Гайд по подготовке технического задания (ТЗ)

1. Что такое техническое задание и зачем оно нужно

Техническое задание (ТЗ) — это документ, который описывает проект, его цели, требования и ограничения для команды разработки. Оно служит общим соглашением между заказчиком, аналитиком и разработчиками.

Задачи ТЗ:

○ зафиксировать требования в понятном виде;

○ определить границы проекта;

○ снизить риск недопонимания;

○ сформировать базу для оценки сроков и стоимости;

○ создать основу для тестирования и приемки.

2. Основные принципы создания ТЗ

1. Ясность.

- Используйте простой и однозначный язык.

- Избегайте жаргона и сложных фраз.

2. Одноуровневость.

- Каждый раздел отвечает на свой вопрос.

- Функциональные, нефункциональные и бизнес-требования разделяются.

3. Точность.

- Конкретизируйте требования: что, зачем, как.

- Не оставляйте «скользкие» формулировки без уточнения.

4. Обоснованность.

- Указывайте бизнес-цель для ключевых требований.

- Связывайте функциональность с результатом.

5. Аудитория.

- Документ должен быть понятен и заказчику, и разработчику.

- Структура помогает разным ролям быстро находить нужную информацию.

3. Рекомендуемая структура ТЗ

3.1. Титульный лист

○ Название проекта

○ Версия документа

○ Дата подготовки

○ Автор / команда авторов

○ Контактные данные

3.2. Введение

○ Краткое описание проекта.

○ Цель документа.

○ Область применения.

○ Краткое описание текущей ситуации (as-is).

3.3. Термины и определения

○ Глоссарий ключевых понятий.

○ Расшифровки аббревиатур.

○ Важно для единообразного понимания.

3.4. Цели и задачи проекта

○ Общая бизнес-цель.

○ Конкретные задачи, которые решает проект.

○ Ожидаемые эффекты (экономия времени, снижение ошибок, рост продаж и т.д.).

3.5. Описание продукта или сервиса

○ Краткое описание функциональности.

○ Целевая аудитория.

○ Основные сценарии использования.

○ Ограничения и границы.

3.6. Функциональные требования

Каждое требование должно содержать:

○ уникальный идентификатор;

○ описание;

○ при необходимости критерий приемки.

Пример разделов:

○ Пользовательские роли.

○ Вход и регистрация.

○ Основные сценарии.

○ Управление данными.

○ Экспорт/импорт.

○ Уведомления.

○ Администрирование.

3.7. Нефункциональные требования

○ Производительность.

○ Надежность и доступность.

○ Безопасность.

○ Масштабируемость.

○ Удобство использования.

○ Совместимость.

○ Требования по локализации и поддержке.

3.8. Ограничения и предположения

○ Технологические ограничения.

○ Внешние интеграции.

○ Ограничения инфраструктуры.

○ Предположения, от которых зависит проект.

3.9. Критерии приемки

○ Условия, при которых заказчик считает задачу выполненной.

○ Методы проверки.

○ Список тестовых сценариев или случаев использования.

3.10. Риски и меры по снижению

○ Список известных рисков.

○ Степень влияния.

○ Варианты смягчения.

3.11. План работ и этапы

○ Основные этапы разработки.

○ Ключевые вехи.

○ Оценочные сроки.

○ Приоритеты.

3.12. Приложения

○ Диаграммы (BPMN, ER, UML).

○ Прототипы и макеты.

○ Примеры данных.

○ Шаблоны документов.

4. Рекомендации по написанию

4.1. Используйте простой язык

○ Предпочитайте короткие предложения.

○ Избегайте многословия.

○ Формулируйте требования в форме «Система должна…» или «Пользователь может…».

4.2. Разграничивайте уровни

○ Бизнес-требования описывают цель.

○ Функциональные — что должно делать решение.

○ Технические — как это должно работать.

○ Нефункциональные — свойства системы.

4.3. Формализуйте требования

○ Присваивайте уникальный ID каждому требованию.

