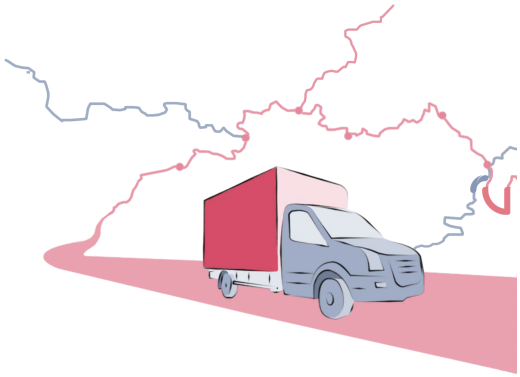
vetu
optimizando en Internet

Sistema SaaS para la resolución del problema del viajante



Manual de usuario

Versión 2.0

infozara.es

SOBRE INFOZARA

Infozara es una empresa que se constituyó en 2006 como spin off de la Universidad de Zaragoza a través del Grupo Nóesis, Grupo Consolidado de Investigación Aplicada del Gobierno de Aragón (España) dirigido por el Profesor Eladio Domínguez.

Infozara tiene una amplia experiencia en la realización de proyectos de I+D+I y en la prestación de servicios a clientes, que cuentan con un alto nivel de valor añadido derivado de las investigaciones industriales realizadas en dichos proyectos.

Todos los productos y desarrollos se han realizado para ser explotados a través de la Web, bajo la forma de lo que actualmente se llama servicios SaaS (Software as a Service).

PROPIEDAD Y CONFIDENCIALIDAD

ÍNDICE

1 Introducción

VETU es un sistema que permite la resolución de problemas complejos de asignación de recursos mediante la modalidad SaaS. Uno de los problemas cuya resolución ofrece VETU, es el conocido problema del viajante, que permite definir las rutas de menor coste para visitar una serie de lugares.

Este documento es el manual de usuario del sistema SaaS del problema del viajante de VETU. Ofrece una guía completa sobre el problema del viajante y su acceso mediante el sistema SaaS.

Es un documento dirigido tanto al gestor de recursos de la empresa, que se encarga de confeccionar las rutas, como a los técnicos informáticos que realizarán la integración entre los sistemas empresariales con el sistema SaaS de VETU.

1.1 En este manual

Este manual está estructurado es los siguientes capítulos:

- Introducción

  En el presente capítulo se describen de forma resumida los principales conceptos del dominio de trabajo del sistema que se presenta.

- Problema del viajante

  Se ofrece una descripción detallada del problema de optimización que resuelve el sistema descrito en el presente manual de usuario.

- Acceso a los servicios

  Se muestra la forma de acceso al sistema SaaS del problema del viajante, mediante los servicios web que publica VETU.

- Lista de mensajes

  Muestra una lista con los mensajes que se pueden producir durante la ejecución de un problema del viajante en el sistema SaaS de VETU.

- Próxima versión

  Se describen las nuevas funcionalidades previstas para la próxima versión del sistema de asignación de flotas que se publicará en VETU.

1.2 SaaS

Los sistemas ofrecidos en forma de software como servicio (SaaS - Software as a Service), posibilitan el acceso a través de Internet a servicios de tecnologías de la información (IT – Information Technologies) de diversa naturaleza, evitando costosas adquisiciones de licencias de software o el pago de elevadas cuotas anuales.

Las ventajas de los sistemas ofrecidos en forma de SaaS son las siguientes:

- Evitan la necesidad de adquirir infraestructura; la infraestructura se comparte entre los usuarios del servicio.

- Se agiliza la entrega de actualizaciones del software evitando costosas actualizaciones; el servicio se actualiza de forma transparente a sus usuarios.
- Se paga por uso.
- Facilitan la integración con sistemas empresariales (mediante estándares -WS-).

VETU posibilita la resolución de problemas de asignación de recursos ofreciendo como SaaS el acceso a un algoritmo que obtiene la solución óptima con un coste mínimo.

El acceso al servicio SaaS de VETU, se proporciona mediante servicios web tal y como se describe más adelante en este manual de usuario.

1.3 VETU

VETU es una plataforma que resuelve diversos problemas de optimización de recursos mediante la modalidad SaaS. Uno de los problemas que publica es el del viajante, pero publica más, como la planificación táctica de personal o la asignación de flotas. Para ver todos los problemas que publica VETU, consulte la web www.vetu.es.

VETU proporciona una forma homogénea para el uso de todos sus problemas de optimización, tanto desde un punto de vista técnico (acceso mediante servicios Web) como desde un punto de vista contractual con los clientes. Toda la información sobre contratación de servicios e resolución de problemas de optimización de recursos se puede consultar en la web de VETU: www.vetu.es.

La plataforma VETU es proporcionada por Infozara Consultoría Informática S. L., entidad inscrita en el Registro Mercantil de Zaragoza, tomo 3368, libro O, folio 52, hoja Z-40867, cuyo domicilio social está ubicado en C/ Marina Española n.º 12, pral. C, Edificio Azabache, Zaragoza, España; NIF: B99104721.

1.4 ¿Quiénes somos?

Infozara es una empresa que se constituyó en 2006 como spin off de la Universidad de Zaragoza a través del Grupo Nóesis, Grupo Consolidado de Investigación Aplicada del Gobierno de Aragón (España) dirigido por el Profesor Eladio Domínguez.

Infozara tiene una amplia experiencia en la realización de proyectos de I+D+I y servicios con un alto nivel de valor añadido, derivado de las investigaciones industriales realizadas en dichos proyectos.

Una característica común a todos los proyectos ha sido la construcción, en cada uno de ellos, de productos en estado precompetitivo y el desarrollo de una metodología de construcción industrial del software como parte integral de un servicio.

Desde su fundación, Infozara:

- Ha participado en diversos proyectos de Desarrollo e Investigación Industrial destacando los proyectos SPOCS (www.spocs.es), LISBB (www.lisbb.es), QRP (qrp.infozara.es), AMBÚ (ambu.infozara.es) y SMOTY (www.smoty.es) del Plan nacional de I+D+i.
- Ha desarrollado productos y componentes industriales en estado precompetitivo en el marco de los proyectos anteriores o como desarrollo posterior ante demanda del mercado.
- Ha construido y está construyendo servicios en la Cloud y en el ámbito del Internet de las Cosas (IoT).

- Tiene una cartera de clientes a los que se les está ofreciendo servicios de valor añadido.

Todos los productos y desarrollos se han realizado para ser explotados a través de la Web, bajo la forma de lo que actualmente se llama servicios SaaS (Software as a Service).

1.5 Problema del viajante

En este manual de usuario definimos el problema del viajante (*Travelling Saleman Problem*, TSP) como el problema de planificar la ruta que realizará un viajante, al menor coste.

Se habla del viajante como concepto general porque es como se ha definido el problema en la literatura científica, pero este problema se puede aplicar a otro tipo de recursos humanos o materiales. También se hablará en general del cocepto localidad como lugares que vistael viajante. Otros casos de aplicación de este problema pueden ser los siguientes:

- Ruta de distribución de mercancías

  El viajante es el vehículo, y las localidades a visitar son los clientes que reciben las mercancías.

- Ruta de soldadura de circuitos impresos

  El viajante es el soldador y las localidades son el punto de soldadura.

- Rutas de carga en almacenes

  El viajante es la y las localidades son los lugares donde está la mercancía.

- Ruta de un cartero:

  El viajante es el cartero y las localidades son los portales donde se entregan las cartas.

Las entradas del problema del viajente son:

- Localidades

  Relación de todas las localidades que debe visitar el viajante.

- Distancias

  Distancia entre cada par de localidades.

El objetivo es encontrar la ruta del viajante, que saliendo de una liocalidad de origen, visita todas las localidades y regresa a la localidad de origen, recorriendo la menor distancia posible.

Este problema se define en detalle en el capítulo 2, para ayudar a comprender el enfoque del TSP que publica VETU.

Si el problema de rutas de su empresa difiere del aquí presentado, o requiere alguna característica no recogida en el TSP de VETU, puede ponerse en contacto con Infozara por cualquiera de los medios indicados en el apartado 1.9, para plantear una solución satisfactoria para usted.

1.6 Nomenclatura

A lo largo del presente documento se hará referencia a los siguientes términos:

SaaS	Software as a Service, Software como Servicio, definido en el apartado 1.2.
TSP	Problema del viajante (*salesman travelling problem*), introducido en apartado 1.5

y descrito en mayor detalle en el capítulo 2.

Los servicios web de VETU y sus funciones se muestran del color azul, como solve(). Los objetos de definición de los problemas a resolver y las soluciones de los problemas de optimización se muestran en color rojo, como TspProblem.

Las direcciones web o direcciones de correo electrónico que se referencian en este documento, se muestran en color azul, como www.vetu.es.

1.7 Soporte

Si es usted cliente de la plataforma VETU de Infozara, puede obtener soporte técnico sobre el TSP según los mecanismos que le haya especificado Infozara.

Si no es usted cliente de la plataforma VETU de Infozara, y tiene cualquier duda o sugerencia, puede ponerse en contacto con el equipo de Infozara por los mecanismos descritos en el apartado 1.9.

1.8 Actualización de versiones

Todos los problemas que ofrece la plataforma VETU, y en particular el TSP, siguen un ciclo de vida que va incorporando nuevas funciones que permiten mejorar el uso que obtienen los clientes de los problemas de optimización. El capítulo 5 muestra una descripción de las nuevas funciones que incorporará la siguiente versión del TSP de VETU.

Si es usted cliente de la plataforma VETU de Infozara, podrá consultar la política de cambio de versiones en su contrato.

1.9 Más información

Si desea más información sobre cualquier aspecto del sistema SaaS del problema del viajante de VETU, puede ponerse en contacto con Infozara, a través de los siguientes medios:

Teléfono:	+34 976 25 43 76
Correo electrónico:	info@vetu.es
	tsp@vetu.es

También puede consultar las siguientes direcciones web:

www.infozara.es

www.vetu.es

www.vetu.es/webvetu/saas.do

www.vetu.es/webvetu/problemaViajante.do

2 Problema del viajante

2.1 Modelo conceptual del problema

La Figura 2-1 muestra el modelo conceptual del TSP. El diagrama se lee de la siguiente forma:

> 'El problema del viajante (TSP) viene definido por una serie de localidades y la distancia entre todas ellas. El problema consiste en encontrar la ruta de menor coste. Cada ruta es una lista ordenada de localidades.

En el siguiente apartado se explican con detalle cada uno de los conceptos que definen el TSP.

