Sistema de comunicación para mensajería de ficheros (HL7)

Memoria del proyecto

de Ingeniería Técnica en Informática de Gestión

realizado por

Lorenzo Gama Ramos

y dirigido por

Jordi Pons Aróztegui

Escola d'Enginyeria

Sabadell, Junio de 2012

El abajo firmante, **Jordi Pons Aróztegui**, profesor de la

Escuela de Ingeniería de la UAB,

CERTIFICA:

Que el trabajo al cual corresponde la presente memoria

ha sido realizada bajo su dirección

por **Lorenzo Gama Ramos**,

Y para que conste firma la presente.

Sabadell, **Junio** de **2012**

_____

Firmado: Jordi Pons Aróztegui

El abajo firmante, **Daniel Fuster Escriba**,

de Unit4,

CERTIFICA:

Que el trabajo al cual corresponde la presente memoria

ha sido realizada bajo su dirección

por **Lorenzo Gama Ramos**,

Y para que conste firma la presente.

Sabadell, **Junio** de **2012**

_____

Firmado: Daniel Fuster Escriba

HOJA DE RESUMEN – PROYECTO FIN DE CARRERA DE LA ESCUELA DE INGENIERIA

Título del proyecto: Sistema de comunicación para mensajería de ficheros (HL7)	
Autor: Lorenzo Gama Ramos	**Fecha:** Junio 2012
Tutores: Jordi Pons Aróztegui / Daniel Fuster Escriba	
Titulación: Ingeniería Técnica en Informática de Gestión	

Palabras clave:

- Català: Missatgeria, Java, HL7
- Castellano: Mensajería, Java, HL7
- English: Messaging, Java, HL7

Resumen del proyecto

- **Català:** El projecte consisteix en la creació d'un sistema de missatgeria que permeti l'enviament i recepció de fitxers generats pel programari de gestió sanitària UNIT4 Ekon Salus. Les tasques a realitzar són, d'una banda, la portabilitat del sistema actual basat en Visual Basic a Java i, de l'altra, la implementació d'un sistema de control i notificació d'errors, ja siguin generats pel contingut dels fitxers o per la comunicació entre els programes d'enviament i recepció que constitueixen el sistema de missatgeria.
- **Castellano:** El proyecto consiste en la creación de un sistema de mensajería que permita el envío y recepción de ficheros generados por el software de gestión sanitaria UNIT4 Ekon Salus. Las tareas a realizar son, por un lado, la portabilidad del sistema actual basado en Visual Basic a Java y, por el otro, la implementación de un sistema de control y notificación de errores, ya sean generados por el contenido de los ficheros o por la comunicación entre los programas de envío y recepción que constituyen el sistema de mensajería.
- **English:** The project involves the creation of a messaging system that allows sending and receiving files generated by health management software UNIT4 Ekon Salus. The tasks are on the one hand, the portability of the current system based on VisualBasic to Java and, on the other hand, the implementation of amonitoring and reporting of errors, whether generated by thecontent of the files or communication between sending and receiving programs that make up the messaging system.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 OBJETIVOS

UNIT4 Ekon Salus es el ERP o software para sanidad que se ofrece a las empresas y organismos del sector sanitario asistencial por parte de UNIT4, constituyéndose como una avanzada solución informática de gestión integrada para facilitar al máximo el tratamiento de su información.

El ERP o software de gestión sanitaria UNIT4 Ekon Salus se puede personalizar hasta el más mínimo detalle sin tener que programar, y las evoluciones de las personalizaciones son instantáneas. Además, está basado en estándares de mercado tanto tecnológicos (HL7, XML, DICOM...) como asistenciales (IDC9, BOT,....).

Dentro del software UNIT4 Ekon Salus, una de las funciones basadas en el estándar HL7 es la creación y envío de peticiones entre departamentos y/o centros sanitarios. Estas peticiones contienen, en la mayoría de los casos, información médica sensible acerca de los pacientes.

HL7 (Health Level Seven) es un conjunto de estándares para el intercambio electrónico de información clínica, desarrollados por la organización HL7 International. Dicha organización desarrolla especificaciones que, posteriormente, son usadas por los implementadores de software para solucionar problemas de integración entre sistemas de información heterogéneos. HL7 proporciona estándares que mejoran la atención en salud, optimizan el flujo de trabajo, reducen la ambigüedad y mejoran la transferencia de conocimientos entre todos los interesados, incluidos los prestadores de servicios de salud, organismos gubernamentales, la comunidad de proveedores y los pacientes.

Para poder enviar esa información, UNIT4 Ekon Salus crea un fichero codificado cumpliendo el formato definido en el estándar HL7. El contenido del fichero está formado por los campos obligatorios más los campos complementarios que variaran dependiendo de la petición.

Dentro del proceso de envío y recepción del fichero generado a través de una petición, se utilizan dos aplicaciones que cumplen la función de emisor y receptor. El proceso de envío y recepción se realiza de manera transparente para el usuario final, no teniendo éste más que realizar o atender una petición.

El objetivo del proyecto es, en este caso, el proceso de estudio, revisión y modificación del sistema que permite el envío y recepción de ficheros, adaptándolo a las necesidades de los clientes en particular y, en general, a las actuales soluciones de software residentes en el mercado.

