**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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Е.Маслов

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 2](#_Toc196667421)

[Глава 1. Постановка задачи 3](#_Toc196667422)

[1.1 Цель создания информационной системы. Характеристики системы 3](#_Toc196667423)

[1.2 Анализ информационных систем 4](#_Toc196667424)

[1.3 Анализ и выбор средств проектирования и разработки ИС 5](#_Toc196667425)

[Глава 2. Проектирование информационной системы 7](#_Toc196667426)

[2.1. Требования к структуре и функциям, выполняемым системой 7](#_Toc196667427)

[2.2. Требования к видам обеспечения 9](#_Toc196667428)

[2.2.1. Требования к надежности системы 9](#_Toc196667429)

[2.2.2. Требования к безопасности системы и защите информации 10](#_Toc196667430)

[2.2.3. Требования к эргономике системы 11](#_Toc196667431)

[2.2.4. Требования к эксплуатации системы 12](#_Toc196667432)

[2.2.5. Требования к патентной чистоте 13](#_Toc196667433)

[2.2.6. Требования по стандартизации 13](#_Toc196667434)

[2.2.7. Требования к лингвистическому обеспечению 15](#_Toc196667435)

[2.3. Экономический расчет 19](#_Toc196667436)

# Введение

В современном мире автомобильный сервис играет ключевую роль в поддержании исправности транспортных средств, обеспечении их безопасности и продлении срока службы. Однако организация сервисного обслуживания автомобилей, особенно для частных клиентов и крупных автопарков, сопряжена с рядом проблем: сложности в поиске подходящих запчастей, планировании ремонтных работ, отсутствии централизованного подхода к учёту истории заказов и управления обслуживанием. В условиях быстрого развития цифровых технологий становится всё более актуальным создание специализированных платформ, которые способны автоматизировать процессы ремонта, диагностики и обслуживания автомобилей.

Выбранная тема выпускной квалификационной работы посвящена разработке информационной системы — веб-сервиса для автосервиса, предназначенного для автоматизации работы сервиса, учёта заказов, диагностики и общения между клиентами и сервисом. Актуальность проекта обусловлена растущим спросом на удобные и надёжные решения для управления сервисами и обеспечения качественного обслуживания автомобилей, а также недостатком комплексных систем, которые бы интегрировали в себе функции диагностики, управления заказами и работы с клиентами.

Практическая значимость данного проекта заключается в упрощении организации сервисного обслуживания, экономии времени и средств как для клиентов, так и для самого автосервиса, а также в повышении качества работы за счёт централизованного хранения данных, автоматизации процессов и улучшения взаимодействия с клиентами. В процессе разработки решались задачи обеспечения надёжности, масштабируемости, безопасности данных, а также интеграции с внешними сервисами и системами для оптимизации работы автосервиса.

Целью ВКР является создание веб-приложения для автосервиса, которое объединяет функциональность для управления заказами, учёта истории обслуживания, диагностики и работы с клиентами в единую информационную систему.

Задачи работы: • провести анализ аналогичных информационных систем для автосервисов; • спроектировать архитектуру и функциональность веб-сервиса; • реализовать систему на основе современных технологий (Django, PostgreSQL); • обеспечить интеграцию с внешними системами (например, для заказа запчастей); • протестировать и внедрить систему; • проанализировать экономическую эффективность проекта.

# Глава 1. Постановка задачи

# 1.1 Цель создания информационной системы. Характеристики системы

Цель разработки информационной системы для автосервиса заключается в создании универсальной и масштабируемой платформы, которая обеспечит удобство для владельцев автомобилей и автосервисов, а также улучшит управление сервисами и взаимодействие с клиентами. В условиях постоянного роста автопарка и увеличивающихся требований к качеству обслуживания, создание эффективного веб-сервиса для автосервисов становится важной задачей.

Система должна позволить пользователям:

* Легко записываться на диагностику и ремонт с помощью онлайн-записи.
* Просматривать историю заказов, включая информацию о выполненных работах и замененных запчастях.
* Генерировать отчет о диагностике и ремонте для пользователей.
* Интерактивно общаться с клиентами через встроенную систему мессенджера.
* Заказывать запчасти, просматривая доступные варианты и их стоимость.

Характеристики разрабатываемой системы:

* Масштабируемость: Архитектура платформы будет поддерживать добавление новых функциональных возможностей, таких как поддержка новых типов ремонтов, интеграции с дополнительными API для продажи запчастей и услуг.
* Многоуровневая структура: Система будет разделена на клиентскую, серверную и базовую части. Это позволит более гибко управлять развитием и улучшать взаимодействие различных модулей.
* Интеграция с внешними API: Платформа будет интегрирована с внешними сервисами для проверки наличия запчастей, получения актуальных данных о стоимости и доставки, а также бронирования услуг.
* Безопасность и конфиденциальность: Будет предусмотрена защита личных данных клиентов, использование механизмов аутентификации и авторизации, а также шифрование данных на всех этапах их обработки.