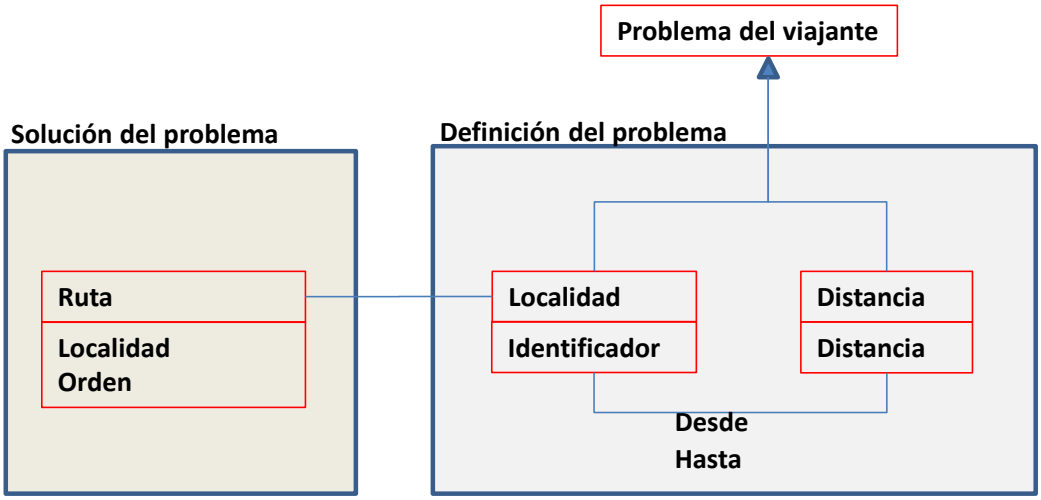


Figura 2-1. Modelo conceptual del TSP

2.2 El problema al detalle

El objetivo que busca el TSPP es crear las rutas de menor coste para un viajante que sale de una localidad de origen, visita todas las localidades, y vuelve a la localidad de origen. La ruta completa que realiza el viajante debe ser la que tiene una mejor distancia total recorrida. La opción de optimización de este problema es el orden en que realizan las visitas a las localidades.

En la definición del problema hay dos entidades: las localidades a visitar y la distancia entre cada par de localidades. La Tabla 2-1 muestra la definición de una Localidad, que tiene el campo único Identificador, y la Tabla 2-2 muestra un ejemplo de definición de localidades para el TSP.

Parámetro	Descripción	Tipo
Identificador	Nombre o identificador único de la localidad.	Texto

Tabla 2-1. Definición de los parámetros de una localidad

Parámetro	Localidades
Identificador	Salamanca
Identificador	Madrid
Identificador	Cáceres
Identificador	Ávila
Identificador	Segovia

Tabla 2-2. Ejemplos de localidades

La entidad Distancia recoge la distancia entre cada par de localidades del TSP, y está definida por los campos que muestra la Tabla 2-3. La matriz de distancias puede ser simétrica, en cuyo caso las distancias entre cada par de localidades es independiente del destino. Si la matriz de distancias no es simétrica, hay dos distancias para cada par de localidad, una para cada sentido. La Tabla 2-4 muestra un ejemplo de matriz simétrica de las distancias entre las localidades definidas en la Tabla 2-2.

Parámetro	Descripción	Tipo
Desde	Identificador de la primera localidad.	Texto
Hasta	Identificador de la segunda localidad.	Texto
Distancia	Distancia entre la localidad Desde y Hasta, cuando se viaja desde la localidad Desde a la Localidad Hasta.	Decimal

Tabla 2-3. Definición de los parámetros de la distancia

Desde	Hasta	Distancia
Salamanca	Madrid	213
Salamanca	Cáceres	201
Salamanca	Ávila	111
Salamanca	Segovia	170
Madrid	Cáceres	296
Madrid	Ávila	112
Madrid	Segovia	94
Cáceres	Ávila	235
Cáceres	Segovia	300
Ávila	Segovia	66

Tabla 2-4. Ejemplos de distancias

2.3 Naturaleza del problema

El TSP es un problema de optimización combinatoria, lo que significa que existen muchas soluciones diferentes, es decir, existen muchas formas de construir las rutas del viajante. Cada una de estas formas de construir las rutas tiene un coste diferente (distancia total recorrida), con lo que el problema consiste en encontrar la ruta de coste mínimo.

Existen muchas rutas diferentes para visitar n localidades. Desde un punto de vista matemático, existe un número finito de rutas diferentes, pero es un número tan elevado que no permite generar todas las rutas para evaluarlas y quedarse con la mejor. Infozara ha desarrollado un algoritmo basado en técnicas de Investigación Operativa, que permite encontrar la solución óptima en varios minutos de cálculo intensivo. Este algoritmo es el que se encapsula detrás de los servicios Web, y está disponible para los clientes de Vetu, sin que

estos tengan la necesidad de conocer técnicas matemáticas, Investigación Operativa o programación informática: el cliente tiene un problema de asignación de flotas y quiere que se resuelva de la mejor forma posible, sin preocuparse de la infraestructura que tiene implantada Infozara para resolver problemas de este tipo.

Un componente de optimización es un cierto concepto que puede tomar un rango de valores. El TSP que se presenta en este manual, solo contempla un criterio de optimización: el orden de las localidades.

Para comprender la naturaleza del TSP, en la Tabla 2-5 se muestran todas las posibles rutas que se derivan del problema planteado en la Tabla 2-2 y en la Tabla 2-4, donde la localidad de origen es Salamanca. Cada una de las rutas tiene un orden de localidades diferente, y presenta una distancia total recorrida diferente. Las mejores rutas son las de 510 Km de distancia total (las rutas son iguales, recorridas en sentido contrario). La Tabla 2-5 muestra la naturaleza del problema: hay muchas rutas diferentes, en cada ruta se recorre una distancia total distinta, y el objetivo es encontrar la mejor ruta, sin necesidad de crearlas todas, porque eso es imposible en problemas con más localidades que las del ejemplo presentado. Existen técnicas matemáticas específicas para resolver este problema, que son las que Infozara publica en VETU. De esta forma un cliente de VETU, envía un problema, obtiene la solución, sin necesidad de saber el tipo de problema que es, ni de las técnicas matemáticas o informáticas necesarias para su resolución.

Ruta	Distancia
Salamanca, Madrid, Cáceres, Ávila, Segovia, Salamanca	704
Salamanca, Madrid, Cáceres, Segovia, Avila, Salamanca	710
Salamanca, Madrid, Ávila, Cáceres, Segovia, Salamanca	1.030
Salamanca, Madrid, Avila, Segovia, Caceres, Salamanca	892
Salamanca, Madrid, Segovia, Caceres, Avila, Salamanca	971
Salamanca, Madrid, Segovia, Avila, Caceres, Salamanca	827
Salamanca, Caceres, Madrid, Avila, Segovia, Salamanca	569
Salamanca, Caceres, Madrid, Segovia, Avila, Salamanca	**510**
Salamanca, Caceres, Avila, Madrid, Segovia, Salamanca	830
Salamanca, Caceres, Avila, Segovia, Madrid, Salamanca	827
Salamanca, Caceres, Segovia, Madrid, Avila, Salamanca	836
Salamanca, Caceres, Segovia, Avila, Madrid, Salamanca	892
Salamanca, Avila, Madrid, Caceres, Segovia, Salamanca	713
Salamanca, Avila, Madrid, Segovia, Caceres, Salamanca	836
Salamanca, Avila, Caceres, Madrid, Segovia, Salamanca	648
Salamanca, Avila, Caceres, Segovia, Madrid, Salamanca	971
Salamanca, Avila, Segovia, Madrid, Caceres, Salamanca	**510**
Salamanca, Avila, Segovia, Caceres, Madrid, Salamanca	710
Salamanca, Segovia, Madrid, Caceres, Avila, Salamanca	648
Salamanca, Segovia, Madrid, Avila, Caceres, Salamanca	830
Salamanca, Segovia, Caceres, Madrid, Avila, Salamanca	713
Salamanca, Segovia, Caceres, Avila, Madrid, Salamanca	1.030
Salamanca, Segovia, Avila, Madrid, Caceres, Salamanca	569
Salamanca, Segovia, Avila, Caceres, Madrid, Salamanca	704

Tabla 2-5. Todas las posibles rutas

3 Acceso a los servicios de VETU

En este capítulo se indica cómo acceder a los servicios Web que publica el sistema SaaS de VETU, para resolver el problema del viajante.

3.1 Arquitectura general de todo el sistema

La arquitectura del sistema SaaS de VETU, que publica Infozara para la resolución de problemas de optimización, es la que muestra la Figura 3-1. En un determinado momento de ejecución de los procesos empresariales del cliente, es necesario dimensionar una plantilla de personal para dar un servicio. En ese momento, se lanza un proceso de llamada que recoge la información de los sistemas empresariales del cliente, y llama a los servicios web del sistema SaaS de VETU. El problema se resuelve en la infraestructura de cálculo de Infozara y la planificación obtenida la recoge el proceso de llamada, que la almacena en los sistemas empresariales del cliente, poniéndolos a disposición del usuario.

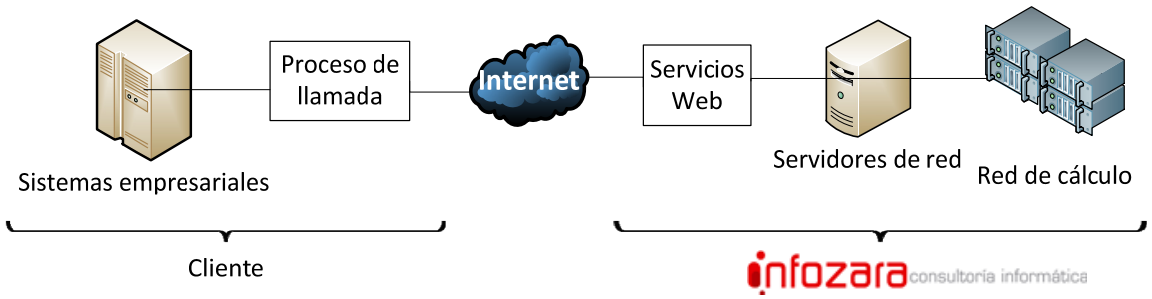


Figura 3-1. Arquitectura del sistema SaaS

Cabe destacar que la infraestructura de red y de cálculo de Infozara no es conocida por el cliente, que lo que quiere es que se resuelva su problema de planificación táctica de personal, sin necesidad de conocer los procesos y máquinas de Infozara.

3.2 Servicios Web

El acceso a la resolución de problemas de optimización mediante SaaS, se hace mediante dos servicios Web: un servicio Web para enviar los datos del problema y obtener la solución, y otro servicio Web para comprobar el estado de ejecución de la resolución del problema.

El servicio Web de comprobación del estado de ejecución es necesario, porque el TSP no se resuelve de forma instantánea, sino que puede requerir de varios minutos, dado que es un problema complejo de optimización combinatoria (véase apartado 2.3). En función del tipo de problema y de su configuración, el tiempo de resolución puede ir desde varios segundos, hasta varios minutos.