1.2 CONVENIO DE COLABORACIÓN UNIT4-UAB

El convenio existente entre la Universidad Autónoma de Barcelona y UNIT4 Ibérica permite a los alumnos la realización de proyectos en los diferentes departamentos de la empresa, siendo el departamento de I+D donde se ha realizado el actual proyecto.

UNIT4 Ibérica es la filial española de la multinacional de origen holandés UNIT4. Es una compañía de software de gestión de empresas que crea, ofrece y soporta software y servicios adaptables con el fin de ayudar a las organizaciones dinámicas a abrazar el cambio de manera simple, rápida y rentable.

En la actualidad, la compañía cuenta con filiales y oficinas en 24 países distribuidos en Europa, Norteamérica, Asia-Pacífico y África, y con actividad comercial en otros países.

El producto principal que la empresa ofrece a sus clientes es UNIT4 ekon, un avanzado software especialmente diseñado para adaptarse a las necesidades específicas de cada una de las empresas, sin renunciar a las ventajas del software estándar y con una visión de total integración de procesos.

Concebido como una solución ágil, orientada a resultados, transparente para los usuarios y sumamente flexible, UNIT4 ekon cuenta con internet como parte natural; es por ese motivo que la garantía de conexión global facilita que los actores de sus procesos de negocio (empleados, clientes y proveedores) participen en los circuitos de información que el cliente determine, desde cualquier lugar y con cualquier dispositivo.

UNIT4 concede una especial atención al sector sanitario con su software para Sanidad (HIS). En estrecha colaboración con instituciones y profesionales tanto del ámbito público como privado, UNIT4 ekon salus ofrece avanzadas soluciones de software de gestión integrada para esta actividad que:

- Se adapta con facilidad a la mayoría de centros asistenciales del país.
- Se centra en el paciente, que es el eje del sistema.
- Ofrece una visión continua y coherente de la actividad asistencial.
- Facilita la rápida configuración de la historia de salud.
- Integra toda la información generada desde el centro, lo que facilita el desempeño tanto de los procesos asistenciales como de la realización de las funciones directivas y administrativas.

1.3 MOTIVACIONES

Los estudios universitarios y, sobretodo, las ingenierías deberían tener de forma obligatoria una vertiente práctica en un entorno real, en la que el alumno pueda aplicar los conocimientos teóricos adquiridos a lo largo de la carrera universitaria. Basándome en esa creencia, expongo a continuación los motivos personales que me han llevado a realizar el proyecto de fin de carrera en UNIT4 Ibérica:

- Al tratarse de una multinacional reconocida en el mundo de las tecnologías de la información, me ha permitido ver de primera mano las metodologías de trabajo y la organización y estructura de empresas con un capital humano y un volumen de negocios similar a UNIT4.

- El hecho de tratarse de un proyecto real me ha ayudado a adquirir ciertos hábitos y experiencia que sólo son asequibles adentrándose en el mundo laboral.

- Resulta ser la forma más eficiente y práctica para realizar el proyecto de fin de carrera.

1.4 ORGANIZACION DE LA MEMORIA

La memoria está organizada en siete capítulos que se detallan brevemente a continuación:

- **Capítulo 1 (Introducción):** En el primer capítulo se define el proyecto a grandes rasgos. Se exponen tanto los objetivos como el producto al que van asociados, para luego presentar la empresa a la cual pertenece dicho producto y las motivaciones que han llevado la realización del proyecto.

- **Capítulo 2 (Definición de requerimientos):** Capítulo en el que se detalla el sistema actual así como el sistema propuesto, realizando un plan de acción con sus beneficios, costes y riesgos.

- **Capítulo 3 (Análisis):** En este capítulo estudiaremos el funcionamiento general del sistema propuesto, identificando a sus usuarios, los componentes que lo constituyen y los procesos principales.

- **Capítulo 4 (Diseño):** Con el objetivo de mejorar la experiencia del usuario, en la fase de diseño se define la interfaz gráfica y la interacción del software con los usuarios del sistema.

- **Capítulo 5 (Implementación):** En la fase de implementación se profundiza en el entorno de desarrollo empleado en la codificación, así como en las herramientas utilizadas para la creación de la interfaz y el tratamiento de ficheros HL7.

- **Capítulo 6 (Pruebas):** En este capítulo se resumen las pruebas unitarias y de integración que se han planteado y su ejecución y resultado.

- **Capítulo 7 (Conclusiones):** En el último capítulo, se detallan los objetivos conseguidos y los no conseguidos, explicando después las posibles mejoras y ampliaciones que pueden implementarse en el futuro y finalizando con las valoraciones personales del proyectista.

2 DEFINICIÓN DE REQUISITOS

2.1 SISTEMA ACTUAL

2.1.1 Descripción

El sistema actual está compuesto por dos aplicaciones programadas en Visual Basic 6 que hacen las funciones de cliente y servidor para el envío de ficheros y recepción. A continuación, se detalla el funcionamiento de cada una de las aplicaciones y la lógica actual del sistema:

- **EnviaHL7:** La aplicación de envío de ficheros utiliza la información de red del terminal receptor para realizar una llamada a éste para saber si está disponible y mandar el archivo seleccionado.

