МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Ижевский государственный технический университет имени М.Т. Калашникова»

(ФГБОУ ВО «ИжГТУ имени М.Т. Калашникова»)

Факультет «Информационные технологии»

Кафедра «Автоматизированные системы обработки информации и управления»

Работа защищена с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине «Программная инженерия»

на тему «Автоматизация деятельности автозаправочной станции»

|  |  |
| --- | --- |
| Выполнили  студенты гр. Б23-782-2 | Корепанов И. К.  Ложкин Р. Д.  Степанов Е. А. |
|  |  |
| Принял  к.т.н., доцент каф. АСОИУ | Касимов Д.Р. |

Рецензия:

степень достижения поставленной цели работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

полнота разработки темы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

уровень самостоятельности работы обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

недостатки работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ижевск – 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc185623062)

[1 Разработка спецификации требований 4](#_Toc185623063)

[1.1 Функциональные требования 5](#_Toc185623064)

[1.2 Нефункциональные требования 5](#_Toc185623065)

[1.3 Разработка макета 6](#_Toc185623066)

[1.4 Use-case диаграммы 7](#_Toc185623067)

[2 Планирование программного проекта по методологии Scrum 11](#_Toc185623068)

[2.1 Mindmap 11](#_Toc185623069)

[2.2 Создание задач для разработчиков 11](#_Toc185623070)

[2.3 Подбор команды разработчиков 12](#_Toc185623071)

[2.4 Приоритезация задач 13](#_Toc185623072)

[2.5 Диаграмма Ганта 15](#_Toc185623073)

[3 Проектирование программного продукта 17](#_Toc185623074)

[3.1 Диаграмма состояний 17](#_Toc185623075)

[3.2 Диаграмма последовательности 18](#_Toc185623076)

[3.3 Диаграмма классов 20](#_Toc185623077)

[4 Управление дефектами программного продукта 23](#_Toc185623078)

[4.1. Создание тест-кейсов 23](#_Toc185623079)

[4.2 Структура проекта на GitHub 25](#_Toc185623081)

[5 Оценка качества программного продукта 27](#_Toc185623082)

[ЗАКЛЮЧЕНИЕ 30](#_Toc185623083)

[ПРИЛОЖЕНИЕ 31](#_Toc185623084)

# ВВЕДЕНИЕ

В современном мире автомобильная индустрия продолжает активно развиваться, и вопросы обеспечения удобства и комфорта для автомобилистов приобретают все большее значение. Традиционные методы поиска и использования заправочных станций, такие как ориентирование по указателям и очереди на заправку, зачастую оказываются недостаточными перед лицом современных требований к скорости и удобству. В связи с этим, на первый план выходят мобильные приложения, которые позволяют автомобилистам оперативно находить ближайшие заправочные станции, получать информацию о доступных видах топлива и ценах, а также планировать маршруты с учетом необходимости заправки. Такие приложения позволяют пользователям в режиме реального времени получать актуальную информацию о состоянии заправочных станций, оперативно реагировать на изменения в доступности топлива и ценах, а также управлять процессом заправки, например, через предварительное бронирование мест на станции.

Разработка программного продукта в рамках данной курсовой работы обусловлена возрастающей потребностью в современных и доступных решениях для обеспечения удобства и комфорта автомобилистов. Разработка приложения для поиска и использования заправочных станций позволит создать удобный и функциональный инструмент, который будет способствовать повышению уровня удобства и комфорта при заправке автомобиля, обеспечит пользователям доступ к актуальной информации и предоставит чувство уверенности в надежности и точности получаемых данных.

# 1 Разработка спецификации требований

Разрабатываемый программный продукт – приложение для мониторинга и управления домашней охранной системы. Оно должно предоставить пользователям интуитивно понятный интерфейс для поиска ближайшей заправки и цен на бензин.

Перед началом разработки был составлен набор первичных требований к продукту, User Story, чтобы обеспечить соответствие потребностям пользователей.

Таблица 1.1 – User Story, первичные требования клиента

|  |
| --- |
| Как клиент заправочной станции я хочу:  1. Как пользователь, я хочу узнать маршрут до ближайшей АЗС, чтобы сэкономить своё время.  2. Как пользователь, я хочу видеть таблицу с курсом на топливо, чтобы примерно рассчитать свои затраты.  3. Как пользователь, я хочу иметь возможность зарегистрировать карту лояльности, чтобы иметь различные бонусы (скидка на топливо).  4. Как пользователь, я хочу иметь возможность оплатить заправку через приложение, чтобы сэкономить своё время.  5. Как пользователь, я хочу, чтобы приложение поддерживало различные методы оплаты (банковские карты, мобильные платежи, электронные кошельки), чтобы мне не приходилось использовать один метод оплаты. |

На их основе были разработаны следующие первичные требования:

1. Приложение должно работать на мобильном телефоне.
2. На главном экране должны быть кнопки регистрации, входа и входа для персонала.
3. На главном экране приложения должна отображаться таблица, содержащая следующие данные (в указанном порядке):
   * Кнопка, которая переносит пользователя на карту заправок
   * Проложить маршрут до ближайшой заправки
   * Промоакции
   * Избранные АЗС
   * Настройки
   * Показатели карты лояльности
   * Табло с курсом цен на различный бензин
4. На главном экране должен отображаться информер, содержащий день недели и время.
5. Приложение работает только при наличии интернет-соединения.

Процесс разработки требований начался с определения первичных потребностей пользователей. На основе этой информации были сформулированы функциональные требования, которые описывают, какие конкретные задачи должна выполнять система. Параллельно были определены нефункциональные требования, которые устанавливают критерии качества, которым система должна соответствовать.

# 1.1 Функциональные требования

1. Регистрация и аутентификация:
   * Создание учетной записи.
   * Вход в систему.
2. Поиск заправочных станций:
   * Отображение ближайших заправок на карте.
   * Фильтрация по типу топлива и услугам.
3. Информация о заправках:
   * Цены на топливо.
   * Часы работы.
   * Контактная информация.
4. Оплата топлива:
   * Поддержка различных методов оплаты.
   * История транзакций.
5. Уведомления:
   * Напоминания о заправке.
   * Уведомления о специальных предложениях.

# 1.2 Нефункциональные требования

1. Производительность:
   * Быстрая загрузка (меньше 5 секунд на вход в приложение) и отклик приложения.
2. Безопасность:
   * Защита данных пользователей (данные не передаются третьим лицам).
   * Безопасные транзакции ( дополнительное подтверждение вторым фактором).
3. Надежность:
   * Стабильная работа без сбоев.
   * Высокая доступность ( работа приложения 24 часа в сутки).
4. Масштабируемость:
   * Поддержка увеличения числа пользователей (работа приложения без ошибок до 20000 онлайн пользователей).

# 1.3 Разработка макета

Также, для представления общего внешнего вида программного продукта, был разработан дизайн-макет интерфейса приложения, Этот макет дает наглядное представление о том, как будет выглядеть и функционировать приложение. Он демонстрирует основные элементы интерфейса. Макет изображен на рисунке 1.1.

