**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«УНИВЕРСИТЕТ УПРАВЛЕНИЯ «ТИСБИ»**

Факультет информационных технологий

Кафедра информационных технологий

Допущена к защите

зав. кафедрой информационных технологий

канд. пед. наук, доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.В. Федорова

«\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

на тему: «Разработка Web-сервиса для путешествий»

Направление подготовки: 09.03.03 Прикладная информатика

Профиль подготовки: Прикладная информатика в экономике

Работу выполнил

cтудент гр. ПИ-112 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.М.Калкаманова

Научный руководитель

канд. пед. наук, доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.В. Федорова

Консультант

Заместитель руководителя отдела

в Отделе организации программ

подготовки кадров высшей

квалификации Центра организации

образовательных программ

АНО ВО «Университет Иннополис» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Р.Р. Ибатуллин

Казань, 2025 г.

Оглавление

[**Введение** 3](#_Toc196406095)

[**Глава 1. Постановка задачи** 5](#_Toc196406096)

[**1.1 Цель создания информационной системы. Характеристики системы** 5](#_Toc196406097)

[**1.2** **Анализ информационных систем** 7](#_Toc196406098)

[**1.3** **Анализ и выбор средств проектирования и разработки ИС** 12](#_Toc196406099)

[**Глава 2. Проектирование информационной системы** 14](#_Toc196406100)

[**2.1. Требования к структуре и функциям (задачам), выполняемым системой** 14](#_Toc196406101)

[**2.2. Требования к видам обеспечения** 19](#_Toc196406102)

[**2.2.1. Требования к надежности системы** 19](#_Toc196406103)

[**2.2.2. Требования к безопасности системы и к обеспечению защиты информации** 21](#_Toc196406104)

[**2.2.3. Требования к эргономике системы** 23](#_Toc196406105)

[**2.2.4. Требования к эксплуатации системы** 26](#_Toc196406106)

[**2.2.5. Требования к патентной чистоте** 29](#_Toc196406107)

[**2.2.6. Требования по стандартизации** 32](#_Toc196406108)

[**2.2.7. Требования к лингвистическому обеспечению** 35](#_Toc196406109)

[**2.3. Экономический расчет** 39](#_Toc196406110)

[**2.3.1. Оценка трудоемкости разработки ИС** 42](#_Toc196406111)

[**2.3.2. Оценка рисков при создании ИС** 46](#_Toc196406112)

[**Глава 3. Разработка структуры программного обеспечения информационной системы** 48](#_Toc196406113)

[**3.1. Средства реализации** 48](#_Toc196406114)

[**3.2. Общая структура программного приложения системы** 51](#_Toc196406115)

[**3.3. Руководство пользователя** 59](#_Toc196406116)

[**3.3.1. Инсталляция информационной системы в соответствии с используемой конфигурацией технического и программного обеспечения системы.** 59](#_Toc196406117)

[**3.3.2. Результаты тестирования с примерами программного интерфейса.** 62](#_Toc196406118)

[**3.4. Внедрение информационной системы** 71](#_Toc196406119)

[**3.4.1. Содержание работ по внедрению ИС** 71](#_Toc196406120)

[**3.4.2. Порядок приемки работ и ввода в промышленную эксплуатацию** 73](#_Toc196406121)

[**3.5. Оценка рыночной стоимости разработанного программного продукта** 75](#_Toc196406122)

[**Приложение 1** 78](#_Toc196406123)

# **Введение**

В современном мире путешествия становятся неотъемлемой частью жизни многих людей. Однако организация поездки, особенно групповой, сопряжена с рядом проблем: сложность маршрутов, поиск надежных попутчиков, бронирование билетов и жилья, недостаток проверенной информации и реальных отзывов. В условиях развития цифровых технологий актуальным становится создание специализированных платформ, способных упростить эти процессы.

Выбранная тема выпускной квалификационной работы посвящена разработке инновационной информационной системы – веб-сервиса Travel Buddy, предназначенного для поиска попутчиков, бронирования билетов и отелей, а также общения между пользователями. Актуальность проекта обусловлена растущим спросом на удобные и надежные решения для самостоятельных и групповых путешествий, а также недостатком комплексных сервисов, объединяющих в себе функции социальной платформы и системы бронирования.

Практическая значимость Travel Buddy заключается в упрощении организации поездок, экономии времени и средств пользователей, а также повышении качества туристического опыта за счёт проверенной информации и встроенной коммуникации. В процессе разработки были рассмотрены и решены задачи обеспечения надежности, скорости, масштабируемости и интеграции с внешними сервисами (Aviasales, Яндекс.Транспорт, Booking и др.).

Целью ВКР является создание веб-приложения, объединяющего различные туристические сервисы в единую платформу для совместных путешествий.

Задачи работы:

* провести анализ аналогичных информационных систем;
* спроектировать архитектуру и функциональность веб-сервиса;
* реализовать систему на основе современных технологий (ASP.NET, PostgreSQL, Blazor);
* обеспечить интеграцию с внешними API;
* протестировать и внедрить систему;
* проанализировать экономическую эффективность проекта.

**Глава 1. Постановка задачи**

**1.1 Цель создания информационной системы. Характеристики системы**

Разработка веб-сервиса Travel Buddy направлена на решение актуальных задач, стоящих перед современными путешественниками. В условиях стремительного роста интереса к самостоятельным поездкам, особенно среди молодежи и людей среднего возраста, возникает необходимость в удобной, универсальной и надёжной платформе, которая объединяет в себе функции поиска попутчиков, бронирования билетов и проживания, а также обеспечивает взаимодействие между участниками поездки.

Целью создания информационной системы Travel Buddy является разработка полнофункционального программного продукта, обеспечивающего комплексную поддержку пользователей на всех этапах планирования и организации путешествия. Система должна сократить время на подготовку поездки, повысить её удобство и безопасность, а также способствовать формированию сообщества пользователей с общими интересами и маршрутами.

Travel Buddy предлагает пользователям следующий функционал:

* Поиск попутчиков по маршруту, интересам, возрасту и другим критериям;
* Встроенный мессенджер для общения и координации планов;
* Интеграция с внешними сервисами бронирования билетов (авиа, ЖД, автобусы) и гостиниц (Booking, Aviasales, Яндекс.Транспорт);
* Формирование единого маршрута с точками остановки, отображаемыми на карте;

Характеристики разрабатываемой системы:

* Масштабируемость: архитектура системы позволяет добавлять новые функции, сервисы и модули без необходимости полной переработки проекта.
* Многоуровневая структура: разделение на клиентскую, серверную и базовую части обеспечивает удобство поддержки и развития.
* Интеграция с API: благодаря использованию внешних API (OpenStreetMap, Aviasales и др.) достигается актуальность информации о маршрутах, ценах и объектах размещения.
* Безопасность и конфиденциальность: реализуются механизмы авторизации, защиты личных данных и фильтрации нежелательного контента.

Travel Buddy ориентирован на следующие целевые группы:

* Люди, путешествующие в одиночку, но заинтересованные в совместной поездке;
* Пользователи, стремящиеся к оптимизации бюджета поездки;
* Туристы, предпочитающие нестандартные маршруты и живое общение с единомышленниками;
* Молодые активные пользователи, привыкшие к цифровым сервисам.

Таким образом, создаваемая информационная система должна стать современным инструментом, способствующим цифровой трансформации сферы туризма. Внедрение веб-сервиса открывает широкие возможности как для пользователей, так и для бизнес-партнеров, обеспечивая удобство, экономию ресурсов и повышение качества туристического опыта.

**1.2** **Анализ информационных систем**

Прежде чем приступить к разработке собственной информационной системы Travel Buddy, необходимо провести анализ существующих решений, функционирующих на рынке туристических и социальных сервисов. Это позволит определить их сильные и слабые стороны, выявить возможности для улучшения и уточнить уникальные особенности разрабатываемого продукта.

