Algoritmo de Selección Automática de Ajustadores de Siniestros en CHUBB

Documentación de Solución e Implementación Recomendada

Autores:

Diego Padilla Vélez
Daniel Eduardo Martínez Martínez

Índice

Repositorio de Código

A continuación se indica el repositorio en donde se puede encontrar el código desarrollado para la solución propuesta para el algoritmo y API, así como la implementación recomendada a la aplicación actual en cabina:

URL del Repositorio:
https://github.com/DanielMartinez98/practicas

En la carpeta de “Pruebas” se encuentra el código fuente del Algoritmo de Selección Automática de Ajustadores, y en la carpeta “Visual Basic” se encuentra el entorno de pruebas desarrollado y la implementación sugerida.

Algoritmo

El Algoritmo del API está compuesto en múltiples partes, de las cuales las más importantes son las siguientes.

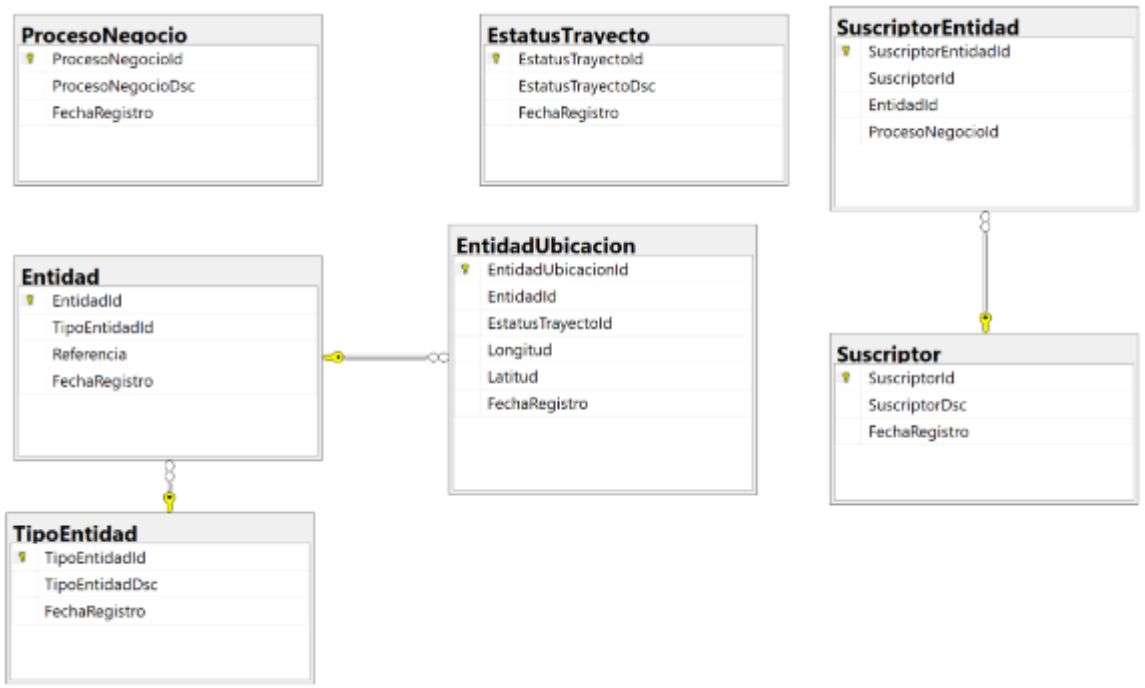
1.) Llamada a la base de datos

Primeramente el algoritmo llama a la base de datos utilizando las siguientes instrucciones para crear la conexión:

1. ubica el ip correspondiente del Api y su método para las prácticas el api se ve de esta forma. http://0.0.0.0:5007/api/AsignarAjustadorController2
2. enviar un json con el siguiente formato
   -{"dir": "Escobedo+Nuevo+Leon+Harlingen+Residencial+Puerta+del+Norte"}
3. se recibirá un json con el formato {“resultado”: ”7”}

Los datos relevantes para la conexión deberán ser introducidos para realizar una conexión a la base de datos utilizada en CHUBB. El algoritmo no realiza ninguna escritura o modificación a los datos solicitados.