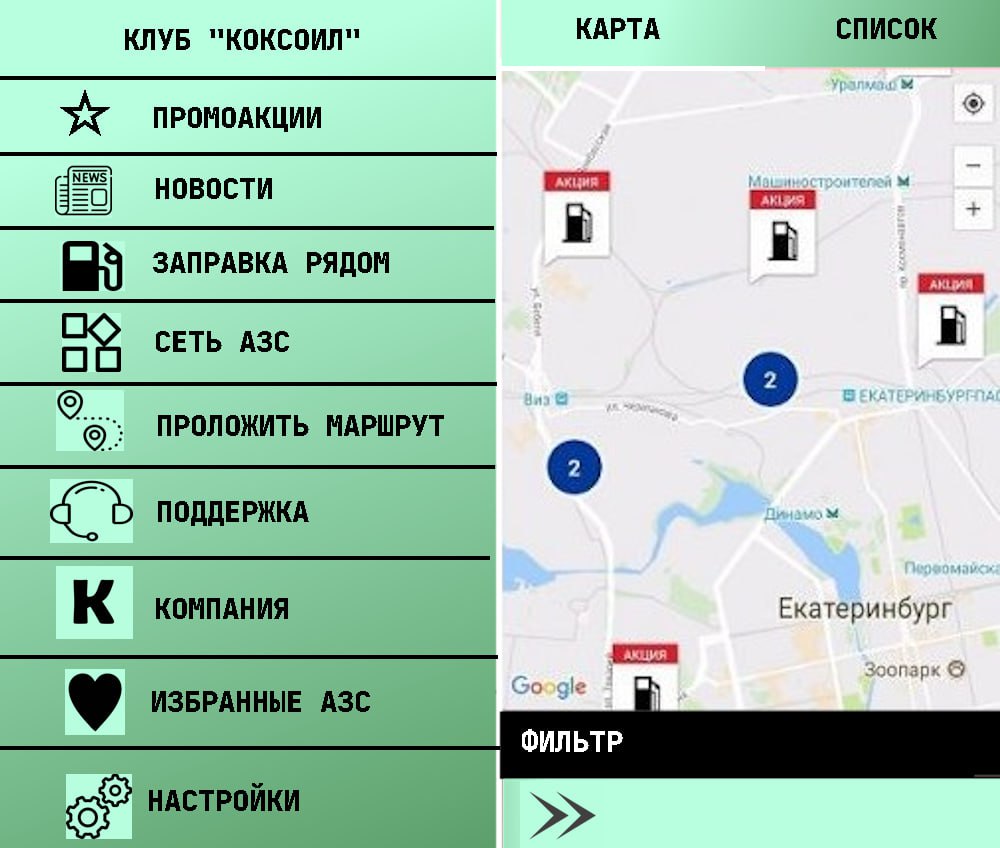


Рисунок 1.1 – Макет страницы приложения

# 1.4 Use-case

Изучив пожелания пользователей , на их основе были разработаны use-case диаграммы для описания необходимых возможностей программного продукта. На таблицах 1.2, 1.3 и 1.4 предоставлены диаграммы для некоторого функционала программы.

Таблица 1.2 - Use Case заправки топливом

|  |  |  |
| --- | --- | --- |
| Название | Заправка топливом | |
| Описание | Пользователь выбирает подходящую ему АЗС и выбирает метод оплаты | |
| Действующие лица | Пользователь | |
| Предусловия | К Пользователь заходит в приложение | |
| Основной поток событий | 1. Пользователь открывает мобильное приложение и входит в свой аккаунт.  2. Пользователь выбирает ближайшую заправочную станцию из списка доступных.  3. Пользователь выбирает тип топлива (бензин, дизель и т.д.) и объем (литры или полный бак).  4. Пользователь выбирает способ оплаты (банковская карта, электронный кошелек и т.д.).  5. Пользователь подтверждает заказ на заправку.  6. Пользователь подходит к выбранной колонке и начинает заправку автомобиля.  7. После завершения заправки, пользователь получает уведомление в приложении о завершении операции.  8. Пользователь получает электронный чек на свою почту или в приложении. | |
| Альтернативный поток событий | | Если на счете пользователя недостаточно средств для оплаты, система уведомляет пользователя и предлагает пополнить баланс.  Если произошла ошибка при оплате, система уведомляет пользователя и предлагает повторить операцию или выбрать другой способ оплаты. |
| Постусловия | | Пользователь покидает заправочную станцию.  Система обновляет баланс пользователя и историю заправок. |

Продолжение таблицы 1.2

|  |  |
| --- | --- |
| Расширенные атрибуты | Пользователь может отменить заправку до начала процесса. Система уведомляет пользователя о успешной отмене. |
| Диаграмма Use Case |  |

Таблица 1.3 – Use Case оплаты заправки через мобильное приложение

|  |  |
| --- | --- |
| Название | Планирование и оплата заправки через мобильное приложение |
| Описание | Позволить пользователю запланировать и оплатить заправку автомобиля заранее через мобильное приложение, обеспечивая удобство и экономию времени. |
| Действующие лица | Пользователь |
| Предусловия | Пользователь установил и зарегистрировался в мобильном приложении.  У пользователя есть доступ к интернету.  Пользователь планирует заправку заранее. |

Продолжение таблицы 1.3

|  |  |
| --- | --- |
| Основной поток событий | 1. Пользователь открывает мобильное приложение и входит в свой аккаунт.  2. Пользователь выбирает опцию "Планирование заправки" в меню приложения.  3. Пользователь выбирает дату и время, когда планирует заправить автомобиль.  4. Пользователь выбирает заправочную станцию из списка доступных.  5. Пользователь выбирает тип топлива (бензин, дизель и т.д.) и объем (литры или полный бак).  6. Пользователь выбирает способ оплаты (банковская карта, электронный кошелек и т.д.).  7. Пользователь подтверждает планирование заправки.  8. После завершения заправки, пользователь получает уведомление в приложении о завершении операции. |
| Альтернативный поток событий | Пользователь может изменить дату, время или заправочную станцию до назначенного времени. Система уведомляет пользователя о успешном изменении. |
| Постусловия | Пользователь покидает заправочную станцию.  Система обновляет баланс пользователя и историю заправок. |

Продолжение таблицы 1.3

|  |  |
| --- | --- |
| Диаграмма Use Case |  |
| Расширенный атрибуты | Пользователь может отменить заправку до начала процесса. Система уведомляет пользователя о успешной отмене. |

# 2 Планирование программного проекта по методологии Scrum

Перед началом разработки было решено использовать гибкую методологию Scrum для организации проекта и процессов разработки. Такой подход позволит оперативно реагировать на изменения требований и обеспечит прозрачность на всех этапах.

# 2.1 Mindmap

Для визуализации задач и определения необходимых шагов для запуска проекта была составлена Mindmap (ментальная карта), отвечающая на вопрос "Что нужно сделать, чтобы запустить проект?". Карта, которую можно увидеть на рисунке 2.1, стала отправной точкой для планирования спринтов и распределения задач внутри команды.

