



UNIVERSIDAD DE OVIEDO



MÁSTER UNIVERSITARIO EN INGENIERÍA WEB

TRABAJO FIN DE MÁSTER

“Monitorización de entornos Oracle de forma distribuida”

Pablo Escalada Gómez

El tutor autoriza la defensa del Trabajo Fin de Máster

Fdo. D Luis Antonio Vinuesa Martínez

Oviedo, Julio de 2015

Resumen

MonitORA es un aplicación para la monitorización de bases de datos Oracle de manera distribuida. Ofrece una alternativa sencilla, multiplataforma, modular, escalable y de bajo coste a otras herramientas del mercado.

Consta de dos componentes:

- El agente, que se instala en el servidor de Base de Datos del cliente, es un servicio que recoge información del sistema mediante consultas SQL o comandos del Sistema Operativo que se ejecutan de acuerdo a una planificación configurable y los almacena temporalmente en una Base de Datos local. Posteriormente los envía al servidor para su procesamiento.
- El servidor central es el encargado de controlar todos los agentes remotos. Permite configurar las tareas que realiza cada agente y su planificación, y almacena en una Base de Datos centralizada la información recuperada de los sistemas de los clientes para que herramientas externas la consulten

Para la comunicación entre el servidor y los agentes se usan dos tecnologías diferentes:

- Servicio Web REST con JSON para la configuración de los agentes y la actualización de las tareas a realizar
- Transferencia segura de archivos XML mediante SCP para el envío al servidor de los datos extraídos por el agente.

Palabras Clave

Oracle, Monitorización, Sistema Distribuido, Cliente-Servidor, Servicio Web REST

Abstract

MonitORA is an application for monitoring Oracle databases in a distributed manner. It offers a simple, cross-platform, modular, scalable and low cost alternative to other tools in the market.

It consists of two components:

- The agent, which is installed on the client server Database, is a service that collects system information using SQL queries or operating system commands that are executed according to a configurable planning and temporarily stored in a local database. Then sends it to the server for processing.
- The central server is responsible for controlling all remote agents. It allows to configure the tasks of each agent and its planning, and stores in a centralized database the information retrieved from client systems to be accessible to external tools.

For communication between the server and the agents two different technologies are used:

- A REST Web Service with JSON for agent configuration and task updating
- Secure transfer of XML files using SCP for sending data from the agent to the server.

Keywords

Oracle, Monitorization, Distributed System, Client-Server, REST Web Service

Índice General

CAPÍTULO 1. MEMORIA DEL PROYECTO	17
1.1 RESUMEN DE LA MOTIVACIÓN, OBJETIVOS Y ALCANCE DEL PROYECTO	17
1.2 RESUMEN DE TODOS LOS ASPECTOS	18
CAPÍTULO 2. INTRODUCCIÓN	19
2.1 JUSTIFICACIÓN DEL PROYECTO	19
2.2 OBJETIVOS DEL PROYECTO	21
2.3 ESTUDIO DE LA SITUACIÓN ACTUAL.....	22
2.3.1 <i>Evaluación de Alternativas</i>	22
2.3.2 <i>Valoración de alternativas</i>	24
CAPÍTULO 3. PLANIFICACIÓN, GESTIÓN DE RIESGOS Y RESUMEN DE PRESUPUESTOS	25
3.1 PLANIFICACIÓN	25
3.1.1 <i>Listado tareas</i>	25
3.1.2 <i>Resumen</i>	27
3.2 RESUMEN DEL PRESUPUESTO.....	29
3.3 GESTIÓN DE RIESGOS	30
3.3.1 <i>Identificación de Riesgos</i>	30
3.3.2 <i>Plan de Gestión de Riesgos</i>	32
CAPÍTULO 4. ANÁLISIS.....	37
4.1 DEFINICIÓN DEL SISTEMA.....	37
4.1.1 <i>Determinación del Alcance del Sistema</i>	37
4.2 REQUISITOS DEL SISTEMA	39
4.2.1 <i>Obtención de los Requisitos del Sistema</i>	39
4.2.2 <i>Identificación de Actores del Sistema</i>	39
4.2.3 <i>Especificación de Casos de Uso</i>	40
4.3 IDENTIFICACIÓN DE LOS SUBSISTEMAS EN LA FASE DE ANÁLISIS.....	43
4.3.1 <i>Descripción de los Subsistemas</i>	43
4.3.2 <i>Descripción de los Interfaces entre Subsistemas</i>	43
4.4 DIAGRAMA DE CLASES PRELIMINAR DEL ANÁLISIS	44
4.4.1 <i>Diagrama de Clases</i>	44
4.4.2 <i>Descripción de las Clases</i>	44
4.5 ANÁLISIS DE CASOS DE USO Y ESCENARIOS	47
4.5.1 <i>Monitorizar</i>	47
4.5.2 <i>Ping</i>	47
4.5.3 <i>Recibir agente</i>	47
4.5.4 <i>Actualizar agente</i>	48
4.5.5 <i>Planificar tarea</i>	48
4.5.6 <i>Ejecutar tarea</i>	49
4.5.7 <i>Enviar resultados</i>	49
4.5.8 <i>Crear snapshot</i>	50
4.5.9 <i>Enviar snapshot</i>	50
4.5.10 <i>Enviar ficheros</i>	51
4.5.11 <i>Responder ping</i>	51

Monitorización de entornos Oracle de forma distribuida

4.5.12	<i>Enviar agente</i>	51
4.5.13	<i>Configurar agente</i>	52
4.5.14	<i>Recibir snapshot</i>	52
4.5.15	<i>Recibir ficheros</i>	53
4.6	ESPECIFICACIÓN DEL PLAN DE PRUEBAS.....	54
CAPÍTULO 5. DISEÑO DEL SISTEMA.....		59
5.1	ARQUITECTURA DEL SISTEMA	59
5.1.1	<i>Diagramas de Paquetes</i>	59
5.1.2	<i>Diagramas de Componentes</i>	61
5.1.3	<i>Diagramas de Despliegue</i>	62
5.2	DISEÑO DE CLASES.....	63
5.2.1	<i>Arquitectura del servidor</i>	63
5.2.2	<i>Arquitectura del agente</i>	66
5.2.3	<i>Diagrama de Clases</i>	67
5.3	DIAGRAMAS DE INTERACCIÓN Y ESTADOS	69
5.3.1	<i>Caso de Uso Recibir agente, Actualizar Agente y Planificar</i>	69
5.3.2	<i>Caso de Uso Ejecutar tarea</i>	70
5.3.3	<i>Caso de Uso Crear snapshot</i>	71
5.4	DISEÑO DE LA BASE DE DATOS	72
5.4.1	<i>Descripción del SGBD Usado</i>	72
5.4.2	<i>Integración del SGBD en Nuestro Sistema</i>	72
5.4.3	<i>Diagrama E-R</i>	73
5.5	ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS	74
5.5.1	<i>Pruebas Unitarias</i>	74
5.5.2	<i>Pruebas de Integración y del Sistema</i>	74
CAPÍTULO 6. IMPLEMENTACIÓN DEL SISTEMA.....		79
6.1	LENGUAJES DE PROGRAMACIÓN	79
6.2	HERRAMIENTAS Y PROGRAMAS USADOS PARA EL DESARROLLO.....	80
6.2.1	<i>Bibliotecas de funciones</i>	80
6.2.2	<i>Programas y herramientas</i>	81
6.3	CREACIÓN DEL SISTEMA	82
6.3.1	<i>Problemas Encontrados</i>	82
CAPÍTULO 7. DESARROLLO DE LAS PRUEBAS.....		83
7.1	PRUEBAS UNITARIAS	83
7.2	PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA	85
CAPÍTULO 8. MANUALES DEL SISTEMA		89
8.1	MANUAL DE INSTALACIÓN	89
8.1.1	<i>Requisitos previos</i>	89
8.1.2	<i>Servidor</i>	89
8.1.3	<i>Agente</i>	91
8.2	MANUAL DE EJECUCIÓN	93
8.2.1	<i>Agente</i>	93
CAPÍTULO 9. CONCLUSIONES Y AMPLIACIONES		94
9.1	CONCLUSIONES	94
9.2	AMPLIACIONES.....	95
9.2.1	<i>Generación de informes</i>	95

9.2.2	<i>Envío de alertas</i>	95
9.2.3	<i>Ampliación del servicio web</i>	95
CAPÍTULO 10.	PRESUPUESTO	97
10.1	GASTOS	97
10.2	PRESUPUESTO CLIENTE	98
CAPÍTULO 11.	REFERENCIAS BIBLIOGRÁFICAS	99
CAPÍTULO 12.	APÉNDICES	101
12.1	CONTENIDO ENTREGADO EN ANEXOS.....	101
12.1.1	<i>Contenidos</i>	101

Índice de Figuras

Ilustración 1 Solarwinds Database Monitor	22
Ilustración 2 Lab128 - Ventana principal.....	23
Ilustración 3 MyOra - Ventana principal	24
Tabla 1 Matriz de Probabilidades e Impacto.....	34
Tabla 2 Requisitos funcionales	39
Tabla 3 Requisitos no funcionales	39
Ilustración 4 Diagrama de Casos de Uso	40
Ilustración 5 - Diagrama de clases de análisis	44
Ilustración 6 Diagrama de paquetes	59
Ilustración 7 Diagrama de Componentes	61
Ilustración 8 Diagrama de despliegue.....	62
Ilustración 9 Diagrama de arquitectura del Servidor.....	63
Ilustración 10 Servidor – Capa de modelo de datos	64
Ilustración 11 Servidor - Capa de persistencia	64
Ilustración 12 Servidor – Capa de negocio	65
Ilustración 13 Servidor - Capa de utilidades.....	65
Ilustración 14 Servidor - Capa de presentación.....	65
Ilustración 15 Diagrama de arquitectura del cliente	66
Ilustración 16 Agente - Capa de negocio.....	66
Ilustración 17 Diagrama de clases del agente	67
Ilustración 18 Diagrama de clases del servidor	68
Ilustración 19 Diagrama de secuencia de Recibir Agente, Actualizar Agente y Planificar.....	69
Ilustración 20 Diagrama de secuencia de Ejecutar Tarea	70
Ilustración 21 Diagrama de secuencia de Crear Snapshot	71
Ilustración 22 Diagrama Entidad-Relación del modelo de datos	73
Ilustración 23 Pruebas unitarias del Servidor.....	83

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

La intención de este proyecto es crear una herramienta de monitorización de bases de datos distribuida y ofrecer una alternativa sencilla, multiplataforma, modular, escalable y de bajo coste a otras herramientas del mercado.

Monitora permite a clientes de pequeño y mediano tamaño monitorizar sus sistemas a bajo coste, de una manera sencilla y segura, optimizando recursos e incrementando la confianza en la robustez de sus sistemas.

En concreto, este proyecto abarca la implementación de dos módulos:

- Agente: Se despliega en el servidor de Bases de Datos Oracle del cliente. Se conecta al servidor mediante un servicio web, ejecuta las tareas planificadas, almacena los resultados en una Base de Datos local y finalmente los vuelca a XML y envía al servidor mediante una conexión segura SCP.
- El servidor central, que contiene una Base de Datos centralizada y un servidor web con un servicio web REST en JSON que hace de controlador de los distintos agentes.

La explotación de los datos obtenidos, la visualización o la creación de eventos automatizados en respuesta a determinadas condiciones quedan fuera del alcance del proyecto.

1.2 Resumen de Todos los Aspectos

La estructura que se seguirá en este documento es la siguiente:

En el segundo capítulo se verá una introducción más detallada sobre los objetivos y razones que motivaron este proyecto.

En el tercer capítulo se detallarán los aspectos relacionados con la planificación y la gestión de riesgos. También se mostrará un resumen del presupuesto del cliente.

En el capítulo cuarto se empezarán a trazar las líneas generales del proyecto. Se fijarán los requisitos y se desglosarán los distintos subsistemas que componen el proyecto. Se definirán los casos de uso más típicos y las pruebas que permitirán comprobar si se cumple lo especificado.

En el capítulo quinto se desarrollarán en más detalle las ideas esbozadas en el capítulo anterior y se relacionan con aspectos del desarrollo. Se define la arquitectura general y el funcionamiento de los diferentes módulos.

En el capítulo sexto se explican las herramientas y tecnologías empleadas, además de una descripción detallada del proceso de desarrollo.

En el capítulo séptimo se analizan las pruebas desarrolladas para verificar que lo implementado se ajusta a la especificación.

Los manuales del sistema se detallan en el capítulo ocho.

En el capítulo nueve se detallan las conclusiones recogidas a lo largo del proceso de desarrollo y las ampliaciones futuras.

En el décimo capítulo se recogen las referencias bibliográficas consultadas durante la realización del proyecto.

En el capítulo once se agrupan los apéndices

Capítulo 2. Introducción

2.1 Justificación del Proyecto

Actualmente existen Bases de Datos en cualquier sistema y para cualquier propósito, desde los sistemas corporativos con decenas de nodos y presupuestos millonarios a servidores independientes en todo tipo de instituciones públicas y PYMES.

El mantenimiento de estos sistemas es complejo y caro en el mejor de los casos. Requiere de técnicos cualificados, acceso a las instalaciones del cliente o acceso remoto a los servidores, implementación y monitorización de políticas de backup.

Monitora ofrece varias ventajas:

1. **Remoto:** Permite monitorizar múltiples clientes desde un mismo lugar, sin desplazamientos y con personal reducido.
2. **Distribuido:** El trabajo se reparte entre el servidor central y los agentes instalados en los clientes, por lo que un servidor puede dar servicio a múltiples agentes, ahorrando costes y cada servidor de BBDD solo requiere la instalación del agente, lo que evita sobrecargar el servidor.
3. **Seguridad:** Las comunicaciones cliente-servidor se inician desde el cliente, con lo que la configuración de los Firewall del cliente es más sencilla. La información sensible se transmite de forma cifrada mediante SCP
4. **Costes:** Monitora es un proyecto open source y se puede desplegar sobre cualquier plataforma sin coste por licencias.
5. **Sencillo:** En el cliente solo es necesario instalar una aplicación, ligera y autocontenido, lo que simplifica el despliegue y problemas de configuración o rendimiento. El soporte para consultas sobre la Base de Datos del cliente y la ejecución de comandos del Sistema Operativo del host permite:
 - Limitar la complejidad de la aplicación al ofrecer un reducido número de funcionalidades
 - Mayor seguridad, ya que permite auditar las tareas que se realizan sobre el cliente, valorando los riesgos y limitando el acceso a ciertas tablas o comandos

Además de estos objetivos funcionales, Monitora permite ampliar los conocimientos sobre diversos aspectos de las tecnologías web:

- Persistencia y Mapeo relacional (ORM)
- Servicios web y transmisión de modelos de datos complejos

- Integración de diferentes tecnologías, modularidad y separación en capas
- Aplicar un conjunto de buenas prácticas relativas al manejo de proyectos y desarrollo ágil

2.2 Objetivos del Proyecto

El objetivo del proyecto es crear una herramienta de monitorización de bases de datos distribuida y ofrecer una alternativa sencilla, multiplataforma, modular, escalable y de bajo coste a otras herramientas del mercado.

Más concretamente, los objetivos son:

- **Modelo de datos:** Diseñar un modelo de datos que permita capturar la máxima información del sistema y facilite su posterior tratamiento.
- **Servidor:** Implementar un nodo central que controla a los agentes, permite la configuración de las tareas que estos van a realizar en los sistemas del cliente y guarda la toda información recibida de los agentes en un servidor de Bases de Datos. Las tareas que se pueden realizar son de dos tipos: consultas a la Base de Datos o comandos del Sistema Operativo
- **Agente:** Aplicación instalada en el servidor de Bases de Datos del cliente que ejecuta las tareas configuradas en el servidor en el momento indicado, las guarda en una Base de Datos local para su posterior tratamiento y envía periódicamente vuelca los resultados a un fichero XML y lo envía al servidor mediante SCP
- **Comunicaciones:** Especificar un protocolo de comunicaciones entre el agente y el servidor e implementar un servicio web REST en JSON que lo soporte.

Además de estos objetivos funcionales, se añaden otros.

- **Robustez:** Se busca crear una base sólida, sobre la cual poder añadir servicios de más nivel, por tanto se utilizarán ficheros de log lo largo de todas las capas del proyecto, una extensa batería de pruebas y en lo posible solo se dependerá de tecnologías maduras.
- **Diseño modular:** Permite añadir nuevas funcionalidades de manera sencilla, aislar los componentes y realizar pruebas de forma más sencilla

2.3 Estudio de la Situación Actual

2.3.1 Evaluación de Alternativas

Existen numerosas herramientas que permiten monitorizar Bases de Datos Oracle, desde simples scripts a paquetes empresariales que soportan múltiples SGDB o incluso la monitorización de los servidores virtuales y entornos Cloud.

La referencia es Oracle Enterprise Manager (OEM), pero su alto coste y su complejidad lo desaconsejan en muchos entornos.

Dentro de todas estas opciones monitora se debe comparar con otras herramientas open source o de bajo coste, multiplataforma, enfocadas a obtener una visión general del servidor.

2.3.1.1 SolarWinds Database Monitor

<http://www.solarwinds.com/database-monitor.aspx>

- ✓ Información en tiempo real
- ✓ Soporta múltiples SGDB (Oracle, DB2, SQL Server, ...)
- ✓ Soporta VMware
- ✓ No instala agente en el servidor
- ✓ Interface web
- ✓ Multiplataforma
- ✗ Versión gratuita limitada a 20 servidores
- ✗ Coste

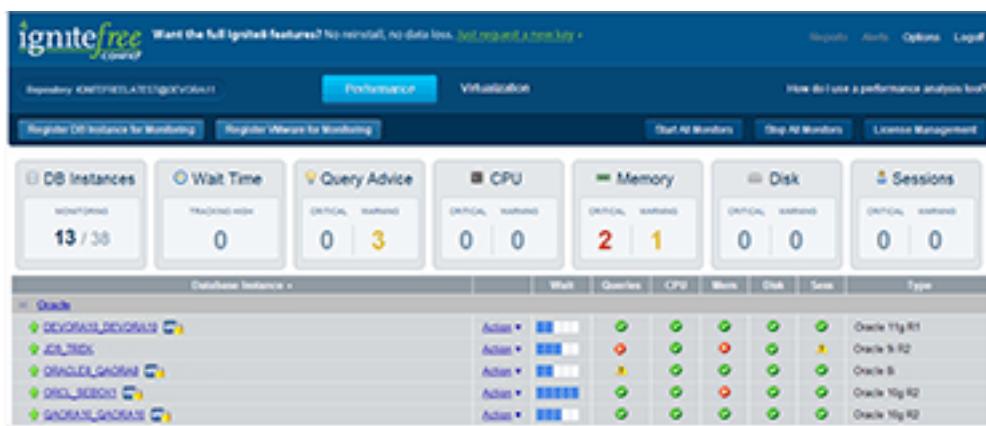


Ilustración 1 Solarwinds Database Monitor

2.3.1.2 Lab128

<http://www.lab128.com/>

- ✓ Información en tiempo real
- ✓ No instala agente en el servidor

- ✓ Interface gráfico nativo
- ✓ No necesita Oracle Diagnostic Pack ni Oracle Enterprise Manager
- ✗ Solo soporta Windows
- ✗ Versión gratuita limitada con soporte para 2 instancias
- ✗ Coste 600\$/mes

Herramienta de monitorización de Bases de Datos Oracle enfocada a obtener datos en tiempo real y optimización.

No es necesario ningún agente en el servidor, usa la biblioteca Oracle Call Interface para recuperar datos de varias vistas del sistema y del Active Session History.

