UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

MOPROM: Módulo de Pruebas de Operaciones Multicanales

Autor: Alberto Prieto Lozano

Tutor: Valentín Moreno Pelayo

Tutor de empresa: Luis Mariano Celada Díaz

ÍNDICE

1. INTRODUCCIÓN 7

1.1 Objetivos 7

1.2 Procedimiento 8

2. ESTADO DEL ARTE 9

3. PROPUESTA 10

3.1 Procedimiento 10

3.2 Sistema MOPROM 11

3.3 Tecnologías y herramientas de desarrollo 12
3.3.1 Herramientas Software 12
3.3.2 Herramientas Hardware 14

4. ANÁLISIS DEL SISTEMA 15

4.1 Descripción del catálogo de requisitos 15

4.2 Requisitos de usuario 17
4.2.1 Requisitos de capacidad 17
4.2.2 Requisitos de restricción 22

4.3 Modelo de Casos de Uso 24
4.3.1 Casos de uso 25

4.4 Requisitos software 29
4.4.1 Requisitos Funcionales 30
4.4.2 Requisitos No Funcionales 35
4.4.2.1 Plataforma 35
4.4.2.2 Interfaz 35
4.4.2.3 Escalabilidad 38
4.4.2.4 Rendimiento 38
4.4.2.5 Disponibilidad 38
4.4.2.6 Seguridad 38
4.4.2.7 Extensibilidad 38

4.5 Arquitectura física del sistema 39

5. DISEÑO DEL SISTEMA 41

5.1 Diseño y arquitectura del sistema 41
5.1.1 Arquitectura distribuida 41

5.1.2 Modelo de datos 44
5.1.3 Modelo lógico 47
5.1.4 Modelo Visual 48

5.2 Diseño y descripción de componentes 49
5.2.1 Componente Principal 50
5.2.1.1 Principal 50
5.2.2.1 SplashWindow 50
5.2.2 Componente Ventana 51
5.2.2.1 VentanaPrincipal 51
5.2.2.2 VentanaEditorXML 52
5.2.2.3 VentanaEditorEntorno 53
5.2.3 Componente Bean 54
5.2.3.1 RegistroBean 54
5.2.3.2 EntornoBean 54
5.2.3.3 NombreRutaBean 54
5.2.4 Componente util 55
5.2.4.1 Validacion 55
5.2.4.2 Parser 56
5.2.4.3 CreadorXML 56
5.2.4.4 Peticiones 56

5.3 Diagramas de actividad 58

5.4 Diagramas de secuencia 60

6. IMPLEMENTACIÓN DEL SISTEMA 66

6.1 MOPROM: un vistazo rápido 66

6.2 Características del programa 67

6.3 Interfaz del programa 68

6.4 ¿Cómo funciona? 69
6.4.1 Arranque del programa 69
6.4.2 Organización de las pruebas 72
6.4.3 Lanzamiento de pruebas 74

7. PLANIFICACIÓN 75

7.1 Descripción 75

7. 2 Fases y duración del proyecto 76

7.3 Diagrama de Gantt 77

8. PRESUPUESTO Y ESTIMACIÓN DE COSTES 78

8.1 Descripción 78

8.2 Gastos Recursos Humanos 79

8.3 Gastos Hardware..... 80

8.4 Gastos Software..... 80

8.5 Otros Gastos 81

8.6 Resumen del presupuesto del proyecto 81

9. CONCLUSIONES 82

10. DEFINICIONES, ABREVIATURAS Y ACRÓNIMOS..... 84

11. BIBLIOGRAFÍA 90

12. ANEXOS 91

12.1 JAVADOC 91

12.2 Fichero LOG..... 93

12.3 Manual de usuario..... 94

ÍNDICE DE FIGURAS

Figura 1: Tabla base requisitos. 16

Figura 2: Diagrama de Casos de Uso 24

Figura 3: Tabla base requisitos software. 29

Figura 4: Arquitectura física de MOPROM. 39

Figura 5: Diagrama arquitectura de tres capas. 42

Figura 6: Esquema XSD (Vista global). 44

Figura 7: Esquema XSD (Desglosado). 45

Figura 8: Código fuente plantilla XSD. 46

Figura 9: Diagrama de clases. 47

Figura 10: Prototipo interfaz. 48

Figura 11: Jerarquía de paquetes. 49

Figura 12: Imagen de carga de MOPROM. 50

Figura 13: Ventana principal. 52

Figura 14: Ventana Editor XML. 53

Figura 15: Ventana Selector Entorno. 53

Figura 16: Diagrama actividad - Lanzar Prueba. 58

Figura 17: Diagrama Actividad - Editar Prueba. 59

Figura 18: Diagrama de Secuencia - Lanzar Prueba. 61

Figura 19: Diagrama de Secuencia - Editar Prueba. 63

Figura 20: Diagrama de Secuencia - Editar Trama. 64

Figura 21: Diagrama de Secuencia - Guardar Resultado. 65

Figura 22: Interfaz de usuario (Ventana Principal). 68

Figura 23: Interfaz de usuario (Editor XML) 68

Figura 24: Fichero "config.ini" 69

Figura 25: Diagrama de Actividad - Iniciar Aplicación. 70

Figura 26: Diagrama de Secuencia - Iniciar Aplicación. 71

Figura 27: Configuración directorios MOPROM. 72

Figura 28: Configuración directorios tramas. 73

Figura 29: Fichero ejemplo XML. 73

Figura 30: Fases y duración del proyecto. 76

Figura 31: Diagrama de Gantt 77

Figura 32: Tabla presupuesto RRHH desglosado. 79

Figura 33: Tabla resumen equipos informáticos. 80

Figura 34: Tabla resumen programas software. 80

Figura 35: Tabla resumen otros gastos. 81

Figura 36: Tabla resumen presupuesto proyecto. 81

Figura 37: Índice JavaDoc. 91

Figura 38: Jerarquía de paquetes JavaDoc. 92

Figura 39: Sumario de métodos JavaDoc. 92

Figura 40: Ejemplo fichero "moprom.log". 93

Figura 41: Ejemplo fallo y registro en LOG. 93

1. INTRODUCCIÓN

En ocasiones los departamentos se encuentran sometidos al mantenimiento y revisión de muchos y distintos sistemas y aplicaciones. En el día a día es frecuente el error, mal funcionamiento o cualquier otro tipo de problemática en estos sistemas, produciéndose pérdidas de tiempo en la realización de estas pruebas o "test" sobre las aplicaciones.

Una mala documentación sobre las aplicaciones mantenidas, o simplemente el desconocimiento del funcionamiento al 100% del sistema por todos y cada uno de los componentes del equipo de trabajo hacen que cualquier incidencia o prueba sobre estos sistemas se traduzca en un mal funcionamiento del conjunto del equipo. Decenas de consultas entre los miembros del equipo se suceden hasta dar con la persona experta en el sistema y su entorno, lo que supone un tiempo de trabajo perdido al no estar documentados ni organizados.

1.1 Objetivos

El objetivo principal de este proyecto fin de carrera es el análisis, diseño, desarrollo e implementación de un sistema de software de gestión del proceso de pruebas de las distintas aplicaciones administradas por el departamento de Banca Telefónica de CajaMadrid.

El programa se llamará MOPROM (Módulo de pruebas de operaciones multicanales) y mediante este sistema se pretende unificar las pruebas de los distintos sistemas en un mismo aplicativo, dando lugar a un marco de trabajo único para estos procesos que todos los usuarios utilizarán. Aprovechando que crearemos un sistema centralizado, intentaremos también automatizar en lo posible el proceso, pues dispondremos de información tal como los usuarios participantes, sus acciones o la configuración de las distintas aplicaciones mantenidas por el departamento.

Todo el proceso de elaboración del proyecto fin de carrera estará centrado en la unificación de las distintas pruebas y por ello obtendremos un sistema fácil de configurar y de sencilla utilización por parte de los usuarios. Se trata de crear una plataforma sencilla para la elaboración de las distintas pruebas pertinentes de forma muy ágil.

Tampoco hay que olvidar que la empresa es un entorno muy dinámico en el que constantemente aparecen nuevas aplicaciones y sistemas, lo que agrava la situación y puede incrementar el descontrol. De esta forma tratamos de crear una plataforma con capacidad adaptativa a los nuevos sistemas, además de mantener organizados todos los sistemas antiguos.

1.2 Procedimiento

Para conseguir los objetivos propuestos, seguiremos los siguientes pasos:

- Definición esquemática de los objetivos principales, y el alcance del sistema.
- Análisis de los requisitos dados por el cliente, y relación de los mismos con los objetivos y alcance marcados.
- Diseño de los diferentes módulos que componen el sistema de forma que se cumplan los requisitos establecidos.
- Implementación de los módulos hasta obtener partes completamente funcionales que cumplan los requisitos de forma eficaz.
- Pruebas de cara a comprobar la plena funcionalidad del sistema y base sobre la que obtener conclusiones.
- Análisis de las posibles mejoras y limitaciones resultantes de nuestro diseño e implementación.

2. ESTADO DEL ARTE

Actualmente el departamento no dispone de ningún software ni procedimiento concreto para realizar las pruebas. Éstas se realizan desde distintos entornos y algunas están poco o nada documentadas de forma que es difícil saber incluso entre los miembros del propio departamento como acceder a ellas, teniendo en ocasiones que consultar vía telefónica o e-mail a otros miembros del equipo para hallar el procedimiento adecuado.

Asimismo para las pruebas individuales de cada sistema no hay una base de datos con ejemplos o plantillas de pruebas a realizar a cada sistema, de forma que en ocasiones hay que hacerlos a mano.

En cuanto al procedimiento de las pruebas, como se trata de aplicaciones corporativas muy concretas no existe un software de terceros que pueda englobar todo el abanico de pruebas y plantillas, que es lo que se espera de MOPROM, ya que el estado actual del sistema es muy caótico y totalmente desorganizado.

El método de trabajo del departamento, sus aplicaciones mantenidas, las aplicaciones de terceros, y los protocolos utilizados para su comunicación interna son muy concretos y personalizados, por lo tanto, para que una aplicación se pueda adaptar a ese método de trabajo, hay que poner en orden todas las pruebas que se quieran realizar y analizar qué tipo de accesos por red tendrán y su complejidad, además desarrollar una buena plantilla que sirva de base para poder catalogar todas las aplicaciones. La finalidad de MOPROM es la de proporcionar una plataforma que aúne toda la información posible sobre los servicios y las aplicaciones, no solamente una plataforma de pruebas.

Debido a la complejidad de las especificaciones, puedo afirmar que no existe software de terceros que pueda realizar las tareas requeridas para satisfacer todos y cada los requisitos del usuario en materia de funcionalidad y organización del sistema por completo. Siendo necesario el desarrollo completo de un nuevo proyecto específico y especializado en tales requisitos.

3. PROPUESTA

El objetivo del proyecto es el desarrollo de un sistema de pruebas que permita unificar las pruebas que se realizan en el departamento así como servir de plataforma de lanzamiento de consultas a través de distintos protocolos.

Tendremos que analizar las posibles alternativas disponibles para poder realizar una buena implementación de la solución, y que ésta sea acorde con lo que espera el cliente de la aplicación.

3.1 Procedimiento

Para generar la propuesta al cliente se siguieron varios pasos:

- Analizar qué tipo de pruebas van a realizarse con la aplicación.

- Catalogar y organizar las pruebas ya disponibles, recopilando la información relativa a las mismas, así como generar nuevas pruebas en base a los conocimientos del personal del departamento.

- Escoger un lenguaje de programación apropiado, así como una herramienta de desarrollo apropiada que esté disponible en el departamento.

- Diseñar una interfaz que pueda satisfacer todas las necesidades del cliente, así como ser intuitiva, sencilla y eficaz, pero a la vez potente y configurable.

Una vez desarrolladas estas ideas y presentadas al departamento, se idearán ciertos parámetros internos de funcionamiento y organización de la aplicación, ya que se pretende que el cliente tome parte en todos los aspectos del desarrollo de la aplicación, ya que en el momento en que el proyecto esté terminado, la responsabilidad en el mantenimiento y las posibles ampliaciones de la aplicación van a delegarse al departamento de Banca Telefónica de CajaMadrid.

3.2 Sistema MOPROM

El Módulo de Pruebas de Operaciones Multicanales (MOPROM) es la propuesta de solución hardware y software para el proyecto. Ésta propuesta está basada en una plataforma de pruebas implementada en una aplicación de escritorio que se ejecutará en la máquina de usuario para facilitar las pruebas a los distintos servidores y servicios de CajaMadrid.

Dado que el usuario solicita una aplicación que sirva de plataforma para poder mantener y catalogar pruebas y a su vez proporcione un sistema de lanzamiento de pruebas, se va a optar por una aplicación de escritorio que aúne ambos requisitos. Ésta aplicación deberá estar disponible para todos los miembros del departamento, y podrá ser ejecutada en todas las máquinas del mismo. Todas las máquinas corren el mismo sistema operativo (Windows XP), lo cual simplifica el proceso de desarrollo ya que teóricamente todas las máquinas son similares en cuanto a hardware y software y tienen potencias similares.

Entre los lenguajes más populares de programación se va a optar por utilizar el lenguaje de programación Java, ya que los miembros del departamento conocen bien el lenguaje y cuentan con las herramientas de desarrollo adecuadas para la programación del sistema bajo Java. Además el lenguaje es muy flexible y nos permitirá desarrollar sin problemas una aplicación que cumpla todos los requisitos que nos exponga el usuario.

En lo referente al catálogo de pruebas que debe mantener el programa, se ha descartado la idea de crear una base de datos ya que el número de pruebas que el sistema debe manejar es bastante reducido y no se tienen expectativas de que aumente mucho más. Desarrollar este catálogo de pruebas sobre una base de datos aumentaría la complejidad del proyecto, incrementaría los costes y los tiempos de desarrollo. Se ha tomado la decisión de implementar el catálogo de pruebas mediante un sistema de archivos XML que servirán para almacenar las distintas pruebas así como sus características. Éstas pruebas estarán basadas en una plantilla XSD que definirá el estándar de esos ficheros XML para poder validarlos luego contra esa plantilla, de forma que aparte de asegurarnos que cumplen el estándar, sabremos con total certeza que su carga en el programa será perfecta. Asimismo creamos una forma organizada y de fácil edición en caso de emergencia, ya que podrían editarse con un programa de edición de textos simple. La idea de crear una base de datos se descartó por su coste y complejidad, ya que el número de pruebas es bastante reducido y no se tienen expectativas de que aumente mucho más.

3.3 Tecnologías y herramientas de desarrollo

Para el desarrollo del proyecto, tanto en sus fases de análisis, diseño e implementación a nivel de código, se van a utilizar diversas herramientas tanto software como hardware que serán expuestas a continuación, ofreciendo una visión global y resumida de los puntos que han favorecido la elección de esas herramientas concretas sobre otras.

3.3.1 Herramientas Software

Sistema operativo – Windows XP Professional Service Pack 3

Es el sistema operativo implantado en CajaMadrid en la mayoría de los equipos. A pesar de no poder elegir el sistema operativo con el que desarrollar el proyecto creo que es uno de los más indicados, ya que proporciona una plataforma bastante estable ya que el sistema operativo se encuentra en un estado de gran madurez. Además la compatibilidad del sistema operativo con las distintas herramientas de desarrollo y planificación es inmejorable.

Documentación – Microsoft Office 2003 Professional

Para toda la parte documental, tanto en la memoria como los manuales de usuario, se utilizará la suite Office 2003. A pesar de ser una suite algo antigua, es ideal para el desarrollo de documentos y proporciona un editor de textos, hoja de cálculo, y un programa para realizar presentaciones. Al ser la plataforma de textos más implantada, los archivos generados son compatibles y editables en prácticamente todas las plataformas.

Planificación – Microsoft Project 2003

Para el apartado de la planificación del proyecto se utilizará la herramienta Microsoft Project en su versión 2003. Es un software de administración de proyectos para asistir en el desarrollo de planes, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuestos y analizar cargas de trabajo. Se realizará la planificación y el diagrama Gantt en ésta plataforma.

Análisis y Diseño – Enterprise Architect 7.5

Para realizar los diferentes diagramas de clases así como otros tipos de diagramas UML se utilizará esta herramienta de diseño y

análisis de UML con la que se puede documentar, modelar, realizar ingeniería inversa y construir y mantener un sistema software orientado a objetos.

Desarrollo – Lenguaje de programación Java

En el apartado de codificación, el lenguaje de programación Java resulta idóneo ya que proporciona una plataforma muy potente, segura y sencilla. El software compilado en Java es multiplataforma gracias a la máquina virtual java, lo que le otorga compatibilidad en distintos sistemas operativos. Al tratarse de un lenguaje de programación muy extendido y con una gran documentación resulta muy sencillo trabajar con él a todos los niveles.

Datos – XML

El catálogo de pruebas que el programa será capaz de mantener se realizará mediante archivos XML. Habrá un archivo XML por cada prueba y éste describirá internamente la información necesaria para cada prueba. Todos los archivos se crearán de acuerdo a una plantilla definida en un archivo XSD de forma que los archivos XML tendrán que pasar una validación previa contra esa plantilla, tanto si van a ser leídos como si van a ser generados desde el programa.

Desarrollo – I.B.M. Rational Software Architect 7.0

Para la programación en Java se utilizará el software RSA. Es un entorno de desarrollo y modelado que combina el lenguaje de modelado UML con el diseño de arquitecturas en C++ y aplicaciones y servicios Web J2EE. Rational Software Architect está construido sobre el software opensource Eclipse y está orientado al análisis de código, además de contener herramientas para la creación de modelos UML.

Otros – Adobe PhotoShop CS4

Para la composición de la pantalla de inicio del programa, el logotipo del programa y otras ilustraciones, iconos y retoques menores fue necesario el uso de esta herramienta de diseño fotográfico. A pesar de ser muy compleja e incluso demasiado potente para las mínimas tareas de retoque que van a ser necesarias, es una herramienta con la que estoy muy familiarizado e intentar realizar estas tareas con otras herramientas menos potentes hubiera supuesto más tiempo en el desarrollo de estas tareas.

3.3.2 Herramientas Hardware

Hardware Desarrollo

Para el desarrollo de la aplicación y documentación se utilizará el siguiente equipo sobremesa:

- Ordenador: HP Compaq DC7100
- Procesador: Pentium 4 – 3 GHz
- Memoria RAM: 256 MB
- Disco duro: 80 GB
- Monitor: TFT Samsung SyncMaster 710n 17”.

Hardware Implantación

Para la implantación del sistema se utilizará un directorio de la red interna del que solo disponen acceso los empleados del departamento que tengan un nivel de permisos determinado. Para la ejecución de la aplicación cada equipo dispondrá un acceso directo en su escritorio para la ejecución del mismo. Este acceso directo ejecutará la máquina virtual Java del propio equipo de forma transparente para el usuario, como si de un programa compilado para Windows se tratase.

4. ANÁLISIS DEL SISTEMA

A continuación se muestran todos los pasos seguidos para alcanzar el análisis completo del sistema y su especificación de requisitos por parte del usuario.

Comenzaremos por realizar el catálogo de requisitos de usuario, el diagrama de casos de uso y el desarrollo de cada caso, y el catálogo de requisitos software.

4.1 Descripción del catálogo de requisitos

El catálogo de requisitos mostrado a continuación es fruto de las reuniones llevadas a cabo con miembros del equipo de trabajo para el que se desarrollará la aplicación.

El catálogo estará estructurado de la siguiente forma:

Requisitos Usuario:

- Capacidad: aquellos requisitos que describirán todas las funcionalidades disponibles en el sistema.

- Restricción: aquellos requisitos que describen aquellas funcionalidades que el usuario no podrá realizar en el sistema.

Requisitos Software:

- Requisitos funcionales: aquellos requisitos que indican qué hace la aplicación, como debe funcionar y como será su reacción a la interacción con los usuarios o entre los distintos módulos.

- Requisitos no funcionales: son aquellos requisitos que imponen restricciones al diseño del sistema y especifican cualidades propias de la ejecución del sistema, como seguridad, rendimiento, usabilidad y otras propias de la evolución del sistema, como facilidad de ampliación o escalabilidad.

Los requisitos se reflejarán sobre la siguiente tabla base:

Identificador	RX-YYNN		
Descripción	"Breve descripción del requisito"		
Necesidad	Esencial ☐	Deseable ☐	Opcional ☐
Prioridad	Alta ☐	Media ☐	Baja ☐
Estabilidad	Si ☐	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☐

Figura 1: Tabla base requisitos.

Campos:

Identificador: Identificador único del requisito. Se nombra con las siglas RX-YYNN, siendo X: "U: Usuario" o "S: Software", y YY:

- CA: Requisito de capacidad.
- RE: Requisito de restricción.

NN se refiere al número de requisito dentro de la categoría.

Campos de la tabla:

Descripción: Breve descripción del requisito.

Necesidad: Indica si un requisito es esencial o no.

Prioridad: Orden temporal del requisito.

Estabilidad: No es estable aquel requisito que está sujeto a posibles modificaciones.

Verificabilidad: Indicador de dificultad para comprobar si el requisito se cumple en el sistema final.

