1. Общие вопросы

1.1. Как правильно структурировать ТЗ на разработку ПО?

Классическая структура:

Введение (цель, область применения).

Общее описание (аналоги, пользователи).

Функциональные и нефункциональные требования.

Интерфейсы и интеграции.

Критерии приемки.

Гибкая структура (Agile):

User Stories с критериями приемки.

Эпики и подзадачи в Jira/Notion.

Минимально жизнеспособный продукт (MVP).

Стандартизированная (ГОСТ 34):

Общие положения.

Назначение и цели.

Технические требования.

Этапы разработки.

1.2. Основные ошибки в ТЗ и как их избежать

Размытые формулировки → Использовать SMART-критерии.

Отсутствие приоритезации → Метод MoSCoW (Must/Should/Could/Won’t).

Игнорирование нефункциональных требований → Включить безопасность, нагрузку.

1.3. Отличие ТЗ от брифа и SRS

ТЗ – юридический документ с техтребованиями (по ГОСТ).

Бриф – краткое описание идеи без деталей.

SRS (IEEE 830) – фокус на требованиях ПО без бюджета.

1.4. Стандарты для ТЗ

ГОСТ 34 – для госзаказов (РФ).

IEEE 830 – международный стандарт SRS.

ISO/IEC/IEEE 29148 – современный стандарт на требования.

2. Формулировка требований

2.1. Функциональные требования (5 примеров)

Регистрация: Пользователь может зарегистрироваться через email/соцсети.

Поиск: Система поддерживает фильтры по цене и рейтингу.

Корзина: Товары сохраняются при выходе из аккаунта.

Оплата: Интеграция с PayPal/Сбербанком.

Админка: Возможность блокировки пользователей.

2.2. Нефункциональные требования (примеры)

Производительность: Отклик < 1 сек при 1000 RPS.

Безопасность: HTTPS, 2FA для админов.

Масштабируемость: Поддержка 10К пользователей онлайн.

2.3. Пользовательские сценарии (use cases)

Оформление заказа:

Актор: Покупатель.

Шаги: Выбор товара → Корзина → Оплата → Подтверждение.

Сброс пароля:

Актор: Пользователь.

Шаги: Запрос → Ссылка на email → Новый пароль.

2.4. Как избежать двусмысленностей

Шаблон: "Система должна [действие] при [условии]".

Глоссарий: Определить термины (например, "Клиент" = платящий пользователь).

Примеры: Вместо "быстрый поиск" → "Поиск выдает результаты за < 0.5 сек".

3. Детализация и спецификации

3.1. Описание архитектуры

Диаграммы:

UML: Use Case, Sequence, Class.

BPMN: Для процессов (например, доставка).

Стек технологий:

Фронтенд: React.js.

Бэкенд: Python/Django.

Схема БД: ER-диаграмма в Draw.io.

3.2. Требования к интерфейсу

UI: Макеты в Figma (разрешения, цвета, шрифты).

UX: Анимации ≤ 300 мс, микроинтеракции.

Доступность: Поддержка WCAG 2.1 (контраст, screen readers).

3.3. Безопасность

Данные: Шифрование (AES-256).

Аутентификация: OAuth 2.0 + JWT.

Защита: WAF, регулярные аудиты.

3.4. Интеграции

Платежи: API ЮKassa (REST/JSON).

Доставка: СДЭК (SOAP).

Аналитика: Google Analytics (GTM).

4. Оценка и планирование

4.1. Оценка трудоемкости

Метод декомпозиции: Разбить на задачи и оценить часы.

Story Points: Fibonacci-шкала в Scrum.

COCOMO: Для сложных проектов.

4.2. Этапы и сроки

Waterfall:

Анализ: 2 недели.

Разработка: 8 недель.

Agile: Спринты по 2 недели.

4.3. Критерии приемки

Пример: "Поиск работает при 100+ параллельных запросах".

Тесты: 90% покрытие кода.

Документация: Руководство пользователя в PDF.

5. Дополнительные аспекты

5.1. Документация

Пользовательская: PDF-инструкция + видео.

Техническая: Swagger, Readme.md.

API: Postman-коллекция.

5.2. Тестирование

Юнит-тесты: Jest/Pytest.

Нагрузочное: JMeter (1000 RPS).

Ручное: Чек-листы на 50+ сценариев.

5.3. Поддержка

Гарантия: 3 месяца на исправление багов.

SLAs: Ответ за 24 часа.

Обновления: Ежеквартальные патчи.

6. Примеры и шаблоны

6.1. ТЗ для мобильного приложения

Проект: "FitTracker" (трекер тренировок).

Требования:

Функции:

Синхронизация с Apple Health.

Кастомизация тренировок.

Технологии: Swift (iOS), Kotlin (Android).

6.2. Шаблон ТЗ в Markdown

markdown

# Техническое задание

## 1. Введение

\*\*Цель\*\*: Разработка CRM для малого бизнеса.

## 2. Функциональные требования

2.1. Клиент может создать заказ.

2.2. Админ видит статистику продаж.

## 3. Нефункциональные требования

3.1. Доступность 99.9% (SLA).