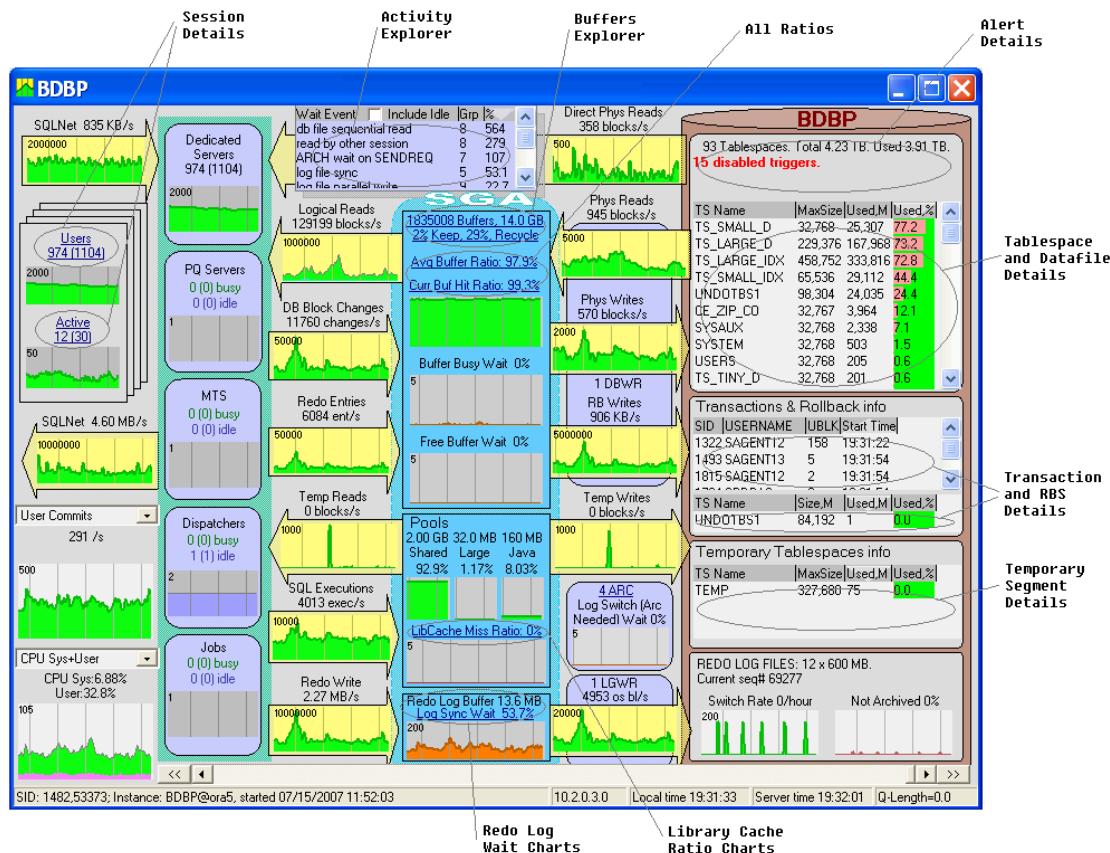


Ilustración 2 Lab128 - Ventana principal

2.3.1.3 MyOra

<http://www.myorasql.com>

- ✓ Permite editar SQL y ejecutar consultas
- ✓ Permite administrar Bases de Datos
- ✓ No necesita agente en el servidor, accede a través de JDBC sin necesidad de Oracle Client
- ✓ Interface gráfico nativo
- ✓ Gratuita
- ✗ Solo permite 100 conexiones simultáneas

- ✗ Solo soporta Windows

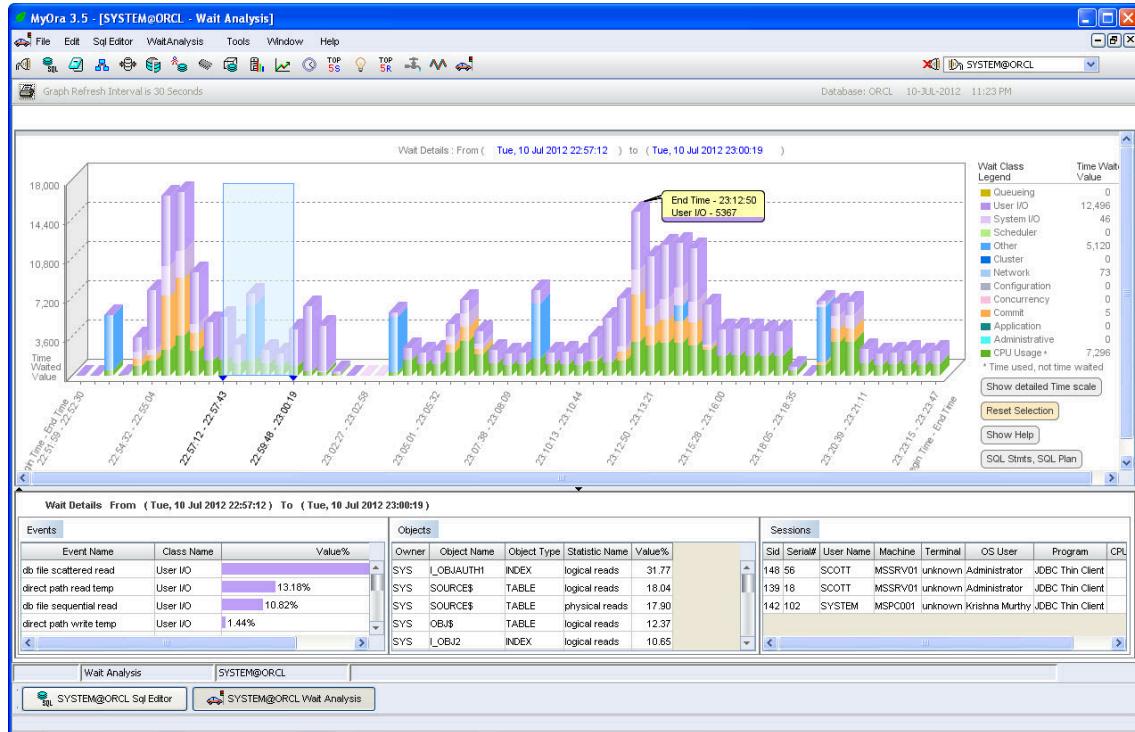


Ilustración 3 MyOra - Ventana principal

2.3.2 Valoración de alternativas

Frente a estas alternativas Monitora tiene varias ventajas e inconvenientes:

- ✗ Carece de interface gráfica. Esta característica, por su envergadura está totalmente fuera del alcance del Proyecto y ya podría considerarse un PFM en sí misma.
- ✗ No captura datos en tiempo real, solo permite ver los datos históricos.
- ✗ Instala un agente en el servidor, con los posibles problemas de rendimiento que ocasiona.
- ✓ La instalación de un agente permite una arquitectura distribuida, liberando al repositorio central de carga de trabajo y permitiendo monitorizar más sistemas simultáneamente.
- ✓ Accede mediante JDBC a la Base de Datos. No necesita usar Oracle Diagnostic Pack ni Oracle Client por lo que se limitan las dependencias con versiones concretas de la BBDD.
- ✓ Permite ejecutar comandos del sistema operativo, con lo que se tiene acceso a virtualmente cualquier herramienta que se necesite o a los logs del Sistema Operativo.
- ✓ Multiplataforma
- ✓ Open source
- ✓ No sobrecarga la red, incluso puede funcionar sin conexión durante largos periodos

Capítulo 3. Planificación, Gestión de Riesgos y Resumen de Presupuestos

3.1 Planificación

3.1.1 Listado tareas

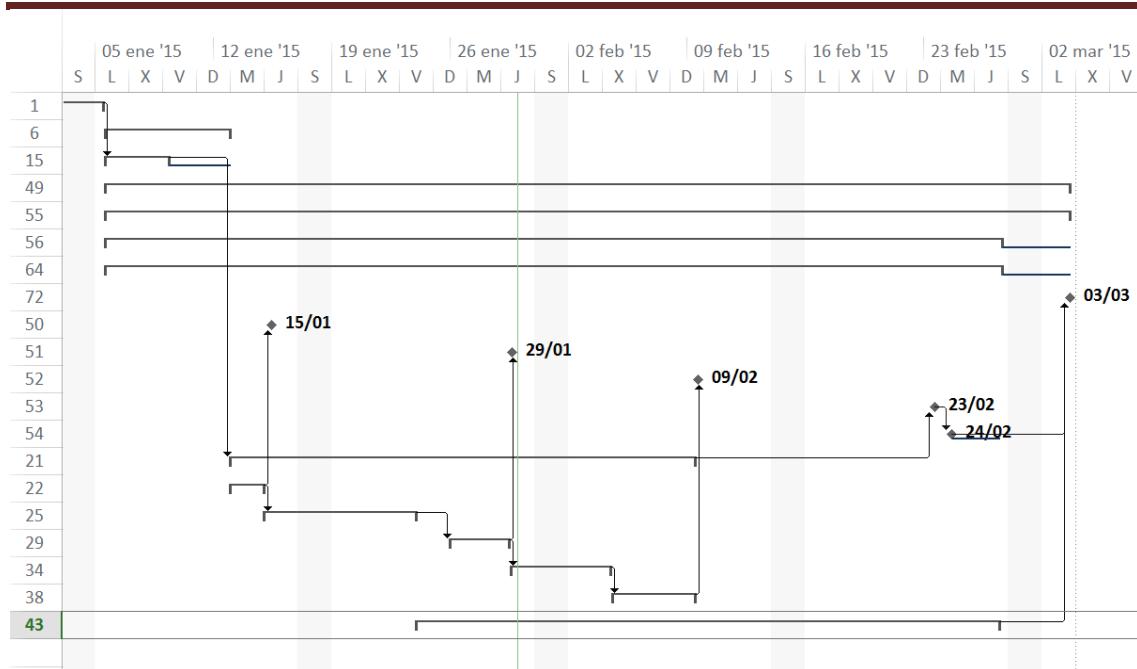
Nombre de tarea	Nivel de esquema	Duración	Comienzo	Nombres de los recursos
Análisis	11,5 días	vie 02/01/15		
Idea general	22 horas	vie 02/01/15	Analista[50%];Jefe de Proyecto[20%];Sala reuniones[1]	
Toma requisitos	24 horas	vie 02/01/15	Analista[50%]	
Opciones y alternativas	24 horas	vie 02/01/15	Analista[50%]	
Bibliotecas funciones	24 horas	vie 02/01/15	Analista[50%]	
Diseño	15,5 días	lun 05/01/15		
Definición protocolo transmisión	24 horas	lun 05/01/15	Analista[50%];Programador 1	
Modularización	21,75 días	lun 05/01/15		
Módulo recogida datos	34 horas	lun 05/01/15	Analista[50%];Programador 1	
Módulo transmisión	32 horas	mar 06/01/15	Analista[50%];Programador 1	
Módulo control	32 horas	mar 06/01/15	Analista[50%];Programador 1	
Servidor	32 horas	mar 06/01/15	Analista[50%];Programador 1	
Cliente web	32 horas	mié 07/01/15	Analista[50%];Programador 1	
Definición repositorio	22 días	vie 09/01/15	Analista[50%];Programador 1	
Configuración entorno	13,63 días	lun 05/01/15		
Máquinas virtuales	24 horas	lun 05/01/15	Administrador Sistemas	
Repositorio remoto	24 horas	lun 05/01/15	Administrador Sistemas;Servidor Repositorio[1]	
Software base	22 horas	mar 06/01/15	Administrador Sistemas;Servidor Centro Control[1];Servidor Repositorio[1]	
Carga de datos iniciales BD	24 horas	mar 06/01/15	Administrador Sistemas;Servidor Repositorio[1]	
Herramientas desarrollo	21 hora	vie 09/01/15	Administrador Sistemas	
Entrega	141,25 días	lun 05/01/15		
Control	241,25 días	lun 05/01/15		
Riesgos	339,13 días	lun 05/01/15		
Análisis	41 hora	lun 05/01/15	Jefe de Proyecto[20%]	
Diseño	41 hora	mar 13/01/15	Jefe de Proyecto[20%]	
Milestone 1	41 hora	jue 15/01/15	Jefe de Proyecto[20%]	
Milestone 2	41 hora	jue 29/01/15	Jefe de Proyecto[20%]	
Implementación	41 hora	lun 09/02/15	Jefe de Proyecto[20%]	
Milestone 3	41 hora	lun 09/02/15	Jefe de Proyecto[20%]	
Documentación	41 hora	vie 27/02/15	Jefe de Proyecto[20%]	
Calidad	339,13 días	lun 05/01/15		
Análisis	41 hora	lun 05/01/15	Jefe de Proyecto[20%]	
Diseño	41 hora	mar 13/01/15	Jefe de Proyecto[20%]	
Milestone 1	41 hora	jue 15/01/15	Jefe de Proyecto[20%]	
Milestone 2	41 hora	jue 29/01/15	Jefe de Proyecto[20%]	

Monitorización de entornos Oracle de forma distribuida | Planificación, Gestión de Riesgos y Resumen de Presupuestos

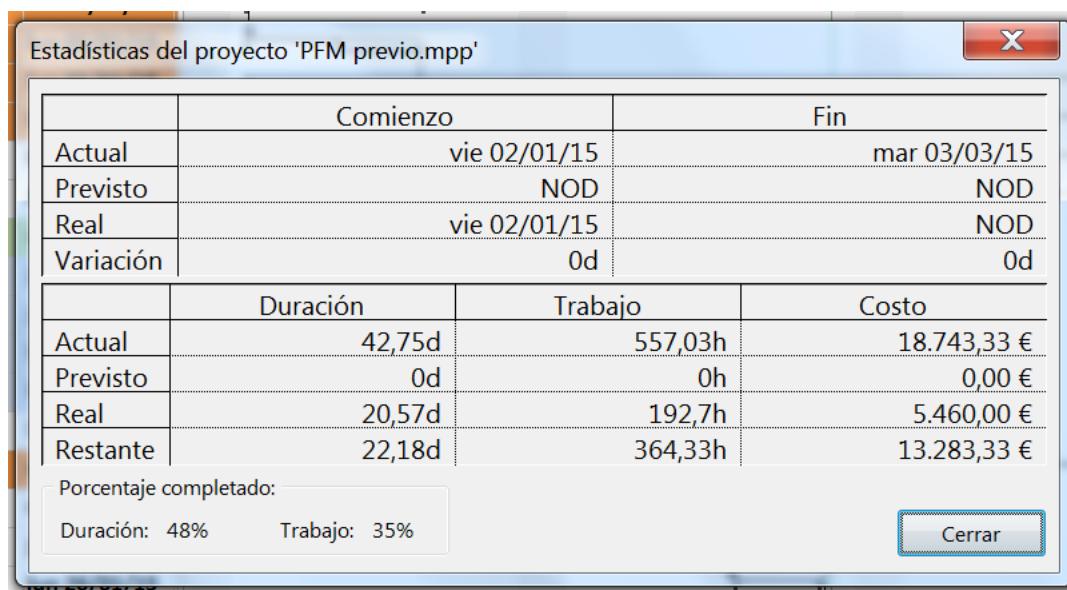
Nombre de tarea	Nivel de esquema	Duración	Comienzo	Nombres de los recursos
Implementación	41 hora	lun 09/02/15	Jefe de Proyecto[20%]	
Milestone 3	41 hora	lun 09/02/15	Jefe de Proyecto[20%]	
Documentación	41 hora	vie 27/02/15	Jefe de Proyecto[20%]	
Finalización	32 horas	mar 03/03/15	Jefe de Proyecto[20%];Salón actos[1];Servidor Centro Control[1];Servidor Repositorio[1]	
Milestone 1	21 hora	jue 15/01/15	Analista[50%];Sala reuniones[1]	
Milestone 2	21 hora	jue 29/01/15	Analista[50%];Sala reuniones[1]	
Milestone3	21 hora	lun 09/02/15	Programador 1;Analista[50%];Sala reuniones[1];Servidor Centro Control[1];Servidor Repositorio[1]	
Revisión final	21 hora	lun 23/02/15	Analista[50%];Sala reuniones[1];Servidor Centro Control[1];Servidor Repositorio[1]	
Despliegue	21 día	lun 23/02/15	Administrador Sistemas;Analista[50%];Programador 1;Servidor Centro Control[1];Servidor Repositorio[1]	
Implementación	119,5 días	mar 13/01/15		
Módulo recogida datos	22 días	mar 13/01/15		
Acceso nativo	32 días	mar 13/01/15	Programador 1	
Persistencia temporal	31 día	mar 13/01/15	Programador 2	
Módulo transmisión	27 días	jue 15/01/15		
Canal lectura	33 días	jue 15/01/15	Programador 2	
Reenvío y control errores	32 días	mar 20/01/15	Programador 1	
Canal push	32 días	jue 22/01/15	Programador 1	
Módulo control	23,5 días	lun 26/01/15		
Loader	33 días	lun 26/01/15	Programador 1	
Información sistema	31 día	lun 26/01/15	Programador 2	
Planificador	31 día	mar 27/01/15	Programador 2	
Consola administración	31 día	mié 28/01/15	Programador 2	
Servidor	24 días	jue 29/01/15	Servidor Centro Control[1];Servidor Repositorio[1]	
Buzón entrada	31 día	jue 29/01/15	Programador 2;Servidor Centro Control[1]	
Buzón salida	31 día	vie 30/01/15	Programador 1;Servidor Centro Control[1]	
Persistencia repositorio	32 días	lun 02/02/15	Programador 2;Servidor Centro Control[1];Servidor Repositorio[1]	
Cliente web	23 días	mié 04/02/15	Servidor Centro Control[1];Servidor Repositorio[1]	
Persistencia	31 día	mié 04/02/15	Programador 2;Servidor Centro Control[1];Servidor Repositorio[1]	
Modelo	31 día	jue 05/02/15	Programador 2;Servidor Centro Control[1];Servidor Repositorio[1]	
Negocio	31 día	vie 06/02/15	Programador 2;Servidor Centro Control[1];Servidor Repositorio[1]	
Presentación	30 días	lun 09/02/15	Programador 2;Servidor Centro Control[1];Servidor Repositorio[1]	
Documentación	124,5 días	vie 23/01/15		
Protocolo transmisión	24 horas	vie 23/01/15	Programador 1;Programador 2	
Base de datos	21 día	jue 29/01/15	Programador 1;Programador 2	
Memoria	210 días	lun 09/02/15	Analista[50%];Programador 1;Programador 2	
Javadoc final	22 días	lun 23/02/15	Programador 1;Programador 2	
Diagramas	22 días	mié 25/02/15	Programador 1;Programador 2	

3.1.2 Resumen

	Mod de tarea	Nombre de tarea	Nivel de	Duració	Comienzo
1		▶ Análisis	1	1,5 días	vie 02/01/15
6	✓	▶ Diseño	1	5,5 días	lun 05/01/15
15		▶ Configuración entorno	1	3,63 días	lun 05/01/15
49		▶ Entrega	1	41,25 días	lun 05/01/15
55		▶ Control	2	41,25 días	lun 05/01/15
56		▶ Riesgos	3	39,13 días	lun 05/01/15
64		▶ Calidad	3	39,13 días	lun 05/01/15
72		Finalización	3	2 horas	mar 03/03/15
50	✓	Milestone 1	2	1 hora	jue 15/01/15
51		Milestone 2	2	1 hora	jue 29/01/15
52		Milestone3	2	1 hora	lun 09/02/15
53		Revisión final	2	1 hora	lun 23/02/15
54		Despliegue	2	1 día	lun 23/02/15
21		▶ Implementación	1	19,5 días	mar 13/01/15
22	✓	▶ Módulo recogida datos	2	2 días	mar 13/01/15
25	✓	▶ Módulo transmisión	2	7 días	jue 15/01/15
29		▶ Módulo control	2	3,5 días	lun 26/01/15
34		▶ Servidor	2	4 días	jue 29/01/15
38		▶ Cliente web	2	3 días	mié 04/02/15
43		▶ Documentación	1	24,5 días	vie 23/01/15



- Número de tareas: 72
- Duración de la tarea mayor: 10 días
- Duración de la tarea menor: 1 hora
- Nº de tareas críticas: 10
- Nº máximo de tareas críticas simultaneas: 2
- Media de holguras: 8,29 días
- Presupuesto total: 16633,33 €
- Duración total del proyecto: 43 días
- Nº de recursos totales: 9
- Nº de recursos personales: 5



3.2 Resumen del Presupuesto

Item	Concepto	Unidades	Importe	Total
1	Software			
1.1	Servidor	42	195,50 €	8.211,00 €
1.2	Agente	72	195,50 €	14.076,00 €
	21% IVA			22.287,00 €
				4.680,27 €
Total				26.967,27 €

3.3 Gestión de Riesgos

3.3.1 Identificación de Riesgos

ID	Nombre	Responsable	Probabilidad	Impacto				P a c 0,5	Priorización	Respuesta
				Presup.	Planific.	Alcance	Calidad			
1	Mala planificación temporal	Todos	Muy Alta		Crítico	Alto	Muy Bajo	0,81	Rojo	Replanificar
2	Aplicación consume muchos recursos	Alumno	Media		Bajo	Bajo	Muy Bajo	0,08	Verde	Optimizar los procesos/módulos críticos
3	Cambios en el alcance del proyecto.	Todos	Alta		Crítico	Crítico	Muy Bajo	0,63	Rojo	Replanificar
4	Cambios en las prioridades	Todos	Baja		Medio	Bajo	Muy Bajo	0,09	Verde	Replanificar
5	Cliente que no está lo suficientemente disponible para proporcionar información precisa de los requerimientos	Director	Media		Medio	Medio	Medio	0,15	Verde	Mejorar la comunicación con el Director de Proyecto
6	Pérdida de datos	Alumno	Muy Baja		Alto	Bajo	Muy Bajo	0,06	Verde	Copias de seguridad, CVS, virtualización
7	Demasiada implicación del cliente en el proceso	Director	Muy Baja		Medio	Medio	Muy Bajo	0,03	Verde	Mejorar la comunicación con el Director de Proyecto
8	Demasiada poca implicación del cliente en el proceso	Director	Baja		Muy Bajo	Muy Bajo	Medio	0,09	Verde	Mejorar la comunicación con el Director de Proyecto
9	Demoras en la toma de decisiones	Todos	Baja		Medio	Medio	Muy Bajo	0,09	Verde	Mejorar la comunicación con el Director de Proyecto
10	Durante el proyecto aparece un cambio de tecnología		Media		Alto	Alto	Alto	0,28	Verde	Diseño modular, buen plan de pruebas
11	El cliente rechaza el proyecto		Muy Baja		Crítico	Crítico	Muy Bajo	0,09	Verde	Mejorar la comunicación con el

ID	Nombre	Responsable	Probabilidad	Impacto				P	A	C	0,5	Priorización	Respuesta
				Presup.	Planific.	Alcance	Calidad						
	porque no es lo que buscaba												Director de Proyecto. Posponer la lectura del Proyecto
12	Falta de experiencia con la tecnología	Alumno	Alta		Alto	Alto	Alto	0,39					Detección temprana y formación
13	El proyecto es mas complejo de lo que parecia al principio, con lo cual se requieren cambios	Todos	Media		Alto	Alto	Muy Bajo	0,28					Detección temprana, reutilización de código, formación, adaptación del alcance, replanificación
15	Existen aspectos de seguridad no considerados.	Alumno	Baja		Medio	Medio	Crítico	0,27					Reutilización de código, buen plan de pruebas, rediseño
16	Falta de formación en las herramientas.	Alumno	Muy Baja		Bajo	Muy Bajo	Bajo	0,02					Detección temprana y formación
17	Fallos en la infraestructura y servicios externos.	Todos	Baja		Muy Bajo	Muy Bajo	Muy Bajo	0,02					Virtualización y redundancia de los equipos
18	Fallos en la planificación del diseño	Alumno	Media		Medio	Medio	Muy Bajo	0,15					Replanificar
19	Fecha de entrega del proyecto ajustada o fuera de plazo.	Alumno	Alta		Crítico	Alto	Muy Bajo	0,63					Replanificar, adaptación del alcance. Posponer la lectura del Proyecto
20	Incumplimiento de compromisos (reuniones individuales, aporte de documentos, etc.)	Todos	Baja		Bajo	Muy Bajo	Muy Bajo	0,05					Mejorar la comunicación con el Director de Proyecto, replanificar

3.3.2 Plan de Gestión de Riesgos

3.3.2.1 Metodología:

Este Plan de Gestión de Riesgos se aplica a todos los riesgos identificados del Proyecto Fin de Máster.