4.2 Requisitos de usuario

4.2.1 Requisitos de capacidad

Identificador	RU-CA01		
Descripción	El sistema se manejará principalmente a través de una pantalla principal que contendrá las funcionalidades del mismo.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☐	Media ☑	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RU-CA02		
Descripción	A través de la pantalla principal se accederá al editor de pruebas en formato XML.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☐	Media ☑	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RU-CA03		
Descripción	El usuario podrá crear pruebas en formato XML a través del editor incorporado en el sistema.		
Necesidad	Esencial ☐	Deseable ☑	Opcional ☐
Prioridad	Alta ☐	Media ☑	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RU-CA04		
Descripción	El usuario podrá lanzar pruebas a los diferentes sistemas a probar desde la pantalla principal de la aplicación.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RU-CA05		
Descripción	El usuario podrá editar pruebas a través del editor XML del sistema.		
Necesidad	Esencial ☐	Deseable ☑	Opcional ☐
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RU-CA06		
Descripción	El usuario podrá generar informes con los resultados de las pruebas.		
Necesidad	Esencial ☐	Deseable ☐	Opcional ☑
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RU-CA07		
Descripción	El usuario podrá cargar las tramas a enviar desde la pantalla principal.		
Necesidad	Esencial ☐	Deseable ☑	Opcional ☐
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RU-CA08		
Descripción	El usuario podrá editar las tramas antes de lanzar la prueba desde la ventana principal.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RU-CA09		
Descripción	El usuario podrá crear nuevas tramas o guardar las modificadas desde la ventana principal.		
Necesidad	Esencial ☐	Deseable ☐	Opcional ☑
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

4.2.2 Requisitos de restricción

Identificador	RU-RE01		
Descripción	El usuario está limitado a realizar una prueba cada vez, no puede realizar una prueba hasta la finalización de la anterior.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RU-RE02		
Descripción	No podrán eliminarse pruebas desde la aplicación, el usuario deberá lanzar el explorador para tal fin.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RU-RE03		
Descripción	No podrá haber dos pruebas con el mismo nombre. Las pruebas son únicas.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

4.3 Modelo de Casos de Uso

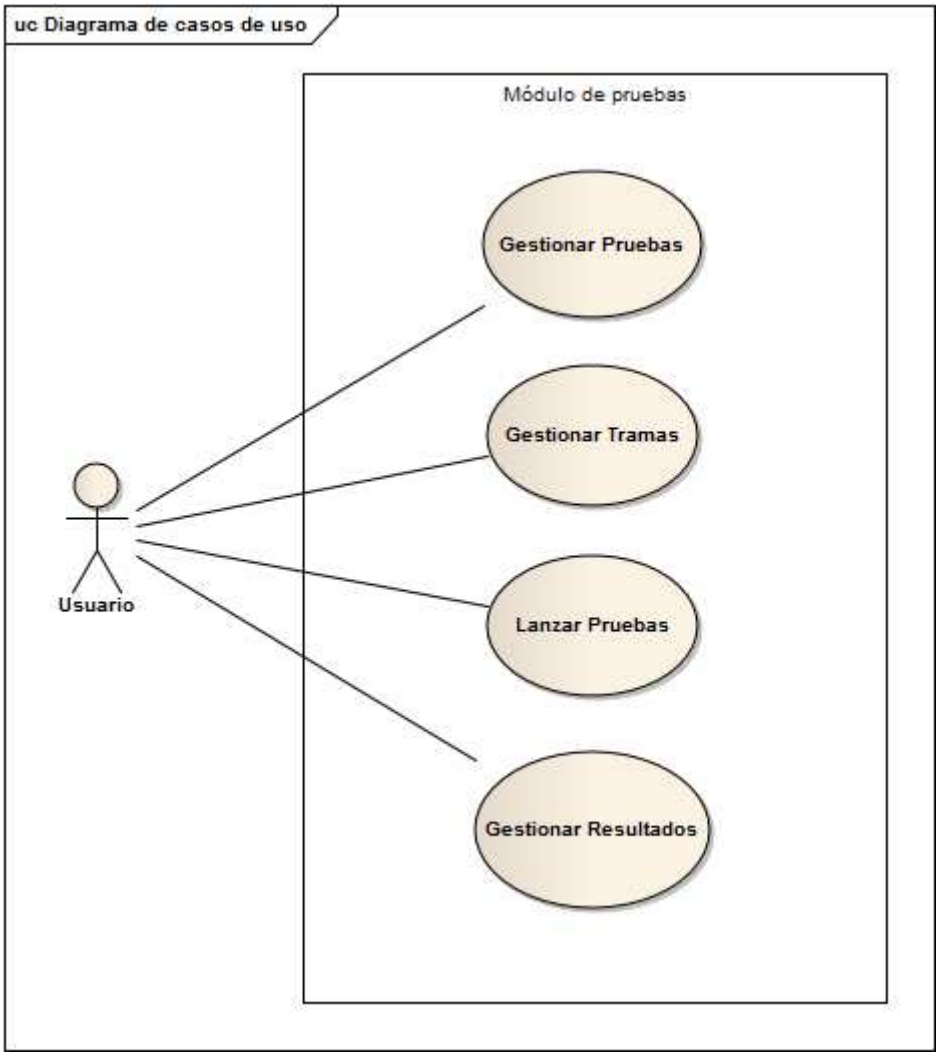


Figura 2: Diagrama de Casos de Uso

4.3.1 Casos de uso

Nombre	Gestionar Pruebas
Actores	Usuario
Objetivo	Administrar las pruebas del sistema llevando a cabo el visionado ordenado en la pantalla principal, así como el alta y modificación de las mismas. El sistema será capaz de trabajar con el formato de pruebas XML de forma que podrá representar sus datos en la interfaz del programa. Además contará con un editor de pruebas en formato XML que permitirá su edición completa, así como el alta de nuevas pruebas.
Precondiciones	Para que una prueba en XML se muestre en la interfaz principal debe haberse validado contra la plantilla estándar XSD para verificar que es correcta. Para la edición de una prueba concreta, ésta debe ser seleccionada en la interfaz principal para poder editarse.
Postcondiciones	Prueba cargada correctamente en la ventana principal. Prueba modificada o nueva prueba a partir del editor.
Escenario Básico	Aplicación iniciada. Pruebas cargadas correctamente.

Nombre	Gestionar Tramas
Actores	Usuario
Objetivo	Proporcionar un sistema de carga y modificación de tramas. Estas tramas constituyen los datos a enviar en las diferentes pruebas. Se podrán cargar tramas en formato texto y serán mostradas en la ventana principal, pudiendo ser editadas antes de lanzar la prueba. Una vez editadas podrán ser sobrescritas o bien se podrá generar una trama nueva.
Precondiciones	Para la edición de la trama ésta debe haberse cargado previamente en el cuadro de texto de la ventana principal mediante la herramienta de selección y carga de archivos.
Postcondiciones	Trama cargada en la ventana principal. Trama modificada o almacenada en formato texto.
Escenario Básico	Aplicación Iniciada. Prueba cargada.

Nombre	Lanzar Pruebas
Actores	Usuario
Objetivo	Realizar consultas a las pruebas especificadas. Con la información sobre la prueba y la trama a enviar, la aplicación lanzará una consulta al servicio o sistema remoto para comprobar su funcionamiento. Funcionará en base a cinco protocolos (POST, GET, URL, SOAP y conexión por Socket).
Precondiciones	El usuario debe seleccionar la aplicación a probar y el entorno de pruebas. Además opcionalmente y si la aplicación lo requiere cargar una trama a enviar. Esta trama podrá ser editable por el usuario desde la interfaz principal.
Postcondiciones	Prueba lanzada a la espera de respuesta por parte del servidor o aplicación remotos.
Escenario Básico	Aplicación y entorno seleccionados. Trama cargada.

Nombre	Gestionar Resultados
Actores	Usuario
Objetivo	El resultado de las pruebas será visualizado en la ventana principal, además se informará del tiempo transcurrido en el cronómetro incorporado. Este resultado podrá ser almacenado en un fichero de texto, o bien se podrá generar un informe completo con datos relativos de la prueba además del resultado de su ejecución, de forma que quedará almacenado en el portapapeles del sistema.
Precondiciones	El usuario debe hacer lanzado una prueba habiendo seleccionado una aplicación y un entorno, así como adjuntar una trama o parámetros si procede. La respuesta debe haber llegado por parte del servidor o aplicación remotos, es decir, la prueba debe de hacer finalizado teniendo el programa una respuesta obtenida y mostrada por pantalla.
Postcondiciones	Resultado de la prueba mostrado en el panel de visualización de la prueba, rellenado con los datos obtenidos del servidor o aplicación remotos. Cronómetro detenido mostrando la duración de la prueba.
Escenario Básico	Prueba finalizada.

4.4 Requisitos software

Los requisitos se reflejarán sobre la siguiente tabla base:

Identificador	RX-YYNN		
Descripción	"Breve descripción del requisito"		
Necesidad	Esencial ☐	Deseable ☐	Opcional ☐
Prioridad	Alta ☐	Media ☐	Baja ☐
Estabilidad	Si ☐	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☐

Figura 3: Tabla base requisitos software.

Campos:

Identificador: Identificador único del requisito. Se nombra con las siglas RX-YYNN, siendo X: "U: Usuario" o "S: Software", y YY:

- FU: Requisito funcional.
- IN: Requisito de interfaz.
- PL: Requisito de plataforma.
- SE: Requisito de seguridad.

NN se refiere al número de requisito dentro de la categoría.

Campos de la tabla:

Descripción: Breve descripción del requisito.

Necesidad: Indica si un requisito es esencial o no.

Prioridad: Orden temporal del requisito.

Estabilidad: No es estable aquel requisito que está sujeto a posibles modificaciones.

Verificabilidad: Indicador de dificultad para comprobar si el requisito se cumple en el sistema final.

4.4.1 Requisitos Funcionales

Identificador	RS-FU01		
Descripción	El sistema dispondrá de un editor de pruebas XML		
Necesidad	Esencial ☐	Deseable ☑	Opcional ☐
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RS-FU02		
Descripción	El programa permitirá cargar un archivo para el lanzamiento de la prueba, bien una trama XML, o bien parámetros de la prueba (dependiendo del tipo de prueba).		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RS-FU03		
Descripción	Las tramas o parámetros de la prueba podrán ser editables en un cuadro en la pantalla principal antes de lanzar la consulta.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☐	Media ☑	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RS-FU04		
Descripción	Las pruebas estarán organizadas según su implementación en los servidores, divididas en las categorías “Desarrollo”, “Tránsito”, “Real” y “Otros”.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☐	Media ☑	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RS-FU05		
Descripción	El programa contendrá un panel con un desplegable en el que seleccionar la prueba a lanzar.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RS-FU06		
Descripción	La pantalla principal implementará un cronómetro con la duración de la consulta desde el lanzamiento de la prueba hasta la llegada de la respuesta desde el servidor.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☐	Media ☑	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RS-FU07		
Descripción	El programa contemplará la posibilidad de marcar el tiempo máximo (timeout) de las pruebas.		
Necesidad	Esencial ☐	Deseable ☑	Opcional ☐
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☑	Baja ☐

Identificador	RS-FU08		
Descripción	Las pruebas de tipo GET podrán lanzarse externamente a través del explorador de Internet.		
Necesidad	Esencial ☐	Deseable ☐	Opcional ☑
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☑	Baja ☐

Identificador	RS-FU09		
Descripción	El resultado de las pruebas se podrá almacenar en un fichero de texto.		
Necesidad	Esencial ☐	Deseable ☐	Opcional ☑
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☑	Baja ☐

Identificador	RS-FU10		
Descripción	El programa permitirá generar un informe con el resultado de las pruebas que quedará copiado en el portapapeles del sistema.		
Necesidad	Esencial ☐	Deseable ☐	Opcional ☑
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

4.4.2 Requisitos No Funcionales

4.4.2.1 Plataforma

Identificador	RS-PL01		
Descripción	El programa deberá ejecutarse en las máquinas del departamento dotadas con Windows XP y la versión de la máquina Virtual Java 1.5 o superior.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

4.4.2.2 Interfaz

Identificador	RS-IN01		
Descripción	La interfaz de la aplicación tendrá un estilo similar a las ya utilizadas en la empresa y contará con los colores y logotipos corporativos.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RS-IN02		
Descripción	La interfaz de la aplicación será una ventana generada en Java, sin requerimientos especiales en cuanto a accesibilidad u otros.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RS-IN03		
Descripción	La pantalla principal contendrá una caja de texto en la que se mostrará el resultado de la prueba.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☑	Media ☐	Baja ☐
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RS-IN04		
Descripción	La ventana principal contendrá un botón denominado “Limpiar” que permitirá borrar el resultado de la prueba anterior así como su tiempo de cronómetro para poder realizar varias pruebas consecutivas.		
Necesidad	Esencial ☐	Deseable ☑	Opcional ☐
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

Identificador	RS-IN05		
Descripción	El panel de lanzamiento de prueba contendrá una caja seleccionable para lanzar la prueba externamente mediante el programa “Internet Explorer” (solo disponible en las peticiones de tipo GET).		
Necesidad	Esencial ☐	Deseable ☑	Opcional ☐
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

4.4.2.3 Escalabilidad

No se han descrito requisitos especiales en cuanto a escalabilidad

4.4.2.4 Rendimiento

No se han descrito requisitos en cuanto a rendimiento. El carácter de la aplicación implica una utilización media-baja de la misma, y durante tiempos de uso muy cortos.

4.4.2.5 Disponibilidad

No se han descrito requisitos específicos en cuanto a disponibilidad, al no tratarse de una aplicación de funcionalidad crítica para la empresa.

4.4.2.6 Seguridad

Identificador	RS-SE01		
Descripción	El programa se encuentra ubicado en un directorio de trabajo protegido en red, y solamente accesible por el grupo de trabajo encargado de las pruebas.		
Necesidad	Esencial ☑	Deseable ☐	Opcional ☐
Prioridad	Alta ☐	Media ☐	Baja ☑
Estabilidad	Si ☑	No ☐	
Verificabilidad	Alta ☐	Media ☐	Baja ☑

4.4.2.7 Extensibilidad

No se describen requisitos especiales en cuanto a extensibilidad.

4.5 Arquitectura física del sistema

En la estructura del sistema vamos a tener tres sistemas claramente diferenciados, aunque en términos estrictos de diseño e implementación solo dos de ellos tendrán importancia. Por un lado tenemos un servidor de aplicación, que es el lugar donde estarán los archivos del programa MOPROM y su ejecutable. El ordenador del usuario será el que acceda a esa unidad de red de disco duro y pueda ejecutar el programa, que estará ejecutándose en la propia máquina del usuario, y la aplicación lanzará sus consultas a los servidores de CajaMadrid que serán los sistemas a probar.

La aplicación final quedará instalada en un disco de red al que tendrá acceso solamente el departamento de Canales Complementarios – Banca Telefónica. De éste modo cada usuario podrá ejecutar la aplicación de forma remota desde su ordenador. Toda modificación en los datos quedará grabada automáticamente en el servidor de la aplicación. Al ejecutarse la aplicación en la propia máquina del usuario, implica que la comunicación entre la misma y los servicios a lanzar pruebas sea directamente desde la máquina del usuario a la máquina donde esté alojado el servicio a comprobar.

La ejecución del programa en la máquina del usuario se realizará siempre en máquinas dotadas con el sistema operativo Windows XP Professional y conexión a la intranet de CajaMadrid. No se requieren requisitos mínimos en cuanto a velocidad de procesador, memoria RAM, etc...

Para la programación de la aplicación se ha utilizado el lenguaje Java, por tanto el programa está compilado en un archivo con extensión ".jar" que se lanzará por medio de un ejecutable batch (".bat"), para asegurar su compatibilidad con la máquina virtual Java instalada en los equipos del departamento. En éste aspecto el único requisito que debe tener la máquina que vaya a ejecutar el programa es que disponga de la máquina virtual Java en su versión 1.5 o superior.

Las aplicaciones y servicios que MOPROM es capaz de probar están instaladas en diversas máquinas, atendiendo a varios protocolos dentro de la red de trabajo interno de CajaMadrid. Por motivos de seguridad y privacidad no se puede dar más información respecto a estos sistemas.

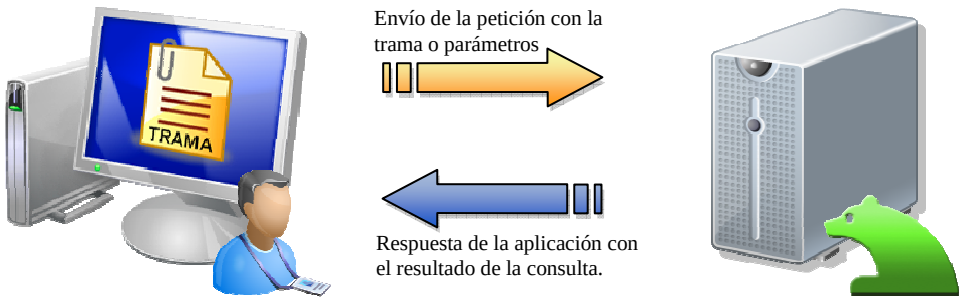


Figura 4: Arquitectura física de MOPROM.

En la figura anterior podemos ver como el ordenador del usuario y el servidor de la aplicación se comunican mediante la red LAN interna de CajaMadrid.

Una vez arrancada la aplicación desde el directorio de red, las consultas de las pruebas se realizan del siguiente modo:

- Se carga la información con los parámetros de la prueba.
- Se selecciona una trama que enviar, o los parámetros (si se aplica).
- Se lanza la prueba contra el servidor de CajaMadrid.
- Se espera al resultado y se muestra por pantalla.

Como se puede apreciar la comunicación se produce bidireccionalmente entre la máquina del usuario y el servidor de la aplicación.

5. DISEÑO DEL SISTEMA

Llegados a esta parte del proyecto tenemos que transformar toda la información y las ideas recogidas en el análisis y transformarlo en un diseño que cumpla con los requisitos dados por el cliente. Se tratarán diversos aspectos relativos a la arquitectura escogida, los diferentes modelos, así como una visión de las clases y procedimientos que van a componer el sistema software.

5.1 Diseño y arquitectura del sistema

5.1.1 Arquitectura distribuida

En el ámbito de las aplicaciones de software, la arquitectura distribuida o multicapa está basada en la idea de que el software puede estar dividido en varios módulos, cada uno de ellos desempeñando una función determinada. Esta distribución de las tareas en diferentes módulos da lugar a una aplicación que:

- Es más sencilla de mantener y mejorar, puesto que hay una clara división de tareas entre los distintos módulos que conforman la aplicación.
- Puede ser ampliada más fácilmente si los requerimientos de rendimiento aumentan.
- Proporciona interoperabilidad, pudiendo desarrollarse nuevas extensiones en la aplicación en alguno de los niveles sin tener que modificar el resto.

Las desventajas principales de una fuerte división en capas son:

- El rendimiento es menor, al tener que atravesarse más capas de código para realizar una acción determinada.
- El tiempo de puesta en funcionamiento de una primera versión funcional de la aplicación puede ser más lento si la aplicación es relativamente sencilla.

Las desventajas con pocas y pierden sentido en cuanto la aplicación es mínimamente compleja. Por ello la arquitectura multicapa es la elección natural para el desarrollo de aplicaciones empresariales con un mínimo de complejidad.

Existen muchas posibilidades en cuanto a la división multicapa, pero vamos a centrarnos en la división en tres capas que es la más común.

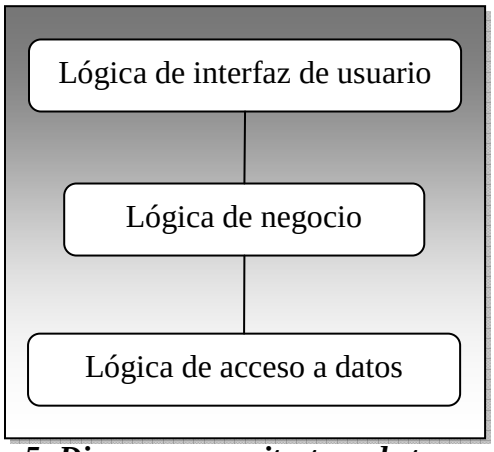


Figura 5: Diagrama arquitectura de tres capas.

En la división en tres capas, división que viene dictada por la propia funcionalidad de la aplicación:

- La parte central de un software consiste normalmente en la interacción con el usuario. Lo que el usuario ve y utiliza (como un listado de elementos o un formulario que el usuario rellena con datos) constituye la lógica de interfaz de usuario. Su función es mostrar al usuario la información de la aplicación y los controles necesarios para que éste interactúe con ella.

- Por debajo de esta capa, encontramos la lógica de negocio. Se trata de la parte del software encargada de realizar todas las operaciones lógicas de la aplicación y el procesado de las decisiones tomadas el usuario. Es la capa que lleva a cabo el trabajo principal de la aplicación y ejecuta las tareas para las que el software fue diseñado. También es la encargada de recibir e interpretar los datos enviados por los usuarios desde la capa de presentación.

- Por último, la lógica de acceso a datos es la encargada de la lectura y escritura de la información utilizada por la aplicación. Se encarga de proporcionar y guardar información de una forma transparente a la capa de negocio, independientemente del soporte sobre el que ésta es almacenada finalmente.

Utilizando esta división, podríamos tener (pero no es obligatorio en absoluto) cada capa de la aplicación ejecutándose en una máquina distinta. Algunas de las ventajas de este enfoque son:

- Aumenta la seguridad de la aplicación en su conjunto. Si un usuario no autorizado consiguiera acceso a uno de los módulos de la aplicación no compromete el sistema entero.

- La escalabilidad es muy grande, puesto que se pueden solucionar problemas de rendimiento en alguna de las capas de forma precisa.

Por último, es importante señalar que, sobre todo en proyectos de software grandes, la aplicación puede ser dividida en un número de capas superior a tres hasta un número arbitrario de ellas, siempre teniendo en cuenta que un exceso de división también puede acarrear resultados negativos.

Para representar estas tres capas de la arquitectura vamos a describir sus modelos uno por uno. En el caso de la capa de lógica de acceso a datos, la representaremos con el modelo de datos, la capa de lógica de negocio con el modelo lógico, y la capa de lógica de interfaz de usuario con el modelo visual.

5.1.2 Modelo de datos

Una vez tomada la decisión de que el origen de datos van a ser archivos en formato XML tendremos que definir una plantilla para poder representarlos. Estas plantillas vienen definidas por ficheros XML.

A continuación se detallará el esquema de la plantilla XSD generada para definir el formato de los ficheros XML que manejará el programa tanto para leer como para escribir datos. Mediante este XSD podemos validar y procesar mediante un parser los ficheros XML que definen las pruebas.

Cada prueba se considera como un test, que contendrá el nombre de la aplicación, el acceso que tendrá (POST, GET, SOCKET, URL, SOAP), y una lista de entornos. Cada entorno contiene su nombre, su URL, la ruta de su trama y comentarios a tener en consideración.

A continuación se muestran unos esquemas gráficos para visualizar las herencias y jerarquías, así como el código fuente del fichero XSD.

El primer esquema es la vista global de la estructura XSD. Vemos que contiene como elemento principal “test”. Dentro de test tenemos los tres tipos principales: TipoEntornos, TipoAcceso, y TipoEntorno.

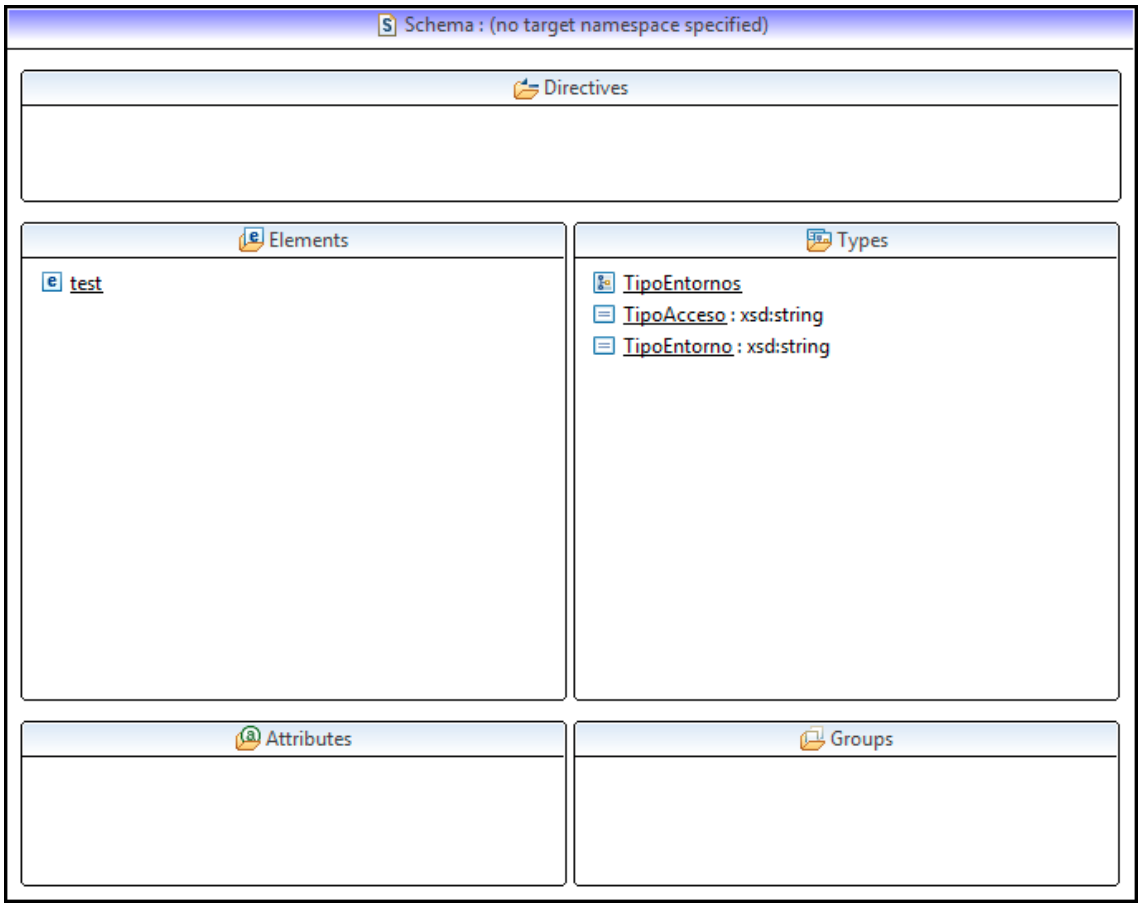


Figura 6: Esquema XSD (Vista global).

