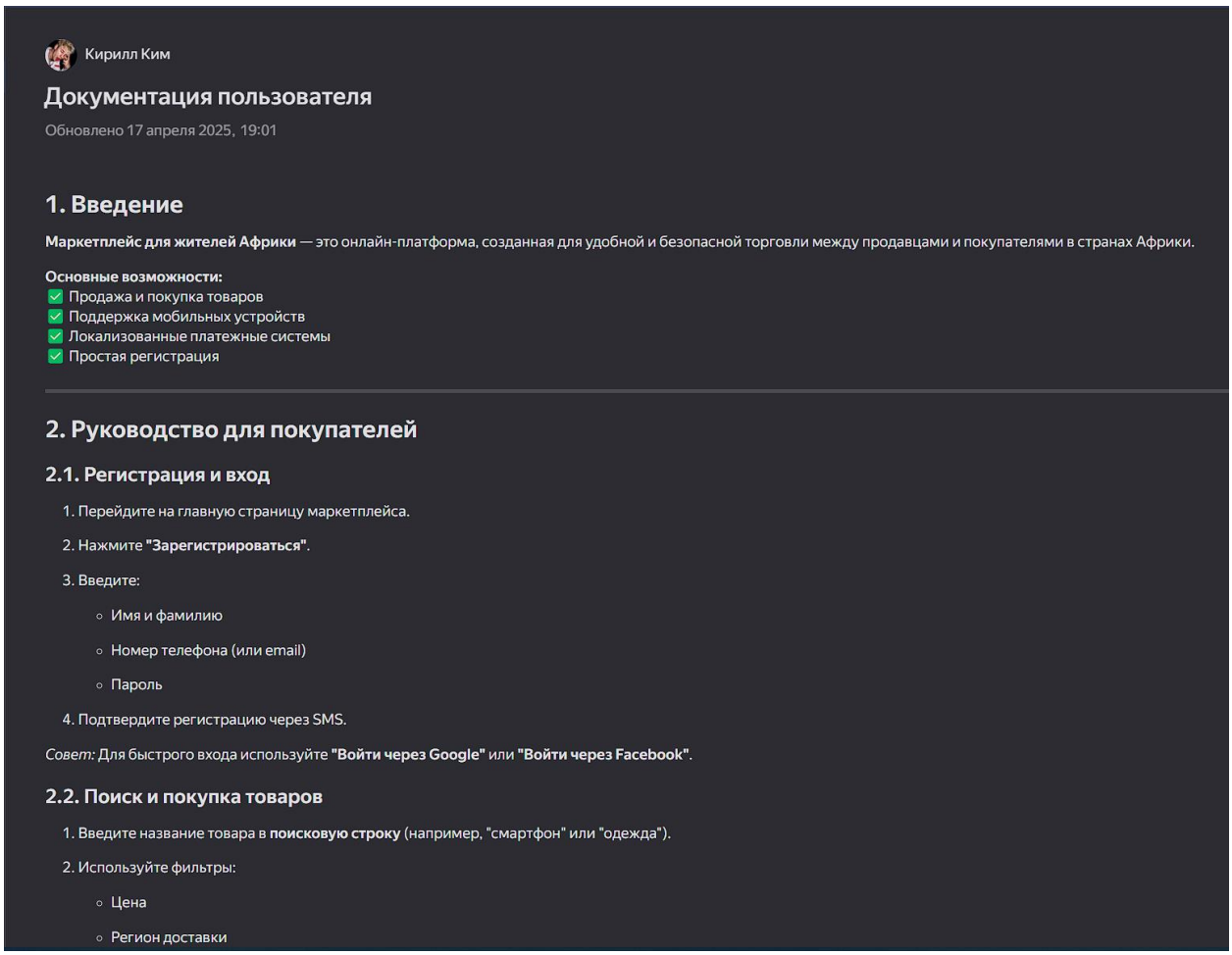


Рисунок 10.1 – Авторы документации пользователя



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 10.2 – 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 10.3 – Документация пользователя часть 2

