One of the ways in which online retailers encourage referrals is by giving their customers a unique coupon code to share with their friends. When this coupon code is subsequently used to make a purchase, both the referrer and new customer get rewarded with a discount.

While email remains the predominant mode of sharing, people are increasingly taking to Twitter to share their coupons. We want you to **design and implement** a **system to track the sharing of these coupons on Twitter**. You should approach the problem like how you would **design a production system** and you may choose to constrain the format of the tweet or the coupon to facilitate this process. Please **include comments** or **a short write up to explain your design decisions**. A working solution that is able to **index new tweets as they are posted** is expected.

Requirements:
1) Retrieve relevant tweets via the twitter API
2) Attribute coupon codes to the correct customer
3) Store information related to the tweet

Additionally, what are some of the **considerations** we should keep in mind **when designing such a system**? Are there **any issues** we might potentially run into? How **can we make this information useful for online retailers**?

----------------------------------------------------------------------------------------------------------------------

## 1. Design a production system to track the sharing of coupon on Twitter

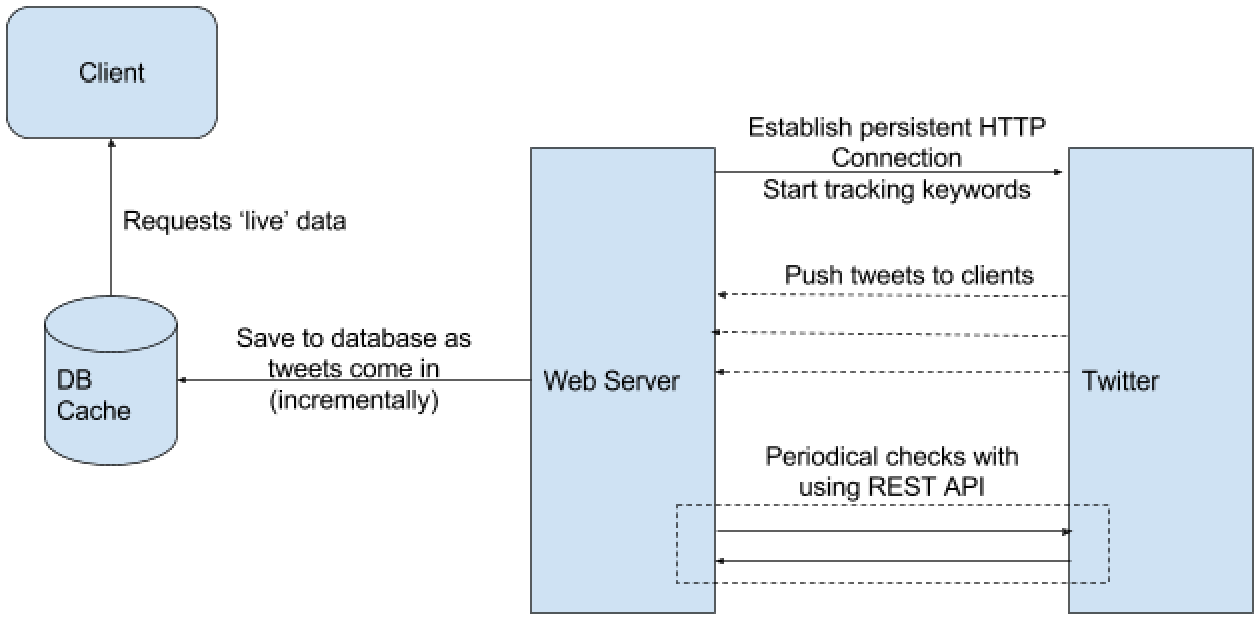
Twitter REST API and Stream API are 2 APIs we can use to access tweets to track coupon.

REST API, specifically search API  will dig up past tweets usually up to 7 days' worth of data. One possible way to approach tracking of coupons could be to schedule regular search queries to gather information (fresh tweets using new coupons or get retweeted tweets) periodically and save to database.

