Entregable 2: justificación técnica

Cálculo de Costos

Para la primer versión de la **Calculadora Logística** se tiene considerado el cálculo del **costo de transporte** de mercancias desde un punto elegido por el usuario hasta alguna de las centrales de abasto de las cuáles se tiene información.

El objetivo de la aplicación será la de brindar la información de cuál es la central de abastos que generará una mayor ganancia a la venta de los productos. Para lograr esto es necesario hacer un cálculo de los costos de producción y de traslado de la mercancía desde el punto de producción/embalaje a los distintos puntos de destino.

Para lograr hacer esto se revisaron diferentes fuentes de información que pudieran generar **rutas** entre dos puntos y que nos diera la información necesaria para poder cruzarla con los datos tales como precios de costos de caseta según tipo de vehículo, y que nos diera información como el tiempo de traslado y los kilómetros de camino.

Después de revisar varias opciones se decidió usar una API de ruteo del INEGI. A pesar de que esta API es relativamente nueva, nos da la información que necesitamos además de que nos permite hacer cruces rápidos con los catálogos de precios de casetas que también tiene el INEGI.

Funcionamiento de la API

Para utilizar la API es necesario generar un *Token* (llave) el cuál puede ser generado en la siguiente liga: http://gaia.inegi.org.mx/sakbe/genera_token.jsp.

Esta llave será utilizada para las diferentes búsquedas que se pueden hacer dentro de la API y no hay restricciones del número de llaves que se pueden generar, aunque solamente es necesaria una. En estos momentos, el Token que estamos utilizando para hacer nuestras pruebas es el siguiente: **IMw3vu3s-tSkM-7Fbx-13si-V1hLQP103alN**

Con esta llave se iran formando url´s con diferentes parámetros, según sea el caso de lo que estemos buscando,en las cuales siempre pondremos como último parámetro el *Token*. Más adelante daremos los ejemplos de la estructura de estos url´s y los parámetros necesarios según sea el caso.

Para poder trazar una ruta se tienen que llevara cabo dos procesos: - Buscar y seleccionar el origen y destino. - Trazar la ruta.

Primero explicaremos el proceso de busqueda y selección de origen - destino La API permite buscar un origen o destino de dos maneras:

- Búsqueda específica de un lugar.
- Búsqueda por coordenadas.

Trazo de ruta

Para poder trazar una ruta se tienen que llevara cabo dos procesos:
- Buscar y seleccionar los puntos de origen.
- Trazar la ruta.

En este caso el usuario solamente tendrá que seleccionar el punto de origen, y el sistema trazará la ruta a las diferentes centrales de abasto y hará el cálculo de costos asociados a cada ruta.

Búsqueda y selección de punto de origen.

Primero explicaremos el proceso de busqueda y selección de origen - destino La API permite buscar un origen o destino de dos maneras:

- Búsqueda específica de un lugar.
- Búsqueda por coordenadas.

Búsqueda específica de un lugar.

Los **parámetros** a utilizar en este caso son: - buscar: es una cadena de caracteres que define el nombre o parte del destino que desea encontrar.

Esto formará un url del tipo:

http://gaia.inegi.org.mx/sakbe/wservice?make=SD&buscar=busqueda&type=json&key=token

Para sistematizar esto se hizo una función llamada *get_ubicaciones()*, que genera la url para hacer la búsqueda de manera automatizada dándole solo el nombre del lugar que estamos buscando.

por ejemplo:

geojson	ent_abr	id_dest	nombre
{“type”:“Point”,“coordinates”:[-102.064042411,19.4214945950001]}	Mich.	27916	Uruapan, Uruapan
{“type”:“Point”,“coordinates”:[-102.072153907,19.426466932]}	Mich.	12633	Parque Urbano Ecológico de Uru
{“type”:“Point”,“coordinates”:[-102.038757761,19.402841622]}	Mich.	36407	Aeropuerto Internacional de Urua
{“type”:“Point”,“coordinates”:[-115.707755809,30.072689048]}	B.C.	24250	Nuevo Uruapan, Ensenada
{“type”:“Point”,“coordinates”:[-116.452818717,31.617410276]}	B.C.	24253	Uruapan, Ensenada
{“type”:“Point”,“coordinates”:[-97.426368738,20.1111334500001]}	Pue.	7429	Uruapan, Ayotoxco de Guerrero
{“type”:“Point”,“coordinates”:[-116.451388501,31.6171727660001]}	B.C.	60694	Familia Pérez (Ejido Uruapan), E
{“type”:“Point”,“coordinates”:[-116.434428302,31.632556847]}	B.C.	8599	Centro Recreativo Ejido Uruapan
{“type”:“Point”,“coordinates”:[-102.044630606,19.425862087]}	Mich.	31700	UNID, Uruapan, Uruapan
{“type”:“Point”,“coordinates”:[-116.45612165,31.6183690990001]}	B.C.	91739	Familia García (Ejido Uruapan),
{“type”:“Point”,“coordinates”:[-115.644710984,30.1030333040001]}	B.C.	103538	Familia Vera (Nuevo Uruapan), E
{“type”:“Point”,“coordinates”:[-108.324439101,27.3392429200001]}	Chih.	59769	Los Llanos de Uruapan, Guazapa
{“type”:“Point”,“coordinates”:[-111.657777797,28.7958333120001]}	Son.	47904	Uruapan (Bomba Negra), Hermos
{“type”:“Point”,“coordinates”:[-98.601682529,22.12360448]}	S.L.P.	110420	Uruapan (El Naranjal), Tamuín
{“type”:“Point”,“coordinates”:[-102.072985057,19.475403316]}	Mich.	31753	ITESU, Uruapan, Uruapan
{“type”:“Point”,“coordinates”:[-102.047276924,19.4259102120001]}	Mich.	31695	UDV, Uruapan, Uruapan
{“type”:“Point”,“coordinates”:[-102.077337567,19.528026574]}	Mich.	31757	UPU, Uruapan