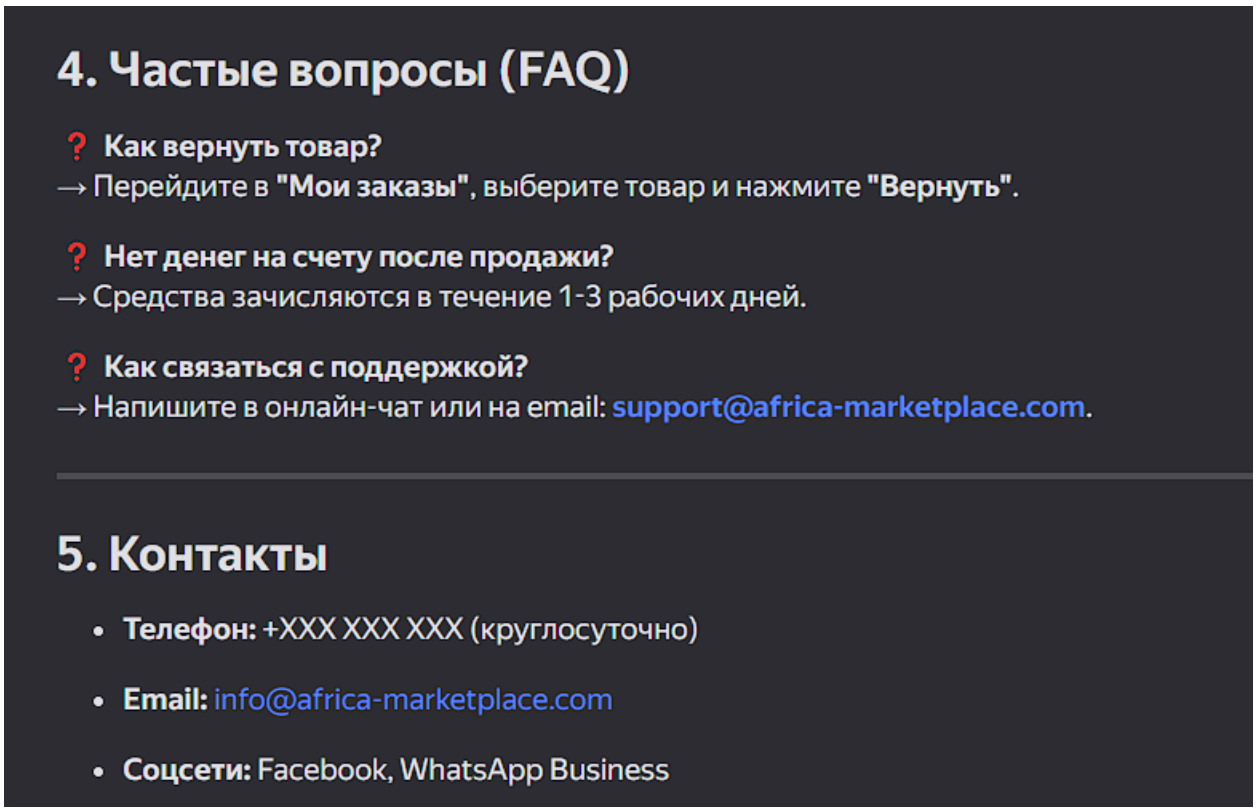


Рисунок 10.4 – Документация пользователя часть 1

Ниже, на рисунках 10.5-10.7, представлен README-файл.

```
# 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 10.5 – 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 10.6 – README-файл часть 2

