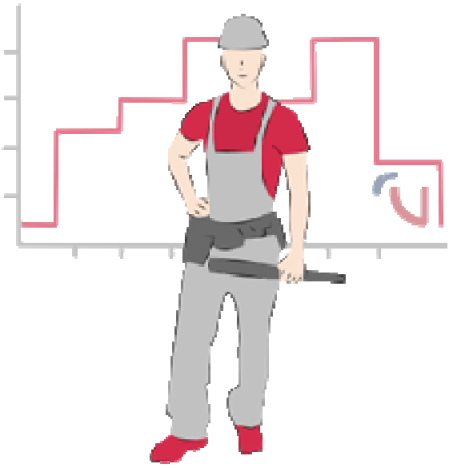
Sistema SaaS para la resolución del problema de planificación táctica de personal



Manual de usuario

Versión 1.0

infozara.es

© 2015 Infozara

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

La información que contiene este documento está legamente protegida y es confidencial a Infozara, sus clientes, y a quienes Infozara lo entregue expresamente con el propósito de evaluar el sistema Vetu. No se puede reproducir este documento de ninguna forma mecánica ni electrónica, incluyendo archivos electrónicos, sin el consentimiento expreso de Infozara.

ÍNDICE

1 Introducción ... 4
1.1 En este manual ... 4
1.2 SaaS ... 5
1.3 VETU ... 5
1.4 ¿Quiénes somos? ... 5
1.5 Planificación de personal ... 6
1.6 Planificación táctica de personal ... 6
1.7 Nomenclatura ... 7
1.8 Soporte ... 7
1.9 Actualización de versiones ... 7
1.10 Más información ... 8
2 Planificación táctica de personal ... 9
2.1 Modelo conceptual del problema ... 9
2.2 El problema al detalle ... 10
2.3 Naturaleza del problema ... 15
3 Acceso a los servicios de VETU ... 22
3.1 Arquitectura general de todo el sistema ... 22
3.2 Servicios Web ... 22
3.2.1 Servicio web de comprobación de estado de problemas ... 23
3.2.2 Servicio web de resolución ... 23
3.3 Proceso de ejecución mediante servicios web ... 30
3.4 Ejemplo de uso desde .NET ... 31
3.5 Ejemplo de uso desde NetBeans ... 35
3.6 Ejemplo de uso desde Eclipse ... 40
4 Casos de uso ... 47
4.1 Plantilla con un tipo de contrato ... 47
4.2 Plantilla con dos tipos de contrato ... 48
4.3 Plantilla con tres tipos de contrato ... 49
4.4 Plantilla de un contrato con jornadas partidas ... 51
4.5 Plantilla de varios contratos con jornadas partidas ... 51
4.6 Contrato definido por turnos de trabajo ... 53
4.7 Demanda de carga de trabajo de lunes a viernes ... 55
4.8 Plantillas diferentes para diario y fin de semana ... 56
4.9 Refuerzo para el fin de semana ... 58
4.10 Plantillas con más de dos días de descanso semanal ... 62
4.11 Conclusión ... 63
5 Listas de mensajes ... 64
5.1 Mensajes de envío de problemas ... 64
5.2 Mensajes del TWFP ... 65
6 Próxima versión ... 69
6.1 Horas en las que son se pueden partir jornadas ... 69
6.2 Patrones de jornada definidos por el usuario ... 69
6.3 Varios objetos WorkersAmount ... 69
7 Anexos ... 70
7.1 Carga de trabajo de los casos de uso ... 70

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 es la planificación táctica de personal, que permite definir la plantilla de trabajadores que son necesarios para cubrir una determinada carga de trabajo con el coste mínimo.

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

Es un documento dirigido tanto al gestor de recursos de la empresa, que se encarga de realizar los cuadrantes de personal, 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.

- Planificación táctica de personal

  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 de planificación táctica de personal, mediante los servicios web que publica VETU.

- Casos de uso

  Se indican diversos ejemplos de uso, que sirven para conocer al detalle el alcance del sistema SaaS de planificación táctica de personal de VETU.

- Lista de mensajes

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

- Próxima versión

  Se describen las nuevas funcionalidades previstas para la próxima versión del sistema de planificación táctica de personal que se publicará en VETU.

- Anexos

  Se muestra diversa información referenciada en el resto de este documento.

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 de planificación táctica de personal, pero publica más, como el problema de 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 Planificación de personal

En este manual de usuario definimos el problema de planificación de personal (*Work Force Problem*, WFP) como el problema de definir los turnos de la plantilla de personal que cubre una carga de trabajo dada.

En función del objetivo que se busque, existen dos variantes del WFP:

- Planificación táctica de personal (*tactical workforce problem*, TWFP)

  El objetivo es calcular la dimensión de plantilla que cubre una carga de trabajo al menor coste. Un ejemplo de la salida de este problema es: 'para cubrir una carga de trabajo determinada, son necesarios 10 trabajadores de 40 horas semanales y 15 trabajadores de 20 horas semanales, con un coste total de xxx euros'.

  Este problema se utiliza para confeccionar la plantilla de trabajadores de un nuevo servicio, para dimensionar plantillas en diversas temporadas en las que hay variaciones de carga de trabajo, o para dimensionar plantillas de empresas de nueva creación.

- Planificación operativa de personal (*operational workforce problem*, OWFP)

  El objetivo es definir los turnos de una determinada plantilla de trabajadores, con el menor coste y con los mejores turnos para el personal, cubriendo una determinada carga de trabajo. Un ejemplo de salida de este problema es: 'para cubrir una carga de trabajo determinada, Juan descansa el martes y el miércoles y entra a trabajar el resto de los días a las 08:00, Luis descansa el sábado y el domingo y entra a trabajar el resto de los días a las 16:00 …, con un coste total de xxx euros'.

Este documento se refiere exclusivamente al problema de planificación táctica de personal (TWFP).

Próximamente, VETU también tiene previsto publicar un servicio SaaS, para la resolución del problema de planificación operativa de personal. En la web de VETU www.vetu.es podrá consultar información más detallada de este problema cuando sea publicado.

1.6 Planificación táctica de personal

El problema de planificación táctica de personal (TWFP) busca la configuración de plantilla de

menor coste que cubre una carga de trabajo. Las entradas del problema son:

- Carga de trabajo temporal, que expresa la cantidad de trabajadores que son necesarios para dar un determinado servicio.
- Contratos que definen las características de las jornadas de trabajo que pueden realizar los empleados.

El objetivo es encontrar la cantidad de trabajadores de cada tipo de contrato, que cubren la carga de trabajo con el menor coste posible.

Este problema se define en detalle en el capítulo 2, y en el capítulo 4 se muestran una serie de casos de uso, que ayudan a comprender el enfoque, dimensión y versatilidad del TWFP que publica VETU.

Si el problema de planificación de personal se su empresa difiere del aquí presentado, o presenta alguna característica no recogida en el TWFP de VETU, puede ponerse en contacto con Infozara por cualquiera de los medios indicados en el apartado 1.11, para platear una solución satisfactoria para usted.

1.7 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.
WFP	Problema de planificación de personal (*workforce problem*)
TWFP	Problema de planificación táctica de personal, (tactical workforce *problem*), introducido en apartado 1.6, y descrito en mayor detalle en el capítulo 2.
OWFP	Problema de planificación operativa de personal, (operational workforce problem), introducido en apartado 1.5.
UM	Unidades monetarias.

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 EntryTWFP.

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

1.8 Soporte

Si es usted cliente de la plataforma VETU de Infozara, puede obtener soporte técnico sobre el TWFP 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.

1.9 Actualización de versiones

Todos los problemas que ofrece la plataforma VETU, y en particular el TWFP, 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 6 muestra una descripción de las nuevas funciones que incorporará la siguiente versión del TWFP de VETU.

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

1.10 Más información

Si desea más información sobre cualquier aspecto del sistema SaaS de planificación táctica de personal 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
	twfp@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/planificacionTacticaPersonal.do

2 Planificación táctica de personal

2.1 Modelo conceptual del problema

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

> 'El problema TWFP viene definido por una serie de contratos y una demanda de carga de trabajo. Los contratos definen las características que tienen que cumplir las jornadas de trabajo de los empleados. La carga de trabajo se expresa como la cantidad de trabajadores necesaria para cada franja horaria. La solución del problema es el conjunto de jornadas que cubren la carga de trabajo y los días de descanso semanal para cada tipo de contrato, al menor coste'.

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

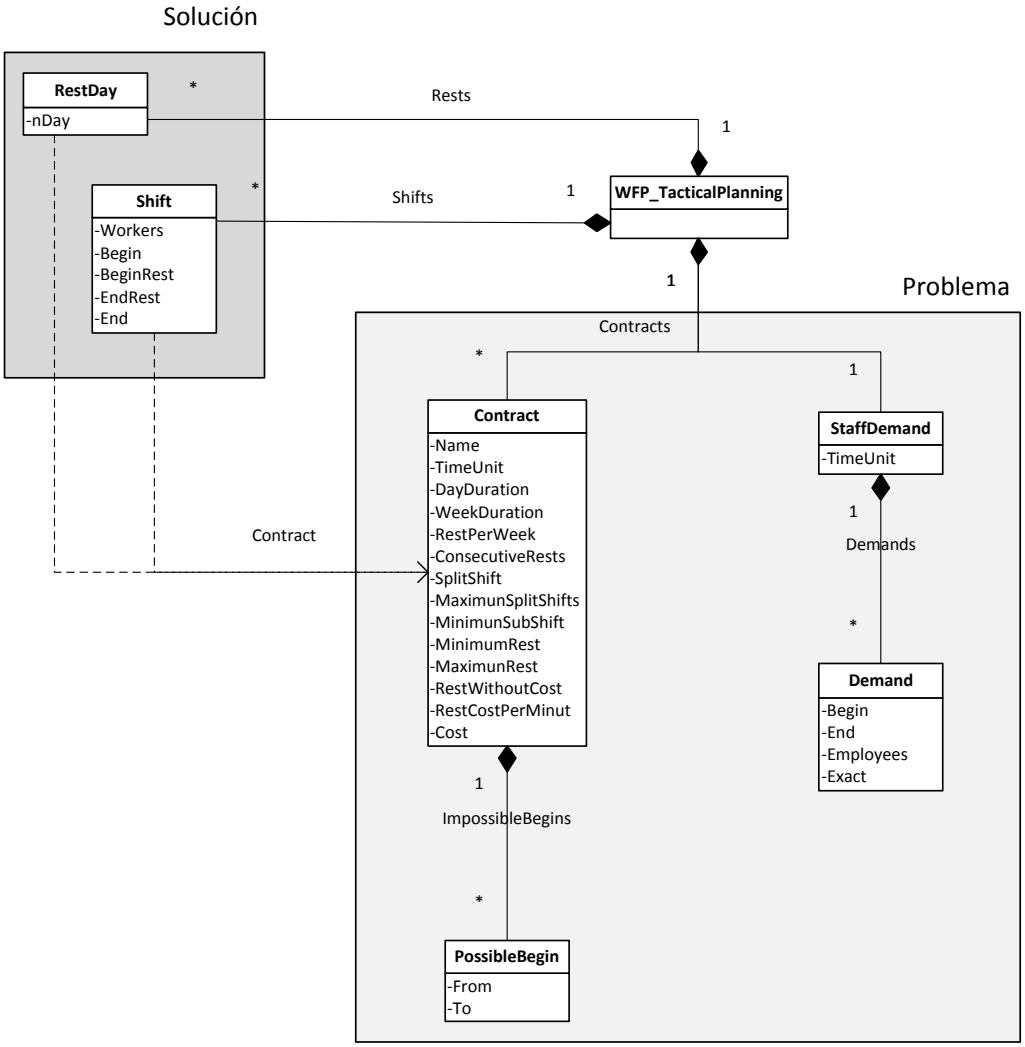


Figura 2-1. Modelo conceptual del TWFP

2.2 El problema al detalle

El TWPF presenta un modelo de contrato con una serie rica de parámetros, que permiten una alta combinación de posibilidades que puede llevar a plantillas de dimensión ajustada a la demanda de carga de trabajo, y con coste mínimo. La Tabla 2-1 muestra la definición de cada uno de los parámetros que definen un contrato.

Parámetro	Descripción	Tipo
Name	Nombre del contrato.	Texto
TimeUnit	Intervalo horario al que pueden comenzar las jornadas de trabajo, expresado en minutos.	Entero
DayDuration	Duración de la jornada diaria de trabajo, expresado en minutos.	Entero
RestsPerWeek	Cantidad de días de descanso semanales.	Entero
ConsecutiveRests	Indica si los días de descanso deber ir seguidos.	Booleano
SplitShift	Indica si se pueden asignar jornadas partidas a los trabajadores de este tipo de contrato.	Booleano
MaximunSplitShifts	Cantidad máxima de jornadas partidas que puede realizar a la semana un trabajador de este tipo de contrato. Solo aplica si SplitShift = true.	Entero
MinimunSubShfit	Duración mínima de las dos subjornadas que se realizan en una jornada partida, expresada en minutos. Solo aplica si SplitShift = true.	Entero
MinimumRest	Duración mínima del descanso en una jornada partida, expresada en minutos. Solo aplica si SplitShift = true.	Entero
MaximumRest	Duración máxima del descanso en una jornada partida, expresada en minutos. Solo aplica si SplitShift = true.	Entero
RestWithoutCost	Duración del descanso que no genera coste, expresada en minutos. Solo aplica si SplitShift = true.	Entero
RestCostPerMinut	Coste por minuto del descanso de la jornada partida que genera coste, expresada en unidad monetaria por minuto. Solo aplica si SplitShift = true.	Entero
Cost	Coste de la jornada, expresado en unidad monetaria.	Entero
PossibleBegins	Lista de posibles comienzos de las jornadas de trabajo de este contrato.	Lista
PossibleBegin	Intervalo de posible comienzo de las jornadas de trabajo de este contrato, a lo largo de la semana.	Objeto
PossibleBegin.From	Comienzo del intervalo del posible comienzo de una jornada de este contrato, expresado en minutos.	Entero
PossibleBegin.To	Fin del intervalo del posible comienzo de una jornada de este contrato, expresado en minutos.	Entero
WorkersAmount	Permite fijar una restricción sobre la cantidad de trabajadores de un contrato que debe tener la plantilla a calcular. Este objeto puede estar vacío, lo que significa que no se pone ni ninguna restricción	Objeto

	sobre el número de trabajadores de ese contrato.	
WorkersAmount.Criterion	Criterio para fijar la cantidad de trabajadores. Puede tomar los valores: • LessOrEqual: la cantidad de trabajadores de este contrato debe ser menor que la cantidad definida por el campo Amount. • Equal: la cantidad de trabajadores de ese contrato debe ser igual que la cantidad definida por el campo Amount. • GreaterOrEqual: la cantidad de trabajadores de ese contrato debe ser mayor que la cantidad definida por el campo Amount.	String
WorkersAmount.Amount	Cantidad de trabajadores definida para limitar el número de trabajadores.	Entero

Tabla 2-1. Definición de los parámetros de un contrato

Para clarificar el significado de los parámetros de definición de los contratos, a continuación se van a mostrar una serie de ejemplos de definición de contratos.

La Tabla 2-2 muestra la definición de un contrato de 40 horas semanales, con una jornada diaria de 480 minutos (8 horas), dos días de descanso seguidos por semana (por tanto, 5 días de trabajo a la semana). No se pueden realizar jornadas partidas, el coste de la jornada diaria es de 50 unidades monetarias, y las jornadas pueden comenzar en cualquier momento de la semana (una semana tiene 10.080 minutos). TimeUnit tiene el valor de 60 minutos, lo que significa que las jornadas pueden comenzar cada hora. Debe haber como máximo 25 trabajadores de este tipo de contrato.

Parámetro	**Valor**
Name	40H
TimeUnit	60
DayDuration	480
RestsPerWeek	2
ConsecutiveRests	True
SplitShift	False
MaximunSplitShifts	
MinimunSubShfit	
MinimumRest	
MaximumRest	
RestWithoutCost	
RestCostPerMinut	
Cost	50
PossibleBegin.Comienzo	0
PossibleBegin.Fin	10.080
WorkersAmount.Criterion	LessOrEqual
WorkersAmount.Amount	25

Tabla 2-2. Contrato de 40 horas semanales

La Tabla 2-3 muestra un contrato de trabajo de 16 horas semanales, en el que los trabajadores suscritos al contrato solo trabajan los fines de semana, en jornadas de 8 horas diarias (en sábado y domingo). Los posibles comienzos de jornada van desde las 00:00 del

sábado (minuto 7.200) a las 00:00 del domingo (minuto 10.080). Pueden partir la jornada de trabajo en dos partes de 4 horas (240 minutos) y una hora (60 minutos) de descanso. De esta forma, un trabajador de este tipo de contrato puede hacer dos tipos de jornada: jornada continua de 8 horas o jornada partida de una hora de descanso y dos subjornadas de 4 horas. Pueden partir la jornada de trabajo los dos días del fin de semana. No se paga por la jornada partida, porque el tiempo máximo de descanso sin cobrar es de una hora. TimeUnit tiene el valor de 30 minutos, lo que significa que las jornadas pueden comenzar cada media hora. Es la definición de un contrato de refuerzo de fin de semana. No se establece ninguna restricción sobre la cantidad de trabajadores de este contrato.

Parámetro	Valor
Name	16H- Fin semana
TimeUnit	30
DayDuration	480
WeekDuration	960
RestsPerWeek	5
ConsecutiveRests	true
SplitShift	true
MaximunSplitShifts	2
MinimunSubShfit	240
MinimumRest	60
MaximumRest	60
RestWithoutCost	60
RestCostPerMinut	0
Cost	50
PossibleBegin.Comienzo	7.200
PossibleBegin.Fin	10.080
WorkersAmount	

Tabla 2-3. Contrato de 16 horas semanales en fin de semana

La Tabla 2-4 muestra la definición de un contrato de 40 horas semanales en turno de tarde, durante todos los días de la semana. Se considera que el turno de tarde comienza todos los días, entre las 16:00 (minuto 960) y las 18:00 (minuto 1.080). Los trabajadores de este contrato descansan dos días a la semana, que pueden estar separados. Cada trabajador puede realizar 3 jornadas partidas a la semana, con una subjornada mínima de 3 horas (180 minutos), un descanso mínimo de una hora (60 minutos) y máximo de 3 horas (180 minutos). Si el descanso de la jornada partida excede de una hora, se paga a 0,17 unidades monetarias por minuto. Con esta configuración de jornadas partidas, los diferentes tipos de jornada que pueden realizar los trabajadores son las que se muestran en la Tabla 2-5, donde la primera fila corresponde a la jornada continua, y el resto a jornadas partidas.

Parámetro	Valor
Name	40H- Tarde
TimeUnit	60
DayDuration	480
RestsPerWeek	2

ConsecutiveRests	false
SplitShift	true
MaximunSplitShifts	3
MinimunSubShfit	180
MinimumRest	60
MaximumRest	180
RestWithoutCost	60
RestCostPerMinut	0,17
Cost	50
PossibleBegin.Comienzo	960
PossibleBegin.Fin	1.080
PossibleBegin.Comienzo	2.400
PossibleBegin.Fin	2.520
PossibleBegin.Comienzo	3.840
PossibleBegin.Fin	3.960
PossibleBegin.Comienzo	5.280
PossibleBegin.Fin	5.400
PossibleBegin.Comienzo	6.720
PossibleBegin.Fin	6.840
PossibleBegin.Comienzo	8.160
PossibleBegin.Fin	8.280
PossibleBegin.Comienzo	9.600
PossibleBegin.Fin	9.720
WorkersAmount	

Tabla 2-4. Contrato de 40 horas semanales en turno de tarde

Duración de la primera subjornada	Duración del descanso	Duración de la segunda subjornada	Coste de la jornada
8 horas	0 horas	0 horas	50
3 horas	1 hora	5 horas	50
3 horas	2 hora	5 horas	50 + 0,17 * 60 = 60,2
3 horas	3 hora	5 horas	50 + 0,17 * 120 = 70,4
4 horas	1 hora	4 horas	50
4 horas	2 hora	4 horas	50 + 0,17 * 60 = 60,2
4 horas	3 hora	4 horas	50 + 0,17 * 120 = 70,4
5 horas	1 hora	3 horas	50
5 horas	2 hora	3 horas	50 + 0,17 * 60 = 60,2
5 horas	3 hora	3 horas	50 + 0,17 * 120 = 70,4

Tabla 2-5. Jornadas para el contrato de 40 horas semanales en turno de tarde

El campo TimeUnit de Contract solo puede tomar los valores 30 (los turnos pueden comenzar cada media hora) y 60 (los turnos pueden comenzar cada hora).

El objeto WorkersAmount permite poner restricciones sobre la cantidad de trabajadores de un tipo de contrato que debe haber en una plantilla. Se pueden especificar cantidades mínimas, máximas o exactas de trabajadores.

La carga de trabajo se expresa como cantidad de trabajadores que son necesarios en una determinada franja horaria. La cantidad de trabajadores se puede expresar de dos formas:

- Cantidad mínima

  Significa que en esa franja horaria se desea un número mínimo de trabajadores, que puede ser superado en la solución obtenida.

- Cantidad exacta

  Significa que en esa franja horaria se quiere un número exacto de trabajadores, es decir, no se desea que en esa franja haya un exceso de trabajadores sobre la carga de trabajo. Se suele utilizar en las primeras horas con carga de trabajo del día.

Cabe destacar que no se debe hacer un uso elevado de cantidades exactas de trabajadores, porque puede llevar a plantear problemas sin solución. Si la carga de trabajo tiene mucha variación en periodos consecutivos, es posible que no exista ninguna combinación de jornadas que cubra de forma exacta esa carga de trabajo, y hay que dar libertad al sistema de cálculo para que fije un exceso de carga de trabajo para alcanzar el coste mínimo total.

La carga de trabajo se puede expresar en intervalos de una hora (1H), media hora (30M) o quince minutos (15M). Este valor se expresa en el campo TimeUnit del objeto StaffDemand.

La Tabla 2-6 muestra un ejemplo de definición de carga de trabajo para dos intervalos. Para el lunes de 09:00 a 10:00 (minutos 540 al 600) se especifica una carga de trabajo exacta de 5 trabajadores, lo que significa que el proceso de cálculo debe generar una planificación donde haya cinco turnos en ese intervalo; nunca se generará una planificación con más de 5 trabajadores en ese intervalo. Para el lunes de 10:00 a 11:00 (minutos 600 a 660) se especifica que debe haber al menos 5 trabajadores, lo que significa que el proceso de cálculo debe generar planificaciones donde haya 5, 6, 7 trabajadores en ese intervalo, de forma que se cubra la carga de trabajo a coste mínimo.

Intervalo		Demanda		Interpretación
L 09:00 a 10:00	540 a 600	5	Exacta	En la planificación, debe haber exactamente 5 trabajadores en ese intervalo.
L 10:00 a 11:00	600 a 660	5	Mínima	En la planificación, debe haber al menos 5 trabajadores en ese intervalo.