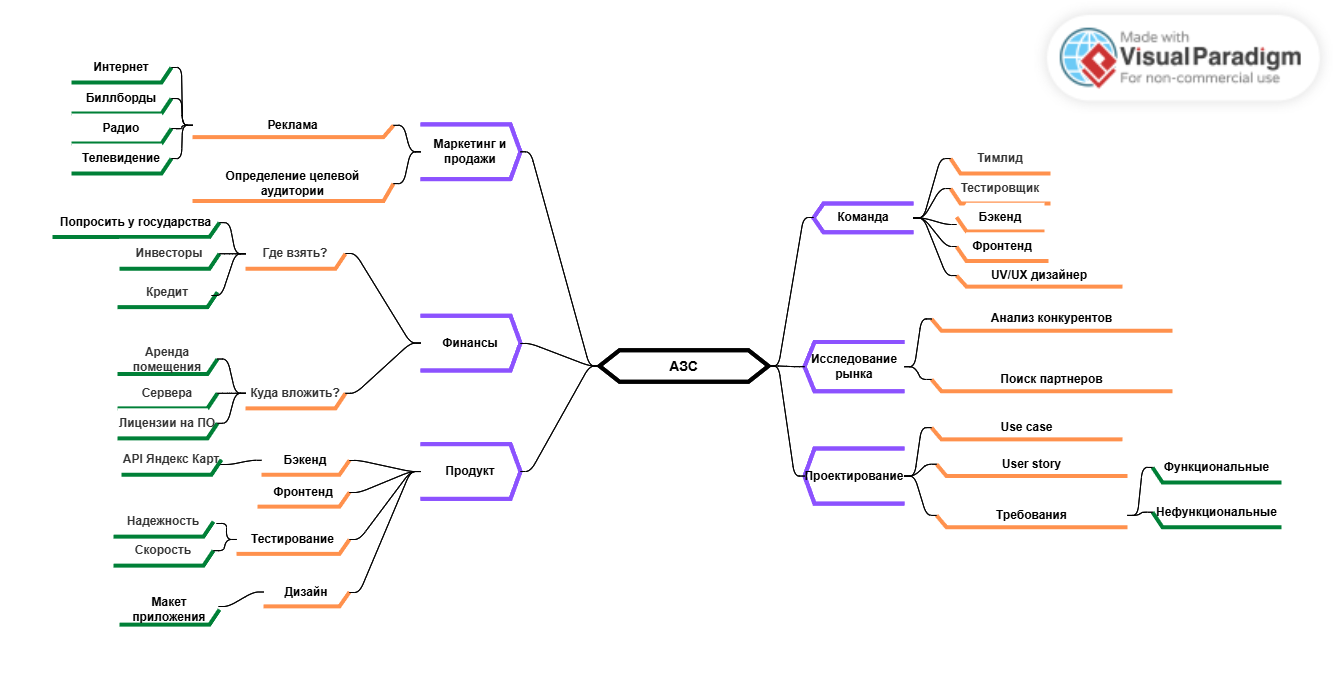


Рисунок 2.1 – Mindmap проекта системы АЗС

# 2.2 Создание задач для разработчиков

На основе анализа требований были определены и сформулированы следующие задачи для разработчиков. Эти задачи далее будут распределены по приоритету и сложности, что позволит организовать эффективный процесс разработки и контролировать выполнение каждого этапа.

Таблица 2.1 – Список задач для разработчиков

|  |
| --- |
| Разработать макет окна с подробной информацией о заправке (включая списки работников и ошибок). |
| Нарисовать кнопку «Отключить заправку» на экране заправки. |
| Добавить интерфейс для настройки лимитов потребления топлива. |
| Разработать окно, которое будет показывать ошибку. |
| Реализовать экран авторизации по макету. |
| Реализовать интерфейс для отображения информации о заправках на главном экране. |
| Реализовать кнопку для добавления новой заправки. |
| Реализовать страницу для отображения подробной информации о заправке. |
| Настроить возможность клика по заправке для открытия этой страницы. |
| Реализовать механизм динамического отображения данных о заправке. |
| Реализовать функционал для отключения заправки через интерфейс приложения. |
| Реализовать функционал для установки и изменения лимитов. |
| Разработать API для аутентификации (проверка логина и пароля). |
| Настроить связь с базой данных для хранения информации о пользователях. |
| Создать структуру базы данных для хранения информации о заправках. |
| Настроить микросервис для передачи данных о заправках на фронтенд. |
| Настроить архитектуру сервера (Трёхуровневая архитектура “Клиент-сервер”). |

# 2.3 Подбор команды разработчиков

Прежде чем приступать к разработке, требовалось сформировать эффективную команду, определив роли, уровни квалификации и круг обязанностей каждого участника. Этот шаг напрямую связан с бюджетом проекта, поэтому был составлен список необходимых специалистов с подробным расчетом их заработной платы, что позволило оценить необходимые финансовые ресурсы и оптимизировать расходы.

Таблица 2.2 – Подобранная команда разработчиков

|  |  |  |
| --- | --- | --- |
|  | **Зарплата (в месяц / в час), руб** | **Задачи** |
| Проджект-менеджер | 140 000 / 833.3 | Администрирует общий ход работы, договоривается с партнерами. |
| Дизайнер | 90 000 / 535.7 | Проектирует интерфейсы и разрабатывает общую логистику программы. |
| Фронтэнд разработчик | 85 000 / 506 | Создаёт интерфейс приложений, с которыми взаимодействуют пользователи. |
| Бэкенд разработчик | 120 000 / 714.3 | Программист, который работает над внутренней частью приложения. Он пишет код, разрабатывает бизнес-логику приложений, задает им алгоритм работы и обеспечивает корректное выполнение пользовательских запросов. |
| тестировщик | 71 064 / 423 | Специалист, принимающий участие в тестировании компонента или системы. |
| DevOps инженер | 150 000 / 892.9 | Cинхронизирует все этапы создания программного продукта: от написания кода до тестирования и релиза. |
| Бизнес аналитик | 110 000 / 653.8 | Специалист, который является некоторым промежуточным звеном между заказчиками и командой разработчиков. |
| Инженер по безопасности | 140 000 / 833.3 | Это специалист, который занимается защитой информации. |

# 2.4 Приоритезация задач

Далее был разработан список задач, расставленных в порядке приоритетности. Приоритеты обозначены числовой шкалой:

* 0 – соответствует наивысшему приоритету (задачи, которые необходимо выполнить в первую очередь)
* 1 – соответствует среднему уровню приоритета (задачи все еще важны, но уже не так сильно)
* 2 – наименьшему (задачи, выполнение которых можно отложить).

Приоритет 0 (Самый высокий)

Дизайнер

* Разработать макет окна с подробной информацией о заправке (включая списки работников и ошибок) — 16 часов
* Добавить интерфейс для настройки лимитов потребления топлива — 8 часов

Фронтенд-разработчик

* Реализовать экран авторизации по макету — 12 часов
* Реализовать интерфейс для отображения информации о заправках на главном экране — 16 часов
* Реализовать страницу для отображения подробной информации о заправке — 20 часов
* Реализовать механизм динамического отображения данных о заправке — 14 часов

Бэкенд-разработчик

* Разработать API для аутентификации (проверка логина и пароля) — 8 часов
* Настроить связь с базой данных для хранения информации о пользователях — 6 часов
* Создать структуру базы данных для хранения информации о заправках — 8 часов
* Настроить микросервис для передачи данных о заправках на фронтенд — 12 часов

DevOps-инженер

* Настроить архитектуру сервера (Трёхуровневая архитектура “Клиент-сервер”) — 16 часов
* Обеспечить настройку серверной инфраструктуры (виртуализация, СУБД, DNS/DHCP) — 20 часов

Общие требования

* Найти сервер (характеристики) — 4 часа

Приоритет 1 (Средний)

Дизайнер

* Нарисовать кнопку «Отключить заправку» на экране заправки — 4 часа
* Разработать окно, которое будет показывать ошибку — 6 часов

Фронтенд-разработчик

* Реализовать кнопку для добавления новой заправки — 8 часов
* Настроить возможность клика по заправке для открытия страницы — 6 часов
* Реализовать функционал для отключения заправки через интерфейс приложения — 10 часов
* Реализовать функционал для установки и изменения лимитов — 12 часов

Бэкенд-разработчик

* Реализовать API для получения списка работников и ошибок для конкретной заправки — 10 часов
* Настроить API для отключения заправки и изменения её статуса в базе данных — 8 часов
* Настроить API для сохранения ограничений на потребление топлива в базе данных — 10 часов
* Реализовать шифрование данных с использованием шифрования и метода TDE — 16 часов

DevOps-инженер

* Настроить резервирование серверных ресурсов — 12 часов
* Настроить систему обнаружения вторжений (DDoS-атак) — 10 часов

Приоритет 2 (Самый низкий)

Фронтенд-разработчик

* Реализовать механизм стирания данных при неверной аутентификации — 6 часов
* Реализовать функционал изменения цвета границ окна в зависимости от состояния заправки — 8 часов
* Реализовать скроллинг по списку заправок — 6 часов

Бэкенд-разработчик

* Реализовать API для уведомления сотрудника — 6 часов
* Настроить обработку ошибок при неверном вводе логина или пароля на сервере — 8 часов

# 2.5 Диаграмма Ганта

С целью оптимизации процесса разработки, задачи были декомпозированы на итерации продолжительностью одну неделю. На основании проведенной оценки трудозатрат была сформирована диаграмма Ганта, представленная на рисунках 2.2-2.4.



