Оглавление

[Базовая информация 1](#_Toc210765375)

[Образование 1](#_Toc210765376)

[Места работы 1](#_Toc210765377)

[Опыт работы 1](#_Toc210765378)

[1. Сбер 1](#_Toc210765379)

[2. Prost Studio 1](#_Toc210765380)

[3. Шаблон Речи 2](#_Toc210765381)

[4. (Ожидания от команды) Моя команда мечты — 3](#_Toc210765382)

[5. (Ожидания от компании) Для меня идеальная компания — 3](#_Toc210765383)

[6. (Куда хочу развиваться) Мне интересно развиваться как бизнес-аналитику — 3](#_Toc210765384)

[Схема работы 4](#_Toc210765385)

[1. Первичное общение с менеджером и заказчиком о деталях проекта. (Понимание примерной сложности проекта и трудозатрат) 4](#_Toc210765386)

[2. Сбор требований у заказчика – актуализация ТЗ (При неполном) 4](#_Toc210765387)

[3. Анализ и уточнение требований (Выявление противоречий) 4](#_Toc210765388)

[4. Общение с командой разработки о технической стороне проекта 4](#_Toc210765389)

[5. Формализация требований 5](#_Toc210765390)

[6. Согласование и утверждение 5](#_Toc210765391)

[7. Тестирование по ТЗ (проводит тестирование по критериям приёмки, указанным в ТЗ) 5](#_Toc210765392)

[8. Актуализация ТЗ при изменениях 5](#_Toc210765393)

[9. Сдача проекта заказчику 5](#_Toc210765394)

[Требования 7](#_Toc210765395)

[1. Методы сборы требования 7](#_Toc210765396)

[a. Интервью 7](#_Toc210765397)

[b. Анкетирование / опросы 7](#_Toc210765398)

[c. Воркшопы / совместные сессии 7](#_Toc210765399)

[d. Наблюдение / shadowing 7](#_Toc210765400)

[e. Анализ документов и систем 7](#_Toc210765401)

[f. Моделирование и прототипирование 8](#_Toc210765402)

[2. Как фиксировать требования 9](#_Toc210765403)

[a. Текстовая документация 9](#_Toc210765404)

[b. Пользовательские истории (User Stories) 9](#_Toc210765405)

[c. Таблицы требований 9](#_Toc210765406)

[d. Диаграммы и схемы 9](#_Toc210765407)

[e. Связь с тестами 9](#_Toc210765408)

[3. Характеристики качественных требований 10](#_Toc210765409)

[a. Ясность (Clear / Understandable) 10](#_Toc210765410)

[b. Полнота (Complete) 10](#_Toc210765411)

[c. Согласованность (Consistent) 10](#_Toc210765412)

[d. Проверяемость / тестируемость (Verifiable / Testable) 10](#_Toc210765413)

[e. Реализуемость (Feasible / Achievable) 10](#_Toc210765414)

[f. Приоритетность (Prioritized) 11](#_Toc210765415)

[g. Устойчивость к изменениям (Stable / Modifiable) 11](#_Toc210765416)

[h. Однозначность (Unambiguous) 11](#_Toc210765417)

[Шаблоны сбора требований 12](#_Toc210765418)

[1. Первичное общение 12](#_Toc210765419)

[i. Общая информация о проекте 12](#_Toc210765420)

[j. Заказчик и заинтересованные стороны 12](#_Toc210765421)

[k. Цели и задачи проекта 12](#_Toc210765422)

[l. Технические вводные 13](#_Toc210765423)

[m. Организационные моменты 13](#_Toc210765424)

[2. Сбор требований у заказчика 13](#_Toc210765425)

[a. Общая информация 13](#_Toc210765426)

[b. Бизнес-требования 14](#_Toc210765427)

[c. Функциональные требования 14](#_Toc210765428)

[d. Нефункциональные требования 14](#_Toc210765429)

[e. Интеграции 14](#_Toc210765430)

[f. Пользователи и роли 15](#_Toc210765431)

[g. Дополнительные пожелания и ограничения 15](#_Toc210765432)

[3. Анализ и уточнение требований 15](#_Toc210765433)

[a. Общая информация 15](#_Toc210765434)

[b. Проверка требований 15](#_Toc210765435)

[c. Проверка полноты требований 16](#_Toc210765436)

[d. Проверка реализуемости и согласованности 16](#_Toc210765437)

[e. Сводная таблица по результатам анализа 17](#_Toc210765438)

[4. Общение с командой разработки о технической стороне проекта 17](#_Toc210765439)

[a. Общая информация 17](#_Toc210765440)

[b. Основные вопросы для обсуждения 17](#_Toc210765441)

[c. Результаты встречи 18](#_Toc210765442)

[5. Формализация требований 18](#_Toc210765443)

[a. Общая информация 18](#_Toc210765444)

[b. Категории требований 18](#_Toc210765445)

[c. Таблица формализованных требований 19](#_Toc210765446)

[d. Принципы формализации 19](#_Toc210765447)

[6. Согласование и утверждение 20](#_Toc210765448)

[a. Общая информация 20](#_Toc210765449)

[b. Участники согласования 20](#_Toc210765450)

[c. Проверка и согласование требований 20](#_Toc210765451)

[d. Документы и материалы для согласования 21](#_Toc210765452)

[7. Тестирование по ТЗ 21](#_Toc210765453)

[a. Общая информация 21](#_Toc210765454)

[b. Виды тестирования 21](#_Toc210765455)

[c. Матрица трассировки требований (RTM) 21](#_Toc210765456)

[d. Действия аналитика на этапе тестирования 22](#_Toc210765457)

[8. Актуализация ТЗ 23](#_Toc210765458)

[a. Общая информация 23](#_Toc210765459)

[b. Инициирование изменений 23](#_Toc210765460)

[c. Анализ и оценка влияния 23](#_Toc210765461)

[d. Согласование изменений 23](#_Toc210765462)

[n. Обновление ТЗ 23](#_Toc210765463)

[9. Сдача проектов заказчику 25](#_Toc210765464)

[a. Общая информация 25](#_Toc210765465)

[b. Инициирование изменений 25](#_Toc210765466)

[c. Анализ и оценка влияния 25](#_Toc210765467)

[d. Согласование изменений 25](#_Toc210765468)

[e. Обновление ТЗ 25](#_Toc210765469)

[Схема ТЗ 27](#_Toc210765470)

[Шаблоны ТЗ 28](#_Toc210765471)

[1. Введение 28](#_Toc210765472)

[2. Общие сведения 28](#_Toc210765473)

[3. Функциональные требования 28](#_Toc210765474)

[4. Интеграции 28](#_Toc210765475)

[5. Нефункциональные требования 29](#_Toc210765476)

[6. Сценарии использования 29](#_Toc210765477)

[7. Приложения 29](#_Toc210765478)

[Итоги ключевых этапов 31](#_Toc210765479)

[1. Первичное общение с менеджером и заказчиком 31](#_Toc210765480)

[2. Сбор требований у заказчика 31](#_Toc210765481)

[3. Анализ и уточнение требований (выявление противоречий) 31](#_Toc210765482)

[4. Общение с командой разработки о технической стороне проекта 31](#_Toc210765483)

[5. Формализация требований 31](#_Toc210765484)

[6. Согласование и утверждение 32](#_Toc210765485)

[7. Тестирование по ТЗ 32](#_Toc210765486)

[8. Актуализация ТЗ при изменениях 32](#_Toc210765487)

[9. Сдача проекта заказчику 32](#_Toc210765488)

[Метрики успеха KPI 34](#_Toc210765489)

[1. Финансовые KPI 34](#_Toc210765490)

[2. Операционные KPI 34](#_Toc210765491)

[3. Клиентские KPI 34](#_Toc210765492)

[4. Проектные / продуктовые KPI 34](#_Toc210765493)

[5. Маркетинговые / продажные KPI 35](#_Toc210765494)

[Подсказки 36](#_Toc210765495)

[1. SQL 36](#_Toc210765496)

[a. Основные команды 36](#_Toc210765497)

[b. Агрегатные функции 36](#_Toc210765498)

[c. Группировка 36](#_Toc210765499)

[d. Соединения таблиц 36](#_Toc210765500)

[e. Работа с текстом 36](#_Toc210765501)

[f. Работа с датами 37](#_Toc210765502)

[2. Базы данных 38](#_Toc210765503)

[a. Что это 38](#_Toc210765504)

[b. Основные типы баз данных 38](#_Toc210765505)

[c. Основные элементы реляционной БД 38](#_Toc210765506)

[d. Виды связей между таблицами 39](#_Toc210765507)

[3. Rest API 40](#_Toc210765508)

[a. Основные понятия 40](#_Toc210765509)

[b. Основные HTTP методы 40](#_Toc210765510)

[c. Полезные термины для БА 40](#_Toc210765511)

[4. Swagger 41](#_Toc210765512)

[a. Что такое Swagger 41](#_Toc210765513)

[b. Основные возможности 41](#_Toc210765514)

[5. Postman 42](#_Toc210765515)

[a. Что такое Postman 42](#_Toc210765516)

[b. Основные возможности 42](#_Toc210765517)

[c. Структура запроса в Postman 42](#_Toc210765518)

[6. JSON 43](#_Toc210765519)

[a. Что такое JSON 43](#_Toc210765520)

[b. Основные элементы JSON 43](#_Toc210765521)

[c. Пример JSON для заявки 43](#_Toc210765522)

[7. XML 45](#_Toc210765523)

[a. Что такое XML 45](#_Toc210765524)

[b. Основные элементы XML 45](#_Toc210765525)

[c. Пример XML для заявки 46](#_Toc210765526)

[8. Apache Kafka 47](#_Toc210765527)

[a. Что такое Apache Kafka 47](#_Toc210765528)

[b. Основные понятия 47](#_Toc210765529)

[c. Как это работает (упрощённо) 47](#_Toc210765530)

[d. Пример применения для бизнес-аналитика 48](#_Toc210765531)

[e. Что важно знать 48](#_Toc210765532)

[9. RabbitMQ 49](#_Toc210765533)

[a. Что такое RabbitMQ 49](#_Toc210765534)

[b. Основные понятия 49](#_Toc210765535)

[c. Как это работает (упрощённо) 49](#_Toc210765536)

[d. Пример для бизнес-аналитика 50](#_Toc210765537)

[e. Что важно знать бизнес-аналитику 50](#_Toc210765538)

[10. Use Case (Сценарий использования) 51](#_Toc210765539)

[a. Что это такое 51](#_Toc210765540)

[b. Зачем нужен Use Case 51](#_Toc210765541)