La estructura de la base de datos ideal para la implementación del algoritmo y API es la siguiente:



Después de que la conexión a la base de datos está establecida, se realiza la siguiente consulta usando SQL:

```
1. SELECT
   Ajustador.EntidadId,Ajustador.disponibilidad,Ajustador.tipo_a
   justador,EntidadUbicacion.Latitud,EntidadUbicacion.Longitud
2. FROM Ajustador,EntidadUbicacion
3. WHERE Ajustador.EntidadId = EntidadUbicacion.EntidadId;
```

Dentro del algoritmo se realiza una llamada utilizando la consulta anterior, pero se sugiere utilizar un *stored procedure* dentro de la base de datos utilizada.

2.) Llamadas a APIs de Google Maps

Dentro del algoritmo se realizan llamadas a distintos APIs de Google Maps. En las llamadas a todos estos API se debe cambiar la API_KEY por la llave apropiada.

En el algoritmo, primeramente se realiza una llamada al API de Geocoding de Google Maps para poder extraer la latitud y longitud aproximada de una dirección por líneas, recibida cuando se realiza la llamada al API que se busca implementar en la aplicación en cabina. El resultado de esta llamada es una localización geográfica (que consiste en latitud y longitud) que será utilizado en partes posteriores del algoritmo. En caso de la llamada al API para la integración a aplicación móvil, no se utiliza este API y se continúa con el siguiente paso.

Posteriormente se hace una llamada al API de Google Distance Matrix con todos los valores de “latitud y longitud” y de los ajustadores donde los compara con la localización geográfica obtenida por la llamada al API de Geocoding o la provista, para poder obtener el tiempo y la distancia estimados que se tomaría para que cada ajustador atienda al siniestro en la ubicación del cliente. De esta llamada se recibe un JSON con los valores relevantes que incluyen la distancia y tiempo estimados, cuyos valores después se extraen de la respuesta, y se añade a un arreglo de los ajustadores a contemplar.

3.) Lectura de Pesos

Para poder usar al algoritmo el sistema debe leer el archivo *pesos.txt* donde se almacenan tres líneas de texto, cada una con los pesos para las siguientes variables, en orden consecutivo.

- **Distancia “Incluyendo Tráfico”**
  - Peso predeterminado: 50
- **Disponibilidad del Ajustador**
  - Peso predeterminado: 40
- **Tipo de Ajustador**

- Peso predeterminado: 10

4.) Algoritmo de Selección

Dentro del algoritmo de selección se calcula lo apropiado que es un ajustador con base en las variables mencionadas anteriormente, utilizando las siguientes instrucciones:

```
1. counts += (p1 * ((1.0f * minim) / Int32.Parse(array[i][5])));
2. counts += p2 * (Int32.Parse(array[i][1]) / 3.0f);
3. count += p3 * (Int32.Parse(array[i][2]));
```

A partir de esto, el ajustador con mayor peso se sugiere como la mejor opción, y el algoritmo regresa el valor del ID del ajustador más apropiado.

5.) Concurrencia

Debido a que pueden existir casos de concurrencia dentro de cabina cuando se estén generando más de un reporte a la vez, se implementó un documento con las últimas sugerencias realizadas por el algoritmo.

Los ID de estos ajustadores se almacenan en el archivo *asignados.txt* donde se guardan por 5 minutos para que los agentes de cabina tengan tiempo para terminar la generación del reporte y se pueda asignar a los ajustadores reflejando la disponibilidad correcta evitando asignaciones múltiples.

Interfaz de Programación de Aplicaciones (API)

Levantamiento de API

Para el levantamiento del API se requiere que en el archivo app *settings.json* se asignen los puertos que se desea asignar junto con el URL que se desea usar para hospedar el API.