  La entrada de datos no se valida y tampoco se comprueba si el fichero seleccionado es un archivo HL7:

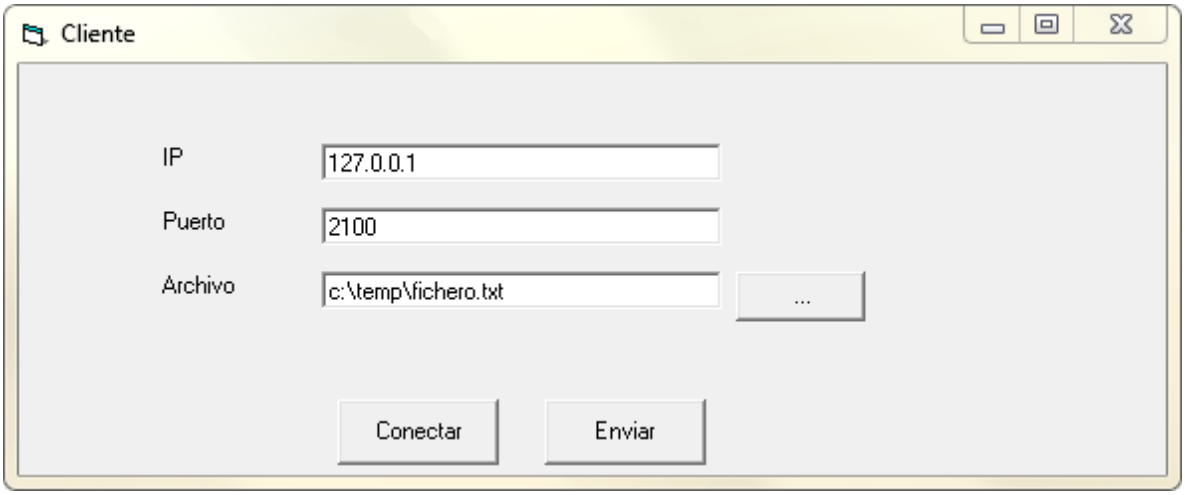

Figura 1-Aplicación de envío en Visual Basic

➲ **RecibeHL7:** El programa de recepción de ficheros sirve para aceptar las peticiones de envío de otros terminales y seleccionar la ruta de destino de tales archivos.

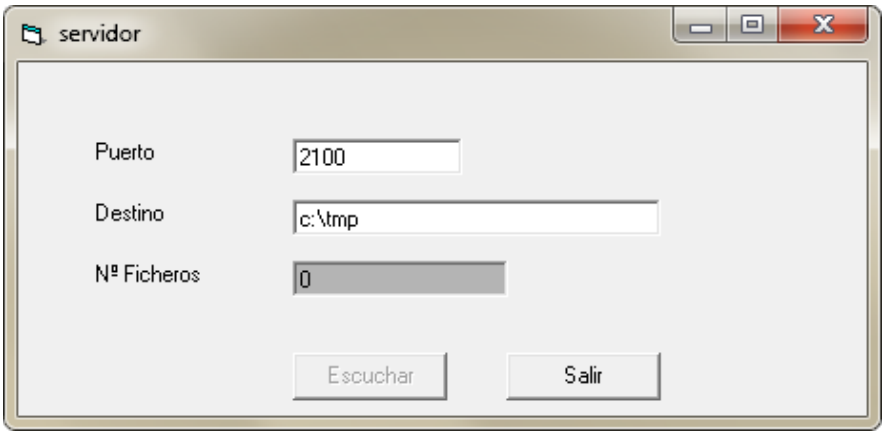

Figura 2 - Aplicación de recepción en Visual Basic

La lógica del sistema actual es muy sencilla. La aplicación "cliente" se ejecuta de forma manual y se pone en modo de escucha para poder recibir una petición de conexión desde la aplicación "servidor" y así, recibir los ficheros que se envíen desde el otro lado.

Por otro lado, la aplicación "servidor" solamente se ejecuta bajo petición del ERP ekon salus, mediante un script que se ejecuta automáticamente cuando se realiza una petición de información médica por parte de un usuario del sistema.