[c. Структура Use Case 51](#_Toc210765542)

[d. Пример Use Case 51](#_Toc210765543)

[e. Подсказки для аналитика 52](#_Toc210765544)

[f. Шаблон Use Case (для вставки в ТЗ) 52](#_Toc210765545)

[11. UML (Unified Modeling Language) 53](#_Toc210765546)

[a. Что это такое 53](#_Toc210765547)

[b. Зачем нужен UML бизнес-аналитику 53](#_Toc210765548)

[c. Основные типы UML-диаграмм (важные для БА) 53](#_Toc210765549)

[d. Какой UML использовать в работе БА 54](#_Toc210765550)

[12. BPMN VS UML 55](#_Toc210765551)

[a. Главные различия BPMN и UML 55](#_Toc210765552)

[b. Когда использовать 55](#_Toc210765553)

[c. Кратко (если спросят на собеседовании): 55](#_Toc210765554)

[13. Use Case VS User Story 56](#_Toc210765555)

[a. Различия 56](#_Toc210765556)

[b. Кратко 56](#_Toc210765557)

[14. MoSCoW 57](#_Toc210765558)

[a. Что это такое 57](#_Toc210765559)

[b. Название расшифровывается по первым буквам: 57](#_Toc210765560)

[c. Классификация требований 57](#_Toc210765561)

[d. Когда использовать MoSCoW 57](#_Toc210765562)

[e. Как применять 57](#_Toc210765563)

[15. Session ID vs JWT (JSON Web Token) 59](#_Toc210765564)

[a. Что такое Session ID 59](#_Toc210765565)

[b. Как это работает (Session ID) 59](#_Toc210765566)

[c. Особенности (Session ID) 59](#_Toc210765567)

[d. Что такое JWT (JSON Web Token) 59](#_Toc210765568)

[e. Как это работает (JWT) 59](#_Toc210765569)

[f. Разбивается на 3 части (JWT): 60](#_Toc210765570)

[g. Особенности (JWT) 60](#_Toc210765571)

[h. Кратко 60](#_Toc210765572)

[16. ТЗ VS ТС 61](#_Toc210765573)

[a. Различия 61](#_Toc210765574)

[b. Кратко 61](#_Toc210765575)

[17. Структура ТС 62](#_Toc210765576)

# Базовая информация

## **Образование**

* Профессиональное обучение в области информационных технологий и цифровой трансформации

## Места работы

1. Сбер (2021-2024)
2. Prost Studio (2024-)

# Опыт работы

## Сбер

* 1. **Команда:** Фронт (2), БЭК (3), менеджер проектов, руководитель ИТ-департамента, Бизнес Аналитик, Системный аналитик, Тестировщик
  2. **Проекты:** Полный цикл, частичный цикл.
  3. **Задачи:** Создание внутреннего продукта (Рефакторинг), добавление функционала к существующему
  4. **Стек:** Jira, MySQL, PHP, PostgreSQL, Swagger, Postman, Confluence, Miro, Visio, Word, Exel, React/Vue, RestAPI

## Prost Studio

* 1. **Команда:** Фронт (2), БЭК (2), менеджер проектов, руководитель ИТ-департамента, Бизнес Аналитик, Тестировщик, дизайнер
  2. **Проекты:** Полный цикл
  3. **Задачи:** Внедрение CRM систем, создание лендинговых страниц, создания страниц презентаций.
  4. **Стек:** Jira, MySQL, PostgreSQL, Swagger, Postman, Confluence, Miro, Visio, Word, Exel, React/Vue, RestAPI, Bitrix24

## Шаблон Речи

* Последним моим местом работы была студия веб-разработки. Данная студия занималась разработкой лендинговых страниц и внедрением crm систем для малого и иногда среднего бизнеса.
* **Наша команда состояла из:** [Prost Studio](#_Prost_Studio)
* Требования я брала напрямую от заказчика, и сама с ним общалась.
* Спринты были достаточно четким и длились 2-4 недели.
* До этого я работала в ПАО «Сбербанк», где занималась разработкой и модернизацией внутренних информационных систем. Моя работа включала как создание новых решений, так и рефакторинг существующих систем с применением современных технологий — от обновления графического интерфейса на фронтенде до внедрения инструментов оптимизации и повышения безопасности на бэкенде.
* Наша команда обычно состояла из: [Сбер](#_Сбер)
* В данной команде я также взаимодействовала с заказчиками напрямую и работала в разных департаментах и с разными командами.
* В обоих компаниях я работала, как с полным циклом, так и с неполным циклом.
* **В мои должностные обязанности входили:**

Сбор и актуализация требований

Тестирование

Постановка задач в Jira

Работа с базами данных

Автоматизация процессов

И т.д.

* **Алгоритм моей работы был такой:** [Схема работы](#_Схема_работы)
* **Для каждого пункта я использовала свои шаблоны и вот некоторые из них:** [Шаблоны сбора требований](#_Шаблоны_сбора_требований)
* **Я использовала разные методы для сбора требований, такие как:** [Методы сборы требования](#_Методы_сборы_требования)
* **И фиксировала их с помощью**: [Как фиксировать требования](#_Как_фиксировать_требования)
* **Я старалась, чтобы мои требования соответствовали качественным критериям, таким как**: [Характеристики качественных требований](#_Характеристики_качественных_требова)
* **В моё ТЗ входили следующие пункты:** [Схема ТЗ](#_Схема_ТЗ)
* **Каждый пункт ТЗ я собирала, опираясь на данные шаблоны:** [Шаблоны ТЗ](#_Шаблоны_ТЗ)
* **Я использовала данные метрики успеха (KPI):** [Метрики успеха KPI](#_Метрики_успеха_KPI)
* **Итогами ключевых этапов являются:** [Итоги ключевых этапов](#_Итоги_ключевых_этапов)
* **У меня был опыт с различными программами и технологиями, такими как:** [Подсказки](#_Подсказки)

## (Ожидания от команды) Моя команда мечты —

Это коллектив, в котором каждый может свободно задать вопрос, открыто обсудить любые сложности, связанные с проектом, и всегда готов помочь коллегам в трудных ситуациях.

## (Ожидания от компании) Для меня идеальная компания —

Это та, чей продукт приносит пользу людям, стремится создавать для заказчиков максимально качественные решения, несмотря на существующие ограничения, а также поддерживает профессиональный рост и развитие своих сотрудников.

## (Куда хочу развиваться) Мне интересно развиваться как бизнес-аналитику —

И участвовать в реализации сложных проектов, где важно глубоко понимать как бизнес-логику, так и техническую сторону решений.

# Схема работы

В мою работу в основном входили данные шаги для разработки проекта полного и неполного цикла:

## Первичное общение с менеджером и заказчиком о деталях проекта. (Понимание примерной сложности проекта и трудозатрат)

* 1. Цель проекта
  2. Аудитория проекта
  3. Сроки сдачи проекта
  4. Текущая ситуация – AS IS (Есть ли старые системы или проект полностью новый, есть ли тестовые решение и т.д.)
  5. Риски и ожидания
  6. Сроки, бюджет и приоритет
  7. Технические детали и ограничения

## Сбор требований у заказчика – актуализация ТЗ (При неполном)

* 1. Цели и задачи
  2. Роли
  3. Функциональные требования
  4. Нефункциональные требования
  5. Бизнес-процессы
  6. Данные и интеграции
  7. Тестирование и приёмка
  8. Ограничения, риски и пожелания

## Анализ и уточнение требований (Выявление противоречий)

* 1. Проверка логики и непротиворечивости
  2. Проверка полноты и точности
  3. Проверка приоритетов и необходимости
  4. Проверка согласованности между участниками
  5. Проверка реализуемости
  6. Проверка формулировки (ясность и тестируемость)

## Общение с командой разработки о технической стороне проекта

* 1. Архитектура и технология
  2. Интеграции и обмен данными
  3. Структура и хранение данных
  4. Производительность и нагрузка
  5. Безопасность и доступы
  6. Тестирование и развертывание
  7. Риски и ограничения

**Заметки после встречи:**

* Комментарии разработчиков к требованиям (реализуемость, риски, альтернативы).
* Список технических ограничений.
* Решения по архитектуре и технологиям.
* Обновлённые оценки сроков и приоритетов.
* Пункты, требующие уточнения с заказчиком.

## Формализация требований

## Согласование и утверждение

* 1. Подготовка итогового пакета документов
  2. Внутреннее согласование
  3. Согласование с заказчиком
  4. Утверждение

## Тестирование по ТЗ (проводит тестирование по критериям приёмки, указанным в ТЗ)

|  |  |  |
| --- | --- | --- |
| Функциональное | Проверить соответствие реализованного функционала требованиям ТЗ | QA / аналитик |
| Интеграционное | Проверить корректность взаимодействия между модулями и системами | QA / разработчики |
| UI/UX-тестирование | Проверить соответствие интерфейсов прототипам и удобству использования | QA / аналитик / заказчик |
| Регрессионное | Убедиться, что изменения не нарушили уже работающий функционал | QA |
| Приёмочное (UAT) | Проверить готовность системы заказчиком по ключевым бизнес-сценариям | Заказчик / аналитик |

## Актуализация ТЗ при изменениях

* 1. Инициирование изменения
  2. Анализ и оценка
  3. Согласование
  4. Актуализация документации
  5. Коммуникация и контроль

## Сдача проекта заказчику

* 1. Подготовка к сдаче
  2. Передача пакета документов
  3. Демонстрация и приёмка
  4. Оформление сдачи

# Требования

## Методы сборы требования

### **Интерв**ью

* + **Описание:** личные встречи с заказчиком, пользователями или экспертами.
  + **Когда применять:** для получения глубокого понимания целей и проблем.
  + **Плюсы:** можно уточнять детали, гибко реагировать на ответы.
  + **Минусы:** трудозатратно, может быть субъективным.

### Анкетирование / опросы

* + **Описание:** стандартные вопросы, анкеты или онлайн-формы.
  + **Когда применять:** когда нужно быстро собрать мнение большого числа людей.
  + **Плюсы:** масштабируемо, легко анализировать результаты.
  + **Минусы:** ограниченная детализация, риск неполных ответов.

### Воркшопы / совместные сессии

* + **Описание:** встречи с группой заинтересованных лиц для обсуждения требований и решений.
  + **Когда применять:** при сложных проектах с множеством участников.
  + **Плюсы:** стимулирует генерацию идей, позволяет согласовать мнения.
  + **Минусы:** требует модератора и хорошей подготовки.