La siguiente figura es el esquema desglosado en el que se pueden ver los atributos dentro de los tipos de datos. El tipo aplicación es de tipo entorno, que es un simple "string" con cuatro posibles valores (desarrollo, tránsito, real, otro). Asimismo entornos es un enumerado de más de un entorno, formado por varios campos. Tipo acceso es un string que puede tomar uno de estos cinco valores: POST, GET, SOCKET, URL Y SOAP.

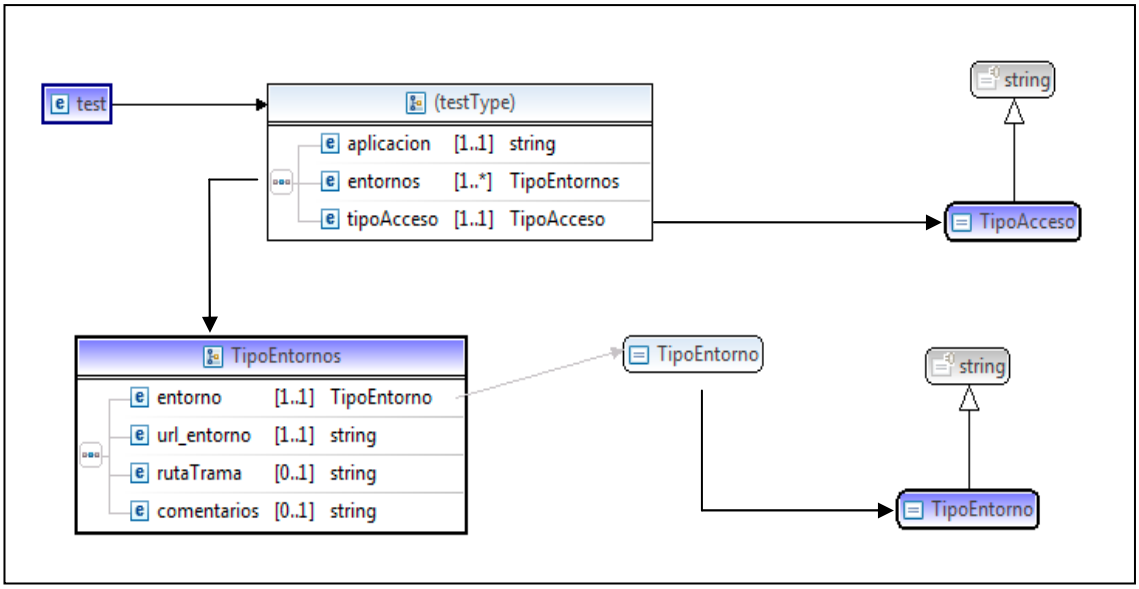


Figura 7: Esquema XSD (Desglosado).

En el siguiente recuadro se encuentra el código en XML que define la plantilla XSD. Se pueden apreciar los distintos tipos de datos y el encapsulado de las distintas estructuras en detalle.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:element name="test">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="aplicacion" type="xsd:string" maxOccurs="1" minOccurs="1" />
        <xsd:element name="entornos" type="TipoEntornos" maxOccurs="unbounded" minOccurs="1" />
        <xsd:element name="tipoAcceso" type="TipoAcceso" maxOccurs="1" minOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>
</xsd:element>


<xsd:complexType name="TipoEntornos">
    <xsd:sequence>
        <xsd:element name="entorno" type="TipoEntorno" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="url_entorno" type="xsd:string" maxOccurs="1" minOccurs="1"/>
        <xsd:element name="rutaTrama" type="xsd:string"  minOccurs="0" maxOccurs="1"/>
        <xsd:element name="comentarios" type="xsd:string"  minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="TipoEntorno">
    <xsd:restriction base="xsd:string" >
        <xsd:enumeration value="DESARROLLO"/>
        <xsd:enumeration value="TRANSITO"/>
        <xsd:enumeration value="REAL"/>
        <xsd:enumeration value="OTRO"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="TipoAcceso">
    <xsd:restriction base="xsd:string" >
        <xsd:enumeration value="POST"/>
        <xsd:enumeration value="GET"/>
        <xsd:enumeration value="SOCKET"/>
        <xsd:enumeration value="URL"/>
        <xsd:enumeration value="SOAP"/>
    </xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

Figura 8: Código fuente plantilla XSD.

5.1.3 Modelo lógico

Una vez analizado el modelo de datos, tenemos que abstraerlos para generar un modelo que pueda representar los datos en el sistema de una forma eficaz y que puedan ser manipulados y transformados por los procedimientos que se desarrollarán en Java.

En el siguiente modelo se puede apreciar el diagrama de clases del programa MOPROM:

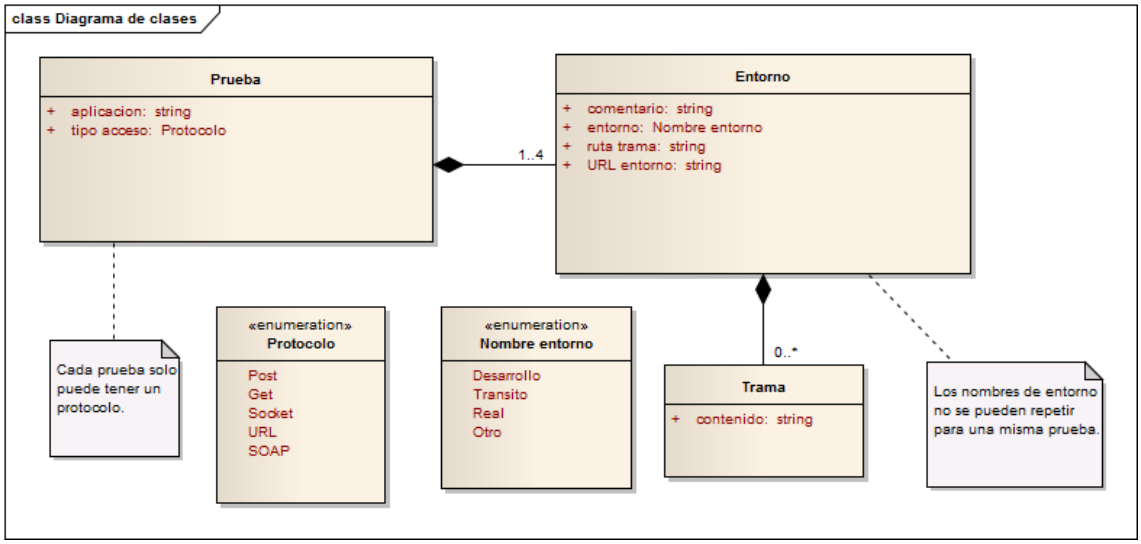


Figura 9: Diagrama de clases.

Como podemos apreciar cada prueba tiene almacenados los datos relativos asociados a una aplicación. Cada prueba tiene asociados varios entornos con la información concreta de cada uno de ellos y a su vez cada trama solo puede pertenecer a un entorno y una aplicación concreta.

5.1.4 Modelo Visual

Tras un análisis de los requisitos dados por el cliente y las especificaciones generales de qué es exactamente lo que esperan de la herramienta, se ha podido esbozar un prototipo de la aplicación, que definirá el funcionamiento general de la misma, así como una primera aproximación a su interfaz de usuario.

La lógica de interfaz de usuario la vamos a representar mediante un prototipo de la interfaz del programa para mostrar a grandes rasgos la apariencia que tendrá la aplicación y como podremos representar en pantalla los datos que maneja internamente el programa.

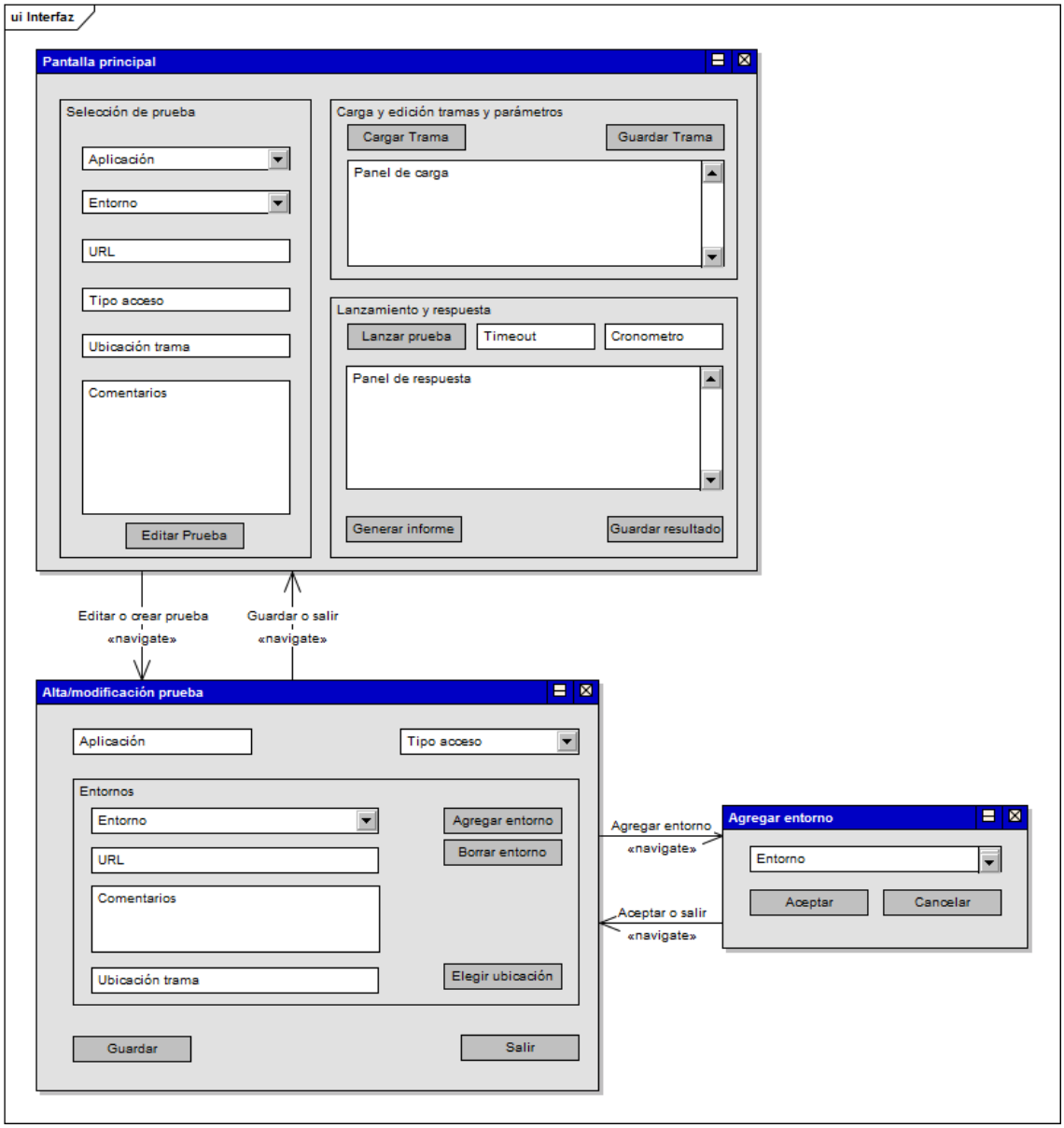


Figura 10: Prototipo interfaz.

Se pueden apreciar en el prototipo las dos ventanas principales. Una es la ventana principal donde se pueden representar todos los datos de las pruebas

almacenadas, y que además sirve de plataforma para realizar pruebas y consultas. La otra ventana es el editor de pruebas en la que se podrán modificar las pruebas almacenadas, o modificar las existentes.

En el diseño no se han representado otras pantallas secundarias de notificación de errores.

5.2 Diseño y descripción de componentes

En este apartado se va a explicar los componentes del sistema y sus funciones principales de MOPROM. En la siguiente ilustración se explica de una manera gráfica la jerarquía de componentes.

El paquete MOPROM estaría contenido dentro de los paquetes es.cm.cc. Los tres primeros componentes son genéricos para denominar la aplicación, de acuerdo al estándar de CajaMadrid para su posterior subida al repositorio de código.

De este modo: es (españa) – cm (cajamadrid) – cc (canales complementarios).

Dentro del departamento de canales complementarios encontramos el repositorio de MOPROM donde las clases que forman el programa Java se encuentran divididas en cuatro paquetes principales, los beans, las clases principales que arrancan el programa, utilidades y las clases de ventana para dibujar la interfaz gráfica.

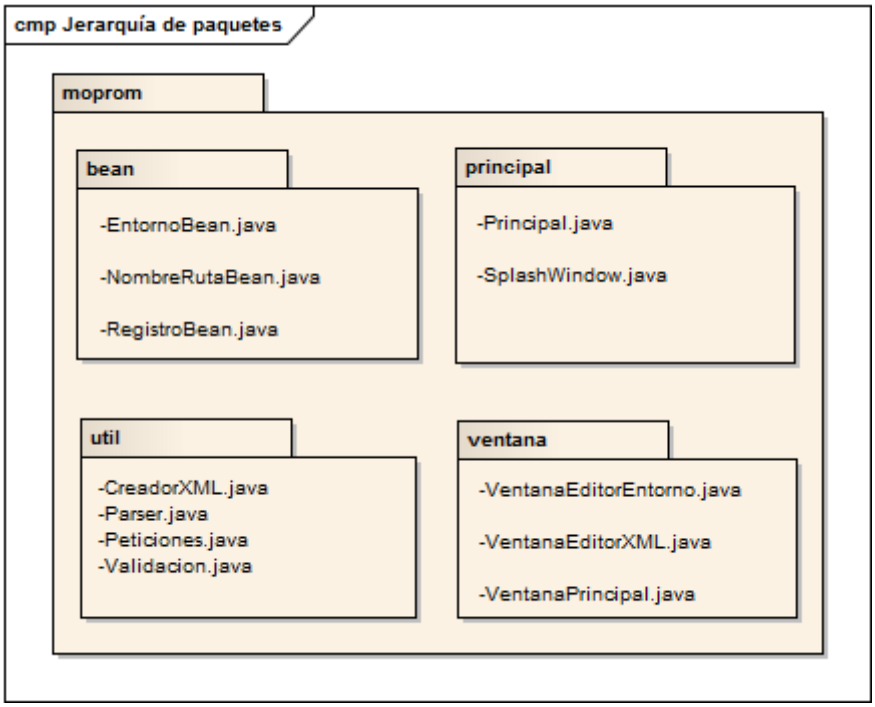


Figura 11: Jerarquía de paquetes.

5.2.1 Componente Principal

El componente Principal, como su propio nombre indica, es la clase principal (o main) de la aplicación. Principal contiene las siguientes clases:

- Principal.java
- SplashWindow.java

Estas clases se encargan de poner en funcionamiento el programa MOPROM.

5.2.1.1 Principal

La clase Principal actúa como un iniciador del sistema, se encarga de crear las variables que va a usar el programa, generar el fichero LOG, así como detectar y abrir el fichero "config.ini" para extraer la información relativa a los iconos del programa, sus directorios, parámetros de ejecución, así como sus archivos clave a la hora de la ejecución del programa. Dado que el fichero config.ini y su información son críticos, lo primero que se hace es una comprobación del mismo y en caso de no encontrarse o no ser correcto su contenido, la aplicación no se iniciará.

Una vez creadas las instancias correspondientes de las ventanas a mostrar, se hace una llamada a la clase SplashWindow, de este modo a partir de este momento será la clase SplashWindow la que se encargue de lanzar la ventana principal del programa.

5.2.2.1 SplashWindow

La clase SplashWindow muestra un logotipo durante un tiempo definido (o hasta que el usuario hace click) y después se cierra y carga la ventana principal del programa (hay que recordar que la instancia de ventana principal ya estaba creada anteriormente). Una vez el timeout se cumple o el usuario hace click en el logotipo, éste se oculta y se muestra la ventana principal.

A continuación se muestra el logotipo de MOPROM sobre un fondo que constituye la imagen de presentación del programa. La imagen fue realizada mediante un montaje en Adobe Photoshop utilizando de fondo una fotografía de la torre Kio de CajaMadrid en plaza castilla, en la que se ve reflejado el depósito de agua del canal de Isabel II.

Figura 12: Imagen de carga de MOPROM.

5.2.2 Componente Ventana

El componente Ventana está formado por las siguientes clases:

- VentanaPrincipal.java
- VentanaEditorXML.java
- VentanaEditorEntorno.java

Las clases del componente Ventana se encargan de mostrar en pantalla las opciones disponibles así como de lanzar los procesos y funciones necesarias para trabajar con los datos en local y con los datos obtenidos al lanzar las pruebas.

5.2.2.1 VentanaPrincipal

En cada ejecución de MOPROM se crea una única instancia de VentanaPrincipal desde la clase Principal. La ventana Principal es el centro de operaciones de MOPROM. En esta ventana se muestran las funciones principales del programa y constituye el motor de la aplicación. Esta clase es capaz de hacer llamadas a procesos de carga de ficheros de forma que pueda rellenar los campos de texto con la información de las pruebas a lanzar. Proporciona una herramienta de carga y edición de tramas así como un panel de lanzamiento de la prueba. Desde la ventana principal se recogen los resultados de las pruebas lanzadas ofreciendo además, opciones de salida a esos datos en formato de texto.

Desde la ventana principal es posible cargar el editor de ficheros XML que van a dar forma a las pruebas que podremos realizar dentro de la aplicación. La instancia de la ventana del editor ya se encuentra creada en la ventana principal, para pasar al editor se bloquea y oculta la ventana principal y se muestra un diálogo de selección (editar prueba o crear una nueva).

A continuación se muestra la ventana principal ejecutada en la máquina virtual Java de Windows XP:

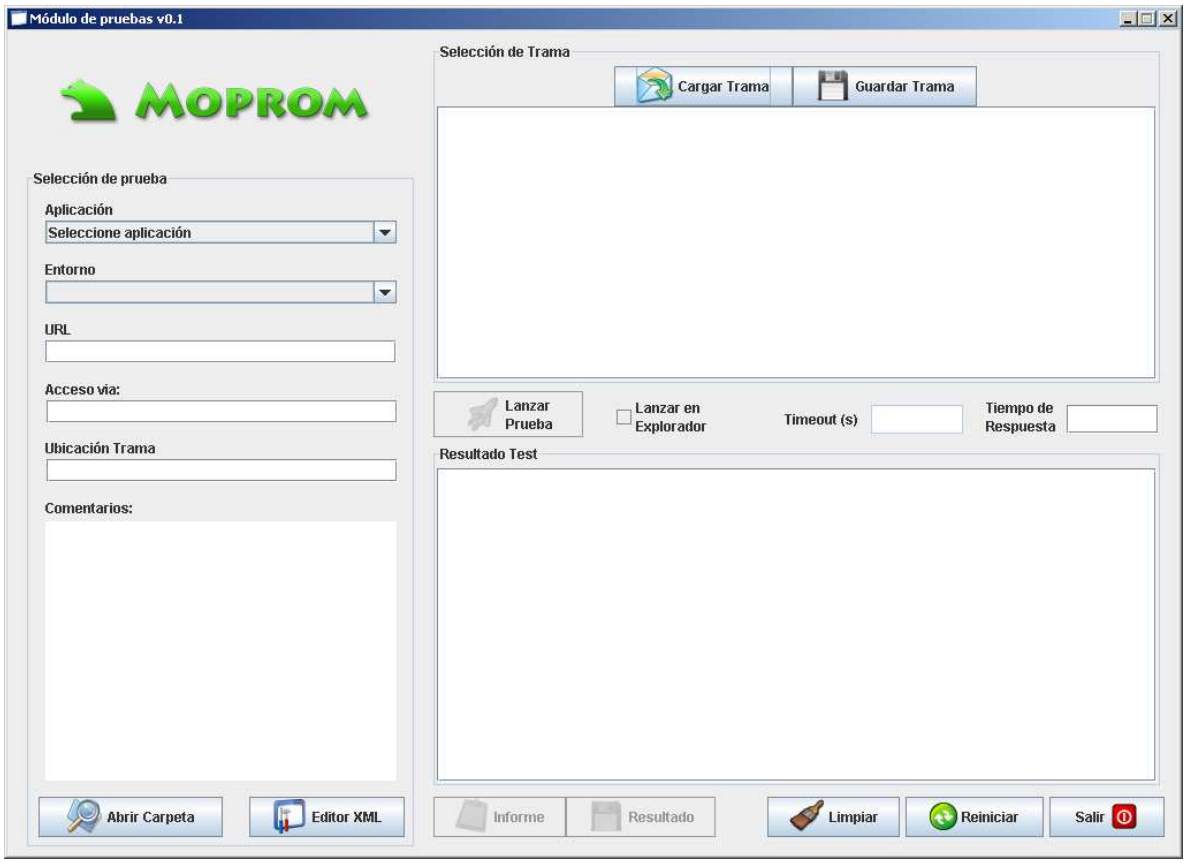


Figura 13: Ventana principal.

5.2.2.2 VentanaEditorXML

VentanaEditorXML es la clase que se encarga de generar la ventana de edición de los ficheros XML. Esta ventana proporciona una herramienta sencilla para poder generar nuevas pruebas o bien para editar las ya existentes. Todos los campos a rellenar en la ventana son comprobados para que solo puedan almacenarse pruebas válidas y aceptadas con la plantilla XSD que marca el estándar de las pruebas.

Si se trata de una edición, en la función para visualizar la ventana XML la clase VentanaPrincipal envía el registro con la información de la prueba, en caso contrario el registro se envía vacío y se indica que es una creación mediante una variable booleana.

Para agregar un entorno a la ventana se hace una llamada a la instancia de la clase VentanaEditorEntorno, que permitirá mediante una lista desplegable crear un nuevo entorno para la prueba que en ese momento se está editando.

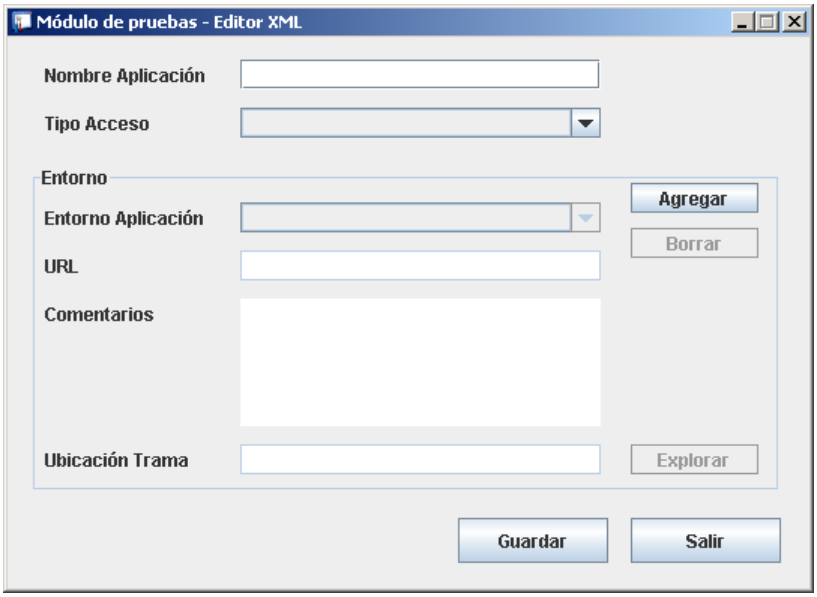


Figura 14: Ventana Editor XML.