El flujo general de uso del sistema SaaS de VETU es el siguiente:

1. Se envía el problema
2. Se comprueba reiteradamente el estado de resolución del problema hasta que esté terminado
3. Se accede a la solución

El servicio Web de resolución de problemas devuelve una clase Request, con un ticket cada vez que se envía un problema. Este ticket se utiliza para acceder al servicio Web de comprobación del estado de la resolución del problema. Si se producen errores, el ticket tendrá el valor -1, y se mostrarán los errores detectados.

Dado que los servicios Web están basados en estándares, es posible acceder a ellos mediante cualquier plataforma de desarrollo. En este manual se presenta el acceso a los servicios Web desde la Plataforma .NET, NetBeans, y Eclipse.

El servicio web de comprobación de estado se llama wsProgress y expone la función checkStatus(). El servicio web de resolución del TSP se llama wsTSP y expone las funciones solve() y getSolution().

En los siguientes apartados se describe cada uno de los servicios web.

3.2.1 Servicio web de comprobación de estado de problemas

El servicio web que permite obtener el estado de la resolución de una instancia de problema enviada a VETU se llama wsProgress, y tiene la función:

String getStatus(String username, String password, String nTicket)

Este servicio web es independiente del tipo de problema a resolver, y expone una función para obtener el estado de la ejecución de un problema a partir de su ticket de identificación. Esta función devuelve uno de los estados que muestra la Tabla 3-1.

Estado	Significado
UNKNOWN	El ticket no corresponde a una solicitud de resolución de ningún problema registrado en el sistema.
ALLOCATED	El problema se ha recibido correctamente, y está listo para su ejecución.
IN_PROCESS	El problema se está resolviendo en este instante.
PROCESSED	El problema ya está resuelto, y se puede acceder a la solución
ERROR	Se ha producido un error en la llamada a la función. Contactar con el administrador.

Tabla 3-1. Estados de resolución de un problema

Para obtener la definición del servicio web, puede escribir la siguiente dirección en cualquier navegador:

http://www.vetu.es/wsProgress/wsProgress?wsdl

3.2.2 Servicio web de resolución del problema

El servicio web que permite la resolución del TSP se llama wsTSP, y tiene dos funciones

Request solve(String username, String password, TspProblem oProblem)

TspSolution getSolution(String nTicket)

Para obtener la definición del servicio web, puede escribir la siguiente dirección en cualquier navegador:

http://www.vetu.es/wsTSP/wsTSP?wsdl

La función solve() recibe la clase TspProblem, que encapsula la instancia del TSP a resolver, y devuelve un objeto de la clase Request, que indica si la instancia enviada es correcta. La Figura 3-2 muestra la versión XML de una clase Request que corresponde a un problema válido: el campo status vale Ok, hay un ticket válido, y el mensaje es 'The problem has been successfully solved'. La Figura 3-3 muestra la versión XML de una clase Request que corresponde a un problema con errores: el campo status tiene el valor Failed, el ticket vale -1, y hay un mensaje de error. Se puede ver la lista de mensajes que puede contener un resultado Request en el apartado 4.1.

```
<?xml version="1.0" encoding="utf-8"?>
<Request>
   <Status>Ok</Status>
   <Ticket>457223874</Ticket>
   <Message Code="SRV0">The problem has been received</Message>
</Request>
```

Figura 3-2. Resultado de un problema sin errores

```
<?xml version="1.0" encoding="utf-8"?>
<Request>
   <Status>Failed</Status>
   <Ticket>-1</Ticket>
   <Message Code="SRV4">The contract has expired</Message>
</Request>
```

Figura 3-3. Resultado de un problema con errores

El objeto TspProblem encapsula la información del problema que se quiere resolver. La Figura 3-4 muestra la versión XML de la definición de un TSP. El elemento raíz TSP_Problem se compone de tres objetos: Configuration, que define si la matriz de distancias es simétrica, Locations, que define las localidades, y Distances, que defina las distancias entre las localidades.

```
<TSP_Problem>
  <Configuration>
    <Symmetric>yes</Symmetric>
  </Configuration>
  <Locations>
    <Location>Salamanca</Location>
    <Location>Madrid</Location>
    <Location>Caceres</Location>
    <Location>Avila</Location>
    <Location>Segovia</Location>
  </Locations>
  <Distances>
    <Distance>
      <From>Salamanca</From>
      <To>Madrid</To>
```

```
      <Length>213</Length>
    </Distance>
    <Distance>
      <From>Salamanca</From>
      <To>Caceres</To>
      <Length>201</Length>
    </Distance>
    <Distance>
      <From>Salamanca</From>
      <To>Avila</To>
      <Length>111</Length>
    </Distance>
    <Distance>
      <From>Salamanca</From>
      <To>Segovia</To>
      <Length>170</Length>
    </Distance>
    <Distance>
      <From>Madrid</From>
      <To>Caceres</To>
      <Length>20.0</Length>
    </Distance>
    <Distance>
      <From>Madrid</From>
      <To>Avila</To>
      <Length>112</Length>
    </Distance>
    <Distance>
    <From>Madrid</From>
      <To>Segovia</To>
      <Length>112</Length>
    </Distance>
    <Distance>
      <From>Caceres</From>
      <To>Avila</To>
      <Length>235</Length>
    </Distance>
    <Distance>
      <From>Caceres</From>
      <To>Segovia</To>
      <Length>300</Length>
    </Distance>
    <Distance>
      <From>Avila</From>
      <To>Segovia</To>
      <Length>66</Length>
    </Distance>
  </Distances>
</TSP_Problem>
```

Figura 3-4. Ejemplo de instancia XML de un TSP

El objeto Configuration define el único parámetro de configuración del problema TSP, que se define el en objeto Symmetric. Si el objeto Symmmetric toma el valor 'yes', significa que la matriz de distancias entre las localidades es simétrica, lo que significa que la distancia para viajar desde la localidad x a la localidad y es la misma que la distancia para viajar desde la localidad y a la localidad x. Si el objeto Symmmetric toma el valor 'no', significa que la matriz de distancias entre las localidades no es simétrica. Si la matriz de distancias es simétrica, es suficiente con especificar una vez la distancia entre cada par de localidades. Si la matriz de distancias no es simétrica, se deben especificar dos distancias para cada pareja de ciudades, una por destino.

El objeto Locations es una lista de objetos Location, que especifican las localidades que debe visitar el viajante. Cada objeto Location contiene el nombre de la localidad. La primera localidad que se especifica dentro del objeto Location, es la localidad de salida del viajante. Como ejemplo de este concepto, en la Figura 3-4, la localidad de salida es Salamanca, es decir, el problema consiste en ordenar las localidades Madrid, Cáceres, Ávila y Segovia para que viajante las visite todas, saliendo desde Salamanca y llegando al final a Salamanca.

El objeto Distances en una lista de objetos Distance, que contienen la información que especifica las distancias entre las localidades. El objeto Distance se define por los siguientes campos:

La posible asignación de un tipo de vehículo a un trayecto se hace mediante el campo Duration, que muestra los siguientes campos:

- From: Localidad de origen.
- To: Localidad destino.
- Length: Distancia desde la localidad de origen a la localidad destino.

El campo Length no se especifica en ninguna medida concreta. Su magnitud podría ser una unidad de distancia (como el kilómetro), con lo que objetivo del problema es minimizar la distancia total recorrida, o una magnitud de tiempo (como el minuto), con lo que el objetivo del problema es minimizar el tiempo total empleado en hacer la ruta completa.

A continuación se va a mostrar el formato de la salida del servicio web que resuelve el TSP en VETU. La Tabla 3-6 muestra un ejemplo en XML, de la solución del TSP que genera el proceso de resolución. El elemento raíz que encapsula toda la información de la solución es TSP_Solution, que se compone de tres objetos: Result, que indica que el problema se ha resuelto correctamente, ObjetiveValue, que muestra el valor objetivo encontrado en la solución, y Route, que es la ruta que realiza el viajante.

```
<TSP_Solution>
  <Result>
    <Status>Ok</Status>
    <Message Code ="TSP0">The TSP problem has been successfully solved</Message>
  </Result>
  <ObjectiveValue>
    <TotalDistance>510</TotalDistance>
  </ObjectiveValue>
  <Route>
    <LocationOrder>
      <Identifier>Caceres</Identifier>
      <Order>0</Order>
    </LocationOrder>
    <LocationOrder>
      <Identifier>Madrid</Identifier>
      <Order>1</Order>
    </LocationOrder>
    <LocationOrder>
      <Identifier>Segovia</Identifier>
      <Order>2</Order>
    </LocationOrder>
    <LocationOrder>
      <Identifier>Avila</Identifier>
      <Order>3</Order>
    </LocationOrder>
  </Route>
</TSP_Solution>
```

Tabla 3-2. Ejemplo de solución del TSP

El objeto Result indica que el problema se ha resuelto correctamente, cuando el campo Status tiene el valor Ok y el mensaje es el de código TSP0. Si se ha producido algún error, el campo Status mostrará el valor Failed, y se mostrarán una serie de mensajes con los errores encontrados. La Tabla 3-7 muestra un ejemplo de solución con errores, donde el campo Status tiene el valor Failed, hay dos mensajes de error, y los objetos ObjetiveValue y Route están vacíos.

```
<?xml version="1.0" encoding="utf-8"?>
<TSP_Solution>
  <Result>
    <Status>Failed</Status>
    <Message Code ="TSP10">Location Burgos is not defined</Message>
    <Message Code ="TSP11">Distances can not be negative values.</Message>
  </Result>
  <ObjectiveValue/>
  <Route/>
</TSP_Solution>
```

Tabla 3-3. Ejemplo de solución del TSP con errores

El objeto ObjectiveValue muestra los valores de las magnitudes de la solución que ha encontrado el proceso de cálculo. Este objeto se compone únicamente del objeto Distance, que es la distancia total que realiza el viajante para visitar todas las localidades, saliendo desde el origen y llegando a ese mismo origen. En el ejemplo de la Tabla 3-6, el viajante sale de Salamanca, visita todas las localidades y vuelve a Salamanca, haciendo un recorrido total de 510 Km.

El objeto Route muestra la ruta de mínimo coste que realiza el viajante. El objeto Route es una lista ordenada de objetos LocationOrder. Cada objeto LocationOrder se compone de dos campos que indican en qué orden se visita cada localidad. El campo Identifier es el nombre de la localidad, y el campo Order indica el orden de esta localidad dentro de la ruta. En el ejemplo de la Tabla 3-6, el viajante sale de Salamanca, visita Cáceres, después visita Madrid, luego va a Segovia, después a Ávila y regresa a Salamanca, en un recorrido total de 510 Km.

3.2.3 Características de objetos y campos

La Tabla 3-4 muestra las características de cada uno de los objetos y campos del problema TSP (objeto TSP_Problem). Todos los objetos son requeridos y ninguno puede estar vacío.

