Informe del Algoritmo para el Cálculo del Precio del Servicio por Viaje

Autor: IAM

Índice

1 Informe Algoritmo Cálculo Precio

1.1 Introducción.

El presente documento tiene como objeto la descripción genérica del algoritmo de cálculo del servicio de viaje llamado “Servicio precio cerrado” que devolverá un precio cerrado para un servicio de viaje.

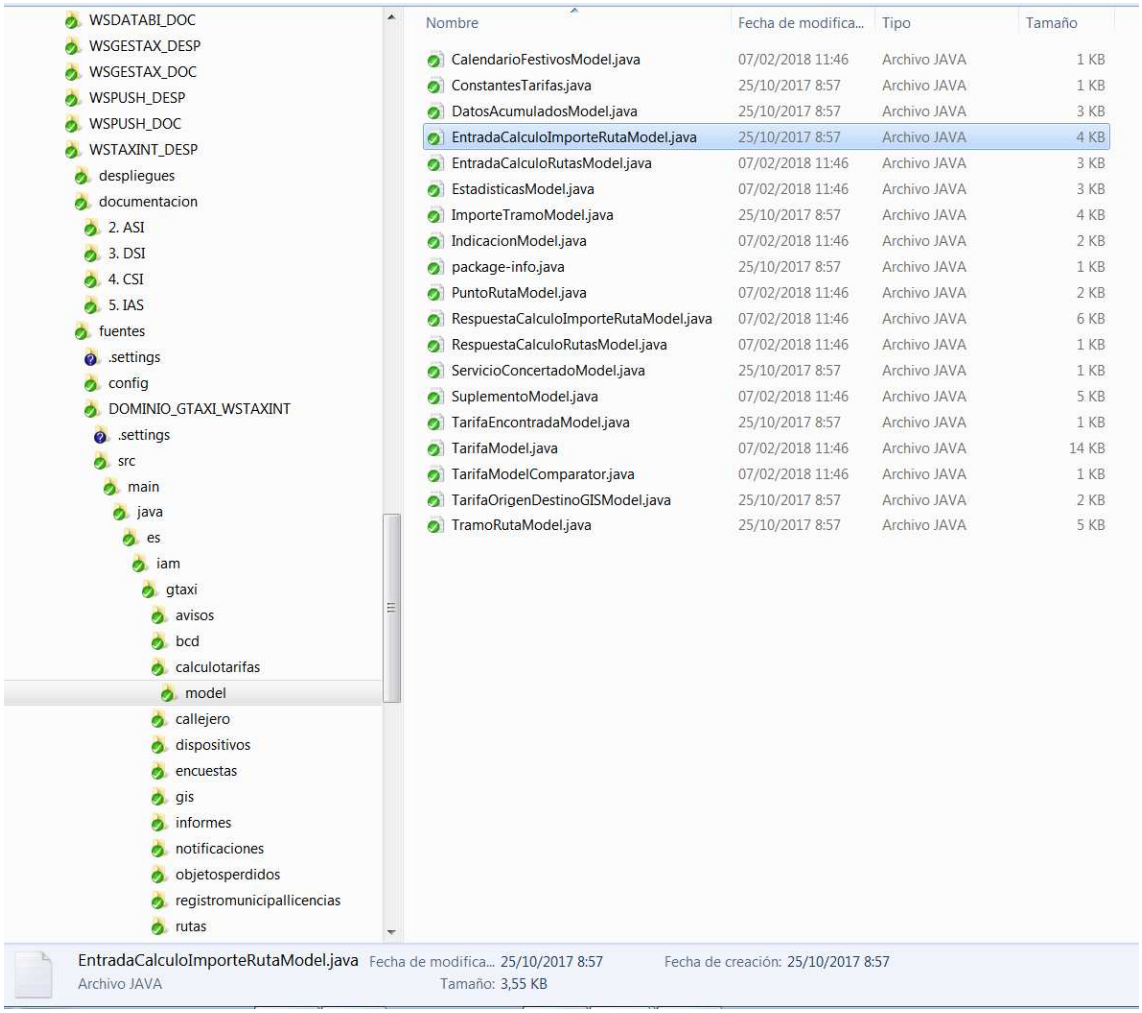
1.2 Descripción del Algoritmo y referencia de cada proceso a la parte del código.