A continuación se muestra el formato a seguir para el archivo:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "Kestrel": {
      "Endpoints": {
        "Http": {
          "Url": "http://[######0]"
        },
        "Https": {
          "Url": "https://[######1]"
        }
      }
    }
 }
```

En donde se debe cambiar [######0] por el URL deseado para hospedar el API bajo HTTP, y [######1] por el URL deseado para hospedar el API bajo HTTPS.

Llamada a API mediante Dirección

Una implementación de la API desarrollada tiene como entrada un JSON que incluye la dirección en líneas introducida en la aplicación actual en cabina.

El cuerpo del JSON requerido por la API tiene la siguiente estructura:

```
[ { "dir" : "#########" } ]
```

En donde ######### representa una dirección por líneas, cambiando espacios por el carácter “+”. Se sugiere que la dirección siga el formato consecutivo siguiente:

(Estado y Municipio) + (Colonia) + (Calle) + (Código Postal)

El uso de mayúsculas y minúsculas en estos datos es indistinto.

Ejemplo:

```
[ { "dir" : "Escobedo+Nuevo+Leon+Puerta+Del+Norte+Harlingen+66054" } ]
```

A continuación se presenta el *header* de petición sugerido para realizar la petición al API montado en un servidor:

Content-Type	application/json
Accept	*/*
Accept-Encoding	gzip, deflate, br

Llamada a API mediante Geolocalización

Una implementación de la API desarrollada tiene como entrada un JSON que incluye la latitud y la longitud que puede ser solicitada al GPS del usuario de la aplicación móvil en caso de implementarse.

El cuerpo del JSON requerido por la API tiene la siguiente estructura:

```
[ { "lat" : "#########", "lon" : "#########" } ]
```

En donde ######### representa la latitud o longitud de la geolocalización del afectado por el siniestro.

Ejemplo:

```
[ { "lat" : "25.778103", "lon" : "-100.288582"} ]
```

A continuación se presenta el *header* de petición sugerido para realizar la petición al API montado en un servidor:

Content-Type	application/json
Accept	*/*
Accept-Encoding	gzip, deflate, br

Aplicación en cabina

Función Sub para llamada a API

Para la implementación del API en la aplicación en cabina se requiere la creación de una función Sub para realizar la conexión y petición relevante para obtener el resultado. Por cuestiones de nomenclatura uniforme, se sugiere que la función Sub lleve por nombre *API_SeleccionAutomaticaAjustador.* A continuación se presenta el código sugerido para la implementación del API en cabina:

Para consulta del código sugerido, revisar la sección de **Repositorio de Código.**

En la siguiente sección, cambiar el componente "[######0]" por el URL de donde se encuentra hospedada la **API de Selección Automática de Ajustador**.

```
1. Sub API_SeleccionAutomaticaAjustador()
2. Dim API_SeleccionAutoAjustador_HostAPI As String,
   API_SeleccionAutoAjustador_EstadoMunicipio As String,
   API_SeleccionAutoAjustador_Colonia As String, API_SeleccionAutoAjustador_Calle
   As String, API_SeleccionAutoAjustador_CodigoPostal As String
3. API_SeleccionAutoAjustador_HostAPI = "[######0]"
```

En la siguiente sección, cambiar el componente "[######1]" por el componente que representa la línea en donde se introduce el **Estado / Municipio** en la aplicación en cabina.

```
4. API_SeleccionAutoAjustador_EstadoMunicipio = [######1].Text
5. API_SeleccionAutoAjustador_EstadoMunicipio =
   Replace(API_SeleccionAutoAjustador_EstadoMunicipio, " ", "+")
```

En la siguiente sección, cambiar el componente "[######2]" por el componente que representa la línea en donde se introduce la **Colonia** en la aplicación en cabina.

```
6. API_SeleccionAutoAjustador_Colonia = "+" & [######2].Text
7. API_SeleccionAutoAjustador_Colonia =
   Replace(API_SeleccionAutoAjustador_Colonia, " ", "+")
```

En la siguiente sección, cambiar el componente "[######3]" por el componente que representa la línea en donde se introduce la **Calle** en la aplicación en cabina.

```
8. API_SeleccionAutoAjustador_Calle = "+" & [######3].Text
9. API_SeleccionAutoAjustador_Calle = Replace(API_SeleccionAutoAjustador_Calle, "
   ", "+")
```

En la siguiente sección, cambiar el componente "[######4]" por el componente que representa la línea en donde se introduce el **Código Postal** en la aplicación en cabina.

```
10. API_SeleccionAutoAjustador_CodigoPostal = "+" & [######4].Text
```

La sección presentada a continuación, representa la petición al servidor de la API por parte de la aplicación de cabina utilizando *WinHttpRequest*. El formato estándar de la petición se ve definido en la documentación de la propia API.