5.2.2.3 VentanaEditorEntorno

Esta clase es una ventana muy simple con la opción de seleccionar el tipo de entorno. Solo se mostrarán los entornos que no estén presentes en la prueba, ya que no se puede repetir un mismo entorno.

Tras seleccionar el entorno adecuado se oculta la ventana y se vuelve a mostrar el editor XML.

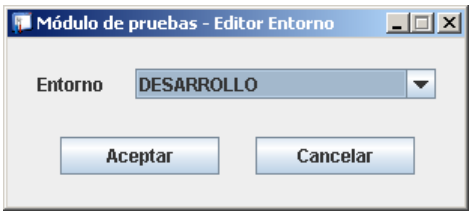


Figura 15: Ventana Selector Entorno.

5.2.3 Componente Bean

Para definir los datos que manejan las ventanas y los registros, se ha optado por albergarlos dentro de “beans” de java, teniendo así encapsulados los datos y ofreciendo además operaciones para su manipulación. Además de este modo todas las variables son privadas, requiriendo del uso de las funciones “set” y “get” para su consulta y modificación.

El componente Bean está formado por las siguientes clases:

- RegistroBean.java
- EntornoBean.java
- NombreRutaBean.java

Cada clase contiene información y funciones acordes a su contenido que serán descritas a continuación:

5.2.3.1 RegistroBean

Ésta clase es una implementación de un registro clásico en forma de bean en Java. Contiene tres variables de tipo “string” (rutaFichero, aplicación, tipoAcceso) y una lista de EntornoBean (entornos). Contiene funciones públicas de modificación y consulta de las variables en forma de “set” y “get”, más el nombre del atributo.

5.2.3.2 EntornoBean

La clase EntornoBean implementa un Bean con los datos relativos al entorno. El Bean contiene cuatro variables de tipo “string” (entorno, url_entorno, rutaTrama, comentario), así como sus funciones públicas de modificación y consulta de las variables en formad de “set” y “get”. Este Bean es utilizado como nodo en la lista de entornos que contiene registroBean.

5.2.3.3 NombreRutaBean

Éste Bean contiene datos relativos al nombre de cada aplicación a probar y su ruta (de fichero). Estos Bean van a ser utilizados como nodos de una lista de aplicaciones, de este modo vamos a poder tener localizados los archivos XML de cada aplicación en todo momento. El Bean se compone de dos “strings” (nombreApp y ruta) así como de las operaciones necesarias para modificar y consultar la información de esos campos (“set” y “get”).

5.2.4 Componente util

El componente util contiene varias herramientas para los procesos más complejos de la aplicación.

El componente util está formado por las siguientes clases:

- Validacion.java
- Parser.java
- CreadorXML.java
- Peticiones.java

5.2.4.1 Validacion

Esta clase constituye una herramienta muy potente para el correcto funcionamiento de MOPROM. En ella se implementan dos funciones muy útiles para el programa.

Por un lado está la funcionalidad de validación de ficheros, que se encarga de comprobar si un fichero XML sigue las pautas de la plantilla XSD del programa. Esto significa que cada fichero XML se valida contra la plantilla XSD para verificar si su formato cumple el estándar que define la plantilla. En caso de no superar la prueba de validación, la información de este fichero no será válida para el programa. Estos errores en la validación de ficheros serán almacenados en el fichero LOG del programa con una descripción del error, además de aparecer en la pantalla una ventana emergente informando del problema. Para la comparación del XML contra el XSD se utiliza el API SAX (Simple API for XML).

Por otra parte y gracias a la herramienta que permite validar un XML contra la plantilla XSD, se puede generar una lista de pruebas válidas para su ejecución en MOPROM, esto es que dado un directorio por defecto en el que estarán contenidas las pruebas, la función generarListadoFicheros abre y chequea uno a uno esos ficheros para verificar su compatibilidad con la plantilla XSD, de ser correcta la comprobación, esta aplicación pasa a formar parte de la lista de aplicaciones válidas para MOPROM, almacenándose el nombre de la aplicación y la dirección física del fichero XML en disco. Una vez ha terminado de analizar todos los archivos de ese directorio devuelve una lista con los ficheros válidos y en caso de haber fallado alguno, se almacenará el error en el fichero LOG del programa y se mostrará una ventana emergente informando del problema.

5.2.4.2 Parser

Esta clase implementa un analizador sintáctico (en inglés “parser”). Utiliza SAX (Simple API for XML) para procesar un fichero XML dado y convertirlo en un formato de registroBean más cómodo y comprensible para la aplicación. Este analizador sintáctico recibe un fichero XML, y una vez marcados en el código las etiquetas de las distintas jerarquías del XML el analizador procede a descomponer y extraer la información almacenándola en el registroBean correspondiente.

La clase incluye detección de errores y en caso de darse algún fallo mostrará un mensaje en un cuadro emergente informando del error y además grabará el error en el fichero LOG.

5.2.4.3 CreadorXML

Esta clase contiene métodos y funciones para que, dado un registroBean, se pueda generar un fichero XML formateando esa información.

Para poder generar el fichero XML antes tienen que darse varios pasos. Teniendo como partida el registro con los datos de la prueba, se va a generar un árbol de nodos enlazados con los contenidos de los campos. Para ello se utiliza la herramienta JDOM que se trata de una biblioteca de código abierto para manipular datos XML y que está optimizada para Java. Los campos del registro se recorren uno a uno de modo que mediante esta herramienta podemos ir creando nodos en un árbol según vamos recorriéndolos. Una vez generado el árbol completo, se hace una llamada para grabar el archivo XML en un fichero de texto. Este fichero XML será compatible con la aplicación ya que respeta el formato que fija el XSD y podrá ser comprobado y validado correctamente con la herramienta de validación.

5.2.4.4 Peticiones

La siguiente clase Java implementa la funcionalidad más importante del programa, que es el lanzamiento de pruebas a distintos protocolos. Esta clase contiene métodos que controla la ejecución de las pruebas y contiene los procedimientos específicos para las llamadas (GET, POST, SOCKET, URL, SOAP).

La clase contiene un método genérico para lanzar pruebas desde la ventana principal. Una vez identificado el tipo de consulta mediante el campo “viaAcceso” lanza el procedimiento correspondiente a la vía de acceso. Esto además permite en el futuro poder implementar un nuevo método de consulta de forma sencilla y sin apenas modificar código. Desde la ventana principal se hace una llamada a este procedimiento, pasando la instancia de ventana principal para que desde el procedimiento de consulta

pueda alterar la información del cronómetro así como rellenar el resultado del panel resultado.

Peticiones contiene un procedimiento dedicado para cada tipo de consulta:

- GET
- POST
- SOCKET
- URL
- SOAP

Los cinco procedimientos se encargan de preparar los datos de salida para que tengan compatibilidad con protocolo a utilizar y lanzan la consulta. A pesar de que hay protocolos muy similares entre si, se decidió encapsularlos por separado para evitar futuros problemas y tener un código más limpio y con una mayor trazabilidad.

Dada la naturaleza de los procesos, que tienen que hacer llamadas a servicios externos a la aplicación pudiéndose producir errores de todo tipo, todas las llamadas están preparadas para que, en caso de error, las excepciones que provoquen sean controladas, evitando que se expandan sin control por el programa. El usuario será informado del error concreto que originó la consulta y será mostrado en el cuadro de resultado de la prueba así como almacenado en el fichero de LOG, ya que en la propia excepción generada, se encuentra la descripción del motivo, que será de gran utilidad para el proceso de monitorización de las pruebas.

5.3 Diagramas de actividad

A continuación se muestran los diagramas de actividad de dos de las funcionalidades principales del programa.[1] Mediante estos diagramas se puede comprender mejor el funcionamiento interno del programa

Diagrama de Actividad – Lanzar Prueba

Este diagrama representa el lanzamiento de una prueba. Se puede apreciar los pasos que sigue el usuario para llegar a lanzar la prueba con una trama o sin ella. También se representa el funcionamiento del timeout del programa en caso de superarse el tiempo de respuesta fijado.

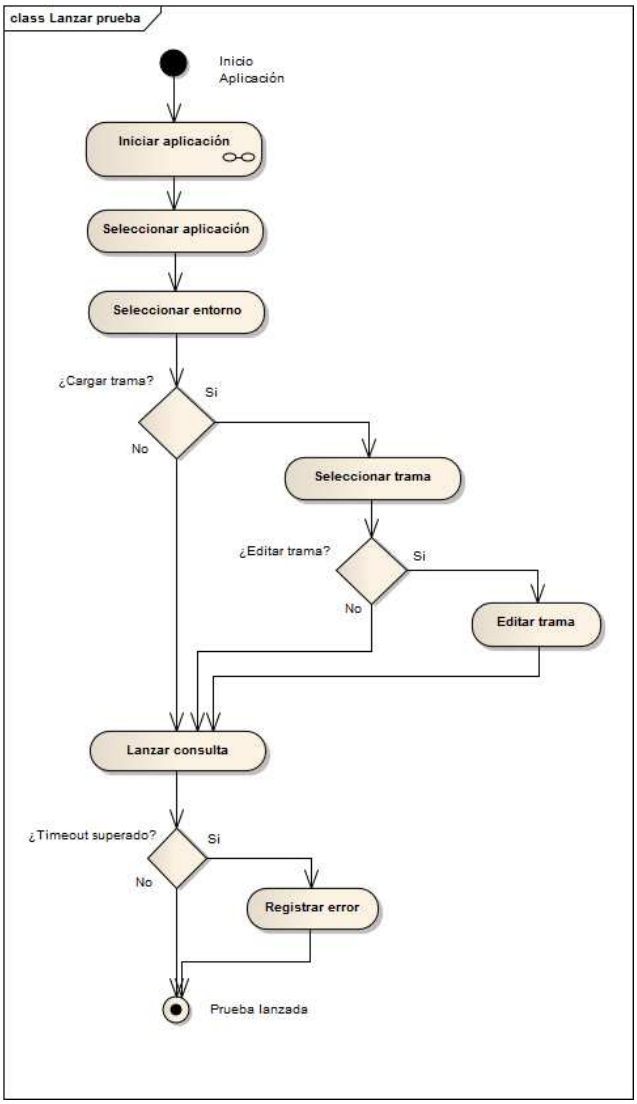


Figura 16: Diagrama actividad - Lanzar Prueba.

[1] No se incluyen los diagramas de todas las funcionalidades del sistema. El resto se representarían de forma similar a los mostrados.

Diagrama de Actividad – Editar Prueba

El siguiente diagrama representa la edición de una prueba desde el editor XML integrado. El usuario tiene la opción de elegir crear una nueva prueba o editar la prueba que estaba visualizando en ese momento.

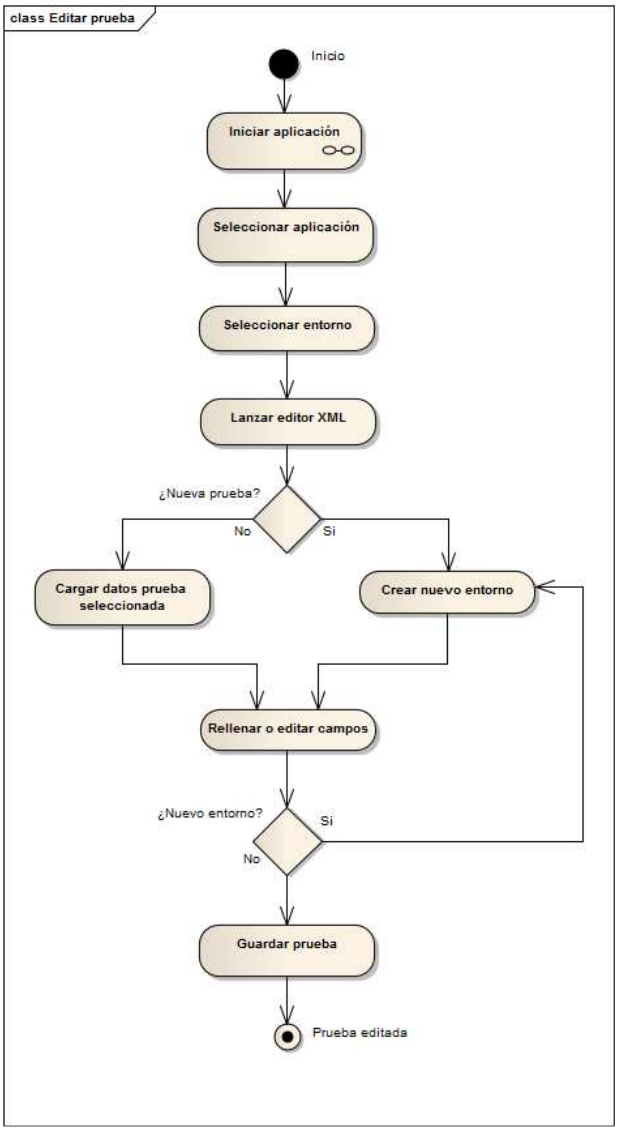


Figura 17: Diagrama Actividad - Editar Prueba.

5.4 Diagramas de secuencia

Diagrama secuencia – Lanzar Prueba

El siguiente diagrama muestra la secuencia de lanzamiento de una prueba.

Descripción:

Con la aplicación iniciada vamos a seleccionar la prueba de “Integración Gasper” en la lista de aplicaciones. La ventana principal se rellenará entonces con los datos relativos a esa prueba.

Ahora seleccionaremos el entorno de tránsito en el listado de entornos y se rellenarán los datos correspondientes a ese entorno. Llega el momento de cargar los datos que enviaremos en la consulta. Para ello pulsaremos sobre el botón “Cargar trama” y nos aparecerá una ventana de selección de archivo. Pulsaremos sobre el archivo llamado “Entrada incidencias.txt” y pulsamos en el botón “abrir”. En ese momento ya tendremos cargado el contenido del archivo en la ventana principal.

Procederemos entonces a modificar la trama para adecuarla a las condiciones que queramos. Para ello haremos click donde corresponda en el cuadro de texto y modificaremos los datos a nuestro gusto. Una vez finalizado ya el último paso sería simplemente pulsar el botón de lanzamiento de prueba. En ese momento se generaría la petición y se iniciaría la cuenta en el cronómetro. Cuando la aplicación de integración gasper nos devuelva la respuesta el cronómetro se parará y se mostrará el resultado de la prueba en la pantalla principal, finalizando así el test.

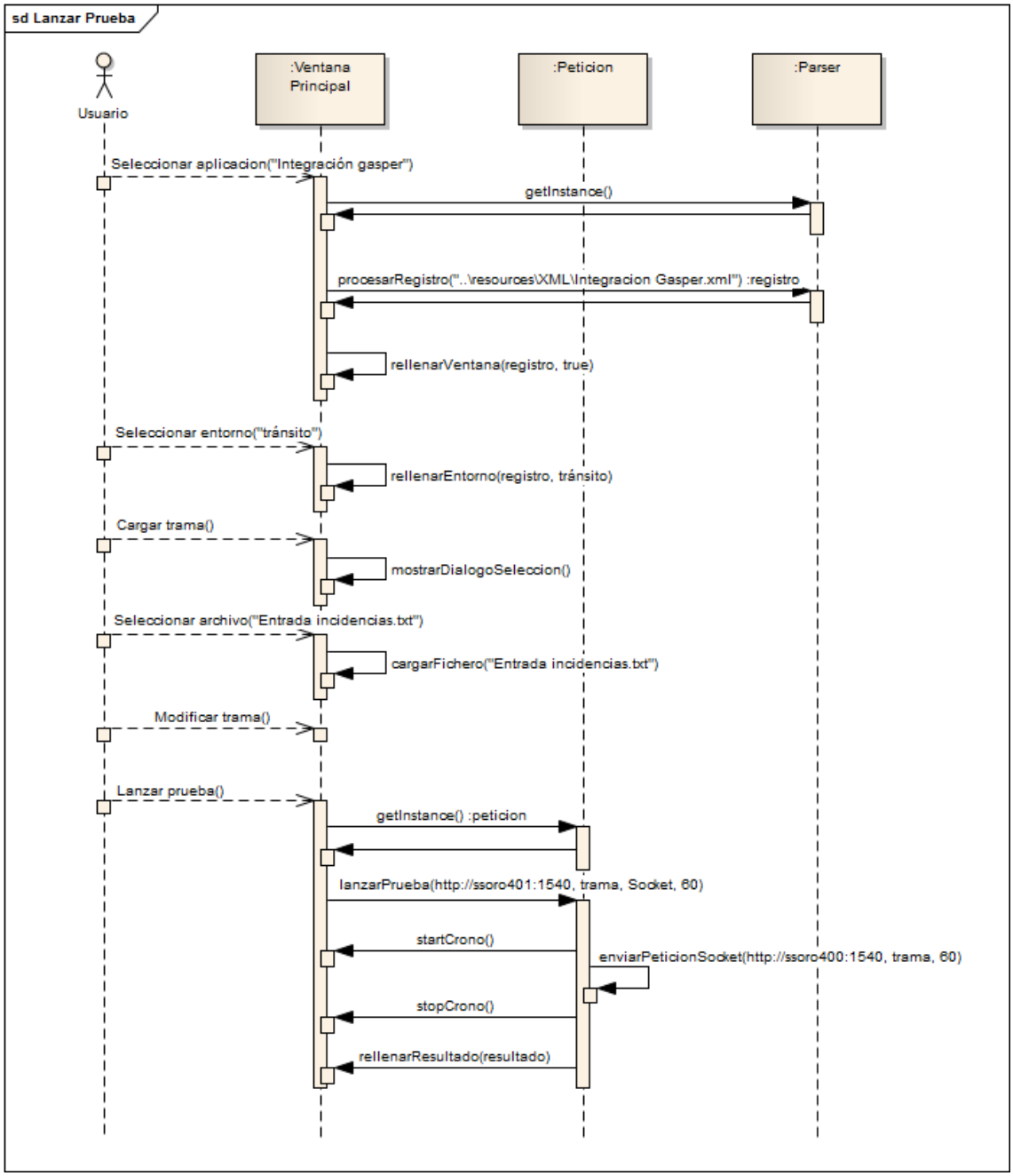


Figura 18: Diagrama de Secuencia - Lanzar Prueba.

Diagrama secuencia - Editar Prueba

Descripción:

Partimos con la aplicación iniciada correctamente y seleccionamos una prueba en el casillero de selección de prueba, en este caso vamos a seleccionar la aplicación: “Oficina telefónica automática”.

Pulsamos el botón “Editor XML”. En la ventana emergente seleccionamos la opción “Modificar Actual”. Se abrirá el editor cargando todos los datos de la prueba a editar. En este caso vamos a agregar un entorno nuevo a la aplicación “Oficina telefónica automática”. Pulsamos el botón “Agregar” y una ventana aparecerá, en la cual podremos seleccionar el nombre del entorno y agregar la URL correspondiente. En este caso elegiremos en el entorno “Tránsito” y escribiremos en la caja de texto la dirección URL: “http://ssoro401:24815/OT/FrontController”.

En la casilla comentario escribimos el comentario correspondiente para la prueba: “Requiere como parámetro una trama”.

Por último hay que indicar la ubicación de la trama. Así que pulsamos el botón “Explorar”. Se abrirá entonces una pantalla en la que tendremos que navegar hasta la carpeta en donde se encuentra la trama. Una vez seleccionada la carpeta pulsamos aceptar y volveremos a la pantalla del Editor XML.

Para terminar la edición y guardar los cambios pulsamos el botón “Guardar”. Observaremos entonces que en el título de la ventana desaparece un asterisco que es el que indica que la prueba actualmente editada no se ha guardado. Ya podríamos entonces salir del editor y volver a la ventana principal.

En el caso de que hubiéramos pulsado el botón salir sin haber guardado hubiera aparecido una ventana de advertencia informándonos de que la prueba no se ha guardado.

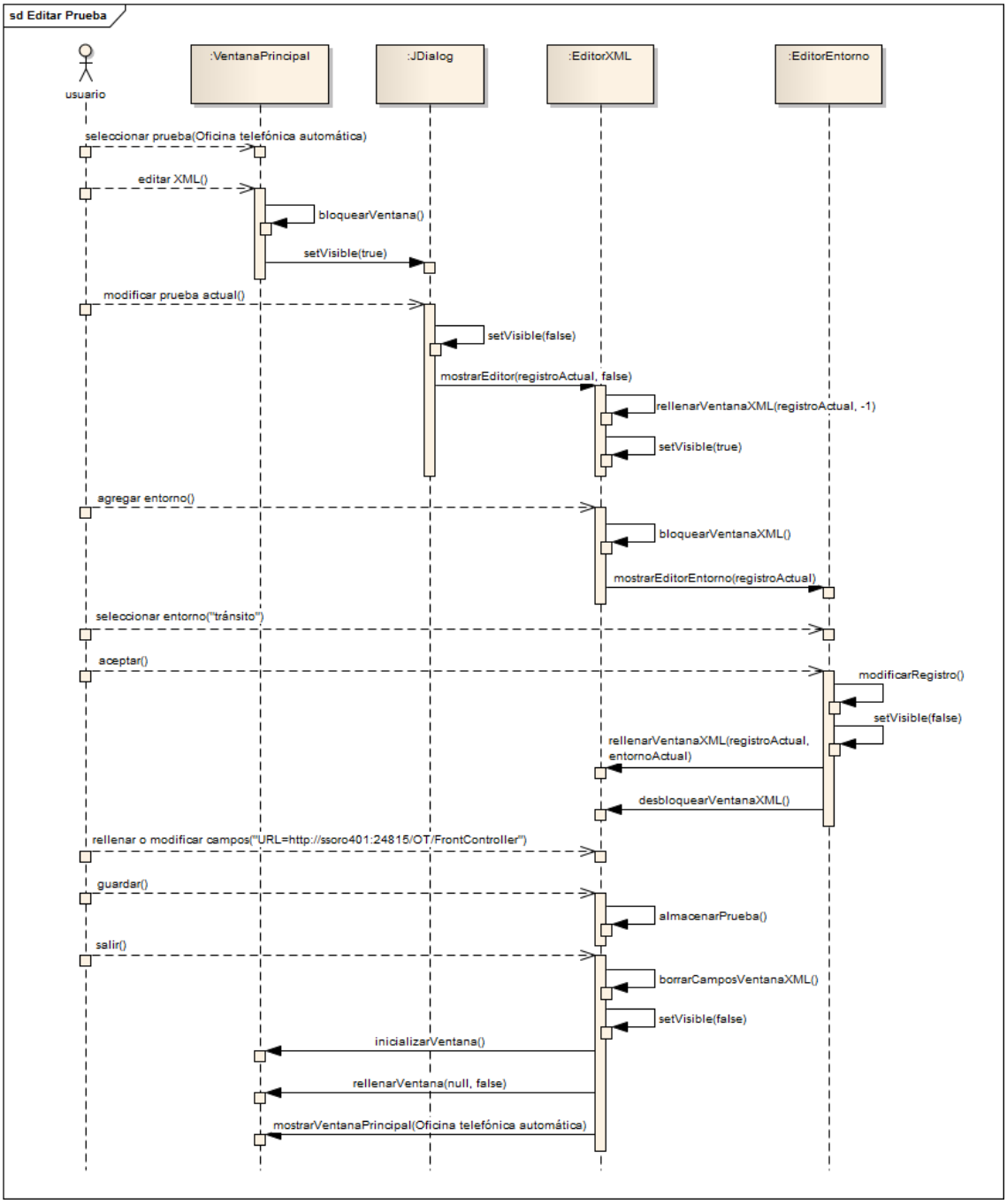


Figura 19: Diagrama de Secuencia - Editar Prueba.