На текущем этапе существует множество цифровых платформ, связанных с организацией путешествий. Эти сервисы можно условно разделить на следующие категории:

Сервисы бронирования (Booking.com, Aviasales, Airbnb, BlaBlaCar);

Транспортные платформы (Яндекс.Транспорт, РЖД, FlixBus, Omio);

Социальные и рекомендательные сервисы (TripAdvisor, Couchsurfing, Туристер.ру);

Навигационные и картографические системы (Яндекс.Карты, Google Maps, OpenStreetMap).

Каждое из этих решений закрывает определённую часть потребностей пользователя, однако нет полностью корректно функционирующей платформы, объединяющей в себе одновременно:

* функциональность поиска попутчиков;
* бронирование всех видов транспорта и проживания;
* встроенную систему общения и отзывов;
* построение маршрутов с отображением точек на карте;
* возможность персонализации под интересы пользователя.

Рассмотрим более подробно ключевые аналоги.

Одним из крупнейших и наиболее известных игроков на рынке бронирования жилья является платформа Booking.com. Сервис предоставляет доступ к тысячам вариантов размещения по всему миру, позволяет фильтровать предложения по множеству параметров, а также содержит систему проверенных отзывов. Однако Booking.com ориентирован исключительно на размещение и частично на бронирование транспорта через партнёрские программы. Пользователю приходится самостоятельно решать вопрос маршрута, транспортных связей и общения с другими путешественниками, что делает использование платформы фрагментарным и не всегда удобным.

Другим ярким примером является агрегатор Aviasales, специализирующийся на поиске и сравнении цен на авиабилеты. Он предоставляет быстрый и удобный способ подобрать оптимальные рейсы от множества авиакомпаний, включая функции уведомлений о снижении цен и гибких дат. Тем не менее, данный сервис охватывает только авиаперевозки и не позволяет спланировать поездку полностью, включая другие виды транспорта, жильё или общение между участниками поездки. Также пользователи перенаправляются на внешние сайты для бронирования, что снижает удобство и целостность пользовательского опыта.

Сервис BlaBlaCar ориентирован на поиск попутчиков в автомобильных поездках. Он позволяет пассажирам и водителям договариваться о совместных маршрутах, что делает путешествия более экономичными. В системе предусмотрена рейтинговая система, профили пользователей, фильтры поиска по маршрутам и городам. Однако функциональность ограничена только автомобильными поездками. Отсутствует возможность бронирования билетов, составления сложных маршрутов с пересадками, а также поиска жилья или общения внутри платформы за пределами конкретной поездки.

Особого внимания заслуживает проект Couchsurfing, который позиционирует себя как социальная сеть путешественников. Основной идеей платформы является предоставление бесплатного жилья и культурного обмена между людьми. Couchsurfing обладает развитым сообществом, системой профилей, отзывов и мероприятий, организуемых участниками. Однако в нём полностью отсутствуют механизмы бронирования транспорта или составления маршрутов, а взаимодействие ограничивается личной инициативой пользователей, что может не подходить для широкого круга туристов.

Платформа TripAdvisor представляет собой один из крупнейших рекомендательных сервисов, содержащий миллионы отзывов о достопримечательностях, ресторанах, отелях и туристических маршрутах. Несмотря на то, что TripAdvisor предоставляет обширную информацию и ориентирован на поддержку путешественников, он практически не имеет функционала для организации совместных поездок или поиска попутчиков. Бронирование осуществляется через сторонние сервисы, а социальное взаимодействие ограничено форумами и комментариями.

Сервис Яндекс.Транспорт, в свою очередь, предоставляет информацию о передвижении общественного транспорта в реальном времени. Он полезен для жителей и гостей российских городов, позволяя планировать маршруты внутри города. Однако сервис не рассчитан на туристические поездки, не поддерживает бронирование билетов, не включает данные по междугородним и международным маршрутам, а также не содержит компонентов социального взаимодействия между пользователями.

Наконец, OpenStreetMap — это открытая картографическая платформа, позволяющая разрабатывать на её основе различные геоинформационные сервисы. Она предоставляет обширные данные для построения маршрутов и отображения объектов инфраструктуры. Тем не менее, это не самостоятельный пользовательский сервис, а скорее технологическая основа для других решений. В ней отсутствует интерфейс для бронирования, взаимодействия или социальной активности, что делает её вспомогательным инструментом.

На основе выделенных категорий составим сравнительные таблицы.

Таблица 1.2.1. Поиск попутчиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сервис | Поддержка попутчиков | Тип поездки | Система рейтингов/отзывов | Комментарий |
| BlaBlaCar | Да | Авто | Да | Только авто, нет маршрутов и жилья |
| Couchsurfing | Частично | Гостевые визиты | Да | Социальная сеть, без бронирований |
| TripAdvisor | Нет | - | Частично | Отзывы о местах, нет попутчиков |
| Travel Buddy | Да | Все виды | В перспективе | Встроенный мессенджер, фильтры |

Таблица 1.2.2. Бронирование транспорта и жилья

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сервис | Бронирование транспорта | Бронирование жилья | Интеграция с внешними API | Комментарий |
| BlaBlaCar | Только авто | Нет | Нет | Поиск попутчиков, бронирование поездок |
| Couchsurfing | Нет | Бесплатное жильё | Нет | Основано на личных договорённостях |
| TripAdvisor | Через партнёров | Через партнёров | Частично | Сам не бронирует, только перенаправляет |
| Travel Buddy | Только просмотр вариантов | Только просмотр вариантов | Да | Решение с API-подключением без возможности бронирования, только просмотра допустимых вариантов |

Таблица 1.2.3. Социальные функции и взаимодействие

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сервис | Мессенджер | Профили | Отзывы и рейтинги | События/ мероприятия |
| BlaBlaCar | Ограничено | Да | Да | Нет |
| Couchsurfing | Частично | Да | Да | Да |
| TripAdvisor | Нет | Да | Нет | Отзывы о местах, нет попутчиков |
| Travel Buddy | Да | Да | Частично | Нет |

Таблица 1.2.4. Построение маршрутов и карты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Сервис | Построение маршрутов | Отображение на карте | Многоэтапные поездки | Интерактивность карты |
| BlaBlaCar | Да | Да | Нет | Нет |
| Couchsurfing | Нет | Нет | Нет | Нет |
| TripAdvisor | Частично | Да | Нет | Нет |
| Travel Buddy | Да | Да | Да | Да |

Таким образом, проведённый анализ демонстрирует, что существующие информационные системы обладают узкой специализацией и не охватывают весь спектр задач, с которыми сталкиваются путешественники. Пользователю приходится использовать одновременно несколько сервисов, что создаёт неудобство, снижает эффективность планирования и увеличивает время на организацию поездки. Проект Travel Buddy устраняет эти недостатки, объединяя лучшие функции упомянутых решений в единую, многофункциональную платформу, способную предложить комплексную цифровую поддержку путешествий.

**1.3** **Анализ и выбор средств проектирования и разработки ИС**

Для создания эффективной, масштабируемой и надёжной информационной системы Travel Buddy необходимо обоснованно выбрать программные и технические средства, которые обеспечат стабильную работу сервиса, удобство разработки и поддержки, а также соответствие современным требованиям к безопасности и производительности.

В качестве основной технологии для серверной части был выбран стек C# + ASP.NET Core, поскольку это современная, кроссплатформенная среда для разработки надёжных веб-приложений. Она обеспечивает высокую производительность, удобную архитектуру, поддержку REST API и отличную интеграцию с базами данных. Для работы с данными используется Entity Framework Core, предоставляющий возможности ORM и упрощающий взаимодействие с базой данных.