```
Dim API_SeleccionAutoAjustador_RequestObject As Object
Dim API_SeleccionAutoAjustador_ResponseString As String
Dim API_SeleccionAutoAjustador_RequestBody As String
Set API_SeleccionAutoAjustador_RequestObject =
CreateObject("WinHttp.WinHttpRequest.5.1")
API_SeleccionAutoAjustador_RequestBody = "{ ""dir"" : "
API_SeleccionAutoAjustador_RequestBody = API_SeleccionAutoAjustador_RequestBody
& " "" " & API_SeleccionAutoAjustador_EstadoMunicipio &
API_SeleccionAutoAjustador_Colonia & API_SeleccionAutoAjustador_Calle &
API_SeleccionAutoAjustador_CodigoPostal & " "" }"
```

En la siguiente sección, se pueden agregar más etiquetas al *header* de la petición, para reflejar los necesarios si se hace alguna modificación al API.

```
With API_SeleccionAutoAjustador_RequestObject
.Option(4) = 13056
.Open "POST", API_SeleccionAutoAjustador_HostAPI, False
.SetRequestHeader "Content-Type", "application/json"
.Send API_SeleccionAutoAjustador_RequestBody
API_SeleccionAutoAjustador_ResponseString = .ResponseText
End With
```

En la siguiente sección se hacen los cambios necesarios sobre el JSON recibido después de la llamada al API. Se almacena el ID del ajustador recomendado en la variable *API_SeleccionAutoAjustador_IDAjustadorApropiado* como un valor entero.

```
API_SeleccionAutoAjustador_ResponseString =
Replace(API_SeleccionAutoAjustador_ResponseString, """", "")
API_SeleccionAutoAjustador_ResponseString =
Replace(API_SeleccionAutoAjustador_ResponseString, "resultado", "")
API_SeleccionAutoAjustador_ResponseString =
Replace(API_SeleccionAutoAjustador_ResponseString, "[", "")
API_SeleccionAutoAjustador_ResponseString =
Replace(API_SeleccionAutoAjustador_ResponseString, "]", "")
API_SeleccionAutoAjustador_ResponseString =
Replace(API_SeleccionAutoAjustador_ResponseString, "{", "")
API_SeleccionAutoAjustador_ResponseString =
Replace(API_SeleccionAutoAjustador_ResponseString, "}", "")
API_SeleccionAutoAjustador_ResponseString =
Replace(API_SeleccionAutoAjustador_ResponseString, ":", "")
Dim API_SeleccionAutoAjustador_IDAjustadorApropiado As Integer
API_SeleccionAutoAjustador_IDAjustadorApropiado =
CInt(API_SeleccionAutoAjustador_ResponseString)
```

La última sección representa la detección de un error si el API retorna el valor de 0 como su ID de ajustador recomendado. De lo contrario, se puede trabajar con el valor obtenido para distintos componentes de la aplicación de cabina. El valor es almacenado en *API_SeleccionAutoAjustador_IDAjustadorApropiado,* y en caso de

que se desee utilizar este valor en algún otro componente se debe reemplazar la línea 36 por el código que comprenda estas operaciones.

```
33. If (API_SeleccionAutoAjustador_IDAjustadorApropiado = 0) Then
34. MsgBox "Ocurrió un error con el API de Selección Automática. Registre el
    ajustador de manera manual."
35. Else
36. MsgBox "El ajustador seleccionado es el de ID " &
    API_SeleccionAutoAjustador_ResponseString
37. End If
38. End Sub
```

Implementación a Botón en Aplicación de cabina

Se sugiere la implementación de un botón dentro de la interfaz en la aplicación en cabina donde se haga llamada al procedimiento Sub comprendido en la sección anterior.

Conforme a la nomenclatura utilizada anteriormente, se sugiere que lleve el nombre de *API_SeleccionAutomaticaAjustador_BotonUI*. Posterior a la creación del botón, se debe de crear una función Sub referente a la acción del *click* en el mismo.

A continuación se presenta la implementación sugerida:

```
1.  Private Sub API_SeleccionAutomaticaAjustador_BotonUI_Click()
2.      API_SeleccionAutomaticaAjustador
3.  End Sub
```