Teniendo en cuenta que no se desarrolla en un entorno empresarial y que por tanto el equipo del proyecto se reduce al alumno con el apoyo puntual del Director de Proyecto, la responsabilidad de cualquier tarea de este plan recae sobre el alumno.

La identificación de los riesgos se basa en la percepción del alumno y esta puede cambiar a lo largo del proyecto.

La metodología a usar se basa en los puntos principales de PMBOK:

- Planificar la gestión de riesgos: En este punto se adapta la metodología original al contexto concreto del PFM. Por su duración y alcance se simplificará al máximo con el fin de no agilizar su aplicación.
- Identificación de riesgos: Se anotan todos los riesgos. En lugar de brainstorming se buscará en proyectos anteriores los riesgos identificados para considerar incluirlos en el actual. En el contexto del PFM se obviarán los riesgos sobre el presupuesto y se tendrán en cuenta otros factores como la calidad.
- Análisis de los riesgos: Se categorizan y se clasificarán según su probabilidad e impacto. Se valorará su importancia y se decidirá cuales merecen un seguimiento detallado.
- Planificación de la respuesta: Para los riesgos más importantes se detallarán las medidas a tomar para su control, los factores de riesgo, los indicadores a vigilar y los planes de contingencia.
- Monitorización y control: Se procederá a evaluar los indicadores de acuerdo a la planificación de cada riesgo y se actualizará el plan de riesgos de acuerdo a la evolución del proyecto.

3.3.2.2 Herramientas y tecnologías

Las técnicas que se van a usar para gestionar los riesgos son:

Evaluaciones regulares.

Regularmente todos los indicadores se evaluarán de acuerdo a lo establecido en las hojas de riesgos.

Reuniones

Se tratarán los riesgos más importantes y sus efectos en las reuniones con el Director de Proyecto. Al menos en cada uno de los hitos intermedios (3)

3.3.2.3 Roles y Responsabilidades

En este contexto, solo se identifica un rol relacionado con la gestión de riesgos:

Coordinador

El alumno, como principal interesado en el proyecto se hará cargo del control de los riesgos del proyecto

3.3.2.4 Presupuesto

Costes plan gestión: 250 €

Costes plan contingencia: 3200 €

3.3.2.5 Calendario

Existen dos actividades: evaluaciones y reuniones.

Evaluaciones

Regularmente, según lo indicado en las hojas de riesgos, se evaluarán los indicadores de riesgo y se valorará su estado. Dado el contexto y la duración del proyecto esta actividad no debería llevar más de un 2% de trabajo

Reuniones

Al menos en las reuniones de cada hito intermedio se valorará con el Director de Proyecto el estado de los riesgos más importantes

3.3.2.6 Categorías de Riesgo

Las siguientes son las principales categorías para los riesgos identificados. Un riesgo puede pertenecer a más de una categoría.

1. Técnico
 - 1.1. Tecnologías
 - 1.2. Desarrollo
 - 1.3. Área Base Datos
 - 1.4. Área web
2. Organización
 - 2.1. Tareas
 - 2.2. Dependencias
 - 2.3. Recursos
3. Gestión de Proyecto
 - 3.1. Planificación

- | |
|---------------------------|
| 3.2. Comunicación |
| 3.3. Control |
| 4. Externo |
| 4.1. Universidad |
| 4.2. Director de Proyecto |

3.3.2.7 Definiciones de probabilidad

Muy Baja	(0%..10%] Valor usado en la Matriz de Probabilidad e Impacto: 10%
Baja	(10%..30%] Valor usado en la Matriz de Probabilidad e Impacto: 30%
Media	(30%..50%] Valor usado en la Matriz de Probabilidad e Impacto: 50%
Alta	(50%..70%] Valor usado en la Matriz de Probabilidad e Impacto: 70%
Muy Alta	(70%..100%] Valor usado en la Matriz de Probabilidad e Impacto: 90%

3.3.2.8 Definiciones de impacto por objetivos

Impacto sobre los objetivos principales <i>Definirlo sobre amenazas y oportunidades</i>					
Objetivos	Muy Bajo/5%	Bajo/15%	Medio/30%	Alto/55%	Crítico/90%
Calidad	Fallo <1 test/1000 loc	Fallo >=1 test/1000 loc y <=5/1000 loc	Fallo >=5 test/1000 loc y <=10/1000 loc	Fallo >=10 test/1000 loc y <=20/1000 loc	Fallo >20 test/1000 loc
Alcance	Las modificaciones suponen <1.0% incremento en tiempo	Las modificaciones suponen >=1.0% y <5.0% variación en tiempo	Las modificaciones suponen >=5.0% y <10.0% variación en tiempo	Las modificaciones suponen >=10.0% y <20.0% variación en tiempo	Las modificaciones suponen >20.0% variación en tiempo
Tiempo	Incremento en tiempo despreciable (<1.0%)	Incremento >=1.0% y <5.0%	Incremento >=5.0% y <10.0%	Incremento >=10.0% y <20.0%	Incremento >=20.0%

Tabla 1 Matriz de Probabilidades e Impacto

Probabilidad	Muy Alta	0,90	0,05	0,14	0,27	0,50	0,81
	Alta	0,70	0,04	0,11	0,21	0,39	0,63
	Media	0,50	0,03	0,08	0,15	0,28	0,45
	Baja	0,30	0,02	0,05	0,09	0,17	0,27
	Muy	0,10	0,01	0,02	0,03	0,06	0,09

	Baja						
		0,05	0,15	0,30	0,55	0,90	
	Muy Bajo	Bajo	Medio	Alto	Crítico		
Impacto							

3.3.2.9 Planes de contingencia

Alcance

El alcance del proyecto está definido inicialmente con el Director del Proyecto.

Una reducción significativa del alcance necesitaría de la aprobación previa del Director (en función de las dificultades encontradas, calidad, ...)

Como medida excepcional se puede considerar trasladar la lectura a la convocatoria de extraordinaria de Julio.

Tiempo

La planificación inicial de las tareas tiene en cuenta los posibles retrasos, además se estima que algunos retrasos se compensarán con los adelantos en otras.

La fecha de entrega tiene un plazo fijo. El proceso de implementación y documentación tienen que estar finalizados 15 días antes de la fecha de defensa de la convocatoria ordinaria de Junio.

Como medida excepcional se puede considerar trasladar la lectura a la convocatoria de extraordinaria de Julio.

Capítulo 4. Análisis

4.1 Definición del Sistema

4.1.1 Determinación del Alcance del Sistema

El proyecto consta de varios módulos que ofrecen la siguiente funcionalidad:

Persistencia:

- Se define un modelo de datos que almacene toda la información relevante del sistema, así como la recuperada de los sistemas de los clientes.
- Se crea una Base de Datos en modo servidor que implemente el modelo de datos anterior

Servidor:

- Accede a la Base de Datos central para recuperar/almacenar todos los datos del sistema.
- Permite, mediante acceso a la BBDD central configurar los agentes
- Despliega un servidor SSH con acceso desde el exterior para que los agentes depositen los ficheros de datos de los sistemas de los clientes

Comunicaciones:

- Definir una API para la comunicación entre el servidor y los agentes con las siguientes operaciones:
 - Ping: El agente conecta con el servidor para comprobar que el servidor está online y comunicar al servidor que el agente está activo.
 - GetAgente: El agente recupera del servidor la configuración de las tareas a realizar en el servidor
 - SetSnapshot: El agente envía al servidor la notificación de que le ha enviado mediante SCP un conjunto de datos del cliente.

Agente:

- Crea una BBDD local para almacenar temporalmente los datos recuperados del cliente
- Conecta periódicamente con el servidor mediante el servicio web anterior para actualizar las tareas que realiza sobre el sistema del cliente.
- Las tareas posibles a realizar en el sistema son, la consulta de la Base de Datos del cliente mediante SQL y la ejecución de comandos del Sistema Operativo del Host.
- Planifica y ejecuta las tareas anteriores, guardando en la BBDD local los resultados
- Implementa un cliente SCP para conectar con el servicio SSH del servidor y depositar los ficheros con los datos recuperados del sistema

- Conecta periódicamente con el servidor para enviarle los resultados de las tareas ejecutadas, mediante SCP le entrega los datos y mediante el servicio web le notifica del envío
- El agente se puede ejecutar como servicio, o mediante una consola de administración interactiva en línea de comandos

4.2 Requisitos del Sistema

4.2.1 Obtención de los Requisitos del Sistema

Tabla 2 Requisitos funcionales

Código	Nombre Requisito	Descripción del Requisito
R1.1	Ping	El cliente debe conectar periódicamente con el servidor para verificar que está online
R1.2	Ejecutar consultas en el cliente	Posibilidad de ejecutar cualquier consulta configurada desde el repositorio central, obtener el resultado y guardarlo en el repositorio local
R1.3	Recolección de ficheros de log	Recolectar periódicamente el alert.ora y otros ficheros de log que se determinen (listener.log y archivos del RAC si es el caso) y subirlos al repositorio central.
R1.4	Programación de tareas	La definición y configuración (periodicidad, etc) de las consultas se realizará a través del repositorio central a donde se tendrá que conectar el agente para obtenerlas.
R1.5	Envío de información	Envío de información recolectada y que se encuentra en el repositorio local al repositorio central

Tabla 3 Requisitos no funcionales

Código	Nombre Requisito	Descripción del Requisito
R2.1	Multiplataforma	Soporte para Windows, Linux, Unix
R2.2	Modular	Separación en capas y componentes para añadir o sustituir funcionalidad
R2.3	Configurable	IP, rutas de ficheros, credenciales, ... tiene que ser fácilmente configurables
R2.4	Seguridad	Envío de ficheros cifrado

4.2.2 Identificación de Actores del Sistema

El actor primario es el administrador que mediante Monitora gestiona las bases de datos remotas. El sistema del cliente y el sistema externo que consume los datos del servidor central son actores secundarios

4.2.3 Especificación de Casos de Uso

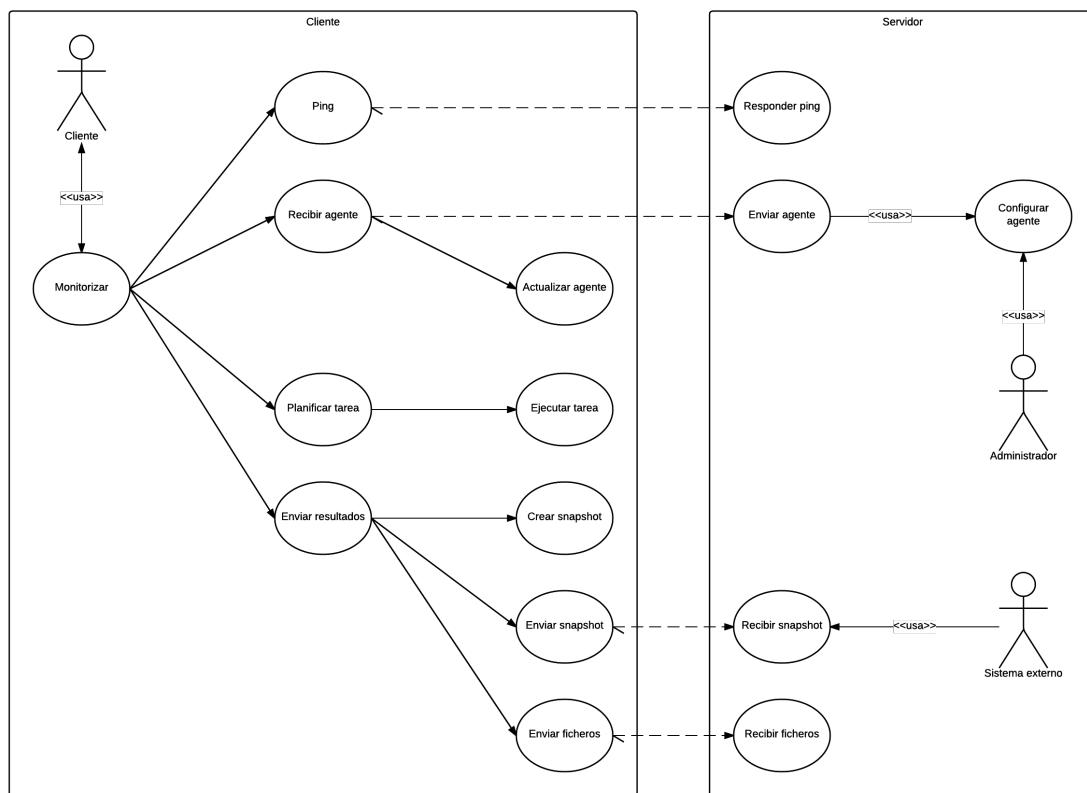


Ilustración 4 Diagrama de Casos de Uso

Nombre del Caso de Uso
Monitorizar
Descripción
El agente queda residente en el sistema destino y periódicamente conecta con el servidor para actualizar la información, planifica las tareas, las ejecuta y envía al servidor el resultado

Nombre del Caso de Uso
Ping
Descripción
El agente conecta con el servidor para informar que está online

Nombre del Caso de Uso
Recibir agente
Descripción
El agente pide al servidor información sobre las tareas a realizar

Nombre del Caso de Uso
Actualizar agente
Descripción
El agente actualiza su información con la información recuperada del servidor

Nombre del Caso de Uso
Planificar tarea
Descripción
El agente interpreta la planificación de cada tarea y se encarga de ejecutarla en el momento preciso

Nombre del Caso de Uso
Ejecutar tarea
Descripción
El agente ejecuta las órdenes indicadas sobre el sistema destino y guarda los resultados para su tratamiento posterior

Nombre del Caso de Uso
Enviar resultados
Descripción
El agente recupera la información almacenada con los resultados de las tareas, los agrupa creando una snapshot y los envía al servidor

Nombre del Caso de Uso
Crear snapshot
Descripción
El agente recupera la información almacenada con los resultados de las tareas, los transforma a XML, los vuelve a fichero y los comprime

Nombre del Caso de Uso
Enviar snapshot
Descripción
El agente conecta con el servidor y le envía la información de las tareas realizadas

Nombre del Caso de Uso
Enviar ficheros
Descripción
El agente sube al servidor mediante SCP los datos en XML de las tareas realizadas. El envío se realiza de forma segura

Nombre del Caso de Uso
Responder Ping
Descripción
El servidor responde al ping del cliente y guarda el instante de tiempo en que se hizo la petición

Nombre del Caso de Uso
Enviar agente

Descripción
El servidor recupera la información del agente solicitado y se la envía al cliente

Nombre del Caso de Uso
Configurar agente
Descripción
El administrador de BBDD crea/modifica los datos necesarios para gestionar agentes, tareas, planificaciones, etc accediendo directamente a la BBDD del servidor

Nombre del Caso de Uso
Recibir snapshot
Descripción
El servidor recibe y actualiza sus datos sobre las operaciones realizadas en cada agente y verifica que los datos recibidos son correctos

Nombre del Caso de Uso
Recibir ficheros
Descripción
El agente recupera los ficheros subidos por el cliente, comprueba que son correctos y actualiza la BBDD

4.3 Identificación de los Subsistemas en la Fase de Análisis

4.3.1 Descripción de los Subsistemas

Los subsistemas encontrados inicialmente son:

- **Cliente** - Planifica y ejecuta las operaciones sobre la BBDD objetivo
- **Servidor** - Repositorio central donde se vuelca la información de todos los clientes.
- **Comunicaciones** – Permite la comunicación entre el cliente y el servidor y el envío de los resultados de las operaciones
- **Persistencia** - Almacena los datos de las operaciones, tanto el cliente como en el servidor

4.3.2 Descripción de los Interfaces entre Subsistemas

La comunicación entre subsistemas se produce mediante llamadas a funciones locales, excepto en el caso del envío de datos cliente/servidor que se realiza mediante un servicio REST con JSON y un servicio SCP.

4.4 Diagrama de Clases Preliminar del Análisis

4.4.1 Diagrama de Clases

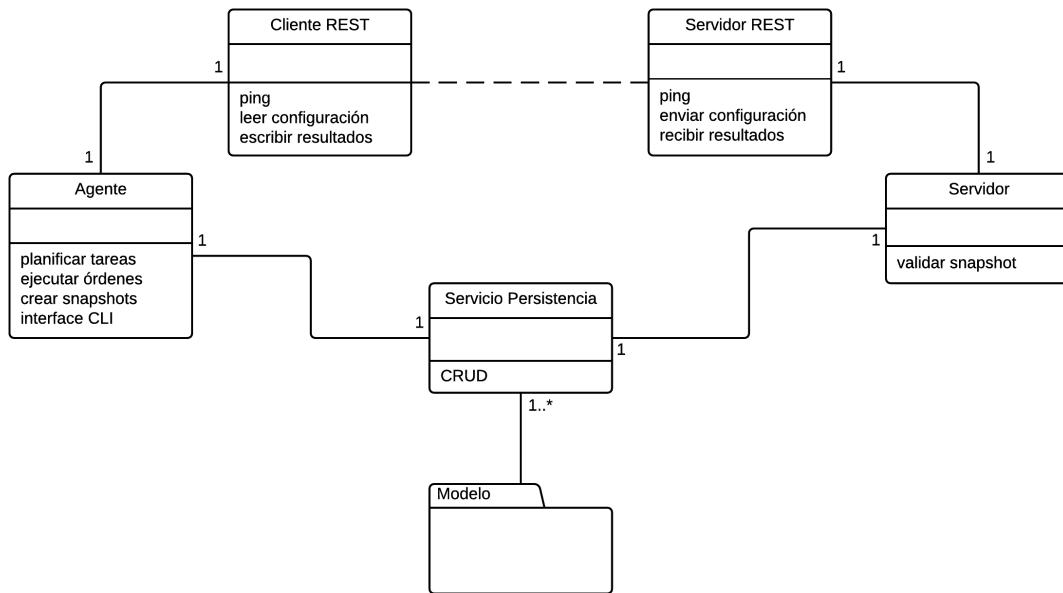


Ilustración 5 - Diagrama de clases de análisis

4.4.2 Descripción de las Clases

4.4.2.1 Comunicaciones

Nombre de la Clase
Cliente REST
Descripción
Cliente de servicio web REST
Responsabilidades
Conectar con el servidor Recuperar mediante GET la información solicitada Enviar mediante POST la información solicitada
Atributos Propuestos
Métodos Propuestos
ping : Conecta con el servidor indicando el identificador de cliente y verifica que el servidor está online leer configuración : Conecta con el servidor indicando el identificador de cliente y recupera la información actualizada escribir resultados : Conecta con el servidor y le envía el resultado de las tareas realizadas en el cliente

Nombre de la Clase
Servidor REST
Descripción
Servidor de servicio web REST
Responsabilidades
Esperar a las conexiones de los clientes Enviar la información solicitada mediante GET Recibir la información recibida mediante POST
Atributos Propuestos
Métodos Propuestos
ping: Contesta al servidor con el instante en que se recibió la petición enviar configuración: Envía al cliente la información actualizada del cliente solicitado recibir resultados: Recibe del cliente los resultados de las tareas realizadas en el cliente

4.4.2.2 Agente

Nombre de la Clase
Agente
Descripción
Planifica las tareas a realizar en el cliente, las ejecuta en el momento indicado y crea los snapshots con los resultados. Permite ejecutarlo como servicio o mediante un interface de línea de comandos
Responsabilidades
Planificar tareas Ejecutar tareas y guardar el resultado Recuperar resultados y crear snapshots Ejecución como servicio o interactiva
Atributos Propuestos
Métodos Propuestos
planificar tareas: interpreta la planificación y lanza la tarea en el momento indicado ejecutar órdenes: ejecuta la tarea y guarda el resultado en BBDD crear snapshot: recupera el resultado de la BBDD, lo vuelve a XML y lo empaqueta en un solo archivo interface CLI: permite ejecutar el cliente como servicio o de forma interactiva mediante línea de comandos

4.4.2.3 Servidor

Nombre de la Clase
Servidor
Descripción
Comunica la capa de persistencia con la de comunicaciones y valida que los datos son correctos
Responsabilidades
Validar datos
Atributos Propuestos
Métodos Propuestos

validar snapshot: verifica que los datos de la snapshot son correctos

4.4.2.4 Persistencia

Nombre de la Clase
Servicio Persistencia
Descripción
Clase para el acceso a la BBDD
Responsabilidades
Gestionar el acceso a la BBDD
Atributos Propuestos
Métodos Propuestos
Crear/Borrar/Modificar/Buscar: Operaciones CRUD para todas las tablas

Se crean clases adicionales para cada entidad del modelo de datos. Estas clases no tienen lógica de negocio.