Tabla 2-6. Definición de carga de trabajo

En el apartado 7.1 se muestra la versión XML de una carga de trabajo definida a nivel de hora (TimeUnit=60), para toda la semana.

Resolver el problema significa:

- Calcular:
  - la cantidad de trabajadores de cada tipo de contrato,
  - los descansos semanales de los trabajadores,

  - las jornadas de los trabajadores,
  - el coste por contrato,
  - el coste total de la plantilla,
- todo ello:
  - cubriendo la carga de trabajo,
  - con un coste total mínimo.

Las fuentes de coste son:

- Coste de jornadas de cada contrato
- Coste de partir jornadas

2.3 Naturaleza del problema

El TWFP es un problema de optimización combinatoria, lo que significa que existen muchas soluciones diferentes, es decir, existen muchas plantillas de trabajadores que pueden cubrir una determinada carga de trabajo. Cada una de estas plantillas tiene un coste diferente, con lo que el problema consiste en encontrar la plantilla de coste mínimo que cubra una determinada carga de trabajo.

Desde un punto de vista matemático, existe un número finito de plantillas diferentes, pero es un número tan elevado que no permite generar todas las plantillas 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 sus clientes, sin que estos tengan la necesidad de conocer técnicas matemáticas, Investigación Operativa o programación informática: el cliente tiene un problema de planificación táctica de personal y que quiere que se resuelva de la mejor forma posible, sin preocuparse de la infraestructura que tiene implantada Infozara para resolver problemas de este tipo.

Como ejemplo de la complejidad de resolución del TWFP, vamos a plantear un problema sencillo, que iremos aumentando en dificultad, y posteriormente plantearemos un problema de mayor dimensión.

Supongamos una instancia de TWFP de un solo día, para cubrir la demanda de carga de trabajo que muestra la Figura. Esta demanda significa que de 08:00 a 10:00 son necesarios tres trabajadores, de 10:00 a 12:00 son necesarios 4 trabajadores, de 12:00 a 14:00 son necesarios 6 trabajadores, de 14:00 a 16:00 son necesarios 4 trabajadores, de 16:00 a 18:00 son necesarios 6 trabajadores, de 18:00 a 20:00 son necesarios 2 trabajadores. En total, se deben cubrir 50 horas de trabajo. El problema consiste en calcular las jornadas que cubren esta carga de trabajo, con el menor coste.

Comenzaremos con un tipo de contrato de 40 horas semanales, con jornadas de trabajo de 8 horas diarias, que no pueden partir jornada. El coste de la jornada de este contrato es de 40 UM. La solución a este problema se muestra en la Figura 2-3: son necesarios 9 trabajadores, que suponen 72 horas de trabajo, y un coste de 360 UM. Esto representa 32 horas de exceso de trabajo. Las jornadas de trabajo son tres trabajadores que empiezan a las 08:00 y terminan a las 16:00, 4 trabajadores que comienzan a las 10:00 y terminan a las 18:00, y 2 trabajadores que comienzan a las 12:00 y terminan a las 20:00. La línea roja representa la demanda de carga de trabajo y la línea azul representa la carga de trabajo que cubren las jornadas de trabajo calculadas. Si la línea azul está por encima de la roja, significa que hay exceso de trabajadores, y si la línea azul tapa a la roja, significa que la cantidad de

trabajadores en esa hora es igual a la demandada.

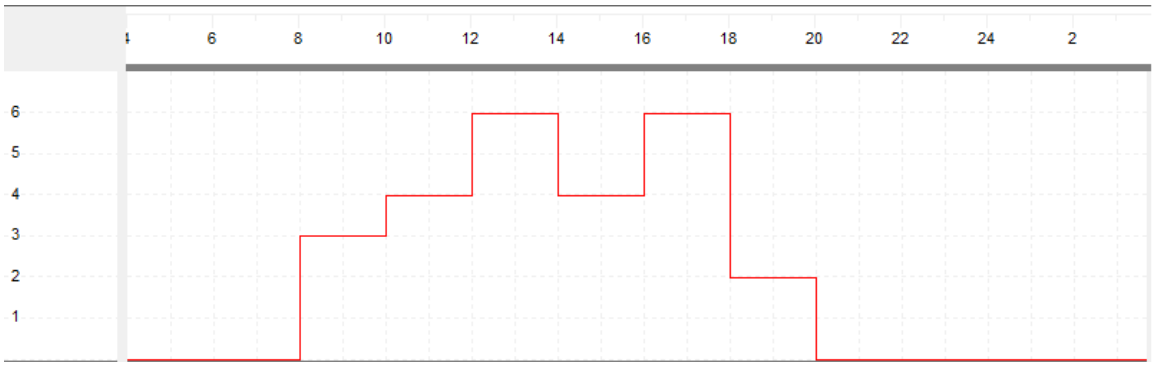


Figura 2-2. Carga de trabajo del ejemplo

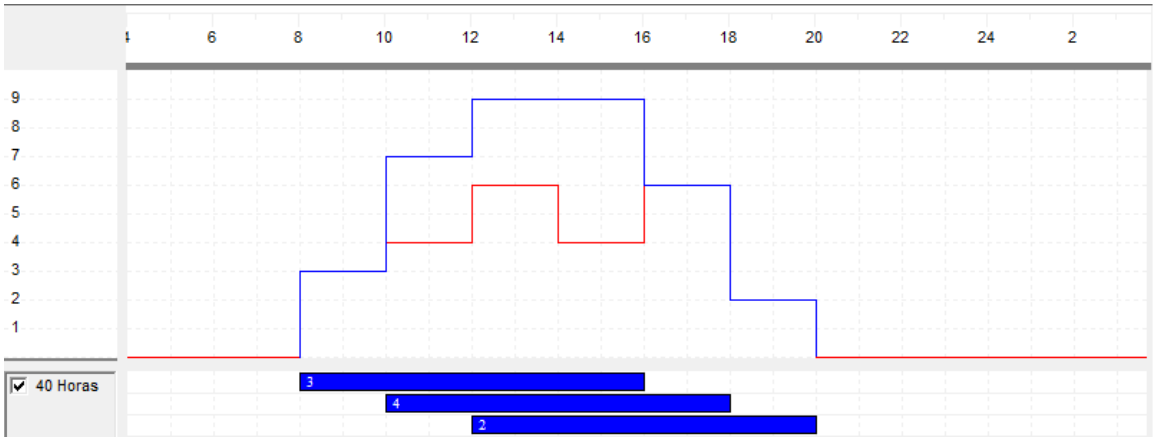


Figura 2-3. Solución para el contrato de 40 horas

Ahora vamos a introducir un condicionante diferente en este tipo de contrato: se pueden partir jornadas con un descanso entre 01:00 y 03:00, con una mínima subjornada de 02:00. La primera hora de jornada partida no se paga, y la segunda y tercera se pagan a 12 UM por hora de descanso. Ahora la solución es más difícil de encontrar que en el caso original, porque hay que decidir en cada hora, no solo la cantidad de trabajadores que comienzan jornada, sino además, el tipo de jornada que realizan (continua, partida con una hora de descanso, partida con dos horas de descanso, o partida con tres horas de descanso, con subjornadas de 2, 3 o 4 horas). Una posible solución es la que muestra la Figura 2-4. Ahora son necesarios 7 trabajadores con un coste de 292 UM y una carga de trabajo de 56 horas (exceso de 6 horas de carga de trabajo). Debido a que las jornadas partidas ofrecen mucha flexibilidad para ajustar la cargad de trabajo a la demanda, se ha reducido el número de trabajadores de 9 a 7, el exceso de horas de trabajo de 32 a 6, y el coste se ha reducido de 360 a 292. Las jornadas de esta solución corresponden a una jornada que comienza a las 06:00, descansa de 12:00 a 13:00 y termina a las 17:00, 2 jornadas que comienzan a las 08:00, descansan de 14:00 a 16:00 y terminan a las 18:00, una jornada que comienza a las 09:00, descansa de 11:00 a 12:00 y termina a las 18:00, una jornada que comienza a las 10:00, descansa de 15:00 a 16:00 y termina a las 19:00, y una jornada que comienza a las 12:00 y termina a las 20:00 (esta es la única jornada continua).

Sobre el caso inicial, podemos introducir otro tipo de contrato, un contrato de 30 horas semanales, y 6 horas diarias, que no parte jornada, y un coste de 30 UM por jornada. La solución de este caso se muestra en la Figura 2-5. En este caso son necesarios 8 trabajadores de 30 horas y uno de 40 horas, con un coste de 280 UM, con un exceso de carga de trabajo de 6 horas. La jornada de 8 horas comienza a las 10:00 y termina a las 18:00, y las jornadas

de seis horas son las restantes. En este caso aumenta la complejidad sobre el caso inicial, porque para resolver el problema, tenemos que calcular la cantidad de jornadas de cada tipo de contrato (40 o 30 horas) que comienzan cada hora.

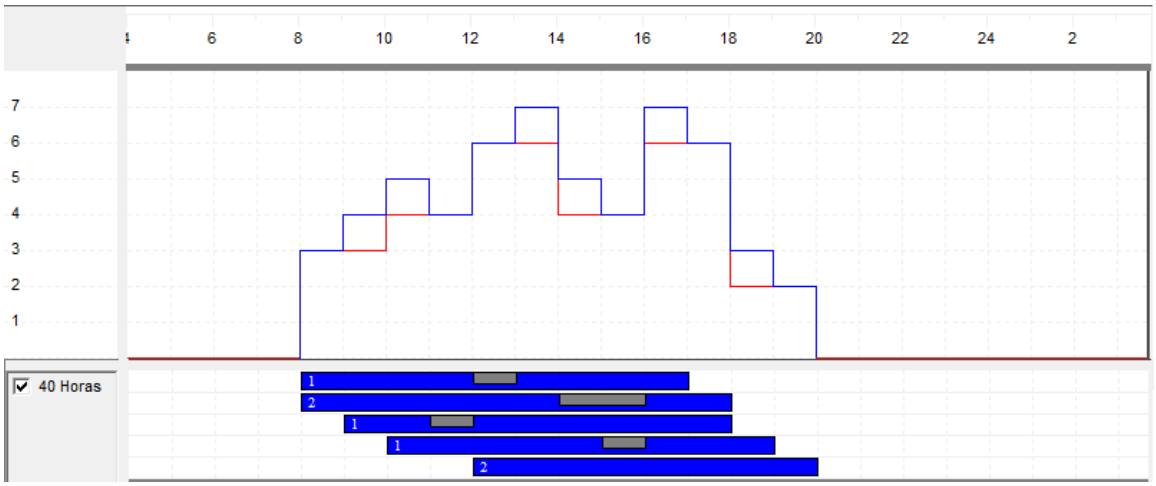


Figura 2-4. Solución para el contrato de 40 horas con jornadas partidas

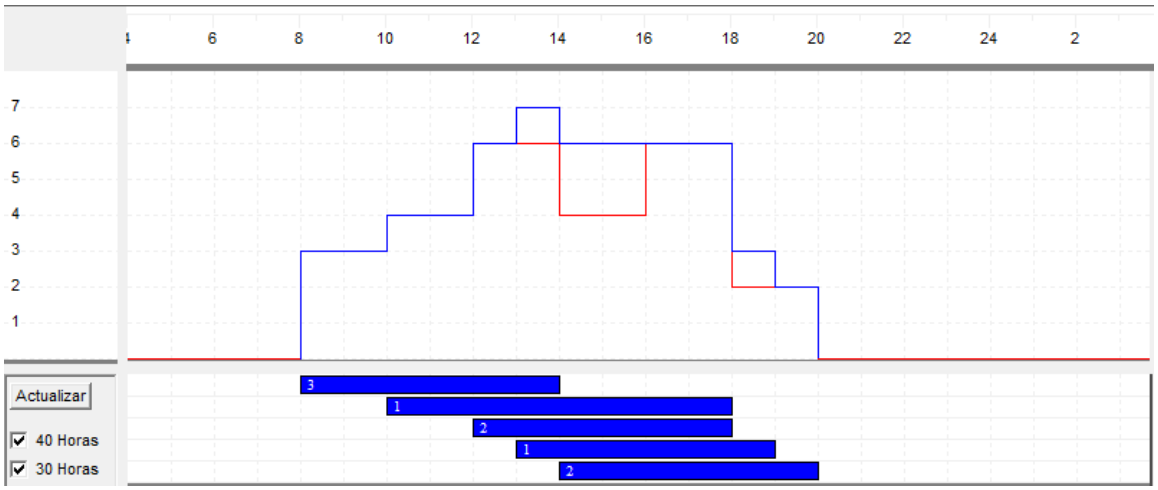


Figura 2-5. Solución para los contratos de 40 y 30 horas

Sobre el caso de los contratos de 40 y 30 horas vamos a introducir un nuevo tipo de contrato: 20 horas semanales, 4 horas diarias, sin partir jornada, con un coste de 20 UM por jornada. La solución se muestra en la Figura 2-6. En este caso se cubre la demanda de carga de trabajo de forma exacta, es decir, la carga de trabajo de las jornadas coincide con la demanda de carga de trabajo. Hay una jornada de 8 horas (de 10:00 a 18:00), 5 jornadas de 6 horas (2 de 08:00 a 14:00 y 3 de 12:00 a 18:00) y 3 jornadas de 4 horas (una de 08:00 a 12:00 y 2 de 16:00 a 20:00). El coste total de esta solución es 250 UM. Comparando estos resultados con el caso en que había dos tipos de contratos, vemos que el coste baja desde 292 UM hasta 250 UM. Si considerásemos la oportunidad de permitir jornadas partidas para los tres tipos de contratos, observaríamos no se puede mejorar la solución dada en la Figura 2-6, porque las jornadas partidas tiene coste. Este caso es más difícil de resolver porque hay más variables de cálculo (se deben calcular la cantidad de jornadas de tres tipos de contratos que comienzan a cada hora).

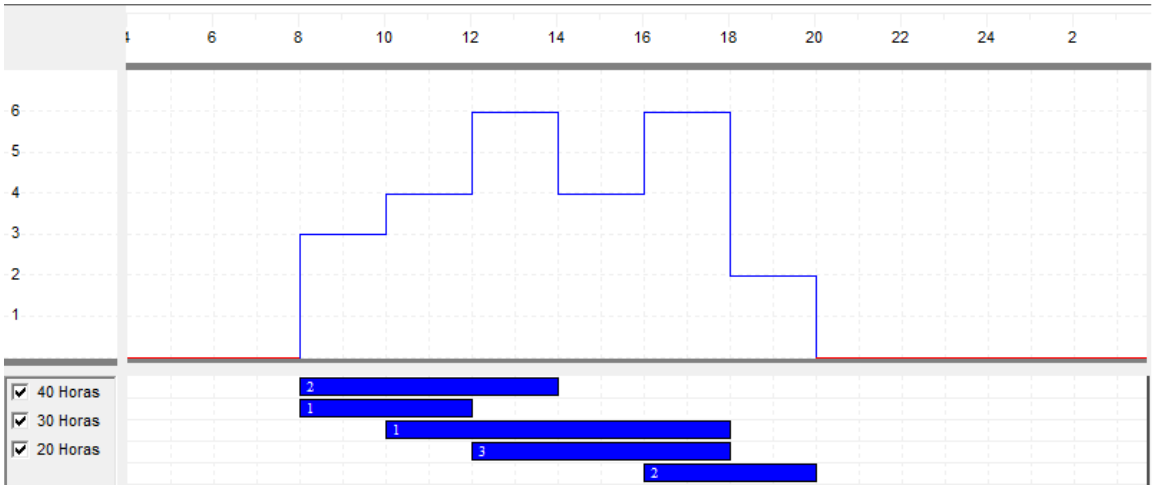


Figura 2-6. Solución para contratos de 40, 30 y 20 horas

Como conclusión, se puede decir que el TWFP es un problema que admite diversas soluciones. Las soluciones se calculan decidiendo en cada hora la cantidad de jornadas que deben empezar para cada tipo de contrato. A medida que aumentan los tipos de contratos (especialmente si hay jornadas partidas) aumenta el número de decisiones a tomar, y aumenta la complejidad de cálculo. Por otra parte, hemos resuelto el problema para un día por simplicidad, pero en realidad el problema se resuelve para una semana, donde hay más decisiones que tomar: los días de descanso de la plantilla, y la cantidad de jornadas partidas por día, para no superar el máximo semanal. El caso semanal es más difícil de resolver que el diario, porque hay más decisiones que tomar. El valor de cada una de estas decisiones se puede combinar con el valor del resto de decisiones, llevando a un número elevado de posibles soluciones, imposibles de tener en mente en una resolución manual, e imposible de computar en un ordenador. Por ese motivo se hace totalmente necesario el uso de algoritmos específicos, como el desarrollado por Infozara en la plataforma SaaS VETU, a disposición de las empresas que tienen la problemática de dimensionar plantillas de trabajo.

Para ver la complejidad en la resolución del TWFP con varios tipos de contrato y cargas de trabajo mayores de las presentadas en este apartado, en la Figura 2-7 se muestra el solución correspondiente al lunes, para la carga de trabajo definida en los casos del uso del capítulo 4, con los contratos definidos en la Tabla 2-7. Hay una demanda de carga de trabajo de 843 horas, y hay un exceso de carga de trabajo de 3 horas. Para observar con mayor claridad la composición de la plantilla, la Figura 2-8 muestra las jornadas para el contrato de 40 horas, la Figura 2-9 muestra las jornadas para el contrato de 30 horas, y la Figura 2-10 muestra las jornadas para el contrato de 20 horas. En estas figuras aparece una nueva línea (de color gris), que muestra la carga de trabajo de las jornadas del contrato en cuestión. Se observa que hay presencia de trabajadores de 40 horas durante todo el día, aunque su presencia es mayor durante el pico de la mañana; también hay presencia de trabajadores de 30 horas durante todo el día, pero su presencia es mayor durante el pico de la tarde; los trabajadores de 20 horas aparecen fundamentalmente en los picos, tanto de la mañana como de la tarde, y no tienen presencia ni a primera hora, ni a mediodía, ni a última hora del día.

Parámetro	40 horas	30 horas	20 horas
Name	40H	30H	40H
TimeUnit	60	60	60
DayDuration	480	360	240
RestsPerWeek	5	5	5
ConsecutiveRests	true	true	true

SplitShift	true	true	true
MaximunSplitShifts	2	2	2
MinimunSubShfit	120	120	120
MinimumRest	60	60	60
MaximumRest	60	60	60
RestWithoutCost	60	60	60
RestCostPerMinut	0	0	0
Cost	40	30	20
PossibleBegin.Comienzo	0	0	0
PossibleBegin.Fin	10.080	10.080	10.080
WorkersAmount			