Diagrama de Secuencia – Editar Trama

Descripción:

Desde la ventana principal el usuario pulsa el botón “Cargar Trama”, mostrándose un diálogo de selección de archivo. Una vez seleccionado el archivo a abrir, el programa carga la información del archivo en la pantalla principal. El usuario entonces puede editar la información que crea pertinente. Tras pulsar en “guardar trama” el sistema muestra en cuadro de diálogo, el usuario entonces podrá sobrescribir el fichero o crear uno nuevo.

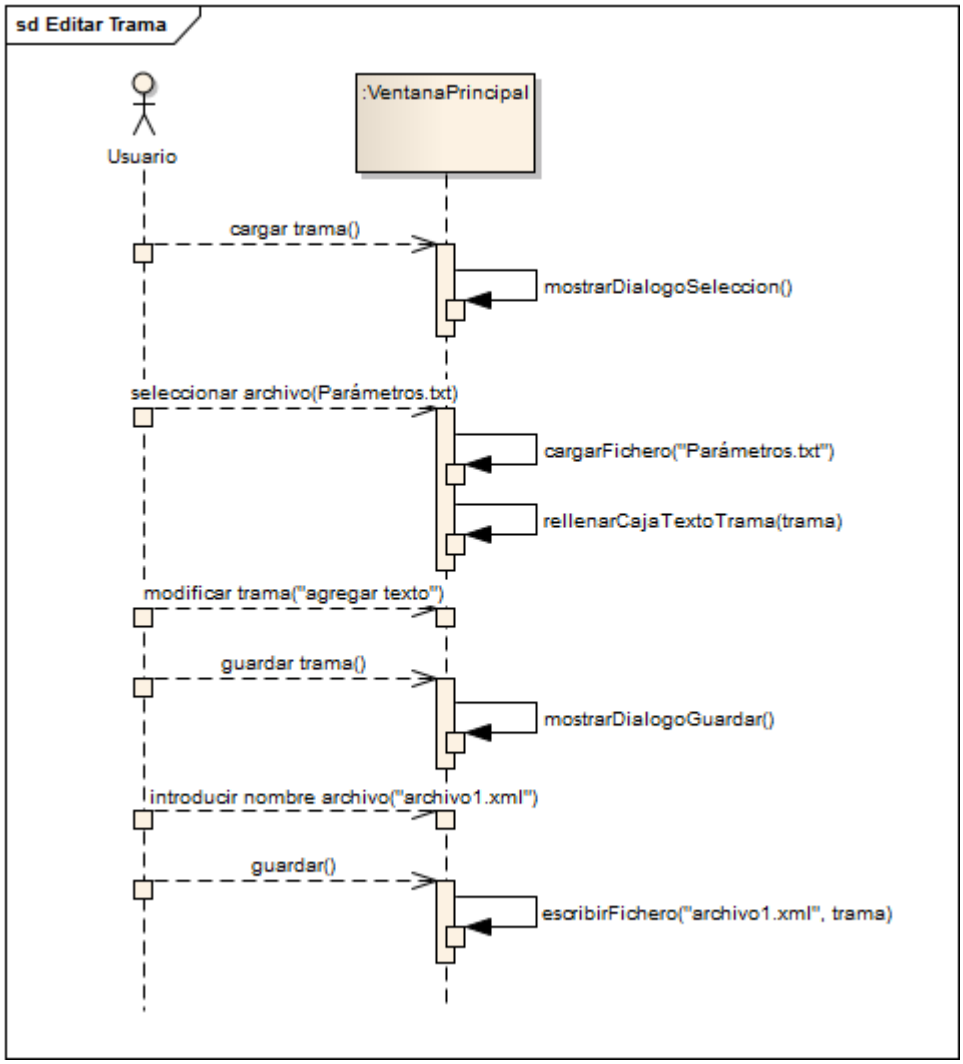


Figura 20: Diagrama de Secuencia - Editar Trama.

Diagrama de Secuencia – Guardar Resultado

Descripción:

Partiendo de una prueba lanzada y con la respuesta ya recibida, el usuario pulsa el botón de guardar resultado. El programa prepara los datos y muestra una pantalla donde el usuario debe introducir el nombre de archivo. Una vez introducido el programa almacena el resultado de la prueba.

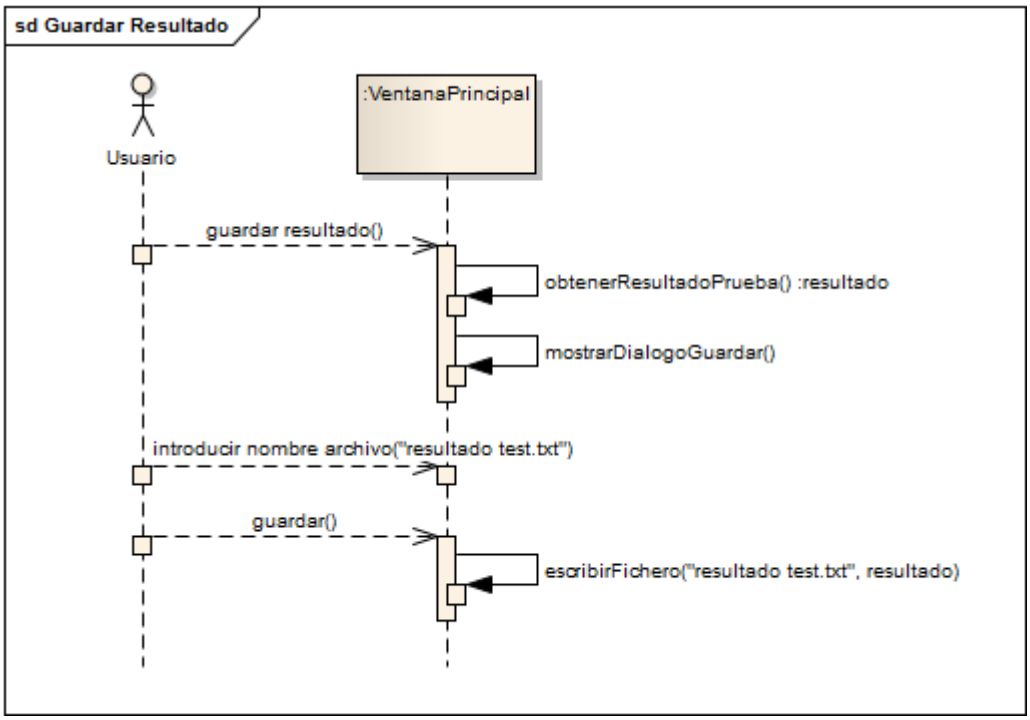


Figura 21: Diagrama de Secuencia - Guardar Resultado.

6. IMPLEMENTACIÓN DEL SISTEMA

En este apartado veremos el resultado final de la aplicación MOPROM y explicaremos brevemente todas las funcionalidades del programa y su funcionamiento interno, así como su interfaz definitiva.

6.1 MOPROM: un vistazo rápido

MOPROM es el Módulo de Pruebas de Operaciones Multicanales. Nace con la necesidad de agrupar y unificar las pruebas llevadas en el departamento de Banca Telefónica de CajaMadrid a los distintos servicios y aplicaciones mantenidas.

El objetivo principal de MOPROM es proporcionar una plataforma unificada al usuario en una interfaz interactiva que permita realizar pruebas a los sistemas mantenidos.

Desde la interfaz principal de MOPROM podemos ver un listado de las distintas aplicaciones soportadas por la aplicación y acceder a datos técnicos sobre las mismas. Además para cada aplicación podremos lanzar una prueba al servicio o aplicación para comprobar su funcionamiento.

Para realizar estas pruebas tan solo hay que ejecutar el programa y la intuitiva interfaz nos guiará en el proceso para probar los servicios. Tan solo con seleccionar la aplicación y pulsar sobre el botón “Lanzar Prueba”, ya podríamos realizar un testeo sobre la aplicación. Las pruebas pueden lanzarse hacia distintas url’s dependiendo del estado seleccionado para la prueba, bien puede ser en desarrollo, real, tránsito o pruebas. Asimismo cada prueba puede llevar adjuntos datos (trama XML) o parámetros, éstos serán cargados desde la ventana principal del programa. Una vez realizada la prueba el programa mostrará el resultado en la caja de resultado de prueba de forma que el usuario podrá comprobar si la prueba se ha realizado o no con éxito.

El usuario tiene la posibilidad además de una vez realizada una prueba guardar el resultado de la prueba en un informe en formato “.txt”, o un informe copiado al portapapeles, lo cual lo hace muy útil para pegar en un correo electrónico por ejemplo.

Para enviar tramas de datos o parámetros en las pruebas, el programa cuenta con un selector de archivos y editor en la propia interfaz de los datos que se van a enviar. Por ejemplo podemos encontrarnos que hay que modificar un código de cliente en una trama antes de enviarla. De este modo hacemos el proceso de pruebas mucho más eficaz, cómodo y rápido. Además permite guardar los cambios realizados en el archivo, o generar nuevos archivos de parámetros o tramas.

El programa cuenta además con un cronómetro que muestra el tiempo empleado en la prueba además de funcionar como un temporizador, de forma que si indicamos que una prueba, por ejemplo, tenga un “timeout” de 10 segundos, en caso de excederse ese tiempo en el cronómetro la prueba será considerada como no válida y el test finalizará con un aviso.

6.2 Características del programa

A continuación se detallan las características más importantes de MOPROM:

- Envío de peticiones mediante los siguiente protocolos:

  - POST
  - GET
  - SOCKET
  - URL
  - SOAP

- Carga y edición de tramas a enviar en formato XML.
- Carga y edición de parámetros a enviar en formato TXT.
- Edición de pruebas mediante un editor XML incorporado.
- Fichero LOG de errores ocurridos durante el programa.
- Generación de informes en formato TXT con el resultado de la prueba.
- Volcado en portapapeles de informe completo de prueba.
- Timeout configurable para cada prueba.
- Cronómetro en tiempo real con la duración de las consultas.
- Posibilidad de lanzar las llamadas por protocolo GET a través de Internet Explorer.

6.3 Interfaz del programa[2]

Para el diseño de la interfaz se optó por dividirla en dos bloques, uno para la elección de la prueba y otro para el panel de resultados, pero el departamento consideró oportuno que las tramas o parámetros que se enviasen en las consultas pudieran ser editables en pantalla, con lo que hubo que rediseñar la interfaz para albergar esa opción. De este modo la interfaz quedó dividida en tres grandes bloques que se muestran a continuación:

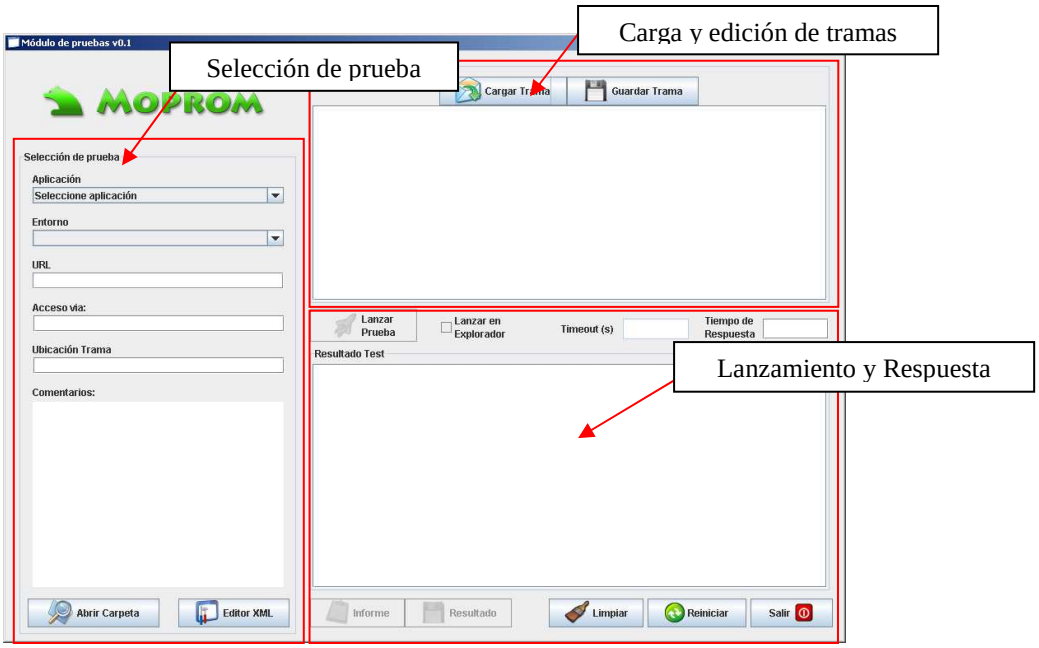


Figura 22: Interfaz de usuario (Ventana Principal).

Asimismo, la aplicación dispone de un editor propio de pruebas XML. Éste se ha integrado mediante otra pantalla ya que incluirlo en la interfaz principal no era una prioridad y además añadiría complejidad a la misma, por lo que se ha optado por separarlo de la pantalla principal. Al pulsar el botón “Editor XML” se muestra el editor, cuya interfaz es la siguiente:

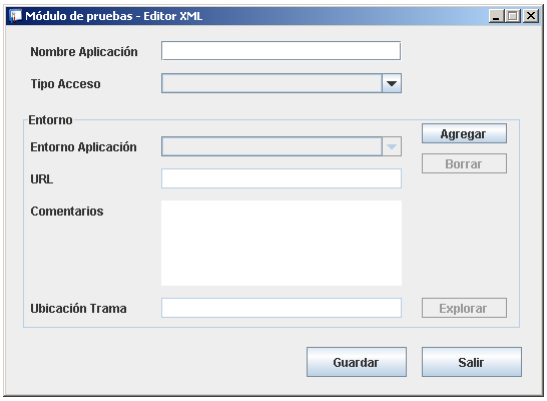


Figura 23: Interfaz de usuario (Editor XML).

[2] Para una descripción más detallada de la interfaz y su funcionamiento consultar el manual de usuario en el Anexo.

6.4 ¿Cómo funciona?

6.4.1 Arranque del programa

El programa consiste en una compilación del código java en un archivo ".jar", que una vez cargado mediante la máquina virtual Java, inicia la carga de un fichero llamado "config.ini".

Para el correcto arranque y funcionamiento del programa, el fichero de configuración (config.ini) que debe contener 17 parámetros, que incluyen directorios de trabajo, archivos clave (por ejemplo la plantilla .XSD), otros parámetros de configuración y todos los iconos e imágenes del programa.

```
plantillaXSD        = resources\\XML\\plantilla-pruebas.xsd
logoSuperior        = resources\\data\\logo moprom cabecera.png
iconoExit           = resources\\data\\Windows-Turn-Off-24x24.png
iconoLanzar         = resources\\data\\launch-32x32.png
iconoPortapapeles = resources\\data\\clipboard-32x32.png
iconoGuardar        = resources\\data\\Floppy-32x32.png
iconoExplorar       = resources\\data\\folder-explorer-32x32.png
iconoReiniciar      = resources\\data\\Button-Refresh-24x24.png
iconoLimpiar        = resources\\data\\Paintbrush-32x32.png
iconoCargar         = resources\\data\\ei0021-32x32.png
iconoEditar         = resources\\data\\Windows-Tools-32x32.png
iconoAplicacion   = resources\\data\\Windows-Luna-32x32.png
tiempoSplash        = 5000
ficherosXML         = resources\\XML
rutaDefectoTramas = resources\\XML\\Tramas
imagenSplash        = resources\\data\\logo moprom splash.png
default_timeout   = 60
```

Figura 24: Fichero "config.ini"

Una vez cargados los parámetros de entrada y localizados los ficheros mediante las rutas indicadas en el fichero “config.ini”, el programa localiza la plantilla de pruebas “plantilla-pruebas.xsd” que será utilizada para parsear los ficheros que contienen los parámetros de las pruebas y de este modo rellenar los campos de la ventana principal. De este modo ya tenemos cargado el catálogo de pruebas que podemos utilizar en el programa.

Una vez se hayan procesado todos los ficheros de pruebas, el programa está listo para mostrar la lista de pruebas en la ventana principal. Podremos seleccionar así en el desplegable de aplicación la prueba que queramos realizar, y de forma automática se cargarán los parámetros relativos a dirección URL, método utilizado para la prueba, comentarios, entornos, etc...

Aquí se puede apreciar el diagrama de actividad del arranque del programa:

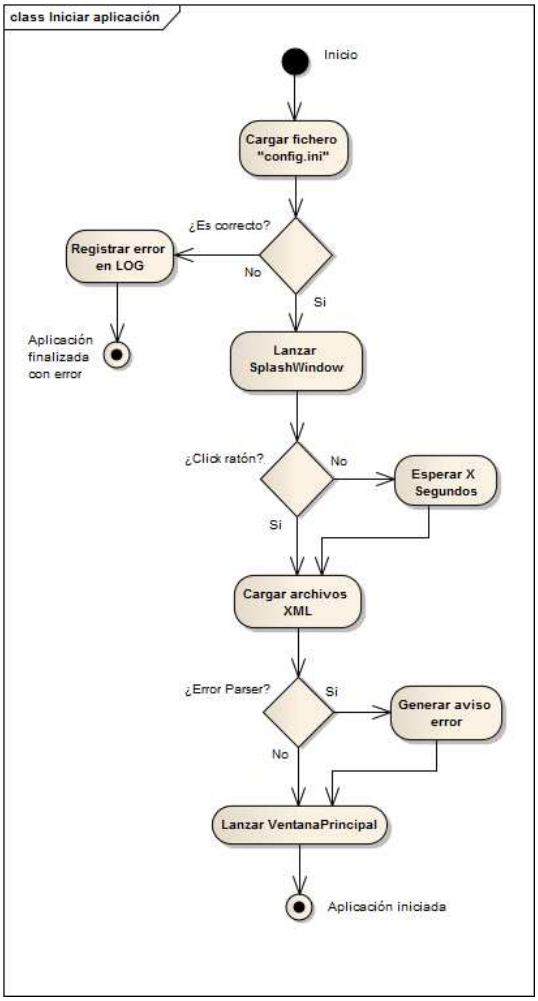


Figura 25: Diagrama de Actividad - Iniciar Aplicación.

Para entrar más en detalle en la importación de los ficheros XML y su validación, se adjunta también el diagrama de secuencia del arranque de la aplicación.

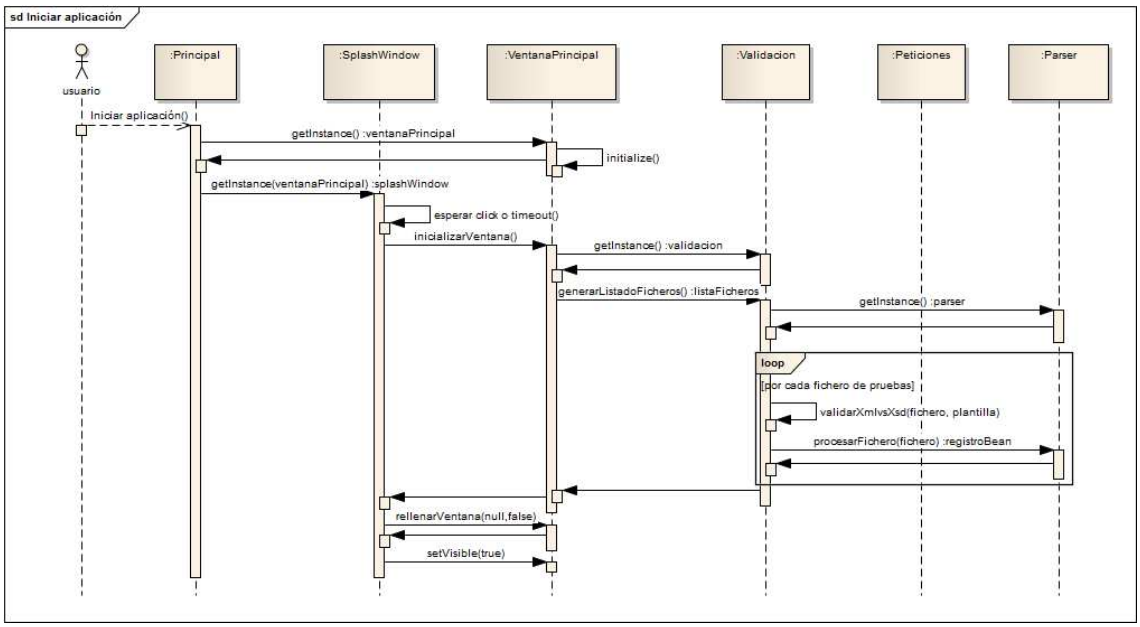


Figura 26: Diagrama de Secuencia - Iniciar Aplicación.

En lo relativo al lanzamiento de la prueba, algunas pruebas requieren el envío de datos en forma de trama, o de parámetros. Para ello la ventana principal dispone de un botón que permite la carga de un fichero tipo trama XML a enviar, o bien de parámetros, que será procesado para la extracción de los mismos mediante un parser. En el caso de ser necesario editar alguna información, parámetro o simplemente escribir desde cero los datos a enviar, esto se puede hacer desde la propia ventana principal de forma que el usuario puede interactuar de forma directa sobre la petición que está a punto de enviar a la aplicación.

Seleccionada la aplicación, su entorno y lo que vamos a enviar en la petición (trama o parámetros), MOPROM está listo para enviar su petición, pero antes, si el usuario quiere, tiene la posibilidad de alterar el "timeout" por defecto del programa (indicado en el fichero config.ini), y de esta forma se alterará el tiempo que espera el programa antes de considerar que una llamada ha fallado. Asimismo, en las peticiones de tipo GET, el usuario tiene la opción de lanzar la consulta en el navegador Internet Explorer de forma externa al programa, con lo que MOPROM no comprobará si la petición se ha realizado correctamente.

Una vez la aplicación que está sometida al test manda la respuesta, o el timeout llega a su fin, MOPROM mostrará el resultado de la prueba en el recuadro de "resultado test", indicando si ha habido un fallo mediante un mensaje de error, o bien mostrando la respuesta de la aplicación remota a la llamada desde MOPROM.

6.4.2 Organización de las pruebas

Para la organización de las pruebas se ha optado por ubicarlas en el directorio de trabajo de programa como se muestra en la figura a continuación:

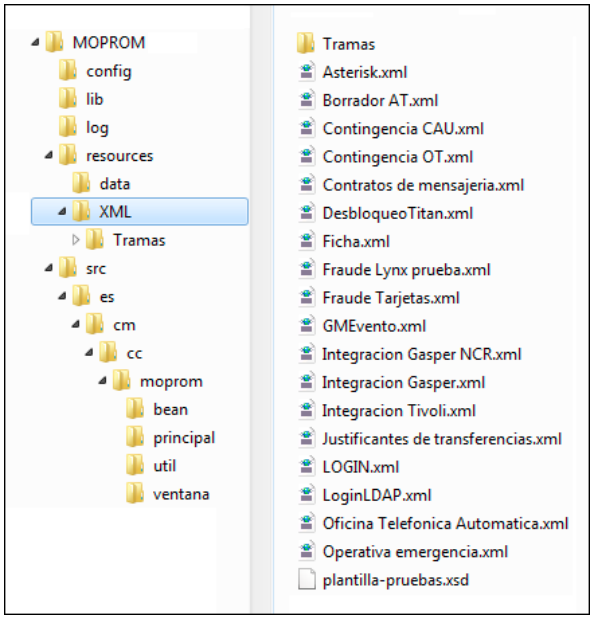


Figura 27: Configuración directorios MOPROM.