Веб-интерфейс реализован с использованием Blazor — фреймворка от Microsoft, который позволяет разрабатывать интерактивные веб-приложения на C# вместо JavaScript. Это обеспечивает единство языка на клиентской и серверной части, сокращает количество ошибок и упрощает разработку.

В качестве СУБД выбран Microsoft SQL Server — надёжная объектно-реляционная система управления базами данных, которая поддерживает сложные запросы, транзакции и хорошо масштабируется под высокие нагрузки.

Для построения маршрутов и отображения карт используется OpenStreetMap, который предоставляет открытые и постоянно обновляемые картографические данные. Работа с ними осуществляется через сторонние библиотеки и API, например, Leaflet.

Для размещения серверной части приложения планируется использование облачных решений, таких как Windows Azure, что обеспечивает гибкость масштабирования, стабильную работу и удобные инструменты резервного копирования и обновления.

Минимальные требования к серверу для запуска проекта на старте:

* CPU: 2 ядра;
* RAM: 4 ГБ;
* SSD-диск: 50 ГБ;
* ОС: Ubuntu 22.04 LTS.

С учётом планируемого роста числа пользователей, архитектура проекта предусматривает возможность горизонтального масштабирования и разнесения сервисов по микросервисной модели.

Инструменты разработки и командной работы

В процессе разработки используются следующие инструменты:

* Visual Studio / Rider — для написания и отладки кода;
* Postman — для тестирования REST API;
* Figma — для проектирования интерфейса;
* Git + GitHub — для контроля версий и совместной работы;
* Trello — для управления задачами и документацией.

Выбор вышеуказанных средств обусловлен их стабильностью, поддержкой со стороны сообщества, документированностью и масштабируемостью. Все инструменты и технологии являются современными и активно развиваются, что даёт уверенность в долгосрочной поддержке проекта и простоте внедрения новых функций.

Таким образом, использование ASP.NET, Blazor, MS SQL и внешних API позволяет создать надёжную, гибкую и удобную в эксплуатации информационную систему, полностью соответствующую требованиям проекта.

**Глава 2. Проектирование информационной системы**

**2.1. Требования к структуре и функциям (задачам), выполняемым системой**

Информационная система Travel Buddy разрабатывается как многофункциональная веб-платформа для планирования и организации совместных путешествий. Для полноценной реализации её задач необходимо определить чёткие требования как к архитектуре системы, так и к выполняемым функциям.

Система основана на трёхуровневой архитектуре, включающей:

Клиентский уровень (Frontend) — реализован с использованием Blazor WebAssembly. Обеспечивает взаимодействие пользователя с системой через удобный и адаптивный веб-интерфейс. Все элементы управления и визуализация маршрутов, списков попутчиков, бронирований, профилей и чатов формируются на этом уровне.

Серверный уровень (Backend) — построен на ASP.NET Core и выполняет функции обработки бизнес-логики, обработки пользовательских запросов, авторизации, маршрутизации данных, обращения к API внешних сервисов и взаимодействия с базой данных. Это ядро всей системы.

Уровень данных (Database) — реализован с помощью Microsoft SQL Server. Обеспечивает хранение информации о пользователях, маршрутах, бронированиях, сообщениях, отзывах и других объектах, необходимых для функционирования платформы.

**Архитектура базы данных**

Система Travel Buddy использует реляционную базу данных, построенную с учётом требований безопасности, масштабируемости и расширяемости. Основу базы составляют таблицы:

* Users — хранит учетные записи пользователей (включая email, имя, фото, паспортные данные, дату рождения, город);
* Messages — сообщения между пользователями;
* UserRoute и RouteStops — маршруты и остановки;
* Roles, UserRoles — роли и права пользователей;

Функциональные требования

1. Регистрация и авторизация пользователей

* Возможность регистрации нового пользователя с вводом email, имени, пароля и дополнительных данных. Регистрация осуществляется с сохранением данных в Users.
* Поддержка email-подтверждения (EmailConfirmed) по OAuth 2.0.
* Возможность восстановления пароля.
* Защита от несанкционированного доступа с помощью механизма шифрования паролей.
* Роли пользователя (модератор, администратор) управляются через Roles и UserRoles.
* Авторизация поддерживает сторонние провайдеры через UserLogins.

2. Профили пользователей

* Панель управления личными данными, фотографией, интересами, языками, предпочтениями. В таблице Users хранятся: FullName, ProfilePictureUrl, City, BirthDate, паспортные данные.
* Возможность загрузки фото, редактирования данных, отображения публичного профиля.
* Возможность смены старого пароля на новый.

3. Коммуникация между пользователями

* Встроенный мессенджер с возможностью отправки текстовых сообщений, ссылок, медиафайлов, просмотр истории сообщений, уведомления о новых сообщениях.
* Таблица Messages реализует хранение переписки. Поля: SenderId, RecipientId, Content, SentAt.
* Поддержка email-рассылки.

4. Построение маршрутов

* Добавление и редактирование маршрута поездки: город отправления, промежуточные точки, финальный пункт.
* Поддерживаются: многосегментные маршруты, различные типы транспорта (TransportationType), отображение времени и стоимости.
* Поддержка сложных маршрутов с несколькими пересадками и типами транспорта.
* Сохранение маршрутов с привязкой к профилю пользователя.
* Таблица UserRoute связывает пользователя с определённым маршрутом. Таблица RouteStops хранит точки маршрута с координатами (широта/долгота), отелями, транспортом, датами.
* Реализуется на клиенте с использованием картографических сервисов (Leaflet + OpenStreetMap).

5. Планирование поездок

* Интеграция с внешними API: Aviasales (авиабилеты), Яндекс.Транспорт (городской транспорт), Booking (отели).
* Просмотр расписаний, цен, доступных мест.
* Возможность сохранения выбранных вариантов.
* Хранение истории маршрутов с возможностью отмены или изменения.
* Таблица RouteStops содержит данные о гостиницах (HotelName, HotelPrice, HotelCheckInDate, HotelCheckOutDate) и транспорте (TransportationPrice, координаты отправления/прибытия, время и тип транспорта).
* В дальнейшем возможно расширение до полноценной системы бронирования с подтверждением.

6. Интеллектуальный модуль анализа города

* В рамках функционала построения маршрута реализована интеграция с нейросетевым модулем, который автоматически обрабатывает название города, введённое пользователем, и возвращает краткую справочную информацию о нём (география, достопримечательности, культурные особенности).
* Автоматическая загрузка изображений города с использованием OpenAPI.
* Изображения городов сохраняются временно в кеше и отображаются в интерфейсе при просмотре маршрута.

7. Поиск и подбор попутчиков

* Умный поиск маршрутов по фильтру, в схожие даты и по схожим направлениям.
* Отображение совпадений по маршрутам и предпочтениям.
* Поиск реализован через frontend

Таким образом, структура и функциональность информационной системы Travel Buddy формируют целостную, логически выверенную платформу, способную обеспечить комплексную цифровую поддержку пользователей на всех этапах путешествия — от планирования маршрута и подбора попутчиков до бронирования и общения.

Архитектура системы демонстрирует современный подход к проектированию веб-приложений: многоуровневое разделение компонентов, реляционная модель данных с чёткими связями между сущностями, интеграция с внешними API и реализация функциональности через надежный стек технологий (ASP.NET Core, Blazor, SQL Server, OpenStreetMap). Такая организация обеспечивает гибкость, отказоустойчивость и масштабируемость проекта, позволяя легко добавлять новые модули и адаптироваться под потребности целевой аудитории.

Travel Buddy ориентирован на долгосрочное развитие и может быть масштабирован как в рамках веб-среды, так и с переходом к нативным мобильным приложениям. Вся архитектура выстроена так, чтобы платформа могла не только удовлетворить текущие требования пользователей, но и уверенно развиваться в будущем, оставаясь конкурентоспособной на цифровом рынке туристических решений.

