6) Создайте техническую архитектуру проекта по аб тестированию продукта он-лайн кинотеатра с учетом кросс-девайс аналитики по следующей гипотезе: Если договориться с банком о 99% кэшбэке на подписку первого месяца, то это повысит конверсию в подписку на 30%, благодаря упрощенному принятию решения со стороны пользователя.

На схеме необходимо отобразить:

1) Управленческий процесс по договоренностям с внешними партнерами

2) Архитектуру данных с указанием систем, из которых будем скачивать данные

3) Внутрикомандное взаимодействие

**Гипотеза**

В эксперименте с Автоплеем мы предположили, что благодаря фиче пользователь будет дольше оставаться на платформе и посмотрит больше фильмов. А ещё решили, что она избавит человека от «мук выбора».

Главными метриками стали TVTU (total view time на юзера) и количество просмотренного контента за сессию.

### Создание фичи

На основе гипотезы команда пишет техническое задание и описывает, что мы хотим увидеть, когда зритель приходит на платформу и пробует новый функционал.

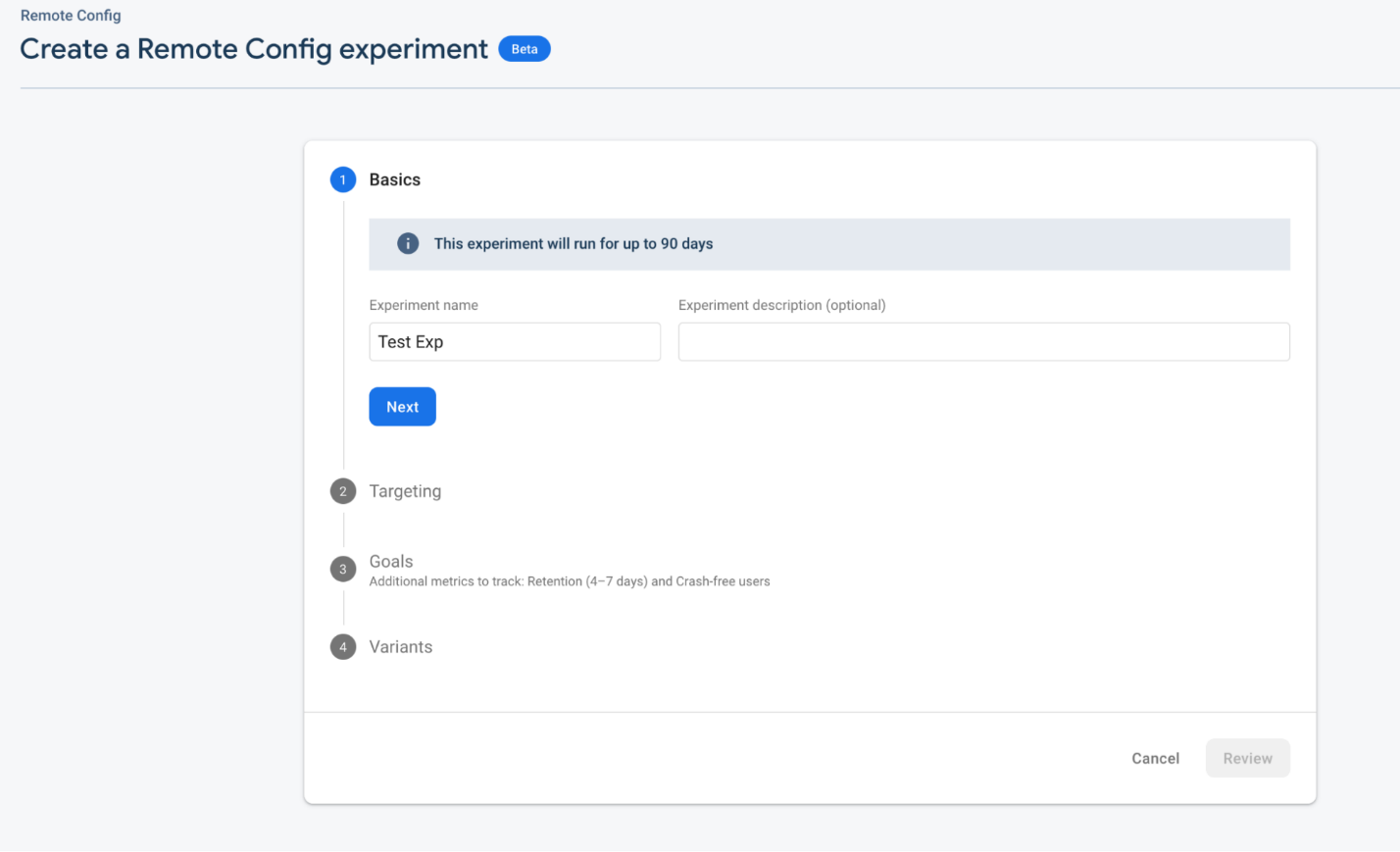
ТЗ уходит разработчикам, созданная фича — тестировщикам, а после возвращается к нам. И начинаются эксперименты.