4.5 Análisis de Casos de Uso y Escenarios

4.5.1 Monitorizar

Monitorizar	
Precondiciones	Fichero de configuración contiene la IP del servidor
Poscondiciones	El servidor está online y permite conexiones desde el cliente
Actores	Iniciado periódicamente por el agente
Descripción	Ejecutar periódicamente: <ol style="list-style-type: none"> 1. Ping 2. Recibir agente y actualizar 3. Planificar tareas 4. Enviar resultados
Variaciones (escenarios secundarios)	<ul style="list-style-type: none"> • Escenario Alternativo 1: El servidor no responde <ul style="list-style-type: none"> ○ Salir del ciclo actual y volver a intentarlo más tarde
Excepciones	<ul style="list-style-type: none"> • Error de persistencia: <ul style="list-style-type: none"> ○ Mostrar mensaje de error • Error de I/O: <ul style="list-style-type: none"> ○ Mostrar mensaje de error
Notas	

4.5.2 Ping

Ping	
Precondiciones	Fichero de configuración correcto
Poscondiciones	El servidor está online y permite conexiones desde el cliente
Actores	Iniciado periódicamente por el agente
Descripción	<ol style="list-style-type: none"> 1. El cliente REST conecta mediante GET con el servidor 2. Verifica que la respuesta es correcta 3. Devolver si la respuesta es correcta o no
Variaciones (escenarios secundarios)	<ul style="list-style-type: none"> • Escenario Alternativo 1: El servidor no responde <ul style="list-style-type: none"> ○ En el paso 3 del escenario principal, devolver FALSO
Excepciones	<ul style="list-style-type: none"> • Identificador de cliente incorrecto: <ul style="list-style-type: none"> ○ Mostrar mensaje de error
Notas	

4.5.3 Recibir agente

Recibir agente	
Precondiciones	Fichero de configuración contiene la IP del servidor

Poscondiciones	El agente tiene la información actualizada del servidor
Actores	Iniciado periódicamente por el agente
Descripción	<ol style="list-style-type: none"> 1. El cliente REST conecta mediante GET con el servidor 2. Verifica que la respuesta es correcta 3. Llama al caso de uso “Actualizar agente”
Variaciones (escenarios secundarios)	<ul style="list-style-type: none"> • Escenario Alternativo 1: El servidor no responde <ul style="list-style-type: none"> ○ No se ejecuta el paso 3 del escenario principal
Excepciones	<ul style="list-style-type: none"> • Identificador de cliente incorrecto: <ul style="list-style-type: none"> ○ Mostrar mensaje de error
Notas	

4.5.4 Actualizar agente

Actualizar agente	
Precondiciones	
Poscondiciones	El agente tiene la información actualizada del servidor
Actores	Iniciado periódicamente por el agente
Descripción	<ol style="list-style-type: none"> 1. El agente actualiza la información usando los servicios de la capa de persistencia
Variaciones (escenarios secundarios)	
Excepciones	<ul style="list-style-type: none"> • Error en la capa de persistencia: <ul style="list-style-type: none"> ○ Mostrar mensaje de error
Notas	

4.5.5 Planificar tarea

Planificar tarea	
Precondiciones	Existen tareas en el agente
Poscondiciones	
Actores	Iniciado periódicamente por el agente
Descripción	<ol style="list-style-type: none"> 1. El agente recupera de la capa de persistencia la planificación y las tareas a realizar 2. Analiza la planificación 3. Crea para cada tarea una orden de trabajo en el momento indicado por la planificación 4. En el momento indicado llama al caso de uso “Ejecutar tarea”
Variaciones (escenarios secundarios)	
Excepciones	<ul style="list-style-type: none"> • Error en la planificación:

	<ul style="list-style-type: none"> <input type="radio"/> Mensaje de error en el log de la aplicación <input type="radio"/> Continuar con la planificación de otras tareas
Notas	La planificación en sí misma se delega a una librería externa

4.5.6 Ejecutar tarea

Ejecutar tarea	
Precondiciones	Existen órdenes de trabajo (tareas planificadas)
Poscondiciones	Existe una tabla en el sistema con el resultado de la ejecución de la tarea
Actores	Iniciado periódicamente por el agente
Descripción	<ol style="list-style-type: none"> 1. Se crea la tabla donde se guarda el resultado 2. Se ejecuta la orden (consulta a la BBDD o comando de Sistema Operativo) 3. Se guarda el resultado en la tabla de resultados
Variaciones (escenarios secundarios)	<ul style="list-style-type: none"> • Escenario Alternativo 1: La tabla de resultados ya existe <ul style="list-style-type: none"> <input type="radio"/> Continuar con el paso 2 del escenario principal
Excepciones	<ul style="list-style-type: none"> • Error en la capa de persistencia: <ul style="list-style-type: none"> <input type="radio"/> Mensaje de error en el log de la aplicación <input type="radio"/> Continuar con la ejecución de otras tareas
Notas	La ejecución en sí misma se delega a una librería externa

4.5.7 Enviar resultados

Enviar resultados	
Precondiciones	Existen resultados de órdenes de trabajo
Poscondiciones	Los resultados enviados ya no existen en el agente
Actores	Iniciado periódicamente por el agente
Descripción	<ol style="list-style-type: none"> 1. Se lee la tabla de resultados que 2. Se crea un snapshot con todos los resultados recuperados llamando al caso de uso “Crear snapshot” 3. Se envía el fichero de datos del snapshot llamando al caso de uso “Enviar ficheros” 4. Se envía el snapshot llamando al caso de uso “Enviar snapshot” 5. Se borran los ficheros del agente
Variaciones (escenarios secundarios)	<ul style="list-style-type: none"> • Escenario Alternativo 1: Error en el envío del fichero de datos <ul style="list-style-type: none"> <input type="radio"/> Salir del proceso de envío. Se reintentará más tarde • Escenario alternativo 2: Error en el envío del snapshot <ul style="list-style-type: none"> <input type="radio"/> Salir del proceso de envío. Se reintentará más tarde
Excepciones	<ul style="list-style-type: none"> • Error en la capa de persistencia: No se pueden leer las tablas <ul style="list-style-type: none"> <input type="radio"/> Mensaje de error en el log de la aplicación

	<ul style="list-style-type: none"> ○ Continuar con la lectura de otras tablas
Notas	

4.5.8 Crear snapshot

Crear snapshot	
Precondiciones	Existen resultados de órdenes de trabajo Fichero de configuración con la ruta de la carpeta de snapshots en el agente
Poscondiciones	Existe un fichero en el agente con el contenido de las tablas de resultados
Actores	Iniciado periódicamente por el agente
Descripción	<ol style="list-style-type: none"> 1. Se vuelcan las tablas de resultados a ficheros XML 2. Se crea un fichero de metadatos con información sobre las tablas volcadas 3. Se comprimen todos los ficheros en un fichero zip
Variaciones (escenarios secundarios)	
Excepciones	<ul style="list-style-type: none"> • Error en la capa de persistencia: No se pueden leer las tablas <ul style="list-style-type: none"> ○ Mensaje de error en el log de la aplicación ○ Continuar con la lectura de otras tablas • Error de escritura: No hay permisos de escritura en la carpeta de snapshots <ul style="list-style-type: none"> ○ Mostrar mensaje de error
Notas	

4.5.9 Enviar snapshot

Enviar snapshot	
Precondiciones	Existe un fichero de datos de snapshot Fichero de configuración válido
Poscondiciones	El servidor ha recibido el snapshot
Actores	Iniciado periódicamente por el agente
Descripción	<ol style="list-style-type: none"> 1. El cliente REST envía mediante POST al servidor el snapshot 2. Verifica que la respuesta es correcta
Variaciones (escenarios secundarios)	<ul style="list-style-type: none"> • Escenario Alternativo 1: Error en el envío del fichero de datos <ul style="list-style-type: none"> ○ Salir del proceso de envío. Se reintentará más tarde
Excepciones	<ul style="list-style-type: none"> • El servidor no conecta o devuelve error: <ul style="list-style-type: none"> ○ Mensaje de error en el log de la aplicación ○ Salir del proceso de envío. Se reintentará más tarde
Notas	

4.5.10 Enviar ficheros

Enviar ficheros	
Precondiciones	Existe un fichero de datos de snapshot Fichero de configuración válido
Poscondiciones	El servidor ha recibido el fichero
Actores	Iniciado periódicamente por el agente
Descripción	<ol style="list-style-type: none"> 1. El agente envía por SCP el fichero al servidor 2. Verifica que la respuesta es correcta
Variaciones (escenarios secundarios)	
Excepciones	<ul style="list-style-type: none"> • El servidor no conecta o devuelve error: <ul style="list-style-type: none"> ○ Mensaje de error en el log de la aplicación ○ Salir del proceso de envío. Se reintentará más tarde
Notas	

4.5.11 Responder ping

Responder ping	
Precondiciones	
Poscondiciones	
Actores	Respuesta a una petición al servicio web
Descripción	<ol style="list-style-type: none"> 1. Comprueba que el agente existe en la capa de persistencia 2. Responde a la petición con la hora
Variaciones (escenarios secundarios)	<ul style="list-style-type: none"> • Escenario Alternativo 1: El agente no existe <ul style="list-style-type: none"> ○ Responder a la petición con un error
Excepciones	<ul style="list-style-type: none"> • Error en la capa de persistencia: <ul style="list-style-type: none"> ○ Mensaje de error en el log de la aplicación ○ Responder a la petición con un error
Notas	

4.5.12 Enviar agente

Enviar agente	
Precondiciones	
Poscondiciones	
Actores	Respuesta a una petición al servicio web
Descripción	<ol style="list-style-type: none"> 1. Comprueba que el agente existe en la capa de persistencia 2. Recupera todo los datos del agente 3. Responde a la petición con los datos del agente en formato JSON

Variaciones (escenarios secundarios)	<ul style="list-style-type: none"> Escenario Alternativo 1: El agente no existe <ul style="list-style-type: none"> Responder a la petición con un error
Excepciones	<ul style="list-style-type: none"> Error en la capa de persistencia: <ul style="list-style-type: none"> Mensaje de error en el log de la aplicación Responder a la petición con un error
Notas	

4.5.13 Configurar agente

Configurar agente	
Precondiciones	
Poscondiciones	
Actores	El administrador de BBDD
Descripción	<ol style="list-style-type: none"> Accede directamente a la BBDD Crea/modifica las tablas correspondientes al agente
Variaciones (escenarios secundarios)	
Excepciones	<ul style="list-style-type: none"> Error en la BBDD: <ul style="list-style-type: none"> Mensaje de error al ejecutar las operaciones sobre la BBDD
Notas	

4.5.14 Recibir snapshot

Recibir snapshot	
Precondiciones	El servidor web tiene que estar online
Poscondiciones	Los datos del snapshot se encuentran en la BBDD
Actores	Respuesta a una petición al servicio web
Descripción	<ol style="list-style-type: none"> Valida los ficheros del snapshot llamando al caso de uso “Recibir ficheros” Guarda el snapshot usando los servicios de la capa de persistencia Responde a la petición con OK
Variaciones (escenarios secundarios)	<ul style="list-style-type: none"> Escenario Alternativo 1: Error en la validación de los ficheros del snapshot <ul style="list-style-type: none"> Responder a la petición con un error
Excepciones	<ul style="list-style-type: none"> Error en la capa de persistencia: <ul style="list-style-type: none"> Mensaje de error en el log de la aplicación Responder a la petición con un error
Notas	

4.5.15 Recibir ficheros

Recibir ficheros	
Precondiciones	El servidor SSH tiene que estar online Fichero de configuración válido
Poscondiciones	Los datos del snapshot se encuentran en la BBDD
Actores	Respuesta a una petición al servicio web
Descripción	<ol style="list-style-type: none"> 1. Comprueba que el fichero de datos existen en la carpeta del servidor 2. Extrae los ficheros XML 3. Inserta los datos en las tablas 4. Devuelve OK
Variaciones (escenarios secundarios)	
Excepciones	<ul style="list-style-type: none"> • Error en la capa de persistencia: <ul style="list-style-type: none"> ○ Mensaje de error en el log de la aplicación ○ Responder a la petición con un error • Error de lectura del fichero: No hay permisos de lectura o fichero corrupto <ul style="list-style-type: none"> ○ Mensaje de error en el log de la aplicación ○ Responder a la petición con un error
Notas	

4.6 Especificación del Plan de Pruebas

Para detectar errores y corregirlos lo antes posible se contemplan varios tipos de pruebas:

- Pruebas unitarias:** Desde la funcionalidad más básica y siguiendo un esquema de desarrollo ágil, se implementan las pruebas a la vez que la funcionalidad principal lo que permite definir claramente las interfaces de los componentes y partir de una base robusta. En estas pruebas nos apoyamos en las librerías JUnit y Mockito.
- Pruebas de Integración:** En este proyecto las pruebas de integración tienen una importancia vital. Para probar la capa de servicios del módulo de persistencia y el servicio web se implementan numerosos test con JUnit
- Pruebas del sistema:** Verificamos la correcta integración del agente-servidor y el envío de ficheros mediante SSH

Caso de Uso: Ping	
Prueba	Resultado Esperado
Ping desde un agente no válido	El sistema lanza una excepción
Ping desde un agente válido	El sistema devuelve la hora de petición y OK

Caso de Uso: Recibir agente	
Prueba	Resultado Esperado
Pedir un agente no válido	El sistema lanza una excepción
Pedir un agente válido	El sistema devuelve el agente con toda su información y OK

Caso de Uso: Actualizar agente	
Prueba	Resultado Esperado
Actualizar agente no existente en el sistema	El sistema crea el agente con toda la información
Actualizar agente existente en el sistema	El sistema sobrescribe la información del agente

Caso de Uso: Planificar tarea	
Prueba	Resultado Esperado
Planificar tarea con expresión CRON incorrecta	Excepción
Planificar tarea con línea cron correcta	El contador de tareas se incrementa en 1 Existe una nueva orden de trabajo

Caso de Uso: Ejecutar tarea	
Prueba	Resultado Esperado
Tabla de resultados ya existe	No se crea una nueva. No se lanza excepción

Prueba	Resultado Esperado
Tabla de resultados no existe	Se crea una nueva tabla
Prueba	Resultado Esperado
Error en la orden a ejecutar	Se lanza excepción
Prueba	Resultado Esperado
Error al guardar los resultados de la orden en la tabla	Se lanza excepción

Caso de Uso: Crear snapshot	
Prueba	Resultado Esperado
Error al volcar tabla a XML	Se lanza excepción
Prueba	Resultado Esperado
No hay permisos de escritura en la carpeta	Se lanza excepción
Prueba	Resultado Esperado
Error al comprimir carpeta	Se lanza excepción
Prueba	Resultado Esperado
Error al crear fichero de metadatos	Se lanza excepción

Caso de Uso: Enviar ficheros	
Prueba	Resultado Esperado
Error al enviar fichero datos	Se lanza excepción

Caso de Uso: Enviar snapshot	
Prueba	Resultado Esperado
Error al enviar snapshot	Se lanza excepción

Caso de Uso: Enviar resultados	
Prueba	Resultado Esperado
Error al crear snapshot	Log de error + tratar siguiente resultado
Prueba	Resultado Esperado
Error al enviar fichero datos	Log de error + tratar siguiente resultado
Prueba	Resultado Esperado
Error al enviar snapshot	Log de error + tratar siguiente resultado

Caso de Uso: Responder ping	
Prueba	Resultado Esperado
Ping desde cliente no válido	Devolver error

Caso de Uso: Enviar agente	
Prueba	Resultado Esperado
Petición de agente no válido	Devolver error
Prueba	Resultado Esperado
Error de persistencia	Devolver error

Caso de Uso: Recibir fichero	
Prueba	Resultado Esperado
No hay permiso de escritura en carpeta	Devolver error
Prueba	Resultado Esperado
Error al descomprimir fichero	Devolver error
Prueba	Resultado Esperado
Error al insertar XML en tablas	Devolver error

Caso de Uso: Recibir snapshot	
Prueba	Resultado Esperado
Error de persistencia	Devolver error
Prueba	Resultado Esperado
Error al recibir fichero	Devolver error

Capítulo 5. Diseño del Sistema

5.1 Arquitectura del Sistema

5.1.1 Diagramas de Paquetes

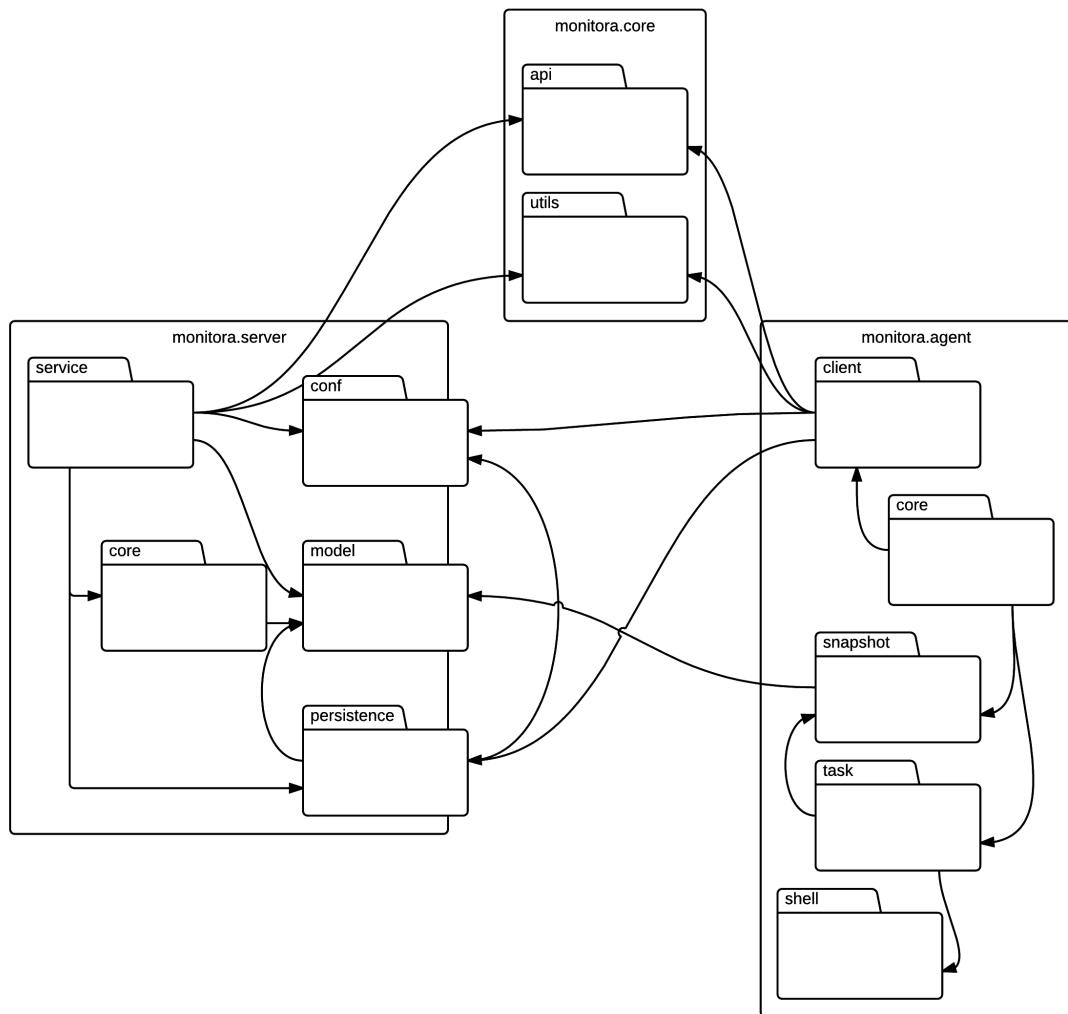


Ilustración 6 Diagrama de paquetes

5.1.1.1 Monitora.core

Paquete con clases comunes al cliente y a servidor

- **Api:** Clases adicionales para la API del servicio web
 - Ack: Clase que encapsula la respuesta al ping desde el servidor al agente
- **Utils:** Clases se utilidad

- ZipUtils: Clase para la gestión de ficheros comprimidos

5.1.1.2 *Monitora.server*

Paquetes para la gestión del servicio web, el modelo de datos y la capa de persistencia

- **Service:** Contiene el servidor web y la API del servicio REST. Las entidades se envían mediante JSON
- **Core:** Capa de servicios para el acceso a la capa de persistencia. Contiene una serie de interfaces que definen los servicios y las clases que los implementan
- **Conf:** Utilidades para la gestión del fichero de configuración y factorías de acceso a los servicios

Model: Entidades del modelo de datos. Usa anotaciones JPA y JAX-RS/Jackson

- **Persistence:** Capa de persistencia. Accede a la BBDD mediante JPA e Hibernate

5.1.1.3 *Monitora.agent*

Paquetes para la gestión del cliente web, la planificación de tareas y la gestión de snapshots

