Федеральное государственное автономное образовательное учреждение высшего образования

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Высшая школа бизнеса

# ОТЧЁТ

## Управление требованиями и проектирование ИС. Проектирование приложения для планирования путешествий “ComfortTravel”

Выполнили:
Серебренников Данил Михайлович, ББИ227
Наливайченко Дарья Дмитриевна, ББИ 277
Таболина Наталья Владимировна, ББИ227

Проект соответствует / не соответствует требованиям (нужное подчеркнуть)

Москва 2024

ОГЛАВЛЕНИЕ

# ГЛОССАРИЙ

1. ИС – Информационная система.
2. ComfortTravel – Приложение для планирования путешествий.
3. ТЗ – Техническое задание.
4. MoSCoW – Методика приоритизации требований, которая разделяет задачи на 4 категории: Must, Should, Could, Would.
5. KPI – Ключевые показатели эффективности.
6. API (Application Programming Interface) – это программный интерфейс, который позволяет приложениям, веб-сервисам и программам взаимодействовать и обмениваться данными между собой.
7. OAuth – Протокол авторизации для доступа к ресурсам.
8. UI/UX – Пользовательский интерфейс/опыт пользователя.
9. iOS – Операционная система мобильных устройств Apple.
10. Android – Операционная система для мобильных устройств.
11. Кроссплатформенность – Способность приложения работать на разных операционных системах.
12. Push-уведомления – Сообщения, которые приложение отправляет пользователям.
13. Инкрементная модель – Модель разработки, предполагающая постепенное добавление функций.
14. Scrum – Методология гибкой разработки ПО.
15. REST API – Интерфейс для взаимодействия клиент-серверных систем.
16. Оффлайн-доступ – Возможность использования приложения без подключения к интернету.
17. Система мониторинга – Система для отслеживания производительности приложения и анализа проблем.
18. Аутентификация – Процесс проверки подлинности пользователя.
19. Авторизация – Процесс предоставления прав доступа пользователю после аутентификации.
20. Микросервисы – Архитектурный стиль, при котором приложение разделено на независимые модули (сервисы).
21. Тонкий клиент – Программное обеспечение, где основная обработка данных выполняется на сервере, а клиентская часть отвечает за интерфейс.
22. Масштабируемость – Способность системы поддерживать рост числа пользователей и объема данных без снижения производительности.

# ВВЕДЕНИЕ

**Название проекта:** проект по разработке приложения «Планировщик путешествий «ComfortTravel»

**Цель проекта:** выделить требования к разработке ИС, спроектировать приложение и подготовить основную документацию по ТЗ, реализации, тестированию и сопровождению

**Описание проекта:** необходимо разработать приложение для пользователей мобильных устройств, которое позволит планировать путешествия. Не подразумевается возможность покупки или бронирования, только предоставление информации и функции записной книжки. Основными функциями являются:

- Отслеживание путешествия по плану
- Просмотр основной информации по городу
- Поиск отелей/ресторанов/музеев/достопримечательностей
- Создание заметок к путешествию (список взятых вещей и фотографии)

# ТРЕБОВАНИЯ К СИСТЕМЕ

## Список вопросов к требованиям

1. Какие услуги должны быть представлены в приложении «ComfortTravel»?
2. Каких целей вы хотите достичь с помощью планировщика путешествий?
3. Какую целевую аудиторию видите основными пользователями/клиентами?
4. Какие временные рамки и бюджет для реализации проекта?
5. Есть ли примеры конкурентных продуктов, которые стоит изучить? Что вам в них нравится или не нравится?
6. Какие платформы (веб-версия, мобильные приложения и т.д.) будут использоваться для планировщика путешествий?
7. Должны ли быть личные кабинеты у пользователей? Если да, то через что планируется осуществлять регистрацию?
8. Какую информацию о пользователях вы планируете собирать и как она будет использоваться?
9. Какие географические регионы/территории будут являться приоритетными на начальном этапе запуска приложения?
10. Будет ли маршруты включать только крупные туристические города или также будет предоставлять информацию о малоизвестных городах и направлениях?
11. Какую информацию о местах назначения вы хотите предоставлять пользователям в рамках планировщика?

12. Какой список видов культурных и развлекательных мест? Если подразумевается составление списка по каждому из видов, то по каким критериям они выбираются?
13. Могут ли пользователи оставлять отзывы к культурным и развлекательным местам? Если да, какие должны быть критерии для написания отзывов от пользователей?
14. Какой список вещей для функции «заметки»?
15. Какие функции распространяются на составление плана путешествия? Что должно входить в описание каждой из «точек» путешествия?
16. Подразумевается ли отслеживание выполнения «пунктов» путешествия в реальном времени?
17. Какое описание к культурным и развлекательным местам требуется? Какие обязательные и желательные пункты?
18. Могут ли пользователи осуществлять поиск по культурным и развлекательным местам? Если да, какие критерии поиска должны быть осуществлены?
19. Какие сайты для поиска и выгрузки информации желательны, а какие нет? Какие, по Вашему мнению, у них достоинства и недостатки?
20. Какие дополнительные функции вы хотите включить в планировщик? Бронирование мест, автоматическое построение маршрутов и т.п.
21. Каковы требования и пожелания к пользовательскому интерфейсу?
22. Каков ваш план по интеграции с внешними системами (например, сервисами по бронированию, картами и пр.)?
23. Какие атрибуты качества для приложения важны для вас (например, скорость работы, безопасность, масштабируемость)?
24. Каковы ваши ожидания по поводу поддержки и обновлений системы после ее запуска?
25. Какие ключевые показатели эффективности (KPI) планируете использовать для отслеживания оценки результативности и успеха проекта?
26. Каким образом планируется монетизировать приложение?
27. Какие преимущества предлагаете пользователям в сравнение с уже существующими альтернативами?
28. Какие стратегии удержание клиентов планируются?
29. Какую основную проблему пользователей вы решаете данным планировщиком путешествий?
30. Будет ли предоставлена возможность оффлайн-доступа к планировщику?
31. Будет ли планировщик включать социальные функции? (Например, обмен составленными маршрутами)

**Основные выявленные требования и ограничения**

На основе ответов на вышеперечисленные вопросы, наша команда составила основной список требований к приложению:

1. Надежность и безопасность системы: защита личных данных пользователя, предотвращение несанкционированного доступа к аккаунту, в особенности при регистрации с помощью посторонних приложений (VK, Google)
2. Хранение истории путешествий, маршрутов, заметок: возможность просмотра прошедших поездок
3. Удобный и быстрый доступ к личному кабинету и функциям приложения: понятный интерфейс и быстрая отзывчивость приложения
4. Автоматическое обновление “плана” путешествия в реальном времени: отметка пройденных этапов
5. Масштабируемость длины “плана” путешествия: возможность добавления новых пунктов без вреда интерфейсу и доступа к пунктам
6. Поддержка и обновление системы: регулярное обеспечение безопасности, устранение ошибок, возможное добавление функций и изменение интерфейса
7. Автоматическое добавление отзывов и обновление рейтинга: регулярное обновление базы данных с отзывами пользователей о различных культурных и развлекательных местах
8. Ограничение по выбору даты и времени для пунктов плана путешествия: невозможность выбрать прошедшие даты и время, а также каждый новый пункт должен быть позже предыдущего
9. Кроссплатформенность: доступность приложения как на IOS, так и на Android
10. Хранение базы данных с информацией по городам и регулярное обновление актуальности
11. Возможность оффлайн доступа к приложению

Одновременно с этим мы отметили рамки в выполняемом проекте, которые показывают границы при разработке приложение, его интерфейса и функций:

1. Программное обеспечение будет разработано исключительно для планирования путешествий и предоставления информации по туристическим местам. Интеграция с внешними системами бронирования отелей и билетов не будет реализована.
2. Приложение будет поддерживать масштабирование: будет включена поддержка приложения при увеличении количества пользователей, новых городов и длины самого маршрута.

3. Аутентификация и авторизация пользователей будет включена в функциональность программного обеспечения. При этом привязка к аккаунту будет осуществлена через логины в социальных сетях.
4. Обработка и хранение персональных данных пользователей будут реализованы с учетом основных требований по безопасности
5. «ComfortTravel» создаётся исключительно для мобильных пользователей, при этом будет разработано адаптировано под последние версии операционных систем
6. Локализация будет ограничена лишь русским и английским языками. Поддержка других языков не планируется на этапе первой версии продукта

**Подробное техническое задание**

После того, как основное понимание функционала ИС и ограничений по проекту было выявлено, мы составили более полный список требований, который был приоритизирован по методу MoSCoW. Этот способ позволяет разделить задачи на 4 категории: must, should, could и would. Это поможет нам распределить ресурсы в первую очередь на самые важные требования по проекту. Ниже мы рассмотрим каждую категорию подробнее.

## 1. Введение

1.1 Наименование системы:

Автоматизированная система «Comfort Travel» для планирования персонализированных маршрутов.

1.2 Наименование и условное обозначение темы работы:

Разработка мобильного приложения для планирования путешествия по России с возможностью выбора культурных и развлекательных мест

1.3 Основание для разработки:

Основанием для разработки является утвержденное техническое задание от заказчика.

1.4 Источники разработки:

Система разрабатывается на основе технического задания, опроса потенциальных пользователей, внутренних исследований и анализа текущих потребностей целевой аудитории в планировании поездок.

## 2. Назначение и цели создания системы

Основным функционалом приложения является создание маршрутов путешествий по России. Так как это востребованное направление сейчас, то есть необходимость в создании собственного отечественного продукта, который составит конкуренцию зарубежным ПО. Данная система

предназначена для планирования персонализированных маршрутов путешествий с возможностью выбора мест, культурных объектов, и развлечений на основе предпочтений пользователя.
Система позволяет:

- Создать и управлять личным аккаунтом пользователя;
- Создавать индивидуальные маршруты;
- Отслеживать планируемые события;
- Предоставлять информацию о достопримечательностях;
- Управлять заметками и историей путешествий.

## 3. Требования к системе

3.1 Функциональные требования

3.1.1 Категория Must (задачи, без выполнения который продукт не будет работать в принципе или не имеет смысл существования)

- Личный кабинет пользователя. Пользователи должны иметь возможность регистрации, авторизации и изменения данных профиля.
- Создание маршрутов. Возможность создания персонализированных маршрутов, с выбором места на основе их предпочтений и интересов
- Отслеживание прогресса. Доступ к постоянному отслеживанию плана своего путешествия, включая этапы маршрута и запланированные мероприятия
- Справочник по городам. Доступ у пользователя к информации о городах, включая основные достопримечательности, мероприятия и транспорт
- Планирование путешествия доступно только для определенных городов и регионов. Если по части путешествия нет информации в базе, приложение должно ограничить доступ.
- Для следующего пункта путешествия нельзя выбрать дату и время, которые уже прошли или которые будут раньше «предыдущего» пункта в плане путешествия
- Справочник по достопримечательностям. Возможность поиска информации по названию культурных и развлекательных мест

3.1.2 Категория Should (задачи, которые также должны быть выполнены, но они не первостепенные)

- Оффлайн-доступ. Возможность сохранения плана путешествия для отслеживания без мобильной сети
- Разграничение функционала, который доступен и не доступен при оффлайн пользовании

- Заметки. Создание заметок двух типов: папка с фотографиями и комментариями, а также чек-лист взятых вещей для проверки багажа
- Доступ к истории путешествий и заметкам через учетную запись
- Поддержка нескольких языков, а именно русского и английского
- Обратная связь по достопримечательностям. Возможность оставлять отзывы о культурных и развлекательных местах
- Push-уведомления как напоминание при приближении пункта плана
- Регулярное обновление информации по городам

3.1.3 Категория Could (задачи, которые желательны, но которые стоит выполнить, если остались ресурсы)

- Возможность делиться планами путешествий с друзьями или другими пользователями
- Функциональность для просмотра маршрутов на карте, с возможностью прокладывать путь и определять время пути между выбранными точками
- Поиск не только по названию, но и по категориям
- Возможность сортировки культурных и развлекательных мест по рейтингу и по популярности

3.1.4 Категория Would (задачи, которые хотелось бы сделать, но они необязательны и могут быть реализованы позднее)

- Возможность нескольким пользователям совместно составлять план путешествия
- Настройка профиля. Изменение личного профиля (аватарки, фон, статус)

## 3.2 Нефункциональные требования

3.2.1 Категория Must

- Документация, описывающая функциональность и инструкции по созданию маршрутов и использованию карт
- Высокая производительность и быстрое выполнение операций. Время ожидания для загрузки плана и отображения информации должно оставаться в пределах установленного лимита.
- Надежность и устойчивость к сбоям
- Резервное копирование данных

3.2.2 Категория Should

- Система должна быть доступна не менее 99% времени
- Соответствие требованиям безопасности при обработке и хранении данных, а также законодательству и другим регуляторным требованиям

- Удобный и понятный интерфейс
- Поддержка масштабирования при увеличении количества пользователей, новых городов и длины самого маршрута
- Система отчетности и мониторинга производительности для оперативного реагирования на проблемы и улучшения системы
- Быстрая загрузка приложения
- Реализация системы оперативных обновлений и патчей для устранения уязвимостей и добавления новых функций

3.2.3 Категория Could

- Визуальные отметки при прохождении пунктов путешествия, чтобы выделить прогресс

## 4. Условия эксплуатации

4.1 Окружение:

Приложение должно работать на мобильных устройствах под управлением iOS и Android.

