Assignment # 1

Objective : Build a REST API to fetch the details of a US zip code. The source of the US zip code data can be obtained from http://download.geonames.org/export/zip/US.zip.

Inside of the US.zip file you will find a US.txt file that is tab delimited that contains the following fields:

country code : iso country code, 2 characters
postal code : varchar(20)
place name : varchar(180)
admin name1 : 1. order subdivision (state) varchar(100)
admin code1 : 1. order subdivision (state) varchar(20)
admin name2 : 2. order subdivision (county/province) varchar(100)
admin code2 : 2. order subdivision (county/province) varchar(20)
admin name3 : 3. order subdivision (community) varchar(100)
admin code3 : 3. order subdivision (community) varchar(20)
latitude : estimated latitude (wgs84)
longitude : estimated longitude (wgs84)
accuracy : accuracy of lat/lng from 1=estimated, 4=geonameid, 6=centroid of addresses or shape

Using the data in the format described above, develop the following rest endpoints.

1. Create an API endpoint that when invoked as "curl http://<hostname>/zip/07030" where 07030 is a zip code, it should return the following attributes as a JSON in the response

JSON response:

1. Place name
2. County name
3. State name
4. State code

2. Create an API endpoint called http://<hostname>/zipcounty/count/NJ that will count the number of counties and zip codes for the state initials. Please make sure to gracefully address if the user enters in XX or any other 2 letter state abbreviation that is not valid. A graceful error message should be returned from the endpoint.

JSON response:

{

"stateName": <place name>,

"stateCode": <admin code1>,

"numberOfCounties": <count of number of counties>,

"numberOfZipCodes": <count of number of zip codes>

}

An example JSON response for NJ

{

"stateName": New Jersey,

"stateCode": NJ,,

"numberOfCounties": 21,

"numberOfZipCodes": 722

}

The assignment will be reviewed for the following:

1. Code maintainability, Readability/Easy to follow and understand
2. Modularity
3. Design Principles used
4. Extensible - e.g. Ability to specify the list of attributes to return rather than a static list
5. TDD best practices