Рисунок 2.2 – Диаграмма Ганта по первой неделе

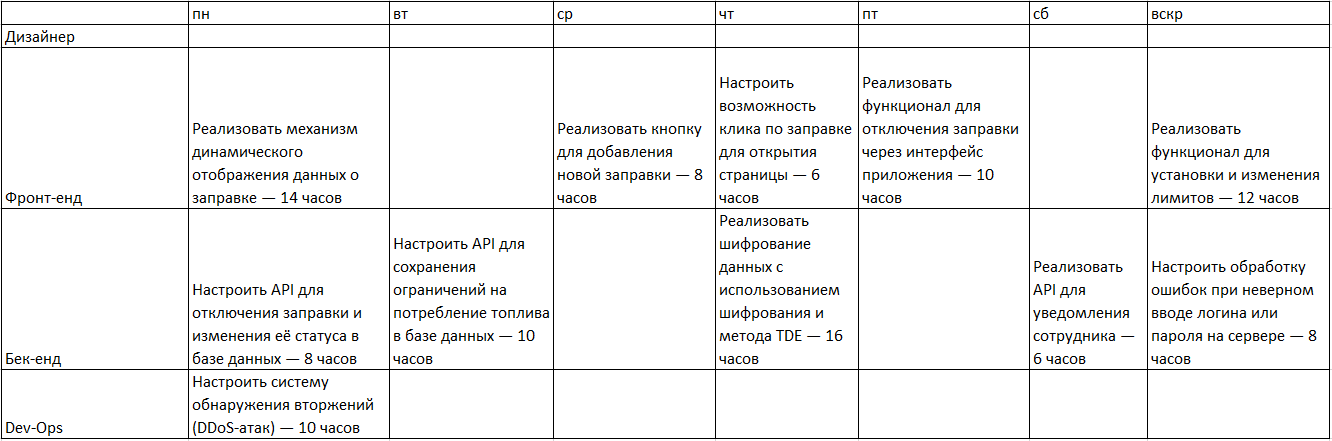


Рисунок 2.3 – Диаграммы Ганта по второй неделе

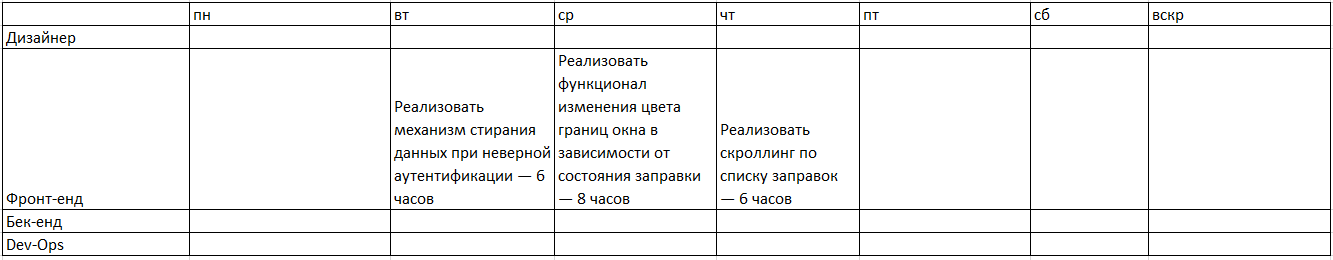


Рисунок 2.4 – Диаграмма Ганта по третьей неделе

В общей сложности вышло, что суммарная стоимость всего проекта с учетом зарплат сотрудникам и иных необходимых вложений – 387,814 рубля.

# 3 Проектирование программного продукта

В ходе разработки программного продукта было проведено его проектирование. Были созданы UML-диаграммы состояний (state machine diagram), последовательности (sequence diagram) и классов (class diagram).

3.1 Диаграмма состояний

На основе основного объекта деятельности разрабатываемого ПО, была составлена UML-диаграмма состояний, приведенная на рисунке 3.1. Данная диаграмма описывает состояния системы при оплате заказа.