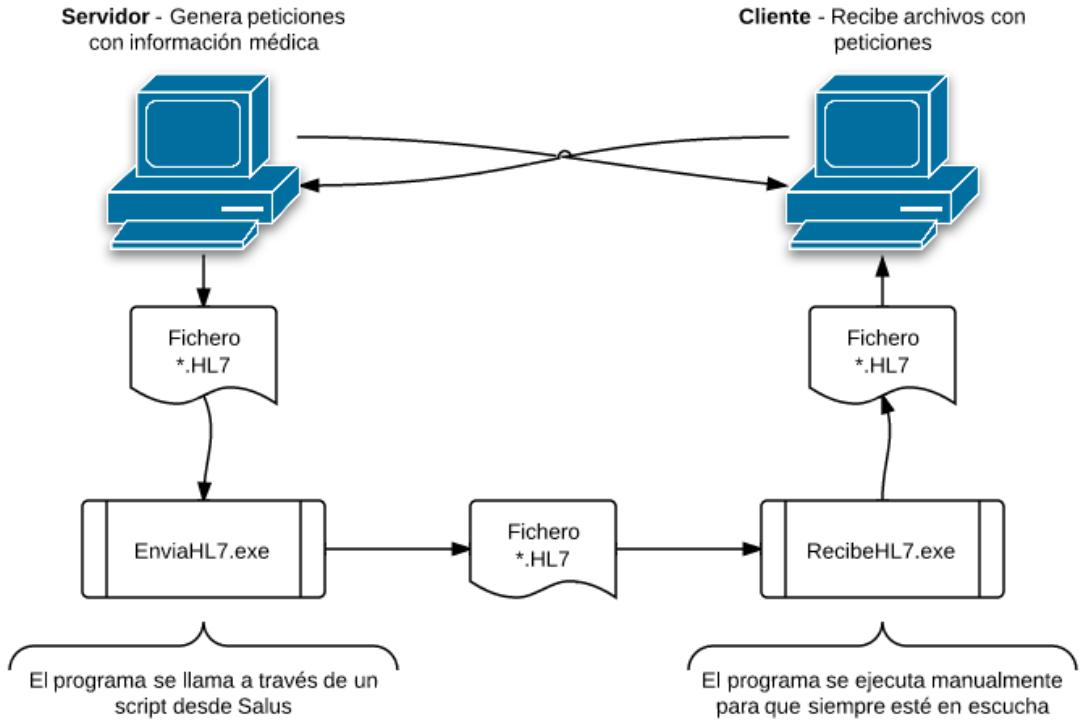

Figura 3 - Lógica del sistema actual

2.1.2 Diagnóstico

El sistema actual es sencillo y eficaz. En condiciones óptimas envía el fichero, cumpliendo así con su objetivo crítico. Paralelamente, el sistema falla cuando se trata de detectar y registrar fallos, inconsistencias en el formato y forma del fichero o problemas de conexión. Asimismo, carece de herramientas adecuadas a las necesidades de los administradores de sistemas, tales como un visualizador de eventos y/o un sistema de alertas.

La conclusión obvia es que el sistema actual se diseñó bajo unos requisitos que, en su momento, eran suficientes pero que ahora se han quedado obsoletos respecto a las características del software de UNIT4.

2.1.3 Alternativas

- Desarrollar los cambios necesarios en el sistema para que además de enviar y recibir los archivos, éste permita que la aplicación "servidor", que se encarga del envío, también genere el fichero codificado directamente desde el proceso de solicitud de peticiones.
  - Alternativa rechazada por solapamiento de procesos, ya que UNIT4 Ekon Salus implementa la generación de dichas peticiones y no se estima necesario un proceso complementario que no aporta por si solo valor añadido al producto.

- Desarrollar los cambios necesarios en el sistema de mensajería para que se adapte a los nuevos requisitos exigidos por la empresa. A su vez, se portará el sistema actual desarrollado en Visual Basic 6 a Java, introduciendo cambios en el diseño y la implementación de las aplicaciones de manera que el producto resultante sea más robusto y adaptable a los futuros cambios de requisitos.
  - Alternativa aceptada al no solaparse con ninguno de los procesos ya implementados por el software ekon salus y suponer una mejora considerable del sistema actual.