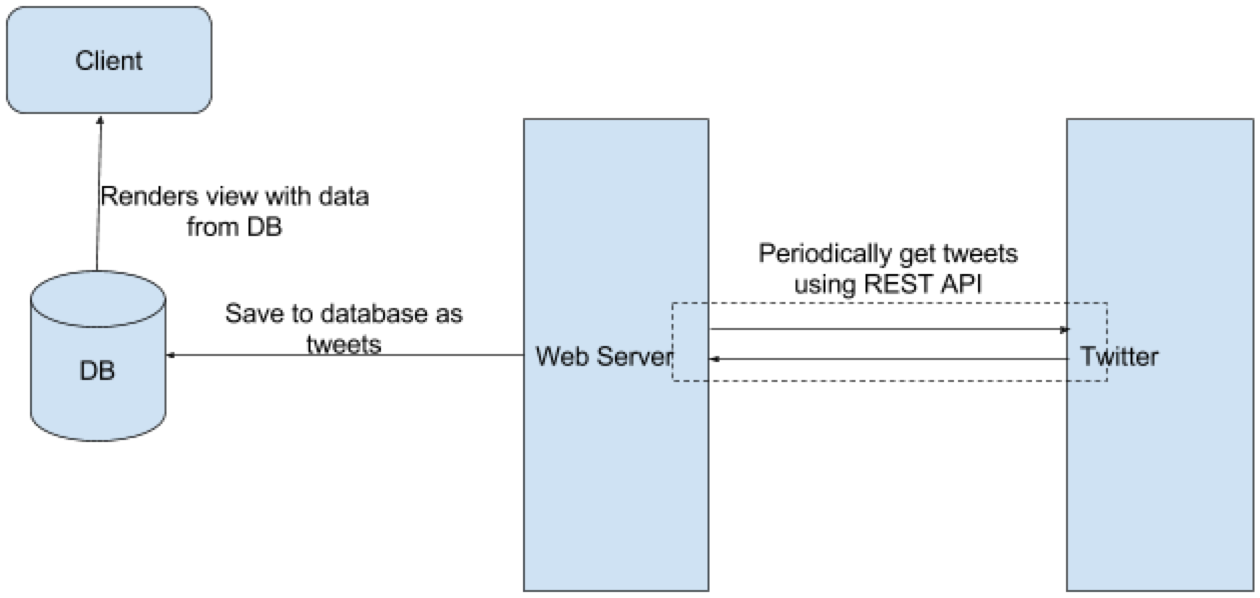
A more fitting approach would be to use Twitter Stream API which processes tweets real-time. We could specify certain keywords to help fetch relevant tweets in this case, "refr.cc", which we assumed to be part of all referrals of Referral Candy. There will be an open HTTP connection to keep tracking new posts.

The solution could be to use them in tandem, where there will be a open HTTP connection for Twitter to push tweets and also have periodical checks using REST API to get past data..

**Ideal System Design**


Client
Requests 'live' data
DB Cache
Save to database as tweets come in (incrementally)
Web Server
Establish persistent HTTP Connection
Start tracking keywords
Push tweets to clients
Twitter
Periodical checks with using REST API

**My Implementation**


Client
Renders view with data from DB
DB
Save to database as tweets
Web Server
Periodically get tweets using REST API
Twitter

I am using twitter library which does not return response data incrementally for Stream API. Tweetstream library supports Streaming API but is not supported in Windows OS (daemonising or ePoll/kqueue). These are libraries built on Ruby. Thus, in view of time, I used the REST API to get the latest tweets which will load periodically every 5 min to capture new unique tweets. I have created 3 tables: Tweet, Contact and Retailer. Tweet table will index all the tweets that contain the keyword “refr.cc”. I have extracted the retailer and coupon code from the url in the tweet. The Contact table specifically shows the coupon code that has been attributed to each customer along with the retailer’s name and retweet count. The Retailer table shows a list of how many times it has been mentioned. The results are not exhaustive as the search function only returns maximum of 100 tweets per page and within the past 7 days.