Tabla 2-7. Definición de contratos

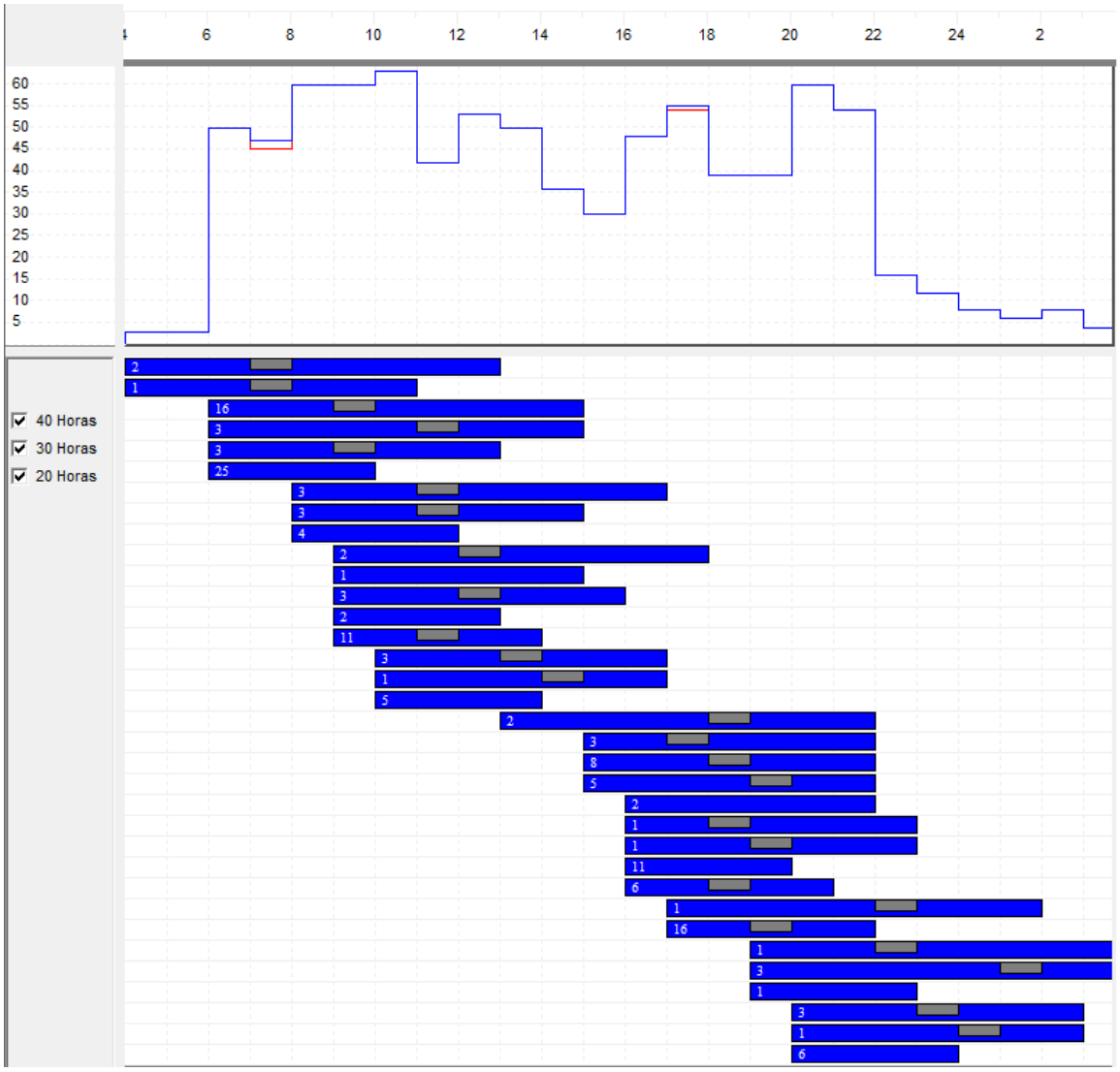


Figura 2-7. Solución del lunes para la carga de trabajo de los casos de uso

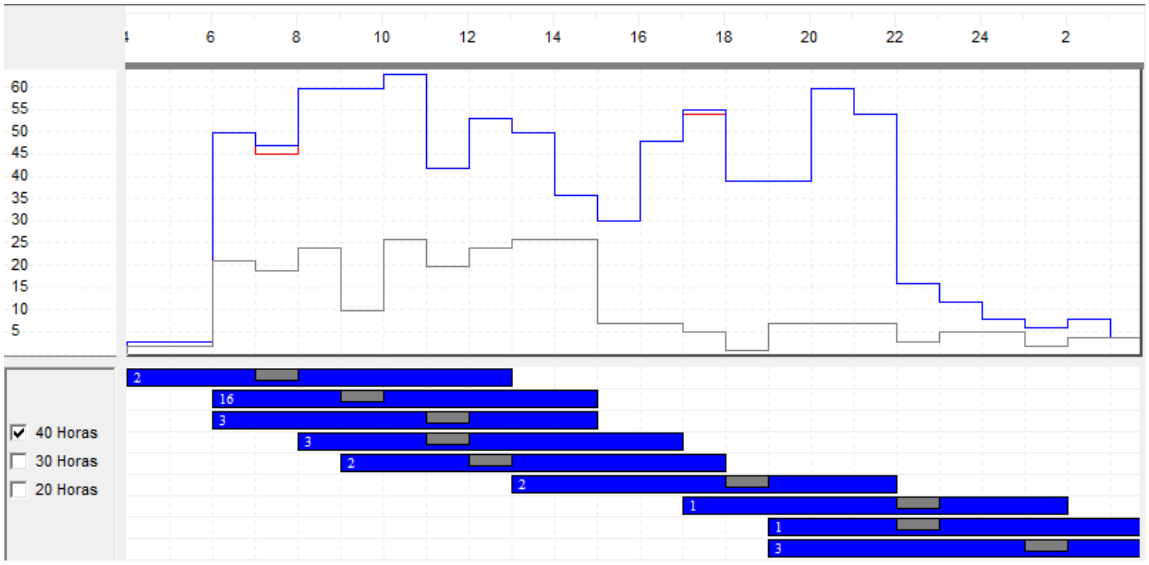


Figura 2-8. Solución para el contrato de 40 horas

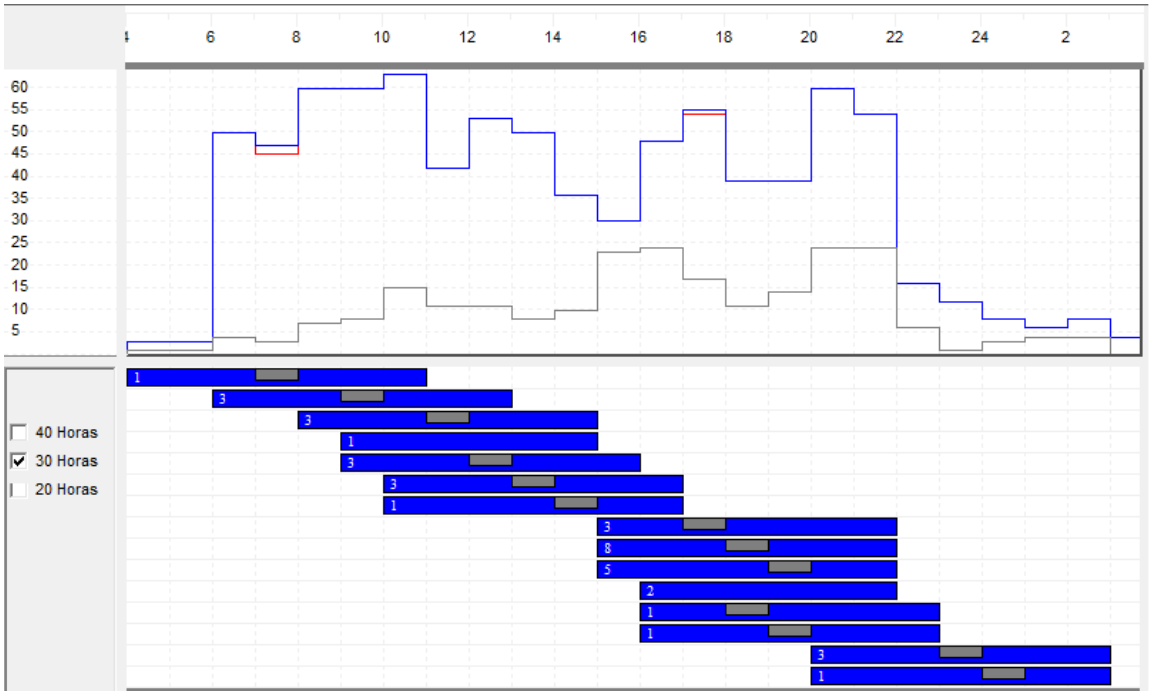


Figura 2-9. Solución para el contrato de 30 horas

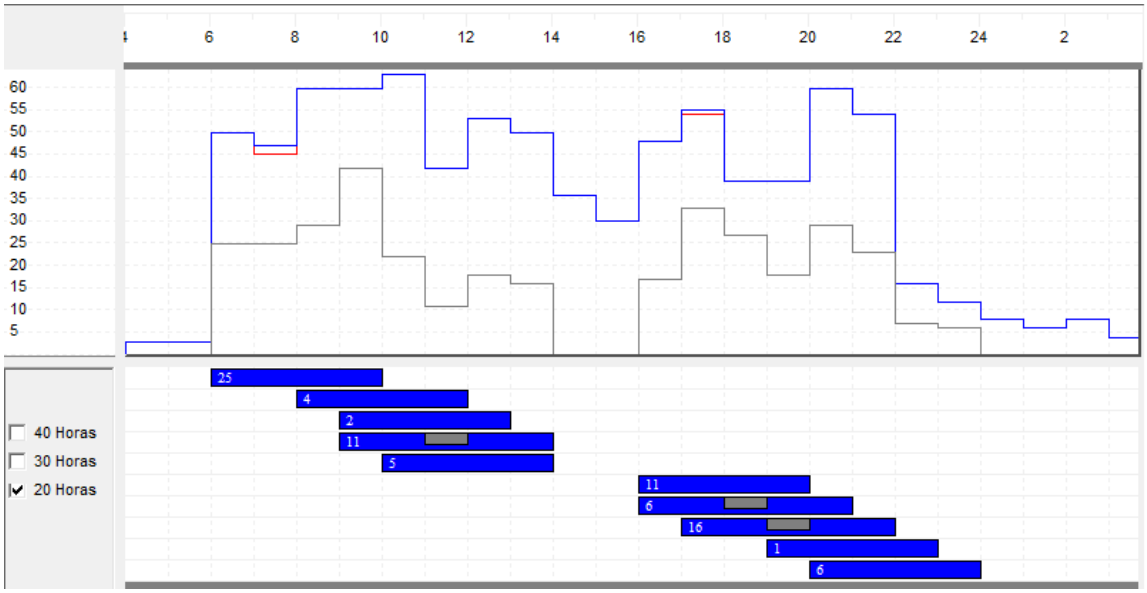


Figura 2-10. Solución para el contrato de 20 horas

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 de planificación táctica de personal.

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.

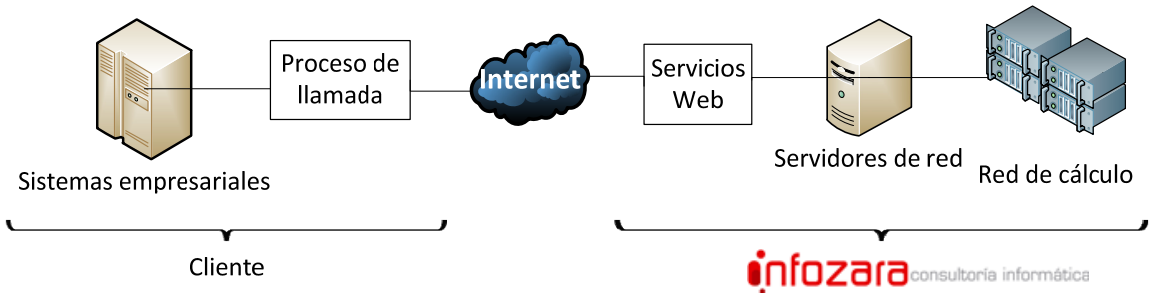


Figura 3-1. Arquitectura del sistema SaaS

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 TWFP no se resuelve de forma instantánea, sino que puedes 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 TWFP se llama wsTWFP 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 pasword, 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

El servicio web que permite la resolución del TWFP se llama wsTWFP, y tiene dos funciones

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

WfSolution getSolution(String username, Strign password, String nTicket)

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

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

La función solve() recibe la clase WFPTacticalPlanning, que encapsula la instancia del TWFP 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 succesfully 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 5.1.

```
<?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 WFPTacticalPlanning encapsula la información del problema que se quiere resolver. La Figura 3-4 muestra la versión XML de la definición de un TWFP. El elemento raíz TWFP_Problem se compone de dos objetos: StaffDemand, que define la carga de trabajo, y Contracts, que es una lista de objetos Contract, que definen un contrato.

```
<?xml version="1.0"?>
<TWFP_Problem>
  <StaffDemand>
    <TimeUnit>60</TimeUnit>
    <Demands>
      <Demand>
        <Begin>0</Begin>
        <End>1440</End>
        <Employees>5</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1440</Begin>
        <End>2880</End>
        <Employees>6</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>2880</Begin>
        <End>4320</End>
        <Employees>7</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4320</Begin>
        <End>5760</End>
```

```
        <Employees>6</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5760</Begin>
        <End>7200</End>
        <Employees>5</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7200</Begin>
        <End>8640</End>
        <Employees>4</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8640</Begin>
        <End>10080</End>
        <Employees>4</Employees>
        <Exact>false</Exact>
      </Demand>
    </Demands>
  </StaffDemand>
  <ContractsDefinition>
    <TimeUnit>60</TimeUnit>
    <Contracts>
      <Contract>
        <Name>40H</Name>
        <DayDuration>480</DayDuration>
        <RestsPerWeek>2</RestsPerWeek>
        <ConsecutivesRests>true</ConsecutivesRests>
        <SplitShift>true</SplitShift>
        <MaximunSplitShifts>2</MaximunSplitShifts>
        <MinimunSubShift>120</MinimunSubShift>
        <MinimunRest>60</MinimunRest>
        <MaximunRest>180</MaximunRest>
        <RestWithoutCost>60</RestWithoutCost>
        <RestCostPerMinut>0.17</RestCostPerMinut>
        <Cost>80</Cost>
        <PossibleBegins>
          <PossibleBegin>
            <From>0</From>
            <To>10080</To>
          </PossibleBegin>
        </PossibleBegins>
        <WorkersAmount/>
      </Contract>
      <Contract>
        <Name>30H</Name>
        <DayDuration>360</DayDuration>
        <RestsPerWeek>2</RestsPerWeek>
        <ConsecutivesRests>true</ConsecutivesRests>
        <SplitShift>false</SplitShift>
        <MaximunSplitShifts>2</MaximunSplitShifts>
        <MinimunSubShift>120</MinimunSubShift>
        <MinimunRest>60</MinimunRest>
        <MaximunRest>180</MaximunRest>
        <RestWithoutCost>60</RestWithoutCost>
        <RestCostPerMinut>0.17</RestCostPerMinut>
        <Cost>60</Cost>
```

```
        <PossibleBegins>
          <PossibleBegin>
            <From>0</From>
            <To>10080</To>
          </PossibleBegin>
        </PossibleBegins>
        <WorkersAmount/>
      </Contract>
    </Contracts>
  </ContractsDefinition>
</TWFP_Problem>
```

Figura 3-4. Ejemplo de instancia de un TWFP

El objeto StaffDemand tiene el campo TimeUnit, expresado en minutos, que define la unidad temporal que va a definir la carga de trabajo, en este caso es una hora (60 minutos), lo que significa que se va a especificar la carga de trabajo para cada hora de la semana. En este caso, los intervalos se deben definir para cada hora. Por ejemplo, el intervalo [60, 120] es válido, pero el intervalo [60,90] no es correcto porque 90 no es múltiplo del TimeUnit (60 minutos en este caso). Los TimeUnit permitidos para la carga de trabajo son 15, 30 y 60 minutos; cualquier otro valor producidará un error en la llamada a la función solve() del servicio web wsTWFP. StaffDemand tiene una lista Demands de objetos Demand. Cada uno de estos objetos Demand define la carga de trabajo re querida en un intervalo de tiempo.

Un objeto Demand viene definido por cuatro campos:

- Begin: Comienzo del intervalo para el que se especifica la carga de trabajo.
- End: Fin del intervalo para el que se especifica la carga de trabajo.
- Employees: Número de trabajadores necesarios durante el intervalo para el que se define la carga de trabajo.
- Exact: Indica si la cantidad de trabajadores presentes en la planificación que se va a calcular durante ese intervalo debe ser exacta, o se puede superar.

La carga de trabajo del ejemplo de la Figura 3-4 se expresa mediante siete intervalos, uno para cada día. El lunes (desde el minuto 0 al 1440) se necesitan al menos 5 trabajadores, el martes (desde el minuto 1440 al 2880) se necesitan al menos 6 trabajadores, el miércoles (desde el minuto 2880 al 4320) se necesitan al menos 7 trabajadores, el jueves (desde el minuto 4320 al 5760) se necesitan al menos 6 trabajadores, el viernes (desde el minuto 5760 al 7200) se necesitan al menos 5 trabajadores, el sábado (desde el minuto 7200 al 8640) se necesitan al menos 4 trabajadores, y el domingo (desde el minuto 8640 al 10080) se necesitan al menos 4 trabajadores.

Como se deduce del ejemplo anterior si TimeUnit es una hora, no es necesario expresar la carga de trabajo para cada una de las horas de la semana, sino que se puede expresar por intervalos que aglutinen a varias horas; en el ejemplo anterior los intervalos son de 24 horas (un día).

La carga de trabajo es acumulativa a nivel de TimeUnit. Esto significa que si se especifica carga de trabajo para una misma TimeUnit desde varios intervalos que se solapan, la carga de trabajo total de la TimeUnit es la suma de las cargas de trabajo especificadas desde cada intervalo.

En el ejemplo de instancia del TWFP de la Figura 3-4, el objeto lista Contracts tiene dos contratos, definidos por el objeto Contract. Los campos de este objeto son los que se definieron en el apartado 2.2. El primer contrato corresponde a trabajadores de 40 horas semanales que descansan los dos días seguidos, no parten su jornada de trabajo, y pueden comenzar jornadas de trabajo en cualquier momento de la semana. El segundo contrato corresponde a trabajadores de 30 horas semanales que descansan los dos días seguidos, que no pueden partir su jornada de trabajo, y pueden comenzar jornadas de trabajo en cualquier momento de la semana. La estructura del objeto Contract cosrresponde a la definida en el apartado 2.2 (Tabla 2-1).

La Tabla 3-2 muestra un ejemplo de solución del TWFP que genera el proceso de resolución de

VETU. El objeto raíz que encapsula toda la información de la solución es TWFP_Solution, que se compone de tres objetos: Result, que indica se he resuelto correctamente el problema, ObjetiveValue, que muestra los valores objetivo de la solución, y TimeTable, que es el conjunto de jornadas que se han generado en la solución.

```
<?xml version="1.0" encoding="utf-8"?>
<TWFP_Solution>
  <Result>
    <Status>Ok</Status>
    <Message Code ="TWFP0">The TWFP problem has been succesfully solved</Message>
  </Result>
  <ObjectiveValue>
    <Workers>175</Workers>
    <Journeys>875</Journeys>
    <DemandedTime>313800</DemandedTime>
    <CoveredTime>420000</CoveredTime>
    <Cost>46650</Cost>
    <ContractInformation ContractName="40">
      <WorkersCount>85</WorkersCount>
      <Shifts>425</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>25500</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">73</WorkersDay>
        <WorkersDay Day="1">73</WorkersDay>
        <WorkersDay Day="2">85</WorkersDay>
        <WorkersDay Day="3">78</WorkersDay>
        <WorkersDay Day="4">72</WorkersDay>
        <WorkersDay Day="5">19</WorkersDay>
        <WorkersDay Day="6">25</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
    <ContractInformation ContractName="30">
      <WorkersCount>90</WorkersCount>
      <Shifts>450</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>21150</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">74</WorkersDay>
        <WorkersDay Day="1">56</WorkersDay>
        <WorkersDay Day="2">53</WorkersDay>
        <WorkersDay Day="3">51</WorkersDay>
        <WorkersDay Day="4">65</WorkersDay>
        <WorkersDay Day="5">73</WorkersDay>
        <WorkersDay Day="6">78</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
  </ObjectiveValue>
  <TimeTable>
    <ContractShifts ContractName="40">
      <WeekPatrons>
        <WeekPatron>
          <Amount>12</Amount>
          <RestDay>0</RestDay>
          <RestDay>1</RestDay>
        </WeekPatron>
        <WeekPatron>
          <Amount>7</Amount>
          <RestDay>3</RestDay>
```

```
          <RestDay>4</RestDay>
        </WeekPatron>
        <WeekPatron>
          <Amount>6</Amount>
          <RestDay>4</RestDay>
          <RestDay>5</RestDay>
        </WeekPatron>
        <WeekPatron>
          <Amount>60</Amount>
          <RestDay>5</RestDay>
          <RestDay>6</RestDay>
        </WeekPatron>
      </WeekPatrons>
      <Shifts>
        <Shift>
          <Workers>2</Workers>
          <Begin>300</Begin>
          <End>780</End>
        </Shift>
        …
        <Shift>
          <Workers>7</Workers>
          <Begin>9600</Begin>
          <End>10080</End>
        </Shift>
      </Shifts>
    </ContractShifts>
    <ContractShifts ContractName="30">
      <WeekPatrons>
        <WeekPatron>
          <Amount>16</Amount>
          <RestDay>0</RestDay>
          <RestDay>1</RestDay>
        </WeekPatron>
        <WeekPatron>
          <Amount>18</Amount>
          <RestDay>1</RestDay>
          <RestDay>2</RestDay>
        </WeekPatron>
        <WeekPatron>
          <Amount>19</Amount>
          <RestDay>2</RestDay>
          <RestDay>3</RestDay>
        </WeekPatron>
        <WeekPatron>
          <Amount>20</Amount>
          <RestDay>3</RestDay>
          <RestDay>4</RestDay>
        </WeekPatron>
        <WeekPatron>
          <Amount>5</Amount>
          <RestDay>4</RestDay>
          <RestDay>5</RestDay>
        </WeekPatron>
        <WeekPatron>
          <Amount>12</Amount>
          <RestDay>5</RestDay>
          <RestDay>6</RestDay>
        </WeekPatron>
      </WeekPatrons>
```

```
      <Shifts>
        <Shift>
          <Workers>3</Workers>
          <Begin>180</Begin>
          <End>540</End>
        </Shift>
        …
        <Shift>
          <Workers>23</Workers>
          <Begin>9720</Begin>
          <End>10080</End>
        </Shift>
      </Shifts>
    </ContractShifts>
  </TimeTable>
</TWFP_Solution>
```

Tabla 3-2. Ejemplo de solución del TWFP

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 TWFP0. 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 3-3 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 TimeTable están vacíos.

```
<?xml version="1.0" encoding="utf-8"?>
<TWFP_Solution>
  <Result>
    <Status>Failed</Status>
    <Message Code ="TWFP10">TimeUnit = 10 for StaffDemand is not valid</Message>
    <Message Code ="TWFP12">The demand for interval [60,120] can not be covered
                            by any contract</Message>
  </Result>
  <ObjectiveValue/>
  <TimeTable/>
</TWFP_Solution>
```

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

El objeto ObjectiveValue muestra los valores de las magnutides de la solución que ha encontrado el proceso de cálculo. Este objeto se compone de cinco campos y un objeto:

- Workers: Número de trabajadores que se necesitan para dar el servicio.
- Shifts: Número de jornadas que realizan los trabajadores.
- DemandedTime: Minutos que tiene la carga de trabajo definida.
- CoveredTime: Minutos de trabajo que cubren los trabajadores.
- Cost: Coste total de la plantilla que da el servicios.
- ContractInformation: Muestra información a nivel de contrato. Se compone de cuatro campos y un objeto:
    - WorkersCount: Número de trabajadores necesarios para este contrato.
    - Shifts: Número de jornadas que realizan los trabajadores de ese contrato.
    - SplitedShifts: Número de jornadas partidas que realizan los trabajadores de ese contrato.
    - Cost: Coste de las jornadas de trabajo calculadas para ese contrato.
    - WorkersPerDay: Indica el número de trabajadores de ese contrato que son

necesarios cada día de la semana.

El objeto ObjetiveValue del ejemplo de la Tabla 3-2 muestra que la solución de coste mínimo para la carga de trabajo dada en el problema, son necesarios 175 trabajadores, que realizan 875 jornadas (5 jornadas por trabajador), y cubren una carga de trabajo de 420.000 minutos (7.000 horas). Como la carga de trabajo demandaba 313.800 minutos (5.230 horas), se produce un exceso de trabajo de 106.200 minutos (1.770 horas). El coste total de la plantilla es de 46.650 UM. Son necesarios 85 trabajadores de 40 horas, que realizan 425 jornadas, todas continuas, con uns coste para el contrato de 25.500 UM. El reparto de trabajadores a lo largo de la semana es de 73 el lunes y el martes, 85 el miércoles, 78 el jueves, 72 el viernes, 19 el sábado y 25 el domingo. Son necesarios 90 trabajadores de 30 horas semanales, que realizan 450 jornadas, todas continuas, con un coste para el contrato de 21.150 UM. El reparto de trabajadores a lo largo de la semana es de 74 el lunes, 56 el martes, 53 el miércoles, 51 el jueves, 65 el viernes, 73 el sábado y 78 el domingo.

El objeto TimeTable de TWFP_Solution muestra los patrones de descanso semanal y las jornadas de trabajo que se han calculado para cada tipo de contrato. Los patrones de descanso semanal los muestra el objeto WeekPatrons y las jornadas las muestra el objeto Shifts. El objeto WeekPatrons se compone de objetos WeekPatron. Este objeto tiene dos campos, que definen la cantidad de trabajadores que realizan el patrón de descanso semanal. En el contrato de 30 horas semanales de la Tabla 3-2, hay 16 trabajadores que descansan lunes y martes, 18 que descansan martes y miércoles, 19 que descansan miércoles y jueves, 20 que descansan jueves y viernes, 5 que descansan viernes y sábado, y 12 que descansan sábado y domingo. El objeto Shifts es una lista de objetos Shift, que indican las jornadas calculadas para cada contrato. En la Tabla 3-2 solo se muestran dos objetos Shift por contrato, por cuestiones de espacio, pero son muchos más. En cualquiera de los ejemplos que se pueden encontrar en la web de VETU, se puede observar una solución con todos los objetos Shift. El objeto Shift tiene dos variantes, en función de que la jornada sea continua o partida. La Tabla 3-4 muestra una ejemplo de jornada continua en la que 7 trabajadores comienzan jornada en el minuto 600 (10:00 del lunes) y terminan en el minuto 1.080 (18:00 del lunes). La Tabla 3-5 muestra un ejemplo de jornada partida, en la que hay 15 trabajadores que comienzan la jornada de trabajo en el minuto 600 (10:00 del lunes), descansan desde el minuto 840 al 900 (desde las 14:00 a las 15:00 del lunes), y terminan la jornada en el minuto 1.140 (las 19:00 del lunes).