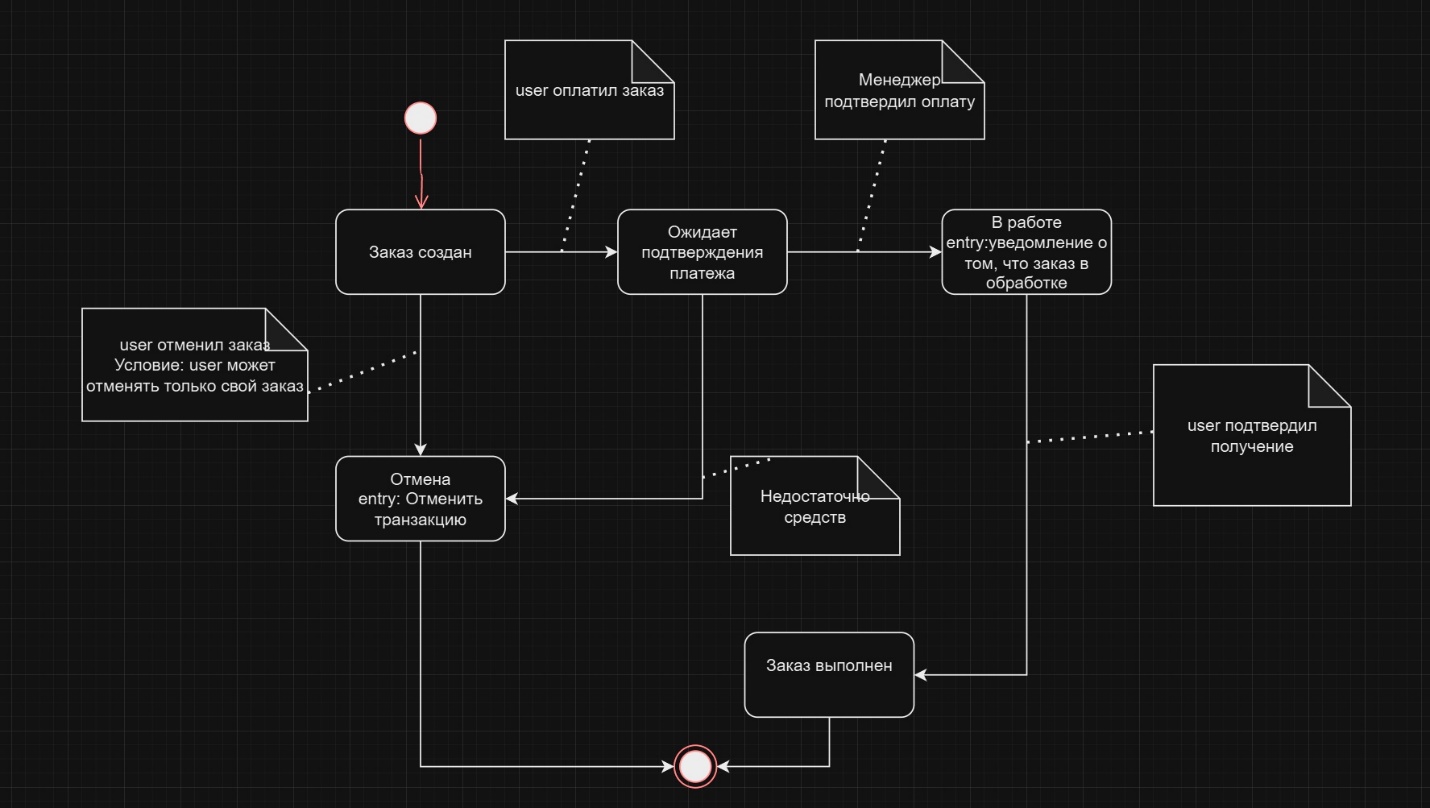


Рисунок 3.1 – UML-диаграмма состояний системы

Разработанная диаграмма отражает процесс взаимодействия клиента с приложением для отправки заявки на услуги АЗС. Она позволяет наглядно представить, как приложение реагирует на действия пользователя и какие события инициируют изменения в состоянии системы.

Создание диаграммы состояний позволило:

1. Структурировать процесс отправки заявки и предусмотреть возможные исключительные ситуации.
2. Определить ключевые точки взаимодействия пользователя с приложением.
3. Обеспечить разработчикам и тестировщикам четкое понимание всех состояний системы, что упрощает последующую реализацию и тестирование.

Диаграмма состояний является важным этапом проектирования, так как она демонстрирует логику работы приложения и помогает предвидеть потенциальные ошибки. Она станет основой для дальнейшей разработки и тестирования процесса отправки заявок.

# 3.2 Диаграмма последовательности

Затем была составлена диаграмма последовательности, описывающая процессы, происходящие в системе при ее срабатывании – реакция на триггер. Диаграмма приведена на рисунке 3.2.

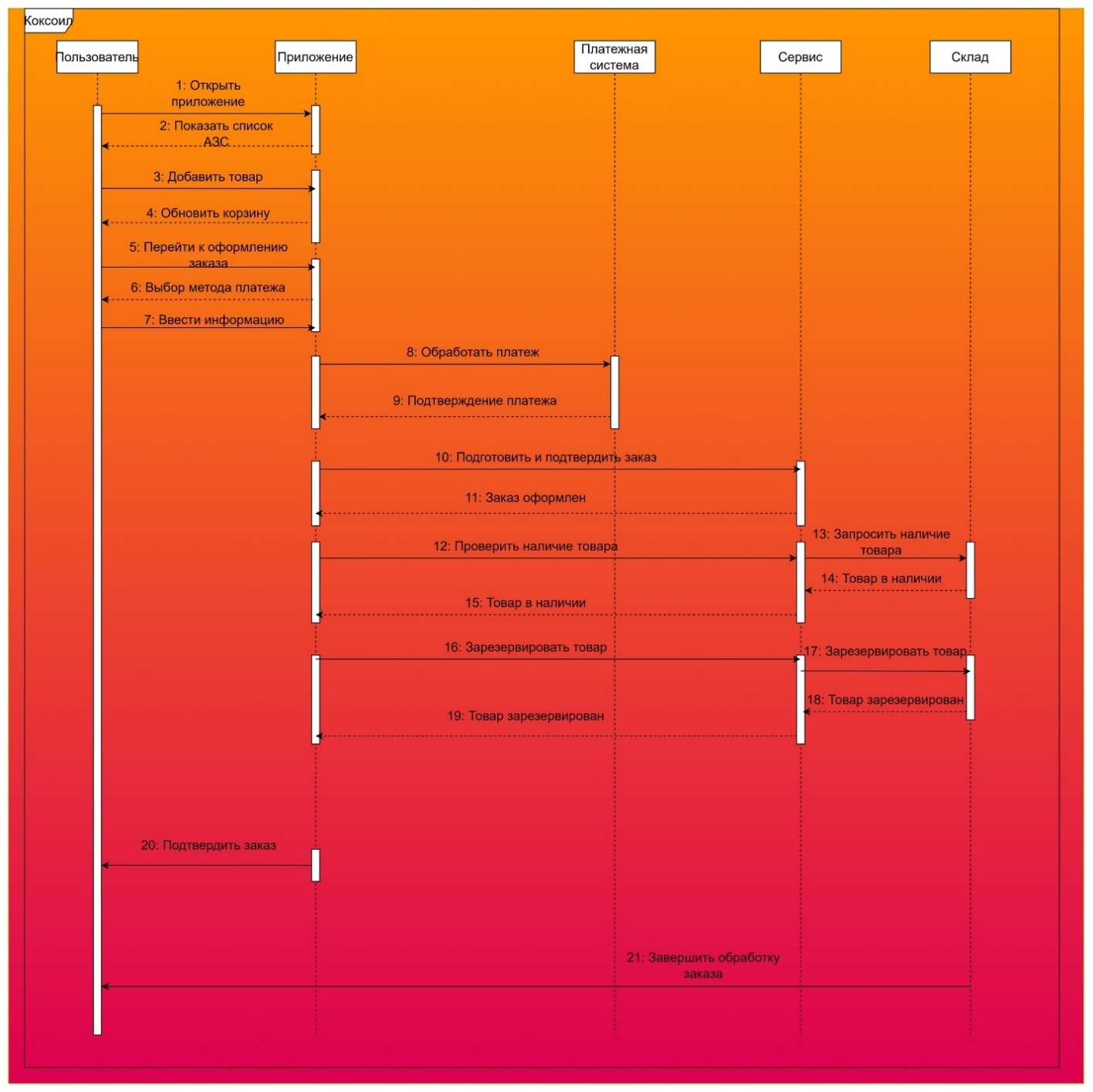


Рисунок 3.2 – Диаграмма последовательности выполняемых процессов

Диаграмма описывает процесс выполнения заказа в системе, включающей нескольких участников:

1. **Пользователь** — инициатор процесса.
2. **Приложение** — интерфейс, через который пользователь взаимодействует с системой.
3. **Платежная система** — модуль, обрабатывающий платежи.
4. **Сервис** — промежуточный модуль для управления заказами.
5. **Склад** — отвечает за проверку наличия и резервирование товаров.

Описание этапов:

1. Пользователь открывает приложение (шаг 1).
2. Приложение показывает список доступных товаров (шаг 2).
3. Пользователь добавляет товар в корзину (шаг 3).
4. Пользователь открывает корзину для оформления заказа (шаг 4).
5. Пользователь выбирает метод оплаты (шаг 6) и вводит необходимую информацию (шаг 7).
6. Платежная система обрабатывает платеж (шаг 8) и подтверждает его (шаг 9).
7. Приложение передает запрос на подготовку заказа (шаг 10).
8. Сервис оформляет заказ и проверяет наличие товара (шаги 11–12).
9. Склад подтверждает наличие товара (шаг 13–14).
10. Сервис резервирует товар (шаг 16), а склад подтверждает резервирование (шаг 17–18).
11. Приложение подтверждает пользователю, что заказ оформлен (шаг 19).
12. Пользователь подтверждает окончание обработки заказа (шаг 20), и процесс завершается (шаг 21).

Диаграмма демонстрирует взаимодействие между участниками, передачу сообщений и последовательность операций для успешной обработки заказа.