- **Client:** Cliente del servicio web
- **Core:** Capa de servicios para el acceso al resto de paquetes
- **Snapshot:** Gestión de snapshots (creación y envío)
- **Task:** Planificación de tareas
- **Shell:** Utilidades para la ejecución de comandos del sistema operativo

5.1.2 Diagramas de Componentes

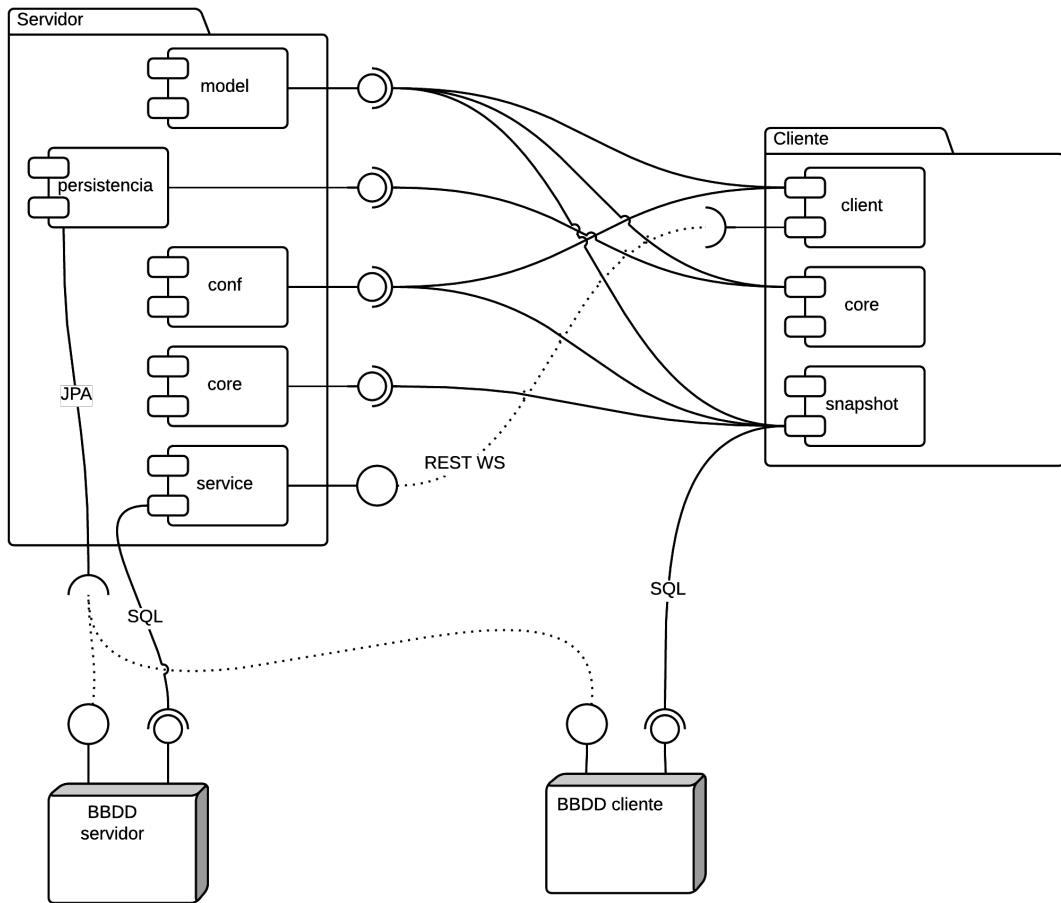


Ilustración 7 Diagrama de Componentes

En este diagrama se pueden ver la distribución de los diferentes paquetes entre el cliente y el servidor.

Destaca claramente la fuerte conexión entre el agente y el servidor ya que comparten el mismo modelo de datos, por tanto los paquetes **model**, **persistence** y la capa de servicios de **core** son las mismas. También las utilidades del paquete **conf** se usan en los dos componentes.

El acceso a las BBDD se produce de dos maneras; las entidades del modelo están mapeadas con JPA, pero las tablas dinámicas donde se vuelcan los datos recuperados del cliente no se conocen a priori, por lo que no se pueden mapear a entidades, y se accede directamente mediante SQL.

5.1.3 Diagramas de Despliegue

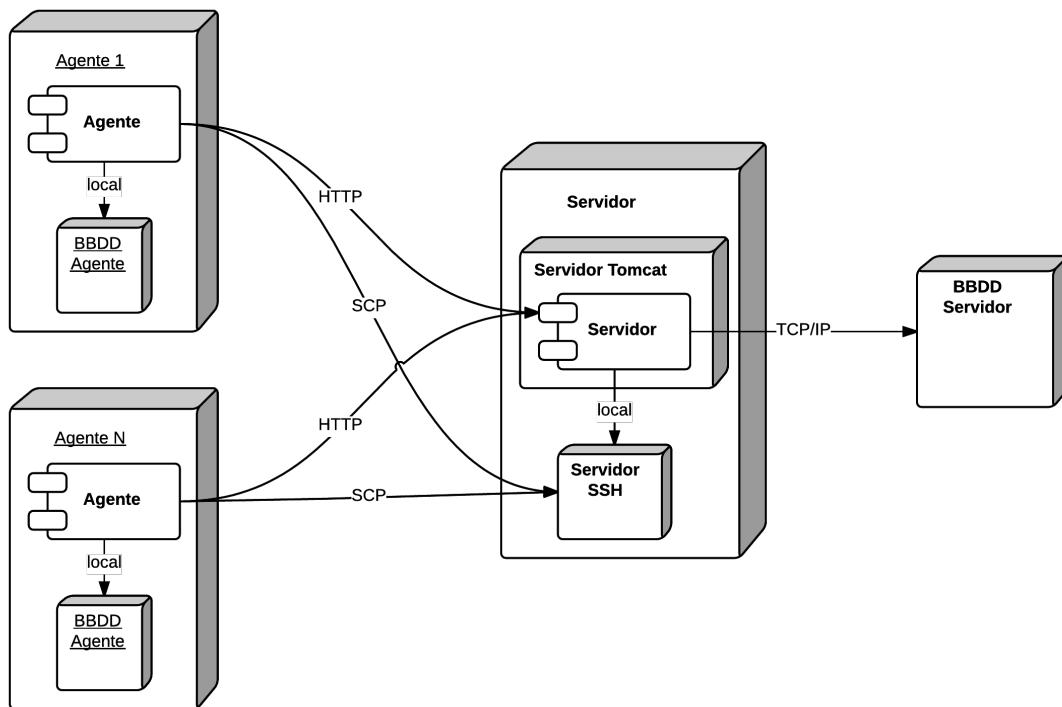


Ilustración 8 Diagrama de despliegue

5.1.3.1 BBDD Servidor

Una instancia de Base de datos en modo servidor a la que accede por TCP/IP

5.1.3.2 Servidor

Contiene una instancia de un servidor se aplicaciones J2EE con la aplicación servidor. Además tiene un servidor SSH para la transferencia segura de archivos desde los agentes

5.1.3.3 Agentes (N instancias)

Ejecutan el cliente y contiene una instancia de una base de datos local para el almacenamiento temporal de información

5.2 Diseño de Clases

En el diseño se ha intentado reflejar los objetivos y requisitos funcionales descritos anteriormente.

5.2.1 Arquitectura del servidor

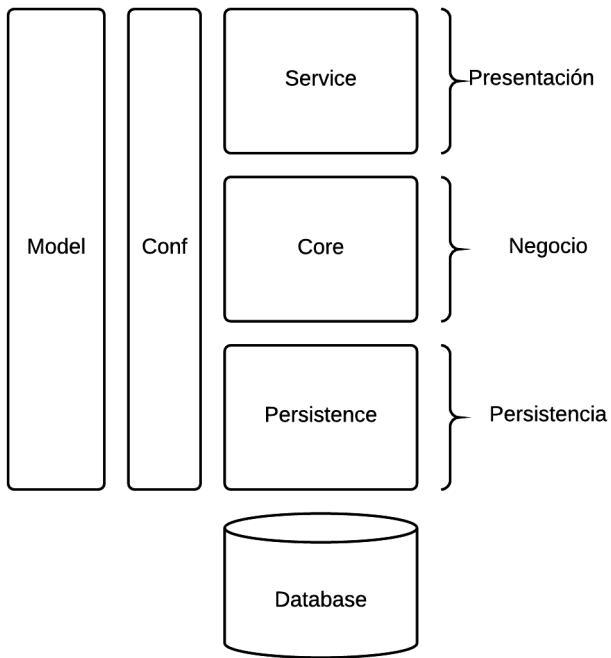
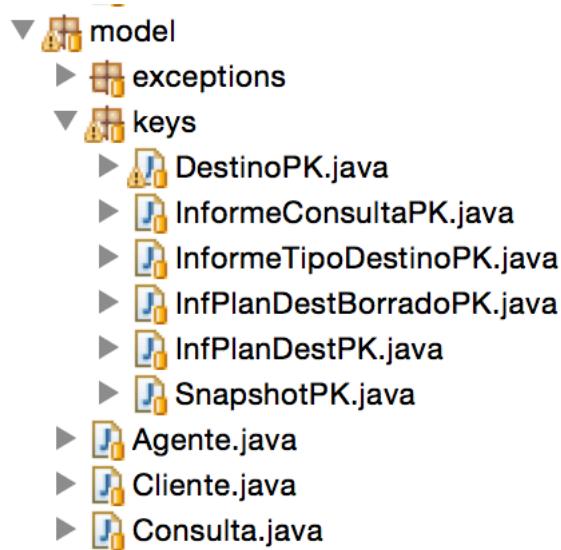
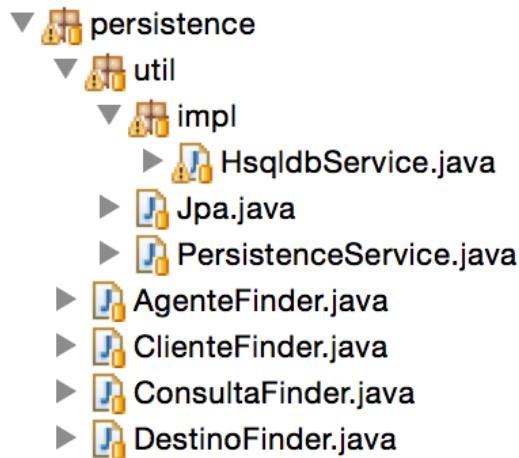


Ilustración 9 Diagrama de arquitectura del Servidor

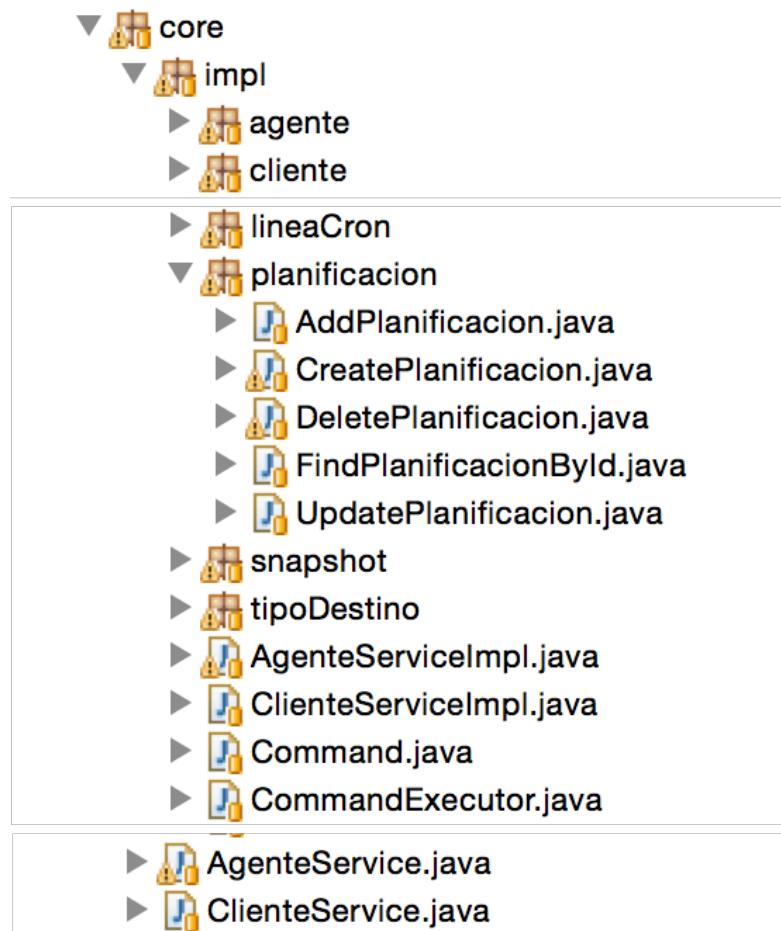
La arquitectura muestra un modelo en tres capas clásico con los paquetes Conf y Model transversales a toda la arquitectura.

*Ilustración 10 Servidor – Capa de modelo de datos*

El paquete model contiene las entidades del modelo de datos

*Ilustración 11 Servidor - Capa de persistencia*

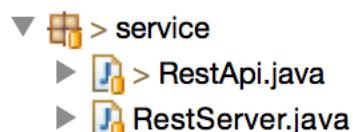
La capa de persistencia aísla el sistema de la BBDD. Publica una un fachada PersistenceService a las capas superiores, implementada en HSQLDB y contiene una serie de clases Finder que encapsulan las operaciones de búsqueda en JPA

*Ilustración 12 Servidor – Capa de negocio*

En la capa de negocio se define un Servicio para cada entidad de la capa de persistencia, usando una fachada para cada servicio y una serie de Clases que implementan el patrón Command para la implementación. En MonitoraServerService se definen las operaciones que dan soporte a la API del servicio web

*Ilustración 13 Servidor - Capa de utilidades*

A todos los servicios se accede mediante factorías en el paquete conf. La clase Conf permite el manejo de fichero de configuración y se implementa con un patrón Singleton

*Ilustración 14 Servidor - Capa de presentación*

Finalmente la capa de presentación, en este caso el servicio web, también tiene una fachada y su implementación

5.2.2 Arquitectura del agente

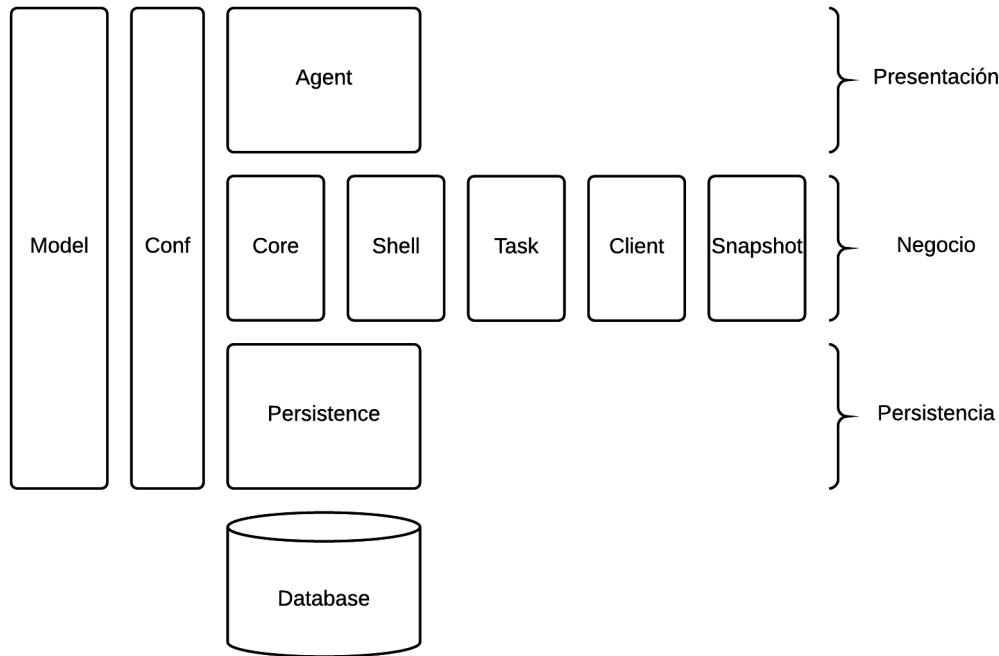


Ilustración 15 Diagrama de arquitectura del cliente

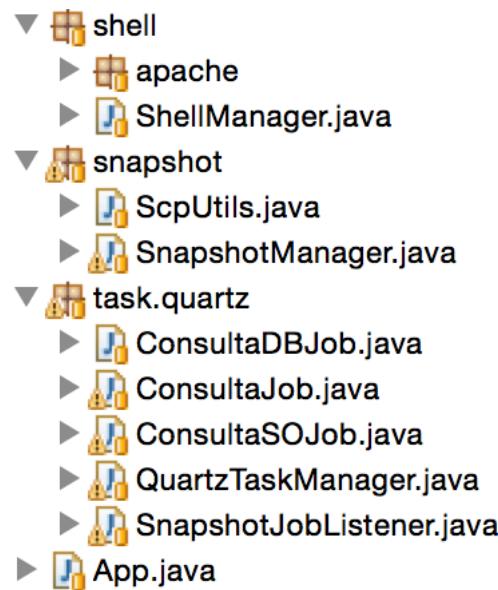


Ilustración 16 Agente - Capa de negocio

En el agente seguimos el mismo diseño en capas, con Servicios declarados como Fachadas, Factorías y el uso de Command para la implementación de las clases Job

