**Project Summary**

Create a website that displays a heatmap of rental prices per square foot in an area.

**Project Description**

Use node and 3taps to crawl a set of craigslist geographies (seattle.craigslist.org, sfbay.craigslist.org,…) for apartments based a JSON config file.

The best way to do this appears to be with their polling api (<http://3taps.github.io/3taps-data-commons/polling_api.html>).

I have not quite figured out the semantics of this API but I am sure you can, I have a request into them to clarify its use though and will update you when I hear back.

With that said with the search API (<http://3taps.github.io/3taps-data-commons/search_api.html>) the query looks something like this:

1. Get initial 100 results

<http://api.3taps.com/search/?source=CRAIG&category=RHFR&metro=USA-SEA&status=offered&retvals=price,currency,location,timestamp,body,annotations&rpp=100>

1. Read the number of results from num\_matches along with the anchor.
2. Query repeatedly updating page until you have the results for the related query, e.g.

<http://api.3taps.com/search/?source=CRAIG&category=RHFR&metro=USA-SEA&status=offered&retvals=price,currency,location,timestamp,body,annotations&rpp=100&page=2>

In either case when you get the data back you want to tokenize the body data, you will have to come up with some searches against the body, we basically want to know as much as we can, for example:

* How many square feet (used of not specified by 3taps JSON)
* Number of bedrooms (used of not specified by 3taps JSON)
* Number of bathrooms (used of not specified by 3taps JSON)
* Garage
* Location
  + Address (used of not specified by 3taps JSON)
  + Zip (used of not specified by 3taps JSON)
  + Country (used of not specified by 3taps JSON)
  + City (used of not specified by 3taps JSON)
  + State (used of not specified by 3taps JSON)
  + Latitude
  + Longitude
* Condition
  + New paint
  + New appliances
  + Stainless appliances
  + Impeccable
  + Well Maintained
  + Renovated
  + Nice
  + Newer
* Appliances
  + Dryer
  + Washer
  + Dishwasher
  + Fireplace
* Type
  + Townhome
  + Single family
* Financials
  + Lease Term
  + Deposit
  + Rent (used of not specified by 3taps JSON)
* Pets
  + Cats
  + Dogs

You will have to do some thinking on you do you’re matching for the above, is it just multiple searches for common representations? Edit distance? It doesn’t have to be perfect but reasonable.

In some cases the answer will be concretely returned in the API (for example # of bedrooms) if this isn’t specified in the return you will need to look in the body but if it’s in the return that always takes precedence.

In other cases it may not be possible to reasonably guess from the body, if that’s the case a just use what’s in the API return if available and make note of decision about reliability / difficulty so it can be looked into later.

We may not use all of these but given the goal were just better off capturing them so we can use them later.

If a result doesn’t have an address or lat/long – in other words we can’t place it on the map you can ignore the results.

Some additional things to include in your database schema include:

* Date posted
* Date added to local database

We will want all of this in a PostGres DB so we can run queries later, I might try to model my DB schema based on the 3taps returns and/or their source if online; it seems they have done a good job looking at the data to think about schema though this isn’t purely necessary.

If you are unable to determine something, for example if a unit is a townhome, leave the field in the DB empty.

Expose a JSON interface via node and nginx to query the database based on an address for JSON structure that expresses enough detail to create a heatmap. I suspect this is just latitude / longitude and price per sqft, and address information.

Notice we are not using the other parameters; the thinking is we can use them for a few things later:

1. To compute a quality score – If it’s recently remodeled, has new paint, is a condo/townhome vs an apartment, etc. then it’s a better value; that should be factored into the heatmap; I have not thought about how this would be done much since its ultimately dependent on what data we can get reliably. The value though would be used like a weight. If you can think of a reasonable way to do this please include it in your bid otherwise I can accept a sqft / price as the basis for the heatmap for now.
2. To compute a barrier to entry score – Just an initial idea but if deposits are high or lease terms are high maybe this is “inconvenient” and can be used to weight the value price.

The heatmap can be created with one of these:

<http://www.patrick-wied.at/static/heatmapjs/>

<http://www.heatmapjs.com/>

On a leafletjs map

<http://leafletjs.com/>

Create a simple blank web page that contains nothing more than necessary to show the project works, namely some HTML referencing the Javascript necessary to query your JSON interface on load, draw the leafletjs map with the heatmap.

The web page will also need to try to get location via HTML5’s GeoLocation API, if that fails it will center on a location read from the same configuration file used by the node crawler, that configuration will default to “Seattle, WA”.

HTML and CSS will pass basic W3C validation.

Configure a rackspace server instance with the necessary dependencies (nginx, node, postgres, etc) to host the site.

Configure a cron job to run a tool you will right that to update the db, this will run every 24 hours; I want this to not require handholding so the tool needs to clean up any temporary goo that It may have to create along the way.

Create a brief document describing how to re-configure the server (rpm add foo, configure this, etc).

**Expected Deliverables**

1. Source code zipped.
2. Document describing how to configure.
3. Working website based on project description.
4. Project reasonably fast.
5. All JSON passes jsonlint.com.
6. All CSS passes w3c’s css validator.
7. All HTML passes w3c’s html validator.

**Similar Sites**

1. PaddMapper
2. Zillow
3. Redfin
4. Hipmunk
5. TaskRabbit
6. AirBnb
7. Angieslist
8. Groupon