# 3.3 Диаграмма классов

Позже была составлена диаграмма классов системы, представленная на рисунке 3.3.

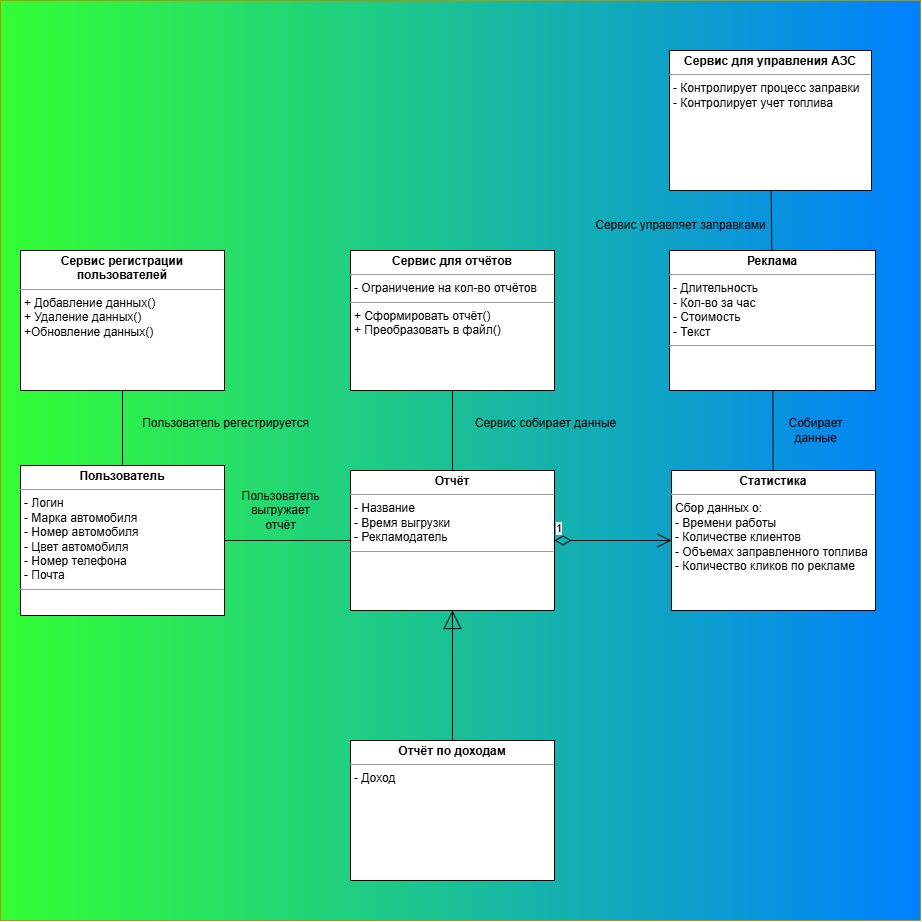


Рисунок 3.3 - Диаграмма классов

Основные классы и их описание:

1. Пользователь

Атрибуты:

* Логин
* Марка автомобиля
* Номер автомобиля
* Цвет автомобиля
* Номер телефона
* Почта

Описание:  
Представляет пользователя системы, который регистрируется и взаимодействует с сервисами, а также формирует запросы для выгрузки отчетов.

2. Сервис регистрации пользователей

Методы:

* Добавление данных
* Удаление данных
* Обновление данных

Описание:  
Отвечает за управление учетными записями пользователей: создание, удаление и обновление их данных.

3. Сервис для управления АЗС

Методы:

* Контролирует процесс заправки
* Контролирует учет топлива

Описание:  
Управляет процессами, связанными с заправкой автомобилей, и ведет учет расхода топлива.

4. Сервис для отчетов

Методы:

* Сформировать отчет
* Преобразовать в файл

Описание:  
Создает отчеты на основе собранных данных, обеспечивает возможность их экспорта в файл.

5. Отчет

Атрибуты:

* Название
* Время выгрузки
* Рекламодатель

Описание:  
Хранит информацию о сформированном отчете, который может быть выгружен пользователем.

6. Отчет по доходам

Атрибуты:

* Доход

Описание:  
Специализированный тип отчета, содержащий данные о доходах.

7. Реклама

Атрибуты:

* Длительность
* Количество за час
* Стоимость
* Текст

Описание:  
Управляет рекламным контентом, используемым в системе.

8. Статистика

Атрибуты:

* Времена работы
* Количество клиентов
* Объем заправленного топлива
* Количество кликов по рекламе

Описание:  
Собирает данные о деятельности системы, такие как количество клиентов и объем топлива.

Диаграмма демонстрирует связь между классами, их атрибутами, методами и взаимодействием для решения задач автоматизации управления АЗС и отчетности.

# 4 Управление дефектами программного продукта и совместная работа над проектом

В ходе тестирования программного продукта был выявлен критический баг, который, при определенных условиях, может привести к серьезным последствиям, включая невозможность корректного оформления заказа и, как следствие, негативный пользовательский опыт и ущерб репутации разработчиков. Назначен: Разработчик(фронт), время работы - 8ч.

Также в процессе тестирования программного продукта был обнаружен серьезный дефект, который может привести к значительным проблемам, включая невозможность просмотра заправок и, как результат, негативные отзывы пользователей и ущерб репутации разработчиков.Назначен: Разработчик (фронт), время работы - 5ч

Далее был обнаружен незначительный дефект, который может повлиять на пользовательский опыт, но не критичен для основного функционала. Назначен: Разработчик (фронт), время работы - 3ч

# 4.1. Создание тест-кейсов

С целью предотвращения возникновения непредвиденных проблем в дальнейшем, было принято решение о применении практики тестирования на основе тест-кейсов. Сотрудники составляют подробные сценарии проверок для каждой функциональности приложения, что обеспечивает своевременное обнаружение дефектов и позволяет разработчикам оперативно их устранять. Баги подробно описаны в ПРИЛОЖЕНИЕ А.

### Тест-кейсы для приложения по заправке

#### 1. Позитивный тест-кейс:

**Уникальный номер:** 001

**Заголовок:** Добавление топлива в корзину.

**Предусловия:** Пользователь находится во вкладке «Заправки».

**Окружение:** Android IOS.

**Постусловия:** -

**Шаги:**

1. Зайти во вкладку «Заправки».
2. Добавить любое топливо в корзину.
3. Перейти во вкладку «Корзина».
4. Сверить выбранные позиции с отображёнными.

**Ожидаемый результат тест-кейса:** Выбранное топливо добавлено в корзину.

**Статус:** Успешно.

**Фактический результат тест-кейса:** Выбранное топливо добавлено в корзину.

#### 2. Негативный тест-кейс:

**Уникальный номер:** 007

**Заголовок:** Ввод некорректных символов в поле «телефон» (русские буквы).

**Предусловия:** Пользователь находится в окне «Ваш заказ».

**Окружение:** Android IOS.

**Постусловия:** -

**Шаги:**

1. Зайти во вкладку «Заправки».
2. Добавить любое топливо в корзину.
3. Открыть вкладку «Корзина».
4. Нажать кнопку «к оформлению».
5. Ввести в поле «телефон» буквенный набор.
6. Выбрать способ оплаты.
7. Нажать кнопку оформить заказ.