Objeto o campo	Requerido	Vacío	Cardinalidad
TSP_Problem	SI	NO	1
Configuration	SI	NO	1
Symmetric	SI	NO	1
Locations	SI	NO	1
Location	SI	NO	2 o más
Distances	SI	NO	1
Distance	SI	NO	2 o más
From	SI	NO	1
To	SI	NO	1
Length	SI	NO	1

Tabla 3-4. Características de objetos y campos del problema del TSP

La Tabla 3-5 muestra las características de cada uno de los objetos y campos de la solución del problema TSP (objeto TSP_Solution). Todos los objetos son requeridos, y ninguno puede tener un valor vacío.

Objeto o campo	Requerido	Vacío	Cardinalidad
TSP_Solution	SI	NO	1
Result	SI	NO	1
Status	SI	NO	1
Message	SI	NO	1 o más
ObjectiveValue	SI	NO	1
TotalDistance	SI	NO	1
Route	SI	NO	1
LocationOrder	SI	NO	1 o más
Identifier	SI	NO	1
Order	SI	NO	1

Tabla 3-5. Características de objetos y campos de la solución del CAP

3.3 Proceso de ejecución de un TSP mediante los servicios web

A la vista de los diversos estados que devuelve el servicio web de estado de resolución, y de los objetos que maneja el servicio web de resolución del TSP, el pseudocódigo de un programa que resuelve el TSP mediante llamada a los servicios web de VETU, es el que se muestra a continuación. En los ejemplos de uso bajo diversas plataformas de desarrollo que vienen a continuación, se mostrará una implementación detallada de este pseudocódigo.

```
TSP_Problem oEntrada
Definir las localidades a visitar
Definir las distancias entre las localidades
Request oResultado = wsTSP.solve(oEntrada)
If oResultado.Status = Failed
        Tratar errores (por ejemplo, mostrar errores al usuario)
Else If oResultado = Ok
        Bool bTerminado = false
        String strEstado
        While bSalir = false
                strEstado = wsProgress.checkStatus( oResultado.Ticket )
                If( strEstado = UNKNOWN OR strEstado = PROCESSED OR strEstado = ERROR)
                        bTerminado = true
                Else If(strEstado = ALLOCATED OR strEstado = IN_PROCESS)
                        Fijar timer de 1 minuto
                Else
                        bTerminado = true;
                End If
        End While
        Switch strEstado
                Caso strEstado = PROCESSED
```

```
                    TSP_Solution oSalida = wsTSP.getSolution( oResultado.Ticket )
                    If oSalida.Result.Status = Ok
                        Almacenar oSalida en sistemas empresariales
                    Else If oSalida.Result.Status = Failed
                        Mostar errores al usuario
                    End If
                Caso strEstado = UNKNOWN
                    Mostar error al usuario (el ticket no es válido)
                Caso strEstado = ERROR
                    TratarErrores (por ejemplo, mostrarlos al usuario)
                Default
                    Tratar error desconocido
        End
End If
```

Figura 3-5. Pseudocódigo de ejecución del TSP mediante los servicios web.

3.4 Ejemplo de uso desde .NET

En este apartado, se muestra un programa de ejemplo para resolver una instancia de TSP, mediante el sistema SaaS de VETU, escrito en C# (Microsoft .NET [1](1)) de Microsoft, bajo Visual Studio Express 2015 para escritorio de Windows[2].

Visual Studio crea todas las clases necesarias para acceder a servicios web, a partir de la definición WSDL del servicio. Para crear las clases de acceso al servicio web de resolución del TSP de VETU, se deben seguir los siguientes pasos[3]:

1. Pulsar con el botón derecho sobre el icono de la solución de Visual Studio
2. Pulsar sobre la opción de menú 'Add Service Reference …'
3. Pulsar el botón 'Advanced…'
4. Pulsar el botón 'Add Web Reference'
5. Escribir http://www.vetu.es/wsTSP/wsTSP?wsdl en la caja de edición URL
6. Pulsar el botón que está a la derecha de la caja de edición URL
7. Hacer click sobre la línea http://www.vetu.es/wsTSP/wsTSP?wsdl
8. Pulsar el botón 'Add Reference'

En este momento se han creado una serie de clases que permiten acceder al servicio web wsTSP, con acceso a las funciones solve y getSolution. En este momento se dispone de las clases necesarias para acceder al servicio web. Las clases principales que se han creado son las siguientes:

- wsTSPService: Es la clase de acceso al servicio web. Tiene las funciones solve y getSolution.
- tspProblem Es la clase que encapsula la definición de la instancia de problema que se quiere resolver. Contiene a las clases Configuration, que establece las opciones de

[1] Microsoft .NET es una marca registrada por Microsoft Corporation.
[2] Microsoft Visual Studio Express 2015 es una marca registrada por Microsoft Corporation.
[3] Esta secuencia de acciones podría cambiar según la versión de Visual Studio.

configuración del problema, locations, que define las localidades que visita el viajante, y una lista de objetos distance, que contiene las distancias entre las localidades.

- tspSolution: Es la clase que encapsula la solución del problema que se ha resuelto. Contiene a las clases result, objectiveValue y una lista de objetos locactionOrder, que tiene la información de la resolución del problema, del valor encontrado para la ruta mínima, y la ruta del viajante, respectivamente. Cada una de estas clases contiene a otras clases, según el formato definido para la solución en el apartado 3.2.2 de este capítulo.

Para crear las clases de acceso al servicio wsProgressService, se deben repetir los pasos anteriores, escribiendo en el punto 4.- la URL del servicio web de acceso al estado de resolución de problemas: http://www.vetu.es/wsProgress/wsProgress?wsdl. Una vez hecho, se habrá creado la clase wsProgressService, que es el acceso al servicio web, y tiene la función checkStatus.

El código fuente necesario para la resolución de un ejemplo de TSP mediante los servicios web de VETU se muestra en la Figura 3-6. Este código fuente no sigue las mejores prácticas de programación, porque su objetivo es la simplicidad, para mostrar el acceso al servicio SaaS de VETU. Los números de línea de la columna de la izquierda se muestran solo para facilitar la siguiente explicación. Desde las líneas 1 a la 109 se definen los datos del problema a resolver, y desde la línea 112 hasta el final, es el acceso a los servicios web.

En la línea 2 se crea la instancia de la clase tspProblem, que encapsula toda la información del problema, que son la configuración, las localidades a visitar, y las distancias entre localidades.

En la línea 5 se crea el objeto Configuration, y en la línea 6 se establece que las diastancias entre localidades son simétricas, es decir la distancia entre un origen y un destino es la mismo en los dos sentidos de circulación.

En la línea 9 se crea el objeto Locations, que es un array de cadenas de texto con el nombre de las localidades. Entre las líneas 11 y 15, se especifican los nombres de las localidades que debe visitar el viajante. La localidad de origen es la primera en el objeto Locations (Salamanca, en este ejemplo).

En la línea 18 se crea el objeto Distances, que es un vector con todas las distancias entrea cada par de localidades. En generla si hay n localidades y las distancias no son simétricas, hay n^2-n distancias, y si la matriz de distancias es simética, hay (n^2-n)/2 distancias. En este ejemplo n=5 y las distancia son simétricas, as´que que se debe especificar (5^2-5)/2 = 10 distancias. Especificar cada distancia requiere de cuatro líneas de código fuente. Tomando como ejemplo el código entre las líneas 20 y 24, en la línea 21 se crea el objeto distance, como miembro de índice 0 en el array Distances, en las líneas 22 y 23 se indican el origen (Salamanca) y el destino (Madrid), y en la línea 24 se especifica la distancia entre estas dos localidades. Este proceso se debe reperit apra las nueve distancias restantes, que se hace entre las líneas 26 y 69.

Las líneas 72 y 73 especifican que el campo Length de cada objeto distance se va a enviar en la llamada al servicio web. Esto es necesario, porque el generado de código de Visual Studio establece eeste comportamiento con los campos de tipo double.

Una vez creada la instancia del problema tspProblem, con toda la información que define al problema, se debe establecer comunicación con el servicio web, para enviarle el problema. El acceso al servicio web wsTSPService se crea en la línea 76, y en la línea 82 se envía el problema, haciendo uso de la función solve de la clase wsTSPService. La función solve devuelve un objeto de la clase request, que indica si el problema se ha recibido correctamente. Para saber si el problema tiene errores, en la línea 89 se comprueba el valor

del campo Status del objeto request. Si el campo Status tiene el valor "Failed", significa ques e ha propducido errores. Estos errores se muestran en el objeto Message del objeto request. Estos mensajes corresponden a los definidos en el capítulo 4 de este manual. Si se han producido errores, no se puede resolver el problema.

Si el campo status tiene el valor "Ok" (línea 96), significa que el problema se ha recibido correctamente, y se a va resolver. La idea ahora es que se debe acceder al servicio de progreso periódicamente para comprobar si ya está resuelto el problema, y cuando ya esté resuelto, se accede a la solución. En la línea 109 se comprueba el estado de avance en la reolcuión del problema, accediendo al servicio web wsProgresssService, mediante la función checkStatus. A esta función se accede con el Ticket del objeto result, que es como un identificador del problema. La función checkStatus devuelve el estado de resolución del problema (apartado 3.2.1). La comprobación se repite hasta que el resultado sea PROCESSED (ya se ha resuelto el problema), o el resultado sea UNKNOWN o ERROR (ambos expresan error).

