Необходимо реализовать сервис на фреймворке Spring, который будет отдавать пользователям контент, отсортированный алгоритмом UCB1.

Термины

ContentId — идентификатор контента. Строковое представление UUID.
UserId — идентификатор пользователя. Строковое представление UUID.
Timestamp — время в любом удобном формате, например Unix Timestamp.
Worth – «ценность» контента. Отношение количества лайков к количеству просмотров.

Сервис принимает следующие события:
1) ADD_CONTENT(ContentId, Timestamp)
2) ADD_VIEW(ContentId, UserId, Timestamp)
3) ADD_LIKE(ContentId, UserId, Timestamp)

У сервиса есть HTTP ручка /play/{UserId}, по которой он должен вернуть список из 10 ContentId, отсортированный по алгоритму UCB1. Если контента не хватает, то вернуть сколько есть.

Ограничения

- У контента есть время жизни, после которого он не должен отдаваться (30 минут). Допускается плюс-минус 2 минуты ко времени отсечки.
- В случае перезагрузки сервиса он должен уметь восстановить своё состояние (другими словами, весь контент со всей статистикой за последние 30 минут).
- Контент, который пользователь уже посмотрел, не должен повторно ему отдаваться.
- События просмотров и лайков отправляются пользователем перед тем, как запросить очередную пачку контента.
- В течении 5 минут после отдачи списка контента, если событий просмотра не было, считается, что этот пользователь этот контент просмотрел (т.е. +1 событие ADD_VIEW каждому контенту из списка). После 5 минут ожидания, неподтверждённые просмотры аннулируются (они могут прийти позже — это нормальная ситуация).

Система источник

На стороне источника сообщений необходимо предусмотреть, что у каждого контента есть предопределённый Worth. Сервис не должен о нём знать. Worth для всего корпуса контента распределяется нормально в диапазоне 0.0-0.05 и определяет, с какой частотой пользователи будут слать для него событие ADD_LIKE.

Нагрузка

- 3000 единиц контента одновременно в ротации.
- 10_000 пользователей.
- Для ручки /play/{UserId} базовый RPS=20.

Т.е. раз в 50 ms выбирается случайный пользователь, отправляет просмотры и лайки по ранее полученному контенту и делает запрос.

Задача

а) Продумать архитектуру сервиса, транспорта событий и сохранения состояния.
б) Реализовать сервис.
в) Реализовать инструментарий для тестирования сервиса (в том числе нагрузочного)