**Ожидаемый результат тест-кейса:** Вывод сообщения об ошибке «Некорректный ввод телефона».

**Статус:** Провал.

**Фактический результат тест-кейса:** Заказ сформирован, в базу данных вместо телефона занесён введённый набор символов.

#### 3. Деструктивный тест-кейс:

**Уникальный номер:** 013

**Заголовок:** Множественное добавление одного вида топлива в корзину.

**Предусловия:** Пользователь находится во вкладке «Заправки».

**Окружение:** Android IOS.

**Постусловия:** -

**Шаги:**

1. Зайти во вкладку «Заправки».
2. Добавить любое топливо в корзину.
3. Открыть вкладку «Корзина».
4. Нажать кнопку «к оформлению».
5. У выбранного топлива нажать кнопку «+» 31 раз.

**Ожидаемый результат тест-кейса:** При достижении лимита по количеству (30 шт) кнопка «+» становится неактивной.

**Статус:** Провал.

**Фактический результат тест-кейса:** При достижении лимита по количеству (30 шт) кнопка «+» работает в штатном режиме.

# 4.2 Структура проекта на GitHub

Репозиторий называется "zh0pa4" и принадлежит пользователю Shxwi. В нем содержатся следующие файлы:

1. Laba1 - Laba8: лабораторные работы.
2. Photo: изображение, связанное с проектом.
3. README.md: файл с описанием репозитория.

На рисунке 4.1 предоставлен график изменений репозитория и его коммитов за период с 22 сентября по 15 декабря.

На рисунке 4.2 предоставлено сдержимое репозитория.

На рисунке 4.3 предоставлен список изменнений репозитория.

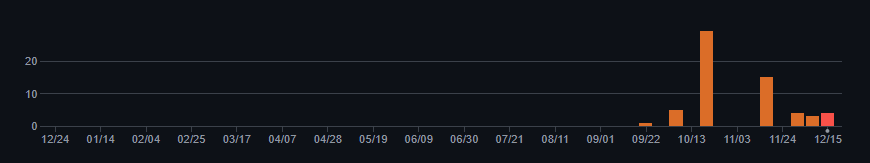


Рисунок 4.1 – График коммитов.

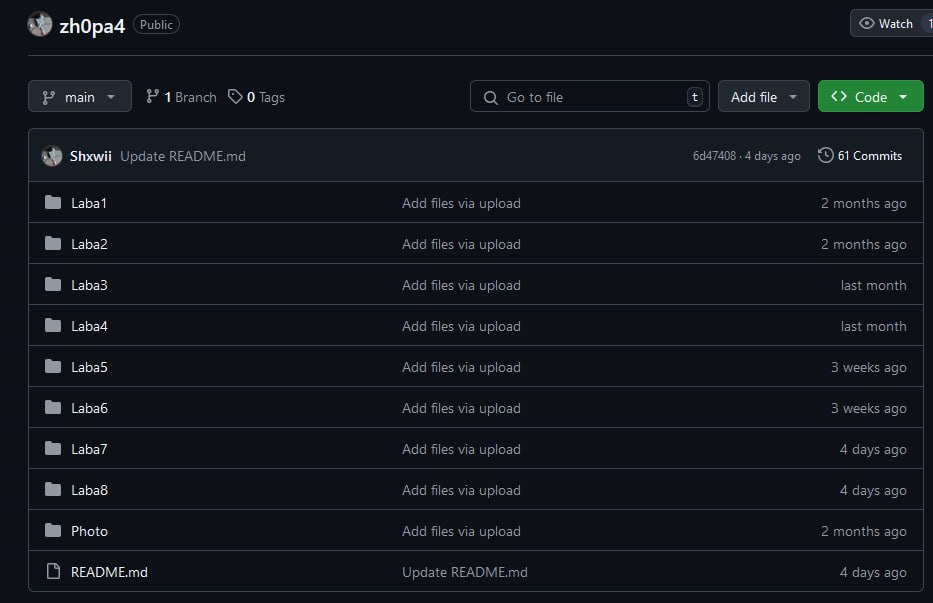


Рисунок 4.2 – Содержимое репозитория.

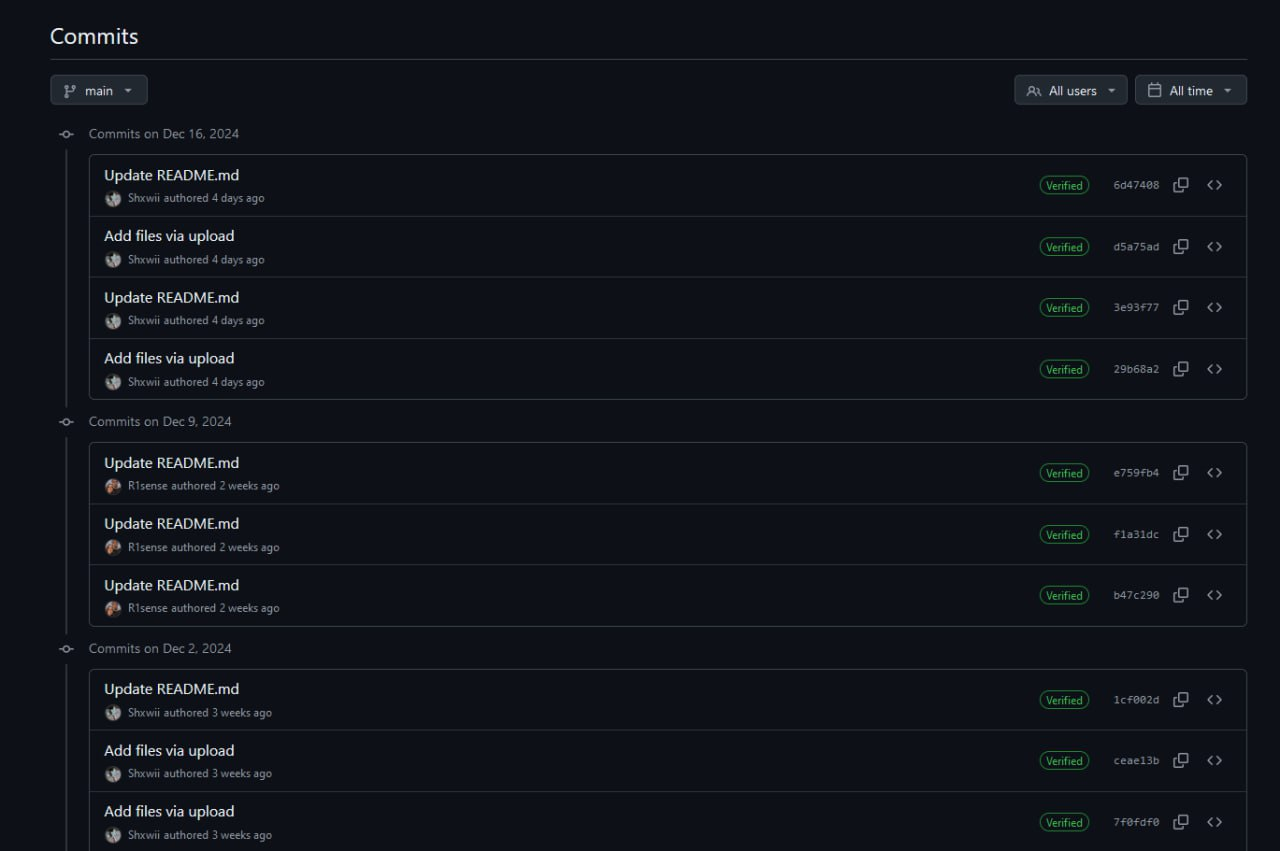


Рисунок 4.3 - Список изменений.

# 5 Оценка качества программного продукта

По завершении этапа разработки программного обеспечения была проведена комплексная оценка его качества командой разработчиков. Целью данной оценки являлось определение соответствия разработанного продукта заданным критериям качества и выявление потенциальных областей для улучшения.