2.2 SISTEMA PROPUESTO

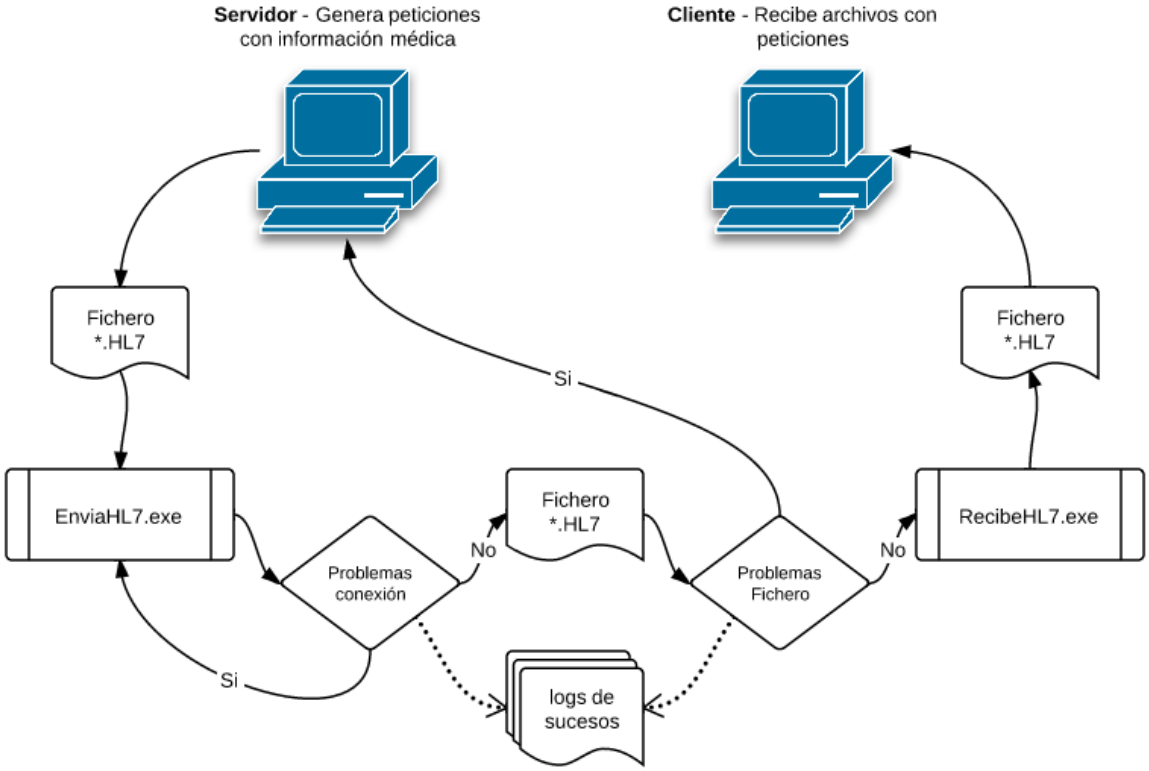

Figura 4 - Lógica del sistema propuesto

La lógica del sistema propuesto supone un cambio bastante considerable respecto al esquema actual. Para empezar, durante el ciclo de vida del fichero HL7, la aplicación EnviaHL7 se encargará de comprobar que no haya problemas de conexión y analizará el contenido de los archivos codificados en busca de errores de codificación.

Por su parte, la aplicación RecibeHL7 también analizará dichos ficheros para comprobar que no se ha sufrido ningún tipo de pérdida de información durante el proceso de envío.

Para terminar, todos los procesos realizados por las partes implicadas, así como las incidencias encontradas tanto en el proceso de envío como en el análisis de los archivos serán registrados en una serie de logs que podrán ser luego tratados por los administradores de sistemas.

2.2.1 Objetivos

En la siguiente tabla se describen todos los objetivos que se han propuesto para la realización del proyecto. Los objetivos han sido divididos en tres clases dependiendo de la prioridad de los mismos, resultando ser los objetivos críticos los de obligada consecución para realizar con éxito el proyecto:

Objetivo	Crítico	Prioritario	Secundario
Rediseñar y portar el producto de VB6 a Java.	✓		
Añadir proceso de verificación del contenido del mensaje.	✓		
Implementar log de sucesos para los administradores de sistemas.	✓		
Creación de un proceso de detección y catalogación de problemas durante el proceso de comunicación.		✓	
Inclusión de un sistema de alertas mediante widget en Ekon Salus y/o dispositivos móviles.			✓

Tabla 1 - Objetivos

2.2.2 Objetos

A continuación, se puede observar un esquema de los objetos que forman parte del sistema de mensajería, así como los elementos de entrada necesarios para su correcto funcionamiento y los elementos que forman parte de la salida de los objetos.

- **Aplicación de envío**
  - Entrada:
    - Ubicación del fichero *.HL7.
    - IP y puerto de destino.

- Salida:
  - Fichero *.HL7.
  - Información de envío.
  - Información acerca del contenido del fichero.

➲ **Aplicación de recepción**

- Entrada:
  - Puerto de escucha.
  - Ubicación destino del fichero *.HL7.
- Salida:
  - Fichero *.HL7.

2.2.3 Acontecimientos y acciones

En la tabla que nos encontramos a continuación, hay una serie de sucesos, previstos durante la interacción de las aplicaciones, que forman parte del sistema de comunicación junto a las acciones que se espera realice el sistema.

Acontecimiento: Un usuario envía un fichero HL7 sin errores de codificación y hay problemas de conexión. **Acción:** Guardar el archivo en la ruta especificada por la aplicación RecibeHL7 y registrar acontecimiento.
Acontecimiento: Un usuario envía un fichero HL7 con errores de codificación y no hay problemas de conexión. **Acción:** Enviar/No enviar fichero, reportar fichero con errores detectados y registrar acontecimiento
Acontecimiento: Un usuario envía un fichero HL7 sin errores de codificación y se detectan problemas de conexión. **Acción:** Identificar problemas de conexión, intentar el reenvío, reportar error y registrar acontecimiento

Tabla 2 - Acontecimientos y acciones

2.2.4 Funcionalidades

La siguiente lista contiene las funcionalidades que el sistema de comunicación tendrá implementadas una vez haya finalizado el desarrollo del proyecto y sin las cuales no se habrán conseguido alcanzar los objetivos antes expuestos.

- Envío y recepción de ficheros.
- Recepción de ficheros desde distintos servidores (multicliente).
- Registro de sucesos.
- Reenvío de ficheros.
- Extracción de información para documentar al usuario.

2.2.5 Otros requisitos no funcionales

Otros requisitos no asociados a los objetivos del proyecto pero que afectan al desarrollo y duración del mismo se listan a continuación.

- La duración máxima del proyecto es de 560 horas.
- La fecha límite de entrega del proyecto es el 30 de junio del 2012.

2.2.6 Restricciones de software

Para el desarrollo de las nuevas aplicaciones de envío y recepción se utilizará el lenguaje de programación JAVA a través del entorno de desarrollo integrado *Eclipse*.

Para la implementación de la interfaz gráfica de usuario y el análisis de ficheros codificados en formato HL7 sólo se podrán utilizar plugins o librerías con licencia GNU de libre uso y distribución.

Opcionalmente, para el desarrollo del widget de alertas en Ekon Salus se utilizaría el entorno de desarrollo Karat.

2.2.7 Condicionantes de implantación y funcionamiento

Para la correcta implantación y funcionamiento se deben tener en cuenta los siguientes condicionantes:

- El producto resultante del desarrollo del proyecto no puede substituir ni solaparse con ningún proceso de Ekon Salus.
- Los datos de entrada de las aplicaciones de envío y recepción deben ser introducidos automáticamente por norma general.
- No se puede modificar el fichero en ningún caso, sólo se puede ver para comprobar la validez del mismo.