````
## 🔧 Полезные команды

- Остановить контейнеры:
  ```bash
  docker compose down
  ```

- Повторно пересобрать образы:
  ```bash
  docker compose up --build
  ```

- Получить доступ к контейнеру `app`:
  ```bash
  docker exec -it app bash
  ```
````

Рисунок 10.7 – README-файл часть 3

Приложение А - Тесты. Анализаторы кода.

В данном приложении подробно рассматриваются принципы и особенности статического и динамического анализа кода, а также их практическое применение в рамках разработки программного обеспечения. Совместное использование этих методов позволяет существенно повысить качество конечного продукта, выявляя ошибки как до, так и во время выполнения программы.

Статический анализ кода

Статический анализ кода – это метод автоматизированного анализа исходного кода без его запуска. Основные цели статического анализа:

- Выявление синтаксических и стилистических ошибок.
- Обнаружение потенциальных проблем, таких как утечки памяти, неправильное использование переменных или нарушение стандартов кодирования.
- Проведение проверки на соответствие код-стандартам и внутренним требованиям проекта.

Методы статического анализа основываются на анализе исходного текста, синтаксическом и семантическом разборе кода, что позволяет обнаружить ошибки ещё до запуска программы. Важно отметить, что статический анализ не способен выявить ошибки, связанные с логикой работы программы при реальном выполнении (например, ошибки, проявляющиеся при определённых данных).

При помощи инструментов вроде ESLint, Pylint и SonarQube разработчики могут оперативно выявить расхождения с код-стандартами и исправить их до этапа компиляции или выполнения. Обычно такие результаты отражают как ошибки форматирования и структурные недочёты, так и потенциальные утечки памяти или неправильное использование переменных. Однако важно понимать, что статический анализ не может гарантировать обнаружение всех логических ошибок, возникающих в результате реального исполнения программы.