### Наблюдение / shadowing

* + **Описание:** наблюдение за пользователями при выполнении текущих процессов.
  + **Когда применять:** для выявления скрытых проблем и реальных бизнес-процессов.
  + **Плюсы:** реальные данные о процессах, практический опыт.
  + **Минусы:** занимает время, пользователи могут вести себя иначе под наблюдением.

### Анализ документов и систем

* + **Описание:** изучение существующей документации, спецификаций, регламентов, баз данных.
  + **Когда применять:** если система уже работает или есть стандартизированные процессы.
  + **Плюсы:** экономит время, позволяет увидеть несоответствия между документацией и реальностью.
  + **Минусы:** не всегда отражает текущую практику.

### Моделирование и прототипирование

* + **Описание:** создание схем процессов, прототипов интерфейсов, макетов страниц.
  + **Когда применять:** когда нужно визуализировать требования для согласования.
  + **Плюсы:** облегчает понимание для всех участников проекта.
  + **Минусы:** требует навыков моделирования.

## Как фиксировать требования

### Текстовая документация

* + **Что фиксируем:** подробные описания функционала, ограничения, бизнес-правила.
  + **Форматы:** Word, Confluence, Google Docs.

### Пользовательские истории (User Stories)

* + **Формат:** “Как [роль] я хочу [действие], чтобы [результат]”.
  + **Пример:** “Как администратор, я хочу блокировать пользователя, чтобы ограничить доступ при нарушении правил”.
  + **Плюсы:** компактно, ориентировано на пользователя, легко переносится в Agile-системы (Jira, Trello).

### Таблицы требований

* + **Что фиксируем:** список требований с атрибутами: ID, описание, приоритет, источник, статус.
  + **Формат:** Excel, Google Sheets, Airtable.
  + **Плюсы:** удобно отслеживать изменения, фильтровать по приоритетам, связывать с тест-кейсами.

### Диаграммы и схемы

* + **BPMN:** бизнес-процессы.
  + **UML:** последовательности, классы, варианты использования.
  + **Прототипы:** Figma, Balsamiq для визуализации интерфейсов.
  + **Плюсы:** наглядно, помогает уточнять требования с заказчиком и командой.

### Связь с тестами

* + **Описание:** каждому требованию сопоставляются тест-кейсы (TestRail, Zephyr).
  + **Зачем:** чтобы убедиться, что требование реализовано и проверено.

## Характеристики качественных требований

**Качественные требования** — это такие требования, которые правильно сформулированы и пригодны для реализации и тестирования. Чтобы требования были качественными, они должны обладать рядом характеристик.

**Основные характеристики качественных требований:**

### Ясность (Clear / Understandable)

Требование должно быть однозначным и легко понимаемым для всех участников проекта (разработчиков, заказчиков, тестировщиков).

* + Пример: «Система должна отправлять уведомление пользователю после регистрации» вместо «Система должна уведомлять».

### Полнота (Complete)

Все аспекты функционала или ограничений должны быть описаны.

Пример: если речь о форме регистрации — указаны все поля, правила валидации, сообщения об ошибках.

### Согласованность (Consistent)

Требования не должны противоречить друг другу.

Пример: одно требование говорит «страница загружается за 1 секунду», другое — «страница может загружаться 5 секунд» — противоречие.

### Проверяемость / тестируемость (Verifiable / Testable)

Требование должно быть измеримым или проверяемым.

Пример: «Система должна обрабатывать 1000 заказов в минуту» вместо «Система должна работать быстро».

### Реализуемость (Feasible / Achievable)

Требование должно быть технически и финансово выполнимо в рамках проекта.

### Приоритетность (Prioritized)

Требования должны быть оценены по важности для бизнеса.

Пример: критичные, важные, желательные.

### Устойчивость к изменениям (Stable / Modifiable)

Требование должно быть легко изменяемым при корректировках без разрушения всей документации.

### Однозначность (Unambiguous)

Не должно быть двусмысленных формулировок.

Пример: «быстро» → «время отклика ≤ 2 секунд».

# Шаблоны сбора требований

## Первичное общение

### Общая информация о проекте

|  |  |  |
| --- | --- | --- |
| Пункт | Вопрос / описание | Ответ / комментарий |
| Название проекта | Как называется проект / продукт? |  |
| Цель проекта | Зачем создаётся проект? Какую задачу решает бизнес? |  |
| Основная идея / суть решения | В чём заключается основная функция или ценность продукта? |  |
| Ожидаемый результат | Что заказчик считает успешной реализацией проекта? |  |
| Статус проекта | Новый проект / редизайн / модернизация существующего решения |  |

### Заказчик и заинтересованные стороны

|  |  |  |
| --- | --- | --- |
| Пункт | Вопрос / описание | Ответ / комментарий |
| Основной заказчик / инициатор | Кто принимает финальные решения по проекту? |  |
| Контактные лица | Кто будет вовлечён (менеджер, маркетолог, IT, юрист и др.)? |  |
| Основные пользователи системы | Кто будет пользоваться продуктом (клиенты, сотрудники и т.п.)? |  |
| Ответственные за согласования | Кто утверждает требования и принимает ТЗ? |  |

### Цели и задачи проекта

|  |  |  |
| --- | --- | --- |
| Пункт | Вопрос / описание | Ответ / комментарий |
| Основные цели | Какие бизнес-проблемы должен решить проект? |  |
| Ключевые функции | Что точно должно быть реализовано в первой версии (MVP)? |  |
| Дополнительные пожелания | Какие функции “желательны”, но не критичны? |  |
| KPI / Метрики успеха | По каким показателям заказчик будет оценивать результат? |  |

### Технические вводные

|  |  |  |
| --- | --- | --- |
| Пункт | Вопрос / описание | Ответ / комментарий |
| Планируемая платформа / стек | На чём планируется реализация (CMS, кастом, CRM, мобильное приложение и т.д.)? |  |
| Есть ли существующая система / сайт | Нужна ли миграция данных, доработка или интеграция со старой системой? |  |
| Интеграции | Есть ли внешние системы (CRM, ERP, платёжки, мессенджеры)? |  |
| Безопасность / доступы | Есть ли особые требования по хранению данных, авторизации, ролям? |  |

### Организационные моменты

|  |  |  |
| --- | --- | --- |
| Пункт | Вопрос / описание | Ответ / комментарий |
| Сроки | Есть ли желаемые или фиксированные сроки запуска? |  |
| Бюджет | Есть ли ограничения по бюджету (ориентир)? |  |
| Формат взаимодействия | Как планируется коммуникация (митинги, мессенджеры, Confluence и т.д.)? |  |
| Частота встреч | Как часто нужно проводить синхронизации / отчёты? |  |
| Дополнительные материалы | Есть ли уже документация, макеты, прототипы, описание бизнес-процессов? |  |

## Сбор требований у заказчика

### Общая информация

|  |  |  |
| --- | --- | --- |
| Пункт | Вопрос / описание | Ответ / комментарий |
| Название проекта |  |  |
| Дата и участники встречи |  |  |
| Цель встречи |  |  |
| Версия документа |  |  |

### Бизнес-требования

|  |  |  |
| --- | --- | --- |
| № | Вопрос / описание | Ответ заказчика / комментарий |
| 1 | Какую основную бизнес-проблему решает проект? |  |
| 2 | Какие ключевые цели и результаты ожидаются? |  |
| 3 | Кто конечный пользователь решения? |  |
| 4 | Какие бизнес-процессы затрагивает проект? |  |
| 5 | Есть ли существующие процессы / решения, которые нужно улучшить? |  |
| 6 | Какие метрики успеха (KPI) планируется достичь? |  |

### Функциональные требования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание функции / сценария | Подробности / ожидания заказчика | Приоритет (H/M/L) |
| 1 | Регистрация и авторизация пользователей | Возможность авторизации через email и Google | H |
| 2 | Личный кабинет | Просмотр и редактирование профиля, истории заказов | H |
| 3 | Фильтр / поиск | Поиск по названию, дате, категории | M |
| 4 | Система уведомлений | Email-уведомления о статусе заказа | M |
| 5 | Отчётность / статистика | Ежемесячные отчёты в PDF | L |

### Нефункциональные требования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Категория | Требование | Комментарий |
| 1 | Производительность | Загрузка страниц ≤ 2 сек |  |
| 2 | Безопасность | Хранение данных с шифрованием |  |
| 3 | Совместимость | Поддержка мобильных и десктопных устройств |  |
| 4 | Доступность | Система работает 24/7, время простоя ≤ 1% |  |
| 5 | Юзабилити | Интерфейс понятен без обучения |  |

### Интеграции

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Система / сервис | Описание интеграции | Формат / API | Комментарий |
| 1 | CRM | Передача данных о клиентах | REST / JSON |  |
| 2 | Платёжный шлюз | Оплата онлайн | API банка |  |
| 3 | Telegram / WhatsApp | Уведомления и чат с клиентами | Bot API |  |

### Пользователи и роли

|  |  |  |  |
| --- | --- | --- | --- |
| № | Роль пользователя | Доступные функции | Ограничения |
| 1 | Администратор | Управление пользователями, настройками, отчётами | Полный доступ |
| 2 | Менеджер | Управление заказами и клиентами | Без доступа к настройкам |
| 3 | Клиент | Создание, просмотр и редактирование заявок | Только свои данные |

### Дополнительные пожелания и ограничения

|  |  |  |
| --- | --- | --- |
| Категория | Вопрос / уточнение | Ответ заказчика / комментарий |
| UI/UX | Есть ли референсы, примеры дизайна, пожелания по стилю? |  |
| Технические | Есть ли предпочтения по CMS, языкам, хостингу? |  |
| Правовые | Нужно ли согласовать политику конфиденциальности / оферту? |  |
| Сроки и этапы | Когда планируется MVP / запуск проекта? |  |

## Анализ и уточнение требований

### Общая информация

|  |  |  |
| --- | --- | --- |
| Пункт | Описание | Значение |
| Название проекта |  |  |
| Дата анализа |  |  |
| Ответственный аналитик |  |  |
| Версия документа |  |  |

### Проверка требований

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № | ID требования | Краткое описание | Источник (кто озвучил) | Противоречие / дублирование | Комментарий аналитика | Решение / действие | Статус |
| 1 | FR-001 | Пользователь может редактировать заявку после отправки | Менеджер | Противоречит FR-005 (заявка должна быть зафиксирована после отправки) | Нужно уточнить бизнес-логику | Обсудить с заказчиком | В работе |
| 2 | FR-004 | Система автоматически отправляет письма клиентам | Руководитель | Может противоречить политике безопасности (запрет внешних писем) | Проверить с IT и заказчиком | На согласовании |  |
| 3 | NFR-002 | Система загружается ≤ 1 сек | Заказчик | Нереалистично для текущего сервера | Уточнить параметры производительности | Уточнить с разработкой |  |
| 4 | FR-010 | Фильтр заказов по дате и клиенту | Интервью заказчика | Дублирует FR-011 (фильтр по всем полям) | Объединить | Выполнено | Закрыто |