2.2.8 Plan de validación

A lo largo del proyecto, éste se ha desarrollado de manera secuencial y, en cada fase que se ha realizado, se han ejecutado pruebas objetivas que evidenciaran que los procesos producen un resultado y que el producto resultante cumple con los requisitos.

2.3 PLAN DE ACCIÓN

2.3.1 Actividades

En la siguiente tabla, se detallan todas las tareas que se realizarán durante el desarrollo del proyecto. Estas tareas están divididas en cinco fases diferenciadas que se realizan de forma lineal en el siguiente orden: Introducción y planificación, Análisis, Diseño, Codificación y pruebas y, en último lugar, Documentación:

TAREA	DURACIÓN
SISTEMA DE COMUNICACIÓN PARA MENSAJERÍA DE FICHEROS	**472 horas**
INTRODUCCIÓN Y PLANIFICACIÓN	**20 HORAS**
Objetivos	4 horas
Análisis del producto existente	4 horas
Planificación temporal	8 horas
Evaluación de costes	4 horas
ANÁLISIS	**20 HORAS**
Definición de requisitos	10 horas
Análisis de requisitos	10 horas
DISEÑO	**40 HORAS**
Diseño de la IGU	20 horas
Patrones de diseño	12 horas
Diseño de la interacción	8 horas
CODIFICACIÓN Y PRUEBAS	**300 HORAS**
CODIFICACIÓN	**260 HORAS**
Interfaz gráfica de usuario	40 horas

Sistema de mensajería de ficheros	80 horas
Sistema de validación de ficheros HL7	140 horas
PRUEBAS	**40 HORAS**
Definición de pruebas	12 horas
Realización de pruebas	28 horas
DOCUMENTACIÓN	**92 HORAS**
Memoria del proyecto	80 horas
Manuales de usuario	12 horas
ENTREGA DE LA MEMORIA	
DEFENSA DEL PROYECTO	

Tabla 3 - Tareas y duración

2.3.2 Diagrama de Gantt

En la siguiente página, podemos visualizar el diagrama de Gantt del proyecto. En él se puede observar la distribución y orden de las tareas a realizar durante el proyecto así como la distribución temporal de cada una de ellas y la fecha de inicio y fin previstos.

Al realizarse una planificación temporal lineal, todas ellas están asociadas de principio a fin con la tarea inmediatamente anterior, exceptuando la entrega y defensa del proyecto, que tiene fechas prefijadas.

El inicio oficial del proyecto está fijado en el día 14 del mes de noviembre de 2011 y la fecha prevista para la finalización del mismo es el día 25 del mes de abril de 2012.