```
<Shift>
  <Workers>7</Workers>
  <Begin>600</Begin>
  <End>1080</End>
</Shift>
```

Tabla 3-4. Ejemplo de jornada contínua

```
<Shift>
  <Workers>15</Workers>
  <Begin>600</Begin>
  <RestBegin>840</RestBegin>
  <RestEnd>900</RestEnd>
  <End>1140</End>
</Shift>
```

Tabla 3-5. Ejemplo de jornada partida

3.3 Proceso de ejecución mediante 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 TWFP, el pseudocódigo de un programa que resuelve el TWFP 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.

```
TWFP_Problem oEntrada
Definir contratos en oEntrada
Definir carga de trabajo en oEntrada
Request oResultado = wsTWFP.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
                        TWFP_Solution oSalida = wsTWFP.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 TWFP 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 TWFP, mediante el sistema SaaS de VETU, escrito en C# (Microsoft .NET [1](1)) de Microsoft, bajo Visual Studio 2010[2].

Visual Studio crea todas las clases necesarias para acceder a servicios web, a partir de la

[1] Microsoft .NET es una marca registrada por Microsoft Corporation.
[2] Visual Studio .NET es una marca registrada por Microsoft Corporation.

definición WSDL del servicio. Para crear las clases de acceso al servicio web de resolución del TWFP 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/wsTWFP/wsTWFP?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/wsTWFP/wsTWFP?wsdl
8. Pulsar el botón 'Add Reference'

En este momento se han creado una serie de clases que permiten acceder al servicio web wsTWFP, 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:

- WSTWFPService: Es la clase de acceso al servicio web. Tiene las funciones solve y getSolution.
- WPF_TacticalPlanning Es la clase que encapsula la definición de la instancia de problema que se quiere resolver. Contiene a las clases staffDemand y contractsDefinition, que contienen la definición de la carga de trabajo y de los contratos, respectivamente. La clase staffDemand contiene la definición del campo TimeUnit y de los objetos demand. La clase contractsDefinition contiene la definición del campo TimeUnit de los contratos e instancias de la clase contract, que define los parámetros de los contratos.
- wfSolution: Es la clase que encapsula la solución del problema que se ha resuelto. Contiene a las clases result, objectiveValue y contractShifts, que tiene la información de la resolución del problema, de lo valores encontrados y las jornadas definidas, 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 WSExecutionProgressService, 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 WSExecutionProgressService, que es el acceso al servicio web, y tiene la función checkStatus.

El código fuente necesario para la resolución del TWFP mediante los servicios web de Vetu se muestra en la Figura 3-6. Los números de línea de la columna de la izquierda se muestran solo para facilitar la siguiente explicación. Desde las líneas 1 a la 91 se definen los datos del problema a resolver. En las líneas 2 y 3 se crea la instancia de la clase TacticalPlanning, que encapsula toda la información del problema.

En la línea 6 se crea la clase StaffDemand, en la línea 7 se define el campo TimeUnit de la demanda, en la línea 9 se crea el vector de demandas, y desde la línea 10 a la 17, se define una carga de trabajo de cinco trabajadores para cada hora de la semana.

[3] Esta secuencia de acciones podría cambiar según la versión de Visual Studio.

En la línea 20 se crea la definición de contratos, en la 21 se define el campo TimeUnit de los contratos, y en la 24 se crea el array de contratos (dos contratos). En la línea 27 se crea el primer contrato, y se añade al array de contratos en la posición 0 (línea 28). Desde la línea 31 a la 61 se definen los datos del contrato. En la línea 64 se crea el segundo contrato, y se añade al array de contratos en la posición 1 (línea 65). Desde la línea 68 a la 91 se definen los datos del segundo contrato.

Una vez definido el problema, se debe enviar al servicio web: en la línea 95 se crea el acceso al servicio web, y en las líneas 98 y 99 se envía el problema al servicio web. La llamada a la función solve devuelve un objeto Result, que indica si la llamada al servicio web ha tenido algún problema. En las líneas 101 a 104 se comprueba si ha existido algún error. Si no ha habido errores, se entra en la parte de búsqueda de la solución. En las líneas 108 y 109 se crea el acceso al servicio web que informa del estado de resolución del problema. Desde la línea 113 a la 126 se define un bucle con un timer que espera a que se resuelva el problema. Desde las líneas 129 a la 151, se comprueba el estado final de resolución. Si el estado es PROCESSED, se accede a la solución del problema (líneas 132 a 134), que viene encapsulada en el objeto wfSolution.

```
1  // Crea la instancia de problema
2  TWFPReference.WFP_TacticalPlanning oInstancia = new
3                       TWFPReference.WFP_TacticalPlanning();
4
5  // Crea la demanda
6  oInstancia.StaffDemand = new TWFPReference.staffDemand();
7  oInstancia.StaffDemand.TimeUnit = 60;
8
9  oInstancia.StaffDemand.Demands = new TWFPReference.demand[24 * 7];
10 for (int h = 0; h < 168; h++)
11 {
12     oInstancia.StaffDemand.Demands[h] = new TWFPReference.demand();
13     oInstancia.StaffDemand.Demands[h].Begin = h * 60;
14     oInstancia.StaffDemand.Demands[h].End = (h + 1) * 60;
15     oInstancia.StaffDemand.Demands[h].Employees = 5;
16     oInstancia.StaffDemand.Demands[h].Exact = false;
17 }
18
19 // Crea la definición de contratos
20 oInstancia.ContractsDefinition = new TWFPReference.contractsDefinition();
21 oInstancia.ContractsDefinition.TimeUnit = 60;
22
23 // Crea la lista de contratos, para dos contratos
24 oInstancia.ContractsDefinition.Contracts = new TWFPReference.contract[2];
25
26 // Crea el primer contrato y lo guarda en la lista de contratos
27 TWFPReference.contract oContract = new TWFPReference.contract();
28 oInstancia.ContractsDefinition.Contracts[0] = oContract;
29
30 // Define los paramétros del contrato
31 oContract.Name                  = "40H";
32 oContract.DayDuration           = 480;
33 oContract.RestsPerWeek          = 2;
34 oContract.ConsecutivesRests     = true;
35 oContract.SplitShift            = true;
36 oContract.MaximunSplitShifts    = 3;
37 oContract.MinimunSubShift       = 120;
38 oContract.MinimunRest           = 60 ;
39 oContract.MaximunRest           = 120;
40 oContract.RestWithoutCost       = 60;
```

```
oContract.RestCostPerMinut      = 0.45;
oContract.Cost                  = 90;

// Crea la lista de posibles comienzos de las jornadas de trabajo, para un
// solo posible comienzo
oContract.PossibleBegins = new TWFPReference.possibleBegin[1];

// Crea el posible comienzo, y lo guarda en la lista
TWFPReference.possibleBegin oPosBegin = new TWFPReference.possibleBegin();
oContract.PossibleBegins[0] = oPosBegin;

// Define los valores del posible comienzo
oPosBegin.From  = 0;
oPosBegin.To    = 10080;

// Crea una condición sobre el número de trabajadores del contrato 40H
oContract.WorkersAmount = new TWFPReference.workersAmount();

// Fija los parámetros de la condición
oContract.WorkersAmount.Criterion = "GreaterOrEqual";
oContract.WorkersAmount.Amount    = 20;

// Crea el segundo contrato y lo guarda en la lista de contratos
oContract = new TWFPReference.contract();
oInstancia.ContractsDefinition.Contracts[1] = oContract;

// Define los paramétros del contrato
oContract.Name                  = "30H";
oContract.DayDuration           = 360;
oContract.RestsPerWeek          = 2;
oContract.ConsecutivesRests     = true;
oContract.SplitShift            = true;
oContract.MaximunSplitShifts    = 3;
oContract.MinimunSubShift       = 120;
oContract.MinimunRest           = 60;
oContract.MaximunRest           = 120;
oContract.RestWithoutCost       = 60;
oContract.RestCostPerMinut      = 0.45;
oContract.Cost                  = 68;

// Crea la lista de posibles comienzos de las jornadas de trabajo, para un
// solo posible comienzo
oContract.PossibleBegins = new TWFPReference.possibleBegin[1];

// Crea el posible comienzo, y lo guarda en la lista
oPosBegin = new TWFPReference.possibleBegin();
oContract.PossibleBegins[0] = oPosBegin;

// Define los valores del posible comienzo
oPosBegin.From = 0;
oPosBegin.To = 10080;


// Crea el acceso al servicio web
TWFPReference.WSTWFPService oCliente = new TWFPReference.WSTWFPService();

// Lanza la petición de resolución
TWFPReference.Result oResultado = oCliente.solve("usuario_Vetu",
"usuario_Vetu", "155.210.85.146", oInstancia);
```

```
if (oResultado.Status == "Failed")
{
    MessageBox.Show(oResultado.Message[0].Value, "Error en la llamada: ");
}
else if (oResultado.Status == "Ok")
{
    // Espera a que se ejecute el problema
    ExecutionProgressReference.WSExecutionProgressService oProgessService =
                new ExecutionProgressReference.WSExecutionProgressService();

    string strEstado = "";
    bool bTerminado = false;
    while (!bTerminado)
    {
        strEstado = oProgessServiceClient.checkStatus("usuario_Vetu",
"usuario_Vetu", "155.210.85.146", oResultado.Ticket);

        if (strEstado == "UNKNOWN" || strEstado == "ERROR" || strEstado ==
"PROCESSED")
            bTerminado = true;
        else if (strEstado == "ALLOCATED" || strEstado == "IN_PROCESS")
            System.Threading.Thread.Sleep(5000);
        else // Estado desconocido
             bTerminado = true;
    }
}

// Lee el resultado del problema
if (strEstado == "PROCESSED")
{
    // Lee la solución
    TWFPReference.wfSolution oSolucion =
oCliente.getSolution("usuario_Vetu", "usuario_Vetu", "155.210.85.146",
oResultado.Ticket);

    // Tratar la solución.
}
else if (strEstado == "UNKNOWN")
{
    MessageBox.Show( "El ticket de acceso a la solución no es válido.
Contacte con al administrador" );
}
else if (strEstado == "ERROR")
{
    MessageBox.Show( "Se han producido errores en la planificación" );
}
else
{
    MessageBox.Show( "Se ha producido un error desconocido. Contacte con al
administrador" );
}
```

Figura 3-6. Código fuente C# para la ejecución del TWFP de Vetu

3.5 Ejemplo de uso desde NetBeans

En este apartado, se muestra un programa de ejemplo para resolver una instancia de TWFP, 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 TWFP 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".

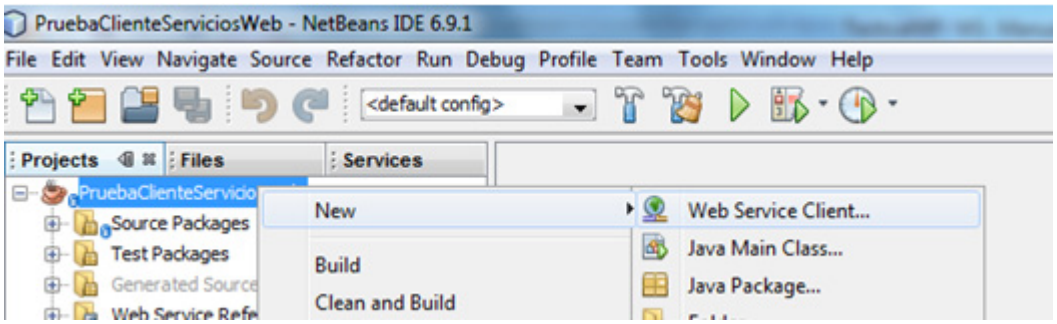


Figura 3-7. Paso 1 - Creación de cliente del servicio web NebBeans

Aparecerá una ventana emergente.

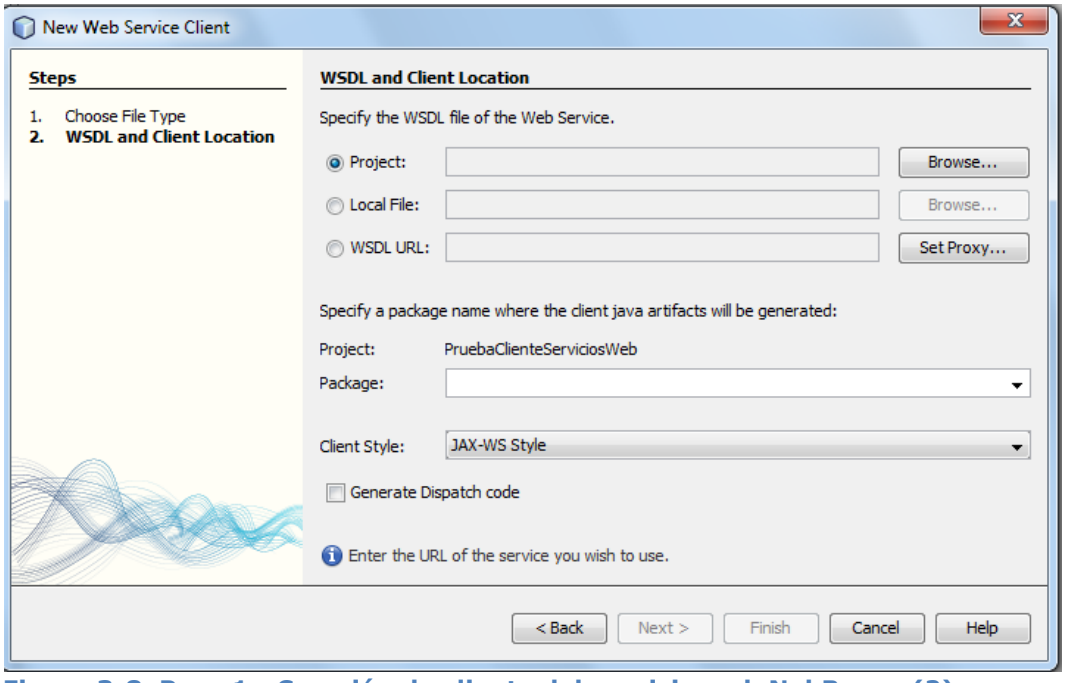


Figura 3-8. Paso 1 - Creación de cliente del servicio web NebBeans (2)

2. Marcar la opción "WSDL URL", escribir http://www.vetu.es/wsTWFP/wsTWFP?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 wsTWFP, 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:

- WsTWFP: Es la clase de acceso al servicio web. Tiene las funciones solve y getSolution.

- WPFTacticalPlanning Es la clase que encapsula la definición de la instancia de problema que se quiere resolver. Contiene a las clases StaffDemand y ContractsDefinition, que contienen la definición de la carga de trabajo y de los contratos, respectivamente. La clase StaffDemand contiene la definición del campo TimeUnit y de los objetos Demand. La clase contractsDefinition contiene la definición del campo TimeUnit de los contratos e instancias de la clase Contract, que define los parámetros de los contratos.
- WfSolution: Es la clase que encapsula la solución del problema que se ha resuelto. Contiene a las clases Result, ObjectiveValue y ContractShifts, que tiene la información de la resolución del problema, de lo valores encontrados y las jornadas definidas, 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 punto 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 del TWFP mediante los servicios web de Vetu se muestra en la Figura 3-9. Los números de línea de la columna de la izquierda se muestran solo para facilitar la siguiente explicación. Desde las líneas 1 a la 98 se definen los datos del problema a resolver. En las líneas 2 y 3 se crea la instancia de la clase TacticalPlanning, que encapsula toda la información del problema.

En la línea 6 se crea la clase StaffDemand, en la línea 7 se define el campo TimeUnit de la demanda, en la línea 9 se crea el vector de demandas, y desde la línea 10 a la 18, se define una carga de trabajo de cinco trabajadores para cada hora de la semana.

En la línea 22 se crea la definición de contratos, en la 23 se define el campo TimeUnit de los contratos, y en la 27 se crea el objeto Contracts que contendrá los contratos. En la línea 31 se crea el primer contrato, y se añade al array de contratos (línea 32). Desde la línea 35 a la 68 se definen los datos del contrato. En la línea 77 se crea el segundo contrato, y se añade al array de contratos en la siguiente posición (línea 72). Desde la línea 75 a la 99 se definen los datos del segundo contrato.

Una vez definido el problema, se debe enviar al servicio web: en la línea 102 se crea el acceso al servicio web, y en la línea 106 se envía el problema al servicio web. La llamada a la función solve devuelve un objeto Request, que indica si la llamada al servicio web ha tenido algún problema. En las líneas 109 a 111 se comprueba si ha existido algún error. Si no ha habido errores, se entra en la parte de búsqueda de la solución. En las líneas 114 y 117 se crea el acceso al servicio web que informa del estado de resolución del problema. Desde la línea 121 a la 135 se define un bucle con un timer que espera a que se resuelva el problema. Desde las líneas 138 a la 163, se comprueba el estado final de resolución. Si el estado es PROCESSED, se accede a la solución del problema (líneas 140 y 141), que viene encapsulada en el objeto wfSolution.

```
1  // Crea la instancia de problema
2  serviciowebwpf.WFPTacticalPlanning wfpTacticalPlanning = new
3  serviciowebwpf.WFPTacticalPlanning();
4
5  // Crea la demanda
6  StaffDemand staffDemand = new StaffDemand();
7  staffDemand.setTimeUnit(60);
8  wfpTacticalPlanning.setStaffDemand(staffDemand);
```

```
9
10  Demands demands = new Demands();
11  staffDemand.setDemands(demands);
12  for (int h = 0; h < 168; h++) { //Se crean 24 * 7 objetos de tipo Demand
13      Demand demand = new Demand();
14      demand.setBegin(h * 60);
15      demand.setEnd((h + 1) * 60);
16      demand.setEmployees(5);
17      demand.setExact(false);
18      demands.getDemand().add(demand);
19  }
20
21  // Crea la definición de contratos
22  ContractsDefinition contractsDefinition = new ContractsDefinition();
23  contractsDefinition.setTimeUnit(60);
24  wfpTacticalPlanning.setContractsDefinition(contractsDefinition);
25
26  // Crea la lista de contratos, para dos contratos
27  Contracts contracts = new Contracts();
28  contractsDefinition.setContracts(contracts);
29
30  // Crea el primer contrato y lo guarda en la lista de contratos
31  Contract contract = new Contract();
32  contracts.getContract().add(contract);
33
34  // Define los paramétros del contrato
35  contract.setName("40H");
36  contract.setDayDuration(480);
37  contract.setRestsPerWeek(2);
38  contract.setConsecutivesRests(true);
39  contract.setSplitShift(true);
40  contract.setMaximunSplitShifts(3);
41  contract.setMinimunSubShift(120);
42  contract.setMinimunRest(60);
43  contract.setMaximunRest(120);
44  contract.setRestWithoutCost(60);
45  contract.setRestCostPerMinut(0.45);
46  contract.setCost(90);
47
48  // Crea la lista de posibles comienzos de las jornadas de trabajo, para un
49  // solo posible comienzo
50  PossibleBegins possibleBegins = new PossibleBegins();
51
52  // Crea el posible comienzo, y lo guarda en la lista
53  PossibleBegin possibleBegin = new PossibleBegin();
54  possibleBegins.getPossibleBegin().add(possibleBegin);
55  contract.setPossibleBegins(possibleBegins);
56
57  // Define los valores del posible comienzo
58  possibleBegin.setFrom(0);
59  possibleBegin.setTo(10080);
60
61  // Crea una condición sobre el número de trabajadores del contrato 40H
62  WorkersAmount workersAmount = new WorkersAmount();
63  contract.setWorkersAmount(workersAmount);
64
65  // Fija los parámetros de la condición
66  workersAmount.setCriterion("GreaterOrEqual");
67  workersAmount.setAmount(20);
68
```

```
// Crea el segundo contrato y lo guarda en la lista de contratos
contract = new Contract();
contracts.getContract().add(contract);

// Define los paramétros del contrato
contract.setName("30H");
contract.setDayDuration(360);
contract.setRestsPerWeek(2);
contract.setConsecutivesRests(true);
contract.setSplitShift(true);
contract.setMaximunSplitShifts(3);
contract.setMinimunSubShift(120);
contract.setMinimunRest(60);
contract.setMaximunRest(120);
contract.setRestWithoutCost(60);
contract.setRestCostPerMinut(0.45);
contract.setCost(68);

// Crea la lista de posibles comienzos de las jornadas de trabajo, para
// un solo posible comienzo
possibleBegins = new PossibleBegins();

// Crea el posible comienzo, y lo guarda en la lista
possibleBegin = new PossibleBegin();
possibleBegins.getPossibleBegin().add(possibleBegin);
contract.setPossibleBegins(possibleBegins);

// Define los valores del posible comienzo
possibleBegin.setFrom(0);
possibleBegin.setTo(10080);

// Crea el acceso al servicio web
WSTWFPService service = new WSTWFPService();
serviciowebwpf.WSTWFP port = service.getWSTWFPPort();

// Lanza la petición de resolución
Request oResultado = port.solve("usuario_Vetu", "usuario_Vetu",
"155.210.85.146", wfpTacticalPlanning);

if (oResultado.getStatus().equals("Failed")) {
    System.out.println("Error en la llamada: " +
oResultado.getMessage().get(0).getValue());
} else if (oResultado.getStatus().equals("Ok")) {
    // Espera a que se ejecute el problema
    WSExecutionProgressService serviceExecutionProgress = new
WSExecutionProgressService();
    WSExecutionProgress portProgress =
serviceExecutionProgress.getWSExecutionProgressPort();

    String strEstado = "";
    boolean bTerminado = false;
    while (!bTerminado) {
        strEstado = portProgress.checkStatus("usuario_Vetu", "usuario_Vetu",
"155.210.85.218", 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
        WfSolution wFSolution = port.getSolution("usuario_Vetu",
"usuario_Vetu", "155.210.85.218", oResultado.getTicket());
        // Tratar la solución.
        // Mostrar resultado
        JAXBContext jc = JAXBContext.newInstance(String.class,
serviciowebwpf.WfSolution.class);
        JAXBIntrospector introspector = jc.createJAXBIntrospector();
        Marshaller marshaller = jc.createMarshaller();
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
        if (null == introspector.getElementName(wFSolution)) {
            JAXBElement jaxbElement = new JAXBElement(new QName("ROOT"),
Object.class, wFSolution);
            marshaller.marshal(jaxbElement, System.out);
        } else {
            marshaller.marshal(wFSolution, System.out);
        }
    } else if (strEstado.equals("UNKNOWN")) {
        System.out.println("Ticket de acceso a la solución no válido.");
    } else if (strEstado.equals("ERROR")) {
        System.out.println("Se han producido errores en la planificación");
    } else {
        System.out.println("Se ha producido un error desconocido. ");
    }
}
```

Figura 3-9. Código fuente Java en NetBeans para la ejecución del TWFP de Vetu

3.6 Ejemplo de uso desde Eclipse

En este apartado, se muestra un programa de ejemplo para resolver una instancia de TWFP, 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 TWFP 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".