4.2 Минимальные требования к аппаратной и программной платформах:

Минимальные требования включают 2 ГБ оперативной памяти и 500 МБ свободного места на устройстве. Должна быть осуществлена поддержка последних версий IOS и Android.

## 5. Сроки и этапы разработки

| Этап | Начало | Окончание |
|---|---|---|
| Инициация проекта | 16.10.2024 | 30.10.2024 |
| Разработка ТЗ | 30.10.2024 | 13.11.2024 |
| Разработка первого инкремента | 13.11.2024 | 11.12.2024 |
| Тестирование первого инкремента | 11.12.2024 | 25.12.2024 |
| Запуск первого инкремента | 25.12.2024 | 09.01.2025 |
| Планирование второго инкремента | 09.01.2025 | 30.01.2025 |
| Разработка второго инкремента | 30.01.2025 | 13.03.2025 |
| Тестирование второго инкремента | 13.03.2025 | 27.03.2025 |

| | | |
|---|---|---|
| Запуск второго инкремента | 27.03.2025 | 03.04.2025 |
| Пострелизная поддержка | После запуска второго инкремента | - |

### 6. Критерии успешности проекта

Данных раздел содержит критерии, которые заказчик и наша команда совместно выделили для реализации проекта:

1. Все требуемые функции реализованы в два этапа
2. Разработанный первый инкремент позволяет выпустить приложение не позднее 09.01.2025
3. Разработанный второй инкремент позволяет выпустить приложение не позднее 03.04.2025
4. Разработанные системы и процессы задокументированы
5. Бюджет проекта не превышен
6. Все сроки этапов разработки соблюдены

## МОДЕЛЬ ЖИЗНЕННОГО ЦИКЛА И МЕТОДОЛОГИЯ РАЗРАБОТКИ

### Модель жизненного цикла ПО

Для проекта по разработке приложения «Планировщик путешествий ComfortTravel» могут подойти несколько моделей ЖЦ. При анализе мы выбрали несколько наиболее подходящих, чтобы сравнить их преимущества и недостатки, а далее выделить лучшую. Наша команда остановилась на таких моделях, как Водопадная, Спиральная и Инкрементная. Рассмотрим каждую подробнее:

**Водопадная модель (Waterfall)**

Последовательная модель, в которой каждая стадия жизненного цикла выполняется одна за другой. Основными плюсами являются четкая структура и завершенность каждой фазы перед началом следующей. Это позволяет снизить затраты на корректировку ПО, а также хорошо видеть процесс при разработке. Однако это также накладывает и основной минус: сложно вносить изменения на поздних этапах, так как фазы «заморожены»,

Если говорить о нашем проекте, то модель подходит, так как все требования четко сформулированы на этапе Технического задания, и значительные изменения не планируются. Однако, если заказчик решит развивать приложение и включать различные интеграции, то эта модель будет не так хороша.

**Спиральная модель (Spiral)**

Это итерационная модель, сочетающая элементы водопадной и инкрементной моделей. Разработка проходит в нескольких циклах, каждый из которых включает планирование с определением целей, анализ рисков и оценку альтернатив, проектирование продукта, оценку. Основным преимуществом является гибкость модели, что позволяет реагировать на изменения. Кроме того, за счёт анализа рисков можно проводить больше тестирований. Главный недостаток – это высокие затраты на разработку, из-за чего она больше подходит для больших проектов.

Спиральная модель сочетается с разработкой «ComfortTravel», так как с большей вероятностью приложение планирует улучшения на следующих этапах, в том числе основанные на обратной связи от пользователей. Кроме того, так как довольно много функций связанны друг с другом, то важно иметь возможность корректировки. Однако это не настолько большой проект, поэтому наша команда старается избегать излишних затрат.

**Инкрементная модель**

Разработка происходит поэтапно, в конце каждого цикла – готовый продукт. Функционал добавляется инкрементами, каждый из которых тестируется и вводится в эксплуатацию. Такая модель позволяет плавно вводить новые функции, а также даёт возможность быстрого выпуска базовой версии приложения и легкость сопровождения. Основным минусом является то, что она не подходит для разработки ПО, требующего сразу всю функциональность. Кроме того, приложение должно иметь масштабируемость по архитектуре, а весь план улучшений должен быть продуман заранее, чтобы итоговый продукт был целостным.

В нашем проекте все недостатки этой модели становятся преимуществами, так как функции можно разделить для постепенных нововведений, а также возможно небольшое расширение по концепции. Кроме того, есть возможность выпустить приложение с самыми основными требованиями (создание плана и просмотр информации по городам), а затем настроить остальное, в том числе корректируя продукт по обратной связи от пользователей.

Так как итоговое ПО подходит лишь для мобильных устройств, возможность выйти на рынок раньше может быть решающей, а благодаря тестированию и фидбеку происходит снижение рисков ошибок в будущем.

Проанализировав основные преимущества и недостатки каждой модели, наша команда пришла к выводу, что самой подходящей является Инкрементная, так как она наиболее удобна идеей с настройкой функционала (в отличие от Водопадной), а также дешевле, так как нет анализа рисков.

## Методология разработки ПО

Для проекта по разработке приложения «ComfortTravel» мы рассмотрели несколько методологий разработки ПО, которые могут подойти для реализации. Проанализировав их, мы выбрали наиболее подходящие для сравнения, чтобы оценить их преимущества и недостатки и выбрать оптимальный вариант. В итоге наша команда остановилась на таких методологиях, как **RUP**, **Agile** (в частности, **Scrum**) и **eXtreme Programming (XP)**. Рассмотрим каждую из них подробнее:

### Rational Unified Process (RUP)

Эта методология, которая делит проект на четыре фазы: инициация, детальное проектирование, разработка и развертывание. В каждой из фаз выполняются определенные задачи, и каждую итерацию проект продвигается через них.

*Недостатки для нашего проекта:*

RUP предлагает слишком обширную документацию и формальности, что может замедлить процесс разработки, сделав его менее гибким. Отсутствие гибкости критично для проекта, так как на начальных этапах приложения необходима быстрая адаптация к изменениям в требованиях.

### Microsoft Solutions Framework (MSF)

Это более гибкая методология, которая в свою очередь может быть адаптирована для различных типов проектов. Эта методология поддерживает как итеративные, так и последовательные подходы и включает принципы управления рисками, разработку архитектуры и обеспечение качества

*Недостатки для нашего проекта:*

MFS часто применяется в средах, более ориентированных на управление проектами и на корпоративные стандарты. Эта методология может стать большим препятствием для небольшой команды, где необходима важна быстрая адаптация и частые релизы.

### Extreme Programming (XP)

Данная методология ориентирована на обеспечение высокого качества программного обеспечения и быструю адаптацию к изменениям. Эта практика

подразумевает программирование в паре, непрерывное тестирование и частые релизы.

*Недостатки для нашего проекта:*

Большой акцент делается на технических аспектах и качестве кода, что может отвлечь от бизнес-ориентированных задач, которые важны в проекте, ориентированном на конечного пользователя.

**Agile**

Эта методология является набором принципов для гибкой разработки ПО, которая фокусируется на взаимодействии и сотрудничестве, быстрой адаптации к изменениям.

Данная методология в целом подходит под наш проект, однако есть более актуальная методология, которая является конкретным фреймворком, реализующим принципы Agile

**Scrum**

Это гибкий, итеративный фреймворк, который разбивает проект на короткие итерации. В каждой итерации команда стремится создать работающий инкремент продукта. Данная методология опирается на те же принципы, что и Agile.

*Почему подходит:*

- Методология позволяет команде быстро реагировать на изменения и адаптироваться к новым требованиям.
- Частые итерации позволяют постоянно получать обратную связь от пользователей и быстро вносить улучшений.
- Ежедневные встречи, ретроспективы позволяют команде всегда быть в курсе состояние проекта.
- Также это подходит для проектов с неопределенностью в требованиях и необходимостью быстрой адаптации. Это в свою очередь позволяет быстро разрабатывать и выпускать минимально жизнеспособные версии продукта (MVP) для тестирования и сбора пользовательской обратной связи.

Методология поддерживает динамичное управление приоритетами, что важно для добавления новых функций в ответ на требование рынка и пользователей.

Таким образом, Scrum является наиболее подходящей методологией. Другие методологии, такие как RUP и MSF, слишком тяжелы и формализованы для

нашего проекта, а XP слишком сфокусирован на технических практиках, что отвлекает от бизнес-ориентированного развития приложения. Однако, Agile, как общий подход, тоже подходит, но Scrum предлагает более четкую структуру и процессы, которые помогут эффективно управлять проектом.

## ДИАГРАММА ПРЕЦЕДЕНТОВ

Кроме Use-case диаграммы мы описали пользовательские сценарии, которые помогут обеспечить надежность, безопасность и комфортное использование системы. Разберем основные ошибки и как реагирует система:

1. Ошибка при регистрации через сторонние приложения (VK, Google)
   Причина: не удалось авторизоваться через стороннее приложение из-за проблем с доступом к учетной записи или сбоев в работе API.
   Пользователь видит уведомление: «Не удалось выполнить вход. Попробуйте позже или используйте альтернативный способ». Пользователю предлагается авторизироваться через почту, если его аккаунт привязан к ней.
   Действие для администратора: записывается ошибка в журнале системы с указанием причины (сбой API VK или Google).

2. Ошибка загрузки данных при просмотре истории путешествий или заметок
   Причина: потеря связи с сервером или повреждение данных. Пользователь получает уведомление: «Не удалось загрузить историю путешествий. Проверьте соединение с интернетом или попробуйте снова позже».
   Действие для администратора: в журнале фиксируется ошибка с подробным описанием проблемы. Если проблема заключалась в подключении, система инициирует автоматическую проверку доступности сервера и предпринимает меры для восстановления.

3. Ошибка при добавлении нового пункта в план путешествия (нарушение временных ограничений)
   Причина: Пользователь пытается добавить пункт с датой или временем, которые уже прошли или не соответствуют последовательности. Появляется предупреждение: «Вы не можете выбрать дату или время, которые прошли или предшествуют предыдущему пункту плана».