**2.2. Требования к видам обеспечения**

**2.2.1. Требования к надежности системы**

Надежность системы Travel Buddy является критически важным аспектом, так как от неё зависит доверие пользователей и бесперебойность работы сервиса. Требования включают:

1. Отказоустойчивость

Кластерная архитектура:

* Серверная часть развертывается в виде кластера с минимум двумя нодами для обеспечения отказоустойчивости.
* База данных использует репликацию (Master-Slave) для предотвращения потери данных при сбоях.

Балансировка нагрузки:

* Использование Nginx или Azure Load Balancer для распределения запросов между серверами.
* Автоматическое переключение на резервный сервер при отказе основного.

2. Резервное копирование

Ежедневные бэкапы:

* Полное резервное копирование базы данных каждые 24 часа.
* Инкрементальные бэкапы каждые 6 часов для минимизации потерь данных.
* Хранение бэкапов в географически распределенных дата-центрах (например, Azure Blob Storage + локальная копия).

Восстановление:

* Возможность восстановления системы на любую точку времени в течение последних 30 дней.
* Автоматизированные скрипты для развертывания резервных копий.

3. Мониторинг и логирование

Инструменты:

* Prometheus для сбора метрик (CPU, RAM, дисковое пространство, время отклика API).
* Grafana для визуализации данных и настройки алертов.
* ELK-стек (Elasticsearch, Logstash, Kibana) для анализа логов.

Критические метрики:

* Время отклика API не должно превышать 500 мс.
* Загрузка CPU сервера — не более 70% в пиковые часы.
* Доступность базы данных — 99.99%.

4. Тестирование

Юнит-тесты:

* Покрытие кода не менее 80% (используется xUnit/NUnit).

Интеграционные тесты:

* Проверка взаимодействия между модулями (например, авторизация + бронирование).

Нагрузочное тестирование:

* Использование JMeter для имитации 10 000 одновременных пользователей.
* Проверка устойчивости системы при DDoS-атаках.

5. Целевые показатели надежности

* Uptime: Не менее 99.5% в месяц (максимально допустимый простой — 3.6 часа/месяц).
* MTTR (Mean Time To Repair): Не более 30 минут для критических сбоев.
* RPO (Recovery Point Objective): Максимальная потеря данных — 1 час.

**2.2.2. Требования к безопасности системы и к обеспечению защиты информации**

Безопасность Travel Buddy включает защиту данных пользователей, предотвращение атак и соответствие законодательным нормам.

1. Шифрование данных

* HTTPS: Обязательное использование TLS 1.3 для всех соединений.
* Хранение паролей: Алгоритм bcrypt с "солью" (стоимость хеширования — 12 раундов).
* Шифрование БД: Поля с персональными данными (паспорт, email) шифруются с помощью AES-256.

2. Аутентификация и авторизация

* Многофакторная аутентификация (MFA):
* Поддержка SMS, Google Authenticator, email-кодов.
* OAuth 2.0:
* Интеграция с Google, Facebook, VK.
* Ролевая модель (RBAC):
* Роли: Гость, Пользователь, Модератор, Администратор.
* Права:
* Гость: просмотр маршрутов.
* Пользователь: бронирование, чат.
* Модератор: блокировка пользователей.
* Администратор: доступ к панели управления.

3. Защита от атак

* SQL-инъекции: Параметризованные запросы + ORM (Entity Framework Core).
* XSS: Санитизация ввода (HTML encode), CSP (Content Security Policy).
* CSRF: Токены в формах + SameSite cookies.
* Brute-force: Капча после 3 неудачных попыток входа.

4. Соответствие стандартам

* ФЗ-152 (РФ): Хранение персональных данных на серверах в России.
* GDPR (ЕС):
* Право на удаление данных (функция "Забыть меня").
* Экспорт данных пользователя в JSON.

5. Физическая безопасность

* ЦОД: Размещение серверов в дата-центрах уровня Tier III+ (например, Selectel, Azure).
* Доступ: Биометрия + двухфакторная аутентификация для персонала.

**2.2.3. Требования к эргономике системы**

Эргономика системы Travel Buddy направлена на обеспечение удобства, эффективности и комфорта пользователей при взаимодействии с платформой. Детальные требования включают:

1. Интуитивный интерфейс

* Логичная структура навигации:
* Главное меню должно содержать основные разделы: "Главная", "Маршруты", "О нас", "Сообщения", "Люди" – слева, "Вход" и "Регистрация", если пользователь не аутентифицирован и "Профиль" – если аутентифицирован – справа.
* Минимизация количества кликов для выполнения ключевых действий (не более 3 шагов).
* Единообразие элементов:
* Кнопки, формы, поля ввода и другие элементы должны быть стилистически согласованы.
* Использование стандартных иконок для унификации восприятия.
* Подсказки и инструкции:
* Встроенные тултипы (подсказки при наведении) для сложных элементов.
* Анимация загрузки (лоадеры) для операций, требующих времени.

2. Визуальная иерархия

* Заголовок Travel Buddy крупный и легко читается.
* Подзаголовок расположен справа на фоне изображения, читаемый белым шрифтом с хорошим контрастом.
* Информационные блоки разделены визуально, что снижает нагрузку на восприятие.

3. Адаптивный дизайн

* Поддержка устройств:
* Корректное отображение на экранах от 320px (мобильные устройства) до 1920px (десктоп).
* Адаптация сетки макета (CSS Grid/Flexbox) для перестроения элементов при изменении разрешения.
* Мобильная версия:
* Упрощённое меню (гамбургер-меню для мобильных устройств).
* Крупные кликабельные элементы (кнопки, ссылки) для удобства касания.
* Оптимизация изображений и контента для быстрой загрузки на мобильных сетях.

4. Цветовая схема и темы

* Тёмная тема снижает нагрузку на глаза при длительной работе.
* Белый шрифт по затемнённому фону имеет высокую контрастность (по стандарту WCAG).
* Цвет активных элементов может быть адаптирован под фирменную палитру

5. Учет принципов UX/UI

* Принцип близости:
* Связанные элементы (например, кнопка "Поиск" и поле ввода) группируются.
* Контрастность:
* Соотношение контраста текста и фона не менее 4.5:1 (соответствие WCAG).
* Обратная связь:
* Визуальный отклик при наведении (hover) и клике (active).
* Сообщения об успешных/ошибочных действиях (тосты, алерты).

6. Доступность (Accessibility)

* Поддержка WCAG 2.1:
* Альтернативные тексты для изображений (alt-атрибуты).
* Навигация с клавиатуры (Tab/Shift+Tab).
* Семантическая вёрстка (теги <header>, <main>, <footer>).
* Шрифты:
* Использование легкочитаемых шрифтов (Roboto, Open Sans).
* Размер шрифта не менее 16px для основного текста.

**2.2.4. Требования к эксплуатации системы**

Эксплуатация информационной системы Travel Buddy должна обеспечивать удобство, устойчивость и эффективность как для конечных пользователей (туристов), так и для персонала, сопровождающего и администрирующего систему. Особое внимание уделяется доступности, простоте обслуживания, возможности масштабирования, автоматизации обновлений и поддержке пользователей.