Динамический анализ кода

Динамический анализ кода – это метод исследования программного обеспечения в процессе его исполнения. В отличие от статического анализа, динамический анализ позволяет оценить поведение приложения в реальных условиях, выявляя ошибки, которые проявляются только во время работы программы. Основные цели динамического анализа:

- Выявление утечек памяти, ошибок в управлении ресурсами и неправильной обработки исключений.
- Анализ производительности, выявление узких мест, профилирование использования памяти и времени выполнения.
- Тестирование корректности взаимодействия различных компонентов системы.

Этот метод позволяет наблюдать за поведением приложения в условиях, приближенных к реальным, и выявлять ошибки, которые не видны при статическом разборе. Используя инструменты, такие как Burp Suit или Java VisualVM, можно обнаружить утечки памяти, проблемы с управлением ресурсами или ошибки обработки исключений.

Заключение

Комплексное использование обоих методов анализа существенно повышает качество разработки, позволяя оперативно находить и устранять как типичные ошибки, так и более сложные дефекты, проявляющиеся в реальных условиях эксплуатации. Это, в свою очередь, снижает затраты на последующее исправление ошибок и повышает надёжность программного обеспечения.

ВЫВОД

Таким образом, в обязанности менеджера проекта входило выстраивать этапы работы, благодаря чему команда всегда видела текущие задачи, их статус (в очереди, в работе, выполнено) и дальнейшие шаги.

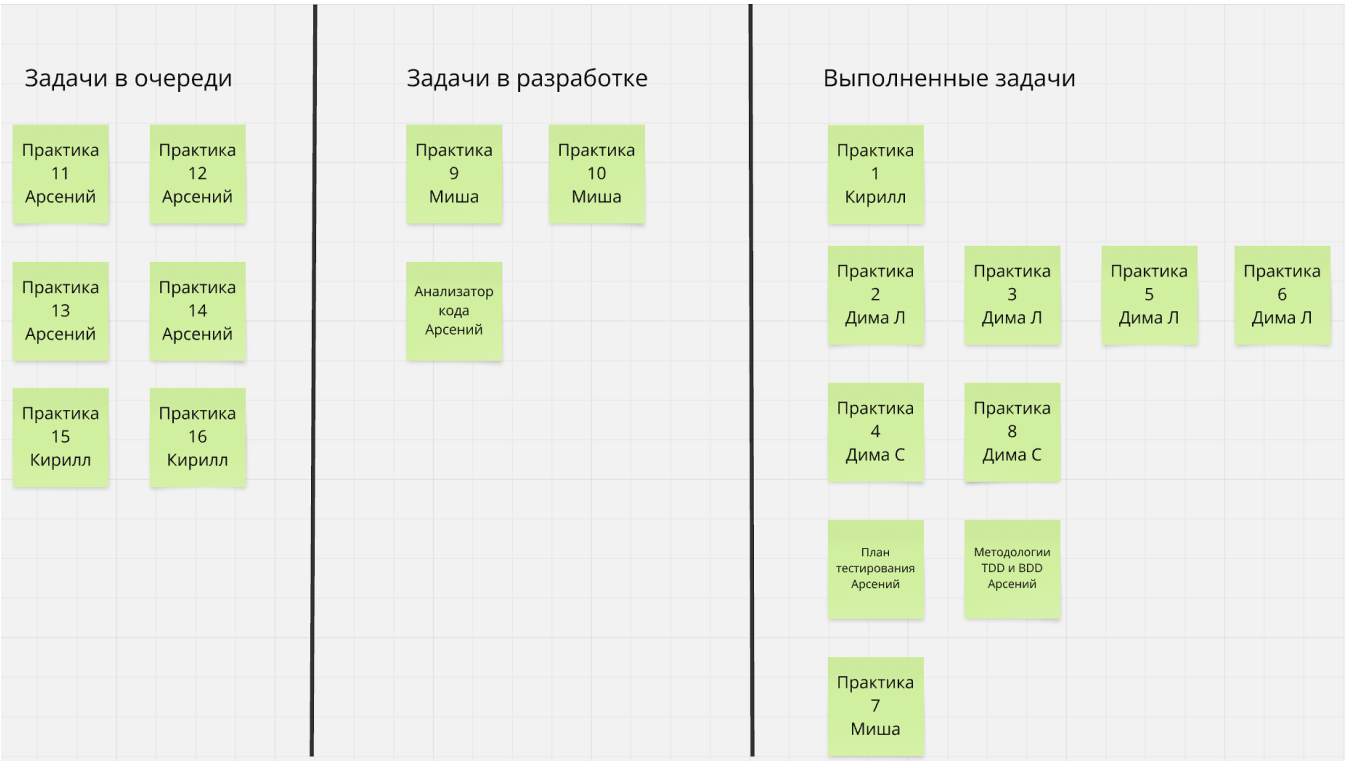


Рисунок А.1 – Список задач

Аналитик определил ключевые требования для пользователя.