4. Ошибка при добавлении или обновлении отзывов и рейтингов

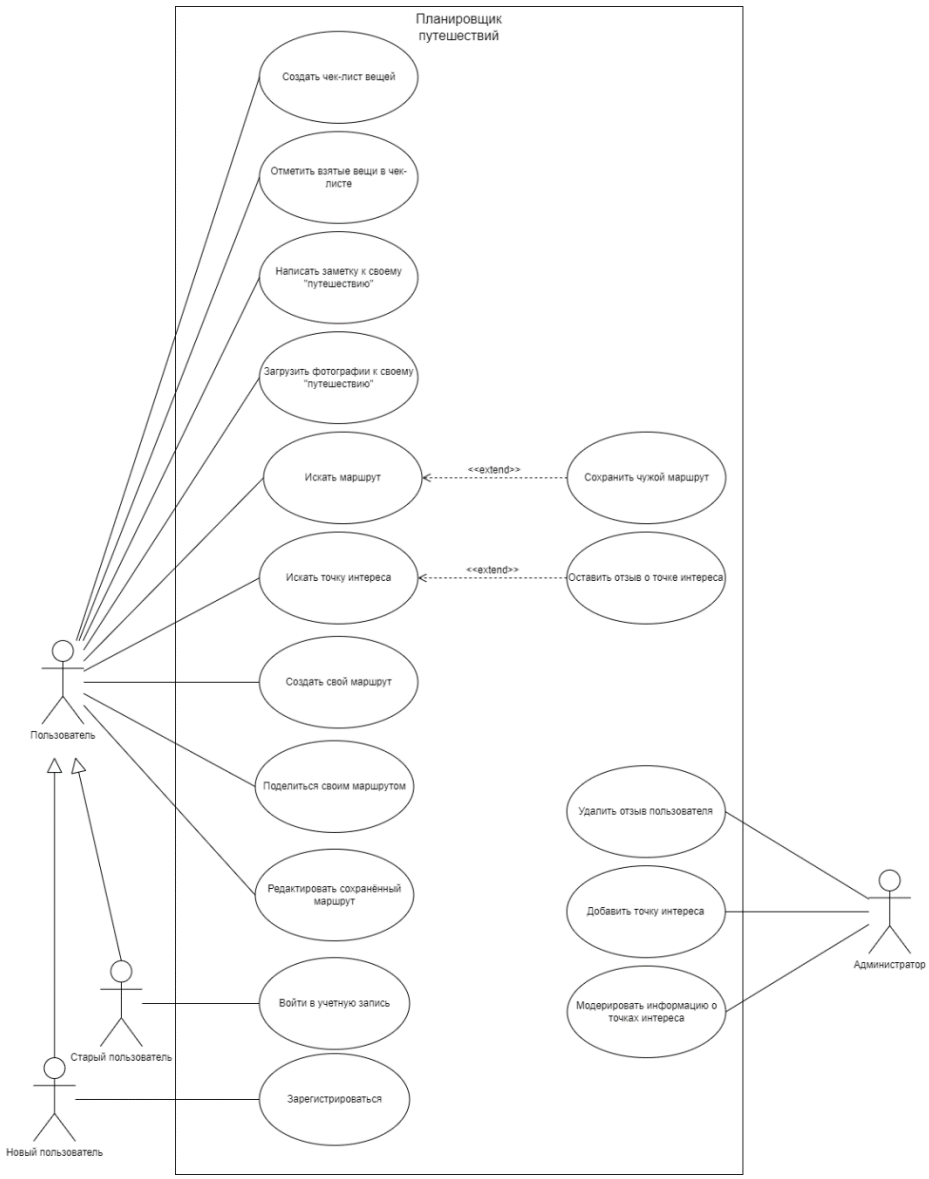
Причина: отзыв не был сохранен из-за сбоя на стороне сервера или базы данных.

Отображается сообщение: «Не удалось сохранить отзыв. Пожалуйста, попробуйте снова позже». Действие для администратора: ошибка записывается в лог с указанием времени и идентификатора отзыва.

5. Ошибка при масштабировании плана путешествия (проблемы с интерфейсом)

   Причина: Превышение допустимого количества пунктов маршрута или некорректное отображение большого числа элементов на экране. Для пользователя появляется сообщение: «Превышено максимальное количество пунктов в маршруте. Попробуйте удалить ненужные элементы или объединить некоторые из них». Визуальный интерфейс сохраняет работоспособность, но временно ограничивает добавление новых элементов.

   Действие для администратора: Ошибка масштабируемости фиксируется с предложением провести дополнительную проверку интерфейса на нагрузку и корректировку параметров отображения.




## ПЕРВИЧНОЕ ПРОЕКТИРОВАНИЕ

Связи в диаграмме включают:

- Ассоциацию: связь между сущностями, которые взаимодействуют друг с другом.
- Агрегацию: связь между сущностями, в которых один класс является частью другого.
- Композицию: связь между сущностями, в которых один класс является составной частью другого, и без него не существует.
- Наследования: связь между сущностями, в которых один класс обладает поведением и структурой ряда других классов.

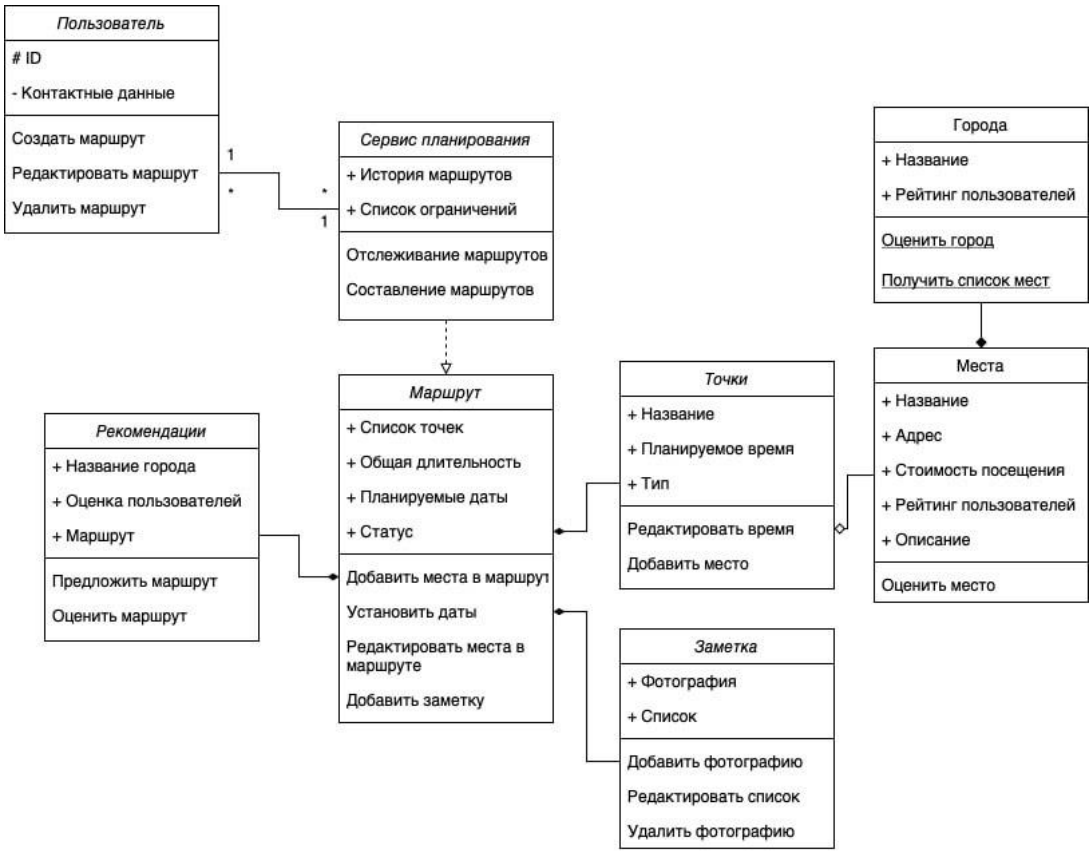
Основные выделенные классы в первичном проектировании:

- Пользователь. Методы: «Создать маршрут», «Редактировать маршрут», «Удалить маршрут»

- Сервис планирования. Методы: «Отслеживание маршрутов», «Составление маршрутов»
- Маршрут. Методы: «Добавить места в маршрут», «Установить даты», «Редактировать места в маршруте», «Добавить заметку»
- Рекомендации. Методы: «Предложить маршрут», «Оценить маршрут»
- Точки в маршруте. Методы: «Редактировать время», «Добавить место»
- Города. Методы: «Оценить город», «Получить список мест»
- Места. Методы: «Оценить место»
- Заметка. Методы: «Добавить фотографию», «Редактировать список», «Удалить фотографию»

Кроме того, были добавлены обозначения для публичных и защищенных данных. На диаграмме это изображено с помощью обозначений:

- Публичные данные (public) – обозначаются с помощью знака "+". Поле доступно всем классам.
- Защищённые данные (protected) – обозначаются с помощью знака "#". Поле доступно классу и его подклассам.
- Приватные данные (private) – обозначаются с помощью знака "-". Приватные элементы доступны только внутри самого класса.



# ДЕТАЛЬНАЯ ДИАГРАММА КЛАССОВ И ВЫБОР АРХИТЕКТУРЫ

## Выбор архитектуры

API (англ. Application Programming Interface) — это программный интерфейс, который позволяет приложениям, веб-сервисам и программам взаимодействовать и обмениваться данными между собой.

При разработке любого программного продукта одним из главных этапов является выбор архитектуры системы. Для проекта «ComfortTravel» архитектурное решение должно учитывать особенности работы с большими объемами данных, интеграцию с внешними сервисами, обеспечение высокой производительности и безопасности системы. Рассмотрим далее более детально предложенные нам архитектуры, чтобы выявить их преимущества и недостатки в контексте целей проекта.

**«Клиент-сервер»**

Данная архитектура разделяет систему на два основных компонента:

1. Клиент. То есть программа или приложение, которое взаимодействует с пользователем и отправляет запросы на сервер.
2. Сервер. А точнее набор программ, который обрабатывает запросы от клиентов, обращается к базе данных, и возвращает результаты клиенту.

Основными преимуществами данной архитектуры являются легкость разделения функциональности на клиентскую и серверную часть; поддержка одновременной работы множества пользователей возможность и возможность увеличивать серверные мощности; централизованные хранение и обработка данных на сервере, что облегчает их управление и защиту. Кроме того, сервер целостно управляет безопасностью данных и доступом к ним, а также помогает управлять API и другими внешними системами, делая взаимодействие централизованным и контролируемым.

**REST API**

Это архитектурный стиль взаимодействия клиентских и серверных систем через стандартизированные запросы. Ключевое преимущество данной технологии заключается в ее простоте масштабирования и возможности изменять отдельные части приложения без необходимости вносить изменения в весь проект. Также REST API позволяет клиенту и серверу развиваться независимо друг от друга. Это означает, что часть приложения может обновляться без необходимости изменений сервера.

Однако несмотря на то, что REST API – эффективная и популярная технология, она не полноценной архитектура. Это организация взаимодействия между компонентами, которая не предоставляет общего описания архитектуры системы.

**Микросервисная архитектура**

Данная архитектура разделяет приложение на независимые сервисы, которые могут быть масштабированы автономно. Это является преимуществом, так как отказ одного микросервиса не приведет к сбою всей оставшейся системы. Также микросервисы позволяют легче управлять интеграциями, являясь ответственными за взаимодействие с конкретным внешним сервисом.

Однако недостатком данной архитектуры является то, что она лучше подходит для крупных и сложных систем с большим количеством независимых модулей, что для рассматриваемого нами приложения будет избыточно.

**Многоуровневая архитектура**

Она представляет собой разделение системы на несколько уровней, где каждый уровень отвечает за конкретную задачу, а именно действие с пользователем и управление данных.

Преимуществом данной архитектуры является гибкость при добавление новых функций или обновлений. Разделение также позволяет изолировать данные от прямого взаимодействия с клиентами, повышая тем самым безопасность. К недостаткам относится сложность в реализации. Требуется больше времени и ресурсов на проектирование, разработку и настройку между уровнями. Разделение на уровни может вызывать большую задержку, если они разделены на нескольких сервисах.

**Serverless**

Это облачная архитектура, в которой разработчики предоставляют функции, способные автоматически запускаться облачным провайдером по требованию, без необходимости управления серверами. Главным преимуществом является отсутствие необходимости управления физически и виртуально. Это позволяет сосредоточиться на разработке и улучшение функциональности, а не инфраструктуры. Также благодаря облачному провайдеру данная архитектура обеспечивает высокий уровень устойчивости к сбоям и доступности функций.