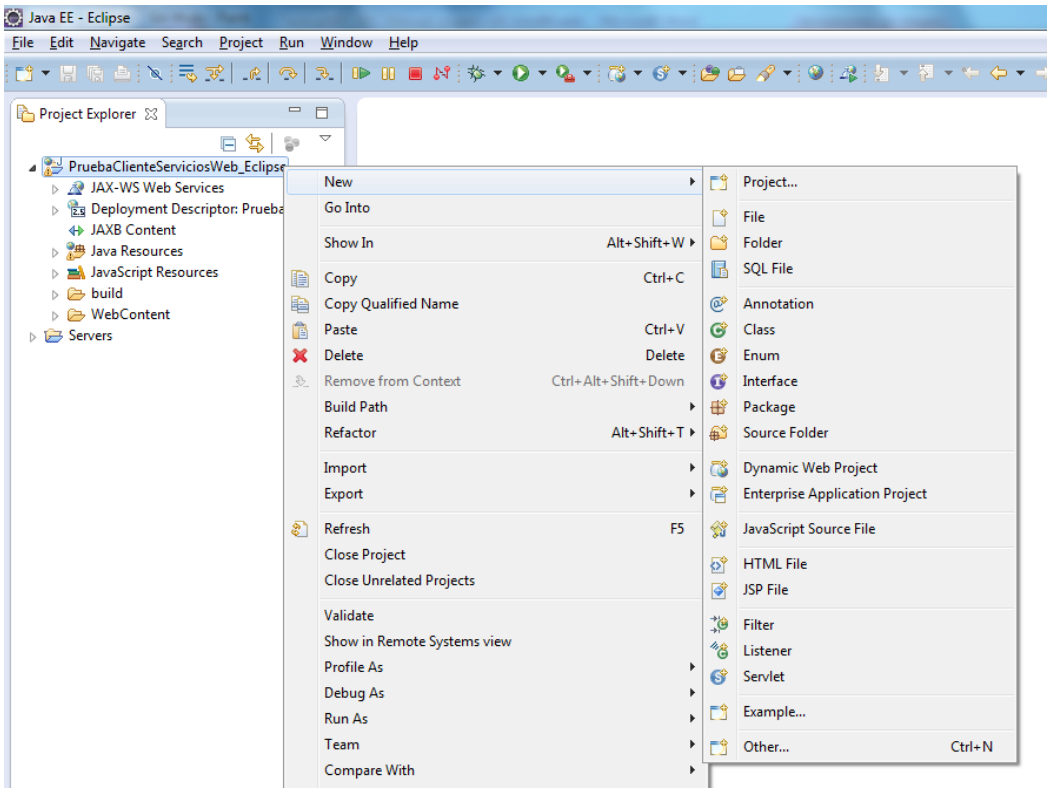


Figura 3-10. Paso 1 - Creación de cliente del servicio web Eclipse

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

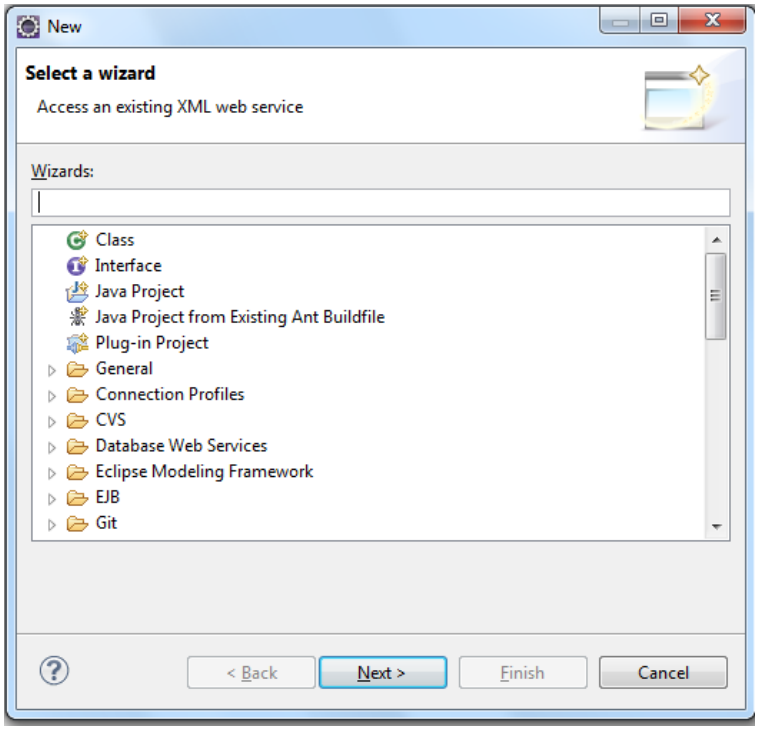


Figura 3-11. Paso 1 - Creación de cliente del servicio web Eclipse (2)

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".

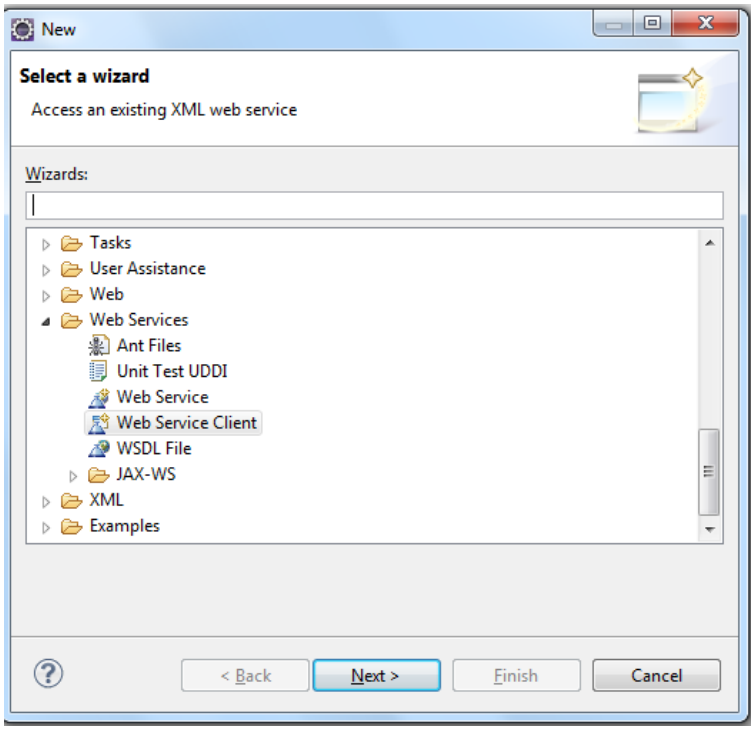


Figura 3-12. Paso 2 - Creación de cliente del servicio web Eclipse

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

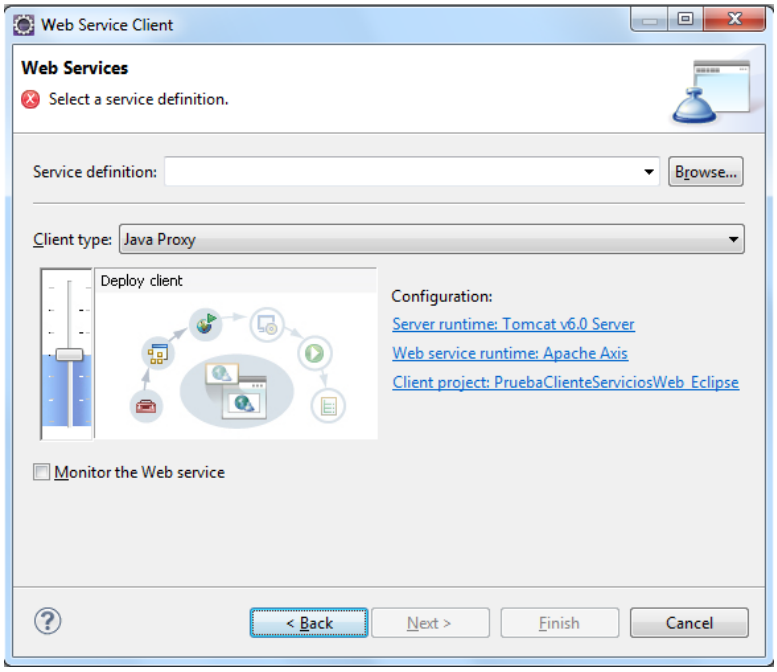


Figura 3-13. Paso 2 - Creación de cliente del servicio web Eclipse (2)

3. En la caja de texto "Service definition" se escribirá lo siguiente: http://www.vetu.es/wsTWFP/wsTWFP?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 wsTWFP, 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:

- WSTWFPService: Es la clase de acceso al servicio web. Tiene las funciones solve y getSolution.
- WPFTacticalPlanning Es la clase que encapsula la definición de la instancia de problema que se quiere resolver. Contiene a las clases StaffDemand y ContractsDefinition, que contienen la definición de la carga de trabajo y de los contratos, respectivamente. La clase StaffDemand contiene la definición del campo TimeUnit y de los objetos Demand. La clase contractsDefinition contiene la definición del campo TimeUnit de los contratos e instancias de la clase Contract, que define los parámetros de los contratos.
- WfSolution: Es la clase que encapsula la solución del problema que se ha resuelto. Contiene a las clases Result, ObjectiveValue y ContractShifts, que tiene la información de la resolución del problema, de lo valores encontrados y las jornadas definidas, 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 punto 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 del TWFP mediante los servicios web de Vetu se muestra en la Figura 3-14. Los números de línea de la columna de la izquierda se muestran solo para facilitar la siguiente explicación. Desde las líneas 1 a la 98 se definen los datos del problema a resolver. En las líneas 2 y 3 se crea la instancia de la clase TacticalPlanning, que encapsula toda la información del problema.

En la línea 6 se crea la clase StaffDemand, en la línea 7 se define el campo TimeUnit de la demanda, en la línea 9 se crea el vector de demandas, y desde la línea 10 a la 18, se define una carga de trabajo de cinco trabajadores para cada hora de la semana.

En la línea 22 se crea la definición de contratos, en la 23 se define el campo TimeUnit de los contratos, y en la 28 se crea el array de contratos (dos contratos). En la línea 31 se crea el primer contrato, y se añade al array de contratos en la posición 0 (línea 32). Desde la línea 35 a la 67 se definen los datos del contrato. En la línea 70 se crea el segundo contrato, y se añade al array de contratos en la posición 1 (línea 71). Desde la línea 74 a la 98 se definen los datos del segundo contrato.

Una vez definido el problema, se debe enviar al servicio web: en la línea 101 se crea el acceso al servicio web, y en la línea 105 se envía el problema al servicio web. La llamada a la función solve devuelve un objeto Request, que indica si la llamada al servicio web ha tenido algún problema. En las líneas 108 a 110 se comprueba si ha existido algún error. Si no ha habido errores, se entra en la parte de búsqueda de la solución. En las líneas 113 y 115 se crea el acceso al servicio web que informa del estado de resolución del problema. Desde la línea 120 a la 134 se define un bucle con un timer que espera a que se resuelva el problema. Desde las

líneas 137 a la 162, se comprueba el estado final de resolución. Si el estado es PROCESSED, se accede a la solución del problema (líneas 139 y 140), que viene encapsulada en el objeto wfSolution.

```
1  // Crea la instancia de problema
2  servicioWebWPF.WFP_TacticalPlanning wfpTacticalPlanning = new
3  servicioWebWPF.WFP_TacticalPlanning();
4
5  // Crea la demanda
6  StaffDemand staffDemand = new StaffDemand();
7  staffDemand.setTimeUnit(60);
8  wfpTacticalPlanning.setStaffDemand(staffDemand);
9
10 Demand[] demands = new Demand[24 * 7];
11 staffDemand.setDemands(demands);
12 for (int h = 0; h < 168; h++) {
13     Demand demand = new Demand();
14     demand.setBegin(h * 60);
15     demand.setEnd((h + 1) * 60);
16     demand.setEmployees(5);
17     demand.setExact(false);
18     demands[h] = demand;
19 }
20
21 // Crea la definición de contratos
22 ContractsDefinition contractsDefinition = new ContractsDefinition();
23 contractsDefinition.setTimeUnit(60);
24 wfpTacticalPlanning.setContractsDefinition(contractsDefinition);
25
26 // Crea la lista de contratos, para dos contratos
27 Contract[] contracts = new Contract[2];
28 contractsDefinition.setContracts(contracts);
29
30 // Crea el primer contrato y lo guarda en la lista de contratos
31 Contract contract = new Contract();
32 contracts[0] = contract;
33
34 // Define los paramétros del contrato
35 contract.setName("40H");
36 contract.setDayDuration(480);
37 contract.setRestsPerWeek(2);
38 contract.setConsecutivesRests(true);
39 contract.setSplitShift(true);
40 contract.setMaximunSplitShifts(3);
41 contract.setMinimunSubShift(120);
42 contract.setMinimunRest(60);
43 contract.setMaximunRest(120);
44 contract.setRestWithoutCost(60);
45 contract.setRestCostPerMinut(0.45);
46 contract.setCost(90);
47
48 // Crea la lista de posibles comienzos de las jornadas de trabajo, para
49 // un solo posible comienzo
50 PossibleBegin[] possibleBegins = new PossibleBegin[1];
51
52 // Crea el posible comienzo, y lo guarda en la lista
53 PossibleBegin possibleBegin = new PossibleBegin();
54 possibleBegins[0] = possibleBegin;
55 contract.setPossibleBegins(possibleBegins);
56
57 // Define los valores del posible comienzo
```

```
possibleBegin.setFrom(0);
possibleBegin.setTo(10080);

// Crea una condición sobre el número de trabajadores del contrato 40H
WorkersAmount workersAmount = new WorkersAmount();
contract.setWorkersAmount(workersAmount);

// Fija los parámetros de la condición
workersAmount.setCriterion("GreaterOrEqual");
workersAmount.setAmount(20);

// Crea el segundo contrato y lo guarda en la lista de contratos
contract = new Contract();
contracts[1] = contract;

// Define los paramétros del contrato
contract.setName("30H");
contract.setDayDuration(360);
contract.setRestsPerWeek(2);
contract.setConsecutivesRests(true);
contract.setSplitShift(true);
contract.setMaximunSplitShifts(3);
contract.setMinimunSubShift(120);
contract.setMinimunRest(60);
contract.setMaximunRest(120);
contract.setRestWithoutCost(60);
contract.setRestCostPerMinut(0.45);
contract.setCost(68);

// Crea la lista de posibles comienzos de las jornadas de trabajo, para
// un solo posible comienzo
possibleBegins = new PossibleBegin[1];

// Crea el posible comienzo, y lo guarda en la lista
possibleBegin = new PossibleBegin();
possibleBegins[0] = possibleBegin;
contract.setPossibleBegins(possibleBegins);

// Define los valores del posible comienzo
possibleBegin.setFrom(0);
possibleBegin.setTo(10080);

// Crea el acceso al servicio web
WSTWFPService service = new WSTWFPServiceLocator();
servicioWebWPF.WSTWFP port = service.getWSTWFPPort();

// Lanza la petición de resolución
Request oResultado = port.solve("usuario_Vetu", "usuario_Vetu",
"155.210.85.146", wfpTacticalPlanning);

if (oResultado.getStatus().equals("Failed")) {
    System.out.println("Error en la llamada: " +
oResultado.getMessage(0).get_value());
} else if (oResultado.getStatus().equals("Ok")) {
    // Espera a que se ejecute el problema
    WSExecutionProgressService serviceExecutionProgress = new
WSExecutionProgressServiceLocator();
    WSExecutionProgress portProgress =
serviceExecutionProgress.getWSExecutionProgressPort();

```

```
118     String strEstado = "";
119     boolean bTerminado = false;
120     while (!bTerminado) {
121         strEstado = portProgress.checkStatus("usuario_Vetu", "usuario_Vetu",
122 "155.210.85.146", oResultado.getTicket());
123 
124         if (strEstado.equals("UNKNOWN") || strEstado.equals("ERROR") ||
125 strEstado.equals("PROCESSED")) {
126             bTerminado = true;
127         } else if (strEstado.equals("ALLOCATED") ||
128 strEstado.equals("IN_PROCESS")) {
129             Thread.sleep(5000);
130         } else // Estado desconocido
131         {
132             bTerminado = true;
133         }
134     }
135 
136     // Lee el resultado del problema
137     if (strEstado.equals("PROCESSED")) {
138         // Lee la solución
139         WfSolution wFSolution = port.getSolution("usuario_Vetu",
140 "usuario_Vetu", "155.210.85.146", oResultado.getTicket());
141         // Tratar la solución.
142         //Mostrar resultado
143         JAXBContext jc = JAXBContext.newInstance(String.class,
144 servicioWebWPF.WfSolution.class);
145         JAXBIntrospector introspector = jc.createJAXBIntrospector();
146         Marshaller marshaller = jc.createMarshaller();
147         marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
148 Boolean.TRUE);
149         if (null == introspector.getElementName(wFSolution)) {
150             JAXBElement jaxbElement = new JAXBElement(new QName("ROOT"),
151 Object.class, wFSolution);
152             marshaller.marshal(jaxbElement, System.out);
153         } else {
154             marshaller.marshal(wFSolution, System.out);
155         }
156     } else if (strEstado.equals("UNKNOWN")) {
157         System.out.println("Ticket de acceso a la solución no válido");
158     } else if (strEstado.equals("ERROR")) {
159         System.out.println("Se han producido errores en la planificación");
160     } else {
161         System.out.println("Se ha producido un error desconocido");
162     }
163 }
164 
165 
```

Figura 3-14. Código fuente Java en Eclipse para la ejecución del TWFP de Vetu

4 Casos de uso

En este capítulo se muestran diversos escenarios de uso del sistema SaaS de VETU que resuelve el TWFP, de forma que el usuario pueda aprender su uso, además de mostrar la utilidad del sistema para dimensionar plantillas de trabajo en diversas necesidades que requiera el cliente del sistema.

Todos los ejemplos se van resolver para una misma carga de trabajo, y se irán realizando comparativas entre ellos. En el anexo se muestra esta carga de trabajo, expresada en franjas temporales de una hora, para toda la semana. Esta carga de trabajo aparece en los ficheros XML de ejemplo disponibles en la web (véase apartado 1.6, donde también se puede ver una representación gráfica de la carga de trabajo).

4.1 Plantilla con un tipo de contrato

En ese caso de uso se resuelve el TWFP con un solo tipo de contrato. Se considera un contrato de trabajadores de 40 horas semanales, con 5 días de trabajo semanales, sin jornadas partidas. Se van a considerar dos casos: con días de descanso seguidos o separados.

Descansos seguidos: la definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 40H ConsecutiveRests' de la página de ejemplos del TWFP de VETU. El resultado del problema puede verse en el fichero ejemplo 'Out TWFP 40H ConsecutiveRests'. La Figura 4-1 muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. Para cubrir la carga de trabajo dada, se necesitan 175 trabajadores de 40H, con un coste total de 52.500 UM. Estos trabajadores ofrecen una carga de trabajo de 420.000 minutos, lo que representa un exceso de 420.000 – 313.800 = 106.200 minutos. Este exceso de carga de trabajo se debe a que la demanda tiene grandes variaciones en horas consecutivas, y no se puede adaptar más con jornadas de 8 horas.

```
<ObjectiveValue>
  <Workers>175</Workers>
  <Journeys>875</Journeys>
  <DemandedTime>313800</DemandedTime>
  <CoveredTime>420000</CoveredTime>
  <Cost>52500</Cost>
  <ContractInformation ContractName="40">
    <WorkersCount>175</WorkersCount>
    <Shifts>875</Shifts>
    <SplitedShifts>0</SplitedShifts>
    <Cost>52500</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">134</WorkersDay>
      <WorkersDay Day="1">129</WorkersDay>
      <WorkersDay Day="2">126</WorkersDay>
      <WorkersDay Day="3">129</WorkersDay>
      <WorkersDay Day="4">138</WorkersDay>
      <WorkersDay Day="5">92</WorkersDay>
      <WorkersDay Day="6">127</WorkersDay>
    </WorkersPerDay>
  </ContractInformation>
</ObjectiveValue>
```

Figura 4-1. Sección ObjectiveValue del ejemplo TWPF 40H ConsecutiveRests

Descansos separados: la definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 40H NoConsecutiveRests' de la página de ejemplos del TWFP de VETU. El resultado del problema puede verse en el fichero ejemplo 'Out TWFP 40H NoConsecutiveRests'. La Figura 4-2 muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. Para cubrir la carga de trabajo dada, se necesitan 165 trabajadores de 40H, con un coste total de 49.500 UM. Estos trabajadores ofrecen una carga de trabajo de 396.000 minutos, lo que representa 396.000 – 313.800 = 82.200 minutos.

```
<ObjectiveValue>
  <Workers>165</Workers>
  <Journeys>825</Journeys>
  <DemandedTime>313800</DemandedTime>
  <CoveredTime>396000</CoveredTime>
  <Cost>49500</Cost>
  <ContractInformation ContractName="40">
    <WorkersCount>165</WorkersCount>
    <Shifts>825</Shifts>
    <SplitedShifts>0</SplitedShifts>
    <Cost>49500</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">127</WorkersDay>
      <WorkersDay Day="1">129</WorkersDay>
      <WorkersDay Day="2">126</WorkersDay>
      <WorkersDay Day="3">129</WorkersDay>
      <WorkersDay Day="4">120</WorkersDay>
      <WorkersDay Day="5">94</WorkersDay>
      <WorkersDay Day="6">100</WorkersDay>
    </WorkersPerDay>
  </ContractInformation>
</ObjectiveValue>
```

Figura 4-2. Sección ObjectiveValue del ejemplo TWPF 40H NoConsecutiveRests

Si comparamos ambos resultados, la asignación de días seguidos de trabajo supone un aumento de coste de 3.300 UM sobre la opción de descansos seguidos y lleva a una plantilla con 10 empleados más.