**Database Schema**

Tweet Table

- Column 1 : Tweet ID (id_str :string)
- Column 2 : Tweet Text (text :string)
- Column 3 : Contact ID (user.id :string)
- Column 4 : Contact Username (user.name :string)
- Column 5 : Referral Link (:string)
- Column 6 : Retailer Name (:string)
- Column 7 : Coupon Code (:string)
- Column 8 : Tweet Timestamp (:datetime)
- Column 9 : Retweet Count (:integer)

Contact Table

- Column 1 : Customer ID (user.id :string)
- Column 2 : Customer Name (user.name :string)
- Column 3 : Coupon Code (:string)
- Column 4 : Retweet Count (:integer)
- Column 5 : Retailer Name (:string)

Retailer Table

- Column 1 : Retailer Name (:string)
- Column 2 : Contact ID (user.id :string)
- Column 3 : Coupon Code (user.name :string)
- Column 4 : Contact Name (:string)
- Column 5 : Tweet Text (:string)
- Column 6 : Retweet Count (:integer)
- Column 7 : Favourite Count(:integer)

As the tweets are being fetched, I grab the URL in the ‘URLS’ array that contains “refr.cc” and get the expanded URL which gives the unwrapped version of the t.co link. I used an Addressable URI library breaks down the URL into parts (hostname, path etc). From this, I can gather the retailer name and coupon code that the contact has tweeted.

The tweets would then be inserted into the Tweet, Contact and Retailer tables. As we receive newer tweets, I would filter out tweets that already exist in the database, based on the composite key (Tweet ID) and insert new tweets along with unique Contact (Customer ID, Customer Name, Coupon Code) and unique Retailer (Retailer Name, Contact ID, Coupon Code).

Streaming API will definitely be easier on the loading of data, as we only get new tweets as they are posted.

After running the server, the root url would show all the tweets (/tweets/index), “/contacts/index” will show the list of contacts, “/retailers/index” will show the list of retailers in the database.

## 2. What are some considerations we should keep in mind when designing such a system?

Using REST API

- We should factor in the limitation that is the API Rate Limit up to 18000 tweets can be requested in 15 minutes. This will be reset only after the 15 minute time period is up.
- Moreover, search API does not return a complete set of all tweets that have been requested. This might mean we could miss out on some referrals tweeted and its journey across Twitter. Especially, as the number of retailers and customers grow, we would not want to miss out on the more tweets which will be the source for further analytics.

Using Streaming API

- For the Twitter Stream API there are limits on number of keywords or IDs that we can follow. Moreover, most of the heavyweight screening of tweets have to be done on our side, as the query for search is not as intuitive
- However, there may not be completeness in data as Twitter only sends tweets anywhere between 1% - 40% of actual number of tweets.

## 3. Are there any issues we might potentially run into?

Streaming API - It said on Twitter website that it will hold the connection open indefinitely except server-side error, excessive client-side lag, network hiccups, routine server maintenance, duplicate logins. So it has to be ensured that the servers on our side be powerful enough to handle longstanding HTTP connections.

Database will be handling mostly text-based data in this case, it may not be too costly at the start, but as the data grows managing millions of tweets will be an impending task. However, with REST API, indexes have to be used to make the performance better to search if the tweet already exists.

Since there is a heavy dependency on Twitter API to furnish us with data, we will have to account to point of failure where Twitter may be down for a few hours or days. Moreover, with the rate limits in place, we have to ensure we make minimal or scheduled API calls.

## 4. How can we make this information useful for online retailers

The large collection of tweets can be valuable overtime for analytics purposes for the retailers to glean customer behaviour and sentiments via data mining by picking out phrases and words used about the product. The retailers will be able to gauge how their product is being assessed by the public. The information on how many times the product has been mentioned, how many retweets there have been and how liked the product is can be deduced. The journey of the coupon can also be tracked where the we can get list of users who have claimed the coupons in real time (using Streaming API). From the list of loyal and potential customers, this could aid in targeted marketing or advertising