К недостаткам можно отнести отсутствие сохранения и управления сессий пользователя, что в нашем проекте является важным критерием. Также стоит обратить внимание на то, что эта архитектура выгружает функции, которые длительное время могут не использоваться. Это в свою очередь может ухудшить пользовательский опыт при высоких загрузках за счет сниженной скорости обработки запроса.

Так как наше ПО небольшое, то архитектура «**клиент-сервер**» является самым оптимальным выбором для проекта «ComfortTravel». Она обеспечивает баланс между реализацией безопасности и масштабированием, а также подходит для мобильных приложений. Данная архитектура позволит легко управлять бизнес-логикой и осуществить все ключевые функции приложения, такие как планирование маршрутов, интеграция с внешними системами, не используя сложные и дорогостоящие архитектурные решения.

Рассмотрим подробнее использование элементов архитектуры.

**Тип клиента: Тонкий клиент**

Тип работы, где основная логика и обработка данных (поиск информации, маршрутов, добавление отзывов и их обновление) будет происходить на сервере, что позволит уменьшить нагрузку на мобильные устройства пользователей и ускорить приложение. Клиентская же часть отвечает за отображение интерфейса и взаимодействие с пользователем. Кроме того, тонкий клиент позволяет минимизировать риск возникновения неисправностей и предоставляет довольно низкие требования к оборудованию. К одному серверу может быть подключено неограниченное количество клиентов. Однако при ошибке на сервере страдают все пользователи.

**Наличие баз данных:**

Для реализации идеи приложения, требуется централизованное хранилище данных для городов, мест, маршрутов, истории путешествий и отзывов. Именно из-за недостатка тонкого клиента наличие баз данных необходимо, чтобы сохранять информацию и не потерять её при сбоях. Кроме того, подобное хранение предоставляет удобный доступ к необходимым данным, регулярное обновление, безопасность и масштабируемость. Пользователи смогут синхронизировать свои данные и историю путешествий, даже при смене устройств или восстановлении аккаунта

**Сервер приложения:**

Для обработки запросов и работы с базой данных необходимо использовать выделенный сервер приложения, который будет управлять данными маршрутов, работать с внешними API, синхронизировать их в режиме реального времени и обновлять отзывы пользователей. Логика данных и бизнес-логика находятся как раз на уровне сервера приложений. Все обращения клиентов к базе данных происходят также через данный уровень, из-за чего повышается гибкость работы и производительность.

**Масштабируемость:**

Для реализации приложения архитектура должна легко масштабироваться с увеличением количества пользователей и объема данных.

**Оффлайн-доступ:**

Клиент-серверная архитектура позволяет сохранять данные в кэше на клиентской стороне для оффлайн-доступа. Данная функция обеспечит удобный доступ к приложению в путешествиях.

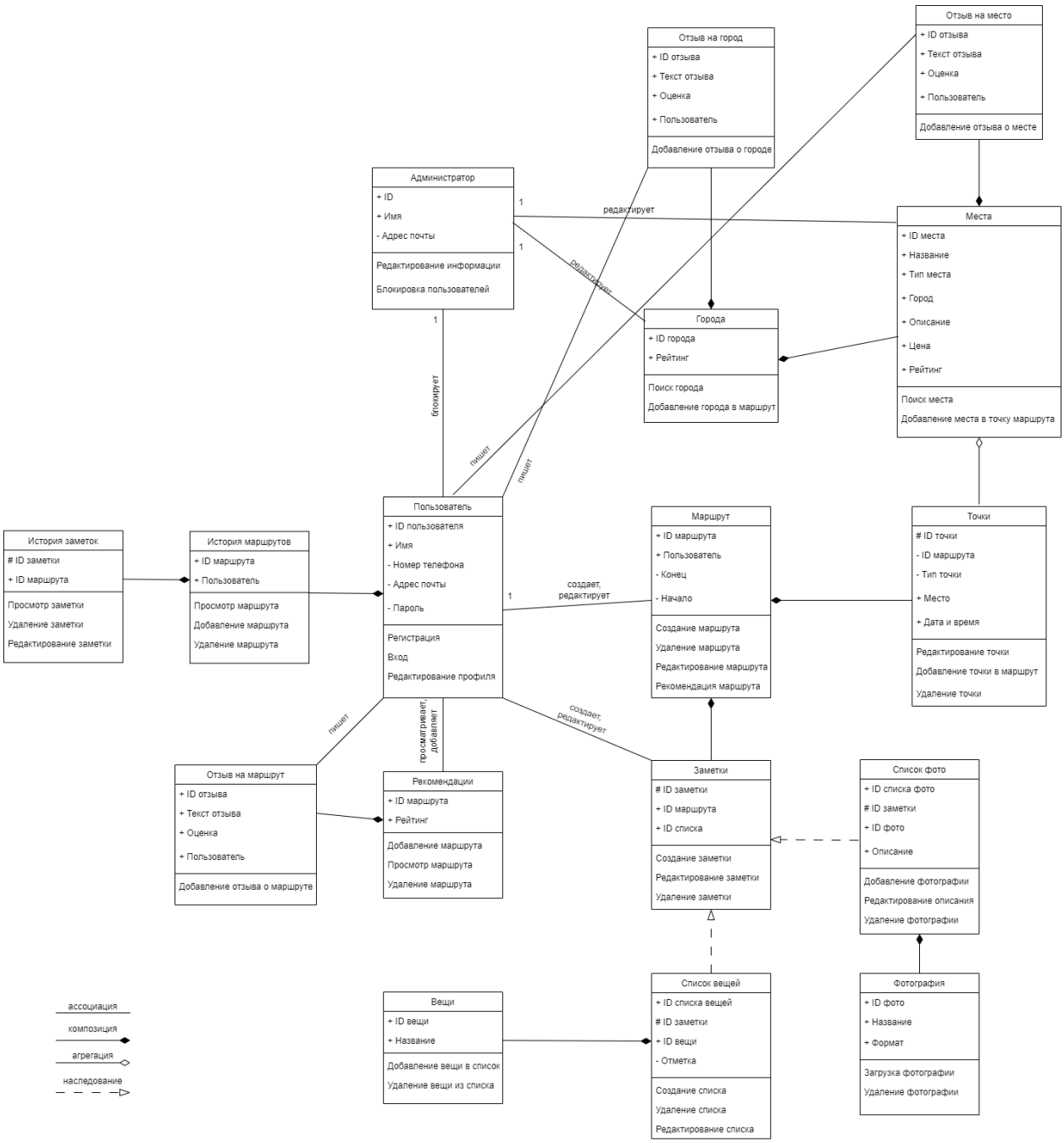
## Детальная диаграмма классов

Связи в диаграмме включают:

- Ассоциацию: связь между сущностями, которые взаимодействуют друг с другом.
- Агрегацию: связь между сущностями, в которых один класс является частью другого.
- Композицию: связь между сущностями, в которых один класс является составной частью другого, и без него не существует.
- Наследования: связь между сущностями, в которых один класс обладает поведением и структурой ряда других классов.

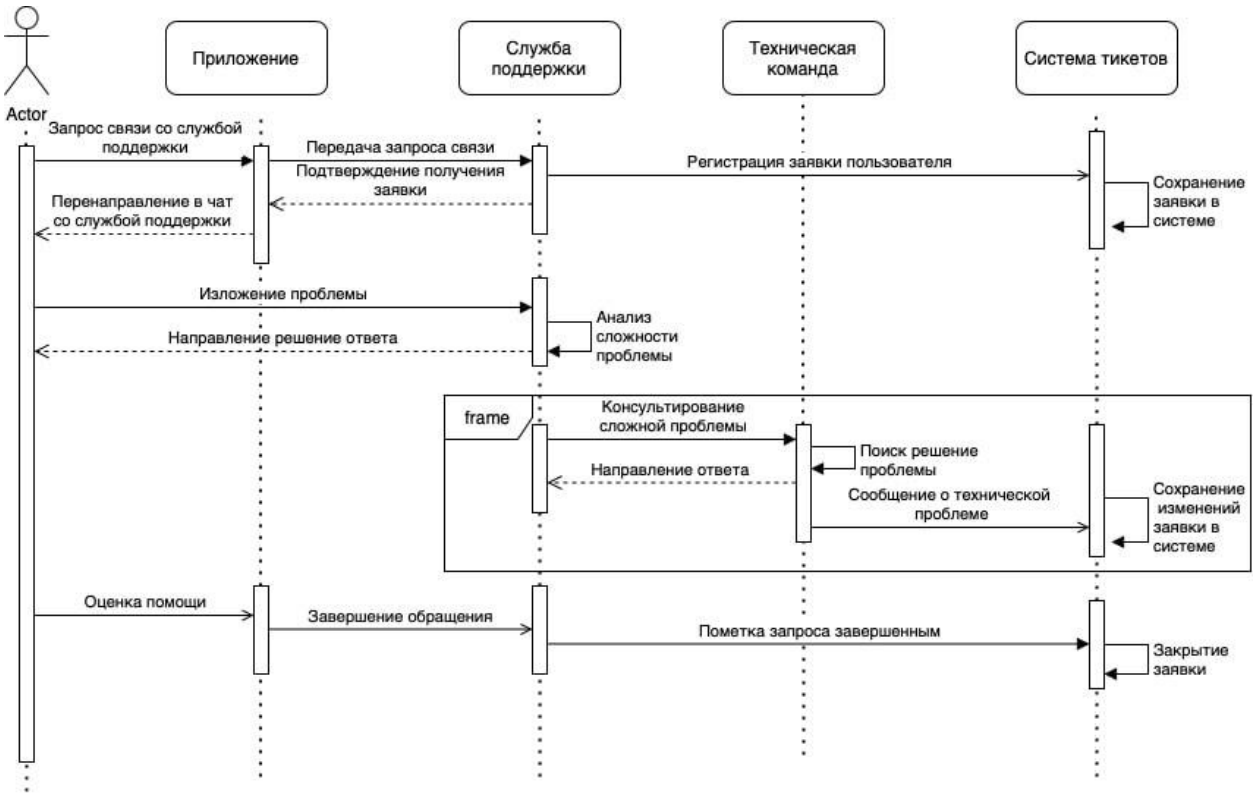
Кроме того, были добавлены обозначения для публичных и защищенных данных. На диаграмме это изображено с помощью обозначений:

- Публичные данные (public) – обозначаются с помощью знака "+". Поле доступно всем классам.
- Защищённые данные (protected) – обозначаются с помощью знака "#". Поле доступно классу и его подклассам.
- Приватные данные (private) – обозначаются с помощью знака "-". Приватные элементы доступны только внутри самого класса.