Важное замечание. Чтобы фича работала, контент нужно размечать. Так что в эксперименте мы обращали внимание только на размеченные фильмы: в основном это были originals.

**Описание процесса А\Б тестирования** функцииAutoplay

Мы подготавливаем дашборды, по которым смотрим, как протекает эксперимент, и следим за всеми метриками. И отбираем две группы пользователей: одной - фича будет доступна, другой - нет. Для последнего используется Firebase, софт, который рандомно бьёт людей на группы.

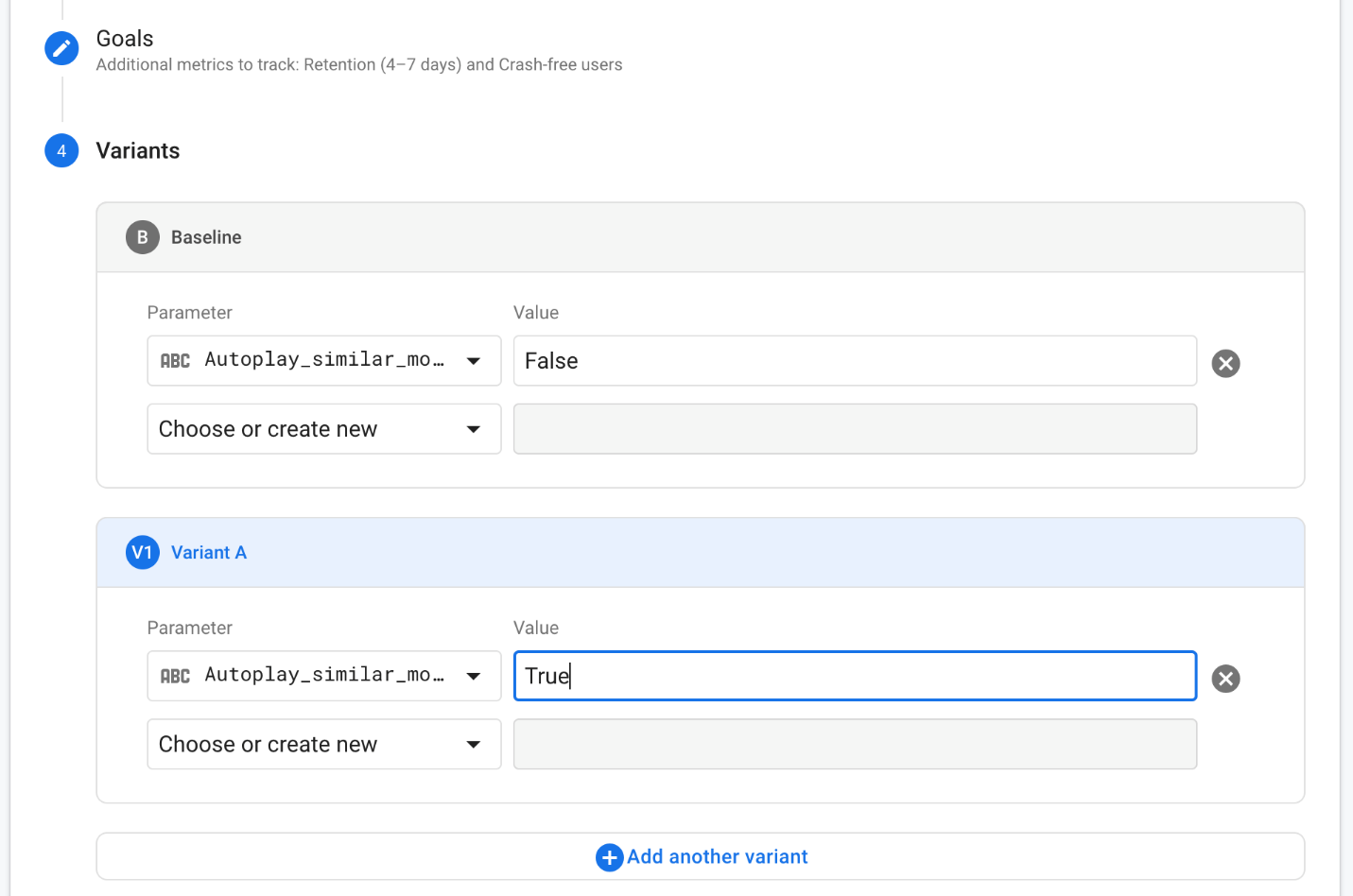
Сначала определяем название эксперимента и описание, куда обычно вставляем ссылку на confluence.