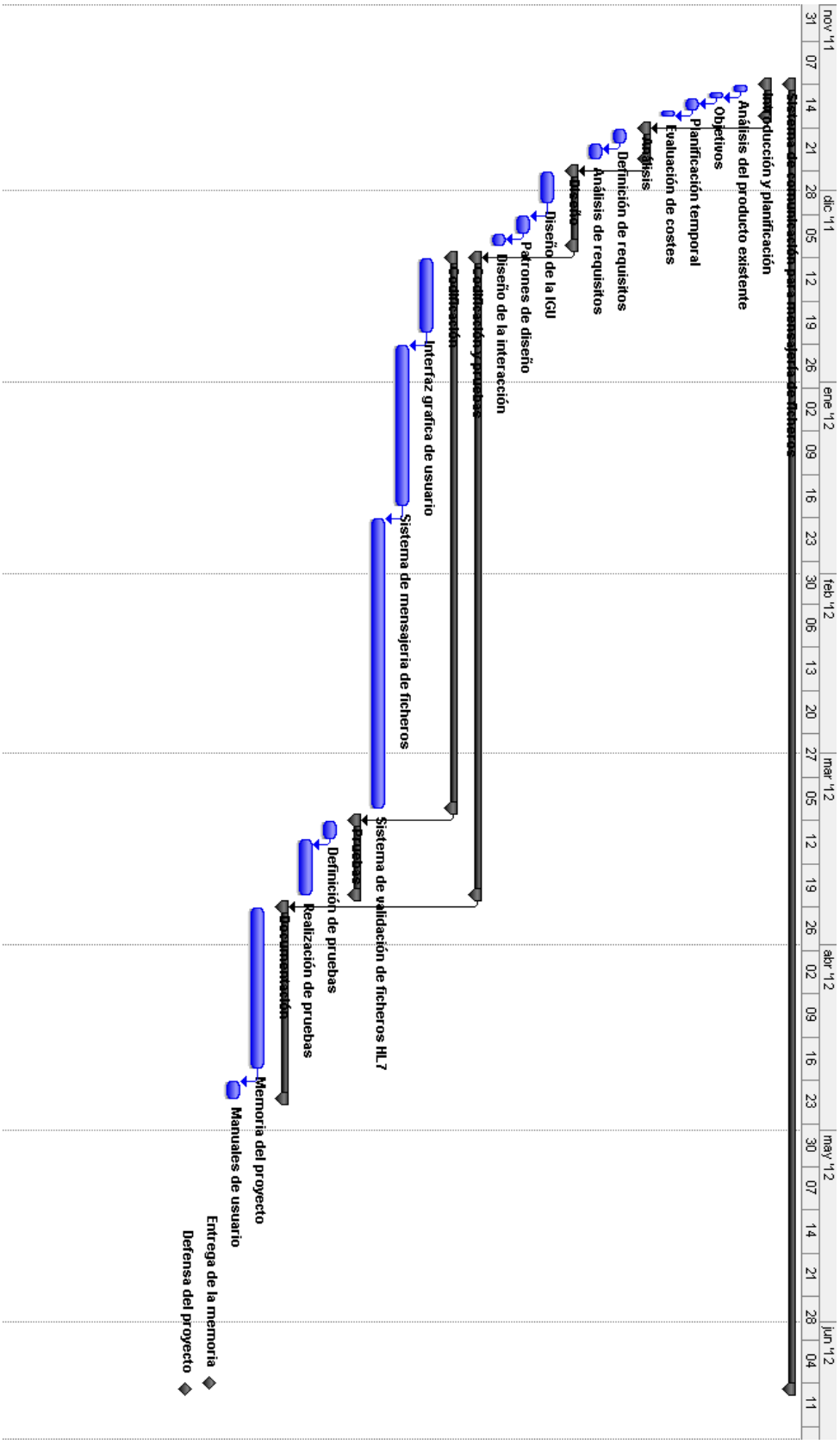

Figura 5 - Diagrama de Gantt inicial

2.3.3 Recursos

A continuación, se exponen los recursos disponibles para realizar el proyecto, partiendo de las actividades, mencionadas anteriormente, que se han tenido que llevar a cabo.

- **Recursos humanos**
  - *Jefe de proyecto:* participa en la planificación del proyecto, en la validación de las distintas fases y en la defensa del proyecto.
  - *Analista:* forma parte activa del análisis, diseño e implementación del proyecto.
  - *Programador:* participa en todo el proceso de implementación y codificación del proyecto.
  - *Técnico de pruebas:* durante las últimas fases del proyecto, se encarga de la fase de pruebas.
- **Recursos materiales**
  - PC HP Compaq 6000 Pro MT, con las siguientes características:
    - *Procesador:* Intel Core 2 Quad Q8400 @ 2,66 GHz.
    - *Memoria RAM:* 6,00 GB.
    - *Sistema operativo:* Windows 7 Professional 64 bits.

- *Software*

SOFTWARE	LICENCIA	DESCRIPCIÓN
Eclipse IDE for Java Developers	EPL – Licencia Publica Eclipse	Entorno de desarrollo integrado que acepta, entre otros lenguajes, Java-
JDK Java 6	GNU GPL – Licencia Publica General	Paquete para desarrolladores Java
WindowBuilder	EPL – Licencia Publica Eclipse	Generador de interfaces gráfica de usuario en Java con la librería Swing
Hapi	GNU GPL – Licencia Publica General	Librería Java para el análisis de archivos HL7

Tabla 4 - Software utilizado

2.3.4 Costes, beneficios y riesgos

- **Costes**

  El capital humano para la realización del proyecto está formado por un analista-programador que además hace las funciones de jefe de proyecto; y por un Product Owner de la empresa que se encargara de tutorizar, en cada una de sus fases, el trabajo del analista programador.

  No se estima que se requieran licencias de software de pago complementarias para la realización del proyecto.

- **Beneficios**

  Los potenciales beneficios del proyecto son:

  - Mayor control del estado de las peticiones.
  - Mejor gestión de las incidencias.
  - Reducción del tiempo de comunicación.

- **Riesgos**

  Los riesgos relevantes adscritos al proyecto son:

  - Planificación optimista sobre la duración de las tareas.
  - Cambio en los requerimientos funcionales/no funcionales.
  - Adición de tareas no previstas.

2.3.5 Conclusión

Después de planificar las tareas necesarias para realizar el proyecto en base a los objetivos del mismo y tras estudiar los recursos disponibles, los costes que suponen y los beneficios que aporta el proyecto al producto actual que tiene UNIT4, aun teniendo en cuenta los riesgos adyacentes al desarrollo, se concluye que el proyecto es viable y beneficioso, tanto para la empresa como para el alumno.

3 FASE DE ANÁLISIS

3.1 INTRODUCCIÓN

Un usuario autentificado puede realizar una petición de información determinada respecto a un paciente. Para que se envíe la petición es necesario que la información esté bien codificada y se pueda establecer conexión, tanto por la parte que envía como por la que recibe. Cada vez que se intente enviar información, se almacenará la información en un registro.