El algoritmo utilizado se encuentra definido en los flujogramas que se adjuntan. Estos flujogramas definen una serie de procesos partiendo del Flujograma principal **CalculoImporteRuta.**

Para el cálculo del importe de la ruta son necesarios unos elementos de cálculo que se encuentran definidos a su vez en subprocesos involucrados, como Obtener y comprobar Tarifas, calcular Tramos, importes, suplementos y Tramos Franquicia. Estos subprocesos son necesarios a la hora de calcular un importe total y se encuentran definidos en los flujogramas secundarios:

1.2.1 *Descripción de los datos de entrada*

Los datos de entrada del flujo principal están definidos en el Modelo de datos en la ruta:



En la cual se define el tipo de los datos de entrada:

- OrigenRuta - private PuntoRutaModel origen
- DestinoRuta - private PuntoRutaModel destino
- FechaInicioRuta - private Date fechaHoraInicio
- HoraInicioRuta - private Date fechaHoraFin
- ParadaRuta - private Boolean inicioEnparada
- ServicioRutaContratado - private Boolean servicioContratado = Boolean.FALSE

PuntoRutaModel {

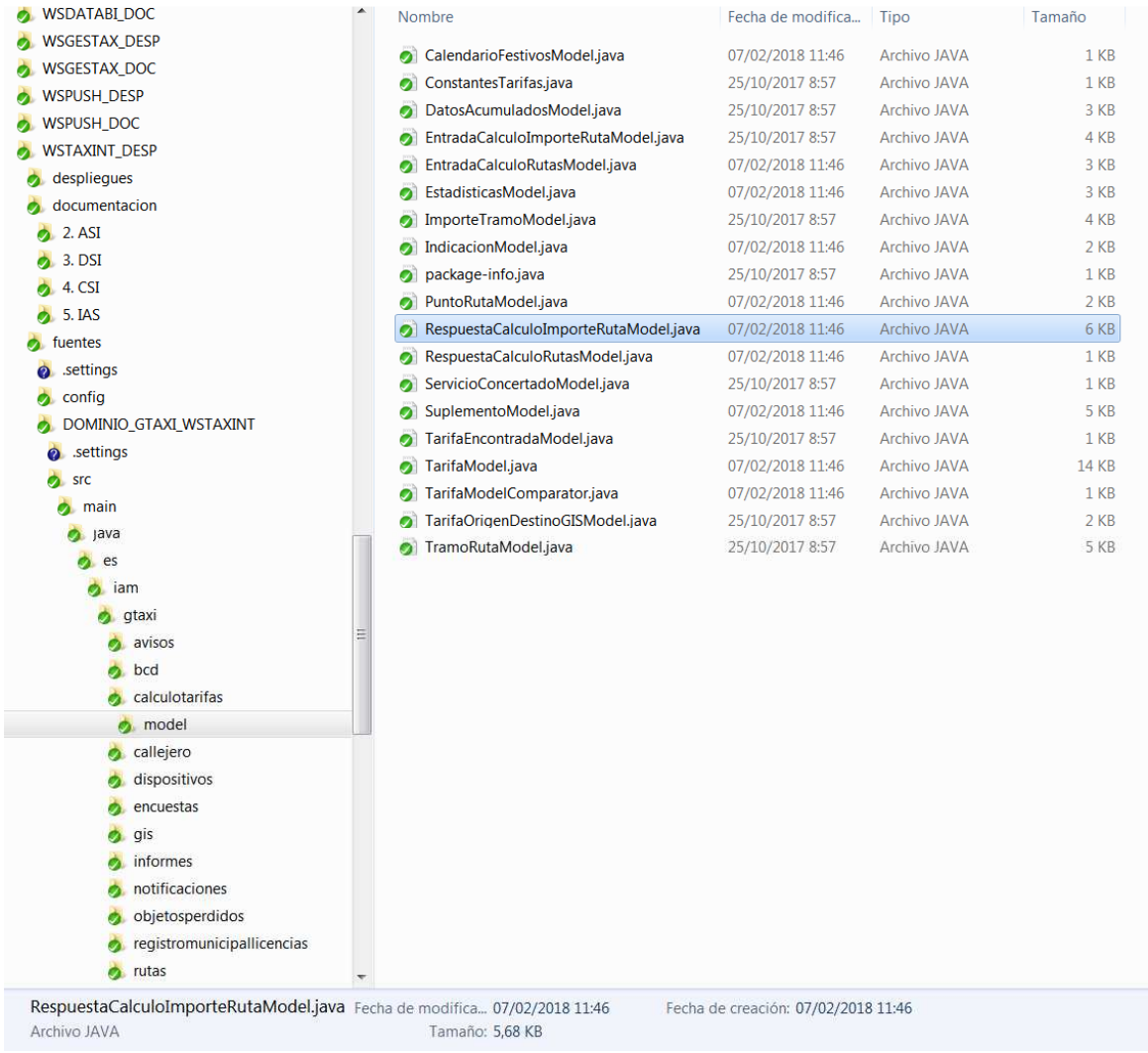
- private List<Integer> **ubicacion** = null;
- private List<Integer> **ubicacionGis** = null;}

