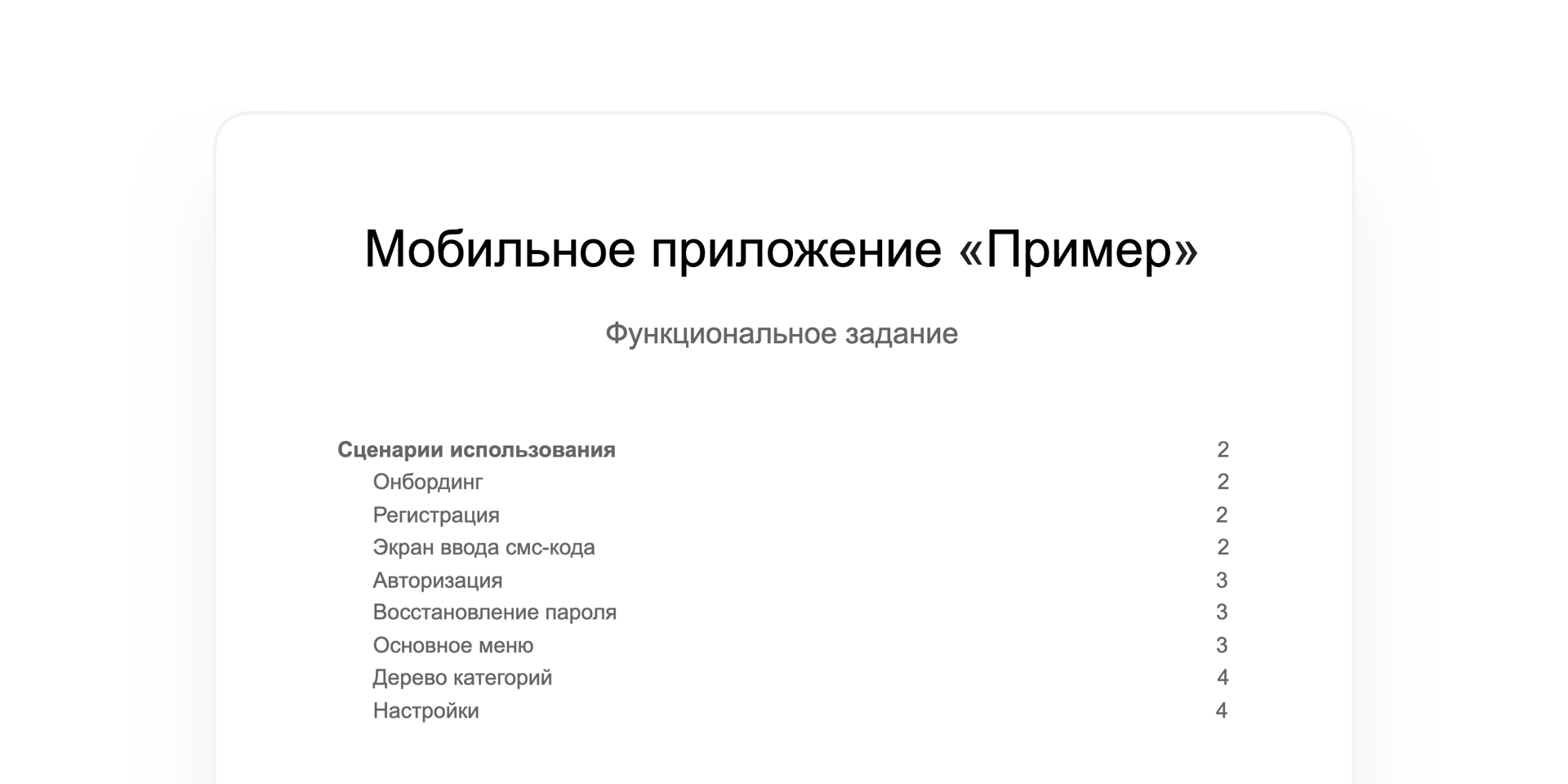
**Формирование ТЗ**

**1. Функциональное задание**

Функциональное задание (ФЗ) — самый мощный артефакт в нашей проектной документации. К нему обращаются на каждом этапе разработки от прототипирования до релиза. На его основе дизайнеры создают экраны, разработчики пишут код, а отдел QA проводит тестирование. И при этом у него нет ни одного магического свойства. Сила ФЗ в том, что оно подробно описывает функции, которые доступны пользователю при работе с приложением. На интервью с клиентом мы узнаём, какие ролевые модели (администратор, модератор, простой пользователь) предусмотрены в приложении, и описываем набор функций для каждой роли: куда пользователь может пойти, что сделать и какой результат его ждёт. Пока мы с клиентом готовим этот артефакт, дизайнеры делают прототипы экранов, которые соответствуют возможностям приложения, прописанным в ФЗ. Готовый документ проверяют разработчики.