4.2 Plantilla con dos tipos de contrato

En ese caso de uso se resuelve el TWFP con dos tipos de contrato. Se considera un contrato de trabajadores de 40 horas semanales, y otro contrato de 30 horas semanales, ambos con cinco días de trabajo semanales, sin jornadas partidas, y con días de descanso seguidos. La definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 40H-30H' de la página de ejemplos del TWFP de VETU. El resultado del problema puede verse en el fichero ejemplo 'Out TWFP 40H-30H'. La Figura 4-3 muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. Para cubrir la carga de trabajo dada, se necesitan 85 trabajadores de 40H, con un coste de 25.500 UM, y 90 trabajadores de 30H, con un coste de 21.600 UM, y un coste total de la plantilla de 47.100 UM. Estos trabajadores ofrecen una carga de trabajo de 366.000 minutos, lo que representa un exceso de 366.000 – 313.800 = 52.200 minutos.

```
<ObjectiveValue>
  <Workers>175</Workers>
  <Journeys>875</Journeys>
  <DemandedTime>313800</DemandedTime>
```

```
    <CoveredTime>366000</CoveredTime>
    <Cost>47100</Cost>
    <ContractInformation ContractName="40">
      <WorkersCount>85</WorkersCount>
      <Shifts>425</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>25500</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">72</WorkersDay>
        <WorkersDay Day="1">72</WorkersDay>
        <WorkersDay Day="2">85</WorkersDay>
        <WorkersDay Day="3">78</WorkersDay>
        <WorkersDay Day="4">71</WorkersDay>
        <WorkersDay Day="5">20</WorkersDay>
        <WorkersDay Day="6">27</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
    <ContractInformation ContractName="30">
      <WorkersCount>90</WorkersCount>
      <Shifts>450</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>21600</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">77</WorkersDay>
        <WorkersDay Day="1">57</WorkersDay>
        <WorkersDay Day="2">53</WorkersDay>
        <WorkersDay Day="3">51</WorkersDay>
        <WorkersDay Day="4">65</WorkersDay>
        <WorkersDay Day="5">72</WorkersDay>
        <WorkersDay Day="6">75</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
  </ObjectiveValue>
```

Figura 4-3. Sección ObjectiveValue del ejemplo TWPF 40H-30H

Si comparamos estos resultados con el caso en el que únicamente hay trabajadores de 40H con días de descanso seguidos, se observa que el exceso de carga de trabajo se reduce de 106.200 a 52.200 minutos, y el coste de la plantilla disminuye de 52.500 a 47.100 UM. Estos resultados se deben a que es más sencillo encajar jornadas de 8 y 6 horas a una demanda de carga de trabajo con tantas variaciones, que solo con jornadas de 8 horas.

4.3 Plantilla con tres tipos de contrato

En ese caso de uso se resuelve el TWFP con tres tipos de contrato. Se considera un contrato de trabajadores de 40 horas semanales, otro contrato de 30 horas semanales, y otro contrato de 20 horas semanales, todos con cinco días de trabajo semanales, sin jornadas partidas, y con días de descanso seguidos. La definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 40H-30H-20H' de la página de ejemplos del TWFP de VETU. El resultado del problema puede verse en el fichero ejemplo 'Out TWFP 40H-30H-20H'. La Figura 4-5 muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. Para cubrir la carga de trabajo dada, se necesitan 79 trabajadores de 40H, con un coste de 23.700 UM, 61 trabajadores de 30H, con un coste de 14.640 UM, y 40 trabajadores de 20H con un coste de 6.800 UM, y un coste total de la plantilla de 45.140 UM. Estos trabajadores ofrecen una carga de trabajo de 347.400 minutos, lo que representa un exceso de 347.400 – 313.800 = 33.600 minutos.

```
<ObjectiveValue>
  <Workers>180</Workers>
  <Journeys>900</Journeys>
  <DemandedTime>313800</DemandedTime>
  <CoveredTime>347400</CoveredTime>
  <Cost>45140</Cost>
  <ContractInformation ContractName="40">
    <WorkersCount>79</WorkersCount>
    <Shifts>395</Shifts>
    <SplitedShifts>0</SplitedShifts>
    <Cost>23700</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">79</WorkersDay>
      <WorkersDay Day="1">79</WorkersDay>
      <WorkersDay Day="2">78</WorkersDay>
      <WorkersDay Day="3">73</WorkersDay>
      <WorkersDay Day="4">67</WorkersDay>
      <WorkersDay Day="5">6</WorkersDay>
      <WorkersDay Day="6">13</WorkersDay>
    </WorkersPerDay>
  </ContractInformation>
  <ContractInformation ContractName="30">
    <WorkersCount>61</WorkersCount>
    <Shifts>305</Shifts>
    <SplitedShifts>0</SplitedShifts>
    <Cost>14640</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">35</WorkersDay>
      <WorkersDay Day="1">33</WorkersDay>
      <WorkersDay Day="2">52</WorkersDay>
      <WorkersDay Day="3">34</WorkersDay>
      <WorkersDay Day="4">41</WorkersDay>
      <WorkersDay Day="5">55</WorkersDay>
      <WorkersDay Day="6">55</WorkersDay>
    </WorkersPerDay>
  </ContractInformation>
  <ContractInformation ContractName="20">
    <WorkersCount>40</WorkersCount>
    <Shifts>200</Shifts>
    <SplitedShifts>0</SplitedShifts>
    <Cost>6800</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">30</WorkersDay>
      <WorkersDay Day="1">18</WorkersDay>
      <WorkersDay Day="2">25</WorkersDay>
      <WorkersDay Day="3">22</WorkersDay>
      <WorkersDay Day="4">25</WorkersDay>
      <WorkersDay Day="5">40</WorkersDay>
      <WorkersDay Day="6">40</WorkersDay>
    </WorkersPerDay>
  </ContractInformation>
</ObjectiveValue>
```

Figura 4-4. Sección ObjectiveValue del ejemplo TWPF 40H-30H-20H

Si comparamos estos resultados con el caso en el que únicamente hay trabajadores de 40H, se observa que el exceso de carga de trabajo se reduce de 106.200 a 33.600 minutos, y el coste de la plantilla disminuye de 52.500 a 45.140 UM. Si comparamos estos resultados con el caso en el que hay trabajadores de 40H y 30H, se observa que el exceso de carga de trabajo

se reduce de 52.200 a 33.600 UM, y el coste de la plantilla disminuye de 47.100 a 45.140. Estos resultados se deben a que es más sencillo encajar jornadas de 8, 6 y 4 horas a una carga de trabajo con tantas variaciones, que solo con jornadas de 8 y 6 horas.

4.4 Plantilla de un contrato con jornadas partidas

En ese caso de uso se resuelve el TWFP con un solo tipo de contrato. Se considera un contrato de trabajadores de 40 horas semanales, con 5 días de trabajo semanales, con jornadas partidas. Cada trabajador puede partir dos jornadas a la semana, con un descanso mínimo de 60 minutos (una hora) y máximo de 180 minutos (3 horas). La definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 40H Split' de la página de ejemplos del TWFP de VETU. El resultado del problema puede verse en el fichero ejemplo 'Out TWFP 40H Split'. La Figura 4-5 muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. Para cubrir la carga de trabajo dada, se necesitan 164 trabajadores de 40H, con un coste de 49.526,8 UM. Estos trabajadores ofrecen una carga de trabajo de 393.600 minutos, lo que representa un exceso de 393.600 – 313.800 = 79.800 minutos.

```
    <ObjectiveValue>
    <Workers>164</Workers>
    <Journeys>820</Journeys>
    <DemandedTime>313800</DemandedTime>
    <CoveredTime>393600</CoveredTime>
    <Cost>49546.8</Cost>
    <ContractInformation ContractName="40">
      <WorkersCount>164</WorkersCount>
      <Shifts>820</Shifts>
      <SplitedShifts>328</SplitedShifts>
      <Cost>49546.8</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">126</WorkersDay>
        <WorkersDay Day="1">126</WorkersDay>
        <WorkersDay Day="2">133</WorkersDay>
        <WorkersDay Day="3">126</WorkersDay>
        <WorkersDay Day="4">134</WorkersDay>
        <WorkersDay Day="5">76</WorkersDay>
        <WorkersDay Day="6">99</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
  </ObjectiveValue>
```

Figura 4-5. Sección ObjectiveValue del ejemplo TWPF 40H Split

Comparando los resultados de este caso con los obtenidos sin jornada partida (primer estudio del apartado 4.1), se observa que la opción de partir 2 jornadas a la semana, reduce la plantilla de 175 a 164 trabajadores, reduce el coste de 52.500 a 49.546'8, y reduce el exceso de trabajo de 106.200 a 79.800.

4.5 Plantilla de varios contratos con jornadas partidas

En este caso de uso se van considerar simultáneamente contratos de 40, 30 y 20 horas semanales, con cinco días de trabajo semanales, jornadas partidas y posibilidad de días de descanso semanal separados. La definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 40H-30H-20H Split NoConsecutiveRests' de la página de ejemplos del TWFP

de VETU. El resultado del problema puede verse en el fichero ejemplo 'Out TWFP 40H-30H-20H Split NoConsecutiveRests'. La Figura 4-6muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. Para cubrir la carga de trabajo dada, se necesitan 84 trabajadores de 40H, con un coste de 25.220,4 UM, 31 trabajadores de 30H y un coste de 7.440 UM, y 50 trabajadores de 20H con un coste de 8.510,2, lo que lleva a un coste total de la plantilla de 41.170,6 UM. Estos trabajadores ofrecen una carga de trabajo de 317.400 minutos, lo que representa un exceso de 317.400 – 313.800 = 3.600 minutos.

```
<ObjectiveValue>
  <Workers>165</Workers>
  <Journeys>825</Journeys>
  <DemandedTime>313800</DemandedTime>
  <CoveredTime>317400</CoveredTime>
  <Cost>41170.6</Cost>
  <ContractInformation ContractName="40">
    <WorkersCount>84</WorkersCount>
    <Shifts>420</Shifts>
    <SplitedShifts>252</SplitedShifts>
    <Cost>25220.4</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">75</WorkersDay>
      <WorkersDay Day="1">71</WorkersDay>
      <WorkersDay Day="2">69</WorkersDay>
      <WorkersDay Day="3">67</WorkersDay>
      <WorkersDay Day="4">77</WorkersDay>
      <WorkersDay Day="5">34</WorkersDay>
      <WorkersDay Day="6">27</WorkersDay>
    </WorkersPerDay>
  </ContractInformation>
  <ContractInformation ContractName="30">
    <WorkersCount>31</WorkersCount>
    <Shifts>155</Shifts>
    <SplitedShifts>93</SplitedShifts>
    <Cost>7440</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">26</WorkersDay>
      <WorkersDay Day="1">28</WorkersDay>
      <WorkersDay Day="2">28</WorkersDay>
      <WorkersDay Day="3">30</WorkersDay>
      <WorkersDay Day="4">5</WorkersDay>
      <WorkersDay Day="5">12</WorkersDay>
      <WorkersDay Day="6">26</WorkersDay>
    </WorkersPerDay>
  </ContractInformation>
  <ContractInformation ContractName="20">
    <WorkersCount>50</WorkersCount>
    <Shifts>250</Shifts>
    <SplitedShifts>150</SplitedShifts>
    <Cost>8510.2</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">22</WorkersDay>
      <WorkersDay Day="1">33</WorkersDay>
      <WorkersDay Day="2">31</WorkersDay>
      <WorkersDay Day="3">34</WorkersDay>
      <WorkersDay Day="4">38</WorkersDay>
      <WorkersDay Day="5">46</WorkersDay>
      <WorkersDay Day="6">46</WorkersDay>
    </WorkersPerDay>
```

```
    </ContractInformation>
  </ObjectiveValue>
```

Figura 4-6. Sección ObjectiveValue del ejemplo TWPF 40H-30H-20H Split NoConsecutiveRests

Comparando el resultado obtenido de este caso de uso con el resultado obtenido para una plantilla con los mismos tipos de contrato, pero sin partir jornada y juntando los días de descanso (caso de uso del apartado 4.3), se observa que el coste se reduce desde 45.146 hasta 41.170'6 UM, y el exceso de carga de trabajo sobre la demanda baja desde 33.600 a 3.600 minutos. Este resultado es debido a la flexibilidad que ofrecen las jornadas partidas, para adaptarse a cargas de trabajo que varían mucho.

4.6 Contrato definido por turnos de trabajo

En este caso de uso se va a calcular la cantidad de trabajadores necesarios para tres turnos de trabajo. Se definen tres turnos de trabajo:

M	Mañana	Cualquier jornada que comience entre las 06:00 y las 10:00 de cada día
T	Tarde	Cualquier jornada que comience entre las 13:00 y las 16:00 de cada día
N	Noche	Cualquier jornada que comience entre las 21:00 y las 23:00 de cada día

Para crear los datos que reflejen este modelo de negocio, se deben crear tres tipos de contrato: 40M para los turnos de mañana, 40T para los turnos de tarde, y 40N para los turnos de noche. Los tres tipos de contrato tienen las mismas características (días de descanso seguidos, jornadas partidas, coste , …) pero difieren en las franjas horarias de comienzo de jornada (objeto PossibleBegin de la lista PossibleBegins). La Tabla 4-1 muestra los intervalos de comienzo de jornada de cada tipo de contrato, expresados en formato hora y sus minutos equivalentes.

La definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 40MTN' de la página de ejemplos del TWFP de VETU. El resultado del problema puede verse en el fichero ejemplo 'Out TWFP 40MTN'. La Figura 4-7 muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. Para cubrir la carga de trabajo dada, se necesitan 95 trabajadores en turno de mañana, 84 trabajadores en turno de tarde, y 8 trabajadores en turno de noche, lo que lleva a un coste total de la plantilla de 56.100 UM.

		40M		40T		40N	
Día	Intervalo	Hora	Minutos	Hora	Minutos	Hora	Minutos
Lunes	Comienzo	04:00	240	13:00	780	21:00	1260
	Fin	10:00	600	16:00	960	23:00	1380
Martes	Comienzo	06:00	1800	13:00	2220	21:00	2700
	Fin	10:00	2040	16:00	2400	23:00	2820
Miércoles	Comienzo	06:00	3240	13:00	3660	21:00	4140
	Fin	10:00	3480	16:00	3840	23:00	4260
Jueves	Comienzo	06:00	4680	13:00	5100	21:00	5580
	Fin	10:00	4920	16:00	5280	23:00	5700
Viernes	Comienzo	06:00	6120	13:00	6540	21:00	7020
	Fin	10:00	6360	16:00	6720	23:00	7140

Sábado	Comienzo	06:00	7560	13:00	7980	21:00	8460
	Fin	10:00	7800	16:00	8160	23:00	8580
Domingo	Comienzo	06:00	9000	13:00	9420	21:00	9900
	Fin	10:00	9240	16:00	9600	23:00	10020

Tabla 4-1. Intervalos de comienzo de los turnos de cada contrato

```
<ObjectiveValue>
  <Workers>187</Workers>
  <Journeys>935</Journeys>
  <DemandedTime>313800</DemandedTime>
  <CoveredTime>448800</CoveredTime>
  <Cost>56100</Cost>
  <ContractInformation ContractName="40M">
    <WorkersCount>95</WorkersCount>
    <Shifts>475</Shifts>
    <SplitedShifts>0</SplitedShifts>
    <Cost>28500</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">95</WorkersDay>
      <WorkersDay Day="1">63</WorkersDay>
      <WorkersDay Day="2">63</WorkersDay>
      <WorkersDay Day="3">64</WorkersDay>
      <WorkersDay Day="4">64</WorkersDay>
      <WorkersDay Day="5">63</WorkersDay>
      <WorkersDay Day="6">63</WorkersDay>
    </WorkersPerDay>
  </ContractInformation>
  <ContractInformation ContractName="40T">
    <WorkersCount>84</WorkersCount>
    <Shifts>420</Shifts>
    <SplitedShifts>0</SplitedShifts>
    <Cost>25200</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">66</WorkersDay>
      <WorkersDay Day="1">66</WorkersDay>
      <WorkersDay Day="2">63</WorkersDay>
      <WorkersDay Day="3">63</WorkersDay>
      <WorkersDay Day="4">60</WorkersDay>
      <WorkersDay Day="5">39</WorkersDay>
      <WorkersDay Day="6">63</WorkersDay>
    </WorkersPerDay>
  </ContractInformation>
  <ContractInformation ContractName="40N">
    <WorkersCount>8</WorkersCount>
    <Shifts>40</Shifts>
    <SplitedShifts>0</SplitedShifts>
    <Cost>2400</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">8</WorkersDay>
      <WorkersDay Day="1">5</WorkersDay>
      <WorkersDay Day="2">4</WorkersDay>
      <WorkersDay Day="3">7</WorkersDay>
      <WorkersDay Day="4">8</WorkersDay>
      <WorkersDay Day="5">4</WorkersDay>
```

```
        <WorkersDay Day="6">4</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
  </ObjectiveValue>
```

Figura 4-7. Sección ObjectiveValue del ejemplo TWPF 40MTN

Comparando con el caso en el que todos los trabajadores podían entran a trabajar a cualquier hora del día (primer caso del apartado 4.1), se observa que el coste sube de 52.500 a 56.100 UM, y la plantilla pasa de 175 a 187 trabajadores. En este caso las jornadas de los trabajadores son mejores, porque se puede asegurar que las jornadas de trabajo en días consecutivos van a comenzar sobre la misma hora, lo que supone mejores turnos para los trabajadores, con un incremento de coste de 3.600 UM.

4.7 Demanda de carga de trabajo de lunes a viernes

Hasta ahora, todos los casos de uso se han realizado con demanda de carga de trabajo para todos los días de la semana. En este caso de uso no se define demanda de carga de trabajo en sábado y domingo, solo de lunes a viernes, y se emplean tres contratos, que pueden partir hasta cinco jornadas a la semana, y con posibilidad de descansar dos días separados a la semana.

La definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 40H-30H-20H Split-MondayToFriday'' de la página de ejemplos del TWFP de VETU. El resultado del problema puede verse en el fichero ejemplo 'Out TWFP 40H-30H-20H Split-MondayToFriday'. La Figura 4-8 muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. Como hay dos días de descanso semanal, y solo hay carga de trabajo de lunes (día 0) a viernes (día 4), todos los trabajadores descansan el sábado (día 5) y el domingo (día 6), como se puede ver en el objeto WorkersDay de cada objeto ContractInformation. También puede verse este efecto en los objetos WeekPattern, que solo tienen los valores <RestDay>**5**</RestDay> y <RestDay>**6**</RestDay>.

```
  <ObjectiveValue>
    <Workers>129</Workers>
    <Journeys>645</Journeys>
    <DemandedTime>248340</DemandedTime>
    <CoveredTime>255000</CoveredTime>
    <Cost>21850.6</Cost>
    <ContractInformation ContractName="40">
      <WorkersCount>72</WorkersCount>
      <Shifts>360</Shifts>
      <SplitedShifts>242</SplitedShifts>
      <Cost>14430.6</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">72</WorkersDay>
        <WorkersDay Day="1">72</WorkersDay>
        <WorkersDay Day="2">72</WorkersDay>
        <WorkersDay Day="3">72</WorkersDay>
        <WorkersDay Day="4">72</WorkersDay>
        <WorkersDay Day="5">0</WorkersDay>
        <WorkersDay Day="6">0</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
    <ContractInformation ContractName="30">
      <WorkersCount>23</WorkersCount>
      <Shifts>115</Shifts>
```

```
      <SplitedShifts>68</SplitedShifts>
      <Cost>3680</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">23</WorkersDay>
        <WorkersDay Day="1">23</WorkersDay>
        <WorkersDay Day="2">23</WorkersDay>
        <WorkersDay Day="3">23</WorkersDay>
        <WorkersDay Day="4">23</WorkersDay>
        <WorkersDay Day="5">0</WorkersDay>
        <WorkersDay Day="6">0</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
    <ContractInformation ContractName="20">
      <WorkersCount>34</WorkersCount>
      <Shifts>170</Shifts>
      <SplitedShifts>155</SplitedShifts>
      <Cost>3740</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">34</WorkersDay>
        <WorkersDay Day="1">34</WorkersDay>
        <WorkersDay Day="2">34</WorkersDay>
        <WorkersDay Day="3">34</WorkersDay>
        <WorkersDay Day="4">34</WorkersDay>
        <WorkersDay Day="5">0</WorkersDay>
        <WorkersDay Day="6">0</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
  </ObjectiveValue>
```

Figura 4-8. Sección ObjectiveValue del ejemplo TWFP 40H-30H-20H Split-MondayToFriday

4.8 Plantillas diferentes para diario y fin de semana

En este caso de uso se va dimensionar una plantilla de trabajadores para los días de diario y otra para el fin de semana. Para los días de diario, se va a contar con contratos de 40 y 30 horas semanales, y para el fín de semana se contará con trabajadores de jornadas diarias de 4 horas (8 horas semanales, porque solo trabajan dos días). Para que los trabajadores de 40 y 30 horas semanales solo trabajen de lunes a viernes se especifica el objeto PossibleBegin que muestra la Figura 4-9. El instante de tiempo 7.140 corresponde a las 23:00 del viernes, que es el momento más tardío en el que podrá comenzar una jornada de trabajo asociada a los contratos de 40 0 30 horas semanales. Para que los trabajadores del contrato de 16 horas solo trabajen los fines de semana, se fija el campo RestsPerWeek a 5 y se define el objeto PossibleBegin como muestra la Figura 4-10. El instante de tiempo 7.200 corresponde a las 00:00 del sábado, y el instante 10.080 corresponde a las 24:00 del domingo.

```
      <PossibleBegins>
        <PossibleBegin>
          <From>0</From>
          <To>7140</To>
        </PossibleBegin>
      </PossibleBegins>
```

Figura 4-9. Objeto PossibleBegin de los contratos de 40 y 30 horas semanales.

```
        <PossibleBegins>
          <PossibleBegin>
            <From>7200</From>
            <To>10080</To>
          </PossibleBegin>
        </PossibleBegins>
```

Figura 4-10. Objeto PossibleBegin del contrato de 8 horas semanales.

La definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 40H-30H-16H TwoStaffs' de la página de ejemplos del TWFP de VETU. El resultado del problema puede verse en el fichero ejemplo 'Out TWFP 40H-30H-16H TwoStaffs'. La Figura 4-8 muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. El objeto WorkersPerDay de los contratos de 40 y 30 horas semanales indica que todos los trabajadores trabaja de lunes a viernes, mientras que ese mismo objeto para el contrato de 16 horas muestra que todos los trabajadores tienes jornada el sábado y el domingo.

```
  <ObjectiveValue>
    <Workers>295</Workers>
    <Journeys>989</Journeys>
    <DemandedTime>313800</DemandedTime>
    <CoveredTime>338760</CoveredTime>
    <Cost>29950</Cost>
    <ContractInformation ContractName="40">
      <WorkersCount>36</WorkersCount>
      <Shifts>180</Shifts>
      <SplitedShifts>138</SplitedShifts>
      <Cost>7291.8</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">36</WorkersDay>
        <WorkersDay Day="1">36</WorkersDay>
        <WorkersDay Day="2">36</WorkersDay>
        <WorkersDay Day="3">36</WorkersDay>
        <WorkersDay Day="4">36</WorkersDay>
        <WorkersDay Day="5">0</WorkersDay>
        <WorkersDay Day="6">0</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
    <ContractInformation ContractName="30">
      <WorkersCount>97</WorkersCount>
      <Shifts>485</Shifts>
      <SplitedShifts>357</SplitedShifts>
      <Cost>15530.2</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">97</WorkersDay>
        <WorkersDay Day="1">97</WorkersDay>
        <WorkersDay Day="2">97</WorkersDay>
        <WorkersDay Day="3">97</WorkersDay>
        <WorkersDay Day="4">97</WorkersDay>
        <WorkersDay Day="5">0</WorkersDay>
        <WorkersDay Day="6">0</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
    <ContractInformation ContractName="16">
      <WorkersCount>162</WorkersCount>
      <Shifts>324</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>7128</Cost>
      <WorkersPerDay>
```

```
        <WorkersDay Day="0">0</WorkersDay>
        <WorkersDay Day="1">0</WorkersDay>
        <WorkersDay Day="2">0</WorkersDay>
        <WorkersDay Day="3">0</WorkersDay>
        <WorkersDay Day="4">0</WorkersDay>
        <WorkersDay Day="5">162</WorkersDay>
        <WorkersDay Day="6">162</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
  </ObjectiveValue>
```

Figura 4-11. Sección ObjectiveValue del ejemplo TWFP 40H-30H-16H TwoStaffs

4.9 Refuerzo para el fin de semana

En este caso de uso vamos a considerar una empresa que dispone de contratos de 40 y 30 horas semanales, que trabajan durante toda la semana. Estos trabajadores son todos fijos, y en general desean descansar los fines de semana. Se ha producido un aumento en la carga de trabajo, y se desea evaluar la cantidad de trabajadores que se deberían contratar para trabajar solo el fin de semana, de forma que los trabajadores fijos puedan descansar el fin de semana. Se decide que se pueden contratar trabajadores de jornadas de 4 y 6 horas diarias (8 y 16 horas semanales, respectivamente), con posibilidad de partir jornadas durante una hora, sin coste adicional. Los trabajadores fijos no pueden partir jornada, y descansan los dos días seguidos. La definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 40H-30H-8H-16H NewContracts' de la página de ejemplos del TWFP de VETU. El resultado del problema puede verse en el fichero ejemplo 'In TWFP 40H-30H-8H-16H NewContracts'. La Figura 4-12 muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. El objeto WorkersPerDay de los nuevos contratos (8 y 12 horas semanales) muestra que solo trabajan los fines de semana, y el mismo objeto muestra que los trabajadores fijos de 30 horas descansan todos el fin de semana, y que hay pocos trabajadores del contrato fijo de 40 horas que trabajan el fin de semana (1 trabajador el sábado y 11 trabajadores el domingo). Esto es así porque los costes de los nuevos contratos, son menores en proporción que los de los contratos fijos. Es necesario contratar 61 trabajadores de 8 horas semanales, y 49 trabajadores de 12 horas semanales.

```
  <ObjectiveValue>
    <Workers>243</Workers>
    <Journeys>885</Journeys>
    <DemandedTime>313800</DemandedTime>
    <CoveredTime>350160</CoveredTime>
    <Cost>57554</Cost>
    <ContractInformation ContractName="40">
      <WorkersCount>77</WorkersCount>
      <Shifts>385</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>30800</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">77</WorkersDay>
        <WorkersDay Day="1">76</WorkersDay>
        <WorkersDay Day="2">76</WorkersDay>
        <WorkersDay Day="3">77</WorkersDay>
        <WorkersDay Day="4">67</WorkersDay>
        <WorkersDay Day="5">1</WorkersDay>
        <WorkersDay Day="6">11</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
```

```
    <ContractInformation ContractName="30">
      <WorkersCount>56</WorkersCount>
      <Shifts>280</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>16800</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">56</WorkersDay>
        <WorkersDay Day="1">56</WorkersDay>
        <WorkersDay Day="2">56</WorkersDay>
        <WorkersDay Day="3">56</WorkersDay>
        <WorkersDay Day="4">56</WorkersDay>
        <WorkersDay Day="5">0</WorkersDay>
        <WorkersDay Day="6">0</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
    <ContractInformation ContractName="8">
      <WorkersCount>61</WorkersCount>
      <Shifts>122</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>4270</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">0</WorkersDay>
        <WorkersDay Day="1">0</WorkersDay>
        <WorkersDay Day="2">0</WorkersDay>
        <WorkersDay Day="3">0</WorkersDay>
        <WorkersDay Day="4">0</WorkersDay>
        <WorkersDay Day="5">61</WorkersDay>
        <WorkersDay Day="6">61</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
    <ContractInformation ContractName="12">
      <WorkersCount>49</WorkersCount>
      <Shifts>98</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>5684</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">0</WorkersDay>
        <WorkersDay Day="1">0</WorkersDay>
        <WorkersDay Day="2">0</WorkersDay>
        <WorkersDay Day="3">0</WorkersDay>
        <WorkersDay Day="4">0</WorkersDay>
        <WorkersDay Day="5">49</WorkersDay>
        <WorkersDay Day="6">49</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
  </ObjectiveValue>
```

Figura 4-12. Sección ObjectiveValue del ejemplo TWFP TWFP 40H-30H-8H-16H NewContracts

En la solución de este caso, son necesarios 77 trabajadores fijos de 40 horas semanales, y 56 trabajadores fijos de 30 horas semanales. Supongamos que la plantilla actual cuenta con 90 trabajadores fijos de 40 horas semanales, y 50 trabajadores fijos de 30 horas. La empresa quiere mantener a todos los trabajadores fijos de 40 y 30 horas, aunque está dispuesto a contratar trabajadores fijos de 30 horas, si se reducen los costes totales. Para lograr este efecto, se fija el objeto WorkersAmount del contrato de 40 horas que muestra la Figura 4-13, donde se fija la cantidad exacta de 90 trabajadores, y se fija el objeto WorkersAmount del contrato de 30 horas que muestra la Figura 4-14. El objeto WorkersAmount del contrato de 30 horas semanales, donde se fija un número mínimo de 50 trabajadores de 30 horas. La definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 40H-30H-8H-16H

WorkersAmount' de la página de ejemplos del TWFP de VETU. El resultado del problema puede verse en el fichero ejemplo 'Out TWFP 40H-30H-8H-16H WorkersAmount'. La Figura 4-15 muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. La mejor solución para este nuevo planteamiento es mantener los 50 trabajadores fijos de 30 horas semanales, junto con los 90 trabajadores fijos, contratando 33 nuevos trabajadores de 8 horas semanales, y otros 42 trabajadores de 16 horas semanales. En esta nueva solución, 66 de los 90 trabajadores fijos de 40 horas semanales descansan el sábado y el domingo como muestra la Figura 4-16, y otros 6 descansan el sábado, como muestra la Figura 4-17. En esta nueva solución, 46 de los 50 trabajadores fijos de 30 horas semanales descansan el fin de semana, como muestra la Figura 4-18.

```
        <WorkersAmount>
           <Criterion>Equal</Criterion>
           <Amount>90</Amount>
         </WorkersAmount>
```

Figura 4-13. Objeto WorkersAmount del contrato de 40 horas semanales

```
        <WorkersAmount>
           <Criterion>GreaterOrEqual</Criterion>
           <Amount>50</Amount>
         </WorkersAmount>
```

Figura 4-14. Objeto WorkersAmount del contrato de 30 horas semanales

```
  <ObjectiveValue>
    <Workers>215</Workers>
    <Journeys>850</Journeys>
    <DemandedTime>313800</DemandedTime>
    <CoveredTime>352080</CoveredTime>
    <Cost>58182</Cost>
    <ContractInformation ContractName="40">
      <WorkersCount>90</WorkersCount>
      <Shifts>450</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>36000</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">83</WorkersDay>
        <WorkersDay Day="1">81</WorkersDay>
        <WorkersDay Day="2">87</WorkersDay>
        <WorkersDay Day="3">81</WorkersDay>
        <WorkersDay Day="4">76</WorkersDay>
        <WorkersDay Day="5">18</WorkersDay>
        <WorkersDay Day="6">24</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
    <ContractInformation ContractName="30">
      <WorkersCount>50</WorkersCount>
      <Shifts>250</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>15000</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">50</WorkersDay>
        <WorkersDay Day="1">48</WorkersDay>
        <WorkersDay Day="2">46</WorkersDay>
        <WorkersDay Day="3">48</WorkersDay>
        <WorkersDay Day="4">50</WorkersDay>
        <WorkersDay Day="5">4</WorkersDay>
```

```
        <WorkersDay Day="6">4</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
    <ContractInformation ContractName="8">
      <WorkersCount>33</WorkersCount>
      <Shifts>66</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>2310</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">0</WorkersDay>
        <WorkersDay Day="1">0</WorkersDay>
        <WorkersDay Day="2">0</WorkersDay>
        <WorkersDay Day="3">0</WorkersDay>
        <WorkersDay Day="4">0</WorkersDay>
        <WorkersDay Day="5">33</WorkersDay>
        <WorkersDay Day="6">33</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
    <ContractInformation ContractName="12">
      <WorkersCount>42</WorkersCount>
      <Shifts>84</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>4872</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">0</WorkersDay>
        <WorkersDay Day="1">0</WorkersDay>
        <WorkersDay Day="2">0</WorkersDay>
        <WorkersDay Day="3">0</WorkersDay>
        <WorkersDay Day="4">0</WorkersDay>
        <WorkersDay Day="5">42</WorkersDay>
        <WorkersDay Day="6">42</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
  </ObjectiveValue>
```

Figura 4-15. Sección ObjectiveValue del ejemplo TWFP 40H-30H-8H-16H WorkersAmount

```
      <WeekPatron>
         <Amount>66</Amount>
         <RestDay>5</RestDay>
         <RestDay>6</RestDay>
       </WeekPatron>
```

Figura 4-16. Trabajadores de 40 horas semanales que descansan el fin de semana

```
      <WeekPatron>
         <Amount>6</Amount>
         <RestDay>4</RestDay>
         <RestDay>5</RestDay>
       </WeekPatron>
```

Figura 4-17. Trabajadores de 40 horas semanales que descansan e l sábado

```
      <WeekPatron>
         <Amount>46</Amount>
         <RestDay>5</RestDay>
         <RestDay>6</RestDay>
       </WeekPatron>
```

Figura 4-18. Trabajadores de 30 horas semanales que descansan el fin de semana

4.10 Plantillas con más de dos días de descanso semanal

En los casos de uso anteriores, se han definido contratos de trabajo con dos días de descanso semanal. También se ha definido contratos de fin de semana, con cinco días de descanso semanal. El sistema de planificación táctica del personal de VETU permite definir contratos con cualquier número de días de descanso semanal, desde 0 (se trabaja todos los días) hasta 6 (se trabaja un solo día a la semana). Como muestra de esta capacidad, en este caso de uso definimos un contrato de jornadas de 32 horas semanales, con jornadas de 8 horas diarias y con tres días de descanso semanal, un contrato de 18 horas semanales, con jornadas de 6 horas diarias y con cuatro días de descanso semanal, y un contrato de 8 horas semanales, con jornadas de 4 horas semanales y 5 días de descanso semanal. La definición del problema se puede encontrar en el fichero ejemplo 'In TWFP 32H-18H-8H RestDays' de la página de ejemplos del TWFP de VETU. El resultado del problema puede verse en el fichero ejemplo 'In TWFP 32H-18H-8H RestDays'. La Figura 4-19 muestra la sección ObjectiveValue de este caso de uso, donde se muestra un resumen de los resultados. Se observa que hay trabajadores de los tres tipos de contrato trabajando todos los días de la semana. El objeto WeekPatrons de cada uno de los tipos de contrato que contiene el fichero con la solución de este problema muestra los patrones de descanso semanal, que cumplen con los requisitos definidos.

```
<ObjectiveValue>
  <Workers>291</Workers>
  <Journeys>867</Journeys>
  <DemandedTime>313800</DemandedTime>
  <CoveredTime>336120</CoveredTime>
  <Cost>28010</Cost>
  <ContractInformation ContractName="32">
    <WorkersCount>106</WorkersCount>
    <Shifts>424</Shifts>
    <SplitedShifts>0</SplitedShifts>
    <Cost>16960</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">78</WorkersDay>
      <WorkersDay Day="1">79</WorkersDay>
      <WorkersDay Day="2">92</WorkersDay>
      <WorkersDay Day="3">89</WorkersDay>
      <WorkersDay Day="4">57</WorkersDay>
      <WorkersDay Day="5">12</WorkersDay>
      <WorkersDay Day="6">17</WorkersDay>
    </WorkersPerDay>
  </ContractInformation>
  <ContractInformation ContractName="18">
    <WorkersCount>73</WorkersCount>
    <Shifts>219</Shifts>
    <SplitedShifts>0</SplitedShifts>
    <Cost>6570</Cost>
    <WorkersPerDay>
      <WorkersDay Day="0">19</WorkersDay>
      <WorkersDay Day="1">29</WorkersDay>
      <WorkersDay Day="2">18</WorkersDay>
      <WorkersDay Day="3">24</WorkersDay>
      <WorkersDay Day="4">41</WorkersDay>
      <WorkersDay Day="5">51</WorkersDay>
      <WorkersDay Day="6">37</WorkersDay>
    </WorkersPerDay>
  </ContractInformation>
  <ContractInformation ContractName="8">
    <WorkersCount>112</WorkersCount>
```

```
      <Shifts>224</Shifts>
      <SplitedShifts>0</SplitedShifts>
      <Cost>4480</Cost>
      <WorkersPerDay>
        <WorkersDay Day="0">40</WorkersDay>
        <WorkersDay Day="1">30</WorkersDay>
        <WorkersDay Day="2">16</WorkersDay>
        <WorkersDay Day="3">16</WorkersDay>
        <WorkersDay Day="4">29</WorkersDay>
        <WorkersDay Day="5">38</WorkersDay>
        <WorkersDay Day="6">55</WorkersDay>
      </WorkersPerDay>
    </ContractInformation>
  </ObjectiveValue>
```

Figura 4-19. Sección ObjectiveValue del ejemplo

4.11 Conclusión

A lo largo del presente capítulo se han mostrado diferentes opciones para modelar una instancia del TWFP y se ha estudiado su efecto en la plantilla que se calcula. Un método de trabajo para la definición de plantillas de trabajo consiste en ir fijando diversas opciones de contrato, estudiar su impacto, y valorar si se aceptan esas opciones. Este proceso de fijar decisiones y valorar su impacto se puede hacer forma iterativa, hasta llegar a dimensionar la mejor plantilla para dar un servicio.

El servicio SaaS de VETU para la resolución del TWFP permite valorar de forma económica cada aspecto de definición de contratos, por lo que es una potente herramienta para la definición de plantillas de trabajadores que deben cubrir una demanda de trabajo.

5 Listas de mensajes

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

5.1 Mensajes de envío de problemas

La Tabla 5-1 muestra la lista de mensajes que puede devolver el objeto Request de la función solve() del servicio web wsTWFP, cuando se envía una solicitud de resolución de una instancia del TWFP.

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	The user is unknown. El usuario enviado con la función solve no está registrado en la plataforma SaaS de VETU. Si es usted cliente, contacte con el servicio técnico.
SRV2	The password is incorrect. La password enviada con la función solve no es correcta. Si es usted cliente, contacte con el servicio técnico.
SRV3	The user has not access to this problem. El usuario enviado con la function solve() no tiene contrato para resolver el TWFP de VETU.
SRV4	The contract has expired El contrato para accceder a a resolución del TWFP de VETU ha caducado y no puede ser utilizado el servicio.

Tabla 5-1. Mensajes que devuelve la función solve() del servicio web WSTWFP

5.2 Mensajes del TWFP

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

Código	Descripción
TWFP0	The FAP problem has been successfully solved El problema se ha resuelto corectamente.
TWFP1	Error solving the problem. Please contact support team. Ha ocurrrido 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.
TWFP2	The problem has no feasible solution. Please check problem data. El problema no tiene solución. Revise las condiciones fijadas como Exact = true en los objetos Demand. Revise las condiones WorkersAmount. Revise los objetos PossibleBegin. Si después de revisar el problema, no encuentra la causa por la que no se puede resolver, contacte con el servicio técnico.
TWFP10	TimeUnit = yy for StaffDemand is not valid. It must be 15, 30 or 60. El campo TimeUnit del objeto StaffDemand tiene un valor incorrecto, porque solo puede valer 15, 30 o 60.
TWFP11	[xx,yy] is not valid for TimeUnit = zz of StaffDemand. Lo dos valores del intevalo deben especificar un intervalo válido en función del valor del campo TimeUnit definido. Los dos valores deben ser múltiplo de TimeUnit. Por ejemplo, si TimeUnit vale 60, el interval [90, 120] no es válido porque 90 no es múltiplo de 60.
TWFP12	The demand for interval [x,y] can not be covered by any contract. Se ha definido una demanda positiva para un determinado intervalo de la demanda de carga de trabajo, y ningún tipo de contrato puede crear jornadas de trabajo en ese intervalo, en function del objeto PossibleBegin. Por ejemplo, si se define carga de trabajo para el intervalo horario [120, 180], y solo hay un contrato que tiene definido PosibleBegin = [180, 360], se producirá este error porque es imposible crear una jornada que cubra la demanda en [120,180].

TWFP20	Contract XX: MaximunSplitShifts = yy is not correct. El número de jornadas partidas yy no puede ser negativo, nulo, ni mayor que el número de jornadas de trabajo que define el contrato.
TWFP21	Contract XX: Minimun Subshift = yy is not correct. La duración yy de la minima subjornada de una jornada partida no puede ser negativa ni nula.
TWFP22	Contract XX: Minimun Subshift = yy can not be greather or equal than shift duracion = zz. La duración yy de la minima subjornada de una jornada partida no puede ser mayor que la duración de la jornada diaría.
TWFP23	Contract XX: Minimun Rest = yy must be greather than 0. El descanso mínimo yy de una jornada partida no puede ser negativo ni nulo.
TWFP24	Contract XX: Maximun Rest = yy must be greather than 0 El descanso máximo yy de una jornada partida no puede ser negativo ni nulo.
TWFP25	Contract XX: Maximun Rest = yy must be greather than Minimun Rest = zz. El descanso máximo de una jornada partida debe ser mayor que el descanso mínimo.
TWFP26	There is not any contract No se ha definido ningún contrato válido para el problema, así que no se puede resolver.
TWFP27	TimeUnit = yy for contracts is not valid. It must be 30 or 60. El campo TimeUnit del objeto Contract tiene un valor incorrecto, porque solo puede valer 30 o 60.
TWFP28	Contract XX: SSS is not a valid criterio for number of workers. Los valores que puede tomar el campo Criterio del objeto WorkersAmount debe tomar uno de los siguientes valores: GreaterOrEqual, Equal, LessOrEqual.
TWFP29	Contract XX: WorkersAmount.Amount = xx is not valid. It must be greater than 0.

	El número de trabajadores que se indica en el campo Amount del objeto WorkersAmount no puede ser negativo ni nulo.
TWFP30	Contract XX: Cost = xx is not valid. It must be greater than 0. El coste de una jornada no puede ser negativo ni nulo.
TWFP31	Contract XX: RestCostPerMinut = xx is not valid. It must be greater than 0. El coste por minuto de una jornada partida no puede ser negativo ni nulo.
TWFP32	Contract XX: There is not any PossibleBegin. Define al least one. Todo contrato debe tener al menos definido un objeto PossibleBegin, para indicar la franja horaria en la que pueden comenzar las jornadas de trabajo asociadas a ese contrato.
TWFP33	Contract XX: PossibleBegin [yy,zz] is no valid. From must be greater or equal than 0. El valor del campo From de un objeto PossibleBegin no puede ser negativo.
TWFP34	Contract XX: PossibleBegin [yy,zz] is no valid. From must be less or equal than 10.080 minutes. El valor del campo To de objeto PossibleBegin debe ser mayor o igual que el campo From.
TWFP35	Contract XX: PossibleBegin [yy,zz] is no valid. To must be greater or equal than 0. El valor del campo To de un objeto PossibleBegin no puede ser negativo.
TWFP36	Contract XX: PossibleBegin [yy,zz] is no valid. To must be less or equal than 10.080 minutes. El valor del campo To de un objeto PossibleBegin no puede mayor que la cantidad de minutos de una semana (10.080).
TWFP37	Contract XX: PossibleBegin [yy,zz] is no valid. To must be greater or equal than From. El valor del campo To del objeto PossibleBegin debe ser mayor o igual que el valor del campo From.
TWFP38	Contract XX: RestsPerWeek = yy is not a valid value. It must be less than 7. El valor del campo RestPerWeek debe ser menor que 7. Si un trabajador tiene 7

	días de descanso a la semana significa que no trabaja ningún día, y por tanto no se generan jornadas de trabajo para él.
TWFP39	Contract XX: There are not any possible rest patterns in the week. Check the defined work load and RestsPerWeek. Según la definición del contrato y la carga de trabajo, no se puede generar ningún patron de descanso semanal para los trabajadores de este contrato. Un ejemplo en que se da este error, es si se define carga de trabajo de lunes a viernes y el contrato solo tiene un día de descanso semanal.
TWFP40	Contract XX: There is not any possible shift pattern in the week. Check the defined work load and contract definition. No se ha podido generar ninguna jornada de trabajo para este tipo de contrato. Se debe revisar la carga de trabajo, la definición de descansos y posibles comienzos del contrato.

6 Próxima versión

Este manual corresponde a la primera versión de la resolución del TWFP de VETU. Aplicando la política de consolidación del sistema SaaS de VETU, y dar un mejor servicio a sus clientes, Infozara está trabajando continuamente en la mejora de sus productos. Estas mejoras para el TWFP 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.

En este capítulo se describen las nuevas funciones que tendrá la próxima versión del TWFP de VETU. Si desea más información sobre estas nuevas funciones, puede ponerse en contacto con el servicio técnico de Infozara.

6.1 Horas en las que son se pueden partir jornadas

En general, se pueden partir jornadas a cualquier hora del día, pero hay colectivos de trabajadores que no permiten partir jornada en ciertas franjas horarias. En la próxima versión del TWFP de VETU, se podrán especificar franjas horarias en las que no se pueden partir jornadas.

6.2 Patrones de jornada definidos por el usuario

En el TFWP se define una serie de parámetros que indican las características de las jornadas partidas que se pueden crear en un cuadrante de turnos. Pero existen momentos concretos en los que por la forma de la carga de trabajo, se podrían aplicar patrones de jornada partida diferentes a los generales. Esta opción permitirá al usuario fijar esos patrones para un determinado contrato.

6.3 Varios objetos WorkersAmount

En la próxima versión del TWFP de VETU se podrán incluir varios objetos WorkersAmount en la definición de contratos, para expresar restricciones como: 'debe haber más al menos 20 trabajadores, pero no más 30'.