1. Требования к пользовательской эксплуатации

Платформа должна быть ориентирована на конечного пользователя — путешественника, не обладающего техническими знаниями, что предполагает:

* Доступность 24/7: веб-приложение должно быть доступно круглосуточно с любого устройства, подключенного к интернету. Важно обеспечить высокий уровень аптайма — не менее 99,5% в месяц.
* Работа через браузер: отсутствует необходимость в установке дополнительного ПО. Поддерживаются все современные браузеры (Chrome, Edge, Firefox, Safari).
* Простота обновлений: обновления интерфейса и функционала не должны прерывать работу пользователя. Применяются методики hot reload и rolling deployments, позволяющие внедрять изменения незаметно.
* Мультиязычный интерфейс: пользователи должны иметь возможность выбрать язык интерфейса (русский, английский и др. — по мере развития).
* Обратная связь: реализуется встроенный чат поддержки или форма обратной связи. Для быстрой коммуникации может быть подключён Telegram-бот или сторонние решения (Crisp, Tawk.to).
* Информационная поддержка: наличие раздела FAQ, подсказок в интерфейсе, обучающих видеороликов и статей. Это снижает нагрузку на службу поддержки.
* Надёжность данных пользователя: система должна сохранять изменения в реальном времени, предупреждать об ошибках и обеспечивать восстановление действий в случае неполадок.

2. Требования к сопровождению и администрированию

Административная часть платформы должна позволять обслуживать систему без простоев и с минимальными затратами:

* Панель администратора: веб-интерфейс для управления:
* пользователями (блокировка, восстановление, смена ролей);
* маршрутами и содержанием;
* жалобами и модерацией сообщений;
* доступом к внутренней статистике.
* Логирование:
* хранение истории авторизаций, ошибок, запросов к API;
* интеграция с системами сбора логов (например, ELK-стек или Azure Monitor).
* Мониторинг системы:
* отслеживание состояния серверов, API, базы данных;
* алерты при падении производительности, задержках в ответах или сбоях.
* CI/CD (непрерывная интеграция и развёртывание):
* автоматическая сборка и тестирование кода при коммите;
* развёртывание новой версии в staging- и production-среду с минимальными рисками;
* инструменты: GitHub Actions, GitLab CI, Azure DevOps.

3. Требования к масштабируемости и резервированию

Система должна быть готова к росту числа пользователей и запросов:

* Горизонтальное масштабирование: использование Docker и Kubernetes позволяет запускать дополнительные копии сервисов при повышенной нагрузке.
* База данных:
* поддержка кластеризации;
* репликация и бэкапы в режиме online;
* защита от потери данных.
* Облачные ресурсы:
* размещение в Azure или аналоге с динамическим масштабированием;
* изоляция сред разработки, тестирования и продакшена.

4. Техническая поддержка

* Многоуровневая поддержка:
* Уровень 1 — ответы на часто задаваемые вопросы (бот, база знаний);
* Уровень 2 — технический специалист через чат или email;
* Уровень 3 — администратор/разработчик при критических ошибках.
* Регламент времени отклика:
* на инциденты — не более 15 минут (в рабочее время);
* на запросы пользователей — не более 24 часов.

5. Документирование и обучение персонала

* Подготовка пользовательской и технической документации.
* Обучающие материалы для новых сотрудников поддержки.
* Сценарии и инструкции для типичных ситуаций (например, обработка жалоб, восстановление доступа, откат обновлений).

**2.2.5. Требования к патентной чистоте**

Разработка информационной системы Travel Buddy должна строго соответствовать законодательным нормам в области интеллектуальной собственности, авторских прав и лицензионных соглашений. Это исключит юридические риски, связанные с использованием сторонних технологий, и обеспечит легальность всех компонентов системы. Требования включают:

1. Использование открытых и лицензионных технологий

Фреймворки и библиотеки:

* Применяются только технологии с разрешительными лицензиями (MIT, Apache 2.0, GPL). Например:
* Blazor (лицензия Apache 2.0) — для клиентской части.
* Entity Framework Core (лицензия MIT) — для работы с базой данных.
* Leaflet (лицензия BSD) — для картографического функционала.
* Запрещается использование пиратского или нелицензионного ПО.

Сторонние API:

* Интеграция с API Aviasales, Яндекс.Транспорт, Booking.com осуществляется только через официальные партнёрские программы или публичные API с соблюдением условий использования.
* Для каждого API оформляется соответствующее соглашение (если требуется платный доступ или коммерческое использование).

2. Оригинальность дизайна и контента

Логотип и фирменный стиль:

* Разрабатываются с нуля или приобретаются у дизайнеров с передачей исключительных прав.
* Проверка на уникальность через сервисы типа TinEye или Яндекс.Картинки.

Иконки и изображения:

* Используются только свободные ресурсы (например, FontAwesome, Unsplash, Flaticon) с указанием авторства (если требуется лицензией).
* Запрещается использование изображений из поисковых систем без подтверждённых прав.

Шрифты:

* Применяются шрифты с открытой лицензией (Google Fonts, Adobe Fonts).

3. Патентная чистота алгоритмов и функционала

Уникальность решений:

* Ключевые алгоритмы (например, поиск попутчиков, построение маршрутов) должны быть разработаны самостоятельно.
* При использовании открытых решений указывается источник и соблюдаются условия лицензии.

Патентный поиск:

* Проверка аналогичных патентов в базах данных (Роспатент, USPTO) для исключения нарушений.
* Особое внимание уделяется функциям, которые могут пересекаться с коммерческими продуктами (например, системы бронирования).

4. Документирование лицензий

Реестр зависимостей:

* Формируется список всех сторонних библиотек, API и инструментов с указанием их лицензий (например, через файл LICENSE.md в репозитории).

Условия использования:

* Для пользователей: публикуется раздел «Правовая информация» с указанием лицензий и авторских прав.
* Для разработчиков: внутренняя документация с требованиями к проверке новых компонентов перед интеграцией.

5. Юридическая экспертиза

Консультации с юристами:

* Проверка лицензионных соглашений для коммерческого использования API.
* Анализ соответствия проекта законам РФ (ГК РФ, часть 4) и международным нормам (GDPR, DMCA).

Резервные решения:

* На случай ограничений API предусматриваются альтернативные технологии (например, замена Booking на Островок).

**2.2.6. Требования по стандартизации**

Информационная система Travel Buddy должна соответствовать международным, отраслевым и национальным стандартам, обеспечивая совместимость, безопасность и качество. Требования охватывают технические, процессуальные и организационные аспекты.

1. Международные стандарты

* ISO/IEC 27001 (Информационная безопасность):
* Реализация мер защиты данных (шифрование, контроль доступа, резервное копирование).
* Сертификация инфраструктуры (при размещении в облачных сервисах, например, Azure ISO 27001 compliant).
* WCAG 2.1 (Доступность веб-контента):
* Контрастность текста (минимум 4.5:1).
* Поддержка навигации с клавиатуры и screen readers.
* Семантическая вёрстка (теги <header>, <nav>, <main>).
* REST API (OpenAPI 3.0):
* Документирование API через Swagger/OpenAPI.
* Стандартные HTTP-коды ответов (200, 400, 401, 500).

2. Национальные стандарты (РФ)

* ГОСТ 34.602-89 (Техническое задание на ИС):
* Соответствие структуры документации (включая разделы «Требования к надежности», «Эргономика»).
* ФЗ-152 (О персональных данных):
* Хранение данных пользователей на серверах в РФ.
* Регистрация в Роскомнадзоре как оператора ПДн.
* ГОСТ Р 57580 (Безопасность финансовых услуг):
* Применяется для модулей оплаты (например, бронирование билетов).

3. Технические стандарты

* W3C (Веб-стандарты):
* Валидная HTML5/CSS3-разметка.
* Кросс-браузерная совместимость (Chrome, Firefox, Safari, Edge).
* Базы данных:
* SQL-стандарты (ANSI SQL) для совместимости с другими СУБД.
* Нормализация данных (3NF для реляционных таблиц).
* Протоколы:
* HTTPS (TLS 1.2/1.3) для всех соединений.
* OAuth 2.0 для авторизации.