В качестве основы для оценки были выбраны следующие критерии, охватывающие различные аспекты качества программного обеспечения:

* Функциональность. Данный критерий оценивает полноту реализованных функций, их точность, возможность взаимодействия с другими системами уровень защищенности и согласованность работы всех функций.
* Надежность. Оценивается способность программного обеспечения сохранять работоспособность в течение определенного времени, устойчивость к отказам (отказоустойчивость), возможность восстановления после сбоев и согласованность работы в различных условиях.
* Удобство применения. Данный критерий характеризует легкость освоения и использования программного обеспечения целевой аудиторией, удобство подготовки данных, понятность результатов работы и простоту внесения изменений в программную документацию.
* Сопровождаемость. Оценивается простота диагностики и устранения ошибок, внесения изменений и модификаций, а также адаптация программного обеспечения к изменяющимся требованиям.
* Рациональность. Данный критерий характеризует эффективность использования ресурсов вычислительной системы при работе программного обеспечения, соответствие затраченных ресурсов достигнутому уровню качества.
* Переносимость. Оценивается возможность переноса программного обеспечения на другие платформы или операционные системы, а также необходимые для этого усилия.

Таблица 5.1 – Оценка прогаммного обеспечения по критерия

|  |  |  |  |
| --- | --- | --- | --- |
| Показатели-характеристики | Атрибуты | Пояснение | Оценка из 10 |
| Функциональность | Полнота функций | То, насколько функции удовлетворяют цели приложения | 9 |
| Точность | То, насколько функции точно отвечают на поставленные требования | 8 |
| Интероперабельность | Как хорошо приложение взаимодействует с остальными | 0 |
| Защищенность | Насколько трудно украсть данные | 10 |
| Согласованность | Как критерий согласован с остальными |  |
| Надежность | Завершенность | Не "сырость" продукта | 4 |
| Отказоустойчивость | Как продукт относится к перегрузам, отменам и тд. | 5 |
| Восстонавливаемость | Затрата времени на восстановления работы продукта и данных | 6 |
| Согласованность |  |  |
| Удобство | Понимаемость | Как быстро в приложении может инуитивно разобраться пользователь | 9 |
| Обучаемость | Как быстро может пользователь обучиться эффективно пользоваться приложением | 10 |
| Привлекательность | Дизайн приложения | 8 |
| Согласованность |  |  |
| Сопровождаемость | Анализируемость | Как быстро можно выявить недостаток в приложении | 8 |
| Изменяемость | Как быстро можно переписать функционал приложения | 9 |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
|  | Стабильность | Насколько стабильно приложение | 6 |
| Тестируемость | Как легко приложение можно протестировать | 10 |
| Согласованность |  |  |
| Эффективность | Реактивность | Как быстро обрабатываются данные о пользователе | 8 |
| Используемость ресусрсов | Как много требует приложение ресурсов (10 - минимум) | 8 |
| Совместимость |  |  |
| Переносимость | Адаптируемость | Насколько универсален код приложения для различных устройств и новых версий | 6 |
| Простота настройки | Как легко настроить приложение под новые условия | 10 |
| Совметсимость | Насколько приложение совместимо с различными ОС и устройствами | Да |
| Заменяемость | Как много путей заменить тот или иной компонент приложения по причине его нехватки | 8 |
| Согласованность |  |  |

Средняя оценка всех параметров: 7.95.

# ЗАКЛЮЧЕНИЕ

Настоящая курсовая работа посвящена полному циклу разработки программного продукта, от этапа формирования требований до итоговой оценки. В ходе выполнения работы были успешно реализованы следующие этапы:

* Формирование требований. На основе проведенных ранее исследований были сформулированы первичные, функциональные и нефункциональные требования к разрабатываемому программному обеспечению. Данный этап позволил определить четкие цели и задачи проекта, а также установить критерии, по которым впоследствии оценивалось качество разработанного продукта.
* Планирование проекта. Был проведен анализ необходимого бюджета для реализации проекта, а также сформирован состав команды разработчиков. Данный этап способствовал эффективному распределению ресурсов и определению ролей участников команды.
* Разработка программного продукта. Разработка осуществлялась с применением методологии SCRUM, что обеспечило гибкость и оперативность в процессе разработки, а также возможность адаптации к изменяющимся требованиям. В результате был создан работоспособный программный продукт, отвечающий сформулированным требованиям.
* Тестирование и отладка. В рамках работы были изучены различные методы тестирования программного обеспечения, а также способы выявления и устранения программных дефектов. Проведенный анализ позволит в дальнейшем более эффективно проводить тестирование и отладку разрабатываемого программного обеспечения.
* Оценка качества программного продукта. По завершении разработки была проведена комплексная оценка качества созданного программного продукта. Полученная оценка была признана справедливой для текущего этапа разработки и послужит основой для дальнейшего совершенствования продукта.

В рамках данной курсовой работы был успешно реализован полный цикл разработки программного обеспечения, начиная от определения требований и заканчивая оценкой готового продукта. Полученные результаты демонстрируют практическое применение изученных теоретических знаний, и позволяют сделать вывод об успешном достижении поставленных целей. Дальнейшая работа над проектом будет направлена на устранение выявленных недостатков и улучшение характеристик программного обеспечения, с целью повышения его качества и функциональности.

# ПРИЛОЖЕНИЕ А

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Краткое описание** | **Подробное описание** | **Шаги по воспроизведению** | **Воспроизводимость** | **Важность** | **Срочность** | **Симптомы** | **Возможность обойти** | **Комментарий** |
| 1 | Невозможность корректного оформления заказа | При определенных условиях программа не позволяет пользователю корректно оформить заказ, что приводит к негативному пользовательскому опыту и ущербу репутации разработчиков. | 1. Открыть приложение.  2. Перейти в раздел "Оформление заказа".  3. Попытаться оформить заказ, следуя всем шагам.  4. Наблюдать за результатом. | Всегда | Высокая | Критическая | Заказ не оформляется, возникают | Нет | Могут быть срьезные последствия |

**Список багов**

Таблица 1 – список багов

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Краткое описание** | **Подробное описание** | **Шаги по воспроизведению** | **Воспроизводимость** | **Важность** | **Срочность** | **Симптомы** | **Возможность обойти** | **Комментарий** |
| 2 | Невозможность просмотра заправок | При определенных условиях программа не позволяет пользователю корректно оформить заказ, что приводит к негативному пользовательскому опыту и ущербу репутации разработчиков. | 1. Открыть приложение.  2. Перейти в раздел "Заправки".  3. Попытаться просмотреть заправки.  4. Наблюдать за результатом.. | Всегда | Высокая | Критическая | Заправки не отображаются или отображаются некорректно | Нет | Может привести к серьёзным последствиям |

Продолжение таблицы 1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Краткое описание** | **Подробное описание** | **Шаги по воспроизведению** | **Воспроизводимость** | **Важность** | **Срочность** | **Симптомы** | **Возможность обойти** | **Комментарий** |
| 3 | Незначительный дефект в пользовательском опыте | При определенных условиях программа не позволяет пользователю корректно оформить заказ, что приводит к негативному пользовательскому опыту и ущербу репутации разработчиков. | 1. Открыть приложение.  2. Перейти в раздел "Настройки".  3. Попытаться изменить настройки.  4. Наблюдать за результатом. | Иногда | Низкая | Обычная | Пользователь испытывает незначительные неудобства | Да | Не сильно критично, можно оставить на потом. |

Продолжение таблицы 1