Solución Prueba Practica – Bancolombia

Trate de implementar la solución lo más fácil que fuera posible utilizando librerías y herramientas existentes para tal caso, generando así una solución óptima en tiempos de respuesta para procesamiento y consulta, ya que se trata de una cantidad considerable de información.

Para dar esta solución tuve que estudiar el funcionamiento de las librerías utilizadas como el API de google GeoCode para hallar las coordenadas de la dirección ingresada por consola, a su vez del driver de mongo para java, que me permitió implementar la consulta utilizando Geospatial Query, permitiéndome obtener el registro más cercano a las coordenadas ingresadas.

Por falta de tiempo no pude realizar muchas validaciones en el ingreso de la dirección, pienso que el ingreso de una dirección errónea se puede mitigar usando desde el front combos y campos de texto con autocomplete o algo asi, validando la nomenclatura deseada en el ingreso de la dirección. Me hubiera gustado realizar ya sea un reporte en jasperReport mostrando las coordenadas gráficamente en el mapa de google para las pruebas (Ver documento “Pruebas y funcionamiento de Geolocation project”), ya que en las pruebas esto lo verifique de forma manual.

1. ¿Cómo se hizo? (Resumen Gerencial)

En resumen la funcionalidad hace lo siguiente:

* Lee el archivo CSV lo procesa y crea una lista de objetos con especificación GeoJson para luego ser ingresados a la base de datos en mongo.
* Crea el índice en la base de datos de tipo “2dsphere” apuntando a el atributo location, para agilizar la búsqueda.
* Cuando se ingresa la dirección se consulta las coordenadas utilizando GeoCode y se realiza la búsqueda en la base de datos en MongoDB con esta coordenada, utilizando Geospatial Query y así poder mostrar la dirección más cercana.

1.1 ¿Cómo se hizo? (Algo más detallado)

Inicialmente necesitaba poder manipular la data en el archivo CSV, así que mediante la utilización de opencsv convierto la data del archivo a un List de String array, para luego ser procesado en mi aplicación Java.

Sabía de la existencia de consultas en MongoDB mediante la utilización de coordenadas, estudie la utilización de este tipo de querys (Geospatial query) y utilizando el operador $near que me permite arrojar desde un collection de coordenadas, la coordenada más cercana a la ingresada.

De igual forma la consulta requiere que los datos estén de acuerdo a la especificación GeoJson, así que los datos después de ser leídos se utilizan para crear la estructura correcta para su ingreso en la base de datos. Ejemplo:

[{

"address": "KR 83 78 B 15 AP 302",

"location": {

"type": "Point",

"coordinates": [-75.58659646918,

6.28278259085]

}

}]

Mi problema en este punto es que solo tenía la dirección ingresada más no la coordenada de esta dirección, para poder realizar la búsqueda.

Para este fin utilice un GeoAPI de google GeoCode que me permite mediante la dirección y ciudad que siempre será “Medellín, Antioquia”, obtener las coordenadas y poder realizar la consulta.

Cabe anotar que para utilizar esta API necesitamos solicitar a google un KEY, que básicamente es un permiso que google nos provee para el uso del API, de igual manera nos permite diferenciar la solicitud de nuestra aplicación.

**Ejemplo de url generada para la API:**

"https://maps.googleapis.com/maps/api/geocode/json?address=Medellín,+Antioquia"+testAddress+"&region=co&key=AIzaSyANtnE6bbBwWwjyNR\_5iX8QVOdnnGV-Mg0"

1. Descripción de las clases en Geolocation project

La funcionalidad se divide en tres clases a grandes rasgos descritas a continuación:



Esta clase es la encargada de abrir la conexión con la base de datos local en mongoDB, contiene a su vez métodos para la creación de colecciones, creación de índices necesarios para la consulta.

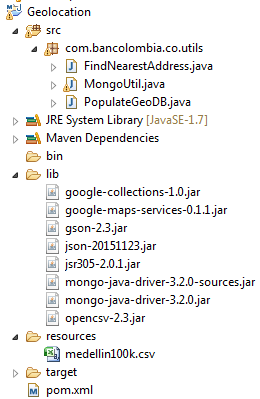


Esta clase es la encargada de poblar la colección geodb, con los datos leídos del archivo CSV.