4. Процессуальные стандарты

* Agile/Scrum:
* Разработка по спринтам (2-4 недели) с ежедневными stand-up meetings.
* Использование Jira/Trello для управления задачами.
* CI/CD (DevOps):
* Автоматизированные pipeline (GitHub Actions/GitLab CI).
* Тестирование: юнит-тесты (80% покрытия), интеграционные тесты.
* Документирование:
* README.md для разработчиков с описанием архитектуры.
* Пользовательские руководства в формате Markdown или Wiki.

5. Отраслевые стандарты (туризм)

* OPC (Open Travel Alliance):
* Совместимость форматов данных для интеграции с туроператорами.
* IATA (для транспортных модулей):
* Использование стандартных кодов авиакомпаний и аэропортов.

6. Контроль соблюдения стандартов

* Статический анализ кода:
* Инструменты: SonarQube, ESLint (для JavaScript-частей).
* Аудит безопасности:
* Регулярные penetration-тесты (раз в квартал).
* Логирование:
* Стандартизированный формат логов (JSON, Syslog).

**2.2.7. Требования к лингвистическому обеспечению**

Лингвистическое обеспечение системы Travel Buddy должно гарантировать удобство, понятность и корректность взаимодействия пользователей с платформой на русском языке, а также предусматривать возможность расширения функционала для поддержки других языков в будущем. Требования охватывают следующие аспекты:

1. Поддержка русского языка

Интерфейс системы:

* Все элементы интерфейса (меню, кнопки, формы, сообщения) должны быть на русском языке.
* Тексты должны быть адаптированы под целевую аудиторию, избегая сложных технических терминов, где это возможно.
* Использование дружелюбного и вежливого тона в сообщениях (например, "Добро пожаловать!", "Пожалуйста, заполните форму").

Контент и подсказки:

* Все инструкции, подсказки (тултипы) и сообщения об ошибках должны быть доступны на русском языке.
* Примеры:
* При пустом поле ввода: "Поле не может быть пустым".
* При успешном действии: "Ваши изменения сохранены".
* Встроенная справка (FAQ) должна быть написана на русском языке с четкими и понятными формулировками.

Форматы данных:

* Даты: отображение в формате ДД.ММ.ГГГГ (например, 15.04.2025).
* Время: 24-часовой формат (например, 14:30).
* Числа: разделитель тысяч — пробел, десятичный разделитель — запятая (например, 1 000,50).
* Валюта: символ рубля (₽) с отображением суммы (например, 1 500 ₽).

2. Локализация пользовательского контента

Профили пользователей:

* Поля профиля (имя, город, интересы) должны поддерживать ввод на русском языке, включая обработку кириллицы.
* Пример: поле "Город" должно корректно отображать названия городов России и других стран на русском языке (например, "Москва", "Санкт-Петербург").

Поиск и фильтрация:

* Поиск по городам, маршрутам и другим параметрам должен учитывать русскую морфологию (например, поиск по "Москва" должен находить также "Москве", "Москву").
* Автодополнение в полях ввода должно работать с кириллицей.

Уведомления и email-рассылки:

* Все системные уведомления (регистрация, бронирование, сообщения) должны отправляться на русском языке.
* Шаблоны писем должны быть адаптированы под русскоязычную аудиторию, включая приветствия и подписи (например, "С уважением, команда Travel Buddy").

3. Подготовка к мультиязычности (английский язык в перспективе)

Архитектура системы:

* Реализация системы должна предусматривать возможность добавления новых языков без изменения кода. Для этого следует использовать:
* Файлы ресурсов (resource files) для хранения текстовых элементов.
* Поддержку стандарта i18n (интернационализации) для легкого перевода интерфейса.
* Все текстовые строки в коде должны быть вынесены в отдельные файлы локализации (например, ru-RU.json, en-US.json в будущем).
* При проектировании базы данных необходимо предусмотреть хранение многоязычного контента (например, таблицы с переводами для городов, описаний отелей).
* API должно поддерживать параметр выбора языка (например, Accept-Language: ru-RU).

4. Проверка орфографии и грамматики

Валидация ввода:

* Система должна проверять орфографию в полях ввода, где это уместно (например, в описаниях маршрутов, сообщениях чата).
* Использование библиотек для проверки русской орфографии (например, Hunspell с русским словарем).

Модерация контента:

* Автоматическая фильтрация нецензурной лексики в сообщениях и отзывах с использованием предварительно составленных списков запрещенных слов.
* Возможность ручной модерации контента администратором.

5. Лингвистическая экспертиза

Тестирование:

* Проверка всех текстовых элементов интерфейса носителями русского языка на предмет понятности и отсутствия ошибок.
* Юзабилити-тестирование с русскоязычными пользователями для оценки удобства формулировок.

Документирование:

* Создание глоссария терминов, используемых в системе, для обеспечения единообразия.
* Подготовка руководства для переводчиков на случай добавления новых языков.

6. Особые требования для чата и коммуникации

Мессенджер:

* Поддержка смайлов и эмодзи, которые популярны среди русскоязычных пользователей.
* Автоматическое определение языка вводимого текста (для будущей модерации).

Уведомления:

* Персонализированные сообщения (например, "Привет, Иван!" вместо "Привет, пользователь!").

Все тексты и форматы должны соответствовать нормам современного русского языка и учитывать особенности цифрового взаимодействия.

**2.3. Экономический расчет**

Экономический расчет Travel Buddy включает проведение комплексной оценки экономической целесообразности разработки, внедрения и последующего использования веб-сервиса. В расчётах учитываются основные этапы разработки, затраты на технические ресурсы, оценка доходности и срок окупаемости проекта. Основной целью расчета является определение рентабельности проекта и обоснование инвестиционной привлекательности.

Для реализации проекта была сформирована проектная команда, включающая следующих специалистов:

* Junior-разработчик (бэкенд)
* TeamLead-разработчик (бэкенд)
* Frontend-разработчик
* UI/UX дизайнер
* Аналитик / Продакт-менеджер
* Руководитель проекта

Для расчета ФОТ использовались усреднённые рыночные значения заработных плат по России.

Таблица 2.3.1. Фонд оплаты труда по должностям

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Должность** | **ЗП/мес., ₽** | **Срок участия** | **ЗП за период, ₽** | **Отчисления (30,2%), ₽** | **Итого ФОТ, ₽** |
| Junior-разработчик | 80 000 | 3 мес. | 240 000 | 72 480 | 312 480 |
| TeamLead-разработчик | 130 000 | 3 мес. | 390 000 | 117 780 | 507 780 |
| Frontend-разработчик | 80 000 | 3 мес. | 240 000 | 72 480 | 312 480 |
| UI/UX-дизайнер | 60 000 | 2 мес. | 120 000 | 36 240 | 156 240 |
| Аналитик / PO | 150 000 | 3 мес. | 450 000 | 135 900 | 585 900 |
| Руководитель проекта | 200 000 | 3 мес. | 600 000 | 181 200 | 781 200 |
| **Итого** |  |  | **2 040 000** | **616 080** | **2 656 080** |

Таким образом, совокупный фонд оплаты труда (ФОТ) составляет 2 656 080 рублей.