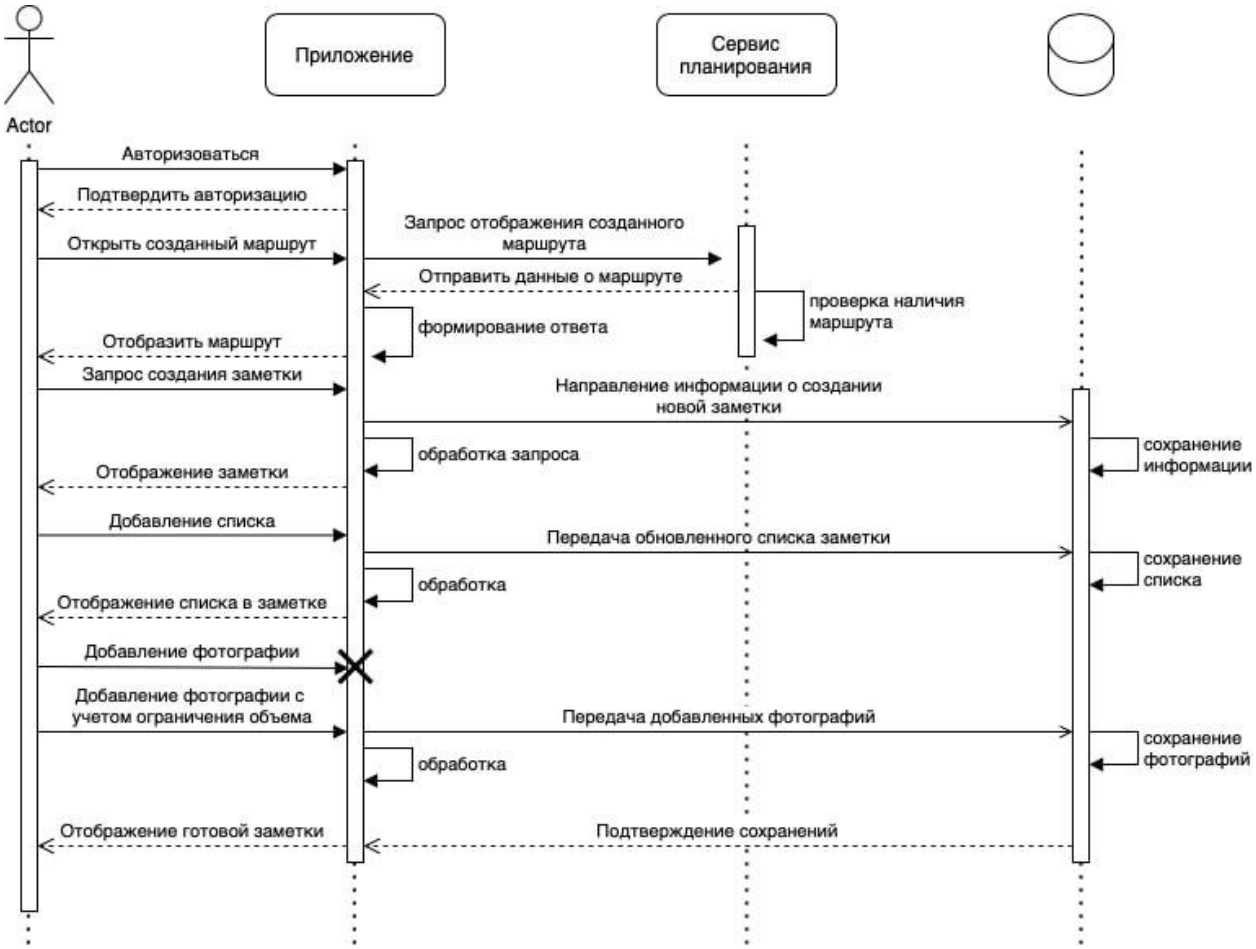



# ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТЕЙ SEQUENCE DIAGRAM

## Служба поддержки




Создание и работка с заметками



Создание и работа с маршрутом

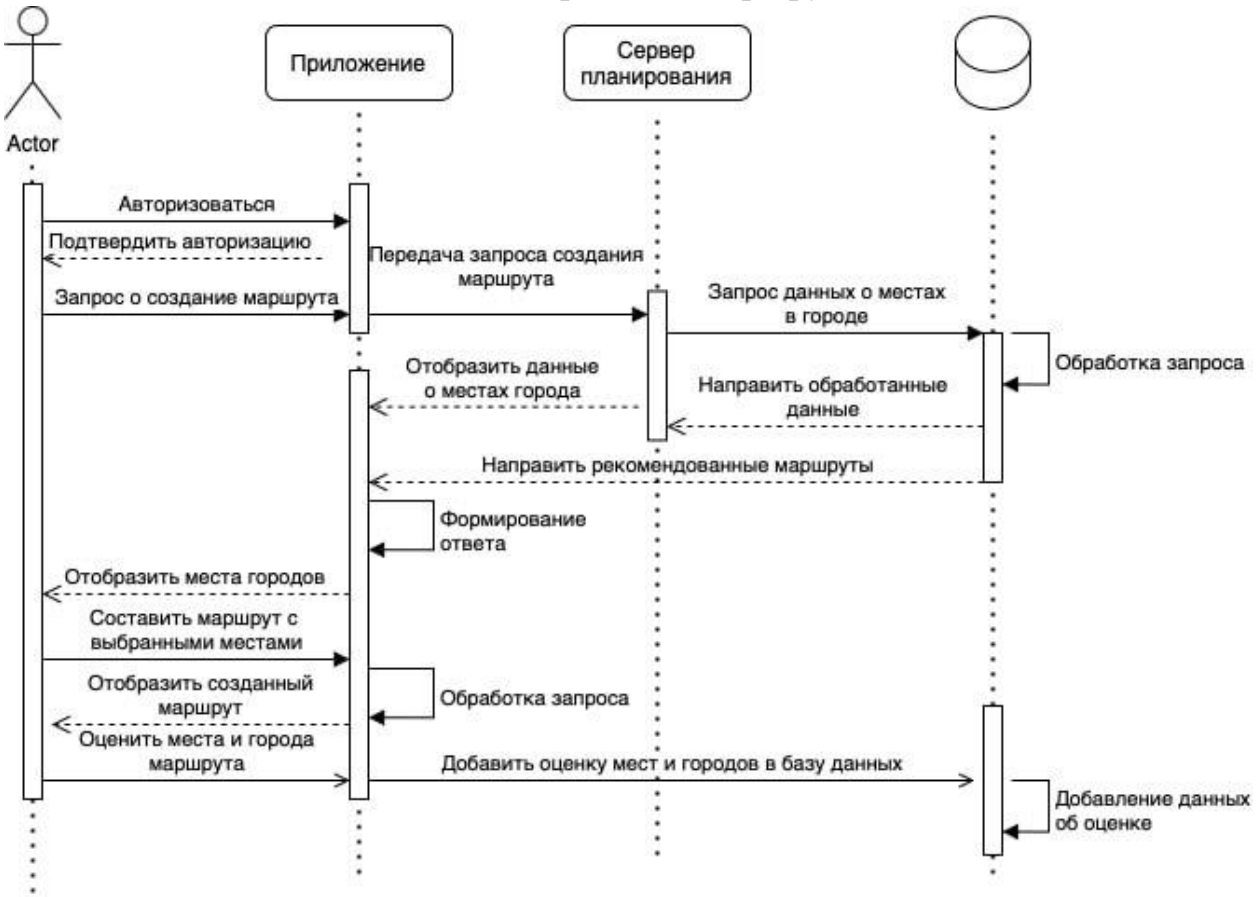
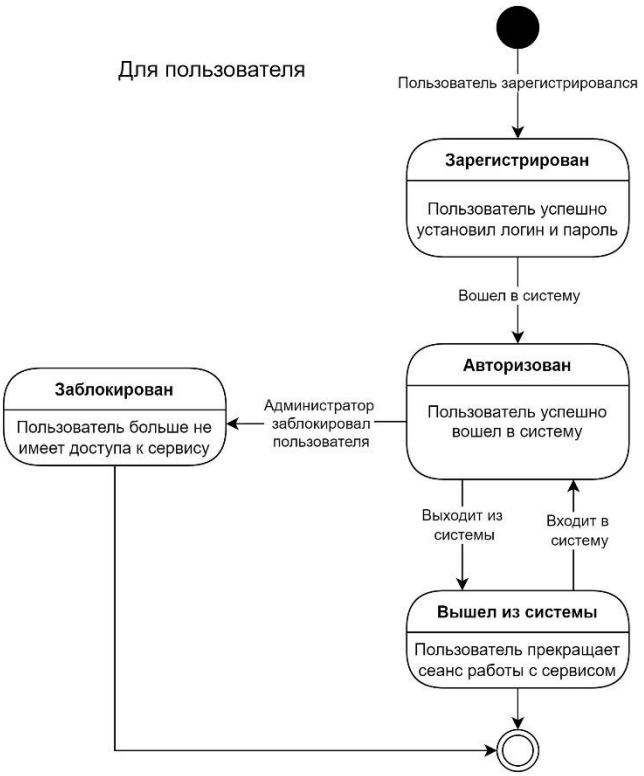
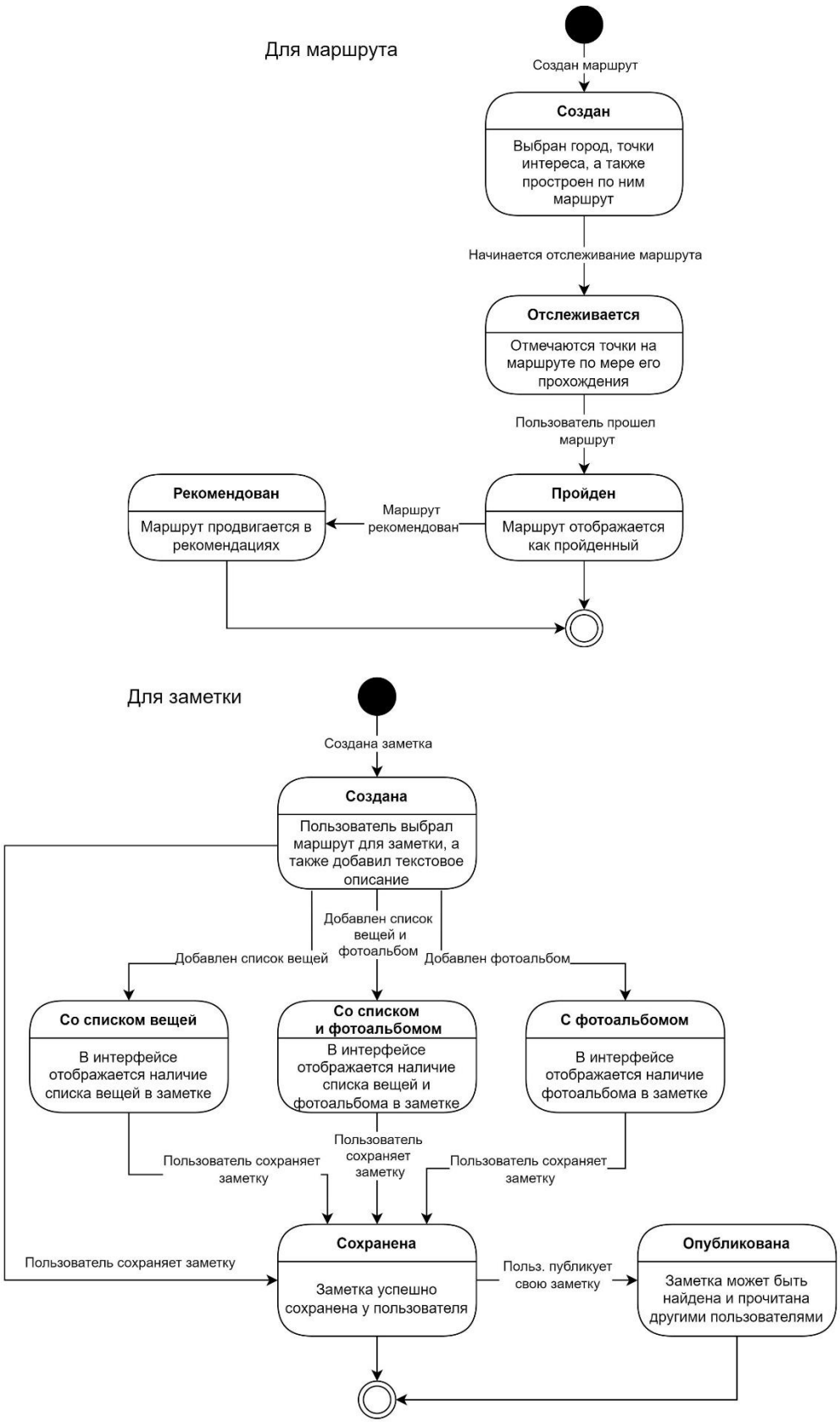



ДИАГРАММА СОСТОЯНИЙ STATE CHART DIAGRAM




Для маршрута
Создан маршрут
Создан
Выбран город, точки интереса, а также простроен по ним маршрут
Начинается отслеживание маршрута
Отслеживается
Отмечаются точки на маршруте по мере его прохождения
Пользователь прошел маршрут
Пройден
Маршрут отображается как пройденный
Маршрут рекомендован
Рекомендован
Маршрут продвигается в рекомендациях
Для заметки
Создана заметка
Создана
Пользователь выбрал маршрут для заметки, а также добавил текстовое описание
Добавлен список вещей
Добавлен список вещей и фотоальбом
Добавлен фотоальбом
Со списком вещей
В интерфейсе отображается наличие списка вещей в заметке
Со списком и фотоальбомом
В интерфейсе отображается наличие списка вещей и фотоальбома в заметке
С фотоальбомом
В интерфейсе отображается наличие фотоальбома в заметке
Пользователь сохраняет заметку
Пользователь сохраняет заметку
Пользователь сохраняет заметку
Сохранена
Заметка успешно сохранена у пользователя
Пользователь сохраняет заметку
Польз. публикует свою заметку
Опубликована
Заметка может быть найдена и прочитана другими пользователями

# РЕВИЗИЯ ДОКУМЕНТАЦИИ К ПРОГРАММНОМУ ПРОДУКТУ

В ходе ревизии документации к программному продукту была проведена всесторонняя проверка всего, что было разработано в рамках проекта. Были проанализированы соответствия между различными артефактами проекта, трассируемость моделей, а также возможные ошибки, упущения или избыточность.