Esta clase está encargada de solicitar por consola la dirección a consultar, mediante la utilización del GeoAPI de google GeoCode consulto las coordenadas de la dirección entrante, para posteriormente realizar la consulta en la data ingresada desde el archivo CSV, de esta forma obtener la coordenada más cercana y por ende la dirección.

**Vista del proyecto:**



1. Poner en marcha el proyecto

Para descargar el proyecto y documentación, lo puedes hacer desde:

<https://github.com/albertojigar/test-bancolombia.git>

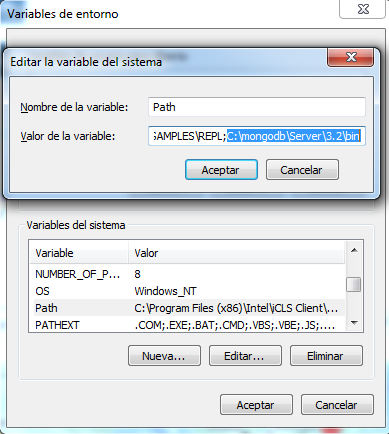
La aplicación es standalone, hecho en eclipse, lo he configurado con maven para el control de versión de las librerías usadas, java 1.7 o superior, usa base de datos MongoDB.

**Requisitos:**

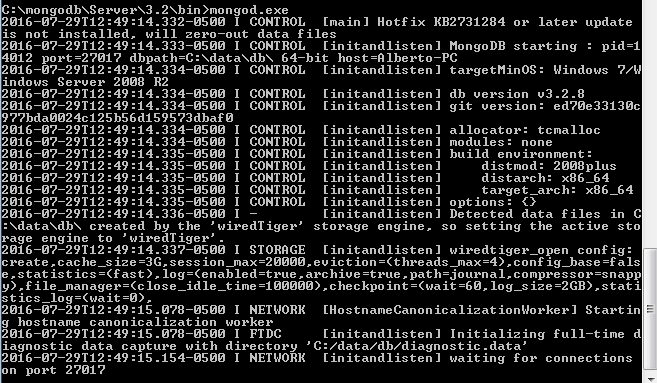
|  |  |  |
| --- | --- | --- |
| Herramienta | Descripción | Link de Descarga |
| Jdk1.7.0\_21 | Virtual machine |  |
| Maven | Versionamiento de librerias |  |
| MongoDB 3.2 | Base de datos no relacional | https://www.mongodb.com/download-center?jmp=hero#community |
| Robomongo 0.9.0 | Cliente Base de datos mongo. (Opcional) | https://robomongo.org/download |
| Eclipse | IDE para desarrollo |  |

**Tips importantes de instalación:**

* En caso de no contar con maven he incluido las librerías en la carpeta lib del proyecto, solo es vincularlas y listo.
* Al instalar MongoDB cambiar la ruta de instalación a C:\mongodb\ (opcional).
* Adicionar en las variables de entorno a la variable Path la ruta de la carpeta bin donde quedo instalado mongodb, algo como: “;C:\mongodb\Server\3.2\bin”. Ver imagen.



* El crea una base de datos por defecto que es local, la aplicación trabaja con ella, no es necesario crear otra.
* Antes de correr la aplicación ponga en marcha la base de datos: Abrir consola de win ingresar a la carpeta bin y ejecutar “mongod.exe”, por favor dejar abierto.



* El API de google GeoCode necesita permisos para conexión por medio de ip, en el proyecto actualmente estoy utilizando el KEY generado en propia cuenta, si necesitan utilizarlo con mi cuenta solo envíenme la ip de donde se está realizando la prueba y yo la registro.

En caso contrario si cuentan con un KEY solo es cambiarla en la línea 42 de la clase FindNearestAddress.java y listo.

GeoApiContext context = **new** GeoApiContext().setApiKey("KEY-A-USAR");

Links de ayuda

**General**

<https://developers.google.com/products/>

**Google-API**

<https://developers.google.com/maps/documentation/geocoding/start?hl=en>

<https://developers.google.com/maps/documentation/geocoding/intro#GeocodingResponses>

**MongoDB**

<https://docs.mongodb.com/manual/reference/operator/query/near/>

<https://docs.mongodb.com/manual/reference/glossary/#term-geospatial>

<https://docs.mongodb.com/manual/core/2dsphere/#create-a-2dsphere-index>

<http://geojson.org/geojson-spec.html>