5.2.3 Diagrama de Clases

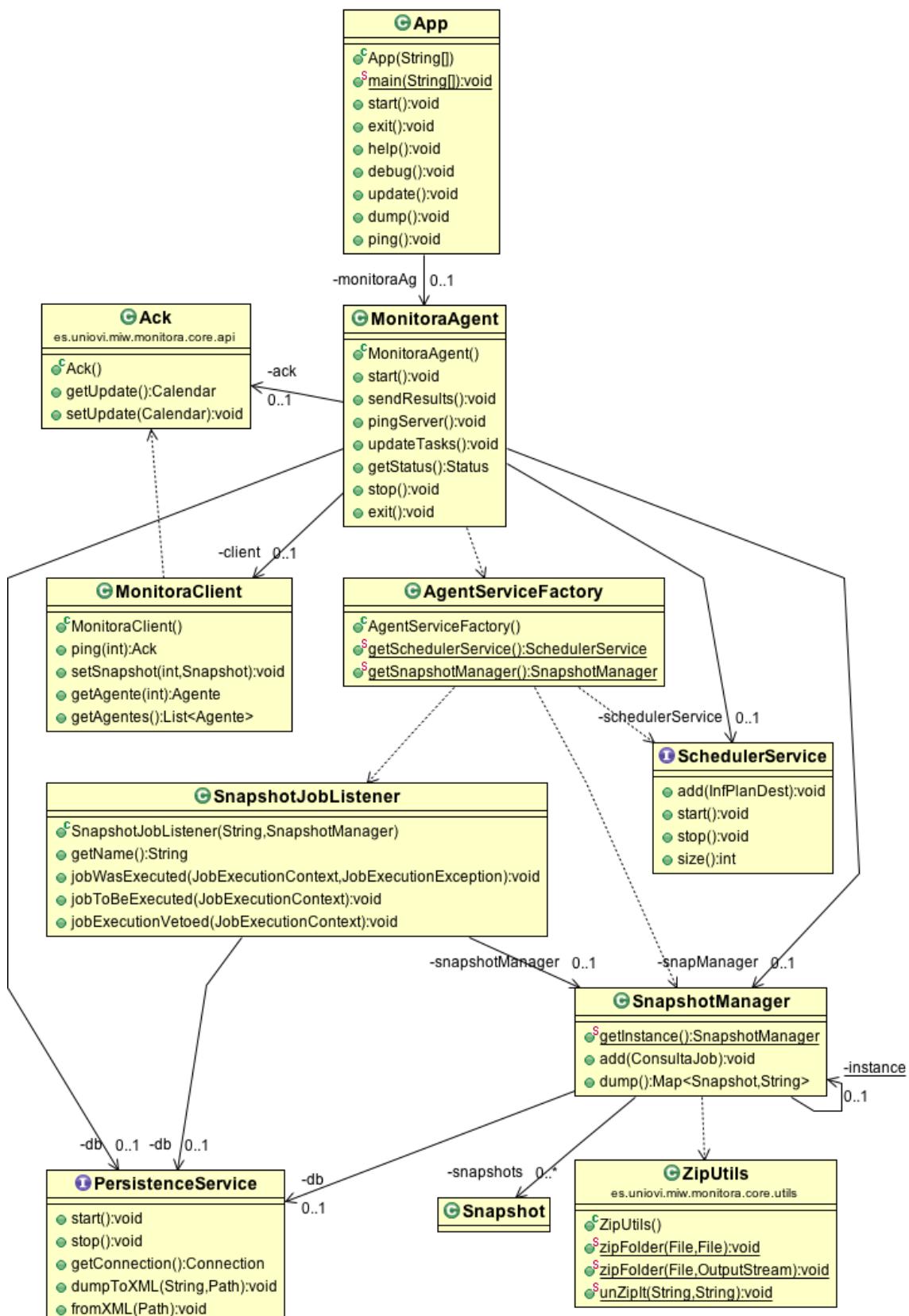


Ilustración 17 Diagrama de clases del agente

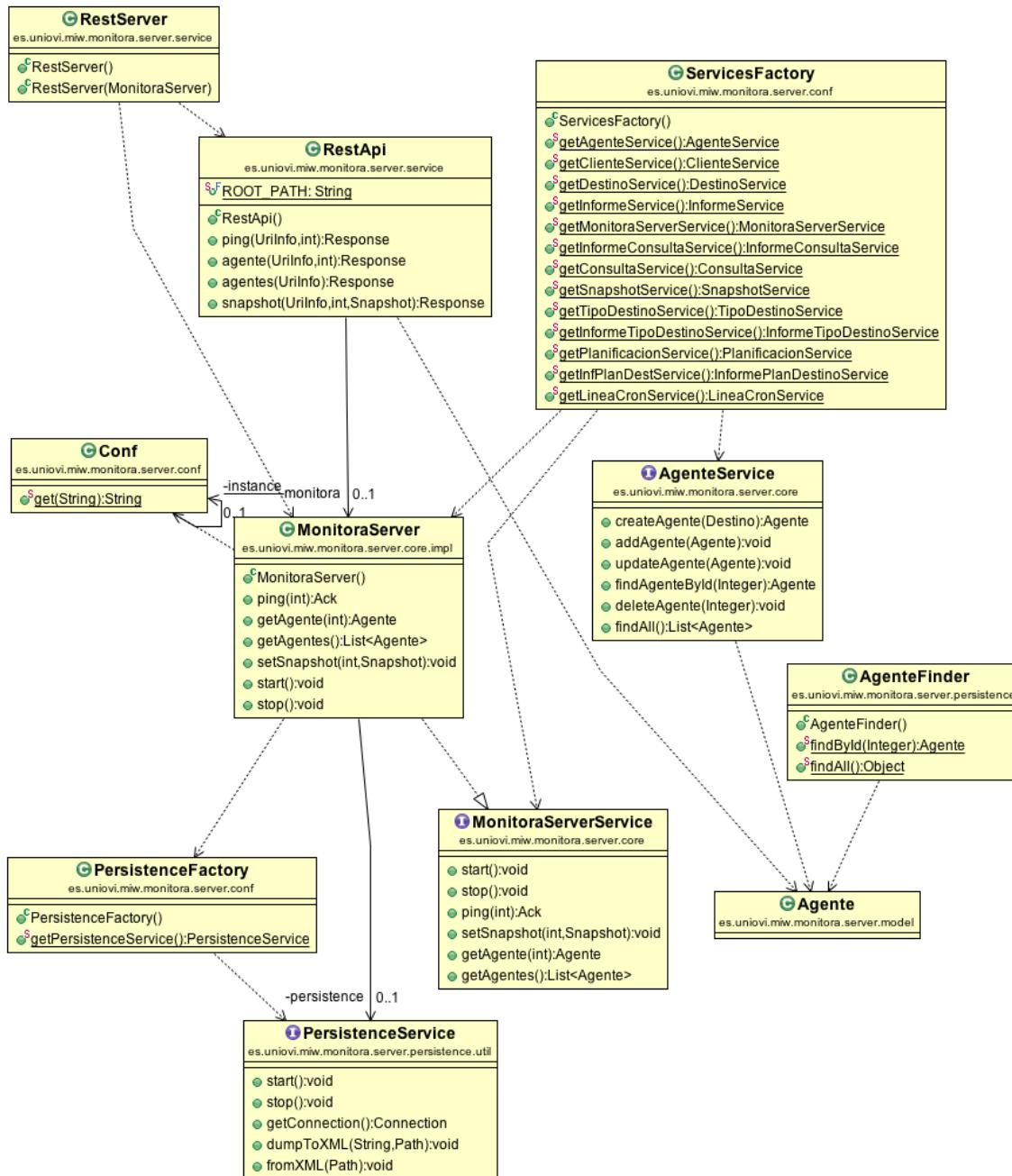


Ilustración 18 Diagrama de clases del servidor

En este diagrama solo se muestra la entidad Agente, el resto de las entidades del modelo tienen las mismas relaciones.

5.3 Diagramas de Interacción y Estados

5.3.1 Caso de Uso Recibir agente, Actualizar Agente y Planificar

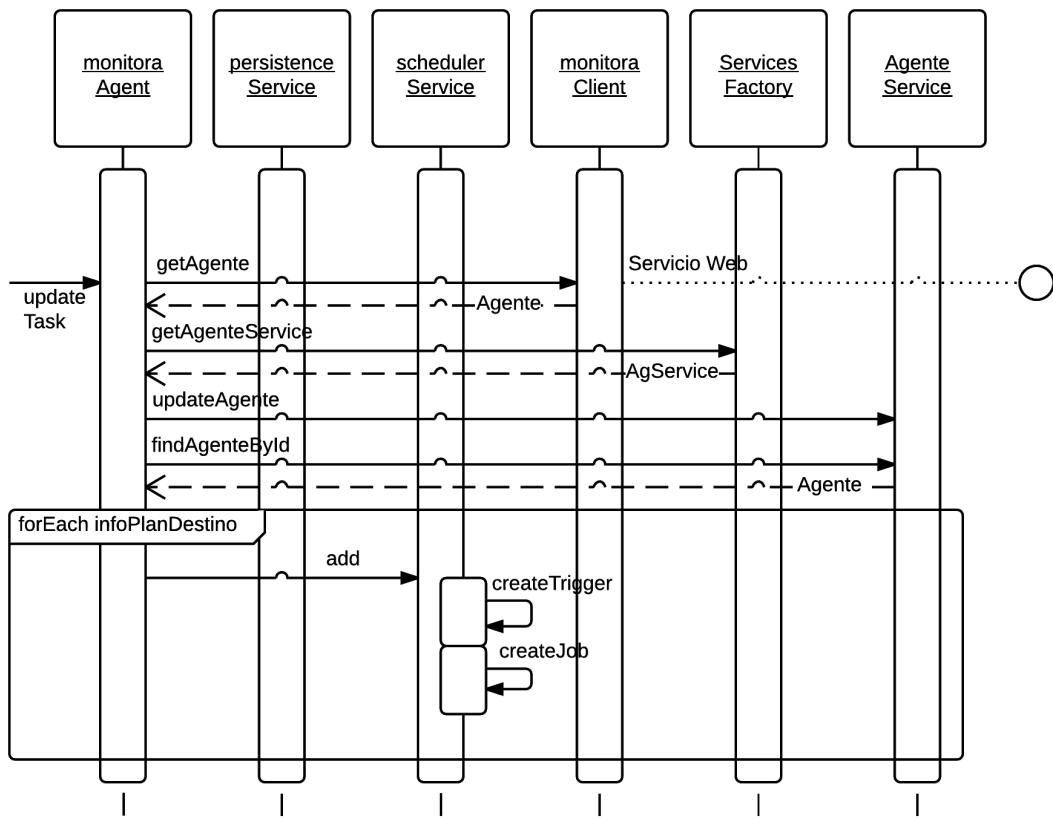


Ilustración 19 Diagrama de secuencia de Recibir Agente, Actualizar Agente y Planificar

5.3.2 Caso de Uso Ejecutar tarea

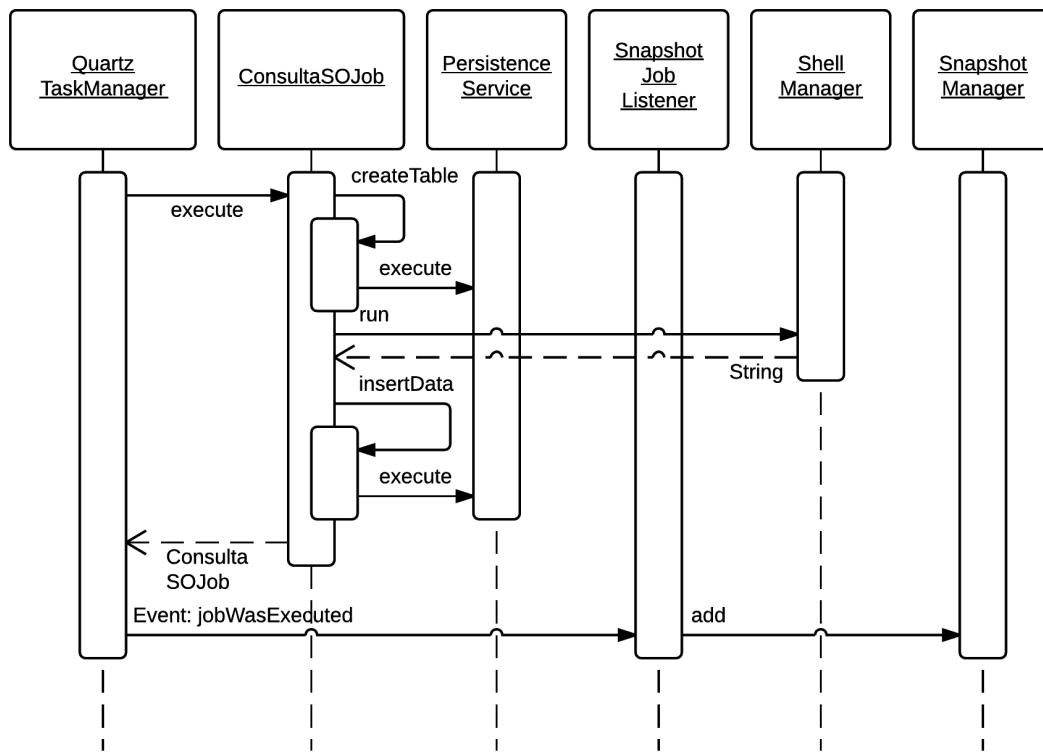


Ilustración 20 Diagrama de secuencia de Ejecutar Tarea

5.3.3 Caso de Uso Crear snapshot

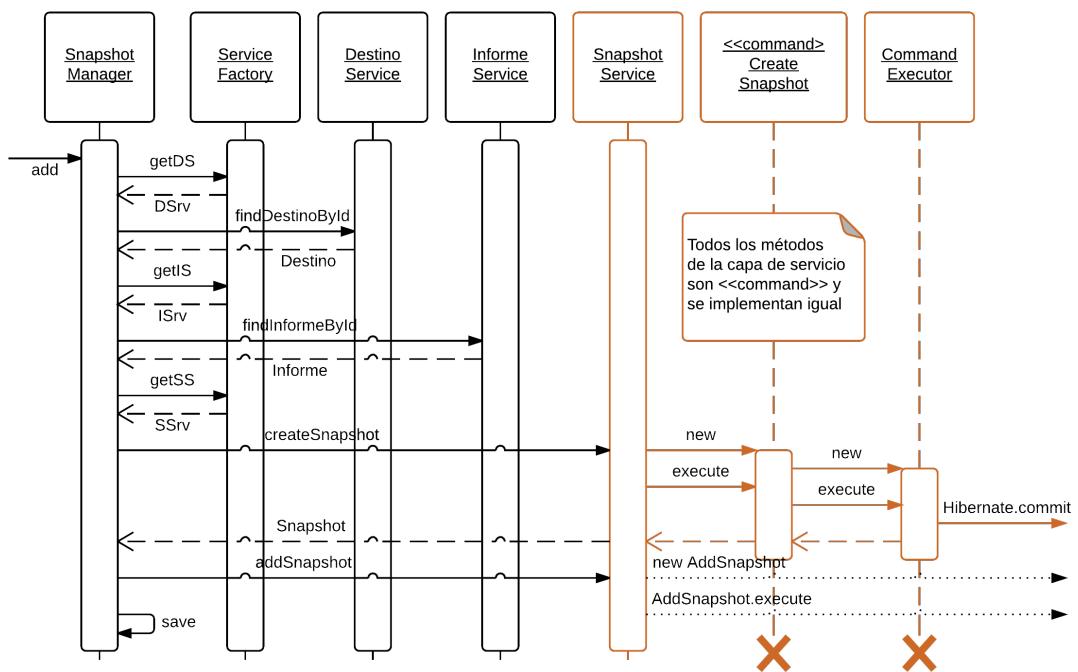


Ilustración 21 Diagrama de secuencia de Crear Snapshot

5.4 Diseño de la Base de Datos

Una parte fundamental de la aplicación es el almacenamiento de los datos en el servidor para su posterior explotación.

El servidor necesita una Base de Datos en modo servidor que almacene toda la información de los agentes y permita consultas desde herramientas externas. Los requisitos de volumen de datos o número de conexiones concurrentes no son altos, por lo que no se necesitan características especiales.

En los clientes, para almacenar temporalmente los datos recogidos es necesario una BBDD. Se ha optado por una BBDD embebida en el agente para facilitar la instalación y mantenimiento.

Ya que los datos que se envían entre el cliente y el servidor están fuertemente relacionados, se transmiten en ambas direcciones numerosas entidades. Para facilitar esta comunicación y las operaciones en la BBDD se opta por replicar el modelo de datos del servidor en el cliente y usar el mismo SGBD en los dos componentes.

Ante la complejidad del modelo se ha optado por usar JPA con Hibernate para facilitar su gestión.

5.4.1 Descripción del SGBD Usado

Tanto en el servidor como en los clientes se ha optado por usar HyperSQL DataBase. HSQLDB es un proyecto open source con una madurez contrastada. Soportado ampliamente por la comunidad Java y por las herramientas relacionadas, compite con Apache Derby y MySQL.

Cubre los requisitos mencionados, se integra perfectamente con Java, JDBC e Hibernate y tanto en la versión embebida como en la servidor ofrece un rendimiento destacable.

Como ya he mencionado, para facilitar el manejo de la BBDD se usará un ORM. Se ha seleccionado Hibernate por su gran adopción en Java y su soporte de SQLDB.

Tanto con el SGDB como con el ORM, solo se usan mediante las especificaciones estándar de Java JDBC y JPA, sin hacer uso de características específicas de la implementación. De esta manera es posible substituir el componente por otro sin grandes cambios.

5.4.2 Integración del SGBD en Nuestro Sistema

Tanto en el servidor como en el cliente se integra el SGDB de dos maneras:

El acceso al modelo de la aplicación se realiza mediante ORM. Definimos un conjunto de entidades que se anotan con JPA (paquete `es.uniovi.miw.monitora.server.model`). Para no acceder a la capa de persistencia directamente, se define una capa de servicios que nos permiten realizar las operaciones CRUD sobre la capa de persistencia (paquete `es.uniovi.miw.monitora.server.core`).

Para la gestión de los datos que se extraen de los sistemas del cliente tenemos que adoptar otro enfoque. En este caso se accede directamente al SGDB mediante JDBC, ya que las tablas y los datos cambian en función de las operaciones a realizar y no se pueden mapear a entidades.

Esta doble sistema de integración también tiene su reflejo en el envío de datos entre el cliente y el servidor. En el primer caso, cuando tratamos con entidades, el servicio web se encarga mediante JAX-RS/Jackson de de/serializar en JSON los datos de manera transparente y así sincronizar las entidades de los dos extremos. En el caso de los datos extraídos del cliente, se vuelcan las tablas a ficheros XML en el cliente y se envían mediante conexión segura y posteriormente el servidor importa los datos en XML en su BBDD.

5.4.3 Diagrama E-R

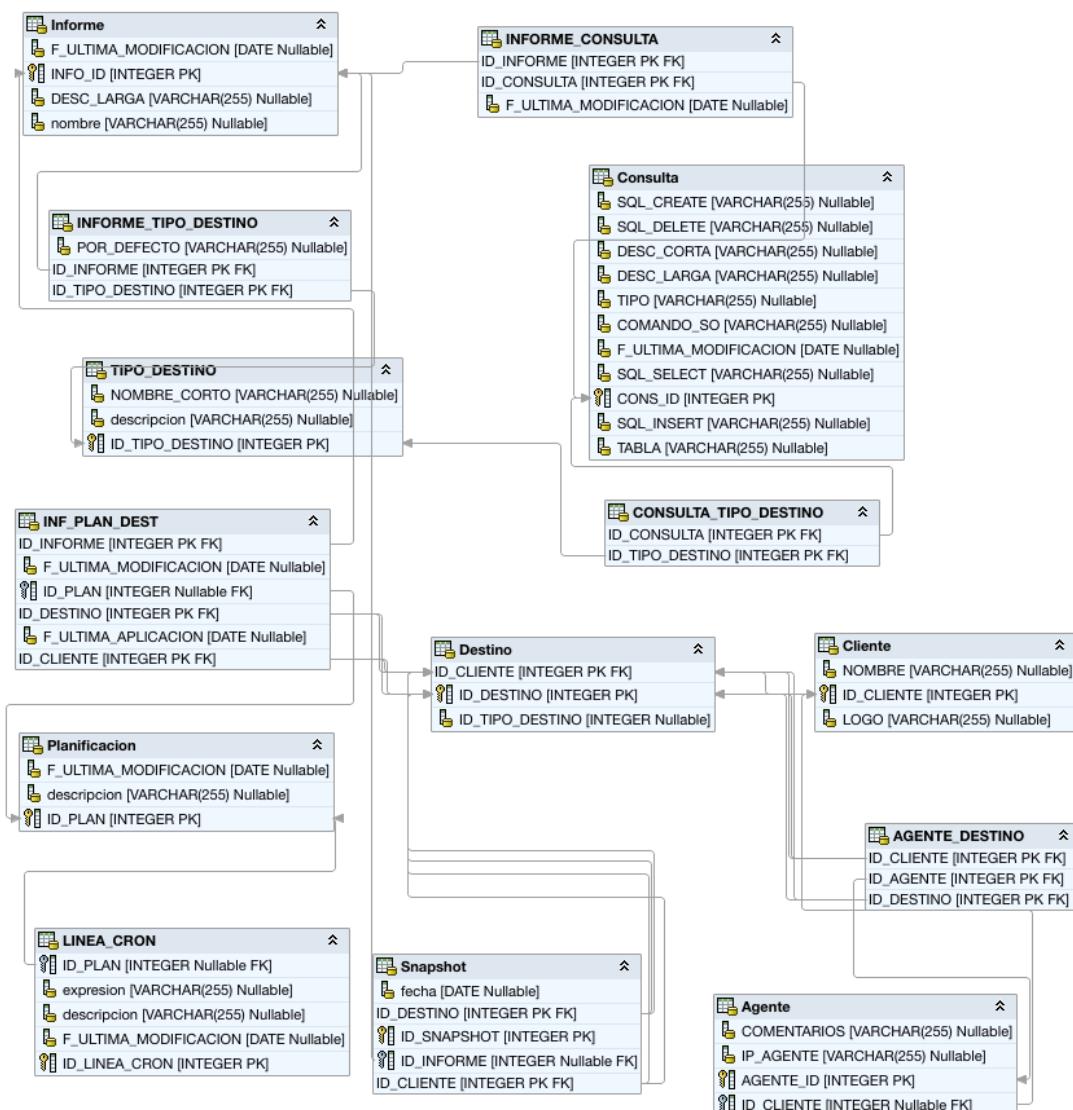


Ilustración 22 Diagrama Entidad-Relación del modelo de datos

5.5 Especificación Técnica del Plan de Pruebas

Para verificar el correcto funcionamiento del proyecto a desarrollar se realizarán distintas baterías de pruebas diferentes que se describen a continuación:

5.5.1 Pruebas Unitarias

Dependiendo de la complejidad de cada módulo se centra en un aspecto o otro.

En la capa de servicios insistimos en probar cada operación y comprobar que las relaciones de las entidades se mantienen en las dos direcciones.

En el servicio web se puede probar independientemente, sin conexión al servidor usando la biblioteca Jersey Test Framework. Se debe comprobar que el mapeo de las entidades a JSON es correcto.

La ejecución de tareas es difícil de probar ya que es asíncrona. En los test unitarios solo comprobamos que la tarea se ha creado o no, sin esperar a su ejecución

5.5.2 Pruebas de Integración y del Sistema

Partiendo de un estado conocido en la BBDD, se prueba especialmente la integración de la capa de servicios y la de persistencia. Después se prueba la transmisión de entidades entre el cliente y el servidor

Una vez construido el sistema completo se realizan las pruebas de sistema. Ayudándonos de una planificación conocida se puede evaluar los snapshots que el agente ha creado en el servidor

Caso de Uso: Ping	
Prueba	Resultado Esperado
Ping desde un agente no válido	El sistema lanza una excepción
Ping desde un agente válido	El sistema devuelve la hora de petición y OK

Caso de Uso: Recibir agente	
Prueba	Resultado Esperado
Pedir un agente no válido	El sistema lanza una excepción
Pedir un agente válido	El sistema devuelve el agente con toda su información y OK

Caso de Uso: Actualizar agente	
Prueba	Resultado Esperado
Actualizar agente no existente en el sistema	El sistema crea el agente con toda la información

Prueba	Resultado Esperado
Actualizar agente existente en el sistema	El sistema sobrescribe la información del agente

Caso de Uso: Planificar tarea	
Prueba	Resultado Esperado
Planificar tarea con expresión CRON incorrecta	Excepción
Prueba	Resultado Esperado
Planificar tarea con línea cron correcta	El contador de tareas se incrementa en 1 Existe una nueva orden de trabajo

Caso de Uso: Ejecutar tarea	
Prueba	Resultado Esperado
Tabla de resultados ya existe	No se crea una nueva. No se lanza excepción
Prueba	Resultado Esperado
Tabla de resultados no existe	Se crea una nueva tabla
Prueba	Resultado Esperado
Error en la orden a ejecutar	Se lanza excepción
Prueba	Resultado Esperado
Error al guardar los resultados de la orden en la tabla	Se lanza excepción

Caso de Uso: Crear snapshot	
Prueba	Resultado Esperado
Error al volcar tabla a XML	Se lanza excepción
Prueba	Resultado Esperado
No hay permisos de escritura en la carpeta	Se lanza excepción
Prueba	Resultado Esperado
Error al comprimir carpeta	Se lanza excepción
Prueba	Resultado Esperado
Error al crear fichero de metadatos	Se lanza excepción

Caso de Uso: Enviar ficheros	
Prueba	Resultado Esperado
Error al enviar fichero datos	Se lanza excepción

Caso de Uso: Enviar snapshot	
Prueba	Resultado Esperado
Error al enviar snapshot	Se lanza excepción

Caso de Uso: Enviar resultados	

Prueba	Resultado Esperado
Error al crear snapshot	Log de error + tratar siguiente resultado
Prueba	Resultado Esperado
Error al enviar fichero datos	Log de error + tratar siguiente resultado
Prueba	Resultado Esperado
Error al enviar snapshot	Log de error + tratar siguiente resultado

Caso de Uso: Responder ping	
Prueba	Resultado Esperado
Ping desde cliente no válido	Devolver error

Caso de Uso: Enviar agente	
Prueba	Resultado Esperado
Petición de agente no válido	Devolver error
Prueba	Resultado Esperado
Error de persistencia	Devolver error

Caso de Uso: Recibir fichero	
Prueba	Resultado Esperado
No hay permiso de escritura en carpeta	Devolver error
Prueba	Resultado Esperado
Error al descomprimir fichero	Devolver error
Prueba	Resultado Esperado
Error al insertar XML en tablas	Devolver error

Caso de Uso: Recibir snapshot	
Prueba	Resultado Esperado
Error de persistencia	Devolver error
Prueba	Resultado Esperado
Error al recibir fichero	Devolver error

Además de estas pruebas, también hay otras a bajo nivel que nos aseguran la construcción de un sistemas robusto desde el principio.