При проверке не было выявлено противоречий между терминами, используемыми в диаграммах, таблицах и моделях. Кроме того, все модели, диаграммы, таблицы и требования отслеживаются друг через друга. Было также выявлено, что ничего не конфликтует с бизнес-логикой разрабатываемого приложения, кроме диаграмм Последовательностей и Состояний, а все названия классов, методов и терминов были сопоставлены между собой в документации. Все ошибки были исправлены.

Анализируя модели, мы проверили избыточные, упущенные или конфликтующие требования, а также наличие всех артефактов. Все необходимые диаграммы и части отражены в ходе нашего проекта, в том числе планы реализации, тестирования и эксплуатации.

При проверке были обнаружены некоторые ошибки, которые наша команда исправила. А именно:

- ТЗ. Функциональные и нефункциональные требования были оформлены в соответствии с ГОСТ 34.602-89. Было проведено структурирование по важности и разделение на смысловые разделы.
- Диаграмма детального проектирования. Были неправильно обозначены отношения между классами и методами. Диаграмма изменена для более корректного отображения структуры данных при проектировании: добавлены наследование, композиция и агрегация.
- Дополнительные сценарии. В ходе разработке приложения наша команда столкнулась с недостаточным пониманием пользовательских и системных сценариев при разных ошибках и сбоях. Это являлось недостатком при разработке, поэтому в разделе «Диаграмма прецедентов» добавлены описания сценариев в ИС при различных ошибках.
- Публичные и защищенные классы. При создании таких артефактов как «Диаграмма первичного проектирования» и «Диаграмма детального проектирования» наша команда не учла защищенность классов, что также усложняет разработку. Модели были изменены для более точного отображения данных.

- Глоссарий. При проверке для более легкого понимания проекта был добавлен структурированный список терминов, используемых при проектировании.
- Диаграмма последовательностей. Добавление use-case в рассмотренные диаграммы.
- Диаграмма состояний. Изменение статусов состояний.

МАТРИЦА ТРАССИРОВКИ ТРЕБОВАНИЙ

| Бизнес-требования | Функциональные требования | Нефункциональные требования | Ограничения |
|---|---|---|---|
| Возможность создавать и планировать путешествия | 1) Создание маршрутов<br>2) Добавление точек из списка мест<br>3) Редактирование маршрута<br>4) Оценка и рекомендация маршрутов<br>5) Возможность удалять маршруты | 1) Система должна обрабатывать маршруты не дольше трех секунд<br>2) Должна быть быстрая доступность данных | 1) Приложение может поддерживаться только на мобильных устройствах |
| Информация о городах и достопримечательностях | 1) Поиск информации о городах и достопримечательностях<br>2) Добавление мест в маршруты | 1) Актуальная информация о статусе мест и городов должна обновляться несколько раз в день | Данные должны быть только из открытых API |
| Пользователь должен иметь возможность управлять заметками | 1) Создание заметки<br>2) Возможность добавлять и редактировать списки<br>3) Возможность прикреплять фотографии<br>4) Возможность удаления заметок | 1) Система должна обрабатывать изменения в заметке не дольше трех секунд | Хранение заметок пользователя может быть ограниченно внутренними базами приложений |
| Безопасность данных | 1) Защищенный вход в личный кабинет с необходимой аутентификацией | 1) Шифрование личных данных пользователя<br>2) Данные должны храниться в соответствие с требованиями GDPR | Способы шифрования и хранения данных пользователей должны осуществляться в соответствие с законодательством по защите данных в странах ЕС |
| Интерфейс и управление должны быть удобными и простыми для пользования | 1) интуитивно понятный в использовании интерфейс<br>2) Возможность пользователю настраивать интерфейс в соответствии с его предпочтениями | 1) Время отклика интерфейса не должно занимать больше отведенного лимита времени | Учет доступности интерфейса для мобильных устройств с различными разрешениями экрана |
| Масштабируемость | – | 1) При масштабируемости приложение не должно лишиться работы и эффективности производительности<br>2) Поддержка интерфейса при увеличение количества этапов маршрута и заметок | 1) При масштабируемости не должна пострадать базовая инфраструктура |
| Приложение должно иметь возможность оффлайн-доступа | 1) Доступ к маршрутам в оффлайн-режиме<br>2) Доступ к заметкам в оффлайн- режиме<br>3) Синхронизация внесенных изменений при переходе к онлайну | Оффлайн данные должны занимать минимальный объем памяти, для оптимизации хранения данных в памяти. | Оффлайн доступ должен предоставлять только просмотр, без возможности вносить изменения в маршруты или заметки |

ПЛАН РЕАЛИЗАЦИИ ПРОЕКТА

## 1. Введение

Название проекта: Разработка мобильного приложения «ComfortTravel», предназначенного для планирования персонализированных маршрутов путешествий

Цель проекта: Создание мобильного приложения, которое позволит пользователям планировать путешествия, отслеживать и создавать свои

маршруты, а также просматривать информацию о достопримечательностях, мероприятиях и культурных объектах.
Модель жизненного цикла: инкрементная модель
Сроки реализации: проект должен быть выполнен за 6 месяцев и сдан не позднее 15.04.2025.

## 2. Этапы реализации

2.1 Инициация проекта
Сроки:
На данный этап отводится 2 недели, то есть крайний срок 30.10.2024.
Задачи:

- Определение целей и границ проекта
- Формирование бизнес-требований
- Утверждение критериев сдачи проекта

Результаты:

- Цели и основные требования к проекту сформированы и задокументированы
- Определены сроки выполнения проекта
- Выделен бюджет на реализацию
- Разработан устав проекта

Ресурсы:

- Проджект-менеджер (1 человек)
- Бизнес-аналитика (2 человека)

2.2 Сбор требований, планирование и разработка ТЗ
Сроки:
На данный этап отводится 2 недели, то есть крайний срок 13.11.2024.
Задачи:

- Проведение интервью с потенциальными пользователями
- Описание бизнес-логики
- Сбор функциональных и нефункциональных требований от заинтересованных сторон
- Приоритизация требований по методу MoSCoW
- Планирование инкрементов
- Создание макета первого инкремента
- UX-исследование

Результаты:

- Сформулировано и задокументировано подробное финальное Техническое задание
- Определен список требований к первому инкременту
- Утвержден дизайн прототипа приложения

Ресурсы:

- Проджект-менеджер (1 человек)
- Бизнес-аналитика (2 человека)
- UX/UI-дизайнеры (2 человека)

2.3 Разработка первого инкремента

Сроки:

На данный этап отводится 4 недели, то есть крайний срок 11.12.2024.

Задачи:

- Разработка первой версии приложения и реализация основного функционала на основе ТЗ
- Интеграция с базами данных

Результаты:

- Реализован основной функционал: регистрация пользователей, создание и изменение маршрутов, просмотр информации по городам, возможность добавления точек в свой план путешествия и создание заметок
- Интегрированы базы данных и API для создания маршрутов
- Реализована административная панель для управления и мониторинга
- Первый инкремент готов к тестированию

Ресурсы:

- Проджект-менеджер (1 человек)
- Frontend-разработчики (2 человека)
- Backend-разработчики (2 человека)
- Инженер баз данных (1 человек)
- QA-инженеры (2 человека)

2.4 Тестирование первого инкремента

Сроки:

На данный этап отводится 2 недели, то есть крайний срок 25.12.2024.

Задачи:

- Тестирование функционала
- Обнаружение и исправление ошибок

Результаты:

- Неполадки устранены
- Проверен и установлен уровень защиты данных
- Протестированы сценарии при сбоях
- Производительность протестирована и налажена
- Приложение готово к запуску

Ресурсы:

- Проджект-менеджер (1 человек)
- Специалист по безопасности (1 человек)
- QA-инженеры (2 человека)

2.5 Запуск первого инкремента

Сроки:

На данный этап отводится 1 неделя, то есть крайний срок 9.01.2025 (с учётом праздников).

Задачи:

- Публикация приложения
- Настройка систем аналитики

Результаты:

- ПО доступна пользователям на IOS и Android
- Реализованы методы сбора данных для анализа работы приложения

Ресурсы:

- Проджект-менеджер (1 человек)
- Инженеры по внедрению (1 человек)
- Специалисты поддержки (2 человека)

2.6 Планирование второго инкремента

Сроки:

На данный этап отводится 3 недели, то есть крайний срок 30.01.2025.

Задачи:

- Сбор и анализ обратной связи от пользователей
- Анализ запросов на новые функции
- Изменения в изначальном разработанном плане на основе собранных данных

Результаты:

- План реализации второго инкремента скорректирован

Ресурсы:

- Проджект-менеджер (1 человек)
- Продакт-аналитик (1 человек)

2.7 Разработка второго инкремента

Сроки:

На данный этап отводится 6 недель, то есть крайний срок 13.03.2025.

Задачи:

- Реализация оставшихся функций
- Настройка оффлайн-доступа

Результаты:

- Реализован функционал: добавление и оценка чужих маршрутов, возможность поделиться своим планом, оставление отзыва на места и города, просмотр истории маршрутов и заметок, персонализация интерфейса, оффлайн-доступ
- Добавлены push-уведомления для напоминания о запланированных мероприятиях
- Реализована поддержка нескольких языков
- Разработан функционал службы поддержки
- Второй инкремент готов к тестированию

Ресурсы:

- Проджект-менеджер (1 человек)
- Frontend-разработчики (2 человека)
- Backend-разработчики (2 человека)
- Инженер баз данных (1 человек)
- QA-инженеры (2 человека)

2.8 Тестирование второго инкремента

Сроки:

На данный этап отводится 2 недели, то есть крайний срок 27.03.2025.

Задачи:

- Тестирование функционала
- Обнаружение и исправление ошибок

Результаты:

- Неполадки устранены
- Проверен и установлен высокий уровень защиты данных
- Протестированы сценарии при сбоях
- Производительность протестирована и налажена
- Оффлайн-доступ протестирован
- Приложение готово к запуску

Ресурсы:

- Проджект-менеджер (1 человек)
- Специалист по безопасности (1 человек)
- QA-инженеры (2 человека)

2.9 Запуск второго инкремента

Сроки:

На данный этап отводится 1 неделя, то есть крайний срок 03.04.2025.