Si el estado de resolución es PROCESSED, en la línea 134 se accede a la solución, invocando a la función getSolution del objeto wsTSPService. La solución se devuelve en el objeto tspSolution. Una vez obtenida la solución, se debe comprobar el valor del campo status del objeto result. Si el valor del campo status es "Ok", la solución es válida, y se puede tratar. Si el valor del campo status es "Failed", se muestran los errores en el campo message, que es un array de objetos Message.

```
1   // Crea la instancia de problema TSP que se va a resolver
2   wsTSPReference.tspProblem oProblem = new wsTSPReference.tspProblem();
3
4   // Define la configuración como distancias simétricas
5   oProblem.Configuration = new wsTSPReference.Configuration();
6   oProblem.Configuration.Symmetric = "yes";
7
8   // Crea las localidades
9   oProblem.Locations = new string[5];
10
11  oProblem.Locations[0] = "Salamanca";
12  oProblem.Locations[1] = "Madrid";
13  oProblem.Locations[2] = "Caceres";
14  oProblem.Locations[3] = "Avila";
15  oProblem.Locations[4] = "Segovia";
16
17  // Crea la lista de distancias entre localidades
18  oProblem.Distances = new wsTSPReference.distance[10];
19
20  // Crea un objeto distancia,y fija el valor de sus campos
21  oProblem.Distances[0] = new wsTSPReference.distance();
22  oProblem.Distances[0].From = "Salamanca";
23  oProblem.Distances[0].To = "Madrid";
24  oProblem.Distances[0].Length = 213.0;
25
26  oProblem.Distances[1] = new wsTSPReference.distance();
27  oProblem.Distances[1].From = "Salamanca";
28  oProblem.Distances[1].To = "Caceres";
29  oProblem.Distances[1].Length = 201.0;
30
31  oProblem.Distances[2] = new wsTSPReference.distance();
32  oProblem.Distances[2].From = "Salamanca";
33  oProblem.Distances[2].To = "Avila";
34  oProblem.Distances[2].Length = 111.0;
35
```

```
oProblem.Distances[3] = new wsTSPReference.distance();
oProblem.Distances[3].From = "Salamanca";
oProblem.Distances[3].To = "Segovia";
oProblem.Distances[3].Length = 170.0;

oProblem.Distances[4] = new wsTSPReference.distance();
oProblem.Distances[4].From = "Madrid";
oProblem.Distances[4].To = "Caceres";
oProblem.Distances[4].Length = 296.0;

oProblem.Distances[5] = new wsTSPReference.distance();
oProblem.Distances[5].From = "Madrid";
oProblem.Distances[5].To = "Avila";
oProblem.Distances[5].Length = 112.0;

oProblem.Distances[6] = new wsTSPReference.distance();
oProblem.Distances[6].From = "Madrid";
oProblem.Distances[6].To = "Segovia";
oProblem.Distances[6].Length = 94.0;

oProblem.Distances[7] = new wsTSPReference.distance();
oProblem.Distances[7].From = "Caceres";
oProblem.Distances[7].To = "Avila";
oProblem.Distances[7].Length = 235.0;

oProblem.Distances[8] = new wsTSPReference.distance();
oProblem.Distances[8].From = "Caceres";
oProblem.Distances[8].To = "Segovia";
oProblem.Distances[8].Length = 300.0;

oProblem.Distances[9] = new wsTSPReference.distance();
oProblem.Distances[9].From = "Avila";
oProblem.Distances[9].To = "Segovia";
oProblem.Distances[9].Length = 112.0;

// Especifica el valor del campo generado por VS
for (int i = 0; i < 10; i++)
   oProblem.Distances[i].LengthSpecified = true;

// Crea el acceso al servicio web
wsTSPReference.wsTSPService oCliente = new wsTSPReference.wsTSPService();

// Lanza la petición de resolución del problema
wsTSPReference.request oResultado = null;
try
{
   oResultado = oCliente.solve("usuario", "password", oProblem);
}
catch (Exception ex)
{
   // Tratar el error
}

if (oResultado.Status == "Failed")
{
   foreach (wsTSPReference.Message1 oMensaje in oResultado.Message)
   {
      // Tratar los mensajes de error del problema
   }
}
```

```
96  else if (oResultado.Status == "Ok")
97  {
98     // Espera a que se ejecute el problema
99     wsProgressReference.wsProgressService oProgessServiceClient =
100      new wsProgressReference.wsProgressService();
101
102    string strEstado = "";
103    bool bTerminado = false;
104
105    while (!bTerminado)
106    {
107        try
108        {
109            strEstado = oProgessServiceClient.checkStatus(
110                                    "usuario", "password", oResultado.Ticket);
111        }
112        catch (Exception ex)
113        {
114            // Tratar el error
115
116            bTerminado = true;
117            strEstado = "ERROR";
118        }
119
120        if (strEstado == "UNKNOWN" || strEstado == "ERROR" ||
121            strEstado == "PROCESSED")
122            break;
123        else if (strEstado == "ALLOCATED" || strEstado == "IN_PROCESS")
124            System.Threading.Thread.Sleep(1000);
125        else
126           break;
127    }
128
129    // Lee el resultado del problema
130    if (strEstado == "PROCESSED")
131    {
132        // Guarda el resultado
133        wsTSPReference.tspSolution oSolucion =
134              oCliente.getSolution("usuario", "password", oResultado.Ticket);
135
136        if (oSolucion.Result.Status == "Ok")
137        {
138            // Tratar la solución
139            foreach (wsTSPReference.locationOrder lo in oSolucion.Route)
140            {
141                // Tratar la solución
142            }
143        }
144        else
145        {
146            // Muestra los mensajes de error
147            foreach (wsTSPReference.Message oMensaje in
148                      oSolucion.Result.Message)
149            {
150                // Tratar los errores
151            }
152        }
153    }
154    else
155    {
```

```
156         // Tratar error
157     }
158 }
```

Figura 3-6. Código fuente C# para la ejecución del TSP de VETU

3.5 Ejemplo de uso desde NetBeans

En este apartado, se muestra un programa de ejemplo para resolver una instancia de TSP, mediante el sistema SaaS de VETU, escrito en Java por medio del entorno de desarrollo NetBeans.

NetBeans crea todas las clases necesarias para acceder a servicios web, a partir de la definición WSDL del servicio. Para crear las clases de acceso al servicio web de resolución del TSP de VETU, se deben seguir los siguientes pasos:

1. Pulsar con el botón derecho sobre el proyecto, elegir "New" y en el desplegable que aparece pulsar sobre "Web Service Client".

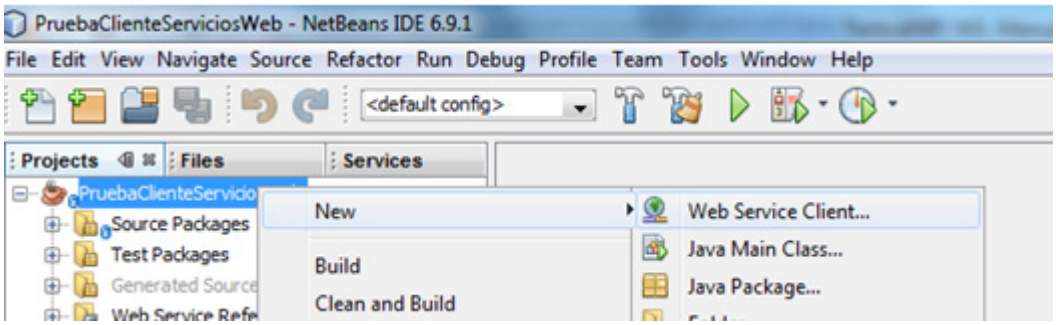


Figura 3-7. Paso 1 - Creación de cliente del servicio web NebBeans

Aparecerá una ventana emergente.

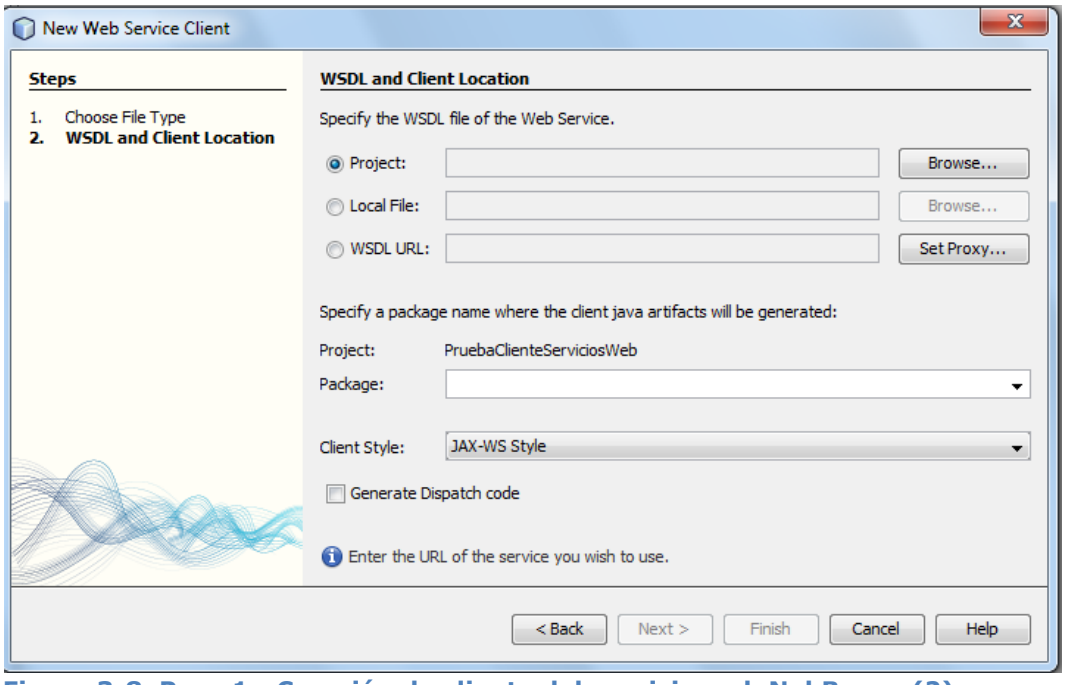


Figura 3-8. Paso 1 - Creación de cliente del servicio web NebBeans (2)

2. Marcar la opción "WSDL URL", escribir http://www.vetu.es/wsTSP/wsTSP?wsdl en la caja de edición y pulsar el botón "Finish".

En este momento se han creado una serie de clases que permiten acceder al servicio web wsTSP, con acceso a las funciones solve y getSolution. En este momento se dispone de las clases necesarias para acceder al servicio web. Las clases principales que se han creado son las siguientes:

- WsTSP: Es la clase de acceso al servicio web. Tiene las funciones solve y getSolution.
- TspProblem Es la clase que encapsula la definición de la instancia de problema que se quiere resolver. Contiene a las clases Configuration, que establece las opciones de configuración del problema, locations, que define las localidades que visita el viajante, y una lista de objetos distance, que contiene las distancias entre las localidades.
- TspSolution: Es la clase que encapsula la solución del problema que se ha resuelto. Contiene a las clases result, objectiveValue y una lista de objetos locactionOrder, que tiene la información de la resolución del problema, del valor encontrado para la ruta mínima, y la ruta del viajante, respectivamente. Cada una de estas clases contiene a otras clases, según el formato definido para la solución en el apartado 3.2.2 de este capítulo.

Para crear las clases de acceso al servicio wsProgress, se deben repetir los pasos anteriores, escribiendo en el paso 2.- la URL del servicio web de acceso al estado de resolución de problemas: http://www.vetu.es/wsProgress/wsProgress?wsdl. Una vez hecho, se habrá creado la clase WsProgress, que es el acceso al servicio web, y tiene la función checkStatus.

El código fuente necesario para la resolución de un ejemplo de TSP mediante los servicios web de VETU se muestra en la Figura 3-9. Este código fuente no sigue las mejores prácticas de programación, porque su objetivo es la simplicidad, para mostrar el acceso al servicio SaaS de VETU. Los números de línea de la columna de la izquierda se muestran solo para facilitar la siguiente explicación.

Desde las líneas 4 a la 82 se definen los datos del problema a resolver, y desde la línea 83 hasta el final, es el acceso a los servicios web.

En la línea 2 se crea la instancia de la clase TspProblem, que encapsula toda la información del problema, que son la configuración, las localidades a visitar, y las distancias entre localidades.