Además de los anteriores datos, también se necesitan los datos proporcionados por GIS:

Apc Origen
Apc Destino
Apc Tramo
Distancia Tramo
Tiempo Tramo

1.2.2 *Descripción de los datos de salida.*

Los datos de salida del flujo principal están definidos en el Modelo de datos en la ruta:



1.2.3 *Descripción de los pasos principales del algoritmo (diagrama1).*

El cálculo principal está contenido en el bean **CalculoTarifasEJBLNImpl** en el método calcularImporte. Este método recupera las tarifas vigentes, lee los tramos y acumula los importes, devolviendo el resultado de los mismos.

1.2.3.1 Comprobar que todos los datos están dentro de la APC

Se comprueba previamente que tanto el origen como el destino están dentro de la APC.

1.2.3.2 Recuperar la lista de tarifas y el calendario de festivos.

Se recuperan las tarifas en el método obtenerTarifa del bean **CalculoTarifasEJBLNImpl.** Devuelve una lista de tarifas.

1.2.3.3 Obtención de la tarifa aplicable en la ruta (diagrama 2, 3 y 4)

1.2.3.3.1.1 Obtener la tarifa aplicable (diagrama 2)

Se filtran las tarifas en base al origen y destino de la tarifa y el origen y destino de la ruta. En base a las mismas se llama al proceso comprobar Tarifa hasta obtener la tarifa adecuada.

1.2.3.3.1.2 Comprobar la tarifa (diagrama3)

Obtiene los datos de la tarifa en el formato **TarifaModel**:

```
public final static String DIA_FESTIVO_SI = "S";
public final static String DIA_FESTIVO_NO = "N";
public final static String DIA_FESTIVO_TODOS = "T";

public final static String HORA_INICIO_POR_DEFECTO = "6:00";
public final static String HORA_FIN_POR_DEFECTO = "6:00";

private String id = null;
private Integer anio;
private Integer tipoTarifa = null;
private String diaSemana = null;
```

```
        private BigDecimal franquiciaKm = BigDecimal.ZERO;
        private BigDecimal franquiciaTiempo = BigDecimal.ZERO;
        private BigDecimal velocidadArrastre = BigDecimal.ZERO;
        private Boolean tieneFranquicia = Boolean.FALSE;
        private BigDecimal importeInicioViaje = BigDecimal.ZERO;
        private BigDecimal importeKm = BigDecimal.ZERO;
        private BigDecimal importeHora = BigDecimal.ZERO;
        private BigDecimal importeMaxZonaA = BigDecimal.ZERO;
        private BigDecimal importeMaxZonaB = BigDecimal.ZERO;
        private Boolean tieneImporteKm = Boolean.FALSE;
        private Boolean tieneImporteHora = Boolean.FALSE;
        private Boolean errorTarifa = Boolean.FALSE;
        private Boolean admiteCuantiaServicioContratado =
Boolean.FALSE;
        private Boolean admiteSuplementos = Boolean.FALSE;
        private List<SuplementoModel> suplementos = new
ArrayList<SuplementoModel>();
        private Boolean disponibleSoloEnParada = Boolean.FALSE;
        private Boolean disponibleServiciocontratado = Boolean.FALSE;
        private String tipoDiaFestivo;
        private String horaInicio;
        private String horaFin;
        private Date fechaEntradaVigor;
```

1.2.3.3.1.3 Comprobar el día (diagrama 4)

Se comprueba si el día es festivo y si la tarifa es coherente en relación a los días festivos.

Se comprueba si la hora de inicio de la ruta y la hora de aplicación de la tarifa es coherente.

Devuelve un booleano.

1.2.3.4 Aplicar la tarifa inicial y el importe inicial

En el método CalcularImporte devuelve un BigDecimal resultado de multiplicar la distancia en Kms por el importe por Km.