Para cada entidad el modelo de datos

Entidad de la capa de Servicios	
Prueba	Resultado Esperado
Creación entidad nula	Devolver error
Prueba	Resultado Esperado
Creacion entidad vacía	Devolver error
Prueba	Resultado Esperado
Creacion entidad	Entidad creada Id entidad nulo

Prueba	Resultado Esperado
Añadir entidad nula	Devolver error
Prueba	Resultado Esperado
Añadir entidad vacía	Devolver error
Prueba	Resultado Esperado
Añadir entidad	Id entidad no nulo Desde la entidad se puede navegar a todas las demás entidades relacionadas en las dos direcciones
Prueba	Resultado Esperado
Borrar entidad nula	Devolver error
Prueba	Resultado Esperado
Borrar entidad	La entidad no existe
Prueba	Resultado Esperado
Buscar entidad nula	Devolver error
Prueba	Resultado Esperado
Buscar entidad	Devuelve la entidad con el ID indicado Desde la entidad se puede navegar a todas las demás entidades relacionadas en las dos direcciones
Prueba	Resultado Esperado
Actualizar entidad	La entidad tiene los nuevos valores

Adicionalmente, una vez creadas todas las entidades, se prueba el funcionamiento de la creación y modificación en cascada, comprobando que se pueden navegar todas las relaciones.

Capítulo 6. Implementación del Sistema

6.1 Lenguajes de Programación

Para la realización del proyecto se ha usado **Java 7** como lenguaje de programación, **JSON** como lenguaje de intercambio de datos en el servicio web y **XML** para el contenido de las tablas con los resultados de las operaciones en el cliente. Para la capa de persistencia se ha usado **HQL** para las entidades del modelo de negocio y **SQL** para las consultas que extraen directamente datos del cliente. En el caso de SQL no se usa ninguna característica especial.

Esta elección de lenguajes no es arbitraria, Java nos permite cumplir con los objetivos de multiplataforma y robustez. El resto de lenguajes vienen ligados a las diferentes tecnologías empleadas.

La única elección se da al escoger JSON como lenguaje de intercambio de datos en el servicio web. La elección no es tanto de lenguaje como de tipo de servicio, se ha escogido un servicio web REST frente a uno basado en SOAP porque es menos verboso y actualmente en la mayoría de los grandes servicios se usa REST frente a SOAP. Una vez escogido el tipo de servicio, la elección del lenguaje viene dada. Aunque es posible publicar servicios web REST con XML, la mayoría lo hacen en JSON

6.2 Herramientas y Programas Usados para el Desarrollo

6.2.1 Bibliotecas de funciones

JUnit (v4.11) <http://junit.org/>

La biblioteca de pruebas unitarias más extendida en Java.

Mockito (v1.95) <http://mockito.org/>

Permite la creación de mockups, que facilitan en gran medida la realización de pruebas

Jersey (v2.10) <https://jersey.java.net/index.html>

Biblioteca para la implementación de servicios web REST en java

Jackson (versión incluida en el distribución de Jersey) <http://wiki.fasterxml.com/JacksonHome>

Biblioteca de manejo de JSON, que permite serializar/deserializar clases Java usando anotaciones

Quartz (v2.2.1) <http://quartz-scheduler.org/>

Biblioteca de scheduling con mayor soporte en Java, permite usar sintaxis CRON

Hibernate (v4.3.5) <http://hibernate.org/>

Framework ORM (Object/Relational Mapping) que implementa la especificación JPA

HSQldb (v2.3.2) <http://hsqldb.org/>

Base de Datos relacional escrita en java, con amplio soporte de la comunidad y totalmente compatible con SQL e Hibernate

DbUnit (v2.5.1) <http://dbunit.sourceforge.net/>

Extensión de JUnit para pruebas con Bases de Datos. En este proyecto se usa por el soporte que da para la lectura/escritura de tablas a formato XML

log4j (v1.2.17) <http://logging.apache.org/log4j/1.2/>

Biblioteca de logging creada por la Apache Software Foundation.

slf4j (v1.7.7) <http://www.slf4j.org/>

Biblioteca de logging que permite abstraerse de la biblioteca usada, cambiando de implementación. En el proyecto se usa como envoltorio sobre la biblioteca log4j

jsch (v 0.1.51) <http://www.jcraft.com/jsch/>

Implementación en java del protocolo SSH2. En el proyecto se usa para enviar ficheros mediante SCP al servidor

commons-exec (v1.2) <https://commons.apache.org/proper/commons-exec/>

Biblioteca para ejecutar procesos externos desde java.

jcommander (v1.30) <http://jcommander.org/>

Biblioteca para crear interfaces de línea de comandos.

6.2.2 Programas y herramientas

Eclipse <https://eclipse.org/>

Entorno de desarrollo integrado (IDE) usado para desarrollar el proyecto.

Maven <https://maven.apache.org/>

Herramienta para la administración de proyectos software. En el proyecto se usa para la gestión de dependencias entre bibliotecas

Github <https://github.com>

Repositorio de código basado en Git. Permite la gestión de código y la colaboración entre varias personas

Lucidchart <https://www.lucidchart.com/es>

Herramienta online para la creación de diagramas

Oracle Data Modeler

<http://www.oracle.com/technetwork/developer-tools/datamodeler/overview/index.html>

Herramienta gráfica para el desarrollo del modelo de datos, con soporte para SQL y diagramas E-R

6.3 Creación del Sistema

6.3.1 Problemas Encontrados

El principal problema viene de diseñar la aplicación partiendo del modelo de datos:

- La creación de las entidades y el mapeo mediante anotaciones JPA ha sido muy laborioso.
- La gran cantidad de relaciones entre entidades hace muy difícil la creación de la jerarquía de entidades y necesita de una gran cantidad de test unitarios. La mayor parte del código del servidor y de los errores encontrados están relacionados con este aspecto.
- Dificulta la serialización/deserialización a JSON en el servicio web y las pruebas.
- En JSON, las relaciones bidireccionales de las entidades generan ciclos. El manejo de estos ciclos depende de la implementación que haga el mapeador, con lo que el consumo del servicio web por otras implementaciones puede dar problemas.

Por otro lado han surgido numerosos interrogantes:

- Cuando se actualiza la configuración del agente ¿cómo se gestiona? ¿Se envían solo los cambios o toda la configuración? ¿Los resultados antiguos se borran o se mantiene? Cuándo una tarea se borra, ¿se borran los datos en el agente o todavía se pueden enviar?
- ¿Cómo repercute la ejecución asíncrona de tareas en el agente al envío de los resultados?

Muchas de estas preguntas vienen dadas por la existencia de la base de datos local en el agente y su sincronización con el servidor central. Se ha adoptado este enfoque para limitar el tráfico de red y la carga del servidor central ¿es la mejor opción?. La alternativa sería prescindir de esa BBDD local y enviar cada resultado individualmente. ¿Cómo afectaría al rendimiento? ¿Tiene sentido en este caso mantener el agente o sería superfluo y pasaríamos a una arquitectura similar a la de las alternativas citadas anteriormente?.

Capítulo 7. Desarrollo de las Pruebas

7.1 Pruebas Unitarias

Se han integrado las pruebas unitarias dentro del ciclo de desarrollo y siguiéndolas recomendaciones de las metodologías ágiles se han creado código y pruebas en paralelo.

En cada aplicación existen dos carpetas con código src/main/java con el código de la aplicación y src/test/java con el código de las pruebas.

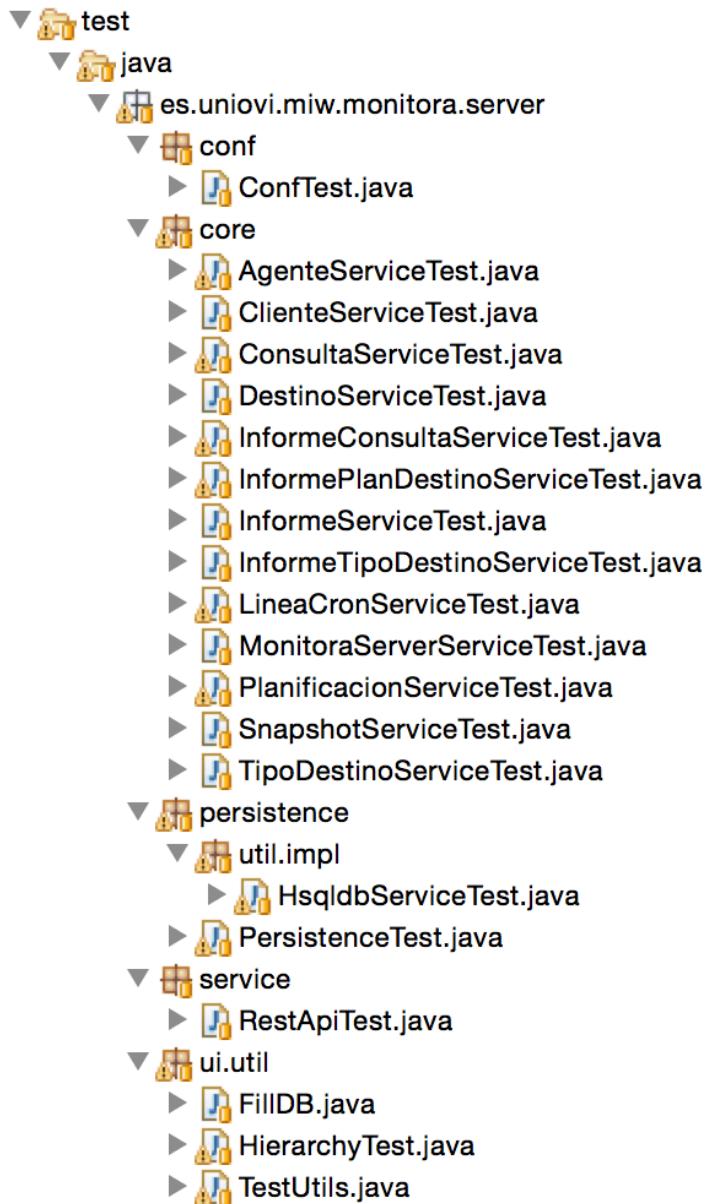


Ilustración 23 Pruebas unitarias del Servidor

La mayor parte de las pruebas corresponden a la capa de servicios del servidor, donde se ha probado exhaustivamente los siguientes casos

Entidad de la capa de Servicios	
Prueba	Resultado Esperado
Creación entidad nula	Devolver error
Prueba	Resultado Esperado
Creacion entidad vacía	Devolver error
Prueba	Resultado Esperado
Creacion entidad	Entidad creada Id entidad nulo
Prueba	Resultado Esperado
Añadir entidad nula	Devolver error
Prueba	Resultado Esperado
Añadir entidad vacía	Devolver error
Prueba	Resultado Esperado
Añadir entidad	Id entidad no nulo Desde la entidad se puede navegar a todas las demás entidades relacionadas en las dos direcciones
Prueba	Resultado Esperado
Borrar entidad nula	Devolver error
Prueba	Resultado Esperado
Borrar entidad	La entidad no existe
Prueba	Resultado Esperado
Buscar entidad nula	Devolver error
Prueba	Resultado Esperado
Buscar entidad	Devuelve la entidad con el ID indicado Desde la entidad se puede navegar a todas las demás entidades relacionadas en las dos direcciones
Prueba	Resultado Esperado
Actualizar entidad	La entidad tiene los nuevos valores

7.2 Pruebas de Integración y del Sistema

Para las pruebas unitarias y de sistema tenemos dos problemas que impiden su automatización

Todas la pruebas dependen del estado de la Base de Datos, por lo que debemos partir de un estado conocido y a partir de ahí comparar los resultados con lo esperado.

Las pruebas del agente relacionadas con la planificación de tareas son complejas, ya que el scheduler es asíncrono y se ejecuta según la planificación . La manera más sencilla de probar es comparar el contenido de las tablas generadas en la ejecución con el esperados

Caso de Uso: Ping		
Prueba	Resultado Esperado	Resultado Obtenido
Ping desde un agente no válido	El sistema lanza una excepción	El sistema lanza una excepción
Prueba	Resultado Esperado	Resultado Obtenido
Ping desde un agente válido	El sistema devuelve la hora de petición y OK	El sistema devuelve la hora de petición y OK

Caso de Uso: Recibir agente		
Prueba	Resultado Esperado	Resultado Obtenido
Pedir un agente no válido	El sistema lanza una excepción	El sistema lanza una excepción
Prueba	Resultado Esperado	Resultado Obtenido
Pedir un agente válido	El sistema devuelve el agente con toda su información y OK	El sistema devuelve el agente con toda su información y OK

Caso de Uso: Actualizar agente		
Prueba	Resultado Esperado	Resultado Esperado
Actualizar agente no existente en el sistema	El sistema crea el agente con toda la información	El sistema crea el agente con toda la información
Prueba	Resultado Esperado	Resultado Esperado
Actualizar agente existente en el sistema	El sistema sobrescribe la información del agente	El sistema sobrescribe la información del agente

Caso de Uso: Planificar tarea		
Prueba	Resultado Esperado	Resultado Obtenido
Planificar tarea con expresión CRON incorrecta	Excepción	Excepción
Prueba	Resultado Esperado	Obtenido
Planificar tarea con línea cron correcta	El contador de tareas se incrementa en 1	El contador de tareas se incrementa en 1

	Existe una nueva orden de trabajo	Existe una nueva orden de trabajo
--	-----------------------------------	-----------------------------------

Caso de Uso: Ejecutar tarea		
Prueba	Resultado Esperado	Resultado Obtenido
Tabla de resultados ya existe	No se crea una nueva. No se lanza excepción	No se crea una nueva. No se lanza excepción
Prueba	Resultado Esperado	Resultado Obtenido
Tabla de resultados no existe	Se crea una nueva tabla	Se crea una nueva tabla
Prueba	Resultado Esperado	Resultado Obtenido
Error en la orden a ejecutar	Se lanza excepción	Se lanza excepción
Prueba	Resultado Esperado	Resultado Obtenido
Error al guardar los resultados de la orden en la tabla	Se lanza excepción	Se lanza excepción

Caso de Uso: Crear snapshot		
Prueba	Resultado Esperado	Resultado Obtenido
Error al volcar tabla a XML	Se lanza excepción	Se lanza excepción
Prueba	Resultado Esperado	Resultado Obtenido
No hay permisos de escritura en la carpeta	Se lanza excepción	Se lanza excepción
Prueba	Resultado Esperado	Resultado Obtenido
Error al comprimir carpeta	Se lanza excepción	Se lanza excepción
Prueba	Resultado Esperado	Resultado Obtenido
Error al crear fichero de metadatos	Se lanza excepción	Se lanza excepción

Caso de Uso: Enviar ficheros		
Prueba	Resultado Esperado	Resultado Obtenido
Error al enviar fichero datos	Se lanza excepción	Se lanza excepción

Caso de Uso: Enviar snapshot	
Prueba	Resultado Esperado
Error al enviar snapshot	Se lanza excepción

Caso de Uso: Enviar resultados		
Prueba	Resultado Esperado	Resultado Obtenido
Error al crear snapshot	Log de error + tratar siguiente resultado	Log de error + tratar siguiente resultado
Prueba	Resultado Esperado	Resultado Obtenido

Error al enviar fichero datos	Log de error + tratar siguiente resultado	Log de error + tratar siguiente resultado
Prueba	Resultado Esperado	Resultado Obtenido
Error al enviar snapshot	Log de error + tratar siguiente resultado	Log de error + tratar siguiente resultado

Caso de Uso: Responder ping		
Prueba	Resultado Esperado	Resultado Obtenido
Ping desde cliente no válido	Devolver error	Devolver error

Caso de Uso: Enviar agente		
Prueba	Resultado Esperado	Resultado Obtenido
Petición de agente no válido	Devolver error	Devolver error
Prueba	Resultado Esperado	Resultado Obtenido
Error de persistencia	Devolver error	Devolver error

Caso de Uso: Recibir fichero		
Prueba	Resultado Esperado	Resultado Obtenido
No hay permiso de escritura en carpeta	Devolver error	Devolver error
Prueba	Resultado Esperado	Resultado Obtenido
Error al descomprimir fichero	Devolver error	Devolver error
Prueba	Resultado Esperado	Resultado Obtenido
Error al insertar XML en tablas	Devolver error	Devolver error

Caso de Uso: Recibir snapshot		
Prueba	Resultado Esperado	Resultado Obtenido
Error de persistencia	Devolver error	Devolver error
Prueba	Resultado Esperado	Obtenido
Error al recibir fichero	Devolver error	Devolver error

Capítulo 8. Manuales del Sistema

8.1 Manual de Instalación

8.1.1 Requisitos previos

Tomcat 7 (<https://tomcat.apache.org/download-70.cgi>) para desplegar el servidor

eclipse-jee (<http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/marsr>)

servidor ssh (consultar el manual del Sistema Operativo)

El primer paso es descargar el código fuente e importar los tres proyectos en eclipse. Una vez importados, el soporte de Maven en eclipse hace el resto (importa todas las dependencias)

8.1.2 Servidor

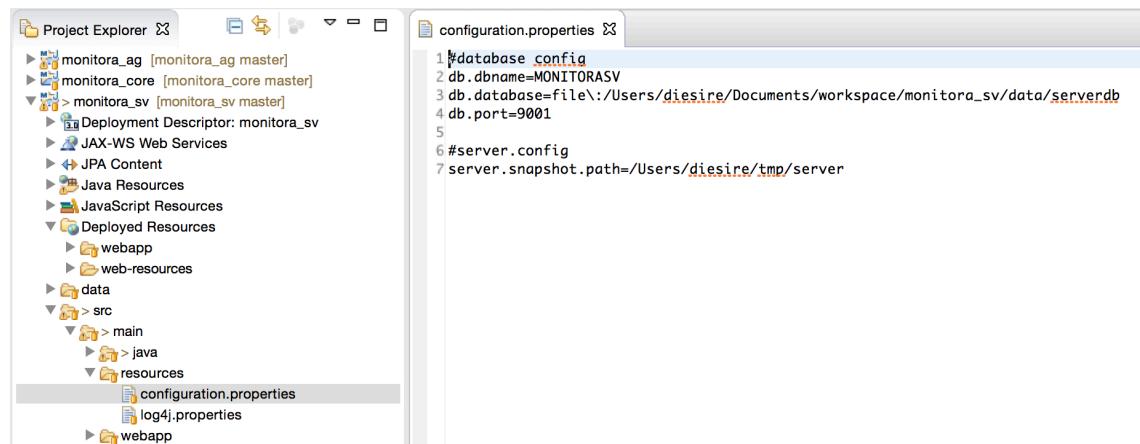
Configurar el servidor editando del fichero configuration.properties:

La conexión a la base de datos

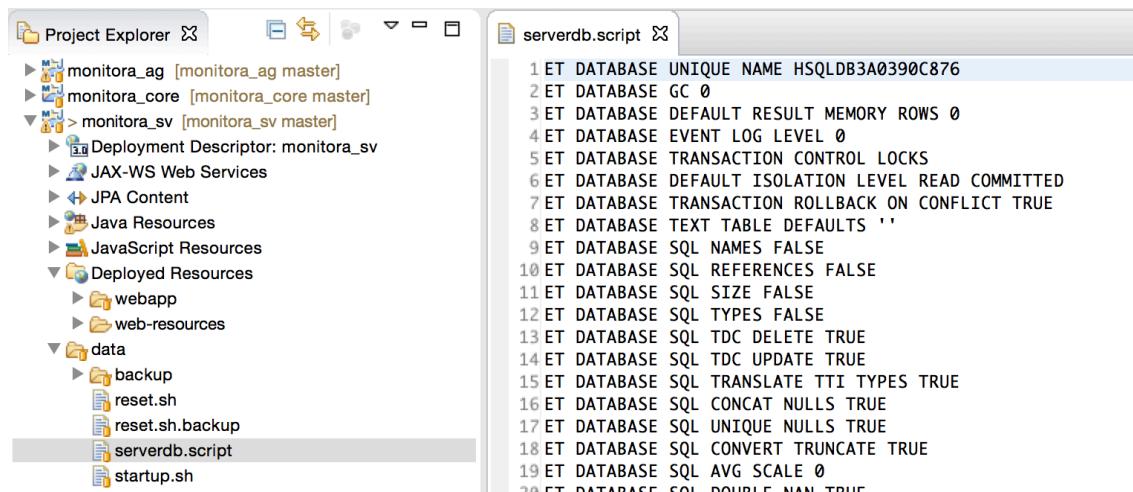
```
#database config
db dbname=MONITORASV
db database=file\:/Users/user/tmp/monitora_sv/data/serverdb
db port=9001
```