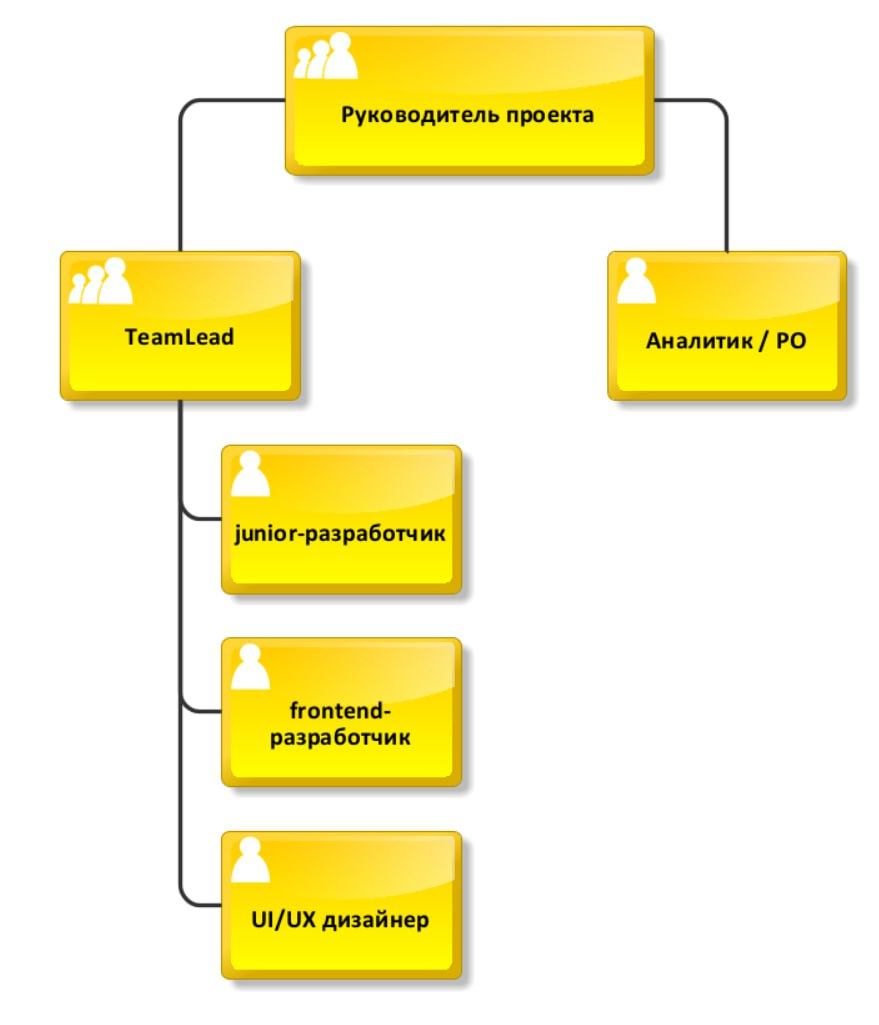


Рис. 2.3.1. Организационная диаграмма веб-сервиса

Затраты на программную реализацию включают стадии:

* Анализ требований и проектирование
* Разработка архитектуры БД и API
* Разработка клиентской части (Blazor)
* Интеграция внешних API
* UI/UX-дизайн интерфейсов
* Тестирование и отладка
* Внедрение и сопровождение

Кроме разработки, необходимо учитывать постоянные и разовые затраты, связанные с эксплуатацией проекта, представленные в таблице 2.3.3.

Таблица 2.3.3. Прочие статьи расходов

|  |  |  |  |
| --- | --- | --- | --- |
| Статья расходов | Количество / период | Единичная стоимость, ₽ | Общая сумма, ₽ |
| Облачный сервер (Azure, 12 мес.) | 1 год | 3 000 / мес | 36 000 |
| Домен и SSL-сертификат | 1 раз в год | 4 000 | 4 000 |
| Хостинг базы данных и резервных копий | 1 год | 2 000 / мес | 24 000 |
| Подписка на инструменты (Rider, DataGrip, GitHub Pro и др.) | 1 год | 26 000 (289 $) | 26 000 |
| Прочие расходы (маркетинг, тестирование, копирайтинг и т.п.) | - | - | 10 000 |
| Итого | 100 000 ₽ | | |

Данные расходы необходимы для устойчивой работы проекта, его размещения в сети, защиты и сопровождения. Подписки на инструменты разработки и дизайна позволяют ускорить процесс и повысить качество продукта.

На рис. 2.3.2. приведена модель IT инфраструктуры веб-сервиса Travel Buddy, выполненная в среде CASE-средства ARIS.

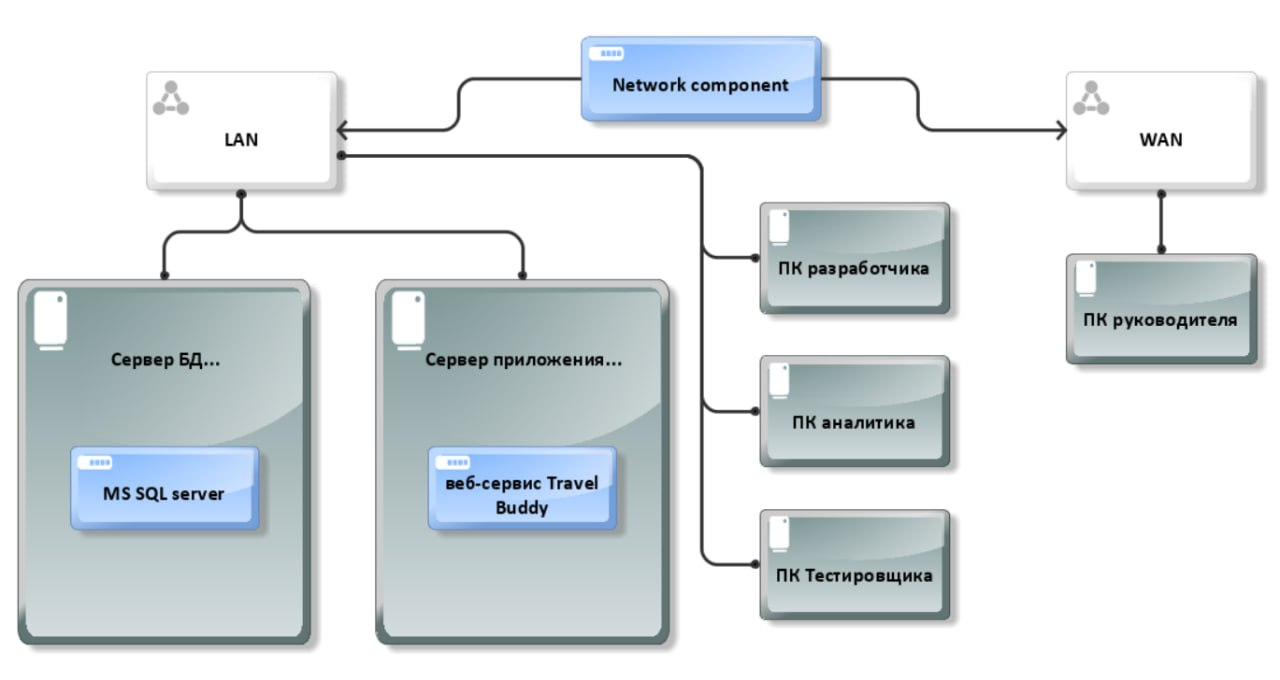


Рис. 2.3.2. Модель IT инфраструктуры веб-сервиса

Подразумевается, что каждый сотрудник работает удаленно с личного компьютера.

**2.3.1. Оценка трудоемкости разработки ИС**

Для оценки трудоёмкости разработки веб-сервиса Travel Buddy применена методика Use Case Points (UCP) — одна из наиболее точных и распространённых методик в области оценки усилий при разработке информационных систем. Она учитывает функциональную сложность, технические особенности проекта и квалификацию команды. Это позволяет сформировать реалистичное представление о затратах времени и ресурсов.