### Проверка полноты требований

|  |  |  |
| --- | --- | --- |
| Категория | Вопрос | Статус / комментарий |
| Бизнес-логика | Все ли процессы покрыты? |  |
| Пользователи и роли | Все ли сценарии учтены для каждой роли? |  |
| Валидации и исключения | Определены ли все ограничения и ошибки? |  |
| Интеграции | Все ли системы и форматы обмена уточнены? |  |
| UI / UX | Есть ли макеты или референсы? |  |
| Тестируемость | Можно ли проверить выполнение каждого требования? |  |

### Проверка реализуемости и согласованности

|  |  |
| --- | --- |
| Вопрос | Ответ / комментарий |
| Есть ли технические ограничения, влияющие на реализацию требований? |  |
| Совпадает ли бизнес-логика с текущими процессами компании? |  |
| Есть ли требования, выходящие за рамки MVP? |  |
| Понимают ли все стороны требование одинаково? |  |
| Есть ли риски несоответствия между фронтендом и бэкендом? |  |

### Сводная таблица по результатам анализа

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Категория | Кол-во требований | Найдено противоречий | Исправлено | На согласовании | Отклонено |
| Функциональные |  |  |  |  |  |
| Нефункциональные |  |  |  |  |  |
| Интеграции |  |  |  |  |  |
| Итого |  |  |  |  |  |

## Общение с командой разработки о технической стороне проекта

### Общая информация

|  |  |  |
| --- | --- | --- |
| Пункт | Описание | Значение |
| Название проекта |  |  |
| Дата встречи / обсуждения |  |  |
| Участники (роли) | Аналитик, тимлид, backend, frontend, devops, QA и др. |  |
| Версия требований / документа |  |  |
| Цель обсуждения | Согласовать техническую реализуемость и определить архитектурные решения |  |

### Основные вопросы для обсуждения

|  |  |  |
| --- | --- | --- |
| Направление | Пример вопросов | Комментарии / ответы команды |
| **Архитектура** | Какая будет архитектура системы (монолит, микросервисы, клиент-сервер)? |  |
| **Технологии** | Какие языки, фреймворки, БД, CMS или библиотеки планируется использовать? |  |
| **Интеграции** | Какие внешние системы и API нужны? Есть ли готовые спецификации / ограничения? |  |
| **Безопасность** | Как будут храниться персональные данные? Есть ли требования по шифрованию и авторизации? |  |
| **UI / UX** | Возможны ли предложенные интерфейсные решения с выбранным стеком? |  |
| **Производительность** | Есть ли ограничения по времени отклика, объёму данных, количеству пользователей? |  |
| **Развёртывание и инфраструктура** | Где будет хостинг (cloud / on-premise)? Кто отвечает за CI/CD? |  |
| **Риски** | Есть ли потенциальные узкие места или риски по срокам / технологиям? |  |
| **Тестирование** | Что нужно от аналитика для подготовки тестовых сценариев и RTM? |  |
| **Оценка сроков** | Требуется ли дополнительная детализация задач в Jira / Trello перед оценкой? |  |

### Результаты встречи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Вопрос / тема | Решение / договорённость | Ответственный | Дедлайн |
| 1 | Выбор базы данных | PostgreSQL | Тимлид backend | 10.10.2025 |
| 2 | API для интеграции с CRM | REST API (JSON), описать структуру | Аналитик | 12.10.2025 |
| 3 | Деплой и тестовая среда | Docker + staging | DevOps | 15.10.2025 |
| 4 | Обработка ошибок при интеграции | Добавить коды ошибок в спецификацию |  |  |

## Формализация требований

### Общая информация

|  |  |  |
| --- | --- | --- |
| Пункт | Описание | Значение |
| Название проекта |  |  |
| Дата формализации |  |  |
| Ответственный аналитик |  |  |
| Версия документа |  |  |
| Цель этапа | Преобразовать собранные и уточнённые требования в формализованный вид для ТЗ |  |

### Категории требований

|  |  |  |
| --- | --- | --- |
| Категория | Пример содержания | Статус |
| **Бизнес-требования (BR)** | Цели проекта, ожидаемые результаты, показатели успеха |  |
| **Функциональные требования (FR)** | Что система должна делать: сценарии, логика, поведение |  |
| **Нефункциональные требования (NFR)** | Скорость, безопасность, производительность, доступность |  |
| **Интерфейсные требования (UI/UX)** | Макеты, прототипы, навигация, юзабилити |  |
| **Интеграционные требования (IR)** | Обмен данными с другими системами, API, форматы |  |
| **Ролевые / пользовательские сценарии (UC)** | Поведение пользователей в системе |  |
| **Ограничения и допущения** | Технические, организационные, правовые ограничения |  |

### Таблица формализованных требований

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Тип | Название / краткое описание | Подробное описание | Критерий приёмки | Приоритет | Источник | Статус |
| BR-001 | Бизнес | Оптимизация обработки заявок | Сократить время обработки до 3 мин | Время < 3 мин в 95% случаев | Высокий | Заказчик | Утверждено |
| FR-004 | Функционал | Создание новой заявки | Пользователь может создать заявку через форму | Форма сохраняет данные и создаёт запись в БД | Высокий | Интервью | В работе |
| NFR-002 | Нефункционал | Скорость загрузки | Главная страница загружается ≤ 2 сек | Проверка через Lighthouse | Средний | Команда | Утверждено |
| UI-005 | UI/UX | Адаптивный дизайн | Корректное отображение на экранах от 360px | Проверка вручную | Средний | Заказчик | В работе |

### Принципы формализации

**Каждое требование должно быть:**

* + - **Однозначным** — не допускающим двойного толкования.
    - **Измеримым** — иметь критерий проверки.
    - **Проверяемым** — можно протестировать.
    - **Не пересекающимся** — не дублировать другие требования.
    - **Прослеживаемым** — иметь источник (кто сформулировал).
    - **Приоритизированным** — оценка важности (High / Medium / Low).

## Согласование и утверждение

### Общая информация

|  |  |  |
| --- | --- | --- |
| Пункт | Описание | Значение |
| Название проекта |  |  |
| Версия ТЗ |  |  |
| Дата согласования |  |  |
| Ответственный аналитик |  |  |
| Цель этапа | Зафиксировать финальную версию требований и получить официальное утверждение |  |

### Участники согласования

|  |  |  |  |
| --- | --- | --- | --- |
| Роль | Имя / ФИО | Компания / отдел | Комментарий / ответственность |
| Заказчик |  |  | Утверждение бизнес-требований |
| Менеджер проекта |  |  | Контроль сроков и бюджета |
| Тимлид / архитектор |  |  | Согласование технической реализуемости |
| QA / Тестировщик |  |  | Проверка тестируемости требований |
| Аналитик |  |  | Фиксирует согласование и обновляет ТЗ |

### Проверка и согласование требований

|  |  |  |  |
| --- | --- | --- | --- |
| ID требования | Описание | Ответ / комментарий | Решение / статус |
| BR-001 | Оптимизация обработки заявок | Заказчик согласен | Утверждено |
| FR-004 | Создание новой заявки | Требуется уточнение по формату полей | На согласовании |
| NFR-002 | Скорость загрузки ≤ 2 сек | Согласовано с командой | Утверждено |
| UI-005 | Адаптивный дизайн | Команда QA подтвердила тестируемость | Утверждено |

### Документы и материалы для согласования

* + Финальная версия ТЗ (с формализованными требованиями).
  + Диаграммы процессов, макеты, прототипы.
  + Матрица трассировки требований (RTM).
  + Журнал изменений (Change Log).
  + План тестирования / UAT.

## Тестирование по ТЗ

### Общая информация

|  |  |  |
| --- | --- | --- |
| Пункт | Описание | Значение |
| Название проекта |  |  |
| Версия ТЗ |  |  |
| Дата тестирования |  |  |
| Ответственный аналитик / QA |  |  |
| Цель этапа | Проверить, что система соответствует утверждённому ТЗ и готова к эксплуатации |  |

### Виды тестирования

|  |  |  |
| --- | --- | --- |
| Тип | Цель | Кто выполняет |
| **Функциональное** | Проверка соответствия реализованного функционала требованиям | QA / аналитик |
| **Интеграционное** | Проверка взаимодействия модулей и внешних систем | QA / разработка |
| **UI/UX-тестирование** | Проверка интерфейсов на соответствие макетам и удобство | QA / аналитик / заказчик |
| **Регрессионное** | Проверка, что изменения не нарушили существующий функционал | QA |
| **Приёмочное (UAT)** | Проверка готовности системы по ключевым бизнес-сценариям | Заказчик / аналитик |

### Матрица трассировки требований (RTM)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID требования | Описание требования | ID теста | Описание теста | Статус | Результат |
| BR-001 | Оптимизация обработки заявок | TC-001 | Создание и обработка заявки в ≤3 мин | В работе |  |
| FR-004 | Создание новой заявки | TC-002 | Заполнение формы и сохранение данных | Пройдено | Успешно |
| NFR-002 | Скорость загрузки ≤2 сек | TC-003 | Проверка через Lighthouse | Пройдено | Успешно |
| UI-005 | Адаптивный дизайн | TC-004 | Просмотр на разных устройствах | В раб |  |

### Действия аналитика на этапе тестирования

* + Проверка соответствия каждого требования ТЗ тестам.
  + Участие в подготовке и проверке RTM.
  + Совместное проведение UAT с заказчиком.
  + Фиксация выявленных дефектов и уточнение спорных моментов.
  + Подтверждение исправлений после тестирования.

