Реализовать брокер очередей в виде веб сервиса.
Сервис должен обрабатывать 2 метода:

1. PUT /queue?v=....

положить сообщение в очередь с именем queue (имя очереди может быть любое), пример:

```
curl -XPUT http://127.0.0.1/color?v=red
curl -XPUT http://127.0.0.1/color?v=green
curl -XPUT http://127.0.0.1/name?v=alex
curl -XPUT http://127.0.0.1/name?v=anna
```

в ответ {пустое тело + статус *200 (ok)}*
в случае отсутствия параметра v - пустое тело + статус 400 (bad request)

2. GET /queue

забрать (по принципу FIFO) из очереди с названием queue сообщение и вернуть в теле http запроса, пример (результат, который должен быть при выполненных put'ах выше):

```
curl http://127.0.0.1/color => red
curl http://127.0.0.1/color => green
curl http://127.0.0.1/color => {пустое тело + статус 404 (not found)}
curl http://127.0.0.1/color => {пустое тело + статус 404 (not found)}
curl http://127.0.0.1/name => alex
curl http://127.0.0.1/name => anna
curl http://127.0.0.1/name => {пустое тело + статус 404 (not found)}
```

при GET-запросах сделать возможность задавать аргумент timeout

```
curl http://127.0.0.1/color?timeout=N
```

если в очереди нет готового сообщения получатель должен ждать либо до момента прихода сообщения либо до истечения таймаута (N - кол-во секунд). В случае, если сообщение так и не появилось - возвращать код 404.

получатели должны получать сообщения в том же порядке как от них поступал запрос, если 2 получателя ждут сообщения (используют таймаут), то первое сообщение должен получить тот, кто первый запросил.

Порт, на котором будет слушать сервис, должен задаваться в аргументах командной строки.

Запрещается пользоваться какими либо сторонними пакетами кроме стандартных библиотек. (задача в написани кода, а не в использовании чужого)

Желательно (но не обязательно) весь код расположить в одном go-файле (предполагается, что решение будет не больше 200 строк кода) для удобства проверки, никаких дополнительных файлов readme и т.п. не требуется, создание классической структуры каталогов (cmd/internal/...) не требуется.

Лаконичность кода будет восприниматься крайне положительно, не нужна "гибкость" больше, чем требуется для решения именно этой задачи, не нужны логи процесса работы программы (только обработка ошибок), никакого дебага и т.д... чем меньше кода - тем лучше!

Оцениваться корректность реализации (заданные условия выполняются), архитектурная составляющая (нет лишних действий в программе, только решающие задачи программы), лаконичность и понятность кода (субъективно, конечно, но думайте о том, насколько будет понятен ваш код для других, это куда более важно в командной разработке, чем сложный "крутой" код).

Если у меня будут вопросы я обязательно задам их на собеседовании.