Como se puede apreciar en la figura anterior las pruebas se encuentran ubicadas en el directorio “XML” junto con la plantilla de pruebas en formato “XSD”. Dentro del directorio “XML” se encuentra también el directorio “Tramas” donde se encuentran ubicadas los datos que pueden ser enviados en la prueba, bien sea una trama “XML” o simplemente archivos de texto que albergan parámetros.

Estas pruebas se encuentran a su vez divididas entre sus entornos de trabajo. Éstos son:

- Desarrollo
- Tránsito
- Real
- Otros

Como se puede apreciar en la siguiente figura, los distintos tipos de pruebas están divididos en subdirectorios de acuerdo a su entorno de trabajo correspondiente.

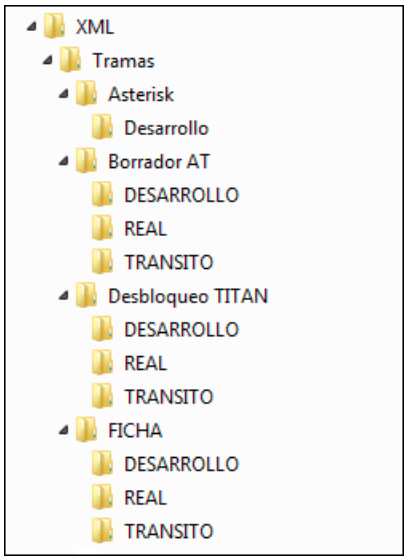


Figura 28: Configuración directorios tramas.

Esta estructura dota de un mayor control y organización de las pruebas. Además de permitir de una forma eficaz una subdivisión rápida entre los distintos adjuntos que pueden lanzarse en las pruebas. Como método de seguridad cuando en la aplicación se seleccione el entorno de tránsito por ejemplo, al cargar una trama se abrirá por defecto la carpeta de tránsito, evitando así que se mezclen entre si las tramas de distintos entornos y enviar una petición con las tramas incorrectas.

A continuación se muestra el contenido de un archivo XML de pruebas:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <test>
<aplicacion>GMEvento</aplicacion>
- <entornos>
<entorno>REAL</entorno>
<url_entorno>http://gmeventos.cm.es/ws-gmm-01.00.00/FrontWS</url_entorno>
<rutaTrama>GMEventos\Real</rutaTrama>
<comentarios>Aplicacion Web services. Hay que mandar una trama XML</comentarios>
</entornos>
- <entornos>
<entorno>DESARROLLO</entorno>
<url_entorno>http://sliro297:11305/ws-gmm-01.00.00/FrontWS</url_entorno>
<rutaTrama>GMEventos\Desarrollo</rutaTrama>
<comentarios />
</entornos>
- <entornos>
<entorno>TRANSITO</entorno>
<url_entorno>http://sliro298:11305/ws-gmm-01.00.00/FrontWS</url_entorno>
<rutaTrama>GMEventos\Transito</rutaTrama>
<comentarios />
</entornos>
<tipoAcceso>SOAP</tipoAcceso>
</test>
```

Figura 29: Fichero ejemplo XML.

En el archivo de ejemplo podemos apreciar la forma en la que está almacenada la información de la prueba. Para más información sobre los ficheros XML consultar el apartado “Modelo de datos” en el Diseño del sistema.

6.4.3 Lanzamiento de pruebas

En el siguiente apartado se indican los pasos a seguir para realizar el lanzamiento de una prueba desde la aplicación MOPROM.

- Ejecutar la aplicación: Una vez ejecutada, se cargarán los ficheros de pruebas XML en el programa.

- Seleccionar una aplicación: En la lista desplegable de aplicaciones tendremos que seleccionar un elemento. Una vez seleccionado se cargarán los datos relativos a la prueba.

- Seleccionar un entorno: En la lista desplegable de entornos elegiremos el entorno sobre el que ejecutar la prueba. Una vez seleccionado (por defecto se selecciona el primer entorno creado) se cargará la URL asociada a ese entorno.

- Carga de trama o parámetros: En caso de que la prueba requiera la carga de determinada trama o de un fichero de parámetros, pulsaremos el botón “Cargar trama” y seleccionaremos el fichero a cargar.

- Configuración de trama o parámetros: Si procede editar la información a enviar en el panel de carga de tramas.

- Configuración del Timeout: Para modificar el timeout tan solo hay que pulsar sobre el cuadro de texto del timeout y marcar un nuevo valor en segundos.

- Lanzamiento de prueba: Al pulsar el botón de lanzamiento de prueba el programa recoge todos los datos y ejecuta la consulta. Un mensaje de información aparecerá en el panel de resultado y el cronómetro se iniciará. Una vez concluida la prueba se mostrará el resultado de la consulta y el tiempo empleado.

- Tratamiento del resultado: Una vez la prueba esté concluida, tenemos la posibilidad de generar un informe con todos los datos relativos a la prueba. El informe se generará en el portapapeles de forma que pueda ser pegado en un procesador de textos o en un correo electrónico. También cabe la posibilidad de que el resultado de la prueba puede guardarse en un fichero de texto “.txt”.

7. PLANIFICACIÓN

7.1 Descripción

En esta sección se describe brevemente la planificación y la duración de las fases del proyecto. Se acompaña una tabla resumen con la descripción de las actividades y un diagrama Gantt.

El proyecto se ha dividido en cinco fases principales

- Análisis del sistema
- Diseño del sistema
- Implementación
- Integración y pruebas
- Documentación

En la fase de análisis, comienzan los primeros esbozos sobre de que va a tratar la aplicación y de su entorno, así como conocer otras aplicaciones de la empresa y su sistema de desarrollo. Asimismo tras unas reuniones con el cliente se pudieron ir documentando los primeros casos de uso y los requisitos software.

En el diseño del sistema se crearon los diagramas de clases de los componentes del sistema y su relación y también se pulieron algunos detalles y se desarrollaron otros usos que iban surgiendo sobre las funcionalidades del sistema, ya que a medida que se iba realizando el diseño iban surgiendo nuevas e interesantes ideas para aplicar al proyecto.

En la fase de Implementación se comenzó a programar en el entorno de RCA (eclipse) en el lenguaje Java. Se comenzó por implementar las interfaces para más tarde hacer funcional el programa. De esta forma se podía cuidar el diseño desde el primer día, y así satisfacer al cliente en este aspecto tan importante. La mayoría del tiempo invertido en el proyecto se ha dado en esta fase al ser la más compleja.

La fase de integración y pruebas, se ha ido realizando prácticamente de forma paralela al desarrollo de las funcionalidades en la fase de implementación, ya que el programa tiene que ser muy estable y la fiabilidad debe que estar garantizada antes de su fecha de finalización.

Dado que el cliente indicó que debe realizarse un manual de usuario con los aspectos del funcionamiento de la aplicación, su puesta en marcha y mantenimiento por parte del personal del departamento, tuvo que realizarse un manual de usuario acorde con el estilo que utiliza CajaMadrid en los manuales de sus aplicaciones propias, ciñéndose a un estilo y apartados clave proporcionados por el cliente.

A continuación se muestra la tabla con cada una de las fases del proyecto y su duración aproximada, así como el diagrama de Gantt que representa las fases y ordenación de las tareas en el calendario.

7. 2 Fases y duración del proyecto

En la siguiente tabla se muestran las fases del proyecto junto con su duración y fechas de inicio y fin. El proyecto está dividido en las siguientes fases:

- Análisis del sistema
- Diseño del sistema
- Implementación
- Integración de pruebas
- Documentación

La duración total es de 132 jornadas laborables realizadas por una sola persona.

Task Name	Duration	Start	Finish
Proyecto	**132 days**	**Tue 01/07/08**	**Wed 31/12/08**
Análisis del sistema	**29 days**	**Tue 01/07/08**	**Fri 08/08/08**
Estudio del sistema	12 days	Tue 01/07/08	Wed 16/07/08
Obtención de requisitos de sistema	9 days	Tue 01/07/08	Fri 11/07/08
Obtención de casos de uso	11 days	Thu 17/07/08	Thu 31/07/08
Modelos de datos	7 days	Thu 17/07/08	Fri 25/07/08
Documentación del análisis	6 days	Fri 01/08/08	Fri 08/08/08
Diseño del sistema	**27 days**	**Fri 01/08/08**	**Mon 08/09/08**
Diseño de la arquitectura	4 days	Fri 01/08/08	Wed 06/08/08
Diseño de las interfaces	15 days	Thu 07/08/08	Wed 27/08/08
Diseño de la base de datos	10 days	Thu 07/08/08	Wed 20/08/08
Definición de componentes	9 days	Thu 21/08/08	Tue 02/09/08
Documentación del diseño	4 days	Wed 03/09/08	Mon 08/09/08
Implementación	**68 days**	**Tue 09/09/08**	**Thu 11/12/08**
Desarrollo de interfaz principal	30 days	Tue 09/09/08	Mon 20/10/08
Funcionalidad carga pruebas	10 days	Tue 09/09/08	Mon 22/09/08
Funcionalidad editor pruebas	15 days	Tue 23/09/08	Mon 13/10/08
Funcionalidad carga parámetros	10 days	Tue 14/10/08	Mon 27/10/08
Funcionalidad lanzamiento de pruebas	20 days	Tue 28/10/08	Mon 24/11/08
Funcionalidad generación de informes	8 days	Tue 25/11/08	Thu 04/12/08
Documentación implementación	5 days	Fri 05/12/08	Thu 11/12/08
Integración y pruebas	**78 days**	**Tue 09/09/08**	**Thu 25/12/08**
Desarrollo del lote de pruebas	35 days	Tue 09/09/08	Mon 27/10/08
Pruebas carga pruebas	20 days	Tue 23/09/08	Mon 20/10/08
Pruebas editor	16 days	Tue 14/10/08	Tue 04/11/08
Pruebas carga parámetros	23 days	Tue 28/10/08	Thu 27/11/08
Pruebas lanzamiento de test	15 days	Tue 25/11/08	Mon 15/12/08
Pruebas generación de informes	4 days	Fri 05/12/08	Wed 10/12/08
Documentación Integración y pruebas	11 days	Thu 11/12/08	Thu 25/12/08
Documentación extra	**3 days**	**Mon 29/12/08**	**Wed 31/12/08**
Manual de usuario	3 days	Mon 29/12/08	Wed 31/12/08

Figura 30: Fases y duración del proyecto.

7.3 Diagrama de Gantt

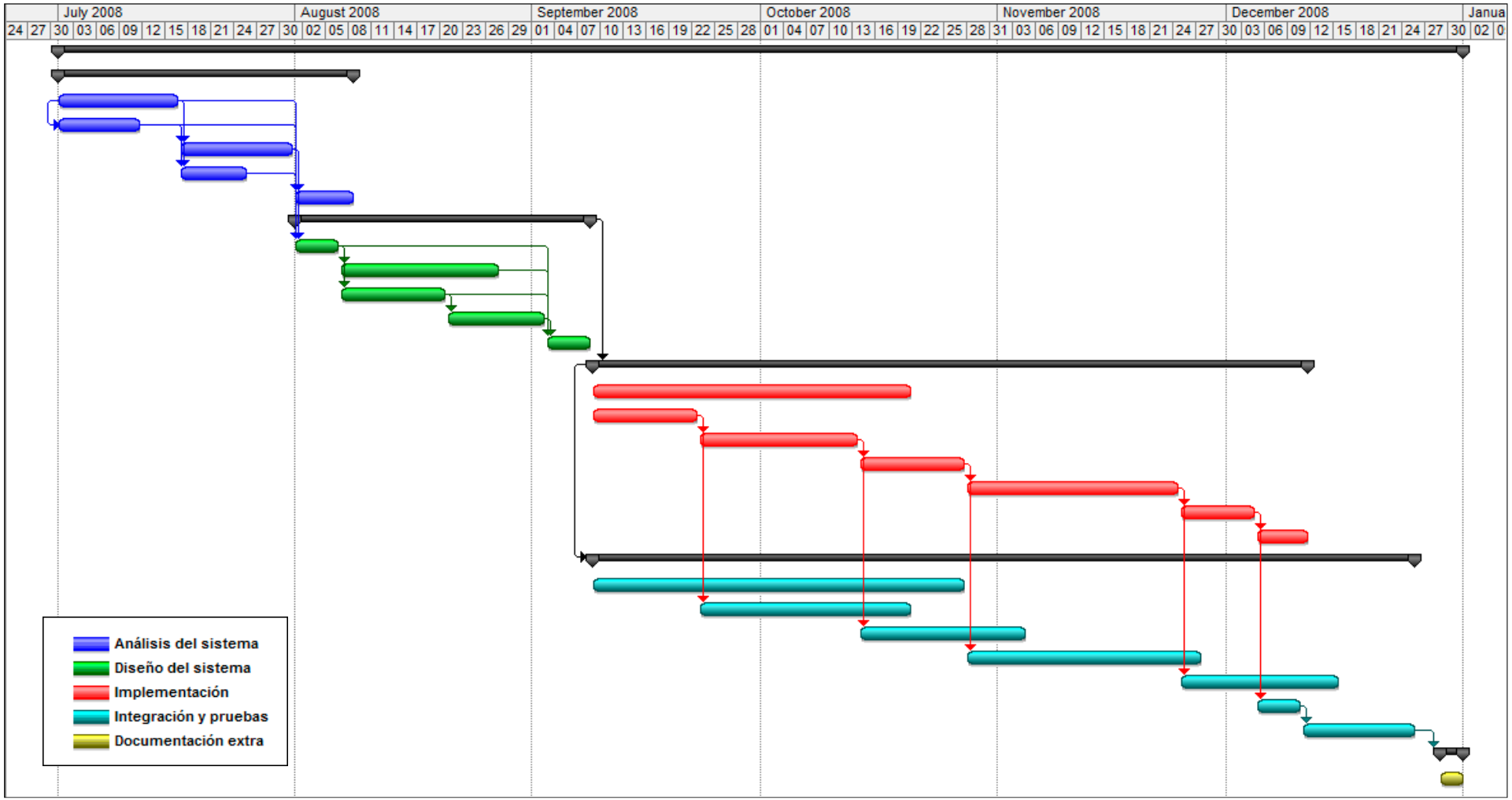


Figura 31: Diagrama de Gantt

8. PRESUPUESTO Y ESTIMACIÓN DE COSTES

8.1 Descripción

En este punto se mostrarán detallados los distintos gastos asociados al proyecto, tanto de mano de obra, como de material hardware y software.

Cada gasto estará detallado en una tabla resumen. En el caso de los recursos hardware y software estos están cerrados al precio de mercado en el momento de la realización del proyecto. Los gastos de personal están estimados al salario medio de un Ingeniero técnico informático sin experiencia previa trabajando 7 horas diarias.

Todos los gastos se muestran con el IVA incluido. Además se incluyen otros gastos derivados.

Los gastos a tratar en el documento se resumen en:

- Gastos Recursos Humanos
- Gastos Hardware
- Gastos Software
- Otros gastos

8.2 Gastos Recursos Humanos

Gracias al desglose de actividades obtenido en Microsoft Project podemos ver el coste por cada actividad en la siguiente tabla.

Actividad	Días	Horas	Total
Proyecto			
Análisis del sistema			
Estudio del sistema	8	56	560 €
Obtención de requisitos de sistema	6	42	420 €
Obtención de casos de uso	7	49	490 €
Modelos de datos	5	35	350 €
Documentación del análisis	4	28	280 €
Total análisis del sistema	**19**	133	1.330 €
Diseño del sistema			
Diseño de la arquitectura	3	21	210 €
Diseño de las interfaces	10	70	700 €
Diseño de la base de datos	6	42	420 €
Definición de componentes	6	42	420 €
Documentación del diseño	3	21	210 €
Total diseño del Sistema	**17**	119	1.190 €
Implementación			
Desarrollo de interfaz principal	19	133	1.330 €
Funcionalidad carga pruebas	6	42	420 €
Funcionalidad editor pruebas	10	70	700 €
Funcionalidad carga parámetros	6	42	420 €
Funcionalidad lanzamiento de pruebas	13	91	910 €
Funcionalidad generación de informes	5	35	350 €
Documentación implementación	3	21	210 €
Total implementación	**44**	308	3.080 €
Integración y pruebas			
Desarrollo del lote de pruebas	23	161	1.610 €
Pruebas carga pruebas	13	91	910 €
Pruebas editor	10	70	700 €
Pruebas carga parámetros	15	105	1.050 €
Pruebas lanzamiento de test	10	70	700 €
Pruebas generación de informes	3	21	210 €
Documentación Integración y pruebas	7	49	490 €
Total integración y pruebas	**50**	350	3.500 €
Documentación extra			
Manual de usuario	2	14	140 €
Total documentación extra	**2**	14	140 €
Total Proyecto	**132**	924	9.240 €

Figura 32: Tabla presupuesto RRHH desglosado.

Nota: Se ha estimado un coste bruto de 10 euros la hora trabajada, y jornadas de 7 horas al día.

8.3 Gastos Hardware

Para la realización del proyecto se utilizó el ordenador de sobremesa estándar de los puestos de trabajo del departamento, que contaba con las siguientes especificaciones hardware:

- Ordenador: HP Compaq DC7100
    - Procesador: Pentium 4 - 3 GHz
    - Memoria RAM: 256 MB
    - Disco duro: 80 GB
- Monitor: TFT Samsung 17”.

En total los costes del equipo ascienden a:

Equipos informáticos	Coste
Ordenador HP Compaq DC7100	600 €
Monitor TFT Samsung 17”	120 €
TOTAL	720 €

Figura 33: Tabla resumen equipos informáticos.

8.4 Gastos Software

Para contabilizar el gasto en recursos software se muestra un listado del software utilizado en el desarrollo del proyecto así como su coste aproximado.

Programas Software	Coste
Windows XP Professional Service Pack 3	250 €
Microsoft Office 2003 Professional	260 €
Microsoft Project 2003	700 €
Enterprise Architect 7.5	200 €
Java JDK	0 €
I.B.M. Rational Software Architect 7.0	2785 €
Adobe PhotoShop CS4	1200 €
TOTAL	5395 €

Figura 34: Tabla resumen programas software.

8.5 Otros Gastos

Durante los 6 meses de estancia en el departamento, se utilizó un puesto de trabajo estándar de empleado, con electricidad, línea telefónica IP e Internet, además de soporte técnico ofrecido por el CAU de CajaMadrid.

Los gastos de alquiler del puesto se detallan en la siguiente tabla:

Otros Gastos	Coste
Alquiler de puesto informático (6 meses)	1200 €
TOTAL	1200 €

Figura 35: Tabla resumen otros gastos.

8.6 Resumen del presupuesto del proyecto

En la siguiente tabla se detalla el presupuesto total del proyecto como suma de los apartados anteriores.

Concepto	Coste
Gastos recursos humanos	9240 €
Gastos hardware	720 €
Gastos software	5395 €
Otros gastos	1200 €
TOTAL	16555 €

Figura 36: Tabla resumen presupuesto proyecto.

9. CONCLUSIONES

Ha sido difícil enfrentarme a un proyecto de estas características, ya que nunca había desarrollado una aplicación gráfica tan completa desde cero. El uso de un lenguaje de programación desconocido para mi (Java) supuso un obstáculo al tener que sumar al desarrollo del proyecto el aprendizaje de un lenguaje y sus herramientas de desarrollo, no obstante al tener una buena base de conocimientos en la programación orientada a objetos no supuso un esfuerzo tan grande como el que yo esperaba.

A título personal el principal aporte que me ha dado este proyecto es adquirir de forma práctica la conciencia de cómo funcionan los procesos de desarrollo de un producto software y como se puede, paso a paso, abstraer e interpretar un problema de la vida real y lograr modelarlo a una solución software adecuada para ese problema mediante las herramientas y conocimientos existentes.

La etapa de diseño ha sido la más importante en el desarrollo del proyecto y la que más me ha aportado como futuro ingeniero. Gracias a ella he aprendido a valorar la necesidad de realizar un diseño exhaustivo sobre el conjunto de requisitos, y tener en cuenta que pueden ocurrir cambios en las mismas, de modo que la plataforma que estamos desarrollando debe ser lo bastante compacta y modular como para afrontar todo tipo de cambios en el diseño. La encapsulación debe ser crucial para conseguir un buen resultado. Cabe destacar también la importancia de reciclar todos los recursos que tenemos disponibles a nuestro alcance para tomar las decisiones adecuadas a la hora de desarrollarlo, ya sea en forma de ideas captadas de otros proyectos, patrones de diseño, plantillas, herramientas propias del lenguaje, o algún método ya implementado en otra plataforma.

En la etapa de implementación he tenido acceso a recursos y herramientas profesionales (como los entornos de desarrollo) así como una primera toma de contacto con el lenguaje orientado a objetos Java. Al caer toda la responsabilidad en el hecho de tener que realizar el proyecto en un lenguaje y herramientas desconocidas para mi, y además tener que hacerlo en un tiempo limitado de seis meses supuso un gran reto y a la vez una gran responsabilidad, llegando incluso al principio a pensar que no sería capaz de poder llegar a tiempo con la entrega, aunque al final incluso hubo tiempo de sobra para repasar a fondo toda la aplicación, someterla a una gran batería de pruebas y depurar algunos aspectos de código con el único fin de entregar el trabajo lo más perfecto posible en los plazos fijados.

Otro de los aspectos que me han parecido más interesantes ha sido el hecho de que el propio programa a desarrollar debía interactuar con otros servicios ajenos al mismo mediante el envío de mensajes y paquetes mediante la red interna, lo cual lo hace mucho más interesante que si de un sistema cerrado e incomunicado se tratase. Además la herramienta resultante ha cubierto muchos vacíos en los procesos de trabajo y la metodología usada, ya que el departamento tenía poco coordinadas las pruebas, la información estaba dispersa, confusa y en ocasiones incompleta. Gracias a MOPROM ahora el departamento disfruta de una herramienta que integra gran parte del conocimiento de las tecnologías y servicios que rodean al departamento, le proporciona

información directa y completa de cada herramienta, y además sirve de plataforma de consultas a los servicios para comprobarlos o introducir cambios en los sistemas.

Gracias al trabajo realizado en el proyecto he tomado conciencia de lo esencial que es tener una buena documentación con la que apoyar el software. A nivel de código ocurre lo mismo, ya que hay que entregar un código lo suficientemente claro, legible y trazable incluso para una persona que ni siquiera sabe de qué trata la aplicación. Eso facilitará la tarea del propio programador, así como de futuros programadores que deban trabajar en posibles mejoras o cambios en el programa, o incluso reutilizar partes de código para otros proyectos.

También ha resultado muy interesante integrar tecnologías de terceros en mi aplicación, como son los analizadores sintácticos para los ficheros XML (parsers SAX y JDOM), la herramienta de LOG (Log4Java) y el generador de documentación de API's en formato HTML (JavaDoc).

Por último destacar la importante toma de contacto que ha supuesto este proyecto en cuanto a la redacción de documentación técnica, que me ha hecho ser consciente de la importancia de sintetizar las ideas a la hora de desarrollar este tipo de documentación.

Como conclusión final solo queda decir que ha sido muy satisfactorio ver crecer el proyecto desde una simple idea, hasta convertirse en algo real y tangible que cumple todos los requisitos definidos a sus comienzos y adquiriendo por el camino gran cantidad de conocimientos sobre las tecnologías, pero sobre todo en cuanto al análisis, abstracción y síntesis de ideas.

10. DEFINICIONES, ABREVIATURAS Y ACRÓNIMOS

Adjunto	Archivo que se envía junto a un mensaje de correo electrónico o una consulta.
API	Application Programming Interface. Interfaz de programación de aplicaciones, es el conjunto de funciones y métodos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
Aplicación	Tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajo.
Arquitectura	Estructura lógica y física de los componentes de un computador.
Base de datos	Conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.
BATCH o BAT	Archivo de procesamiento por lotes que contiene un conjunto de comandos DOS. Se encarga de ejecutar el programa en la máquina virtual.
Bean	Un Bean es un componente software que tiene la particularidad de ser reutilizable y así evitar programar los distintos componentes uno a uno.
Clase	Declaración o abstracción de un objeto cuando se programa según el paradigma de orientación a objetos.
Componente	Los componente de Software son todo aquel recurso desarrollado para un fin concreto y que puede formar solo o junto con otro/s, un entorno funcional requerido por cualquier proceso predefinido. Son independientes entre ellos, y tienen su propia estructura e implementación.
Consulta	Petición o acceso a una base de datos o sistema informático.
Diagrama de Gantt	Gráfica cuyo objetivo es mostrar el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado

Directorio o carpeta	Lista de los archivos, ficheros o programas almacenados en la memoria de un ordenador.
DOC	Formato de archivo de texto de Microsoft Word.
DOS	Disk Operating System. Sistema operativo de disco es una familia de sistemas operativos para PC.
Eclipse	Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma.
Editor	Programa que permite redactar, corregir, archivar, etc., textos registrados en ficheros de símbolos.
E-mail	Correo electrónico.
Entorno	Conjunto de condiciones extrínsecas que necesita un sistema informático para funcionar, como el tipo de programación, de proceso, las características de las máquinas que lo componen, etc.
Fichero o archivo	Espacio que se reserva en el dispositivo de memoria de un computador para almacenar porciones de información que tienen la misma estructura y que pueden manejarse mediante una instrucción única.
Funcionalidad	Relativo a las funciones que es capaz de realizar el programa.
GET	Método de peticiones HTTP.
Hardware	Conjunto de los componentes que integran la parte material de una computadora.
Hilo	Forma de dividir un programa en múltiples tareas de manera que se ejecuten concurrentemente.
HTML	HyperText Markup Language. Lenguaje de Marcas de Hipertexto, es el lenguaje de marcado predominante para la construcción de páginas Web.
Informe	Texto a través del cual se da cuenta de los avances realizados en un proyecto en particular. En este caso en el informe se informa del resultado de las pruebas a los distintos sistemas.

INI	Fichero de configuración. Contiene parámetros con información útil para el programa.
Instancia	Una instancia se produce con la creación de un objeto perteneciente a una clase (instanciar una clase), que hereda entonces sus atributos, propiedades y métodos para ser usados dentro de un programa, ya sea como contenedores de datos o como partes funcionales del programa al contener en su interior funcionalidades de tratamiento de datos y procesamiento de la información que ha sido programada con anterioridad en la clase a la que pertenece.
Interfaz	Medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y engloba todos los puntos de contacto entre el usuario y el equipo, normalmente suelen ser fáciles de entender y fáciles de accionar.
Internet	Red informática mundial, descentralizada, formada por la conexión directa entre computadoras u ordenadores mediante un protocolo especial de comunicación.
Internet Explorer	Navegador Web desarrollado por Microsoft.
IP	Inglés Internet. Protocolo de Internet no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.
IVA	Impuesto sobre el valor añadido.
JAR	Java Archive. Es un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java. Contiene los archivos ".class" necesarios para su ejecución por la máquina virtual.
Java	Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems.
JAVADOC	JavaDoc es una utilidad de Sun Microsystems para la generación de documentación de APIs en formato HTML a partir de código fuente Java.
JDOM	Java Document Object Model. Modelo en objetos para la representación de documentos en Java. JDOM es una biblioteca de código abierto para manipulaciones de datos XML optimizados para Java.

Lenguaje de programación	Idioma artificial diseñado para expresar operaciones que pueden ser llevadas a cabo por máquinas como las computadoras.
LOG	Registro de eventos del programa. Se almacenan los fallos del programa junto con una descripción.
Máquina Virtual	Una Máquina virtual Java (en inglés Java Virtual Machine, JVM) es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.
MOPROM	Módulo de Pruebas de Operaciones Multicanales
Nodo	Cada uno de los elementos de una lista enlazada.
Parámetro	Variable que puede ser recibida por una subrutina o método.
Parser	Analizador sintáctico.
PC	Personal Computer. Ordenador personal.
Plantilla	Es un medio que permite definir un diseño o esquema estandarizado.
Portapapeles	Herramienta del Sistema Operativo que permite almacenar temporalmente información de cualquier tipo.
POST	Método de peticiones HTTP.
Protocolo	Conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red.

Requisito	Circunstancia o condición necesaria para algo. Un requisito funcional define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. Un requisito no funcional es, en la ingeniería de sistemas y la ingeniería de software, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales.
RRHH	Recursos Humanos.
RSA	Rational Software Architect. Entorno de programación desarrollado por la compañía IBM y basado en el IDE Eclipse.
SAX	Simple API for Java. Es un parser de acceso serial para XML. SAX proporciona mecanismos para leer ficheros XML.
Servidor	Computadora que, formando parte de una red, provee servicios a otras computadoras denominadas clientes.
Sistema	Conjunto de componentes que relacionados entre sí ordenadamente contribuyen a determinado objetivo.
SOAP	Simple Object Access Protocol. Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.
Software	Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.
SplashWindow	Ventana destinada a mostrar un rótulo en el arranque del sistema.
String	Cadena de caracteres.
Test	Prueba que se lanza contra los sistemas.
Timeout	Cuenta atrás.
Trama	Paquete de datos.

TXT	Archivo de texto sin formato.
UML	Unified Modeling Language. Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software.
URL	Uniform Resource Locator. Un localizador uniforme de recursos, es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, para su localización.
Usuario	Es la persona a la que va destinada el producto software una vez que ha superado las fases de desarrollo correspondientes.
WebService	Un servicio web (en inglés, Web service) es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.
XML	Extensible Markup Language. Metalenguaje extensible de etiquetas. Las pruebas son cargadas en este formato. Contienen todos los datos relativos a la pruebas.
XSD	XML Schema Definition. Es un lenguaje de esquema basado en la gramática. Sirve para validar ficheros XML mediante su propia sintaxis. Las pruebas en formato XML son validadas contra la plantilla XSD.

11. BIBLIOGRAFÍA

Jacobson, I., Booch, G. y Rumbaugh J. (2000). *El proceso unificado de desarrollo de software.* Madrid: Addison Wesley.

W3C. (2008). *Documentación XML.* En http://www.w3.org/XML/

Horstmann, C. S., Cornell, G. (2007). *Core Java* (8ª edición). Santa Clara: Prentice Hall.

Gould, S. (2000). *Develop n-tier applications using J2EE.* En http://www.javaworld.com/jw-12-2000/jw-1201-weblogic.html

Fowler, M. (2003). *UML Distilled* (3ª edición). Boston: Pearson Education Inc.

Documentación log4java. (2008). Wikipedia, la enciclopedia libre. En http://es.wikipedia.org/wiki/Log4j

Ingeniería de requisitos. (2008). Wikipedia, la enciclopedia libre. En http://es.wikipedia.org/wiki/Ingeniería_de_requisitos

Mukhar, K. y Zelenak, C. (2006). *Beginning Java EE 5: From Novice to Professional.* New York: Apress.

Jason Hunter, Brett McLaughlin. (2002). *Documentación JDOM.* En http://www.jdom.org/downloads/docs.html

Sun Microsystems, Inc. (2003). *Documentación SAX.* En http://java.sun.com/j2se/1.4.2/docs/api/javax/xml/parsers/SAXParser.html

Lenguaje XML. (2008). Wikipedia, la enciclopedia libre. En http://en.wikipedia.org/wiki/XML

Diagrama de clases. (2008). Wikipedia, la enciclopedia libre. En http://es.wikipedia.org/wiki/Diagrama_de_clases

Apache Software Foundation. (2008). *Documentación log4java.* En http://wiki.apache.org/logging-log4j/

Apache Software Foundation. (2008). *Commons Logging.* En http://wiki.apache.org/commons/Logging

12. ANEXOS

12.1 JAVADOC

Junto con la documentación entregada, se ha generado el JavaDoc de la aplicación en formato HTML. JavaDoc es una utilidad de Sun Microsystems para la generación de documentación de APIs en formato HTML a partir de código fuente Java. Esto ha sido posible gracias a una serie de etiquetas o "tags" que se han introducido como comentarios en todas las clases y procedimientos durante la fase de programación, de modo que el código queda preparado para que JavaDoc realice el proceso de generación de los ficheros HTML, que contendrán las descripciones pertinentes de cada función, su funcionamiento, y sus parámetros de entrada / salida.

En la siguiente figura podemos ver el índice del JavaDoc de MOPROM en el que se muestran accesos directos a los paquetes, y todas las clases. Pinchando sobre los hipervínculos podremos navegar hasta la clase que nos interese y revisar por ejemplo los parámetros de entrada y salida de un procedimiento.

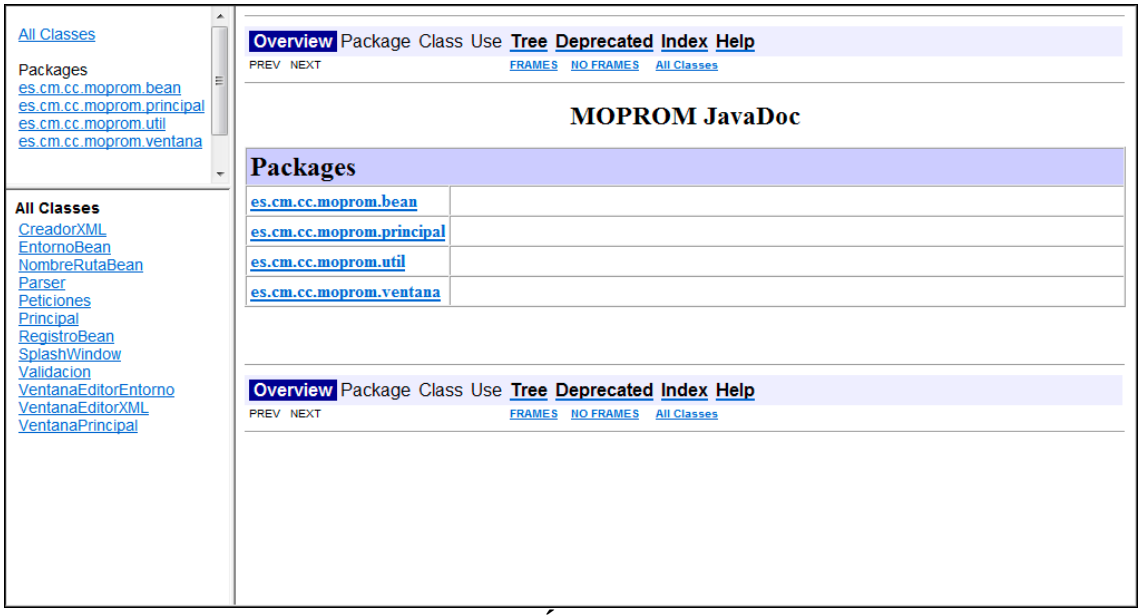
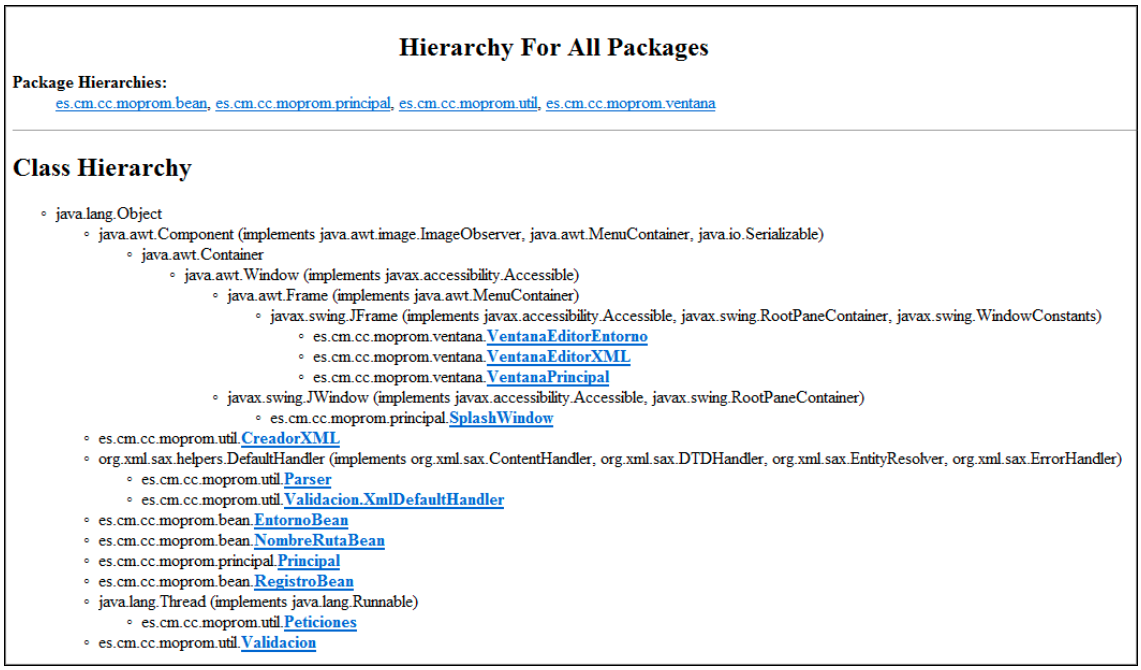


Figura 37: Índice JavaDoc.

En la vista en árbol (tree view) podemos ver la jerarquía completa de todos los paquetes:



Hierarchy For All Packages

Package Hierarchies:
es.cm.cc.moprom.bean, es.cm.cc.moprom.principal, es.cm.cc.moprom.util, es.cm.cc.moprom.ventana

Class Hierarchy

- java.lang.Object
  - java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
    - java.awt.Container
      - java.awt.Window (implements javax.accessibility.Accessible)
        - java.awt.Frame (implements java.awt.MenuContainer)
          - javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
            - es.cm.cc.moprom.ventana.**VentanaEditorEntorno**
            - es.cm.cc.moprom.ventana.**VentanaEditorXML**
            - es.cm.cc.moprom.ventana.**VentanaPrincipal**
        - javax.swing.JWindow (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer)
          - es.cm.cc.moprom.principal.**SplashWindow**
  - es.cm.cc.moprom.util.**CreadorXML**
  - org.xml.sax.helpers.DefaultHandler (implements org.xml.sax.ContentHandler, org.xml.sax.DTDHandler, org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler)
    - es.cm.cc.moprom.util.**Parser**
    - es.cm.cc.moprom.util.**Validacion.XmlDefaultHandler**
  - es.cm.cc.moprom.bean.**EntornoBean**
  - es.cm.cc.moprom.bean.**NombreRutaBean**
  - es.cm.cc.moprom.principal.**Principal**
  - es.cm.cc.moprom.bean.**RegistroBean**
  - java.lang.Thread (implements java.lang.Runnable)
    - es.cm.cc.moprom.util.**Peticiones**
  - es.cm.cc.moprom.util.**Validacion**

Figura 38: Jerarquía de paquetes JavaDoc.

Como se puede apreciar todas las pantallas de navegación contienen hipervínculos a todas las clases lo cual hace que la trazabilidad del código en caso de querer hacer una revisión sea muy rápida.

En la siguiente figura podemos ver como se muestra una consulta a los métodos de una clase de ejemplo.

Method Summary	
java.lang.String	componerURL(java.lang.String trama, java.lang.String url) Metodo que dado una trama que contiene parametros y una url, completa la url con los parametros para poder realizar una peticion GET.
java.lang.String	enviarPeticionGET(java.lang.String url, java.lang.String trama, int timeout) Metodo que realiza peticiones de tipo GET.
java.lang.String	enviarPeticionPOST(java.lang.String direccionURL, java.lang.String trama, int timeout) Metodo que realiza peticiones de tipo POST.
java.lang.String	enviarPeticionSOAP(java.lang.String direccionURL, java.lang.String trama, int timeout) Metodo que realiza peticiones de tipo SOAP.
java.lang.String	enviarPeticionSocket(java.lang.String direccionURL, java.lang.String trama, int timeout) Metodo que realiza peticiones de tipo SOCKET.
java.lang.String	enviarPeticionURL(java.lang.String direccionURL, java.lang.String trama, int timeout) Metodo que realiza peticiones de tipo URL.
void	lanzarPrueba(VentanaPrincipal ventanaPrincipal, java.lang.String url, java.lang.String trama, java.lang.String viaAcceso, int tiempo) Metodo que se encarga dada una peticion de prueba de lanzar el metodo de llamada mas apropiado Dada una ventana principal en la que plasmar el resultado, una url, una trama y una via de acceso hace la llamada correspondiente, y con el retorno (un string con el resultado de la prueba) rellena la ventana principal.
void	run() Metodo principal para lanzar la prueba, se realiza de este modo para poder hacer un hilo.
private java.io.InputStream	string2InputStream(java.lang.String trama) Metodo que simula un inputStream a partir del contenido de un string.

Figura 39: Sumario de métodos JavaDoc.

12.2 Fichero LOG

En la implementación de MOPROM se ha tenido en cuenta que prácticamente ningún programa software es perfecto y que puede darse el caso de que se produzca una ejecución en el programa cuya causa sea difícil de detectar si el programa deja de responder, o no se comunica con nosotros. De este modo se ha implementado mediante "log for java (log4j)" un sistema de fichero LOG que permite que ante un error de ejecución inesperada, se genere un fichero de texto que contenga el tipo de error, la clase que lo ha generado y una breve descripción de qué ha ocurrido, de forma que dispongamos de algo de ayuda en lo que se refiere a la trazabilidad del error para poder solucionarlo más fácilmente. Esto se realiza capturando las excepciones, de forma que ante cualquier llamada fallida el error quedará capturado antes de que se propague sin control.

Como se puede apreciar debajo en el fichero LOG de ejemplo, junto con la hora del suceso, aparece el tipo de error, la clase en la que se ha producido, y el método correspondiente a la clase.

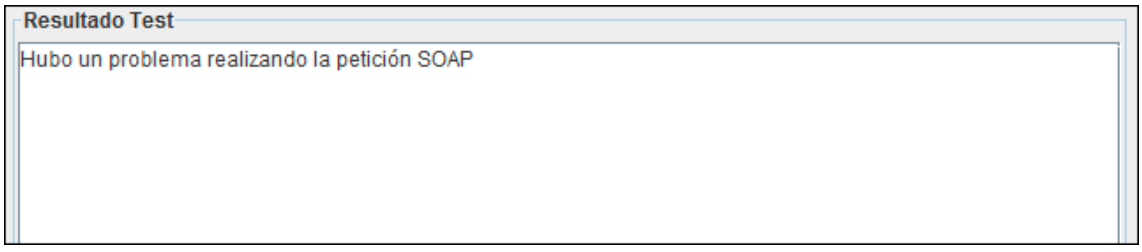
```
2009-08-13 20:22:35,937 [ERROR][Peticiones][enviarPeticionGET] - Hubo un problema realizando la petición GET
2009-08-13 20:22:35,937 [ERROR][Peticiones][enviarPeticionGET] - java.net.UnknownHostException: ssoro400
2009-11-23 11:17:03,833 [ERROR][Peticiones][enviarPeticionSOAP] - Hubo un problema realizando la petición SOAP
2009-11-23 11:17:03,833 [ERROR][Peticiones][enviarPeticionSOAP] - java.io.IOException: Server returned HTTP response code:
500 for URL: http://10.64.108.12:8180/axis2/services/Asterisk
```

Figura 40: Ejemplo fichero "moprom.log".

Gracias a la implementación del fichero LOG y la captura de las excepciones que pudieran darse se ha conseguido un software muy estable.

Debido a la naturaleza del programa (lanzar llamadas a otros programas) en el caso de darse un error en la respuesta del servidor, ésta es capturada y se mostrará tanto en el cuadro reservado a la respuesta de la consulta, como en el fichero LOG.

Podemos ver un ejemplo de cómo se muestra el fallo en la ventana de resultados y en el fichero LOG. En éste nos informa que ha vencido el tiempo de espera.