## Актуализация ТЗ

### Общая информация

|  |  |  |
| --- | --- | --- |
| Пункт | Описание | Значение |
| Название проекта |  |  |
| Версия ТЗ |  |  |
| Дата актуализации |  |  |
| Ответственный аналитик |  |  |
| Цель этапа | Обновить ТЗ с учётом изменений и получить согласование |  |

### Инициирование изменений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | ID изменения (CR) | Дата | Инициатор | Суть изменения | Причина / комментарий |
| 1 | CR-001 | 07.10.2025 | Заказчик | Добавить авторизацию через Google | Удобство для пользователей |
| 2 | CR-002 | 12.10.2025 | Разработка | Упростить фильтр заказов | Техническое ограничение API |

### Анализ и оценка влияния

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID изменения | Влияние на сроки | Влияние на бюджет | Влияние на архитектуру / интеграции | Риски | Рекомендация |
| CR-001 | +3 дня | Нет | Нужно доработать модуль авторизации | Средний | Принять |
| CR-002 | 0 | Нет | Не влияет | Низкий | Принять |

### Согласование изменений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID изменения | Участники согласования | Решение | Дата согласования | Статус |
| CR-001 | Заказчик, Аналитик, Тимлид | Принято | 08.10.2025 | Утверждено |
| CR-002 | Аналитик, Разработка | Принято | 13.10.2025 | Утверждено |

### Обновление ТЗ

* + Внесение изменений в текст ТЗ, диаграммы, макеты, RTM.
  + Обновление версии документа (например: ТЗ\_v1.2 от 07.10.2025).
  + Информирование всех участников проекта о новой версии.

## Сдача проектов заказчику

### Общая информация

|  |  |  |
| --- | --- | --- |
| Пункт | Описание | Значение |
| Название проекта |  |  |
| Версия ТЗ |  |  |
| Дата актуализации |  |  |
| Ответственный аналитик |  |  |
| Цель этапа | Обновить ТЗ с учётом изменений и получить согласование |  |

### Инициирование изменений

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | ID изменения (CR) | Дата | Инициатор | Суть изменения | Причина / комментарий |
| 1 | CR-001 | 07.10.2025 | Заказчик | Добавить авторизацию через Google | Удобство для пользователей |
| 2 | CR-002 | 12.10.2025 | Разработка | Упростить фильтр заказов | Техническое ограничение API |

### Анализ и оценка влияния

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID изменения | Влияние на сроки | Влияние на бюджет | Влияние на архитектуру / интеграции | Риски | Рекомендация |
| CR-001 | +3 дня | Нет | Нужно доработать модуль авторизации | Средний | Принять |
| CR-002 | 0 | Нет | Не влияет | Низкий | Принять |

### Согласование изменений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID изменения | Участники согласования | Решение | Дата согласования | Статус |
| CR-001 | Заказчик, Аналитик, Тимлид | Принято | 08.10.2025 | Утверждено |
| CR-002 | Аналитик, Разработка | Принято | 13.10.2025 | Утверждено |

### Обновление ТЗ

* + Внесение изменений в текст ТЗ, диаграммы, макеты, RTM.
  + Обновление версии документа (например: ТЗ\_v1.2 от 07.10.2025).
  + Информирование всех участников проекта о новой версии.

# Схема ТЗ

Основные пункты ТЗ:

* **Введение** – цели и краткое описание проекта.
* **Общие сведения** – роли участников, используемые внешние сервисы, стек технологий.
* **Функциональные требования** – детальное описание полей форм, обработки запросов и ошибок.
* **Интеграции** – описание взаимодействия с внешними сервисами и API.
* **Нефункциональные требования** – требования к производительности, безопасности и адаптивности.
* **Сценарии использования** – примеры работы пользователей с системой.
* **Приложения** – схемы, макеты интерфейсов, диаграммы, ссылки на прототипы в Figma и старый интерфейс, если он есть.

# Шаблоны ТЗ

## Введение

**Цель:** дать краткое описание проекта и объяснить его назначение.

|  |  |
| --- | --- |
| Пункт | Описание |
| Название проекта |  |
| Цель проекта |  |
| Краткое описание |  |
| Основные задачи |  |
| Ожидаемый результат |  |
| Аудитория / пользователи |  |

## Общие сведения

**Цель:** указать участников, внешние сервисы и используемые технологии.

|  |  |
| --- | --- |
| Пункт | Описание |
| Роли участников | Заказчик, аналитик, разработчики, QA, менеджер проекта |
| Внешние сервисы | CRM, платёжные системы, мессенджеры, API сторонних сервисов |
| Стек технологий | Frontend, Backend, БД, хостинг, фреймворки, CMS |
| Ограничения / допущения | Технические, организационные, юридические |

## Функциональные требования

**Цель:** подробно описать все функции системы, поля форм, обработку запросов и ошибок.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Функция / модуль | Подробное описание | Поля формы | Обработка ошибок | Приоритет | Критерий приёмки |
| FR-001 | Регистрация пользователя | Пользователь может зарегистрироваться через email или Google | Email, пароль, имя | Проверка формата email, длины пароля | Высокий | Успешная регистрация и запись в БД |
| FR-002 | Создание заявки | Пользователь создаёт заявку на услугу | Название услуги, дата, комментарий | Проверка обязательных полей | Высокий | Заявка сохраняется в БД, уведомление отправляется |

## Интеграции

**Цель:** описать взаимодействие с внешними системами и API.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Сервис / система | Тип интеграции | Формат данных | Частота / триггер | Комментарий |
| 1 | CRM | Передача данных о клиентах | JSON / REST API | При создании заявки |  |
| 2 | Платёжный шлюз | Оплата онлайн | API банка | По запросу пользователя |  |
| 3 | Telegram | Уведомления | Bot API | По статусу заявки |  |

## Нефункциональные требования

**Цель:** указать требования к производительности, безопасности и адаптивности.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Категория | Требование | Критерий проверки |
| 1 | Производительность | Загрузка страниц ≤ 2 сек | Lighthouse / тестирование скорости |
| 2 | Безопасность | Хранение данных с шифрованием, авторизация через токены | Аудит безопасности |
| 3 | Адаптивность | Поддержка экранов от 360px до 1920px | Тестирование на разных устройствах |
| 4 | Доступность | Работа системы 24/7, время простоя ≤ 1% | Мониторинг uptime |

## Сценарии использования

**Цель:** показать примеры взаимодействия пользователей с системой.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Роль пользователя | Сценарий | Ожидаемый результат |
| UC-001 | Клиент | Регистрация через email | Успешная регистрация и вход в систему |
| UC-002 | Клиент | Создание заявки | Заявка сохраняется, уведомление отправлено |
| UC-003 | Менеджер | Просмотр заявок | Список заявок отображается с фильтрами |

## Приложения

**Цель:** предоставить дополнительные материалы для понимания и реализации проекта.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тип приложения | Ссылка / файл | Комментарий |
| 1 | Схема процессов | diagram.png | Основные бизнес-процессы |
| 2 | Макеты интерфейсов | figma\_link | Figma прототипы |
| 3 | Диаграммы базы данных | db\_schema.png | Структура таблиц |
| 4 | Старый интерфейс | old\_ui.pdf | Если есть существующая система |

# Итоги ключевых этапов

## 1. Первичное общение с менеджером и заказчиком

* Понимание цели проекта, бизнес-проблемы и ожидаемого результата.
* Определение основных участников и заинтересованных лиц.
* Предварительная оценка сложности, трудозатрат и сроков.
* Зафиксированы ключевые вводные для сбора требований.

## 2. Сбор требований у заказчика

* Зафиксированы бизнес-требования: цели, KPI, сценарии пользователей.
* Определены функциональные требования с приоритетами.
* Выявлены нефункциональные требования: производительность, безопасность, UI/UX.
* Составлен список интеграций, ролей и ограничений.
* Получены дополнительные материалы: макеты, прототипы, документация.

## 3. Анализ и уточнение требований (выявление противоречий)

* Проверены требования на полноту и согласованность.
* Выявлены противоречия, дублирования и неясные моменты.
* Определены риски и потенциальные узкие места.
* Подготовлены рекомендации по корректировке требований.
* Обновлён реестр требований с пометкой статусов (утверждено / уточняется).

## 4. Общение с командой разработки о технической стороне проекта

* Согласована техническая реализуемость требований.
* Определён стек технологий, архитектура и интеграции.
* Уточнены ограничения, риски и особенности тестирования.
* Зафиксированы договорённости по срокам и ответственным за модули.
* Обновлено ТЗ с учётом технических комментариев команды.

## 5. Формализация требований

* Все требования переведены в формализованный вид с уникальными ID.
* Установлены критерии приёмки для каждого требования.
* Требования структурированы по категориям (BR, FR, NFR, UI/UX, IR, UC).
* Приоритет каждого требования определён (High / Medium / Low).
* ТЗ готово к согласованию и утверждению.

## 6. Согласование и утверждение

* Финальная версия ТЗ проверена и согласована с заказчиком и командой.
* Получено официальное утверждение (подпись / электронное подтверждение).
* Утверждены приоритеты, критерии приёмки и версии документа.
* Зафиксирована версия ТЗ (например: ТЗ\_v1.0\_final).

## 7. Тестирование по ТЗ

* Подготовлены тестовые сценарии (Test Cases) по каждому требованию.
* Проведено функциональное, интеграционное, UI/UX и приёмочное тестирование (UAT).
* Составлена матрица трассировки требований (RTM).
* Зафиксированы выявленные дефекты и статус исправлений.
* Получено подтверждение заказчика о соответствии системы ТЗ.

## 8. Актуализация ТЗ при изменениях

* Зафиксированы все изменения требований (Change Request).
* Оценено влияние изменений на сроки, бюджет и архитектуру.
* Обновлено ТЗ, диаграммы, RTM и журнал изменений (Change Log).
* Согласованы новые версии требований с заказчиком и командой.
* Обеспечена прозрачность версий и минимизация конфликтов.

## 9. Сдача проекта заказчику

* + Проведена демонстрация готового продукта.
  + Проверены все ключевые функциональные сценарии и исправлены критические ошибки.
  + Переданы все документы: финальное ТЗ, RTM, Change Log, инструкции пользователя.
  + Подписан акт приёмки / получено электронное подтверждение сдачи.
  + Завершение проекта или переход к гарантийному / поддерживающему периоду.