Целевая аудитория системы:

* Владельцы автомобилей, которые хотят быстро и удобно записываться на обслуживание.
* Операторы автосервисов, которым необходимо улучшить процесс управления заказами и истории обслуживания.
* Сотрудники автосервисов, которым необходимы инструменты для автоматизации взаимодействия с клиентами и контроля состояния заказов.

# 1.2 Анализ информационных систем

Для того чтобы понять, какие функции должны быть интегрированы в нашу систему, необходимо провести анализ существующих решений на рынке. На данный момент существует множество специализированных сервисов для автосервисов и продажи автозапчастей, однако ни один из них не объединяет все ключевые аспекты работы автосервиса и взаимодействия с клиентами.

Существующие системы можно разделить на несколько категорий:

1. Системы для онлайн-записи на обслуживание — такие как CarService и другие локальные решения. Они позволяют пользователям записываться на диагностику или ремонт, но не обеспечивают полную историю заказов и запчастей.
2. Платформы для продажи автозапчастей — как Autodoc, предоставляющие пользователю доступ к огромному ассортименту запчастей, но не имеющие связки с сервисом для выполнения работы.
3. Системы для управления заказами в автосервисах — как MyCarWorkshop, которые охватывают только часть функций, требующихся для полного цикла обслуживания автомобиля.

Пример таблицы для сравнения функционала сервисов:

Таблица 1.2.1. Сравнение функционала различных сервисов

| Сервис | Запись на ремонт | История заказов | Бронирование запчастей | Общение с клиентами | Генерация отчетов |
| --- | --- | --- | --- | --- | --- |
| CarService | Да | Да | Нет | Да | Нет |
| Autodoc | Нет | Нет | Да | Нет | Нет |
| MyCarWorkshop | Да | Да | Да | Да | Да |
| Наш сервис | Да | Да | Да | Да | Да |

Анализ слабых сторон существующих систем:

* CarService: ограниченная функциональность — отсутствует интеграция с сервисами для заказа запчастей и генерации отчетов.
* Autodoc: преимущественно ориентирован на продажу запчастей, а не на полный цикл обслуживания автомобиля.
* MyCarWorkshop: хотя и предлагает хороший функционал для управления заказами и общения, не поддерживает интеграцию с внешними сервисами и не включает генерирование отчетов.

Таким образом, существующие решения частично решают задачи, но они не могут покрыть весь спектр потребностей автосервисов и их клиентов.

# 1.3 Анализ и выбор средств проектирования и разработки ИС

Для создания информационной системы автосервиса необходимо выбрать оптимальные программные и технические средства, которые обеспечат стабильную работу и масштабируемость системы. Выбор технологий определяет не только качество сервиса, но и возможность дальнейшего расширения функционала и его интеграции с внешними сервисами.

Выбор стек технологий:

* Backend: Для серверной части будет выбран Python с фреймворком Django или Flask. Django предлагает мощную встроенную функциональность для быстрой разработки и работы с базами данных. Использование Python обеспечит гибкость и простоту интеграции с внешними сервисами.
* Frontend: Для разработки клиентской части будет использоваться React.js или Vue.js. Эти фреймворки позволяют разрабатывать удобные и динамичные пользовательские интерфейсы.
* База данных: Для хранения данных о заказах и пользователях будет использована PostgreSQL или MySQL — надежные реляционные базы данных, поддерживающие сложные запросы и транзакции.
* Интеграция с внешними API: Платформа будет интегрирована с внешними сервисами для продажи запчастей (например, через API сервисов типа Autodoc) и проверки наличия и стоимости запчастей.

Для повышения безопасности системы будет использована аутентификация через JWT-токены, а данные будут шифроваться при передаче и хранении. Также будет обеспечена защита от SQL-инъекций, XSS-атак и других распространенных угроз.

Для хостинга системы планируется использовать облачные решения, такие как AWS или Azure, что обеспечит гибкость в масштабировании и надежность в хранении данных.

Минимальные системные требования:

* Процессор: 2 ядра;
* Оперативная память: 4 ГБ;
* Диск: 50 ГБ SSD;
* Операционная система: Ubuntu 22.04 LTS.

Платформа будет обеспечивать горизонтальное масштабирование и возможность разделения сервиса на микросервисы в случае роста нагрузки.