Далее выбираем платформу на, которой хотим провести эксперимент, и версию приложения. Мы запускаем тесты отдельно на каждой платформе, потому что результаты могут отличаться из-за интерфейса, аудитории и так далее.

Так же выбираем процент аудитории, которую будем задействовать.

У каждого эксперимента есть свой ключ, который помогает понять, к какой группе относится зритель — А или В (то есть будет ли ему доступна фича или нет). По этому ключу мы выкатываем новую версию приложения.



Спустя определённое количество времени, мы смотрим метрики и делаем вывод, есть ли разница в поведении людей на платформе и насколько она заметна. В нашем случае тесты проводились месяц.

**Расчёты**

Чтобы определить, сколько нам держать эксперимент, мы выбираем MDE (minimal detectible effect) — минимальный эффект, который мы ожидаем увидеть благодаря нашим изменениям. Далее подставляем в стандартную формулу расчёта необходимого количества наблюдений:

*def sample\_size(var, mde, alpha, beta):*

*# Total samples needed N\_test + N\_control*

*n = 2 \* 2\*((z\_alpha + z\_beta)\*\*2)\*var/(mde\*\*2)*

*return n*

Где:

*var* — дисперсия метрики.

*z\_alpha* — z значеие относящиеся к вероятности совершить ошибку первого рода (обычно 0.05 и для него z\_value = 1.96).

*z\_beta* — z значеие относящиеся к вероятности совершить ошибку второго рода (обычно 0.2 и для него z\_value = 0.84).

*mde* — минимальная разница в метрике, которую хотим заметить.

В итоге получаем необходимое количество наблюдений в обеих группах. Понимая сколько примерно пользователей заходит в день, можем примерно прикинуть как долго нужно держать эксперимент.