En la línea 5 se crea el objeto Configuration, y en la línea 6 se establece que las diastancias entre localidades son simétricas, es decir la distancia entre un origen y un destino es la mismo en los dos sentidos de circulación.

En la línea 10 se crea el objeto Locations, que es una colección de cadenas de texto con el nombre de las localidades. Entre las líneas 13 y 17, se especifican los nombres de las localidades que debe visitar el viajante. La localidad de origen es la primera en el objeto Locations (Salamanca, en este ejemplo).

En la línea 20 se crea el objeto Distances, que es una colección con todas las distancias entrea cada par de localidades. En general si hay n localidades y las distancias no son simétricas, hay n^2-n distancias, y si la matriz de distancias es simética, hay (n^2-n)/2 distancias. En este ejemplo n=5 y las distancia son simétricas, as´que que se debe especificar (5^2-5)/2 = 10 distancias. Tomando como ejemplo el código entre las líneas 24 y 28, en la línea 24 se crea el objeto Distance, en las líneas 25 y 26 se indican el origen (Salamanca) y el destino (Madrid),

y en la línea 27 se especifica la distancia entre estas dos localidades. Este proceso se debe repetir para las distancias restantes, que se hace entre las líneas 30 y 82.

Una vez creada la instancia del problema TspProblem, con toda la información que define al problema, se debe establecer comunicación con el servicio web, para enviarle el problema. El acceso al servicio web wsTSP se crea en la línea 84, y en la línea 87 se envía el problema, haciendo uso de la función solve de la clase WsTSP. La función solve devuelve un objeto de la clase Request, que indica si el problema se ha recibido correctamente. Para saber si el problema tiene errores, en la línea 89 se comprueba el valor del campo Status del objeto Request. Si el campo Status tiene el valor "Failed", significa ques e ha propducido errores. Estos errores se muestran en el objeto Message del objeto Request. Estos mensajes corresponden a los definidos en el capítulo 4 de este manual. Si se han producido errores, no se puede resolver el problema.

Si el campo status tiene el valor "Ok" (línea 92), significa que el problema se ha recibido correctamente, y se a va resolver. La idea ahora es que se debe acceder al servicio de progreso periódicamente para comprobar si ya está resuelto el problema, y cuando ya esté resuelto, se accede a la solución. En la línea 99 se comprueba el estado de avance en la resolución del problema, accediendo al servicio web wsProgresss, mediante la función checkStatus. A esta función se accede con el Ticket del objeto Result, que actúa como identificador del problema. La función checkStatus devuelve el estado de resolución del problema (apartado 3.2.1). La comprobación se repite hasta que el resultado sea PROCESSED (ya se ha resuelto el problema), o el resultado sea UNKNOWN o ERROR (ambos expresan error).

Si el estado de resolución es PROCESSED, en la línea 115 se accede a la solución, invocando a la función getSolution del objeto WsTSP. La solución se devuelve en el objeto TspSolution. Una vez obtenida la solución, se debe comprobar el valor del campo status del objeto Result. Si el valor del campo status es "Ok", la solución es válida, y se puede tratar. Si el valor del campo status es "Failed", se muestran los errores en el campo message, que es un array de objetos Message.

```
1  // Crea la instancia de problema
2  serviciowebtsp.TspProblem tspProblem = new serviciowebtsp.TspProblem();
3
```

```
4  // Define la configuración como distancias simétricas
5  Configuration configuration = new Configuration();
6  configuration.setSymmetric("yes");
7  tspProblem.setConfiguration(configuration);
8
9  // Crea las localidades
10 Locations locations = new Locations();
11 tspProblem.setLocations(locations);
12
13 locations.getLocation().add("Salamanca");
14 locations.getLocation().add("Madrid");
15 locations.getLocation().add("Cáceres");
16 locations.getLocation().add("Avila");
17 locations.getLocation().add("Segovia");
18
19 // Crea la lista de distancias entre localidades
20 Distances distances = new Distances();
21 tspProblem.setDistances(distances);
22
23 // Crea un objeto distancia y fija el valor de sus campos
```

```
24 Distance distance1 = new Distance();
25 distance1.setFrom("Salamanca");
26 distance1.setTo("Madrid");
27 distance1.setLength(213.0);
28 distances.getDistance().add(distance1);
29
30 Distance distance2 = new Distance();
31 distance2.setFrom("Salamanca");
32 distance2.setTo("Cáceres");
33 distance2.setLength(201.0);
34 distances.getDistance().add(distance2);
35
36 Distance distance3 = new Distance();
37 distance3.setFrom("Salamanca");
38 distance3.setTo("Avila");
39 distance3.setLength(111.0);
40 distances.getDistance().add(distance3);
41
42 Distance distance4 = new Distance();
43 distance4.setFrom("Salamanca");
44 distance4.setTo("Segovia");
45 distance4.setLength(170.0);
46 distances.getDistance().add(distance4);
47
48 Distance distance5 = new Distance();
49 distance5.setFrom("Madrid");
50 distance5.setTo("Cáceres");
51 distance5.setLength(296.0);
52 distances.getDistance().add(distance5);
53
54 Distance distance6 = new Distance();
55 distance6.setFrom("Madrid");
56 distance6.setTo("Avila");
57 distance6.setLength(112.0);
58 distances.getDistance().add(distance6);
59
60 Distance distance7 = new Distance();
61 distance7.setFrom("Madrid");
62 distance7.setTo("Segovia");
63 distance7.setLength(94.0);
64 distances.getDistance().add(distance7);
65
66 Distance distance8 = new Distance();
67 distance8.setFrom("Cáceres");
68 distance8.setTo("Avila");
69 distance8.setLength(235.0);
70 distances.getDistance().add(distance8);
71
72 Distance distance9 = new Distance();
73 distance9.setFrom("Cáceres");
74 distance9.setTo("Segovia");
75 distance9.setLength(300.0);
76 distances.getDistance().add(distance9);
77
78 Distance distance10 = new Distance();
79 distance10.setFrom("Avila");
80 distance10.setTo("Segovia");
81 distance10.setLength(112.0);
82 distances.getDistance().add(distance10);
```

```
// Crea el acceso al servicio web
WsTSPService service = new WsTSPService();
serviciowebtsp.WsTSP port = service.getWsTSPPort();

// Lanza la petición de resolución
Request oResultado = port.solve("usuario", "password", tspProblem);

if (oResultado.getStatus().equals("Failed")) {
    // Tratar los errores
} else if (oResultado.getStatus().equals("Ok")) {
    // Espera a que se ejecute el problema
   WsProgressService serviceExecutionProgress = new WsProgressService();
   WsProgress portProgress = serviceExecutionProgress.getWsProgressPort();

    String strEstado = "";
    boolean bTerminado = false;
    while (!bTerminado) {
        strEstado = portProgress.checkStatus("usuario", "password",
          oResultado.getTicket());

        if (strEstado.equals("UNKNOWN") || strEstado.equals("ERROR") ||
          strEstado.equals("PROCESSED")) {
            bTerminado = true;
        } else if (strEstado.equals("ALLOCATED") ||
          strEstado.equals("IN_PROCESS")) {
            Thread.sleep(5000);
        } else // Estado desconocido
        {
            bTerminado = true;
        }
    }

    // Lee el resultado del problema
    if (strEstado.equals("PROCESSED")) {
        // Lee la solución
        TspSolution tspSolution = port.getSolution("usuario", "password",
          oResultado.getTicket());
        // Tratar la solución.
    } else {
        // Tratar el error
    }
}
```

Figura 3-9. Código fuente Java en NetBeans para la ejecución del TSP de VETU

3.6 Ejemplo de uso desde Eclipse

En este apartado, se muestra un programa de ejemplo para resolver una instancia de TSP, mediante el sistema SaaS de VETU, escrito en Java por medio del entorno de desarrollo Eclipse.

Eclipse crea todas las clases necesarias para acceder a servicios web, a partir de la definición WSDL del servicio. Para crear las clases de acceso al servicio web de resolución del TSP de VETU, se debe tener instalado en Eclipse el módulo de Java EE y se deben seguir los siguientes pasos:

1. Pulsar con el botón derecho sobre el proyecto, elegir "New" y en el desplegable que aparece pulsar sobre "Other".

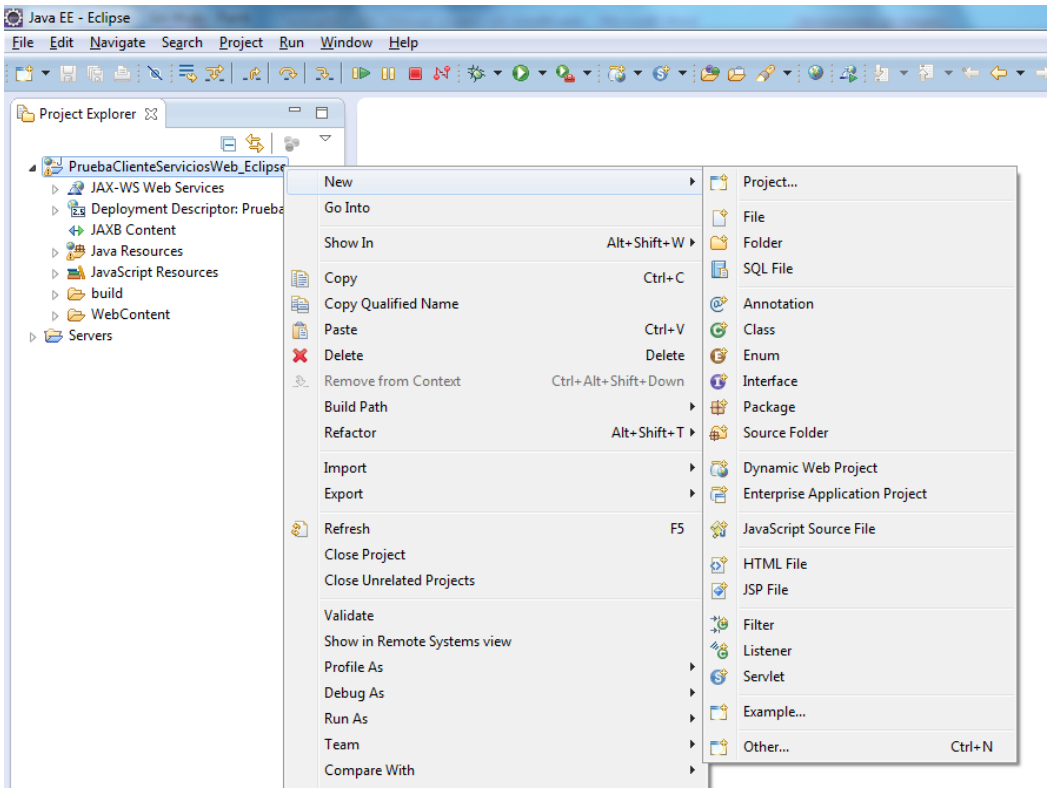


Figura 3-10. Paso 1 - Creación de cliente del servicio web Eclipse

Aparecerá una ventana emergente para seleccionar el tipo de fichero que se desea crear.