```
2009-11-14 20:04:12,259 [ERROR][Peticiones][enviarPeticionSOAP] - Hubo un problema realizando la petición SOAP
2009-11-14 20:04:12,263 [ERROR][Peticiones][enviarPeticionSOAP] - java.net.ConnectException: Connection timed out:
connect
```

Figura 41: Ejemplo fallo y registro en LOG.

12.3 Manual de usuario

A continuación se adjunta el manual de usuario redactado de la aplicación para CajaMadrid siguiendo la plantilla propia del departamento de desarrollo.

En él se incluyen descripciones sobre los procesos y usos del programa, su funcionamiento de cara al usuario así como capturas de pantalla y recuadros que indican su funcionalidad, ofreciendo ejemplos y explicaciones detalladas.

MANUAL DE USUARIO. MÓDULO DE PRUEBAS DE OPERACIONES MULTICANALES (MOPROM). USUARIO

Proyecto: Módulo de Pruebas Operaciones Multicanales
Equipo: Canales Complementarios – Banca Telefónica
Fecha: 15 de diciembre de 2008
Estado: Abierto

1 DESCRIPCIÓN DEL DOCUMENTO 97

1.1 Referencias 97

1.2 Definiciones y acrónimos 97

1.3 Control de modificaciones sobre el documento 97

2 INTRODUCCIÓN 98

2.1 Descripción general del sistema 98

2.2 Características de los usuarios 99

3 NAVEGACIÓN POR LA APLICACIÓN 100

3.1 Visión general y Ventanas 100
3.1.1 Interfaz gráfico de la aplicación. 100
3.1.1.1 Selección de prueba. 101
3.1.1.2 Carga de trama o parámetros. 102
3.1.1.3 Panel de lanzamiento de prueba. 102
3.1.1.4 Resultado de la prueba. 103
3.1.1.5 Panel de generación de informes y reinicio/salida. 103

3.2 Fundamentos 107
3.2.1 Iniciar la aplicación 107
3.2.2 Lanzar una prueba 107
3.2.3 Interpretación de errores y fichero LOG 110

1. Descripción del documento

El objetivo del manual es la comprensión, funcionamiento y entorno necesario para que el usuario pueda utilizar el módulo de pruebas de manera autónoma. Para ello, se pretende realizar una visión general de las funcionalidades contenidas en la aplicación así como la descripción de los componentes utilizados por la misma.

1.1 Referencias

Título	Autor	Fecha	Editor	Fuentes
Manual de usuario Telemarketing ABC.	CajaMadrid	Abril 2003		

1.2 Definiciones y acrónimos

Término	Definición
MOPROM	Módulo de pruebas de operaciones multicanales.
XML	Metalenguaje extensible de etiquetas. Las pruebas son cargadas en este formato. Contienen todos los datos relativos a las pruebas.
JAR	Java Archive. Es un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java. Contiene los archivos .class necesarios para su ejecución por la máquina virtual.
INI	Fichero de configuración. Contiene parámetros a usar por el programa.
BAT	Archivo de procesamiento por lotes que contienen un conjunto de comandos DOS. Se encarga de ejecutar el programa en la máquina virtual.
XSD	XSD (XML Schema Definition). Es un lenguaje de esquema basado en la gramática. Sirve para validar ficheros XML mediante su propia sintaxis. Las pruebas en formato .XML son validadas contra la plantilla .XSD.
TXT	Archivo de texto sin formato.
LOG	Registro de eventos del programa. Se almacenan los fallos del programa junto con una descripción.

1.3 Control de modificaciones sobre el documento

Fecha	Versión	Capítulo Afectado	Observaciones	Autor Modificación

2. Introducción

2.1 Descripción general del sistema

El objetivo de la aplicación es unificar las pruebas del departamento en un solo aplicativo. Consistirá en una aplicación capaz de lanzar pruebas estándar a las diferentes aplicaciones del departamento y utilizando varios métodos de acceso. Estas pruebas estarán definidas mediante un fichero estándar XML basado en una plantilla XSD. Cada prueba debe ser validada contra esa plantilla para entrar al sistema. Las pruebas se componen de diversos parámetros (nombre de la aplicación, método de acceso, URL, entorno, comentarios...). Una vez cargadas las pruebas en la ventana principal el programa estará listo para ejecutar dichas pruebas.

Las funciones de MOPROM (Modulo de Pruebas de Aplicaciones Multicanales) son:

- Gestión de las pruebas: Unificará todas las pruebas de las aplicaciones presentes en el departamento. Cuenta además con un editor de pruebas, permitiendo al usuario crear pruebas nuevas o modificar las existentes.
- Realización de informes para las pruebas: proporcionará dos tipos de informes, un informe completo con todos los datos relativos a la prueba que se copia al portapapeles, y la posibilidad de guardar el resultado de la prueba en un fichero de texto.
- Editor completo de pruebas: Permite la edición y creación de la pruebas mediante la plantilla XSD definida. El editor permite la modificación de pruebas existentes o pueden crearse desde cero, garantizando que las pruebas generadas son totalmente reconocidas por la aplicación (ya que el programa podría reconocer pruebas creadas pruebas manualmente siempre que se ciñan a la plantilla)
- Paso de tramas y parámetros: Las pruebas pueden contener en el mensaje enviado tramas o parámetros incluidos (depende del tipo de prueba). Estas tramas/parámetros pueden ser precargados antes de realizar la prueba o escritos a mano por el usuario. Existe la posibilidad de guardas los cambios realizados o incluso, crear nuevos archivos de tramas y parámetros.
- Entornos: cada prueba contemplará la posibilidad de contener distintos entornos para las aplicaciones, esto es, que podrá contener entornos de “desarrollo”, “tránsito”, “real” y “otros” con su correspondiente URL y puerto de acceso para cada entorno.
- Timeout: la ventana principal tiene definido por defecto un timeout en segundos, pero puede modificarse antes de la realización de cada prueba. El timeout define el tiempo de espera máximo de la aplicación tras el lanzamiento de una prueba.
- Cronómetro: tras la realización de cada prueba se mostrará en la pantalla el tiempo invertido hasta obtener la respuesta del servidor.
- Fichero de configuración: el programa carga un fichero de inicio (.ini) con parámetros configurables que afectan al funcionamiento del programa.

2.2 Características de los usuarios

El público objetivo de este manual es el personal del departamento de Banca Telefónica.

Se requiere al menos un equipo con Windows XP y una versión de máquina virtual Java 1.5 o superior.

3. Navegación por la aplicación

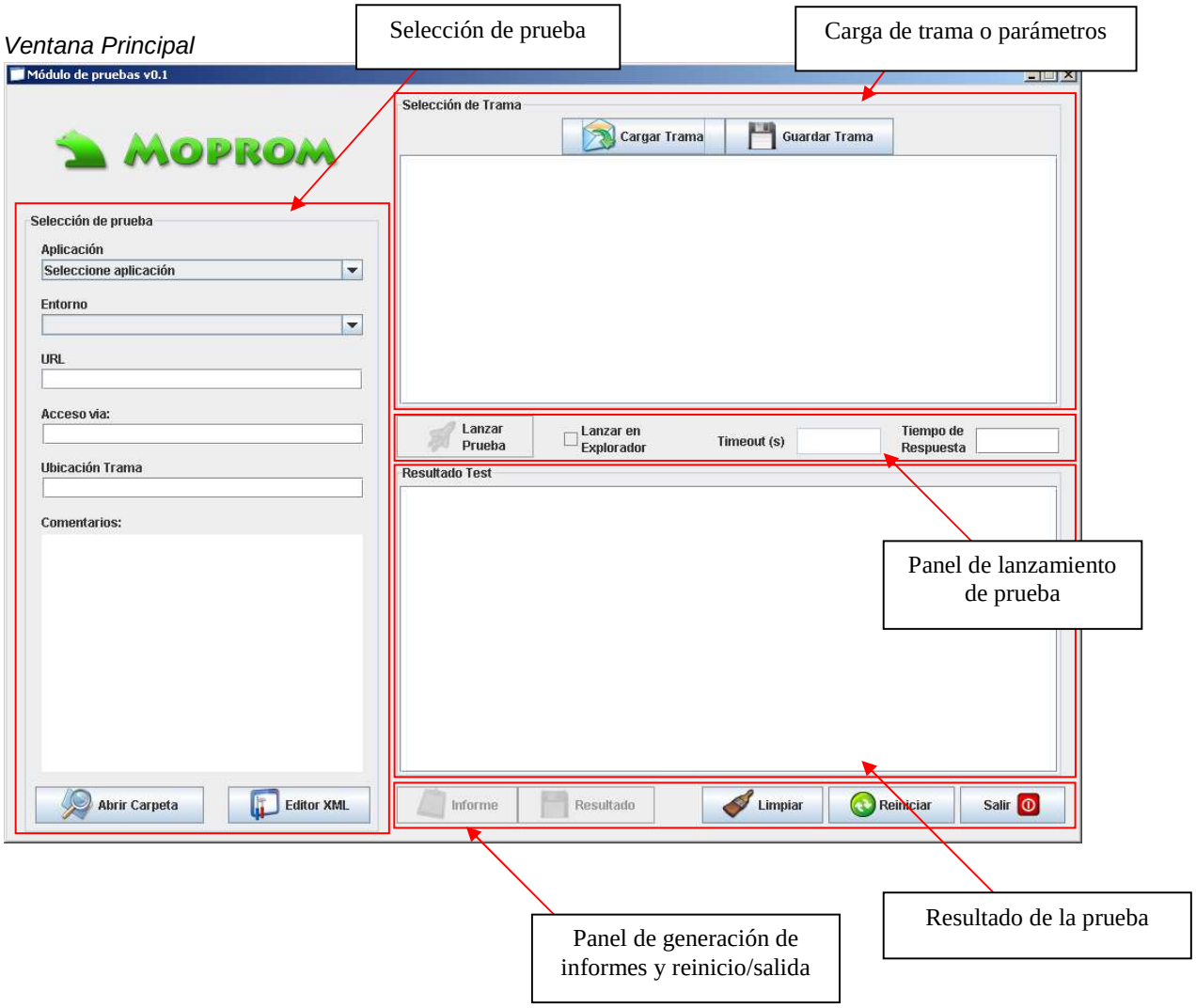
3.1 Visión general y Ventanas

En este capítulo se detallan los conceptos básicos para facilitar la comprensión de la aplicación. Se explican las partes más importantes de la interfaz del usuario, así como una descripción de sus ventanas.

3.1.1 Interfaz gráfico de la aplicación.

A continuación se describe el interfaz gráfico

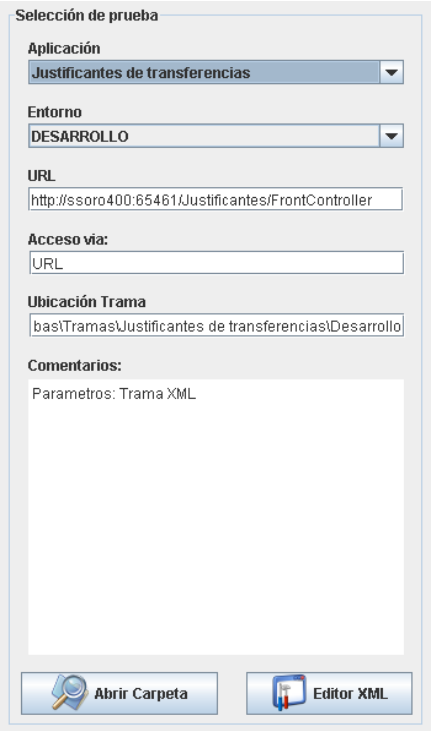
Ventana Principal



La ventana principal cuenta con distintas áreas divididas según se puede ver en la figura anterior.

3.1.1.1 Selección de prueba:

El primer panel es el de selección de prueba. En él podremos elegir entre una de las pruebas que aparecen en el listado desplegable. Una vez seleccionada una prueba se rellenarán automáticamente el resto de los elementos del panel. El contenido de todos los elementos excepto URL dependen del entorno que haya seleccionado en el segundo desplegable. Por defecto aparece seleccionado el primer entorno.



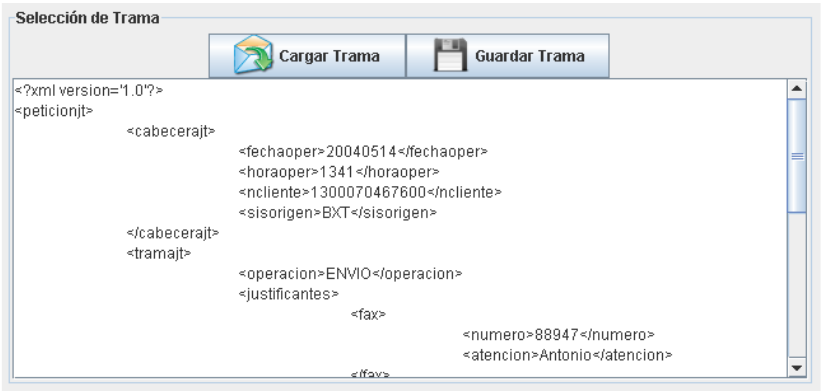
Se puede apreciar cómo al seleccionar una aplicación, el resto de componentes se han rellenado. Solamente el acceso vía (en este caso URL) se mantiene para todos los entornos, ya que cada aplicación solo puede funcionar mediante una vía concreta.

Tras haber seleccionado la prueba ya tendríamos el programa listo para lanzar la petición, no obstante, hay pruebas que requieren la carga de determinadas tramas o parámetros antes de realizar la petición.

Las pruebas tienen definidas, por cada entorno, una ruta que indica dónde están ubicadas las tramas o parámetros que corresponden al tipo de prueba, aunque este parámetro es opcional ya que pueden buscarse manualmente, aunque así se facilita la tarea.

3.1.1.2 Carga de trama o parámetros:

El segundo panel es el de la carga de tramas o parámetros. Contiene dos botones y una caja de texto. La función del primer botón es la de la carga de la trama. Una vez pulsado aparece una ventana de selección de ficheros. Si tenemos definida una ruta de trama en la prueba, esta ventana mostrará automáticamente los archivos contenidos en ese directorio, en otro caso, se abrirá en el directorio por defecto de tramas que esté predefinido para la aplicación.
Pulsando el segundo botón podremos guardar cambios que hayamos realizado sobre la trama cargada, o bien si hemos escrito en la caja estando vacía, estaremos generando un archivo de pruebas nuevo. Solo habría que almacenarlo en el directorio de tramas correspondiente y aparecerá al pulsar el botón de "cargar trama".
Todo lo contenido en el panel de texto es totalmente editable, esto es así porque determinadas tramas que serán cargadas, contienen información incompleta, o necesitan ser editables para cambiar valores (por ej. Números de usuario, de tarjeta... y otros parámetros).
Al lanzar la prueba solo se tendrá en cuenta el texto cargado en el panel, no lo que contenía el archivo físico del que fueron cargados los datos.

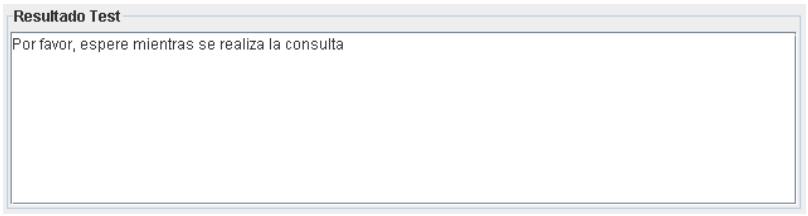


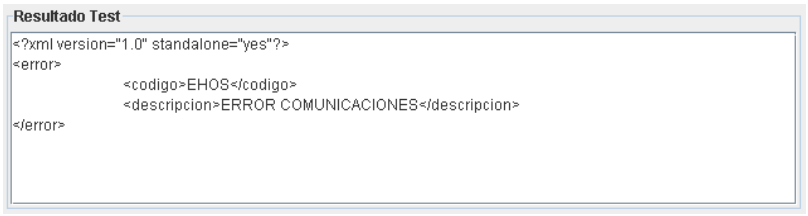
3.1.1.3 Panel de lanzamiento de prueba:

Este panel contiene el botón clave de la aplicación que es el de lanzamiento de prueba. Contiene además un "checkbox" para que las peticiones mediante el método GET sean lanzadas mediante el programa Internet Explorer. En la caja "timeout" aparece un valor en segundos predefinido en el fichero de configuración, pero puede ser cambiado por el usuario antes de realizar cada prueba.

3.1.1.4 Resultado de la prueba

Una vez lanzada la prueba, el panel de resultado de prueba muestra el mensaje: “Por favor, espere mientras se realiza la consulta”. Y una vez la respuesta es recibida, se muestra en el panel el resultado de la consulta (el resultado depende de cada aplicación).





3.1.1.5 Panel de generación de informes y reinicio/salida

Este panel contiene los botones para almacenar el resultado de la prueba, reiniciar y salir del programa.

El botón informe genera un informe completo en el portapapeles para poder ser pegado en un correo electrónico o un fichero Word por ejemplo. El botón Resultado almacena únicamente el resultado del test en un fichero (.txt)
Aquí se muestra un ejemplo de informe generado:

Informe de prueba generado el dia 16-12-2008, a las 09:08 horas.

Nombre de aplicación : LOGIN

Entorno : TRANSITO

URL : http://ssoro401:65461/LOGIN/FrontController

Acceso vía : URL

Ubicación Trama : J:\Banca Telefónica\Alberto\Modulo de Pruebas\Tramas\LOGIN\TRANSITO

Comentario : Parametros: Trama XML

Trama o parámetros envíados :

VACIO

Tiempo de respuesta de la aplicación : 00:00:063

Respuesta de la aplicación :

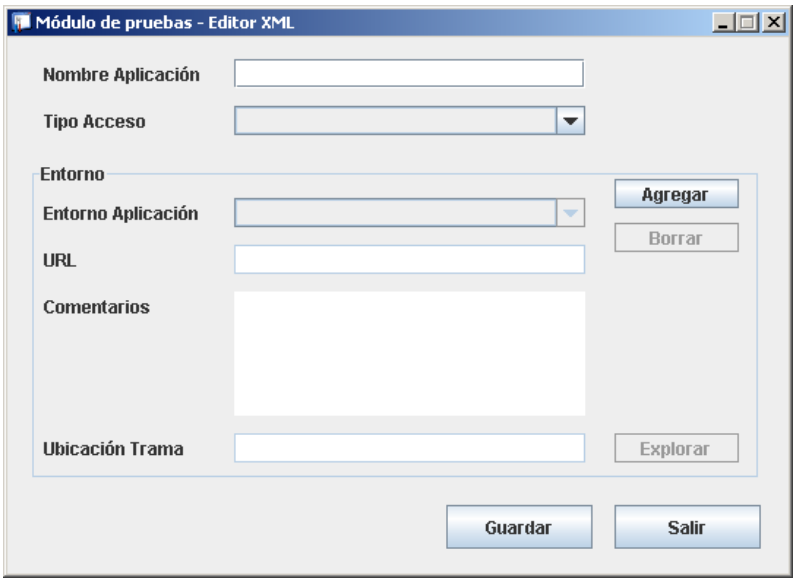
```
<?xml version="1.0" standalone="yes"?>
<error>
        <codigo>EHOS</codigo>
        <descripcion>ERROR COMUNICACIONES</descripcion>
</error>
```

El botón limpiar se encarga de vaciar la caja de resultado y el cronómetro una vez realizada la prueba.

El botón reiniciar limpia todos los campos de la ventana y reinicializa la aplicación, recargando y validando los ficheros de pruebas.

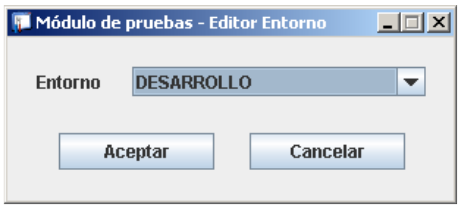
Botón para salir de la aplicación.

Editor XML



El editor de pruebas XML consta de una ventana con campos a rellenar o modificar. Los únicos campos obligatorios son: “nombre aplicación” y “tipo de acceso”, además cada prueba debe estar definida al menos para un entorno. Los campos obligatorios de cada entorno son el tipo de entorno y su URL.

Al pulsar sobre el botón “agregar” dentro de la zona de entorno se abre una pequeña ventana para elegir el tipo de entorno:



Cuando hay más de un entorno dentro de una prueba, el botón borrar se activa, de forma que se pueda borrar ese entorno. Se suprimirá el entorno seleccionado en la lista desplegable.

El botón explorar situado a la derecha de “Ubicación trama” permite elegir la carpeta en la que se encuentran las tramas mediante una ventana de selección de directorio.

No se puede borrar una prueba completa como tal desde el programa. Para ello habrá que acceder al directorio donde están almacenadas las pruebas y borrar manualmente el fichero. Esto puede hacerse pulsando el botón “Abrir Carpeta” en el panel de Selección de Prueba de la ventana Principal. Se abrirá una ventana del explorador en la que nos encontraremos el directorio por defecto donde están almacenadas las pruebas en forma de ficheros con extensión (.xml).
Si se ha hecho algún cambio o borrado el los archivos, habrá que pulsar el botón “Reiniciar” de la ventana principal para que se hagan efectivos los cambios.

3.2 Fundamentos

3.2.1 Iniciar la aplicación

Para iniciar la aplicación haremos doble click sobre el fichero “moprom.jar” (si no se ejecuta correctamente al hacer doble click) habrá que lanzar el fichero Moprom.bat. Este fichero contiene la ruta de ejecución de la máquina virtual (java.exe) y del paquete (moprom.jar) de la aplicación.

La aplicación está contenida en un paquete Java (moprom.jar). Para ejecutarse correctamente requiere que el sistema esté configurado por defecto para utilizar Java versión 1.5 o superior. En algunos equipos, dependiendo de la configuración de las extensiones, al hacer doble click sobre el archivo (.jar) la aplicación se iniciará automáticamente, siempre y cuando la versión por defecto Java sea al menos 1.5. En caso de producir un error o no ejecutarse la aplicación, existe una forma alternativa de ejecutar la aplicación.

La forma alternativa consiste en un fichero (MOPROM.bat) que se encarga de arrancar la versión adecuada de la máquina virtual Java del sistema.

Este fichero contiene las siguientes líneas:

```
@echo off

REM ruta_java contiene la ruta de la maquina virtual java a utilizar

REM ruta_programa contiene la ruta del fichero .jar del programa


Set ruta_java="C:\Archivos de programa\Java\jre1.5.0_11\bin\java.exe"

Set ruta_programa="J:\Banca Telef¢nica\Alberto\MOPROM v1\MOPROM.jar"
```

Este fichero batch se encarga de ejecutar el programa con una versión predefinida de Java en el atributo “ruta_java”. Esta ruta apunta a un directorio local donde se instalan las distintas versiones de la máquina virtual java. Habitualmente en:

“C:\Archivos de programa\Java\j2reX.X.X_XX\bin\java.exe”

Si se elige este método de ejecución del programa, en cada equipo deberá editarse el fichero MOPROM.bat indicando la ruta correspondiente del mismo.

Fichero Config.ini

Para el correcto arranque y funcionamiento del programa, se requiere un fichero de configuración (config.ini) que debe contener 17 parámetros, que incluyen directorios de trabajo, archivos clave (por ejemplo la plantilla .XSD), otros parámetros de configuración y todos los iconos e imágenes del programa.

```
plantillaXSD    = resources\\XML\\plantilla-pruebas.xsd

logoSuperior    = resources\\data\\logo moprom cabecera.png

iconoExit       = resources\\data\\Windows-Turn-Off-24x24.png

iconoLanzar     = resources\\data\\launch-32x32.png

iconoPortapapeles = resources\\data\\clipboard-32x32.png

iconoGuardar    = resources\\data\\Floppy-32x32.png

iconoExplorar   = resources\\data\\folder-explorer-32x32.png

iconoReiniciar  = resources\\data\\Button-Refresh-24x24.png

iconoLimpiar    = resources\\data\\Paintbrush-32x32.png

iconoCargar     = resources\\data\\ei0021-32x32.png

iconoEditar     = resources\\data\\Windows-Tools-32x32.png

iconoAplicacion   = resources\\data\\Windows-Luna-32x32.png

tiempoSplash    = 5000

ficherosXML     = resources\\XML

rutaDefectoTramas = resources\\XML\\Tramas

imagenSplash    = resources\\data\\logo moprom splash.png

default_timeout   = 60
```

3.2.2 Lanzar una prueba

Los pasos a seguir para poder realizar una prueba son:

1) Ejecutar la aplicación: Una vez ejecutada, se cargarán los ficheros de pruebas XML en el programa.
2) Seleccionar una aplicación: En la lista desplegable situada arriba a la izquierda llamada “aplicación” tendremos que seleccionar un elemento. Una vez seleccionado se cargarán los datos relativos a la prueba en el panel de selección de prueba.
3) Seleccionar un entorno: En la lista desplegable entorno seleccionaremos el entorno sobre el que ejecutar la prueba, éste puede ser: desarrollo, tránsito, real y otro. Una vez seleccionado (por defecto se selecciona el primer entorno creado) se cargará la URL asociada a ese entorno y los comentarios del mismo en caso de haberlos.
4) Carga de trama o parámetros: En caso de que la prueba requiera la carga de determinada trama o de un fichero de parámetros, pulsaremos el botón “Cargar trama” y seleccionaremos el fichero a cargar, rellenándose el panel de texto superior con la información contenida en el fichero.
5) Configuración de trama o parámetros: Determinadas tramas o ficheros de parámetros pueden estar almacenados con partes sin rellenar, o datos ficticios que deben ser modificados en cada prueba (por ejemplo un campo con el número o nombre de usuario). Estos campos puede que vengan descritos en el comentario de la prueba, y podrán ser editados en el panel superior antes de lanzar la prueba.
6) Configuración del Timeout: Por defecto el Timeout para las pruebas viene prefijado en el fichero “config.ini”, pero puede ser modificado antes del lanzamiento de cada prueba. Tan solo hay que pulsar sobre el cuadro de texto del timeout y marcar un nuevo valor en segundos.
7) Lanzamiento de prueba: Al pulsar el botón de lanzamiento de prueba el programa recoge todos los datos y ejecuta la consulta. Un mensaje de información aparecerá en el panel de resultado y el cronómetro se iniciará. Una vez concluida la prueba se mostrará el resultado de la consulta y el tiempo empleado. En el caso de aplicaciones que utilicen el acceso por vía GET, podrán ser lanzadas externamente por el navegador Internet Explorer marcando la casilla “Lanzar en Explorador” junto al botón de lanzamiento.
8) Tratamiento del resultado: Una vez la prueba esté concluida, tenemos la posibilidad de generar un informe con todos los datos relativos a la prueba. El informe se generará en el portapapeles de forma que pueda ser pegado en un procesador de textos o en un correo electrónico. También cabe la posibilidad de que el resultado de la prueba puede guardarse en un fichero de texto (.txt)

3.2.3 Interpretación de errores y fichero LOG

Algunos errores propios de las peticiones, bien sean timeout's, errores en la url, etc... serán mostrados en el panel de resultado.

Además todos los errores producidos en el programa serán almacenados en un fichero LOG llamado "log.txt". En él se incluirá una descripción del error y de la excepción que se ha producido.