# Метрики успеха KPI

## ****Финансовые KPI****

Показывают экономическую эффективность бизнеса или проекта.

|  |  |  |
| --- | --- | --- |
| KPI | Описание | Пример |
| Выручка | Общая сумма дохода за период | 1 млн ₽ за месяц |
| Прибыль | Разница между доходами и расходами | 250 тыс ₽ чистой прибыли |
| ROI (окупаемость инвестиций) | Соотношение прибыли к вложениям | 150% ROI на маркетинговую кампанию |
| Себестоимость / маржинальность | Доля затрат от дохода | 40% себестоимость, 60% маржа |

## ****Операционные KPI****

Показывают эффективность процессов и работу команды.

|  |  |  |
| --- | --- | --- |
| KPI | Описание | Пример |
| Время выполнения задачи | Среднее время выполнения ключевой задачи | 2 часа на обработку заявки |
| Производительность сотрудников | Кол-во выполненных операций за период | 50 заявок в день |
| Количество ошибок / багов | Контроль качества работы | <2 багов на релиз |
| Заполнение данных | Корректность и полнота информации в системе | 95% полей формы заполнены |

## ****Клиентские KPI****

Показывают, насколько клиенты довольны продуктом или сервисом.

|  |  |  |
| --- | --- | --- |
| KPI | Описание | Пример |
| NPS (Net Promoter Score) | Индекс готовности рекомендовать продукт | NPS = 70% |
| CSAT (Customer Satisfaction) | Удовлетворённость клиентов | CSAT = 4,5/5 |
| Количество повторных покупок | Лояльность клиентов | 30% клиентов вернулись за месяц |
| Время отклика | Среднее время реакции на запрос клиента | ≤ 15 минут |

## ****Проектные / продуктовые KPI****

Показывают успех проекта и качество продукта.

|  |  |  |
| --- | --- | --- |
| KPI | Описание | Пример |
| Выполнение задач по ТЗ | % выполненных требований | 95% требований реализовано |
| Количество закрытых тикетов | Эффективность работы команды | 120 тикетов за спринт |
| Количество багов на релиз | Качество продукта | ≤ 5 багов на релиз |
| Скорость релиза | Время от идеи до внедрения | Новый функционал за 2 недели |

## ****Маркетинговые / продажные KPI****

Показывают эффективность продвижения и привлечения клиентов.

|  |  |  |
| --- | --- | --- |
| KPI | Описание | Пример |
| Количество лидов | Привлечённые потенциальные клиенты | 500 лидов в месяц |
| Конверсия | % клиентов из общего числа лидов | 15% конверсия в покупку |
| Стоимость привлечения клиента (CAC) | Сумма затрат на привлечение одного клиента | 1 000 ₽ за клиента |
| LTV (Life Time Value) | Доход от клиента за весь срок сотрудничества | 10 000 ₽ на клиента |

# Подсказки

## SQL

### Основные команды

|  |  |  |
| --- | --- | --- |
| Команда | Назначение | Пример |
| SELECT | Выборка данных из таблицы | SELECT \* FROM customers; |
| WHERE | Фильтрация строк | SELECT \* FROM orders WHERE status='Completed'; |
| LIMIT | Ограничение количества строк | SELECT \* FROM orders LIMIT 10; |
| ORDER BY | Сортировка | SELECT \* FROM customers ORDER BY created\_at DESC; |

### Агрегатные функции

|  |  |  |
| --- | --- | --- |
| Функция | Назначение | Пример |
| COUNT() | Подсчёт строк | SELECT COUNT(\*) FROM orders; |
| SUM() | Сумма | SELECT SUM(amount) FROM orders; |
| AVG() | Среднее значение | SELECT AVG(amount) FROM orders; |
| MIN() / MAX() | Минимум / максимум | SELECT MIN(amount), MAX(amount) FROM orders; |

### Группировка

|  |  |  |
| --- | --- | --- |
| Команда | Назначение | Пример |
| GROUP BY | Группировка данных | SELECT status, COUNT(\*) FROM orders GROUP BY status; |
| HAVING | Фильтрация после группировки | SELECT status, COUNT(\*) FROM orders GROUP BY status HAVING COUNT(\*) > 10; |

### Соединения таблиц

|  |  |  |
| --- | --- | --- |
| Тип JOIN | Назначение | Пример |
| INNER JOIN | Только совпадающие строки | SELECT o.id, c.name FROM orders o INNER JOIN customers c ON o.customer\_id = c.id; |
| LEFT JOIN | Все строки из левой таблицы | SELECT c.name, o.id FROM customers c LEFT JOIN orders o ON c.id = o.customer\_id; |
| RIGHT JOIN | Все строки из правой таблицы | SELECT o.id, c.name FROM orders o RIGHT JOIN customers c ON o.customer\_id = c.id; |

### Работа с текстом

|  |  |  |
| --- | --- | --- |
| Команда / функция | Назначение | Пример |
| LIKE | Поиск по шаблону | SELECT \* FROM customers WHERE name LIKE 'A%'; |
| CONCAT() | Склеивание строк | SELECT CONCAT(first\_name, ' ', last\_name) FROM customers; |
| UPPER() / LOWER() | Приведение к верхнему / нижнему регистру | SELECT UPPER(name) FROM customers; |

### Работа с датами

|  |  |  |
| --- | --- | --- |
| Функция / команда | Назначение | Пример |
| NOW() | Текущая дата и время | SELECT NOW(); |
| DATE() | Получить дату из даты-времени | SELECT DATE(created\_at) FROM orders; |
| DATEDIFF() | Разница между датами | SELECT DATEDIFF(NOW(), created\_at) FROM orders; |
| BETWEEN | Диапазон | SELECT \* FROM orders WHERE created\_at BETWEEN '2025-01-01' AND '2025-01-31'; |

## Базы данных

### Что это

**База данных** — это структурированное хранилище информации, с которой работает система или приложение.

**Простыми словами:**

Это место, где система хранит всё — пользователей, заказы, товары, платежи и т.д.

### Основные типы баз данных

|  |  |  |
| --- | --- | --- |
| Тип БД | Особенности | Примеры |
| **Реляционные (SQL)** | Хранят данные в таблицах (строки и столбцы), связи между таблицами | MySQL, PostgreSQL, MS SQL, Oracle |
| **Нереляционные (NoSQL)** | Не таблицы, а документы, коллекции, ключи и значения. Подходят для гибких структур | MongoDB, Redis, Cassandra, Firebase |
| **Графовые** | Хранят данные в виде узлов и связей (удобно для соцсетей, рекомендаций) | Neo4j |
| **Временные/хранилища данных (Data Warehouse)** | Для аналитики и отчетности, объединяют данные из разных систем | BigQuery, ClickHouse, Snowflake |

### Основные элементы реляционной БД

|  |  |
| --- | --- |
| Элемент | Что это |
| **Таблица (Table)** | Аналог Excel-листа — хранит данные в виде строк (записей) и столбцов (полей) |
| **Строка (Row / Record)** | Одна запись — например, один пользователь или заказ |
| **Столбец (Column / Field)** | Свойство — имя, цена, дата |
| **Первичный ключ (Primary Key)** | Уникальный идентификатор записи |
| **Внешний ключ (Foreign Key)** | Связь с другой таблицей |
| **Индекс (Index)** | Ускоряет поиск |
| **Связи (Relations)** | Показывают, как таблицы связаны между собой |

### Виды связей между таблицами

|  |  |  |
| --- | --- | --- |
| Тип связи | Пример | Описание |
| **1:1 (один к одному)** | Один пользователь → один профиль | Редко используется |
| **1:N (один ко многим)** | Один пользователь → много заказов | Самая распространённая |
| **M:N (многие ко многим)** | Много студентов → много курсов | Через промежуточную таблицу |

## Rest API

### Основные понятия

|  |  |
| --- | --- |
| Термин | Объяснение |
| **API (Application Programming Interface)** | Интерфейс для взаимодействия между программами или системами |
| **REST (Representational State Transfer)** | Стиль архитектуры API, основанный на HTTP-протоколе |
| **Endpoint** | URL, по которому можно получить доступ к ресурсу (/orders, /customers) |
| **Resource** | Объект системы, с которым работает API (заказы, клиенты, товары) |
| **Request** | Запрос к API (GET, POST, PUT, DELETE) |
| **Response** | Ответ API (данные в формате JSON или XML, код статуса HTTP) |
| **Status code** | Код ответа (200 — успех, 400 — ошибка запроса, 500 — ошибка сервера) |

### Основные HTTP методы

|  |  |  |
| --- | --- | --- |
| Метод | Назначение | Пример |
| GET | Получение данных | GET /customers — получить список клиентов |
| POST | Создание нового ресурса | POST /orders — создать новую заявку |
| PUT | Полное обновление ресурса | PUT /orders/123 — обновить заявку с ID 123 |
| PATCH | Частичное обновление ресурса | PATCH /orders/123 — обновить только статус заявки |
| DELETE | Удаление ресурса | DELETE /orders/123 — удалить заявку |

### Полезные термины для БА

|  |  |
| --- | --- |
| Термин | Значение |
| **Query parameters** | Параметры в URL для фильтрации и сортировки (/orders?status=completed) |
| **Path parameters** | Параметры в пути (/orders/123) |
| **Header** | Дополнительные данные запроса (авторизация, тип контента) |
| **Payload / Body** | Данные, которые отправляются на сервер (JSON, XML) |
| **Response code** | Код статуса HTTP (200, 201, 400, 404, 500) |
| **Rate limit** | Ограничение на количество запросов в API за врем |

## Swagger

### Что такое Swagger

**Swagger** — это инструмент для документирования и тестирования REST API.

Позволяет визуально видеть все доступные эндпоинты, методы, параметры и примеры запросов/ответов.

Часто используется вместе с **OpenAPI Specification (OAS)** — стандартом описания API.

### Основные возможности

|  |  |
| --- | --- |
| Возможность | Описание |
| Документация API | Все эндпоинты с методами (GET, POST, PUT, DELETE), параметрами и примерами |
| Тестирование | Возможность отправлять запросы прямо из Swagger UI и видеть ответ |
| Генерация кода | Можно автоматически создавать клиентский код для работы с API |
| Экспорт спецификации | JSON или YAML-файл с описанием API для разработчиков или интеграторов |

## Postman

### Что такое Postman

**Postman** — это инструмент для тестирования и работы с API.

Позволяет отправлять запросы (GET, POST, PUT, DELETE), видеть ответы, проверять структуру данных и коды статуса.

**Используется для:**

* + - тестирования интеграций;
    - проверки работы эндпоинтов;
    - демонстрации функционала API.

### Основные возможности

|  |  |
| --- | --- |
| Возможность | Описание |
| Создание запросов | GET, POST, PUT, PATCH, DELETE |
| Работа с параметрами | Query, Path, Headers, Body |
| Просмотр ответа | Формат JSON, XML, HTML, код статуса HTTP |
| Сохранение коллекций | Группировка запросов по проекту или системе |
| Скрипты и тесты | Автоматическая проверка ответа на соответствие требованиям |
| Экспорт / импорт | Обмен коллекциями с командой разработки |

### Структура запроса в Postman

**Метод:** GET / POST / PUT / DELETE

**URL / Endpoint:** например, https://api.example.com/customers

**Headers:** авторизация, тип контента (Content-Type: application/json)

**Params:** query-параметры или path-параметры (/customers?status=active)

**Body:** данные для POST / PUT / PATCH запросов (обычно JSON)

## JSON

### Что такое JSON

**JSON (JavaScript Object Notation)** — формат для обмена данными между системами.

Читаемый для человека и легко парсится системами.