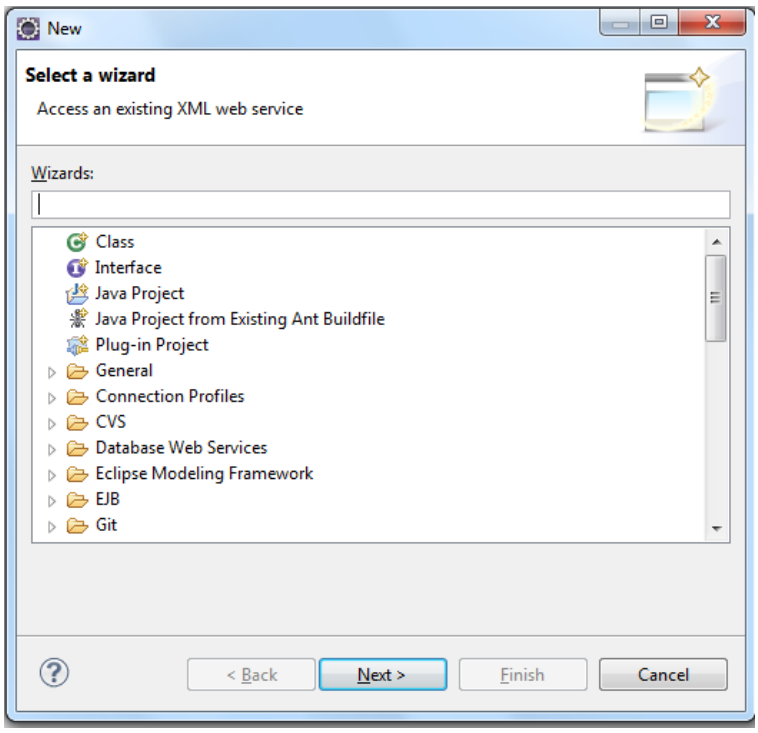


Figura 3-11. Paso 1 - Creación de cliente del servicio web Eclipse (2)

2. Se busca la carpeta de "Web Services" y se elige el tipo de fichero "Web Service Client", y se pulsará sobre el botón "Next".

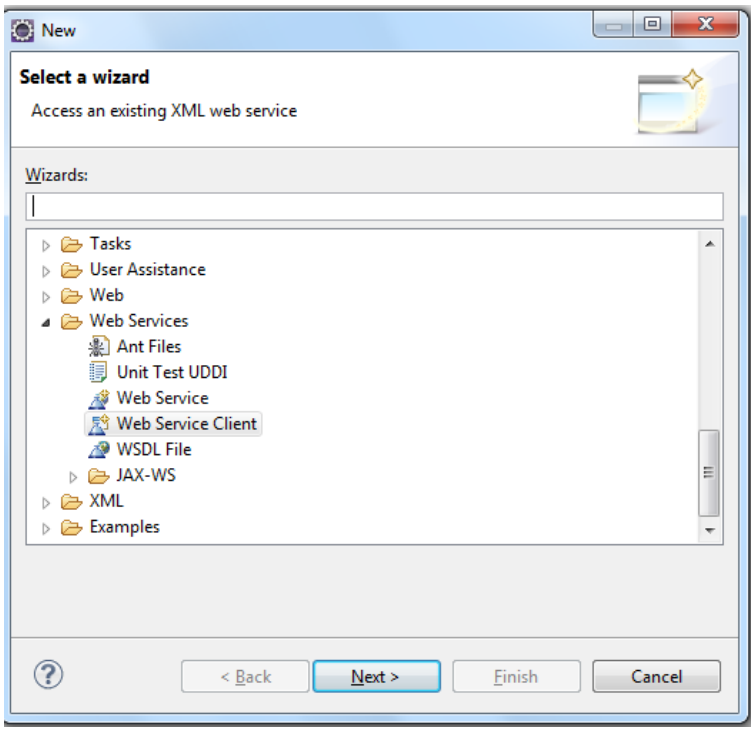


Figura 3-12. Paso 2 - Creación de cliente del servicio web Eclipse

Se mostrará una ventana para configurar el Cliente del Servicio Web.

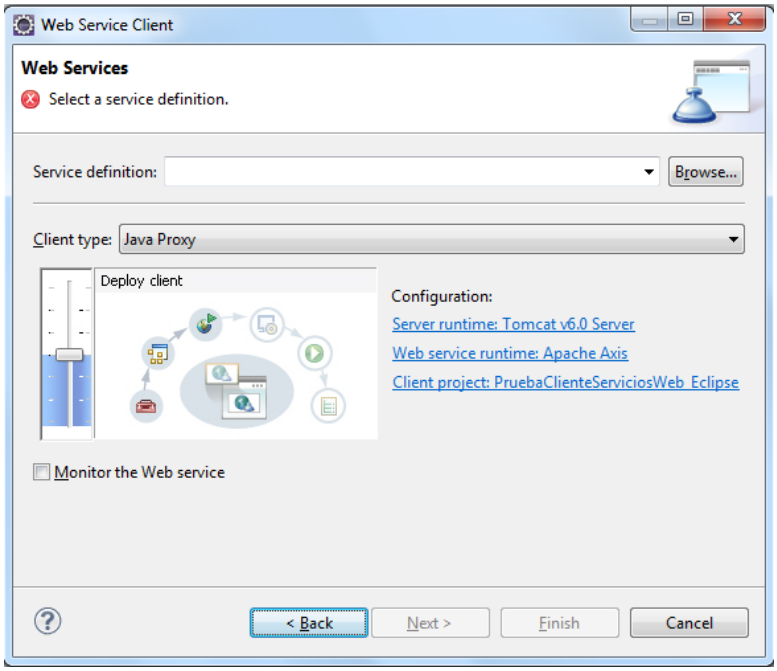


Figura 3-13. Paso 2 - Creación de cliente del servicio web Eclipse (2)

3. En la caja de texto "Service definition" se escribirá lo siguiente: http://www.vetu.es/wsTSP/wsTSP?wsdl. En caso de no tener configurados un servidor o un servicio web, se pulsará sobre el nombre y se elegirá uno de los mostrados en el listado emergente. No hay que tocar el tipo de cliente ni el nivel, tiene que estar en la opción marcada por defecto: "Deploy client".
4. Una vez configurado se pulsará "Finish".

En este momento se han creado una serie de clases que permiten acceder al servicio web wsTSP, con acceso a las funciones solve y getSolution. En este momento se dispone de las clases necesarias para acceder al servicio web. Las clases principales que se han creado son las siguientes:

- WsTSP: Es la clase de acceso al servicio web. Tiene las funciones solve y getSolution.
- TspProblem Es la clase que encapsula la definición de la instancia de problema que se quiere resolver. Contiene a las clases Configuration, que establece las opciones de configuración del problema, locations, que define las localidades que visita el viajante, y una lista de objetos distance, que contiene las distancias entre las localidades.
- TspSolution: Es la clase que encapsula la solución del problema que se ha resuelto. Contiene a las clases result, objectiveValue y una lista de objetos locactionOrder, que tiene la información de la resolución del problema, del valor encontrado para la ruta mínima, y la ruta del viajante, respectivamente. Cada una de estas clases contiene a otras clases, según el formato definido para la solución en el apartado 3.2.2 de este capítulo.

Para crear las clases de acceso al servicio wsProgress, se deben repetir los pasos anteriores, escribiendo en el paso 3.- la URL del servicio web de acceso al estado de resolución de problemas: http://www.vetu.es/wsProgress/wsProgress?wsdl. Una vez hecho, se habrá creado la clase WsProgress, que es el acceso al servicio web, y tiene la función checkStatus.

El código fuente necesario para la resolución de un ejemplo de TSP mediante los servicios web de VETU se muestra en la Figura 3-14. Este código fuente no sigue las mejores prácticas de programación, porque su objetivo es la simplicidad, para mostrar el acceso al servicio SaaS de VETU. Los números de línea de la columna de la izquierda se muestran solo para facilitar la siguiente explicación.

Desde las líneas 4 a la 82 se definen los datos del problema a resolver, y desde la línea 83 hasta el final, es el acceso a los servicios web.

En la línea 2 se crea la instancia de la clase TspProblem, que encapsula toda la información del problema, que son la configuración, las localidades a visitar, y las distancias entre localidades.

En la línea 5 se crea el objeto Configuration, y en la línea 6 se establece que las diastancias entre localidades son simétricas, es decir la distancia entre un origen y un destino es la mismo en los dos sentidos de circulación.

En la línea 10 se crea el objeto Locations, que es un array de cadenas de texto con el nombre de las localidades. Entre las líneas 13 y 17, se especifican los nombres de las localidades que debe visitar el viajante. La localidad de origen es la primera en el objeto Locations (Salamanca, en este ejemplo).

En la línea 20 se crea el objeto Distances, que es un vector con todas las distancias entrea cada par de localidades. En general si hay n localidades y las distancias no son simétricas, hay

n^2-n distancias, y si la matriz de distancias es simética, hay (n^2-n)/2 distancias. En este ejemplo n=5 y las distancia son simétricas, as´que que se debe especificar (5^2-5)/2 = 10 distancias. Tomando como ejemplo el código entre las líneas 24 y 28, en la línea 24 se crea el objeto Distance, en las líneas 25 y 26 se indican el origen (Salamanca) y el destino (Madrid), y en la línea 27 se especifica la distancia entre estas dos localidades. Este proceso se debe repetir para las distancias restantes, que se hace entre las líneas 30 y 82.

Una vez creada la instancia del problema TspProblem, con toda la información que define al problema, se debe establecer comunicación con el servicio web, para enviarle el problema. El acceso al servicio web wsTSP se crea en la línea 84, y en la línea 87 se envía el problema, haciendo uso de la función solve de la clase WsTSP. La función solve devuelve un objeto de la clase Request, que indica si el problema se ha recibido correctamente. Para saber si el problema tiene errores, en la línea 89 se comprueba el valor del campo Status del objeto Request. Si el campo Status tiene el valor "Failed", significa ques e ha propducido errores. Estos errores se muestran en el objeto Message del objeto Request. Estos mensajes corresponden a los definidos en el capítulo 4 de este manual. Si se han producido errores, no se puede resolver el problema.

Si el campo status tiene el valor "Ok" (línea 92), significa que el problema se ha recibido correctamente, y se a va resolver. La idea ahora es que se debe acceder al servicio de progreso periódicamente para comprobar si ya está resuelto el problema, y cuando ya esté resuelto, se accede a la solución. En la línea 99 se comprueba el estado de avance en la resolución del problema, accediendo al servicio web wsProgresss, mediante la función checkStatus. A esta función se accede con el Ticket del objeto Result, que actúa como identificador del problema. La función checkStatus devuelve el estado de resolución del problema (apartado 3.2.1). La comprobación se repite hasta que el resultado sea PROCESSED (ya se ha resuelto el problema), o el resultado sea UNKNOWN o ERROR (ambos expresan error).