○ При необходимости добавляйте приоритет.

○ Делайте ссылки на связанные требования.

4.4. Используйте шаблоны и стандарты

○ Шаблон ускоряет работу и делает документ понятнее.

○ Стандартные форматы: BRD, SRS, user stories, use cases.

○ Внутренний шаблон компании должен быть един для всех проектов.

4.5. Работайте через сценарии

○ Описывайте ключевые пользовательские сценарии.

○ «Кто — что — зачем» помогает не потерять цель.

○ Используйте примеры, когда это нужно.

4.6. Проверяйте на согласованность

○ Требования не должны противоречить друг другу.

○ Не должно быть дублирования.

○ Следите, чтобы терминология была одинаковой.

4.7. Соглашайтесь о терминологии

○ Включайте глоссарий в документ.

○ Ограничивайте количество терминов.

○ Всю новую терминологию определяйте сразу.

5. Как подготовить ТЗ для команды разработки

5.1. Начинайте с бизнес-цели

○ Определите, какую задачу решает проект.

○ Зачем конечный пользователь будет пользоваться продуктом.

○ Чем это важно для бизнеса.

5.2. Опишите пользователей и роли

○ Кто будет использовать систему.

○ Какие роли есть в продукте.

○ Отличия между ролями.

5.3. Опишите основные сценарии

○ Реальные кейсы использования.

○ Порядок действий пользователя.

○ Ожидаемое поведение системы.

5.4. Уточняйте детали в диалоге с разработчиками

○ Не пытайтесь описать все технические детали самостоятельно.

○ Обсуждайте архитектурные решения с архитектором.

○ Согласуйте API, формат данных, ограничения интерфейсов.

5.5. Включайте примеры данных

○ Формат запросов и ответов.

○ Примеры сущностей.

○ Примеры ошибок.

5.6. Документируйте условия сдачи

○ Что считается «готово».

○ Как будут проходить приемочные тесты.

○ Критерии успешной проверки.

6. Структура ТЗ для разработки: примерная схема

1. Введение
2. Общие сведения
3. Цели
4. Термины
5. Заинтересованные стороны
6. Описание продукта
7. Функциональные требования
8. Нефункциональные требования
9. Интеграции
10. Условия внедрения
11. Приемка и тестирование
12. План релиза
13. Приложения

7. Советы по оформлению

○ Используйте оглавление.

○ Нумеруйте разделы и подпункты.

○ Добавляйте таблицы для сложных данных.

○ Визуализируйте процессы диаграммами.

○ Оставляйте поля для комментариев и правок.

○ Проверяйте документ на орфографию.

8. Контроль качества ТЗ

○ Проведите проверку с командой разработки.

○ Пройдитесь по основным сценариям вместе с тестировщиком.

○ Выполните ревью с заказчиком.

○ Оцените полноту: все ли требования покрывают бизнес-цель.

○ Оцените непротиворечивость: нет ли дублирований и конфликтов.

9. Полезные шаблоны и приемы

9.1. User story

○ Как [тип пользователя], я хочу [цель], чтобы [причина].

9.2. Acceptance criteria

○ Дано… Когда… Тогда…

9.3. Таблица требования

ID	Требование	Приоритет	Критерий приемки	Примечание
FR-01	Система должна позволять ...	Высокий	...	...

9.4. Диаграмма потока

○ Используйте BPMN для бизнес-процессов.

○ Используйте UML для структурных моделей.

10. Итоговые рекомендации

○ ТЗ должно быть рабочим документом, а не формальностью.

○ Обновляйте документ по мере уточнения требований.

○ Фиксируйте изменения через версионность.

○ Вовлекайте команду разработки на ранней стадии.

○ Сфокусируйтесь на том, какую ценность приносит каждая функция.

Этот документ предназначен для подготовки технического задания и адаптации его под команду разработки. Строгая структура и четкая формулировка требований помогают уменьшить риски и повысить скорость реализации.