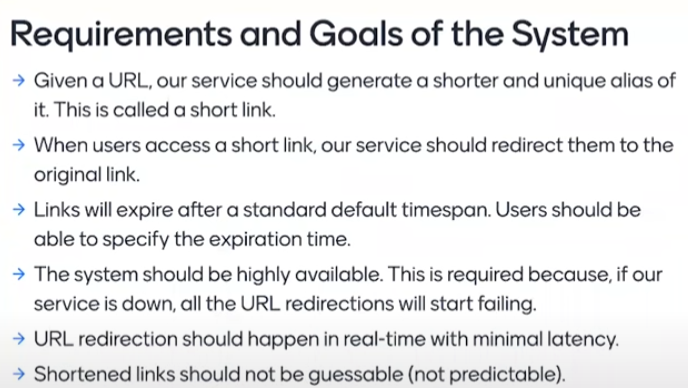
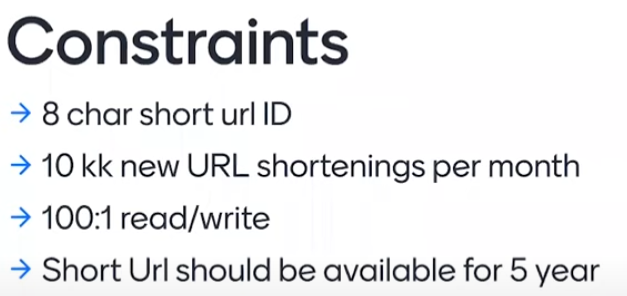
**Сбор требований**



**Лимиты**



**Estimate**

**RPS**

Write – 10 000 000 / (30\*24\*60\*60) = 8 RPS

Read – Write \* 100 = 800 RPS

**DB**

10 000 000 \* 5 years \* 12 months = 600 000 000 записей будет накоплено за 5 лет

600 000 000 \* 2кб(средний вес ссылки) = 1.23 ТБ объем всех данных за 5 лет

**API**

**POST /api/v1/tiny**

* Request {url: initial\_url}
* Response {url: tiny\_url}

**GET /api/v1/tiny**

* Response {url: initial\_url}

Используем base64, с помощью которого сможет сокращать URL. 64 в 8 степени ~ 240 триллионов комбинаций. Это покроет наш запрос на 600 миллионов записей

Выбираемтакой путь, что будем просто генерировать случайное число

**Подходы записи в БД**

PESSIMISTIC\_WRITE – когда мы пытемся записать значение в базу, то мы блокируем базу от других записей, проверяем что такой записи нет в базе, пишем запись и разблокируем базу от других записей. У такого подхода есть минус – когда будет писать в базу несколько юзеров, то у них будут задержки, так как каждый из них будет ждать когда база разблокируется

OPTIMISTIC\_WRITE – при записи в базу мы ничего не блокируем, а если обнаружен дубль, то по новой генерируем значение и пишем снова (в нашем случае это сделать просто, мы просто перегенирируем значение и снова попытаемся его записать). Подходит для тех систем где важна производительность, а так же будет мало конфликтов

**Масштабирование**

Для нашего приложения подойдет load balancer, так как делать микросервисы для такого простого приложения безсмысленно

**БД**