Используется в API, интеграциях, хранении настроек и конфигураций.

### Основные элементы JSON

|  |  |  |
| --- | --- | --- |
| Элемент | Описание | Пример |
| Объект | Состоит из пар "ключ: значение", заключён в { } | { "id": 123, "name": "Иван" } |
| Массив | Список элементов, заключён в [ ] | [1, 2, 3] или [{"id":1},{"id":2}] |
| Ключ (Key) | Имя поля (строка) | "status": "Completed" |
| Значение (Value) | Может быть: строка, число, boolean, объект, массив, null | "amount": 250.5 |
| Boolean | Логическое значение (true или false) | "is\_active": true |
| Null | Пустое значение | "comment": null |

### Пример JSON для заявки

{

"id": 456,

"customer": {

"id": 123,

"name": "Иван Иванов",

"email": "ivan@example.com"

},

"service": "Фотосессия",

"date": "2025-10-10",

"status": "Created",

"amount": 250.5,

"notes": null,

"tags": ["urgent", "vip"]

}

## XML

### Что такое XML

**XML (eXtensible Markup Language)** — формат для хранения и обмена данными между системами.

Читаемый для человека и машин, похож на HTML, но используется для данных.

Часто применяется в API, интеграциях, отчетах и экспорт/импорт данных.

### Основные элементы XML

|  |  |  |
| --- | --- | --- |
| Элемент | Описание | Пример |
| Тег / Элемент | Основной блок данных, заключён в < > | <customer></customer> |
| Атрибут | Свойство элемента, записывается в теге | <customer id="123">Иван</customer> |
| Текстовое содержимое | Данные между открывающим и закрывающим тегом | <name>Иван Иванов</name> |
| Вложенные элементы | Элементы внутри других элементов | <order><id>456</id><amount>250.5</amount></order> |
| Комментарии | Для пояснений, игнорируются системой |  |

### Пример XML для заявки

<order>

<id>456</id>

<customer id="123">

<name>Иван Иванов</name>

<email>ivan@example.com</email>

</customer>

<service>Фотосессия</service>

<date>2025-10-10</date>

<status>Created</status>

<amount>250.5</amount>

<notes/>

<tags>

<tag>urgent</tag>

<tag>vip</tag>

</tags>

</order>

## Apache Kafka

### Что такое Apache Kafka

**Apache Kafka** — это платформа для потоковой передачи данных (event streaming).

Позволяет системам обмениваться данными в реальном времени через «сообщения» (events).

Используется для:

* + - потоковой аналитики,
    - интеграций между микросервисами,
    - передачи событий между системами (CRM, ERP, веб-приложения).

### Основные понятия

|  |  |
| --- | --- |
| Термин | Объяснение |
| Producer | Источник данных, который отправляет сообщения в Kafka |
| Consumer | Получатель данных, который читает сообщения из Kafka |
| Topic | Тема / канал, куда отправляются сообщения (например, orders или payments) |
| Partition | Раздел темы для параллельной обработки и масштабирования |
| Offset | Позиция сообщения в партиции (используется для отслеживания чтения) |
| Broker | Сервер Kafka, который хранит данные и обслуживает Producers/Consumers |
| Cluster | Несколько брокеров, работающих вместе для отказоустойчивости |

### Как это работает (упрощённо)

Producer отправляет сообщение (event) в Topic.

Kafka хранит сообщение в очереди (partition) с уникальным Offset.

Consumer читает сообщение по своему offset.

Сообщения могут быть прочитаны многократно разными потребителями (pub-sub модель).

### **Пример применения для бизнес-аналитика**

CRM → ERP: каждая новая заявка отправляется в Kafka, ERP автоматически обрабатывает заказ.

События пользователей на сайте: клик, оформление заявки, просмотр продукта → в реальном времени отправляется в аналитическую систему.

Интеграции с внешними сервисами: платежи, уведомления, статус заказов передаются через Kafka в разные системы.

### Что важно знать

**Topics и события:** знать, какие данные и в каком виде будут приходить.

**Producer / Consumer:** кто генерирует событие, кто его использует.

**Формат сообщений:** JSON или Avro, поля, типы данных.

**Очередность и надёжность:** сообщения упорядочены внутри partition, важно для бизнес-логики.

**Сценарии использования:** описывать в ТЗ, какие события нужно передавать, какие интеграции реагируют.

## RabbitMQ

### Что такое RabbitMQ

**RabbitMQ** — это брокер сообщений (message broker).

Он позволяет системам обмениваться данными асинхронно — то есть не напрямую, а через очередь сообщений.

**Используется, когда нужно:**

* + - передавать данные между сервисами,
    - разгрузить систему (чтобы не ждать ответ сразу),
    - обрабатывать события в фоне.

### Основные понятия

|  |  |
| --- | --- |
| Термин | Объяснение |
| Producer (отправитель) | Сервис, который отправляет сообщение в очередь |
| Queue (очередь) | Место, где временно хранятся сообщения, пока их не заберут |
| Consumer (получатель) | Сервис, который читает сообщения из очереди |
| Exchange (обменник) | Механизм, который решает, в какую очередь направить сообщение |
| Routing key | Ключ маршрутизации, помогает направлять сообщение в нужную очередь |
| Acknowledgment (ack) | Подтверждение от получателя, что сообщение обработано |
| Dead Letter Queue (DLQ) | Очередь для сообщений, которые не удалось обработать |

### Как это работает (упрощённо)

Producer отправляет сообщение в Exchange.

Exchange распределяет сообщение в одну или несколько очередей (Queue) по routing key.

Consumer читает сообщение из очереди, обрабатывает и отправляет подтверждение (ack).

Если Consumer недоступен — сообщение остаётся в очереди и не теряется.

### Пример для бизнес-аналитика

**Сценарий 1:**

* + - CRM создаёт новый заказ → сообщение с данными заказа отправляется в RabbitMQ → ERP система получает сообщение и автоматически создаёт накладную.

**Сценарий 2**:

* + - Сайт отправляет уведомления → RabbitMQ распределяет задачи на разные микросервисы: email, SMS, push.

**Сценарий 3:**

* + - Фотостудия принимает оплату → событие уходит в RabbitMQ → бухгалтерский сервис получает уведомление и формирует отчёт.

### Что важно знать бизнес-аналитику

**Асинхронность** — системы не ждут ответа друг от друга, это снижает нагрузку и повышает надёжность.

**Очереди** — могут быть разные типы: для заказов, уведомлений, ошибок и т.д.

**Формат сообщений** — чаще всего JSON (структура и поля нужно зафиксировать в ТЗ).

**Маршрутизация** — кто отправляет, кто получает, и по какому ключу (routing key).

**Обработка ошибок** — что делать, если сообщение не доставлено (DLQ, повторная попытка).

**Безопасность и стабильность** — нужно уточнить у разработчиков политику хранения, retries и ack.

## Use Case (Сценарий использования)

### Что это такое

Use Case — это описание того, как пользователь (актор) взаимодействует с системой, чтобы достичь конкретной цели.

Он показывает ценность для пользователя и поведение системы в ответ на его действия.

**Простыми словами:** “Что делает пользователь → и как система на это реагирует.”

### З**ачем нужен Use Case**

Помогает понять бизнес-логику системы

Уточняет функциональные требования

Легко читается заказчиком и разработчиком

Упрощает написание тест-кейсов

Основа для моделирования в UML (диаграммы вариантов использования)

### Структура Use Case

|  |  |
| --- | --- |
| Элемент | Описание |
| Название | Краткое, отражающее цель (например: "Оформление заказа") |
| Акторы | Кто взаимодействует (пользователь, администратор, внешняя система) |
| Предусловия | Что должно быть выполнено до начала (например: пользователь авторизован) |
| Основной поток | Основная последовательность шагов "пользователь → система" |
| Альтернативные потоки | Что происходит, если что-то идёт не так |
| Постусловия | Результат выполнения сценария |
| Исключения / Ошибки | Возможные сбои и реакции системы |

### **Пример Use Case**

**Название:** Оформление заказа

**Акторы:** Покупатель, Система оплаты

**Предусловия:** Пользователь авторизован, товар добавлен в корзину

Основной поток:

Пользователь открывает корзину

Проверяет список товаров

Нажимает “Оформить заказ”

Вводит адрес и выбирает способ оплаты

Система отправляет запрос в платёжный сервис

Получает подтверждение

Создаёт заказ и отображает сообщение “Заказ успешно оформлен”

Альтернативный поток:

Если платёж не прошёл → система выводит сообщение об ошибке и предлагает повторить.

Постусловия:

Заказ создан в системе

Статус заказа — “Оплачен”

### Подсказки для аналитика

Один Use Case = одна цель пользователя.

Не описывай интерфейс (кнопки, цвета) — только логику.

Используй нейтральный стиль (“Пользователь делает”, “Система отвечает”).

Для сложных сценариев можно делать диаграмму Use Case UML (акторы + овалы сценариев).

### Шаблон Use Case (для вставки в ТЗ)

|  |
| --- |
| Название: |
| Акторы: |
| Цель: |
| Предусловия: |
| Основной поток: |
| Альтернативные потоки: |
| Постусловия: |
| Исключения: |
| Связанные требования: |

## UML (Unified Modeling Language)

### Что это такое

**UML —** это унифицированный язык моделирования, используемый для визуализации, описания и документирования архитектуры, процессов и логики системы.

Он помогает бизнес-аналитику и разработчикам **говорить на одном языке**.

**Простыми словами:**

* + - UML — это способ показать **как работает система** через схемы (диаграммы).