На этапе проектирования я читаю готовое ФЗ минимум два раза. Первый — целиком, ни на чём не останавливаясь, чтобы у меня сложилось общее впечатление о работе приложения. Затем я начинаю читать ФЗ на второй раз, более вдумчиво и критически. Это позволяет мне: 1) задать вопросы, которые меня интересуют, чтобы проверить документ на ошибки; 2) дополнить конкретные моменты в приложении исходя из своего опыта; 3) дать более точную оценку проекту.

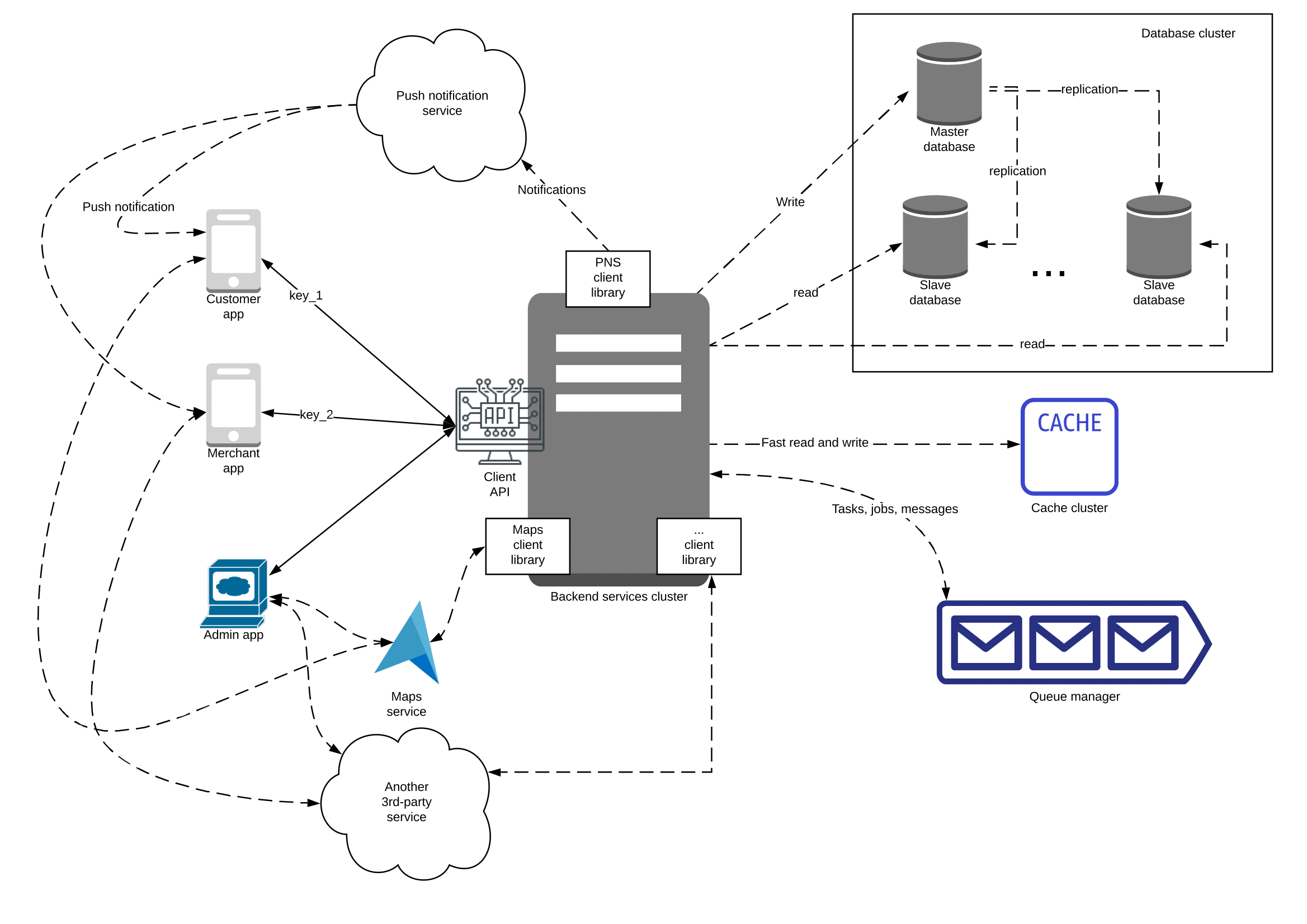
— Павел Разуваев, iOS-разработчик «Лайв Тайпинга»



Содержание функционального задания

**2. Функциональные схемы**

Функциональные схемы (ФС) иллюстрируют, как простые функции приложения группируются в более сложные и взаимодействуют друг с другом. Те, у кого много очков опыта в айти, легко воспользуются этим артефактом. Но если вы ещё не подняли скилл до нужно уровня, прочитать функциональные схемы вам поможет описание компонентов системы.