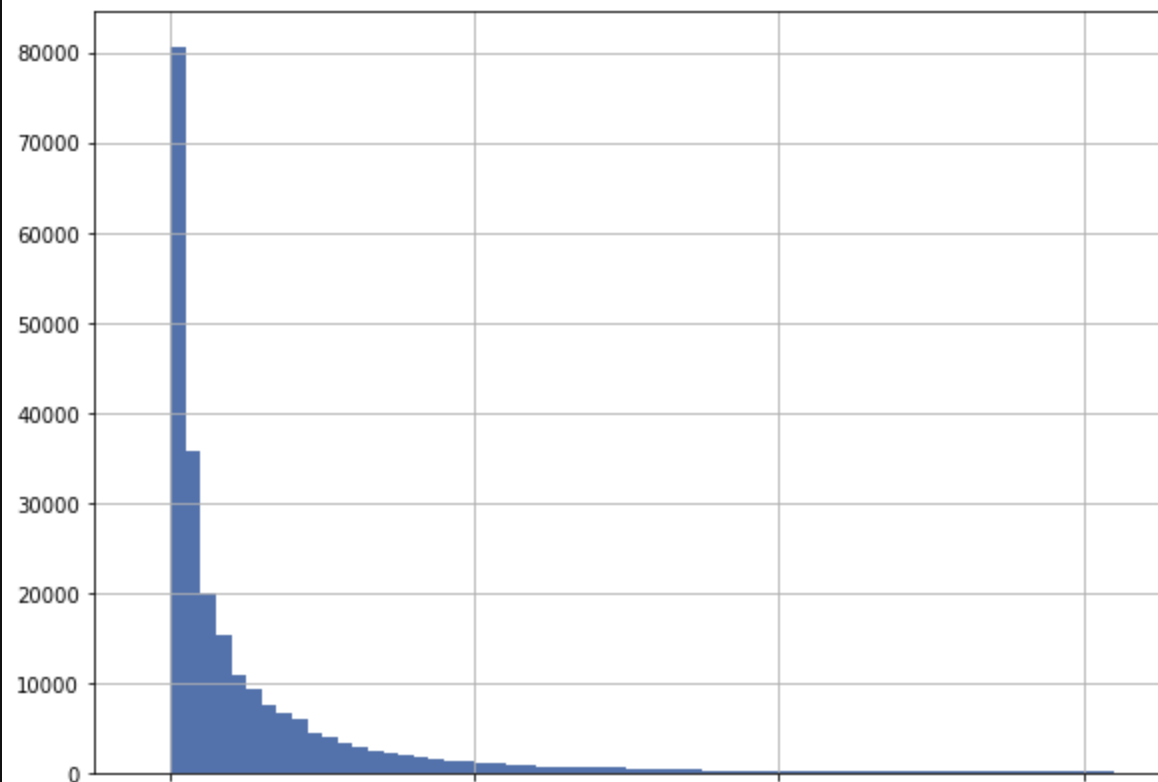
Дальше считаем статистическую значимость получившихся изменений в метрике.

**Т-тест**

Мы запускаем стандартный Т-тест и смотрим на значение p\_value — меньше ли оно определенной границы, (обычно берётся 0,05). Если да, фиксируем статистически значимое изменение в метрике. Если нет, то утверждать, что есть разница, нельзя.

Важно помнить, что если эксперимент не прокрасился это не значит, что вообще нет эффекта. Возможно он просто меньше выбранного нами MDE.

Отдельно стоит отметить как мы проверяем, что можем доверять Т-тесту — как и любому другому тесту для нашей метрики. Учитывая, что распределение TVTU крайне ненормальное.