### Зачем нужен UML бизнес-аналитику

Упрощает описание требований

Позволяет понять бизнес- и системные процессы

Помогает разработчикам и тестировщикам быстрее ориентироваться

Делает ТЗ понятным и структурированным

### Основные типы UML-диаграмм (важные для БА)

|  |  |  |
| --- | --- | --- |
| Тип диаграммы | Назначение | Пример |
| **Use Case Diagram** (диаграмма вариантов использования) | Показывает взаимодействие пользователей (акторов) с системой | “Пользователь оформляет заказ” |
| **Activity Diagram** (диаграмма деятельности) | Отображает шаги процесса, ветвления и условия | “Процесс оплаты заказа” |
| **Sequence Diagram** (диаграмма последовательности) | Показывает порядок сообщений между объектами во времени | “Пользователь → API → БД” |
| **Class Diagram** (диаграмма классов) | Описывает структуру данных и их связи | “Пользователь имеет Заказ, Заказ содержит Товар” |
| **State Machine Diagram** (диаграмма состояний) | Показывает переходы между состояниями объекта | “Статус заказа: Новый → Оплачен → Отправлен → Доставлен” |
| **Component Diagram** | Показывает, из каких модулей состоит система | “Frontend, Backend, БД, API-сервис” |

### Какой UML использовать в работе БА

|  |  |
| --- | --- |
| Цель | Подходящая диаграмма |
| Показать взаимодействие ролей и системы | Use Case |
| Описать пошаговый бизнес-процесс | Activity |
| Прописать логику обмена данными | Sequence |
| Показать структуру сущностей | Class |
| Отобразить жизненный цикл объекта | State |

## BPMN VS UML

### ****Главные различия BPMN и UML****

|  |  |  |
| --- | --- | --- |
| Критерий | **BPMN** | **UML** |
| 🎯 **Цель** | Моделирование бизнес-процессов | Моделирование систем и программных решений |
| 👥 **Ориентация** | На бизнес и пользователей | На разработчиков и аналитиков |
| 📄 **Что показывает** | Потоки задач, роли, события, ветвления, документы | Логику работы системы, связи между компонентами |
| 🔍 **Фокус** | “Кто что делает и когда” | “Как система устроена и как взаимодействует” |
| 🧱 **Основные элементы** | События, задачи, шлюзы, потоки, пулы, дорожки | Акторы, классы, сообщения, состояния |
| 🖼 **Тип диаграммы** | Диаграмма бизнес-процесса | Диаграммы: Use Case, Activity, Sequence, Class, State |
| 📊 **Глубина детализации** | Высокоуровневое описание процесса | Детальное описание логики и архитектуры |
| 💼 **Кто использует** | Бизнес-аналитики, владельцы процессов | Системные аналитики, архитекторы, разработчики |
| 📘 **Нотация** | BPMN 2.0 (Business Process Model and Notation) | UML 2.x (Unified Modeling Language) |

### ****Когда использовать****

|  |  |
| --- | --- |
| Ситуация | Что выбрать |
| Нужно показать **как работает бизнес-процесс** | 🟢 BPMN |
| Нужно описать **что делает система и как она устроена** | 🔵 UML |
| Нужно согласовать **ролей, шаги, документы, ответственных** | 🟢 BPMN |
| Нужно документировать **функциональные требования и API** | 🔵 UML |
| Проект — **автоматизация бизнес-процесса** | BPMN + UML Activity |
| Проект — **разработка CRM, ERP, веб-приложения** | UML (Use Case, Sequence, Class) |

### Кратко (если спросят на собеседовании):

**BPMN** — показывает что делает бизнес.

**UML** — показывает **как это делает система**.

Я использую BPMN для моделирования процессов и UML (Use Case, Sequence, Activity) для системного анализа.

## Use Case VS User Story

### ****Различия****

|  |  |  |
| --- | --- | --- |
| Критерий | **User Story** | **Use Case** |
| 🎯 Цель | Понять, что нужно пользователю | Описать, как это работает |
| 📄 Формат | Простое предложение | Пошаговый сценарий |
| 🧠 Уровень детализации | Высокоуровневый | Подробный |
| 👥 Ориентация | На пользователя и бизнес-ценность | На систему и поведение |
| 📋 Используется в | Agile (Scrum, Kanban) | Waterfall, SRS, ТЗ |
| 📦 Применение | Бэклог, планирование спринта | Документирование требований |
| 📊 Объём | 2–3 строки | 1–2 страницы |
| 📘 Фокус | Что и зачем | Как и в каком порядке |

### Кратко

**User Story** — это "что хочет пользователь и зачем" (ценность).

**Use Case** — это "как система реализует эту цель" (пошаговая логика).

## MoSCoW

### Что это такое

**MoSCoW** — это техника приоритизации требований, задач или функций.

Она помогает разделить, что обязательно должно быть в релизе, а что можно отложить.

### Название расшифровывается по первым буквам:

Must have

Should have

Could have

Won’t have (this time)

### Классификация требований

|  |  |  |
| --- | --- | --- |
| Категория | Значение | Пример |
| 🟥 Must have | Обязательно — без этого система не имеет смысла, критично для MVP | Авторизация, базовый поиск, оформление заказа |
| 🟧 Should have | Желательно — важно, но можно временно обойтись | Сортировка по цене, фильтры по категориям |
| 🟨 Could have | Можно добавить — nice-to-have, если останется время и ресурсы | Отзывы, избранное, персональные рекомендации |
| 🟩 Won’t have (this time) | Не в этом релизе — не реализуется сейчас, но может быть позже | Поддержка Apple Pay, интеграция с Telegram |

### Когда использовать MoSCoW

При планировании MVP или релиза

При ограниченном времени или бюджете

Когда заказчик хочет “всё и сразу”

Для согласования приоритетов между бизнесом и разработкой

При работе по Agile / Scrum / Kanban

### Как применять

Собери список всех требований (функций, улучшений).

Обсуди с заказчиком и командой ценность и влияние каждого.

Отнеси каждое требование к одной из 4 категорий.

Зафиксируй приоритеты в Confluence / Jira / ТЗ / Product Backlog.

При изменениях — пересмотри приоритеты.

## Session ID vs JWT (JSON Web Token)

### Что такое Session ID

**Session ID** — это уникальный идентификатор, который создаётся сервером после того, как пользователь успешно вошёл в систему.

Он используется, чтобы сервер “помнил”, кто вы, между запросами.

**Простыми словами:**

* + - Когда вы авторизуетесь, сервер выдаёт Session ID и хранит у себя, а браузер отправляет его при каждом запросе.

### Как это работает (Session ID)

Пользователь вводит логин и пароль

Сервер проверяет их и создаёт Session ID

Session ID сохраняется:

* + - на сервере — в памяти или базе данных
    - в браузере — в cookie

Каждый запрос от пользователя содержит этот Session ID

Сервер проверяет ID и "узнаёт" пользователя

### Особенности (Session ID)

|  |  |
| --- | --- |
| Преимущество | Недостаток |
| Простая и понятная реализация | Хранение сессий на сервере (нагрузка при большом количестве пользователей) |
| Подходит для веб-сайтов и CRM | Неудобно масштабировать (если несколько серверов) |
| Можно "разлогинить" пользователя с серверной стороны | Требует постоянного хранения данных о сессии |

### Что такое JWT (JSON Web Token)

JWT (JSON Web Token) — это токен авторизации, который содержит зашифрованную информацию о пользователе.

Используется для безсерверной (stateless) аутентификации.

**Простыми словами:**

* + - Сервер не хранит сессию. Он просто выдаёт токен,
    - в котором уже “зашито” кто вы и когда токен истекает.

### Как это работает (JWT)

Пользователь логинится

Сервер проверяет логин и пароль

Генерирует JWT, в который включены:

* + - user\_id
    - время жизни токена
    - подпись (для защиты)

Клиент хранит токен (в localStorage, cookie, заголовке Authorization)

При каждом запросе клиент отправляет токен

Сервер проверяет подпись и срок действия — если всё ок, запрос разрешён

### Разбивается на 3 части (JWT):

**Header** — тип токена и алгоритм шифрования

**Payload** — данные (user\_id, роль, срок действия)

**Signature** — подпись для проверки подлинности

### **Особенности (JWT)**

|  |  |  |
| --- | --- | --- |
| Критерий | Session ID | JWT (JSON Web Token) |
| Где хранится | На сервере и в cookie | Только на клиенте |
| Кто проверяет пользователя | Сервер по сохранённой сессии | Сервер по подписи токена |
| Масштабируемость | Плохая (нужно хранить сессии) | Отличная (stateless) |
| Безопасность | Средняя, можно защитить cookie | Высокая при правильной реализации |
| Разлогин | Можно удалить сессию на сервере | Труднее, нужно “чёрный список” токенов |
| Где используется | Классические сайты, CRM | SPA, мобильные, REST API, микросервисы |

### Кратко

**Session ID** — старый, но надёжный способ хранить состояние пользователя на сервере.

**JWT** — современный формат токена, который несёт данные о пользователе и не требует хранения на сервере.

В веб-проектах с API и мобильными клиентами чаще используют JWT, а в корпоративных системах — Session ID.

## ТЗ VS ТС

### Различия

|  |  |  |
| --- | --- | --- |
| Критерий | **Техническое задание (ТЗ)** | **Техническая спецификация (Tech Spec)** |
| Цель | Описывает, что нужно сделать и зачем | Описывает, как это будет реализовано |
| Фокус | Бизнес-логика и пользовательский функционал | Техническая реализация и архитектура |
| Авторы | Бизнес-аналитик, заказчик | Системный аналитик, разработчик |
| Уровень детализации | Средний (понятен заказчику) | Высокий (понятен разработчику) |
| Этап проекта | Инициация и планирование | Проектирование и разработка |
| Формат | Человеко-ориентированный документ | Технический документ с диаграммами и API |
| Пример | “Пользователь должен иметь возможность оформить заказ.” | “Метод POST /orders создаёт заказ в таблице Orders с полем status=‘pending’.” |

### Кратко

ТЗ отвечает на вопрос “что делаем и зачем”, а техническая спецификация — “как именно это будет работать”.

ТЗ создаёт бизнес-аналитик, техническую спецификацию — разработчик или системный аналитик.

## Структура ТС

* + 1. **Введение** — цели и краткое описание системы.
    2. **Архитектура системы** — схема компонентов и взаимодействий.
    3. **Технологический стек** — используемые языки, фреймворки, БД, сервисы.
    4. **Структура данных** — таблицы, поля, связи, ER-диаграмма.
    5. **API и интеграции** — эндпоинты, форматы запросов/ответов, методы.
    6. **Логика работы системы** — основные бизнес-правила и алгоритмы.
    7. **Безопасность** — авторизация, аутентификация, шифрование, роли.
    8. **Интерфейсы** — макеты, поля форм, взаимодействие с пользователем.
    9. **Нефункциональные требования** — производительность, масштабируемость, надёжность.
    10. **Среда развертывания** — окружения, сервера, CI/CD, конфигурации.
    11. **Журналирование и мониторинг** — логирование, метрики, уведомления.
    12. **Обработка ошибок и отказоустойчивость** — резервы, откаты, восстановление.
    13. **Тестирование и валидация** — тест-кейсы, сценарии приёмки, нагрузочные тесты.
    14. **Приложения** — схемы, диаграммы, примеры JSON/XML, ссылки на Swagger/Figma