Si el estado de resolución es PROCESSED, en la línea 116 se accede a la solución, invocando a la función getSolution del objeto WsTSP. La solución se devuelve en el objeto TspSolution. Una vez obtenida la solución, se debe comprobar el valor del campo status del objeto Result. Si el valor del campo status es "Ok", la solución es válida, y se puede tratar. Si el valor del campo status es "Failed", se muestran los errores en el campo message, que es un array de objetos Message.

```
1  // Crea la instancia de problema
2  servicioWebTSP.TspProblem tspProblem = new  servicioWebTSP.TspProblem();
3
```

```
4  // Define la configuración como distancias simétricas
5  Configuration configuration = new Configuration();
6  configuration.setSymmetric("yes");
7  tspProblem.setConfiguration(configuration);
8
9  // Crea las localidades
10 String[] locations = new String[5];
11 tspProblem.setLocations(locations);
12
13 locations[0] = "Salamanca";
14 locations[1] = "Madrid";
15 locations[2] = "Cáceres";
16 locations[3] = "Avila";
17 locations[4] = "Segovia";
```

```
18
19  // Crea la lista de distancias entre localidades
20  Distance[] distances = new Distance[10];
21  tspProblem.setDistances(distances);
22
23  // Crea un objeto distancia y fija el valor de sus campos
24  Distance distance1 = new Distance();
25  distance1.setFrom("Salamanca");
26  distance1.setTo("Madrid");
27  distance1.setLength(213.0);
28  distances[0] = distance1;
29
30  Distance distance2 = new Distance();
31  distance2.setFrom("Salamanca");
32  distance2.setTo("Cáceres");
33  distance2.setLength(201.0);
34  distances[1] = distance2;
35
36  Distance distance3 = new Distance();
37  distance3.setFrom("Salamanca");
38  distance3.setTo("Avila");
39  distance3.setLength(111.0);
40  distances[2] = distance3;
41
42  Distance distance4 = new Distance();
43  distance4.setFrom("Salamanca");
44  distance4.setTo("Segovia");
45  distance4.setLength(170.0);
46  distances[3] = distance4;
47
48  Distance distance5 = new Distance();
49  distance5.setFrom("Madrid");
50  distance5.setTo("Cáceres");
51  distance5.setLength(296.0);
52  distances[4] = distance5;
53
54  Distance distance6 = new Distance();
55  distance6.setFrom("Madrid");
56  distance6.setTo("Avila");
57  distance6.setLength(112.0);
58  distances[5] = distance6;
59
60  Distance distance7 = new Distance();
61  distance7.setFrom("Madrid");
62  distance7.setTo("Segovia");
63  distance7.setLength(94.0);
64  distances[6] = distance7;
65
66  Distance distance8 = new Distance();
67  distance8.setFrom("Cáceres");
68  distance8.setTo("Avila");
69  distance8.setLength(235.0);
70  distances[7] = distance8;
71
72  Distance distance9 = new Distance();
73  distance9.setFrom("Cáceres");
74  distance9.setTo("Segovia");
75  distance9.setLength(300.0);
76  distances[8] = distance9;
77
```

```
78  Distance distance10 = new Distance();
79  distance10.setFrom("Avila");
80  distance10.setTo("Segovia");
81  distance10.setLength(112.0);
82  distances[9] = distance10;
```

```
83  // Crea el acceso al servicio web
84  WsTSPService service = new WsTSPServiceLocator();
85  servicioWebTSP.WsTSP port = service.getwsTSPPort();
```

```
86  // Lanza la petición de resolución
87  Request oResultado = port.solve("usuario", "password", tspProblem);
88
89  if (oResultado.getStatus().equals("Failed")) {
90      // Tratar los errores
91  } else if (oResultado.getStatus().equals("Ok")) {
92      // Espera a que se ejecute el problema
93    WsProgressService serviceExecutionProgress = new
94      WsProgressServiceLocator();
95    WsProgress portProgress = serviceExecutionProgress.getwsProgressPort();
96
97      String strEstado = "";
98      boolean bTerminado = false;
99      while (!bTerminado) {
100         strEstado = portProgress.checkStatus("usuario", "password",
101           oResultado.getTicket());
102
103         if (strEstado.equals("UNKNOWN") || strEstado.equals("ERROR") ||
104           strEstado.equals("PROCESSED")) {
105             bTerminado = true;
106         } else if (strEstado.equals("ALLOCATED") ||
107            strEstado.equals("IN_PROCESS")) {
108             Thread.sleep(5000);
109         } else // Estado desconocido
110         {
111             bTerminado = true;
112         }
113     }
114
```

```
115     // Lee el resultado del problema
116     if (strEstado.equals("PROCESSED")) {
117         // Lee la solución
118         TspSolution tspSolution = port.getSolution("usuario", "password",
119           oResultado.getTicket());
120         // Tratar la solución.
121     } else {
122         // Tratar el error
123     }
124 }
```

Figura 3-14. Código fuente Java en Eclipse para la ejecución del TSP de VETU

4 Mensajes

En este capítulo se muestran todos los mensajes que pueden devolver las funciones de los servicios web de VETU, para la resolución del TSP.

4.1 Mensajes de envío de problemas

La Tabla 4-1 muestra la lista de mensajes que puede devolver el objeto Request de la función solve() del servicio web wsTSP, cuando se envía una solicitud de resolución de una instancia del TSP.

Código	Descripción
SRV0	The problem has been received. It is ready to solve. El problema enviado es correcto y está listo para su resolución. Este proceso de resolución puede tardar varios minutos. Se puede comprobar el estado de resolución, y una vez terminado, se puede acceder a la solución.
SRV1	User authentication failed. El usuario no se ha podido autenticar correctamente.
SRV2	The problem does not have a valid format. El formato de la entrada del problema no corresponde con el definido por la plataforma VETU, y no se puede resolver.
SRV3	The user has not access to this problem. El usuario enviado con la función solve() no tiene contrato para resolver el problema de VETU.
SRV4	The contract has expired El contrato para acceder a resolución del problema de VETU ha caducado y no puede ser utilizado el servicio.

Tabla 4-1. Mensajes que devuelve la función solve() del servicio web wsTSP

4.2 Mensajes del TSP

La siguiente tabla muestra los mensajes de error que se pueden mostrar en la resolución de una instancia del problema del viajante, resuelto mediante los servicios web de VETU.

Código	Descripción
TSP0	The TSP problem has been successfully solved El problema se ha resuelto correctamente.
TSP1	There are bugs in the entry file. El formato del fichero de entrada del TSP no corresponde con el definido por la plataforma VETU, y el problema no se puede resolver.
TSP2	Error solving the problem. Please contact support team. Ha ocurrido un problema inesperado en la resolución del problema. El equipo técnico de VETU ya lo ha detectado y está estudiando el problema. Puede ponerse en contacto con el servicio técnico.
TSP3	Problem data has not defined the Configuration object. El objeto Configuration define las opciones de configuración del prolema. Es necesario especificar este objeto.
TSP4	Problem data has not defined Locations object. El objeto Locations define las localidades a visitar. Es necesario definir las localidades.
TSP5	Problema data has not defined Distances object. El objeto Distances define la distancia entre todas las localidades. Es necesario definir la distancia entre todas las localidades.
TSP6	Configuration object has not defined the Symmetric object. El objeto Configuration debe definir el objeto Symmetric, que indica si las distancias entre localidades son simétricas.
TSP7	The value of Symmetric object is not valid. It must be either 'yes' or 'no' El valor del objeto Symmetric no es válido. Este objeto solo puede tener dos valores: 'yes' signfica que las distancias son simétricas, y 'no' significa que las distancias no son simétricas.
TSP8	Some Location object has an empty value. Alguno de los objetos Location se ha definido sin valor. Es necesario especificar un valor único para cada localidad.
TSP9	Some Distance object has not defined From object.

	Falta la definición del campo From en algún objeto Distance. Es necesario especificar siempre este campo.
TSP10	Some Distance object has defined an empty value for From object. Algún objeto Distance no ha definido un valor para el campo From. Es necesario especificar la localidad de origen de la distancia del objeto Distance.
TSP11	Some Distance object has not defined To object. Falta la definición del campo To en algún objeto Distance. Es necesario especificar siempre este campo.
TSP12	Some Distance object has defined an empty value for To object. Algún objeto Distance no ha definido un valor para el campo To. Es necesario especificar la localidad de destino de la distancia del objeto Distance.
TSP13	Some Distance object has not defined Length object. Falta la definición del campo Length en algún objeto Distance. Es necesario especificar siempre este campo.
TSP14	Some Distance object has defined an empty value for Length object. Algún objeto Distance no ha definido un valor para el campo Length. Es necesario especificar la distancia entre las localidades del objeto Distance.
TSP15	Some Distance object has defined the not valid value xxx for Length object. Algún objeto Distance no ha definido un valor correcto para el campo Length. El tipo de dato del campo Length es decimal positivo.
TSP16	There must be at least two locations for solving a TSP. El problema del viajante solo tiene sentido para más de dos localidades.
TSP17	Location xxx is referenced at Distances object, but it is not defined at Locations object. La localidad xxx que se referenciada en algún objeto Distance, no está definida en el objeto Locations. Todas las localidades deben estar definidas en el objeto Locations.
TSP18	Distance between xxx and yyy is not defined. No se ha definido la distancia entre las localidades xxx e yyy. Son necesarias

	todas las distancias entre cada dos localidades.

5 Próximas versiones

Este manual corresponde a la versión de la resolución del TSP que publica VETU actualmente. Aplicando la política de consolidación del sistema SaaS de VETU, y para ofrecer un mejor servicio a sus clientes, Infozara está trabajando continuamente en la mejora de sus productos y servicios. Estas mejoras para el TSP vienen dadas por la inclusión de nuevos condicionantes que permitan que el problema pueda reflejar más requerimientos de negocio, que permitan que el sistema pueda aplicarse en más empresas, y abrir las posibilidades de optimización, que lleven a la reducción de costes de la operativa de los clientes de Infozara.

En este capítulo se describe la nueva funcionalidad que tendrá la próxima versión del servicio SaaS que resuelve el TSP en VETU. Si desea más información sobre estas nuevas funciones, puede ponerse en contacto con el servicio técnico de Infozara.

5.1 Ventanas de tiempo

Cada localidad a visitar puede tener una ventana tiempo para realizar la vista. Esta condición puede cambar el orden en que el viajante realiza la ruta. Como ejemplo, en vez de indicar únicamente que se debe visitar Madrid, se puede decir que la visita a Madrid debe realizarse entre las 15:00 y las 16:00.