1.2.3.5 Cálculo del importe de los tramos de la ruta (diagramas 5, 6 y 7)

1.2.3.5.1.1 Cálculo del importe de un tramo (diagrama 5)

Este proceso obtiene el importe a partir de la tarifa de inicio del tramo y la tarifa de fin del tramo

Devuelve un objeto de la clase ImporteTramoModel con las propiedades:

```
        private BigDecimal importe = BigDecimal.ZERO;
        private BigDecimal importeOrigenTramo = BigDecimal.ZERO;
```

```
private BigDecimal importeFinTramo = BigDecimal.ZERO;
private TarifaModel tarifaOrigenTramo;
private TarifaModel tarifaFinTramo;
private Boolean errorVariasTarifas = Boolean.FALSE;
private Boolean errorTarifaNoEncontrada = Boolean.FALSE;
private long tiempoFinTramo = 0;

private long distanciaFinTramo = 0;
```

1.2.3.5.1.2 Cálculo del importe de un tramo con franquicia (diagrama 7)

Calcula el importe del tramo Franquicia cuando el acumulado de la distancia + la distancia del tramo es superior a la Franquicia por Km, o cuando el Acumulado de la distancia + la distancia del tramo es mayor que la Franquicia por km.

Devuelve un objeto de la clase ImporteTramoModel

1.2.3.5.1.3 Cálculo del importe (diagrama 6)

Si la Velocidad Media es mayor que la velocidad de arrastre se aplica el importe por kilómetro en caso contrario se aplica el importe por hora.

Devuelve el importe en un BigDecimal.

1.2.4 *Cálculo del importe por servicio concertado.*

Si la ruta tiene su origen en la Zona A o en la Zona B se aplican los importes establecidos esas zonas. El proceso lo ejecuta el método CalcularImportesServCon que devuelve un objeto de la clase ServicioConcertadoModel.

1.2.5 *Cálculo del importe de los suplementos (diagrama 8).*

Se leen los suplementos aplicables, se comprueba su validez y se van acumulando hasta el fin de los suplementos.

Se devuelve el importe.

1.3 Condiciones del entorno GIS

En el Ayuntamiento de Madrid se dispone como plataforma base para el SIG corporativo, la plataforma ArcGIS Enterprise de ESRI, a través del cual se ofrecen servicios de navegación, cálculos de rutas, etc., basada en algunos componentes que ofrece TomTom.

Los productos/herramientas necesarios para generar los servicios de cálculo de rutas son:

- Malla viaria Navegable (TomTom MultiNet)
- Datos de tráfico histórico (TomTom Speed Profiles)
- Datos de tráfico en tiempo real (TomTom Traffic Bulk Feed)
- ArcGIS Desktop 10.4.1 o superior
- Herramienta Street Data Processing Tool versión 10.4.1.2.
- ArcGIS Server 10.5.1
- Extensión ArcGIS Network Analyst