Инструменты для разработки и командной работы:

* Visual Studio Code / PyCharm — для написания и отладки кода.
* Postman — для тестирования API.
* Figma — для проектирования пользовательского интерфейса.
* Git + GitHub — для контроля версий.
* Trello — для управления задачами и документированием.

Все выбранные инструменты и технологии являются современными, активно развиваются и обладают широкой поддержкой в сообществе, что позволяет гарантировать стабильность и масштабируемость проекта.

Таким образом, информационная система для автосервисов будет использовать передовые технологии для создания надежной, безопасной и масштабируемой платформы, которая обеспечит полный цикл обслуживания клиентов, от записи на диагностику до заказа запчастей и генерации отчетов.

# Глава 2. Проектирование информационной системы

# 2.1. Требования к структуре и функциям, выполняемым системой

Информационная система для автосервиса разрабатывается как веб-платформа для эффективного управления заказами на диагностику и ремонт автомобилей, а также для управления запасными частями. Для полноценной реализации её задач необходимо сформулировать требования к архитектуре и функциональности системы.

Система будет построена на многоуровневой архитектуре, включающей:

1. **Клиентский уровень (Frontend)** — реализован с использованием Django, HTML и CSS для визуализации интерфейса. Он будет обеспечивать взаимодействие пользователя с системой через удобный веб-интерфейс для авторизации, создания и просмотра заказов, истории обслуживания, а также для работы с данными о запасных частях и диагностических отчётах.
2. **Серверный уровень (Backend)** — построен с использованием Django и Django REST Framework, выполняет обработку бизнес-логики, работу с базой данных, маршрутизацию запросов и взаимодействие с внешними сервисами для обработки оплат и отправки уведомлений.
3. **Уровень данных (Database)** — реализован с помощью PostgreSQL или MySQL, обеспечивающий хранение данных о пользователях, заказах, запчастях, диагностических отчётах и истории обслуживания.

**Архитектура базы данных**

Система использует реляционную базу данных, ориентированную на безопасность, масштабируемость и расширяемость. Основные таблицы базы данных:

* **Users** — хранит учетные записи пользователей, включая email, имя, фотографию, адрес и дату рождения.
* **Orders** — данные о заказах на диагностику и ремонт.
* **Parts** — информация о запасных частях.
* **Reports** — диагностические отчёты, связанные с заказами.
* **OrderHistory** — история изменений в заказах.
* **Roles** и **UserRoles** — роли пользователей (например, администратор, механик, приёмщик).

**Функциональные требования**

1. **Регистрация и авторизация пользователей**
   * Регистрация с вводом email, имени, пароля и дополнительных данных.
   * Возможность восстановления пароля и защита от несанкционированного доступа с помощью шифрования паролей.
   * Роли пользователей: администратор, механик, приёмщик, клиент.
2. **Профили пользователей**
   * Панель управления личными данными, с возможностью редактирования профиля (фото, контактные данные, история заказов).
   * Авторизация через сторонние сервисы (например, через Google или социальные сети).
3. **Управление заказами**
   * Создание заказов на диагностику, ремонт и покупку запчастей.
   * Привязка заказов к пользователю (клиенту) и механику.
   * Интеграция с базой данных для отображения истории заказов и состояния каждого заказа.
4. **Управление запасными частями**
   * Возможность добавления и редактирования информации о запасных частях, связанных с заказами.
   * Поддержка списка запчастей для различных типов автомобилей и видов ремонта.
5. **Диагностические отчёты**
   * Возможность создания и отправки диагностических отчётов пользователю.
   * Поддержка возможности добавления фотографий и описания неисправностей, а также рекомендаций по ремонту.
6. **Управление ролями и правами пользователей**
   * Определение доступа к функционалу для различных ролей (например, только администратор может добавлять запчасти, а механик может редактировать отчёты).

Таким образом, структура и функциональность информационной системы автосервиса обеспечат удобное управление заказами и запасными частями, а также эффективное взаимодействие между пользователями различных ролей. Архитектура системы и база данных созданы с учетом требований масштабируемости и надежности, что позволит расширять функциональность и адаптировать систему под новые задачи.