El directorio donde se reciben los snapshots mediante SCP

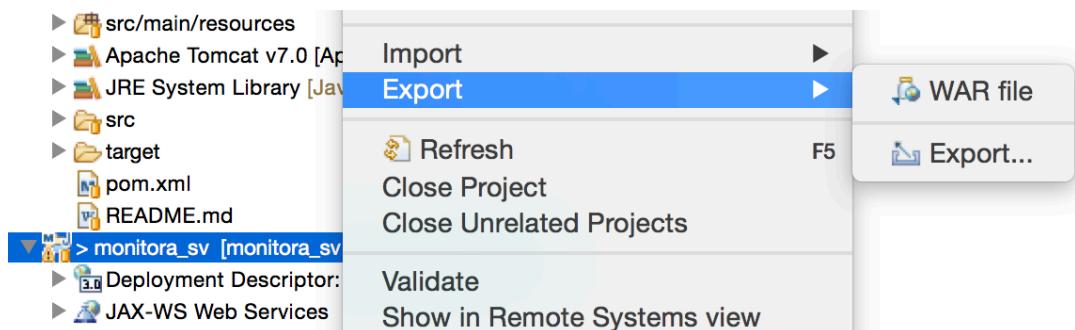
```
#server.config
server.snapshot.path=/Users/user/tmp/server
```



Para configurar la Base de Datos se debe copiar el fichero serverdb.script en la carpeta indicada anteriormente

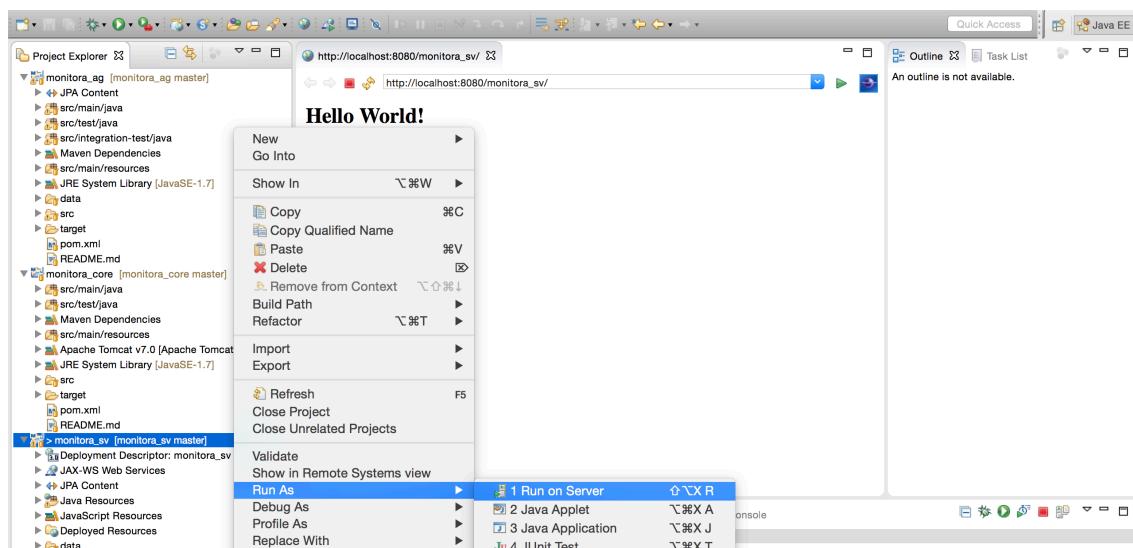


Finalmente, para obtener el fichero WAR que desplegaremos en Tomcat podemos buscarlo en los directorios creados por Maven o generararlo desde el proyecto eclipse.



El fichero WAR se puede desplegar sobre un servidor Tomcat simplemente copiándolo en el directorio webapps.

Alternativamente, para una prueba rápida también se puede ejecutar la aplicación sobre un Tomcat instalado en eclipse con un clic



8.1.3 Agente

En el agente también tenemos que configurar varias cosas:

La conexión a la BBDD del agente

```
#database config
db dbname=monitoraag
db database=file\:/data/clientdb
db port=9002
```

Directorio en el servidor donde se depositan los ficheros de datos mediante SCP

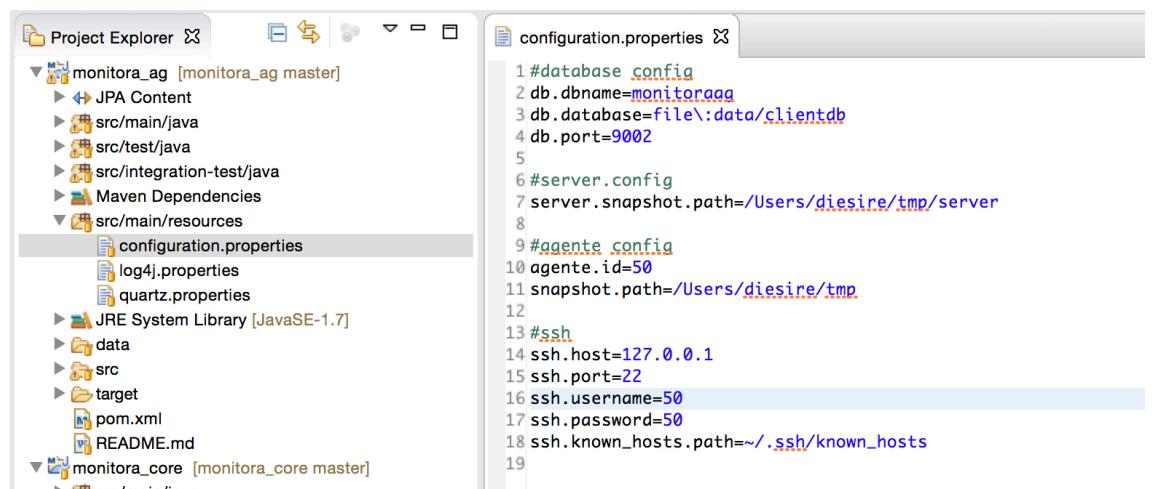
```
#server.config
server.snapshot.path=/Users/diesire/tmp/server
```

Identificador del agente (se puede consultar en la BBDD del repositorio) y directorio temporal donde se encuentran los ficheros de datos antes de enviarlos

```
#agente config
agente.id=50
snapshot.path=/Users/diesire/tmp
```

Configuración del cliente SSH (Antes ejecutar la aplicación, probar la conexión para generar la entrada correspondiente en el directorio known_hosts)

```
#ssh
ssh.host=127.0.0.1
ssh.port=22
ssh.username=user
ssh.password=password
ssh.known_hosts.path=~/ssh/known_hosts
```



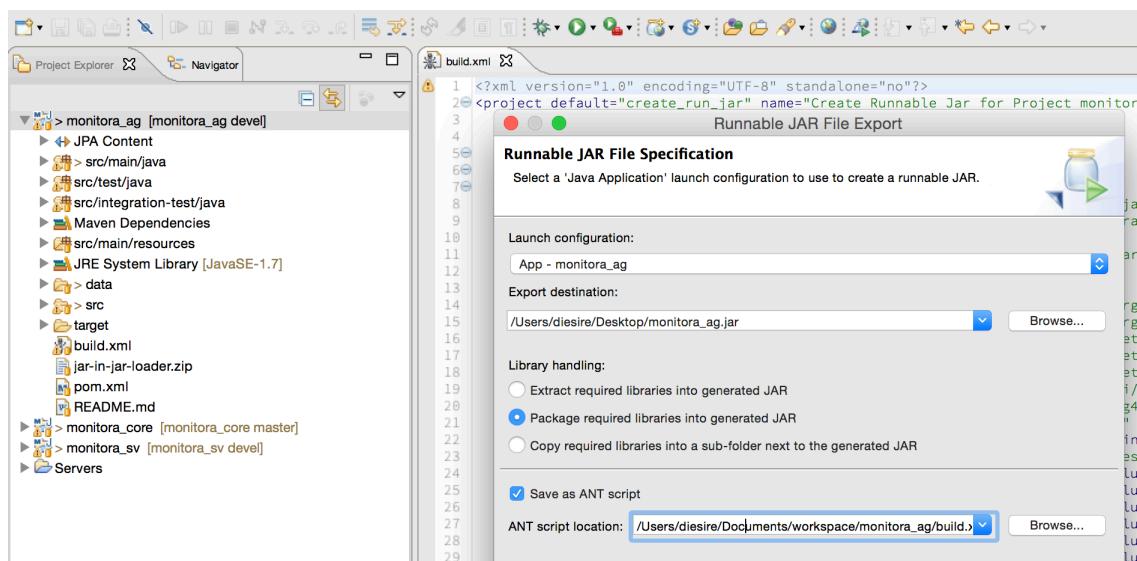
Para configurar la Base de Datos se debe copiar el fichero clientdb.script en la carpeta indicada anteriormente

```

1 SET DATABASE UNIQUE NAME HSQldb3A0390C876
2 SET DATABASE GC 0
3 SET DATABASE DEFAULT RESULT MEMORY ROWS 0
4 SET DATABASE EVENT LOG LEVEL 0
5 SET DATABASE TRANSACTION CONTROL LOCKS
6 SET DATABASE DEFAULT ISOLATION LEVEL READ COMMITTED
7 SET DATABASE TRANSACTION ROLLBACK ON CONFLICT TRUE
8 SET DATABASE TEXT TABLE DEFAULTS ''
9 SET DATABASE SQL NAMES FALSE
10 SET DATABASE SQL REFERENCES FALSE
11 SET DATABASE SQL SIZE FALSE
12 SET DATABASE SQL TYPES FALSE
13 SET DATABASE SQL TDC DELETE TRUE
14 SET DATABASE SQL TDC UPDATE TRUE
15 SET DATABASE SQL TRANSLATE TTI TYPES TRUE
16 SET DATABASE SQL CONCAT NULLS TRUE
17 SET DATABASE SQL UNIQUE NULLS TRUE

```

Para obtener el fichero ejecutable podemos exporta el proyecto de eclipse a un fichero JAR usando el asistente o ejecutando la tarea ANT del fichero build.xml



8.2 Manual de Ejecución

Lo única tarea que hay que hacer para ejecutar el servidor es arrancar Tomcat. En dirección http://<HOSTNAME>:8080/monitora_sv/ se encuentra la aplicación.

8.2.1 Agente

El agente se ejecuta desde línea de comandos, la **ayuda** muestra las opciones disponibles

```
host:Desktop user$ java -jar monitora_ag.jar --help
Usage: monitora_ag [options]
Options:
  --debug
    Debug mode
    Default: false
  -h, --help
    Print this usage message
    Default: false
  -i, --interactive
    Interactive mode
    Default: false
```

Por **defecto**, si no se indica ninguna acción arranca en modo servicio

```
host:Desktop user$ java -jar monitora_ag.jar
To close normally, use [Ctrl]+[C]
```

Con la opción de **debug** se puede ver información detallada del proceso

```
host:Desktop user$ java -jar monitora_ag.jar --debug
JVM Name = 2049@host.local
JVM PID   = 2049
Peak Thread Count = 9
To close normally, use [Ctrl]+[C]
```

En modo **interactivo** podemos ejecutar los comandos del agente manualmente

```
host:Desktop user$ java -jar monitora_ag.jar -i
MonitORA agent interactive mode. To see commands type 'help'
>help
Interactive commands:
  start      Start agent
  ping       Check if server is online
  update     Update agent by retrieving server info
  dump       Send agent information to server
  exit       Exit agent
  debug      Show app info
  help       Print this usage message
>exit
host:Desktop user$
```

Capítulo 9. Conclusiones y Ampliaciones

9.1 Conclusiones

En este proyecto se ha desarrollado un sistema que permite monitorizar el funcionamiento de múltiples Bases de Datos Oracle.

Los objetivos iniciales se han cumplido. Tenemos un sistema distribuido, multiplataforma, con una arquitectura modular que integra de manera coherente diversas tecnologías con el que mediante consultas SQL y herramientas de línea de comandos podemos monitorizar de forma remota varias Bases de Datos Oracle simultáneamente.

Las elecciones tecnológicas han puesto de manifiesto las ventajas e inconvenientes que ya conocíamos desde el principio. Hay dos puntos que destacar:

- El agente. La instalación de un agente en el servidor de BBDD del cliente no es trivial, presenta problemas de confianza, seguridad, sobrecarga del servidor, ... sin embargo simplifica enormemente el desarrollo, ya que usa tecnologías estándar y abstrae del Sistema Operativo y la versión de la Base de Datos presentes en el host.
- El servicio web. Para la comunicación entre los agentes y el servidor se ha implementado un servicio web REST para la configuración del agente y un servidor SSH el envío de los datos recuperados. Frente a otras alternativas como EJB, el servicio web no está sujeto a la plataforma Java y se puede consumir desde cualquier plataforma, pero dado el modelo de datos tan rico que se implementa, añade complejidad al desarrollo.

A lo largo del desarrollo se han encontrado varios casos límite que elevan la complejidad del desarrollo exponencialmente:

- Cuando se actualiza la configuración del agente ¿cómo se gestiona? ¿Se envían solo los cambios o toda la configuración? ¿Los resultados antiguos se borran o se mantiene? Cuándo una tarea se borra, ¿se borran los datos en el agente o todavía se pueden enviar?
- ¿Cómo repercute la ejecución asíncrona de tareas en el agente al envío de los resultados?

Algunas de estas cuestiones se resuelven según la elección tecnológica adoptada. Otras, sin embargo, quedan sin resolver ya que van en contra del objetivo inicial de sencillez.

En resumen, teniendo en cuenta el alcance del proyecto y los objetivos logrados se puede decir que los resultados obtenidos están dentro de lo esperado.

9.2 Ampliaciones

9.2.1 Generación de informes

Crear una nueva aplicación que consuma los datos almacenados en el servidor central para mostrar de forma gráfica información sobre los sistemas monitorizados

¿Cómo implementarlo? Mediante una aplicación web independiente que acceda directamente a la Base de Datos del servidor central.

¿Por qué no se ha incluido? Excede el alcance inicial del proyecto

¿Qué aporta? Información visual sobre el estado del servidor. Se pueden crear varias vistas distintas para diferentes perfiles, una sencilla para el cliente y otra más completa para el administrador de BBDD.

9.2.2 Envío de alertas

Crear un módulo que analice la información recopilada del sistema destino y según unos parámetros configurables notifique a los usuarios de determinados eventos.

¿Cómo implementarlo? Mediante un nuevo módulo del servidor o una aplicación independiente que acceda directamente a la Base de Datos del servidor central. Las notificaciones pueden ser por email o añadirse a la aplicación anterior de Generación de Informes

¿Por qué no se ha incluido? Excede el alcance inicial del proyecto

¿Qué aporta? Valor añadido a los datos recopilados. No solo los muestra, sino que los interpreta y “señala” información relevante.

9.2.3 Ampliación del servicio web

Crear un módulo que permita la consulta/modificación de los datos recopilados de los clientes mediante una extensión del servicio web existente para no acceder directamente a la Base de Datos

¿Cómo implementarlo? Mediante un nuevo módulo del servidor que publique una API que permita acceder a los datos de la Base de Datos del servidor central.

¿Por qué no se ha incluido? Excede el alcance inicial del proyecto y los datos recopilados del cliente son heterogéneos lo que dificulta la tarea

¿Qué aporta? Facilidad para explotar los datos recopilados, una capa de abstracción intermedia entre los datos y los consumidores, mayor seguridad en el servidor al limitar el

acceso a los datos y se podría llegar a enviar los datos de los agentes mediante esta ampliación, en lugar de mediante el servidor SSH

Capítulo 10. Presupuesto

10.1 Gastos

Dentro de las tarifas de cada perfil se incluyen los gastos corrientes (inmueble, agua, luz, limpieza, internet, teléfono, material de oficina, equipos informáticos, salarios de los empleados, seguridad social, etc.)

Además en este proyecto se incluyen varios recursos especiales que se facturan a parte (Servidor Repositorio y Servidor Centro Control, que incluyen el hardware y las licencias necesarias).

Teniendo en cuenta que los tiempos de las tareas se sobreestiman, el 20% para el fondo de imprevistos se extrae de las tarifas de los recursos.

Nombre del recurso	Tipo	Etiqueta de material	Iniciales	Grupo	Capacidad máxima	Tasa estándar	Costo/Uso
Jefe de Proyecto	Trabajo		JP		20%	75,00 €/hora	0,00 €
Analista	Trabajo		An		50%	40,00 €/hora	0,00 €
Programador 1	Trabajo		P1		100%	25,00 €/hora	0,00 €
Programador 2	Trabajo		P2		100%	25,00 €/hora	0,00 €
Administrador Sistemas	Trabajo		Admin		100%	30,00 €/hora	0,00 €
Servidor Repositorio	Material		Repo			100,00 €	0,00 €
Servidor Centro Control	Material		C3			100,00 €	0,00 €
Salón actos	Material		SIA			0,00 €	300,00 €
Sala reuniones	Material		SI			0,00 €	100,00 €

Los costes resumidos de las tareas más importantes son los siguientes

Nombre de tarea	Nivel de esquema	Duración	Costo
Análisis		11,5 días	340,00 €
Diseño		15,5 días	1.350,00 €
Configuración entorno		13,63 días	880,00 €
Entrega		141,25 días	2.085,00 €
Control		241,25 días	740,00 €
Riesgos		339,13 días	105,00 €
Calidad		339,13 días	105,00 €
Finalización		32 horas	530,00 €
Milestone 1		21 hora	120,00 €

Milestone 2	21 hora	120,00 €
Milestone3	21 hora	345,00 €
Revisión final	21 hora	320,00 €
Despliegue	21 día	440,00 €
Implementación	119,5 días	6.700,00 €
Módulo recogida datos	22 días	600,00 €
Módulo transmisión	27 días	1.400,00 €
Módulo control	23,5 días	1.300,00 €
Servidor	24 días	1.400,00 €
Cliente web	23 días	2.000,00 €
Documentación	124,5 días	7.388,33 €

10.2 Presupuesto cliente

A importe del desarrollo se le suma un 20% en concepto de beneficios.

Item	Concepto	Unidades	Importe	Total
1	Software			
1.1	Servidor	42	195,50 €	8.211,00 €
1.2	Agente	72	195,50 €	14.076,00 €
				22.287,00 €
	21% IVA			4.680,27 €
Total				26.967,27 €

Capítulo 11. Referencias Bibliográficas

[GoF94] Design Patterns: Elements of Reusable Object-Oriented Software (ISBN 0-201-63361-2)

[Fowler03] UML Distilled: A Brief Guide to the Standard Object Modeling Language (ISBN 078-5342193688)

[ApacheSF15] “Apache Tomcat 7” <http://tomcat.apache.org/tomcat-7.0-doc/index.html>

[Gunasekara13] “RESTful Web Services with Java” <http://java.dzone.com/articles/restful-web-services-java>

[Hibernate12] “An entity copy was already assigned to a different entity”
<https://hibernate.atlassian.net/browse/HHH-7605>

[HSQLDevGroup15] “HyperSQL User Guide” <http://hsqldb.org/doc/guide/index.html>

[Macdonald14] “Creating a Java REST API with Jersey”
<http://blog.aetherstore.com/2014/02/rest-api-with-jersey/>

[RedHat04] “Hibernate ORM documentation - Chapter 2. Mapping Entities”
<https://docs.jboss.org/hibernate/annotations/3.5/reference/en/html/entity.html>

[SoftwareAG14] “Quartz Scheduler 2.2.x Online Documentation Library” <http://quartz-scheduler.org/generated/2.2.1/html/qs-all/>

[Stackoverflow12] “Setup ShutdownHook and exit application”
<http://stackoverflow.com/questions/8897643/setup-shutdownhook-and-exit-application>

[Stackoverflow13] “How do I merge two detached entities that are equals() but not identical?” <http://stackoverflow.com/questions/20238647/how-do-i-merge-two-detached-entities-that-are-equals-but-not-identical>

[Van den Bogaert14] “Trapping CTRL-C in a Java app”. <http://esus.com/trapping-ctrl-c-java-app/>

Capítulo 12. Apéndices

12.1 Contenido Entregado en Anexos

12.1.1 Contenidos

Carpeta Código

- **monitora_ag-master.zip** Código fuente del agente
- **monitora_core-master.zip** Código fuente de las clases de utilidades del servicio web
- **monitora_sv-master.zip** Código fuente del servidor

Carpeta Planificación

- **planificacion.mpp** Planificación del PFM en Microsoft Project