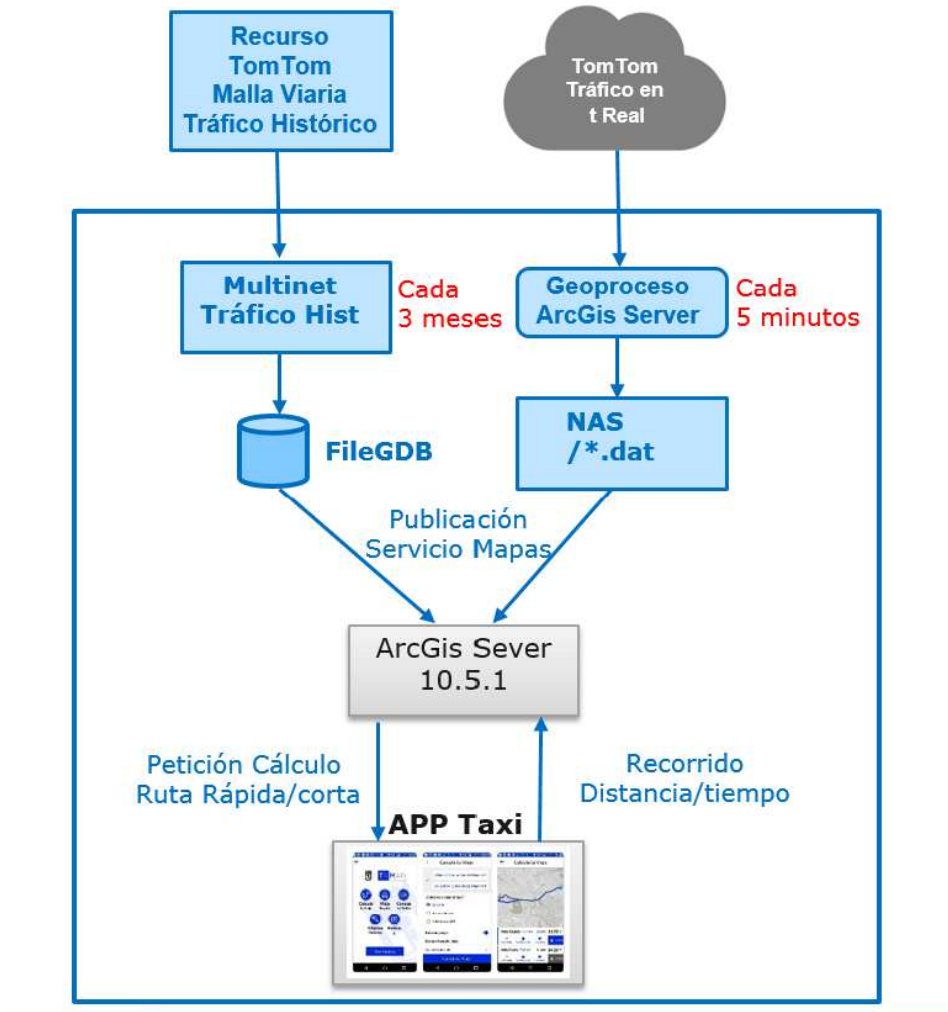
La base para poder realizar el cálculo de rutas es disponer de la malla viaria navegable de TomTom, con datos de tráfico históricos y en vivo. El ámbito territorial abarca toda la Comunidad de Madrid, con un uso principal limitado al municipio de Madrid, uno segundario para los municipios de la APC (área de prestación conjunta) y el resto de uso residual. Los componentes en detalle son:

- Malla viaria Navegable de TomTom (TomTom MultiNet): Base cartográfica vectorial de TomTom compuesta por las capas gráficas y datos alfanuméricos que incluyen direcciones de tráfico, maniobras, jerarquía de red, uso de suelo, etc. y que permiten la creación de servicios de routing (encaminamiento), navegación, seguimiento de vehículos y localización más cercana, entre otros. Se dispone de una malla viaria actualizada cada 3 meses.
- Datos de tráfico histórico asociado a la malla viaria (TomTom Speed Profiles): Datos alfanuméricos asociados a cada tramo de la malla viara (TomTom MultiNet) que permite realizar cálculos de routing en fechas y horas programadas, ya que facilita para cada día tipo, las situación previsible del tráfico en cada franja horaria a partir de datos estadísticos de tráfico histórico. Estos datos se actualizan junto con la malla, cada 3 meses.
- Datos de tráfico en vivo (o tiempo real) asociado a la malla viaria (Denominación: TomTom Traffic Bulk Feed): Compuesto por datos alfanuméricos asociados a los tramos de la malla viaria (TomTom MultiNet) que se facilitan en tiempo real y dan información de velocidades medias y tiempos de recorrido (TomTom Traffic Flow), y la localización de congestiones o incidencias que afectan a la transito vial (TomTom Traffic Incident). Estos datos se recuperan

invocando a un servicio de TomTom cada 5 minutos a través de un geoproceso desarrollado en Phyton.

Partiendo de estos componentes, se genera una base de datos geográfica (GDB), utilizando para ello un equipo con versión mínima 10.4.1 de ArcGIS Desktop y con la herramienta Street Data Processing Tool versión 10.4.1.2 instalada. (Cuando se instale esta herramienta, aparecerá en las Tools de ArcCatalog). Esta GDB incluirá un DATASET con la malla.

Por otro lado, se utiliza un MXD o documento de mapa generado con ArcMap, que tiene definidos todos los parámetros de configuración del cálculo de ruta, y que apuntará a la GDB anterior. Desde este MXD se generan los servicios de Mapas necesarios (.SD) para el cálculo de rutas, que serán publicados en ArcGIS Server 10.5.1 y que son invocados por la aplicación GTAXI.



1.4 Descripción y versiones del software base utilizado.

La aplicación es un proyecto Maven 4.0 realizado en javax 6.0 y servidor Websphere 8.5