Задачи:

- Развертывание новой версии приложения
- Настройка систем аналитики

Результаты:

- ПО доступна пользователям на IOS и Android

Ресурсы:

- Проджект-менеджер (1 человек)
- Инженеры по внедрению (1 человек)
- Специалисты поддержки (2 человека)

## 3. Сроки проекта

| Этап | Начало | Окончание |
|---|---|---|
| Инициация проекта | 16.10.2024 | 30.10.2024 |
| Разработка ТЗ | 30.10.2024 | 13.11.2024 |
| Разработка первого инкремента | 13.11.2024 | 11.12.2024 |
| Тестирование первого инкремента | 11.12.2024 | 25.12.2024 |
| Запуск первого инкремента | 25.12.2024 | 09.01.2025 |
| Планирование второго инкремента | 09.01.2025 | 30.01.2025 |
| Разработка второго инкремента | 30.01.2025 | 13.03.2025 |
| Тестирование второго инкремента | 13.03.2025 | 27.03.2025 |
| Запуск второго инкремента | 27.03.2025 | 03.04.2025 |

| | | |
|---|---|---|
| Пострелизная поддержка | После запуска второго инкремента | |

ПЛАН ТЕСТИРОВАНИЯ ПРОДУКТА

## 1. Введение

1.1 Цели тестирования:

• Повысить вероятность того, что тестируемый сервис будет работать правильно при любых обстоятельствах.

• Повысить вероятность того, что тестируемый сервис будет соответствовать всем описанным требованиям.

• Предоставление актуальной информации о состоянии продукта на данный момент.

1.2 Этапы тестирования:

• Анализ

• Разработка стратегии тестирования и планирование процедур контроля качества

• Работа с требованиями

• Создание тестовой документации

• Тестирование прототипа

• Основное тестирование

• Эксплуатация

1.3 Основные понятия:

• Тестирование программного обеспечения — проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле тестирование — это одна из техник контроля качества, включающая в себя действия по планированию работ, проектированию тестовой документации, выполнению тестирования и анализу полученных результатов.

• Дефект — это несоответствие фактического результата выполнения программы ожидаемому результату. Большая часть дефектов обнаруживается на этапе тестирования программного обеспечения, когда татуировщик проводит сравнение полученных результатов работы программы (отдельного компонента) с ожидаемым результатом, описанным в спецификации требований.

• Тест-план (Test Plan) — это документ, описывающий весь объем работ по тестированию, начиная с описания объекта, стратегии, дат проведения,

критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения.
• Тестовый случай (Test Case) — это документ, описывающий совокупность шагов,
конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

1.4 Процесс тестирование делится на уровни:
• Модульное тестирование или юнит-тестирование (Unit Testing) – проверка корректности работы отдельных модулей исходного кода. Цель модульного тестирования — изолировать отдельные части программы и показать, что по отдельности эти части работоспособны;
• Интеграционное тестирование (Integration Testing) – фаза тестирования программы, при которой отдельные программные модули объединяются и тестируются в группе. Целью интеграционного тестирования является проверка соответствия проектируемых единиц функциональным требованиям, приемным требованиям и требованиям надежности;
• Системное тестирование — это тестирование программного обеспечения, выполняемое на полной системе, с целью проверки соответствия системы исходным требованиям. Обычно подразделяется на альфа-тестирование (проверка корректности функционального наполнения кода) и бета-тестирование (стадия исправления ошибок, финальное тестирование программного продукта перед релизом).

Каждый следующий уровень не может быть выполнен полностью, пока не пройдены все тесты предыдущего уровня. Таким образом, процесс тестирования сводится к последовательному тестированию каждого из уровней. Обнаруженные ошибки исправляют и тестирование начинается заново с юнит-тестирования.

1.5 Тестирование в нашем случае подразделяется на три вида:
• тестирование производительности;
• юзабилити-тестирование;
• тестирование безопасности;
Тестирование производительности будет основано на изучении времени отклика системы на запрос от администратора и тайм-аутов программного комплекса. Проводится на стадии интеграционного тестирования.

Тестирование безопасности необходимо провести на стадии интеграционного тестирования с целью проверки защищенности данных пользователей.

1.6 Объекты тестирования:

Объектами тестирования в данной работе будут являться: конструктор маршрутов, блок заметок и система авторизации. Все тестирование является ручным. Так как внутреннее строение тестировщику (а в дальнейшем и пользователю) неизвестно, то тип тестирования – черный ящик. Также будет проводиться проверка системы полностью, следовательно, по степени изолированности тестируемых компонент тестирование системное.

## 2. Тест-кейсы

В данном подразделе приведены описания некоторых тест-кейсов для проверки качества разработанного программного продукта. Успешное прохождение данных тест-кейсов указывает на корректность работы ПО.

Тест-кейс №1

Общая работа конструктора маршрутов при корректных данных

| | |
|---|---|
| **Описание:** Проверка возможности создать маршрут, соответствующий требованиям | **Предусловие:** Пользователь создает пустой маршрут |
| **Шаги:** | |
| 1) Определить город для маршрута<br>2) Добавить точки интереса<br>3) Построить маршрут между точками | |
| **Ожидаемый результат: успешно** создан маршрут для конкретного города, по которому возможно пройти в указанном порядке | |

Тест-кейс №2

Общая работа блока заметок при корректных данных

| **Описание:** Проверка возможности создания заметок | **Предусловие**: пользователь создает пустую заметку |
|---|---|
| **Шаги:** | |
| 1) Пользователь определяет маршрут для заметки<br>2) Пользователь создает список вещей<br>3) Добавляет вещи в список<br>4) Сохраняет список<br>5) Создает фотоальбом<br>6) Загрузить фото<br>7) Сохраняет фотоальбом | |
| **Ожидаемый результат:** Успешна создана заметка к путешествию со списком вещей для данной поездки, а также с фотоальбомом с фотографиями из неё | |

Тест-кейс №3

Авторизация пользователя

| **Описание:** Проверка возможности авторизации существующего пользователя | **Предусловие:** Пользователь существует и не залогинен |
|---|---|
| **Шаги:** | |
| 1) Пользователь вводит логин<br>2) Вводит пароль<br>3) Нажимает кнопку “Вход” | |
| **Ожидаемый результат:** Авторизация прошла успешно, пользователь остался в приложении | |

С учетом всего вышеописанного суммарное время, которое будет затрачено на тестирование группой из 2 человек, составит примерно 2 недели. В это время включается написание всех тестов, их выполнение и исправление дефектов.

# ПЛАН ЭКСПЛУАТАЦИИ ПРОДУКТА

1. Разработка плана эксплуатации продукции помогает обеспечить эффективное функционирование системы после ее окончательного выпуска. Для проекта «ComfortTravel» данный план должен учитывать поддержку пользователей, корректное управление инфраструктурой, дальнейшее обновление и мониторинг, а также прочие аспекты эксплуатации.

## 2. Подготовка инфраструктуры

2.1. Настройка серверов и баз данных:

- Необходимо подготовить и проверить работоспособность серверную инфраструктуру для размещения сервиса планирования и баз данных.
- Убедиться в том, что базы данных способны масштабироваться и поддерживать резервное копирование данных.
- Настроить резервирование сервисов для повышения отказоустойчивости.

2. 2. Настройка систем мониторинга

Мониторинг производительности и безопасности:

- Внедрить инструменты мониторинга производительности серверов и баз данных.
- Убедиться в настройке оповещений о высоких уровнях нагрузки или критических сбоях.
- Настроить мониторинг безопасности для отслеживания подозрительной активности, несанкционированный доступ к данным и DDOS-атак.

2.3. Подготовка систем резервного копирования

Резервное копирование данных:

- Настройка регулярного резервирования копирования данных, включая пользовательскую информацию, историю маршрутов и заметок.
- Определить частоту резервных копий.
- Разработать и протестировать план восстановления данных в случае критических сбоев.

## 3. Ввод эксплуатации

3. 1. Обучение сотрудников и поддержка пользователей

- Предоставить обеспечение наличие службы поддержки пользователей через внутренний чат или электронную почту.
- Разработать и предоставить справочную документацию для пользователей и создать раздел часто задаваемых вопросов.

- Провести обучение сотрудников службы поддержки для решения частых вопросов и проблем пользователей.

- Провести обучение технической команды эксплуатации, включая разработчиков, системных администраторов работе с продуктом.

## 4. Операционная эксплуатация

4. 1. Техническое обслуживание

Регулярное обновление системы и безопасностью:

- Выпуск обновлений для исправления ошибок и улучшений производительности.

- Регулярное обновление мобильных приложений с добавлением функционала.

- Обновление сертификатов безопасности

4. 2. Мониторинг и отчетность

Производительность системы и ее анализ:

- Постоянный мониторинг использования процессоров, памяти и пропускной способности сети.

- Создание оповещений при превышении критических значений нагрузки.

- Постоянный сбор аналитики о количестве пользователей, типах устройств и самых используемых функций.

- Подготовка отчетов для управления по использованию приложения и предложений по улучшению.

4. 3. Управление обновлениями

Мелкие и крупные обновления:

- Выпуск обновлений с небольшим исправлением ошибок и улучшениями функций. Данные обновления не должны прерывать работу пользователей и должны выполняться в фоновом режиме.

- Необходимо проводить планирование крупных обновлений системы с предварительным уведомлением о запланированных технических обновленьях и возможных сбоев.

- Обеспечение совместимости старых и новых версий приложения.

4. 4. Масштабирование

Увеличение серверных мощностей:

- При увеличении нагрузки и количества пользователей необходимо иметь в распоряжение больше ресурсов и серверов.

## 5. Обновление и развитие продукта

5. 1. Добавление новых функций:

- Сбор запросов и потребностей пользователей на новые функции и улучшения приложения.

- Рассмотрение возможности добавления улучшенных технологий для планирования путешествий, искусственного интеллекта для персонализации и т. д.

5. 2. Поддержка обновленных версий

Совместимость с новыми версиями операционных систем:

- Обеспечение совместимости приложения с новыми версиями IOS и Android. А также поддержка старых версий приложений.

### 6. Завершение эксплуатации

6. 1. Архивация данных

- Если проект будет завершен или переведен на новую систему, необходимо будет обеспечить архивные копии всех данных пользователей.

6. 2. Выключение инфраструктуры

- При завершение необходимо обеспечить уведомление пользователям о сроках завершения поддержки и возможности экспорта данных.

- После завершения всех операций и архивирования данных провести вывод и других компонентов эксплуатации.

## СОПРОВОЖДЕНИЕ ПРОГРАММНОГО ПРОДУКТА

### 1. Введение

Название проекта: Разработка мобильного приложения «ComfortTravel», предназначенного для планирования персонализированных маршрутов путешествий

### 2. Основные задачи

При сопровождении приложения необходимо:

- Поддерживать производительность приложения
- Проводить регулярное улучшение функциональности на основании обратной связи пользователей и продакт-аналитика
- Мониторить данные для анализа на предмет ошибок
- Поддерживать высокий уровень безопасности данных
- Предотвращение сбоев и исправление в случае возникновения ошибок
- Обеспечивать службу поддержки пользователям

### 3. Мониторинг системы

Для поддержки высокой производительности системы необходимо определить основные метрики и инструменты. В качестве метрик будут рассматриваться:

- Частота возникновения ошибок
- Количество уникальных пользователей, которые зашли в приложение за определённый период: дневная (DAU), недельная (WAU) и месячная (MAU) активности
- Процент потребления ресурсов сервера в период выполнения определённых задач
- Время отклика
- Ошибки сервиса и перезагрузки
- Индикаторы масштабирования

Основными инструментами являются системы для отслеживания утечек данных или взлома (к примеру, SIEM (Security Information and Event Management) — системы, которые отслеживают и анализируют события в режиме реального времени), системы отслеживания производительности (APM) и системы мониторинга логирования (к примеру, SigNoz).

## 4. Обновление и улучшение ПО

Так как наша модель жизненного цикла инкрементная, то по мере развития приложения будут проходить регулярные обновления для добавления нового функционала. За плановые обновления отвечает продакт-менеджер, а за критические (то есть те, которые возникают в случае сбоев, для устранения ошибок, улучшения производительности и безопасности). Процесс выпуска обновления совпадает с этапами реализации нового инкремента, которые подробно описаны в документе «План реализации проекта».

Плановые обновления для улучшения ПО должны быть основаны на работе команды по продвижению продукта, а также на анализе отзывов пользователей, которые могут быть собраны через службу поддержки, интервью или опросы.