Разработчик/Дизайнер разработал интерфейс, настроил репозиторий, начал реализацию функционала и документирование кода.

Тестировщик составил план проверки функционала на каждом этапе и написал анализатор кода.

Технический писатель создал документацию.

Также были проведены несколько онлайн-встреч, для резюмирования, что мы сделали.

Таблица А.1 – Даты конференции

Встречи	Кто присутствовал?	Дата
Встреча №1	Руководитель, Разработчик (Дизайнер), Тестировщик	15.04.2025
Встреча №2	Руководитель, Аналитик, Разработчик (Дизайнер), Тестировщик, Технический писатель	16.04.2025

После проведенных конференции список задач уменьшился и стал таким.

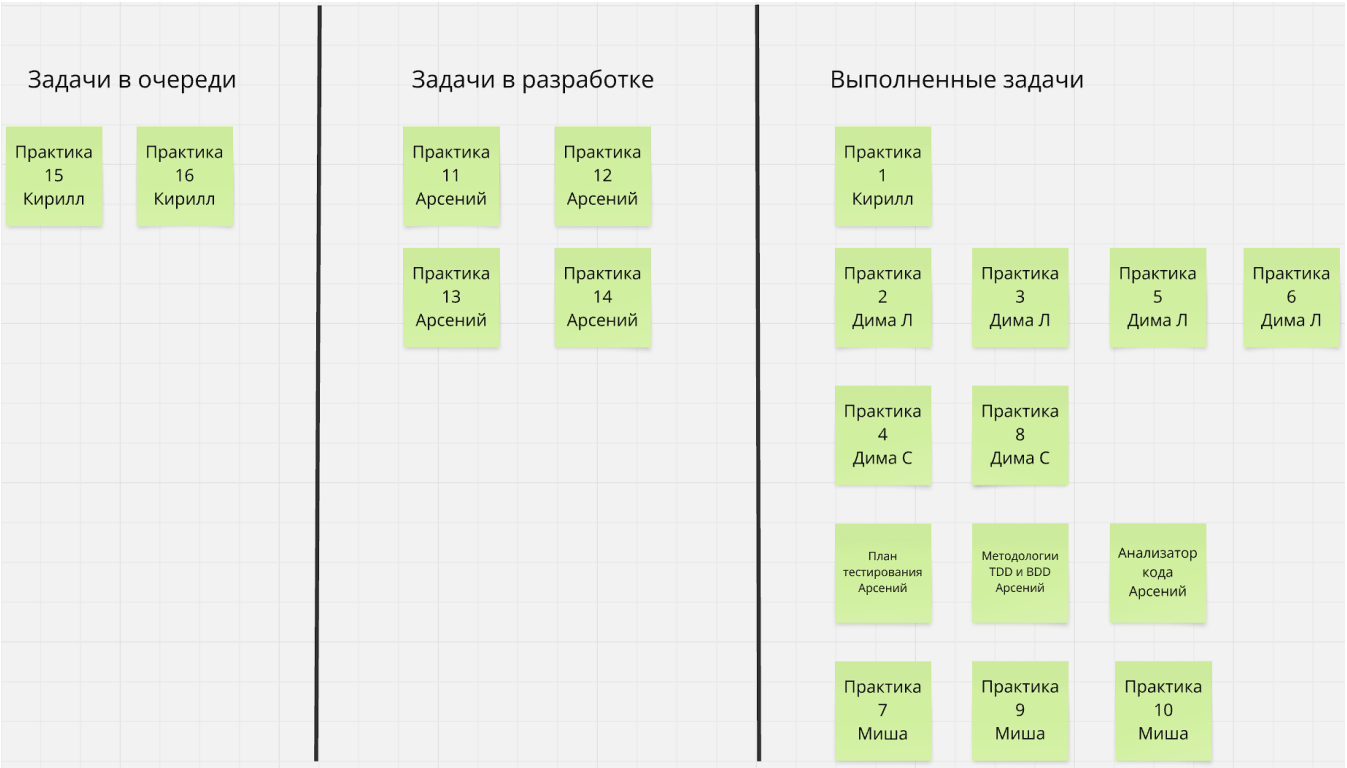


Рисунок А.2 – Список задач

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гусев К. В., Воронцов Ю. А., Михайлова Е. К. Системная и программная инженерия: методические указания по выполнению практических работ. — Москва: РТУ МИРЭА, 2021. — 120 с.
2. Баранюк В. В. Системная и программная инженерия: методические указания по выполнению практических работ. Часть 1. — Москва: РТУ МИРЭА, 2020. — 110 с.
3. Лаврищева Е. М. Программная инженерия и технологии программирования сложных систем: учебник для вузов. — Москва: Юрайт, 2021. — 350 с.
4. Лаврищева Е. М. Программная инженерия. Парадигмы, технологии и CASE-средства: учебник для вузов. — Москва: Юрайт, 2021. — 330 с.
5. Черткова Е. А. Программная инженерия. Визуальное моделирование программных систем: учебник для вузов. — Москва: Юрайт, 2021. — 280 с.
6. Баранюк В. В., Миронов А. Н., Крылова О. С. Системная и программная инженерия: методические указания по выполнению практических работ. Ч. 1. — Москва: РТУ МИРЭА, 2020. — 130 с.
7. Дешко И. П., Кряженков К. Г., Цветков В. Я. Системная и программная инженерия: учебное пособие. — Москва: МАКС Пресс, 2018. — 250 с.