Esta función deberá de implementarse en un menú de búsqueda dentro de la aplicación.

Búsqueda por coordenadas.

En caso de que la ubicación específica de la que queremos partir no exista, la aplicación tendrá un mapa en el cuál se podrá poner un pin desde tu ubicación actual o algún lugar mediante un pin (como en uber). Para hacer la ruta usando las coordenadas se generó la función *get_coord(lon,lat)* la cuál nos da información similar a la función *get_ubicaciones()* que definimos arriba.

La función se puede encontrar de manera detallada en el archivo *get_origendestino.R*. Un ejemplo del uso y las salidas de esta función es:

geojson	source	id_routing_net	nombre	targ
{“type”:“Point”,“coordinates”:[-99.0050716435404,19.6174030724993]}	158339	99589	Avenida Central	1583

Obtener ruta e información de costos

Una vez que se ha definido el punto de origen por cualquiera de los dos métodos arriba mencionados, se obtendrá el trazo de la ruta. Los parámetros que los usuarios podrán escoger son:

- tipo_vehiculo: especifica el tipo de vehículo
- ejes: especifica el número de ejes excedentes. Es opcional, si se omite el valor por default será 0.

Además de estas, hay que configurar: - proj: es el tipo de sistema de referencia, entre los valores permitidos están GRS80 (Geográfica) y MERC (Spherical Mercator). Este parámetro es opcional, si se omite el valor por default será GRS80. - type: tipo de formato a regresar (json o xml). - key: número único que permite acceder a la API.

A continuación presentamos un ejemplo de los datos que entran y la información que se obtiene

```
## [1] "El costo de las casetas es: 1423 mxn."
```

```
## [1] "El costo por eje excedente es: 549 mxn."
```

```
## [1] "El viaje es de un total de : 398.77 kilómetros totales"
```

```
## [1] "La duración del viaje es aproximadamente de : 245.23 minutos"
```

Con esta información podemos fácilmente calcular costos:

Costos de peaje:

$$Peaje = CostoCaseta + EjeExcedente$$

Costos de gasolina:

Para realizar el cálculo del costo de gasolina se considera un rendimiento de 3 kms por litro de acuerdo a una pequeña investigación que se realizó *https://imt.mx/archivos/Publicaciones/PublicacionTecnica/pt368.pdf*

Para calcular el costo de gasolina se hixo una función que obtiene el precio promedio de los diferentes combustibles, esto es:

```
##     tipo costo     tipo-costo
## 1 Diésel 20.61 Magna $20.61/l
```

Entonces el cálculo de costo de gasolina se da con la siguiente fórmula:

$$GastoGas = \frac{Kms}{3} * PrecioCombustible$$

Así, el costo total en transporte sería de :

$$Costo = Peajes + GastoGas$$

Así podríamos calcular los costos de transporte.

Cálculo de precios

Para el cálculo de precios se tiene como base el código de scrapping de la página del SNIIM. Este código nos permite hacer pedidos de información a la página con ciertos parámetros y transformar las tablas que devuelve de formato html a una tabla de Python la cuál puede ser manipulada para hacer agregaciones y calcular precios.

A continuación se presenta un diagnóstico general de este método:

Datos disponibles

El SNIIM tiene varias maneras de consultar la información. Para nuestros fines la página cuenta con tres secciones en las cuáles **podemos encontrar los precios de los 68 productos**. Estas secciones son:

- Frutas y hortalizas.
- Granos.
- Precuarios.

El script de scrapping original tiene la estructura para manejar las secciones de *Frutas y hortalizas* y *Granos* y se adaptó la sección de *Precuarios*

El proceso por el que pasa cada consulta de un cultivo es la siguiente:
- Se busca la información el cultivo en una central.
- Se detectan la tabla con la relación de cantidades y costos.
- Se detecta la cantidad que se está manejando (kilos, cajas, etc.)
- Se estandarizan los precios a kilos.

Finalmente con esto se hace el cálculo:

$$Ingreso = PrecioDelProducto * CantidadDeProducto$$