1. Определение веса действующих лиц (Actors)

Каждое действующее лицо в системе — пользователь, администратор, внешний сервис — имеет определённую сложность взаимодействия. Вес определяется по следующей шкале:

* Простой актор (например, система аутентификации) — 1 балл
* Средний (например, REST API внешнего сервиса) — 2 балла
* Сложный (например, живой пользователь или администратор с множеством действий) — 3 балла

Таблица 2.3.1.1. Определение веса действующих лиц

|  |  |  |
| --- | --- | --- |
| **Действующее лицо** | **Тип** | **Коэффициент** |
| Пользователь системы | Сложный | 3 |
| Администратор системы | Сложный | 3 |
| API внешнего сервиса | Средний | 2 |
| Бэкенд/сервер | Средний | 2 |

Общий вес: A = 2×3 + 2×2 = 10

2. Оценка сложности вариантов использования (Use Cases)

Каждая функция в системе (use case) оценивается по типу:

* Простой (до 3 шагов, 1 экран) — 5 баллов
* Средний (до 7 шагов, несколько экранов) — 10 баллов
* Сложный (более 7 шагов, взаимодействие с внешними сервисами, сложные сценарии) — 15 баллов

Таблица 2.3.1.2. Оценка сложности вариантов использования

|  |  |  |
| --- | --- | --- |
| **Вариант использования** | **Тип** | **Коэффициент** |
| Регистрация и авторизация | Простой | 5 |
| Создание маршрута | Средний | 10 |
| Поиск попутчиков | Средний | 10 |
| Встроенный мессенджер | Сложный | 15 |
| Интеграция с внешними API | Сложный | 15 |
| Управление пользователями | Средний | 10 |
| Визуализация карты маршрута (OpenStreetMap) | Сложный | 15 |

UC = 1×5 + 3×10 + 3×15 = 5 + 30 + 45 = 80

UUCP = A + UC = 10 + 80 = 90

3. Оценка технической сложности (TCF)

Методика UCP включает оценку технических факторов (performance, безопасность, повторное использование компонентов и т.д.). Сумма всех технических весов: 26.

Формула: TCF = 0,6 + 0,01 × Σ(весов)

TCF = 0,6 + 0,01×26 = 0,86

4. Оценка квалификации команды (EF)

В фактор квалификации команды входят такие показатели, как опыт, мотивация, стабильность требований и др.

Таблица 2.3.1.3. Оценка квалификации команды (EF)

|  |  |  |  |
| --- | --- | --- | --- |
| **Показатель** | **Вес** | **Значение** | **Взвешенное** |
| Знакомство с технологией | 1.5 | 5 | 7.5 |
| Опыт в разработке | 0.5 | 4 | 2 |
| Объектно-ориентированное мышление | 1.0 | 4 | 4 |
| Наличие продакт-менеджера | 0.5 | 5 | 2.5 |
| Мотивация | 1.0 | 5 | 5 |
| Стабильность требований | 2.0 | 4 | 8 |
| Частичная занятость | -1 | 2 | -2 |
| Сложные языки | -1 | 2 | -2 |
| **Итого:** |  |  | **25,5** |

EF = 1,4 + (-0,03×25,5) = 1,4 – 0,765 = 0,635

**Расчёт UCP и трудоемкости**

UCP = UUCP × TCF × EF;

UCP = 90 × 0,86 × 0,635 = 49,2

Принимаем трудозатраты: 20 часов на 1 UCP

Общая трудоемкость = 49,2 × 20 = 984 ч

**Итого:**

* UUCP (Unadjusted Use Case Points) = 90
* TCF (Technical Complexity Factor) = 0,86
* EF (Environmental Factor) = 0,635
* UCP = 90 × 0,86 × 0,635 = 49,2
* На 1 UCP ≈ 20 часов работы → 984 часов в общей сложности

На основе методики оценки трудоёмкости UCP (Use Case Points), расчёт показал общую продолжительность проекта в 984 человеко-часов. При средней загрузке специалистов в 160 часов в месяц и составе команды из 6 человек, срок реализации составляет 1 календарный месяц.

На основании вышеизложенных данных о фонде оплаты труда, трудоёмкости проекта и дополнительных эксплуатационных расходах производится расчет общей стоимости проекта. Этот этап позволяет обобщить все понесённые затраты и представить их в виде единой сводной таблицы. Полученные значения дают представление о необходимом объёме инвестиций для реализации веб-сервиса Travel Buddy, включая разработку, техническое обеспечение и сопутствующие расходы.

Таблица 2.3.4. Общая стоимость проекта

|  |  |
| --- | --- |
| **Категория расходов** | **Сумма, ₽** |
| Программное и тех. обеспечение | 100 000 |
| Фонд оплаты труда | 2 656 080 |
| **Общая стоимость проекта** | **2 756 080** |

Общая стоимость проекта позволяет наглядно оценить итоговую финансовую нагрузку и служит базой для дальнейшего анализа эффективности вложений.

Далее рассмотрим ключевые экономические показатели, отражающие эффективность и инвестиционную привлекательность проекта. К ним относятся предполагаемые источники дохода, сроки окупаемости и рентабельность.

Таблица 2.3.5. Эффективность и окупаемость

|  |  |
| --- | --- |
| **Источник дохода** | **Доход/год** |
| Подписка (1000 × 150 ₽ × 12 мес) | 1 800 000 |
| Комиссии от бронирований | 120 000 |
| **Итого доход:** | 1 920 000 |

Эффективность и окупаемость содержит расчёты ожидаемой выручки, срока возврата инвестиций и оценки годового ROI. Эти показатели позволяют сделать вывод о целесообразности запуска проекта и его перспективности в рыночных условиях.

Срок окупаемости = Общая сумма инвестиций / Годовой доход

Срок окупаемости = 2 756 080 / 1 920 000 ≈ 1,4 года

Для оценки рентабельности проекта рассчитывается показатель ROI (Return on Investment) по формуле:

ROI = (Прибыль / Инвестиции) × 100%

Инвестиции: 2 756 080 ₽

Годовая прибыль: 1 920 000 – 570 000 = 1 350 000 ₽

ROI ≈ (1 350 000 / 2 756 080) × 100% ≈ 48,9% в год после выхода на окупаемость.

Проект Travel Buddy требует значительных начальных вложений, связанных с персоналом и разработкой. Однако его потенциальный рынок, масштабируемость и комбинация функций (создание маршрутов + поиск попутчиков + чат) делают его перспективным. Несмотря на срок окупаемости в полтора года, последующая прибыльность, SaaS-модель и возможность выхода на международный рынок свидетельствуют о высокой инвестиционной привлекательности проекта.

**2.3.2. Оценка рисков при создании ИС**

Любой проект в сфере информационных технологий сопряжён с определёнными рисками, которые могут повлиять на сроки, бюджет или качество результата. Для минимизации негативных последствий важно выявить и проанализировать потенциальные риски на раннем этапе.

Таблица 2.3.2.1. Основные риски проекта.

|  |  |  |  |
| --- | --- | --- | --- |
| Риск | Вероятность | Влияние | Меры снижения |
| Задержка сроков из-за увеличения объёма работ | Средняя | Высокое | Использование Agile, контроль спринтов |
| Изменения требований заказчика | Средняя | Среднее | Чёткая постановка задач, прототипирование |
| Технические ошибки или уязвимости | Высокая | Высокое | Покрытие тестами, код-ревью, CI/CD |
| Недоступность внешних API | Средняя | Среднее | Кэширование, резервные API |
| Проблемы с хостингом и производительностью | Низкая | Высокое | Использование надёжного облачного провайдера |
| Нехватка финансирования | Низкая | Высокое | Бюджетное планирование, минимизация затрат на MVP |
| Угрозы безопасности данных | Средняя | Высокое | Шифрование, авторизация, защита от атак |
| Слабый отклик пользователей | Средняя | Среднее | Маркетинговая стратегия, опросы, улучшение UX |

Проведённый анализ рисков позволяет выработать стратегию управления ими на всех этапах проекта: от планирования и реализации до тестирования и поддержки.