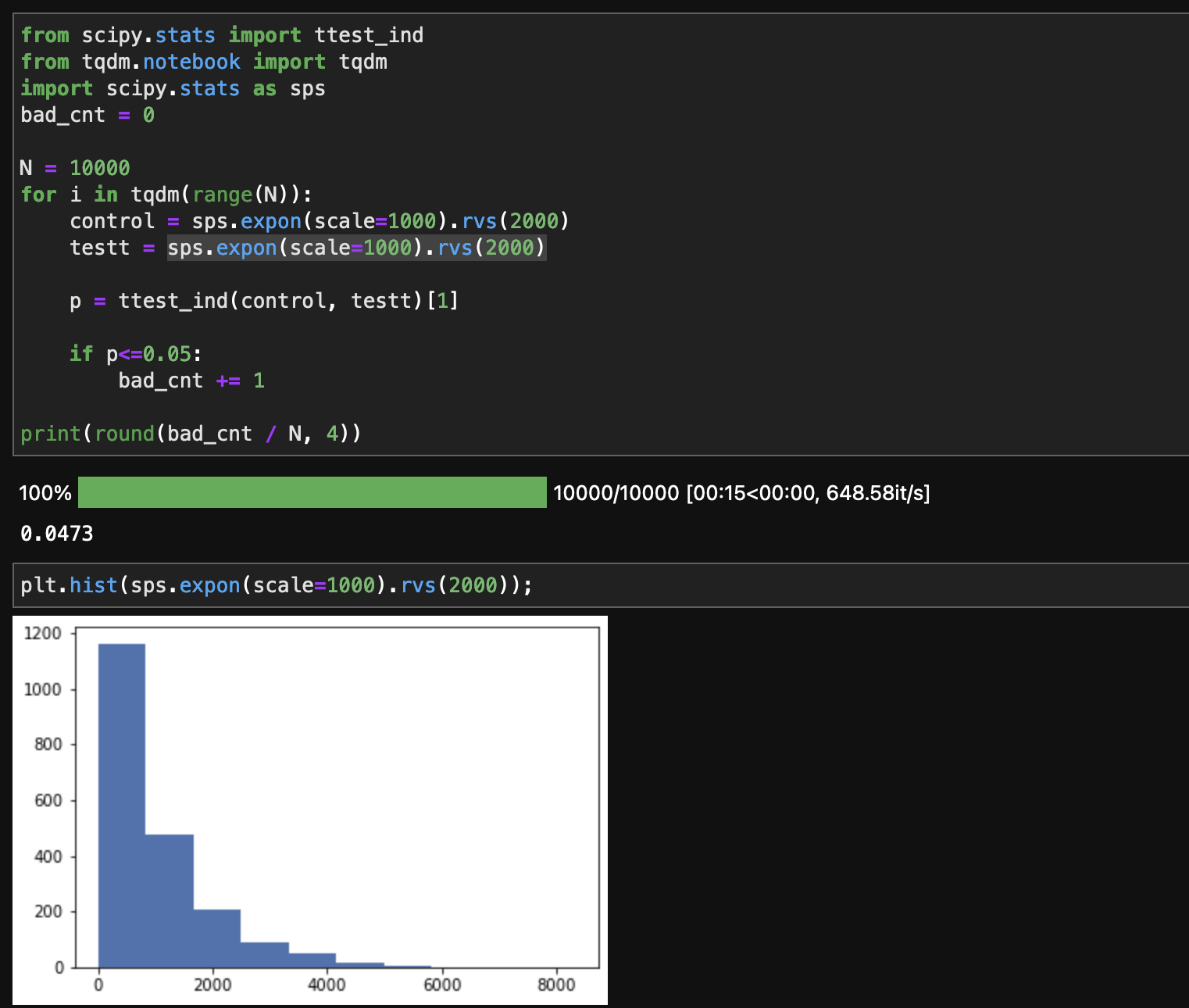
Для этого мы берём значение этой метрики у необходимой части пользователей (например только у пользователей тестируемой платформы) до эксперимента, когда нет разбиения на А и В группы. Далее рандомно бьём этих пользователей на A и В группы, проводим на них Т-тест и фиксируем значение p\_value. Если оно меньше принятого нами (например 0,05), то фиксируем в отдельном счетчике, что эксперимент прокрасился.

Повторяем 10000 раз, каждый раз рандомно разбивая на А и В группы. Если число прокрасившихся экспериментов примерно 5%, то тесту (как и будущим экспериментам) можно доверять.

Если прокрашиваться будет меньше 5%, то это не так критично. Но это значит что метрика не такая чувствительная и мы реже будем фиксировать эффект, там где он реально есть.

Но если прокрашиваться будет значительно чаще установленных 5%, то и в реальных экспериментах мы будем получать ошибки чаще, нежели рассчитывали.

В качестве примера можем взять экспоненциальное распределение, которое похоже на TVTU:



Тут видно что Т-тест вполне подходит для такого распределения

**Результаты**

Согласно A/B тестированию на примере реализованного автозапуска, наша гипотеза сработала: пользователи вовлечены, больше половины из них досматривают фильмы, которые предложил Autoplay.

Общий TVT изменился не сильно, всего на 3,8%, потому что не все пользователи добрались до размеченного контента и увидели фичу. Но результаты среди групп, которые смотрели размещённый контент, отличные. Мы рассчитывали минимум на плюс 5%. Но TVTu сессий оказался выше на 21%. А TVT VOD — выше на 23%.

Самыми сильными были изменения на Android TV.