На схеме изображен верхний уровень системного разбиения: основные функциональные объекты — приложения, админки, сервер, пуши, карты, база данных, кэш — и их связи.

**3. Описание компонентов системы**

Вспомогательный артефакт, который уточняет работу ФС. Схемы изображают функциональные модули и их связи, а описание подробно рассказывает о том, что это за компоненты, чтобы человеку было удобнее читать схему. Нужен, когда мы хотим пояснить детали людям из команды клиента: это могут быть как разработчики, так и те, кто не связан с программированием напрямую. Пишем его по необходимости, поэтому по шкале важности он получает только один огонёк из трёх.

**4. Технические заметки**

Артефакт описывает, как разработчики реализуют функции из ФЗ. Мы не хотим тратить деньги клиента на очевидные вещи, поэтому делаем технические заметки только на те места, которые кажутся нам рискованными и требующими внимания: любые алгоритмы, расчёты, интеграции.

До начала работы над проектом в техзаметках фиксируется информация, которая помогает более комплексно понимать технические требования проекта и быстро включает в него новых людей. Заметки избавляют от необходимости рассуждать на митингах, как именно реализовать ту или иную фичу. Благодаря этому ты меньше отвлекаешь тиммейтов во время работы, а тиммейты — тебя.

— Андрей Дёмин, Android-разработчик «Лайв Тайпинга»

Мы бы очень хотели видеть технические заметки в проектах, которые приходят к нам на поддержку и развитие. Когда к нам поступает новая система, нам нужно понять, как она работает внутри. Если артефакта нет, то нам приходится разбираться в чужой работе самостоятельно — обычно это долго и больно.

**5. Спецификация API (application programming interface)**

API — язык, на котором приложение «общается» с серверной частью. Когда пользователь совершает действие, внутри приложения формируется запрос, который улетает в сервер, обрабатывается и возвращается в виде ответа. Спецификация устанавливает нормы этой коммуникации. Артефакт не используется в приложения без бэкенда.

**6. Карта рисков**

Мы составляем карту рисков для того, чтобы показать клиенту опасные места в проекте: размытые задачи и интеграции, с которыми мы ещё не работали. Почему это важно? Есть задачи, выполнение которых нельзя точно оценить в процессе проектирования. Если мы не скажем об этом клиенту, у него сложатся неверные ожидания по срокам и стоимости проекта. Артефакт получает одну комету, потому что такие задачи в нашей практике появляются редко.

**7. Документация на фичу**

Этот артефакт — референс к гостовскому ТЗ: он собирает технические и функциональные характеристики на одну фичу в одном месте. Нужен, когда к нам на поддержку приходит готовый проект и мы добавляем в него новые функции или исправляем баги.

Есть обязательные артефакты, без которых невозможно представить приложение, — это функциональное задание и технические заметки. В проектах, которые приходят на поддержку, их заменяет документация на фичу. Наличие остальных артефактов зависит от сложности проекта и опыта команды. Мы делаем некоторые вещи с закрытыми глазами, поэтому собираем только те документы, которые несут реальную пользу проекту. Этот подход выгоден и нам, и клиенту: мы не тратим ресурсы на банальные вещи и больше вкладываемся в то, что повлияет на работоспособность приложения и оценку пользователей.

**Структура ТЗ**

**Информация о запросе.** Тут мы прописываем идею будущего приложения, причину его создания и целевую аудиторию, на которую будет рассчитан продукт, указываем принцип его работы.

План работы. Прописываем сроки создания ТЗ, кто участвует в команде разработки и какие задачи будет выполнять. Расписываем подробно по неделям, что будет сделано и кем.

**Коммерческое предложение.** Презентуем смету, в которой прописана стоимость работы. При необходимости разрабатываем несколько вариантов.

**Техническая поддержка.** Здесь пишем о том, какие есть варианты технической поддержки после запуска приложения, какие могут возникнуть ситуации, как и насколько быстро мы их решаем.