## 5. Обеспечение уровня безопасности

### 5.1. Обеспечение защиты данных

Приложение использует передовые технологии для защиты персональных данных пользователей, обеспечивая их безопасность как на этапе передачи данных, так и при хранении.

Шифрование данных при передаче и хранении:

- Все данные, которые передаются между клиентом и сервером должны шифроваться с использованием современных протоколов, которые способны предотвратить перехват данных в процессе их передачи.

Регулярное резервное копирование данных:

- Приложение реализует регулярное резервное копирование всей информации, включая персональные данные пользователей, маршруты, заметки.

- Для снижения риска потери данных в результате аварий или сбоев, резервные копии хранятся в облачных провайдеров. Это позволяет быстро восстановить данные в случае аварийной ситуации.

- Резервное копирование должно выполняться ежедневно, а данные копии хранятся в течение длительного периода для возможности восстановления данных на любой момент времени.

Аутентификация и авторизация с использованием безопасных методов:

- Для входа в приложение используется популярный стандарт OAuth для предоставление безопасного доступа к приложениям. Это поможет минимизировать риски, связанные с кражей учетных данных.

- Так же необходимо поддерживать двухфакторную аутентификацию, которая добавит еще один уровень защиты, который будет требовать ввода кода, отправленного на личный телефон пользователя.

- Авторизация пользователей должна происходит с использованием политики контроля доступа, чтобы предотвращать несанкционированный доступ к данным.

## 5. 2. Уязвимость и защита

Для обеспечения надежной защиты приложения от внешних и внутренних угроз проводится постоянный мониторинг и анализ безопасности.

Регулярное сканирование на уязвимости:

- Приложение должно периодически сканироваться на предмет уязвимостей с помощью специализированных инструментов и устранять их до того, как они могу быть активированы.

- Системы мониторинга анализируют активность приложения на наличие подозрительный действий, которые могут нести потенциальную угрозу

Анализ кода и тестирование на проникновение:

- Необходимо регулярно проводить анализ исходного кода с целью выявления уязвимостей, связанный с неправильной обработкой данных, утечкой памяти и другими критическими проблемами безопасности.

- В рамках повышения уровня защиты приложение должно проводить тестирование на проникновение, имитирующее реальные атаки с целью нахождения и устранения уязвимостей в системе.

Оперативные исправления и критические обновления:

- Во время эксплуатации приложения могут выявляться уязвимости, которые команда немедленно должна исправить и выпустить критическое обновление.

- При нахождение глобальной угрозы или новых видов атак обновления безопасности должны выпускаться с минимальной задержкой, а также направлять оповещения пользователям об необходимости установки обновлений и их защиты.

**5.3. План реагирования на инциденты** — это структурированный подход к управлению и реагированию на события, которые могут ставить под угрозу данные пользователей и производительность приложения. Этот план может включать следующие этапы:

Мониторинг и обнаружение инцидентов:

- Внедрение систем мониторинга, которые осуществляют постоянное отслеживание всех действий, происходящие в приложение, включая аномальную активность, подозрительные попытки входа и изменения данных

- В случае обнаружения возможного инцидента система уведомляет команду безопасности и активирует процесс реагирования.

Классификация и оценка инцидента:

- Все найденные инциденты классифицируются по степени важности и критичности в зависимости от уровня угрозы для безопасности данных и работоспособности приложения.

Оповещение и исправление инцидента:

- В случае выявления инцидента, несущего угрозы безопасности пользователям и нарушение работы приложения, принимаются меры по изоляции угрозы, подразумевающие блокировку доступа к уязвимым компонентам системы.

- Необходимо также оповестить пользователей о возможных рисках в случаях, если инцидент затрагивает их персональные данные в соответствие с требованиями GDPR.

Устранение и восстановление:

- Команда реагирования на инциденты должны незамедлительно заниматься устранением причины, то есть устранение уязвимости, изменение политики безопасности.

- После устранения инцидента проводится полное восстановление нормальной работы системы и восстановления всех испорченных данных.

Отчетность и дальнейшее обучение:

- После устранения найденного инцидента создается полный отчет с описанием причины, действиям по устранению и выводами, которые смогут предотвратить подобных инцидентов в будущем.

- Команда безопасности регулярно обучается обновленным методам защиты и реагирования на инциденты, чтобы быть подготовленными к подобному типу угроз.

### 5. 4. Управление рисками

Управление рисками включает постоянный процесс оценивания и минимизация рисков, связанных с безопасностью данных и надежностью приложения.

Оценка потенциальных рисков

- Регулярное проведение анализа системы на предмет возможных технических и организационных рисков.

Разработка стратегий по минимизации рисков:

- Для каждого найденного риска разрабатываются стратегии его минимизации. Это может включать технические меры, операционные и организационные.

Периодический пересмотр мер безопасности:

- Периодически проводится аудит всех мер безопасности, чтобы быть уверенными в том, что меры соответствуют последним требованиям и угрозам.

- Обновление инструментов безопасности и внедрение новых технологий по мере появления новых угроз и рекомендаций.

## 6. Служба поддержки

Для обеспечения качественного обслуживания пользователей приложения **"ComfortTravel"**, организована многоуровневая система поддержки, которая включает в себя следующие этапы и методы взаимодействия.

Уровни поддержки:

1-ый уровень(базовая поддержка):

Основные задачи: ответы на простые вопросы, помощь пользователей с регистрацией и авторизацией, восстановлением паролей и навигацией по приложению

Кто обрабатывает: сотрудники службы поддержки, прошедшие базовое обучение по функционалу приложения.

Примеры вопросов: «Как создать маршрут». «Как восстановить пароль». «Как изменить настройки профиля». «Как создать заметку».

2-ой уровень (техническая поддержка):

Основные задачи: Решение проблем, связанных с функциональностью приложения, включая сбои в работе приложения и в отображение маршрутов, проблемы с загрузкой данных.

Кто обрабатывает: Технические специалисты, имеющие доступ к диагностическим инструментам, и имеют возможность устранить функциональные ошибки.

Пример проблем: Приложение не сохраняет изменения в маршруте, проблема с загрузкой данных и описания о городах и мест.

3-ий уровень (разработчики):

Основные задачи: Решение более сложных технических проблем, подразумевающих ошибки в коде, сбои в работе серверов или проблемы с интеграцией сторонних серверов.

Кто обрабатывает: Команда разработчиков приложения, имеющая доступ к исходному коду способные вносить исправления в него или архитектуру приложения.

Примеры проблем: Приложение некорректно отображает созданные маршруты, внешние серверы возвращают неверные данные, сбои в работе баз данных.

Способы поддержки:

1.Электронная почта:

Подходит для вопросов, не требующих мгновенного ответ. Пользователь с помощью почты могут отправлять запросы в службу поддержки, и получить ответ в интервале от одного до двое суток.

2. Внутренний чат поддержки:

Самый быстрый и оперативный способ взаимодействия со службой поддержкой. Пользователи могут задавать вопросы и получать быстрые ответы в реальном времени прямо в приложение.

3. Телефонная линия:

Используется для решения более сложны вопросов или критических проблем, которые требуют незамедлительного вмешательства. Также подходит для пользователей, которые предпочитают удобное и устное общение.

Процесс обработки заявок

Для эффективного решения проблем пользователей должен внедрен четко структурированный процесс обработки заявок, позволяющих контролировать ход решения и обеспечивает своевременное информирование пользователя.

Этапы обработки заявки:

1.**Обращение пользователя**. Пользователь сталкивается с проблемой или вопросом, в ходе чего ему необходимо обратиться в службу поддержки через один из доступных каналов.

2. **Фиксация заявки в системе тикетов.** После полученного запроса, заявка автоматически регистрируется в системе управления заявками. Что позволит отслеживать каждый запрос, назначая его на соответствующих специалистов и контролируя статус выполнения.

3. **Классификация и передача на соответствующий уровень поддержки.** Заявка анализируется и классифицируется по сложности.

4. **Решение проблемы и обратной связи с пользователем.** Проблема решается соответствующей командой, и пользователь получает уведомление о решение или статусе выполнения. А также направляются подробные инструкции предоставляются пользователю, в случае, если требуется его участие.

5. **Заключительное тестирование.** В случае, если проблема требовала вмешательства разработчиков, то проводится дополнительное, заключительное тестирование, чтобы окончательно убедиться в том, что проблема полностью устранена и не приведет к новым сбоям.

Система поддержки пользователей в **"ComfortTravel"** создана для того, чтобы быстро и эффективно решать любые проблемы, с которыми сталкиваются пользователи, начиная от простых вопросов и заканчивая сложными техническими сбоями. Многоуровневая структура поддержки, вместе с четким процессом обработки заявок, позволяет поддерживать высокое качество обслуживания и оперативно реагировать на запросы пользователей.

## 7. Расписание и план сопровождения

| Этап разработки | Ответственные лица | Срок выполнения |
|---|---|---|
| Поддержка пользователей | Специалисты службы поддержки | Постоянно |
| Мониторинг производительности | Системный администратор, DevOps | Ежедневно |
| Резервное копирование данных | DevOps-инженеры | Автоматически(ежедневно) |
| Обновление приложения | Разработчики, QA-инженеры | Один раз в 2-3 месяца |

| | | |
|---|---|---|
| Выпуск критических исправлений | Разработчики, специалисты по безопасности | В случае необходимости |
| Анализ и тестирование обновлений | Тестировщики, QA-инженеры | Перед каждым обновлением |
| Масштабирование инфраструктуры | DevOps-инженеры, системные администраторы | Ежеквартально |
| Обновление инфраструктуры | Системный администратор | Один раз в год |
| Аудит безопасности и защиты данных | Специалисты по безопасности | Ежеквартально |
| Тестирование на уязвимость | Специалисты по безопасности | Один раз в квартал |
| Анализ производительности и оптимизации | Техническая команда | Ежеквартально |
| Управление инцидентами | Техническая команда, служба поддержки | В случае необходимости |
| Обучение команды | Руководители команд, специалисты по обучению | Ежеквартально |
| Сбор и анализ обратной связи | Служба поддержки, аналитики | Постоянно |
| Документирование и создание обучающих материалов | Команда документации и маркетинга | По мере выпуска каждого крупного обновления |

# ЗАКЛЮЧЕНИЕ

В ходе работы над проектом по созданию приложения «ComfortTravel» наша команда достигла поставленных целей, выполнив все необходимые этапы для проектирования информационной системы. Основная задача заключалась в создании Технического Задания и разработке артефактов для дальнейшей передачи документации команде разработчиков для реализации приложения по планированию путешествий, которое бы позволило

пользователям составлять персонализированные маршруты, получать актуальную информацию о достопримечательностях, а также управлять заметками и историей поездок.

По итогам нашей работы мы выделили основные требования, разработали необходимы артефакты, через которые команда разработчиков сможет понять концепцию системы и каркас будущего ПО. Кроме того, мы создали четкий план работы, который позволил не только нам, но и другим командам организованно подойти к реализации проекта.

Для более детальной разработки были проработаны важнейшие артефакты, такие как диаграмма классов, диаграмма последовательности и диаграмма состояний. Эти диаграммы дали нам возможность ясно представить структуру системы, потоки данных и взаимодействие между элементами. Также была создана матрица трассировки, которая обеспечила отслеживаемость всех требований и их реализацию в ходе разработки. Эти артефакты являются основой для дальнейшего программирования и тестирования системы, помогая избежать логических ошибок и улучшая понимание структуры проекта.

Последним этапом было создание основной документации по реализации, тестированию, эксплуатации и сопровождению, благодаря которым на следующих этапах жизненного цикла проекта описанные задачи, метрики, цели и структуры позволят завершить разработку без ошибок и задержек. Немаловажной частью нашей работы стала ревизия документации, которая позволила выявить и исправить ошибки на ранних этапах разработки. Этот шаг обеспечил соответствие всех элементов системы, позволив нам успешно завершить разработку проекта в рамках сроков.

Полученный документ будет направлен команде разработки, тестирования и поддержки для дальней работы над проектом.