# 2.2. Требования к видам обеспечения

# 2.2.1. Требования к надежности системы

Надежность системы является критически важным фактором, так как она должна обеспечивать бесперебойную работу сервиса для всех пользователей, включая механиков, администраторов и клиентов. Требования к надежности включают следующие аспекты:

1. **Отказоустойчивость**
   * Система должна работать в условиях возможных сбоев в отдельном компоненте. Для этого развертывание серверной части будет происходить в виде кластера с минимум двумя серверами, чтобы обеспечить резервирование и отказоустойчивость.
   * Для базы данных будет использоваться репликация для предотвращения потери данных при сбоях.
2. **Резервное копирование**
   * Регулярное создание резервных копий базы данных для обеспечения сохранности данных и возможности восстановления системы в случае сбоя.
   * Хранение резервных копий в защищённом хранилище (например, в облаке) для предотвращения потери данных.
3. **Мониторинг и логирование**
   * Внедрение системы мониторинга с возможностью отслеживания критических метрик, таких как время отклика API, загрузка серверов, состояние базы данных и прочее.
   * Логирование всех действий в системе для обеспечения прозрачности и быстрого восстановления после сбоев.
4. **Тестирование**
   * Применение юнит-тестов для проверки корректности работы основных функций и интеграционных тестов для проверки взаимодействия между модулями системы.
5. **Целевые показатели надежности**
   * Система должна иметь показатель доступности (uptime) не менее 99.5% в месяц и время восстановления (MTTR) не более 30 минут для критических сбоев.

# 2.2.2. Требования к безопасности системы и защите информации

Безопасность системы автосервиса критически важна, поскольку она должна защищать персональные данные пользователей и обеспечивать защиту от атак.

1. **Шифрование данных**
   * Все данные передаются через защищённые каналы (HTTPS) с использованием TLS 1.2 или выше.
   * Пароли пользователей хранятся с использованием алгоритма bcrypt с "солью".
2. **Аутентификация и авторизация**
   * Реализация двухфакторной аутентификации для повышения безопасности.
   * Разграничение доступа через ролевую модель (RBAC) с различными уровнями прав для пользователей (администратор, механик, клиент).
3. **Защита от атак**
   * Защита от SQL-инъекций с использованием параметризованных запросов и ORM (например, Django ORM).
   * Меры против XSS (санитизация ввода) и CSRF (защита токенами).
4. **Соответствие стандартам**
   * Соблюдение стандартов защиты данных, включая GDPR для пользователей из ЕС.
5. **Физическая безопасность**
   * Серверы будут размещены в защищённых дата-центрах с ограниченным доступом для персонала, а доступ к данным будет защищён с использованием двухфакторной аутентификации для сотрудников.

# 2.2.3. Требования к эргономике системы

Эргономика системы автосервиса направлена на обеспечение удобства и эффективности пользователей при взаимодействии с платформой. Детализированные требования включают:

1. **Интуитивный интерфейс**:
   * Логичная структура навигации:
     + Главное меню должно содержать основные разделы: "Главная", "Мой кабинет", "История заказов", "Запчасти", "Обслуживание" – слева, "Вход" и "Регистрация" (если пользователь не аутентифицирован) и "Профиль" (если аутентифицирован) – справа.
     + Минимизация кликов для выполнения ключевых действий (не более 3 шагов).
   * **Единообразие элементов**:
     + Стилевое согласование кнопок, полей ввода и других элементов.
     + Использование стандартных иконок для унификации восприятия.
   * **Подсказки и инструкции**:
     + Встроенные тултипы для сложных элементов.
     + Анимация загрузки для операций, требующих времени.
2. **Визуальная иерархия**:
   * Заголовок крупный и читаемый.
   * Подзаголовки, читаемые на фоне изображения с хорошим контрастом.
   * Информационные блоки разделены визуально для облегчения восприятия.
3. **Адаптивный дизайн**:
   * Поддержка устройств с экранами от 320px (мобильные устройства) до 1920px (десктоп).
   * Использование гибкой сетки (CSS Grid/Flexbox) для адаптации интерфейса.
   * Мобильная версия с упрощённым меню и крупными кликабельными элементами.
4. **Цветовая схема и темы**:
   * Темная тема для снижения нагрузки на глаза.
   * Высокая контрастность текста с фоном.
   * Адаптация активных элементов под фирменную палитру.
5. **Учет принципов UX/UI**:
   * Принцип близости: группировка связанных элементов.
   * Соотношение контраста текста и фона не менее 4.5:1.
   * Обратная связь через визуальные отклики при взаимодействии с элементами интерфейса.
6. **Доступность (Accessibility)**:
   * Поддержка WCAG 2.1 для обеспечения доступности.
   * Навигация с клавиатуры и семантическая вёрстка.