7 Anexos

En este capítulo se muestra diversa información referenciada en el resto del documento

7.1 Carga de trabajo de los casos de uso

La muestra la carga de trabajo que se utiliza para resolver los ejemplos de este manual de usuario. Se muestra la versión XML del objeto StaffDemand. En las siguientes figuras se muestra de forma gráfica la carga de trabajo para cada uno de los días de la semana. En todas estas figuras, se muestra una escala temporal, y en el eje vertical se muestra la cantidad de trabajadores necesarios.

```
<StaffDemand>
  <TimeUnit>60</TimeUnit>
  <Demands>
    <Demand>
      <Begin>0</Begin>
      <End>60</End>
      <Employees>0</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>60</Begin>
      <End>120</End>
      <Employees>0</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>120</Begin>
      <End>180</End>
      <Employees>0</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>180</Begin>
      <End>240</End>
      <Employees>0</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>240</Begin>
      <End>300</End>
      <Employees>3</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>300</Begin>
      <End>360</End>
      <Employees>3</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>360</Begin>
```

```
        <End>420</End>
        <Employees>50</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>420</Begin>
        <End>480</End>
        <Employees>45</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>480</Begin>
        <End>540</End>
        <Employees>60</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>540</Begin>
        <End>600</End>
        <Employees>60</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>600</Begin>
        <End>660</End>
        <Employees>63</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>660</Begin>
        <End>720</End>
        <Employees>42</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>720</Begin>
        <End>780</End>
        <Employees>53</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>780</Begin>
        <End>840</End>
        <Employees>50</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>840</Begin>
        <End>900</End>
        <Employees>36</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>900</Begin>
        <End>960</End>
        <Employees>30</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>960</Begin>
```

```
        <End>1020</End>
        <Employees>48</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1020</Begin>
        <End>1080</End>
        <Employees>54</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1080</Begin>
        <End>1140</End>
        <Employees>39</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1140</Begin>
        <End>1200</End>
        <Employees>39</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1200</Begin>
        <End>1260</End>
        <Employees>60</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1260</Begin>
        <End>1320</End>
        <Employees>54</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1320</Begin>
        <End>1380</End>
        <Employees>16</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1380</Begin>
        <End>1440</End>
        <Employees>12</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1440</Begin>
        <End>1500</End>
        <Employees>8</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1500</Begin>
        <End>1560</End>
        <Employees>6</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1560</Begin>
```

```
        <End>1620</End>
        <Employees>8</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1620</Begin>
        <End>1680</End>
        <Employees>4</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1680</Begin>
        <End>1740</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1740</Begin>
        <End>1800</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1800</Begin>
        <End>1860</End>
        <Employees>56</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1860</Begin>
        <End>1920</End>
        <Employees>48</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1920</Begin>
        <End>1980</End>
        <Employees>63</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>1980</Begin>
        <End>2040</End>
        <Employees>57</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>2040</Begin>
        <End>2100</End>
        <Employees>60</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>2100</Begin>
        <End>2160</End>
        <Employees>45</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>2160</Begin>
```

```
      <End>2220</End>
      <Employees>56</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>2220</Begin>
      <End>2280</End>
      <Employees>50</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>2280</Begin>
      <End>2340</End>
      <Employees>39</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>2340</Begin>
      <End>2400</End>
      <Employees>30</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>2400</Begin>
      <End>2460</End>
      <Employees>51</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>2460</Begin>
      <End>2520</End>
      <Employees>57</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>2520</Begin>
      <End>2580</End>
      <Employees>45</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>2580</Begin>
      <End>2640</End>
      <Employees>42</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>2640</Begin>
      <End>2700</End>
      <Employees>66</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>2700</Begin>
      <End>2760</End>
      <Employees>66</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>2760</Begin>
```

```
        <End>2820</End>
        <Employees>16</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>2820</Begin>
        <End>2880</End>
        <Employees>9</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>2880</Begin>
        <End>2940</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>2940</Begin>
        <End>3000</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>3000</Begin>
        <End>3060</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>3060</Begin>
        <End>3120</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>3120</Begin>
        <End>3180</End>
        <Employees>3</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>3180</Begin>
        <End>3240</End>
        <Employees>4</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>3240</Begin>
        <End>3300</End>
        <Employees>53</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>3300</Begin>
        <End>3360</End>
        <Employees>42</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>3360</Begin>
```

```
      <End>3420</End>
      <Employees>60</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>3420</Begin>
      <End>3480</End>
      <Employees>60</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>3480</Begin>
      <End>3540</End>
      <Employees>60</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>3540</Begin>
      <End>3600</End>
      <Employees>45</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>3600</Begin>
      <End>3660</End>
      <Employees>56</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>3660</Begin>
      <End>3720</End>
      <Employees>47</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>3720</Begin>
      <End>3780</End>
      <Employees>39</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>3780</Begin>
      <End>3840</End>
      <Employees>30</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>3840</Begin>
      <End>3900</End>
      <Employees>42</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>3900</Begin>
      <End>3960</End>
      <Employees>51</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>3960</Begin>
```

```
        <End>4020</End>
        <Employees>45</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4020</Begin>
        <End>4080</End>
        <Employees>42</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4080</Begin>
        <End>4140</End>
        <Employees>63</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4140</Begin>
        <End>4200</End>
        <Employees>66</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4200</Begin>
        <End>4260</End>
        <Employees>15</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4260</Begin>
        <End>4320</End>
        <Employees>9</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4320</Begin>
        <End>4380</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4380</Begin>
        <End>4440</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4440</Begin>
        <End>4500</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4500</Begin>
        <End>4560</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4560</Begin>
```

```
        <End>4620</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4620</Begin>
        <End>4680</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4680</Begin>
        <End>4740</End>
        <Employees>53</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4740</Begin>
        <End>4800</End>
        <Employees>42</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4800</Begin>
        <End>4860</End>
        <Employees>63</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4860</Begin>
        <End>4920</End>
        <Employees>60</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4920</Begin>
        <End>4980</End>
        <Employees>60</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>4980</Begin>
        <End>5040</End>
        <Employees>45</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5040</Begin>
        <End>5100</End>
        <Employees>56</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5100</Begin>
        <End>5160</End>
        <Employees>47</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5160</Begin>
```

```
        <End>5220</End>
        <Employees>39</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5220</Begin>
        <End>5280</End>
        <Employees>27</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5280</Begin>
        <End>5340</End>
        <Employees>42</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5340</Begin>
        <End>5400</End>
        <Employees>51</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5400</Begin>
        <End>5460</End>
        <Employees>45</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5460</Begin>
        <End>5520</End>
        <Employees>42</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5520</Begin>
        <End>5580</End>
        <Employees>63</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5580</Begin>
        <End>5640</End>
        <Employees>66</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5640</Begin>
        <End>5700</End>
        <Employees>16</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5700</Begin>
        <End>5760</End>
        <Employees>9</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>5760</Begin>
```

```
      <End>5820</End>
      <Employees>6</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>5820</Begin>
      <End>5880</End>
      <Employees>6</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>5880</Begin>
      <End>5940</End>
      <Employees>0</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>5940</Begin>
      <End>6000</End>
      <Employees>0</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>6000</Begin>
      <End>6060</End>
      <Employees>0</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>6060</Begin>
      <End>6120</End>
      <Employees>0</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>6120</Begin>
      <End>6180</End>
      <Employees>32</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>6180</Begin>
      <End>6240</End>
      <Employees>24</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>6240</Begin>
      <End>6300</End>
      <Employees>48</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>6300</Begin>
      <End>6360</End>
      <Employees>57</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>6360</Begin>
```

```
        <End>6420</End>
        <Employees>57</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>6420</Begin>
        <End>6480</End>
        <Employees>45</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>6480</Begin>
        <End>6540</End>
        <Employees>50</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>6540</Begin>
        <End>6600</End>
        <Employees>47</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>6600</Begin>
        <End>6660</End>
        <Employees>36</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>6660</Begin>
        <End>6720</End>
        <Employees>39</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>6720</Begin>
        <End>6780</End>
        <Employees>45</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>6780</Begin>
        <End>6840</End>
        <Employees>51</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>6840</Begin>
        <End>6900</End>
        <Employees>45</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>6900</Begin>
        <End>6960</End>
        <Employees>42</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>6960</Begin>
```

```
        <End>7020</End>
        <Employees>60</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7020</Begin>
        <End>7080</End>
        <Employees>63</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7080</Begin>
        <End>7140</End>
        <Employees>29</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7140</Begin>
        <End>7200</End>
        <Employees>19</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7200</Begin>
        <End>7260</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7260</Begin>
        <End>7320</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7320</Begin>
        <End>7380</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7380</Begin>
        <End>7440</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7440</Begin>
        <End>7500</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7500</Begin>
        <End>7560</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7560</Begin>
```

```
        <End>7620</End>
        <Employees>32</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7620</Begin>
        <End>7680</End>
        <Employees>27</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7680</Begin>
        <End>7740</End>
        <Employees>54</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7740</Begin>
        <End>7800</End>
        <Employees>63</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7800</Begin>
        <End>7860</End>
        <Employees>36</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7860</Begin>
        <End>7920</End>
        <Employees>27</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7920</Begin>
        <End>7980</End>
        <Employees>47</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>7980</Begin>
        <End>8040</End>
        <Employees>41</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8040</Begin>
        <End>8100</End>
        <Employees>18</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8100</Begin>
        <End>8160</End>
        <Employees>15</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8160</Begin>
```

```
        <End>8220</End>
        <Employees>33</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8220</Begin>
        <End>8280</End>
        <Employees>29</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8280</Begin>
        <End>8340</End>
        <Employees>15</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8340</Begin>
        <End>8400</End>
        <Employees>18</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8400</Begin>
        <End>8460</End>
        <Employees>27</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8460</Begin>
        <End>8520</End>
        <Employees>21</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8520</Begin>
        <End>8580</End>
        <Employees>8</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8580</Begin>
        <End>8640</End>
        <Employees>6</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8640</Begin>
        <End>8700</End>
        <Employees>3</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8700</Begin>
        <End>8760</End>
        <Employees>3</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8760</Begin>
```

```
        <End>8820</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8820</Begin>
        <End>8880</End>
        <Employees>0</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8880</Begin>
        <End>8940</End>
        <Employees>4</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>8940</Begin>
        <End>9000</End>
        <Employees>4</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>9000</Begin>
        <End>9060</End>
        <Employees>3</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>9060</Begin>
        <End>9120</End>
        <Employees>12</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>9120</Begin>
        <End>9180</End>
        <Employees>18</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>9180</Begin>
        <End>9240</End>
        <Employees>27</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>9240</Begin>
        <End>9300</End>
        <Employees>36</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>9300</Begin>
        <End>9360</End>
        <Employees>21</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>9360</Begin>
```

```
      <End>9420</End>
      <Employees>32</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>9420</Begin>
      <End>9480</End>
      <Employees>29</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>9480</Begin>
      <End>9540</End>
      <Employees>15</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>9540</Begin>
      <End>9600</End>
      <Employees>24</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>9600</Begin>
      <End>9660</End>
      <Employees>42</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>9660</Begin>
      <End>9720</End>
      <Employees>39</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>9720</Begin>
      <End>9780</End>
      <Employees>30</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>9780</Begin>
      <End>9840</End>
      <Employees>33</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>9840</Begin>
      <End>9900</End>
      <Employees>63</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>9900</Begin>
      <End>9960</End>
      <Employees>63</Employees>
      <Exact>false</Exact>
    </Demand>
    <Demand>
      <Begin>9960</Begin>
```

```
        <End>10020</End>
        <Employees>24</Employees>
        <Exact>false</Exact>
      </Demand>
      <Demand>
        <Begin>10020</Begin>
        <End>10080</End>
        <Employees>30</Employees>
        <Exact>false</Exact>
      </Demand>
    </Demands>
  </StaffDemand>
```

Figura 7-1. Carga de trabajo que se utilizar en los ejemplos de este manual

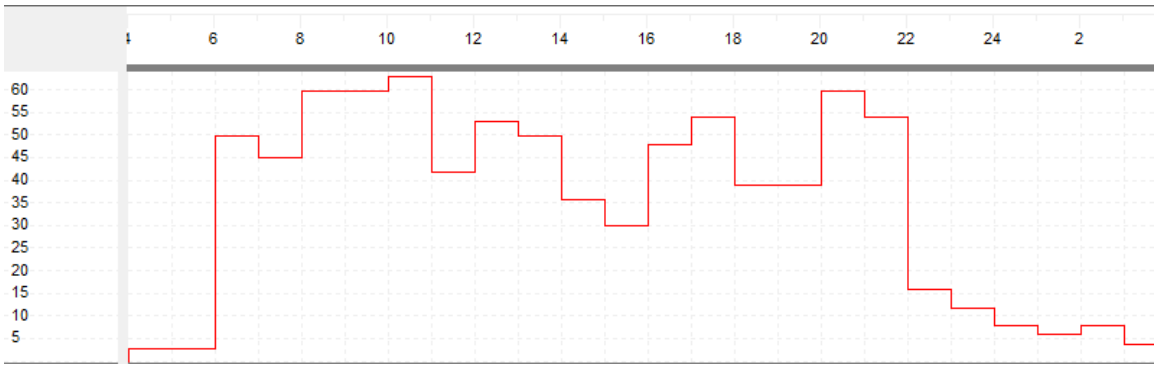


Figura 7-2. Carga de trabajo del lunes

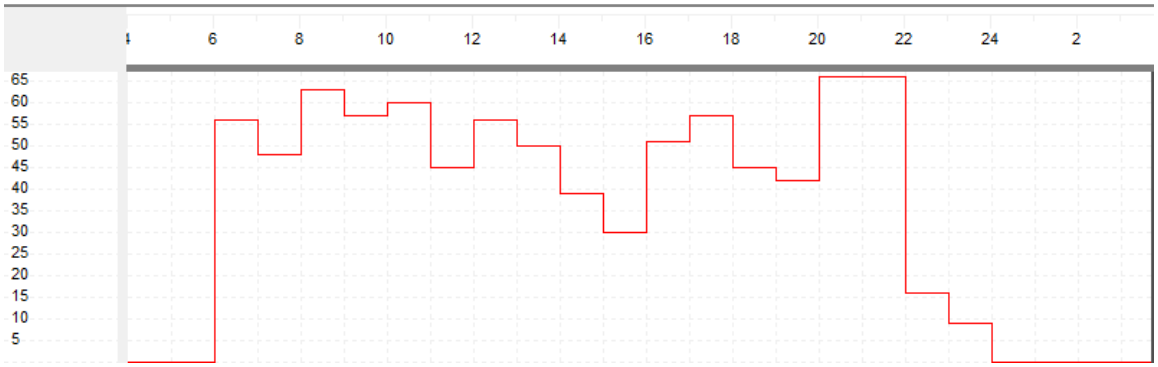


Figura 7-3. Carga de trabajo del martes

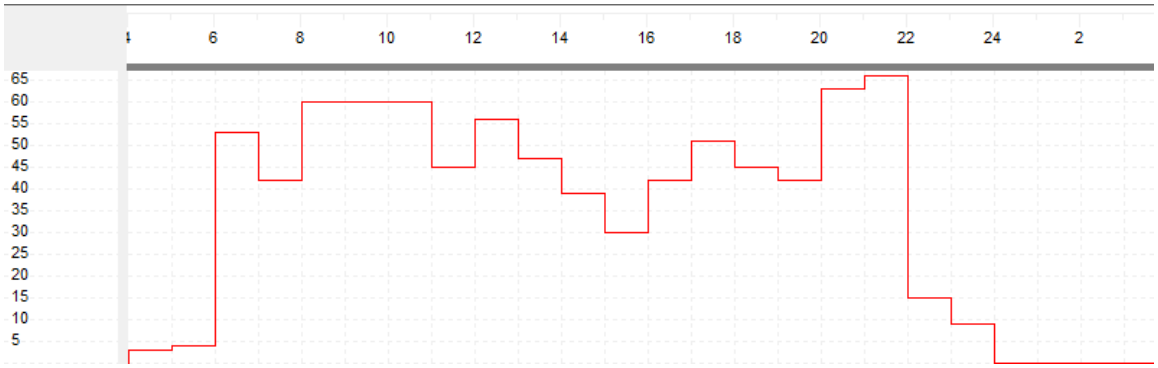


Figura 7-4. Carga de trabajo del miércoles

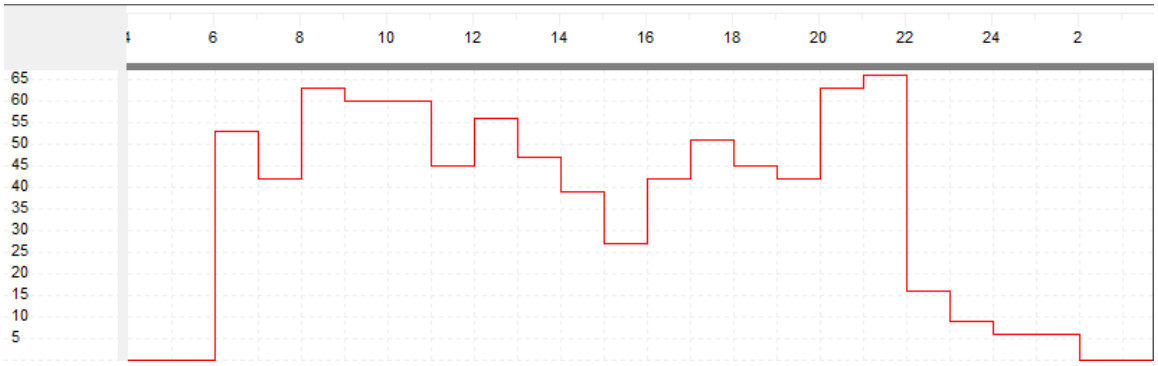


Figura 7-5. Carga de trabajo del jueves

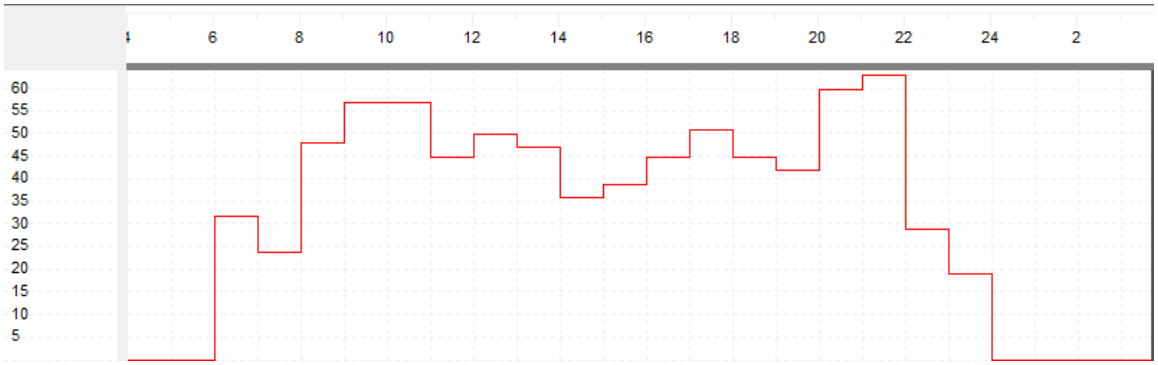


Figura 7-6. Carga de trabajo del viernes

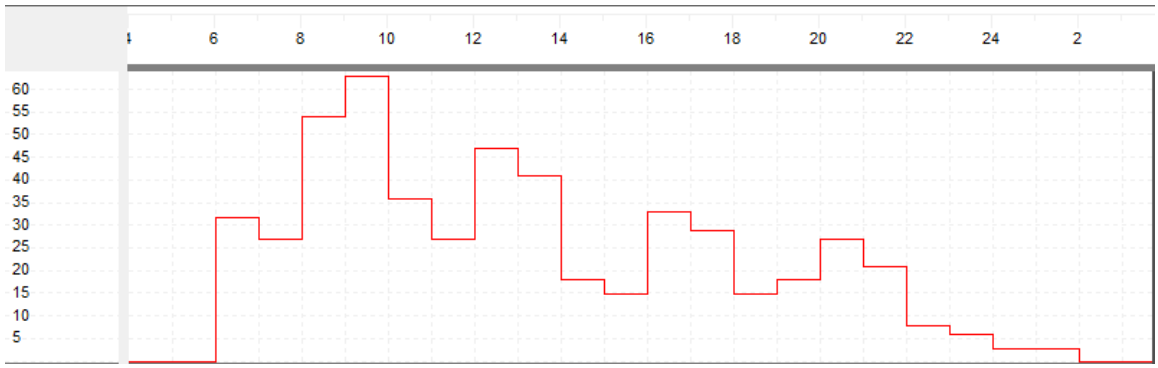


Figura 7-7. Carda de trabajo del sábado

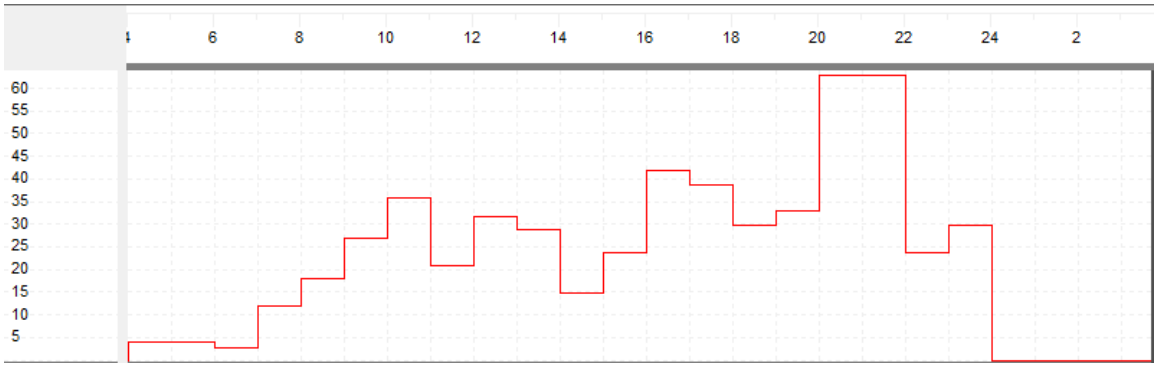


Figura 7-8. Carga de trabajo del domingo

De lunes a viernes la carga de trabajo es parecida, con pequeñas variaciones en la demanda de carga de trabajo. La carga de trabajo del sábado es decreciente, y la del domingo es creciente. Durante la mayoría de cada par de horas, se producen variaciones de mayor o menor tamaño en las demandas de carga de trabajo.

No se puede cubrir de forma exacta la demanda de carga de trabajo, por las bajadas de demanda que se produce de 07:00 a 08:00 de lunes a sábado. Si se cubre la demanda de 06:00 a 07:00, siempre habrá un exceso de carga de trabajo de 07:00 a 08:00, a menos que se permitan jornadas partidas de subjornada mínima de 01:00.