# 2.2.4. Требования к эксплуатации системы

Эксплуатация системы автосервиса должна обеспечивать удобство и эффективность как для конечных пользователей (клиентов), так и для персонала, обслуживающего систему. Важные моменты:

1. **Требования к пользовательской эксплуатации**:
   * Доступность 24/7: приложение должно быть доступно круглосуточно через веб-браузер.
   * Поддержка всех современных браузеров (Chrome, Edge, Firefox, Safari).
   * Простой процесс обновления без прерывания работы пользователей.
   * Мультиязычный интерфейс для поддержки разных языков.
2. **Требования к сопровождению и администрированию**:
   * **Панель администратора** для управления пользователями, заказами и запчастями.
   * **Логирование** с хранением истории авторизаций, ошибок и запросов.
   * **Мониторинг системы** для отслеживания производительности и стабильности.
3. **Требования к масштабируемости и резервированию**:
   * **Горизонтальное масштабирование** для увеличения нагрузки.
   * Репликация и бэкапы базы данных для защиты данных.
4. **Техническая поддержка**:
   * Уровни поддержки для решения запросов и инцидентов.
   * Регламент времени отклика на инциденты.

# 2.2.5. Требования к патентной чистоте

Для системы автосервиса необходимо соблюдать следующие требования:

1. **Использование открытых и лицензионных технологий**:
   * Применение технологий с разрешительными лицензиями (MIT, Apache 2.0, GPL) для фреймворков и библиотек.
2. **Оригинальность дизайна и контента**:
   * Логотип и фирменный стиль должны быть разработаны с нуля или приобретены с передачей исключительных прав.
3. **Патентная чистота алгоритмов и функционала**:
   * Уникальность ключевых алгоритмов и функционала, например, обработка заказов, работа с запчастями и диагностика.

# 2.2.6. Требования по стандартизации

Информационная система автосервиса 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, для покупки запчастей и услуг ремонта).
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 (автосервис)**
   * **ISO 9001 (Управление качеством)**:
     + Соответствие стандартам управления качеством для автосервисных услуг.
     + Совместимость с требованиями обеспечения качества обслуживания клиентов.
   * **ISO 13485 (Управление качеством для медицинских устройств)**:
     + Применяется в случае работы с автосервисами, занимающимися обслуживанием автомобилей с медицинским оборудованием (например, для перевозки инвалидов).
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 автосервиса 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 ₽).

1. **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 маркам автомобилей, заказам и другим параметрам должен учитывать русскую морфологию (например, поиск по "Форд" должен находить также "Форде", "Форда").
  + Автодополнение в полях ввода должно работать с кириллицей.

**Уведомления и email-рассылки**:

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

1. **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 марок автомобилей и описаний услуг).
   * API должно поддерживать параметр выбора языка (например, Accept-Language: ru-RU).
2. **Проверка орфографии и грамматики**  
   **Валидация ввода**:
   * Система должна проверять орфографию в полях ввода, где это уместно (например, в описаниях запчастей, сообщения о неисправностях).
   * 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.

1. **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.

1. **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 для информационной системы автосервиса включает в себя комплексную оценку 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** (будет добавлен позже).

Затраты на программную реализацию включают следующие 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 и др.) | 1 год | 26 000 | 26 000 |
| Прочие расходы (маркетинг, тестирование, копирайтинг и т.п.) | - | - | 10 000 |
| **Итого** |  |  | 100 000 |

Эти 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.

**Рис. 2.3.2. Модель IT инфраструктуры веб-сервиса** (будет добавлен позже).

**2.3.1. Оценка трудоемкости разработки ИС**

Для оценки трудоёмкости разработки информационной системы автосервиса применяется методика Use Case Points (UCP), которая учитывает функциональную сложность, технические особенности проекта и квалификацию команды.

1. **Определение веса действующих лиц (Actors)**

Каждое действующее лицо в системе имеет определённую сложность взаимодействия. Вес определяется по следующей шкале:

* Простой актор (например, система аутентификации) — 1 балл
* Средний актор (например, REST API внешнего сервиса) — 2 балла
* Сложный актор (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 |  |  |

1. **Оценка сложности вариантов использования (Use Cases)**

Каждая функция в системе 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 | Сложный | 15 |
| **UC** = 1×5 + 3×10 + 3×15 = 80 |  |  |

**UUCP = A + UC = 10 + 80 = 90**

1. **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**

1. **Оценка квалификации команды (EF)**

**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) = 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**

**2.3.2. Оценка рисков при создании ИС**

Любой проект в сфере информационных технологий сопряжён с определёнными рисками. 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 |
| Угрозы безопасности данных | Средняя | Высокое | Шифрование, авторизация, защита от атак |
| Слабый отклик пользователей | Средняя | Среднее | Маркетинговая стратегия, улучшение UX |

Анализ рисков позволяет выработать стратегию управления ими на всех этапах проекта.