3.2 VISIÓN GENERAL

3.2.1 Diagrama de contexto

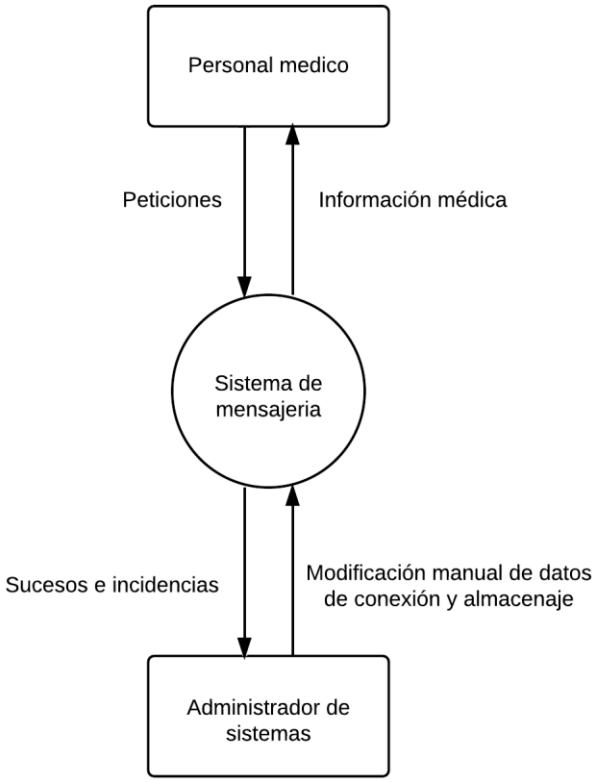

Figura 6 - Diagrama de contexto

En el diagrama de contexto, podemos observar la interacción de los usuarios con el sistema de mensajería, incluyendo los flujos de entrada y salida de información contrario.

3.2.2 Descripción de los componentes

USUARIOS	DESCRIPCIÓN
Personal médico	Toda aquella persona que trabaja en un hospital realizando trabajo de índole médica. No tienen conocimientos informáticos e interactúan con el sistema de mensajería realizando peticiones y recibiendo información con contenido sensible de pacientes. El sistema de mensajería es transparente para este tipo de usuarios.
Administrador de sistemas	Realiza las labores de instalación y mantenimiento de sistemas y redes. Interactúa con el sistema de mensajería modificando los valores de entrada del mismo si se estima necesario y recibe información de sucesos e incidencias asociadas al envío y recepción de mensajes.

Tabla 5 - Usuarios del sistema

3.3 MODELO DE OBJETOS

3.3.1 Diagrama

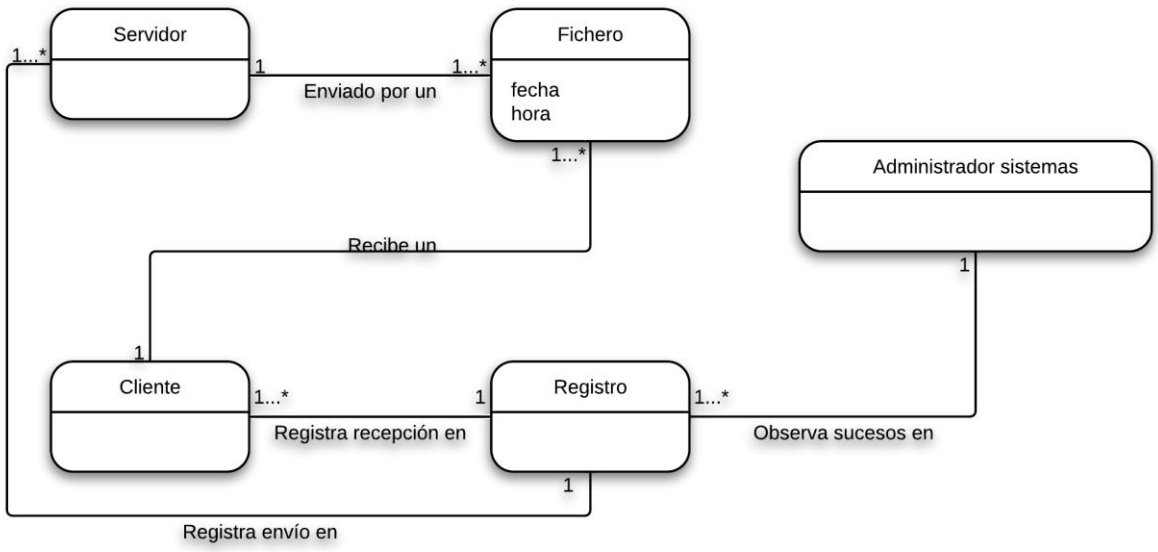

Figura 7 - Diagrama de objetos

En el diagrama de objetos podemos observar cómo, una vez identificados en el diagrama de contexto los usuarios del sistema y su relación con el mismo, hay una serie de relaciones entre los objetos que forman parte del sistema a desarrollar.

La interacción que hay entre estos objetos es en algunos casos unilateral y, en otros, multidireccional, también se puede apreciar que componentes son receptores y/o cuales envían información.

3.3.2 Descripción de componentes

La siguiente tabla contiene una breve descripción de los objetos que forman parte del sistema, explicando la información que poseen y su relación con otros objetos.

OBJETOS	DESCRIPCIÓN
Fichero	Objeto que contiene un fichero. Sus atributos principales son *fecha* y *hora* del proceso de envío/recepción
Servidor	Objeto que envía uno o más objetos del tipo Fichero a un objeto Cliente.
Cliente	Objeto que recibe uno o más objetos del tipo Fichero de un objeto Cliente.
Administrador sistemas	Objeto que observa las acciones de los objetos Servidor, Cliente y Fichero a través de un objeto Registro
Registro	Objeto que almacena las operaciones realizadas por objetos Servidor, Cliente y Fichero

Tabla 6 - Objetos del sistema

3.4 MODELO FUNCIONAL

3.4.1 Diagrama de flujo de datos (DFD)

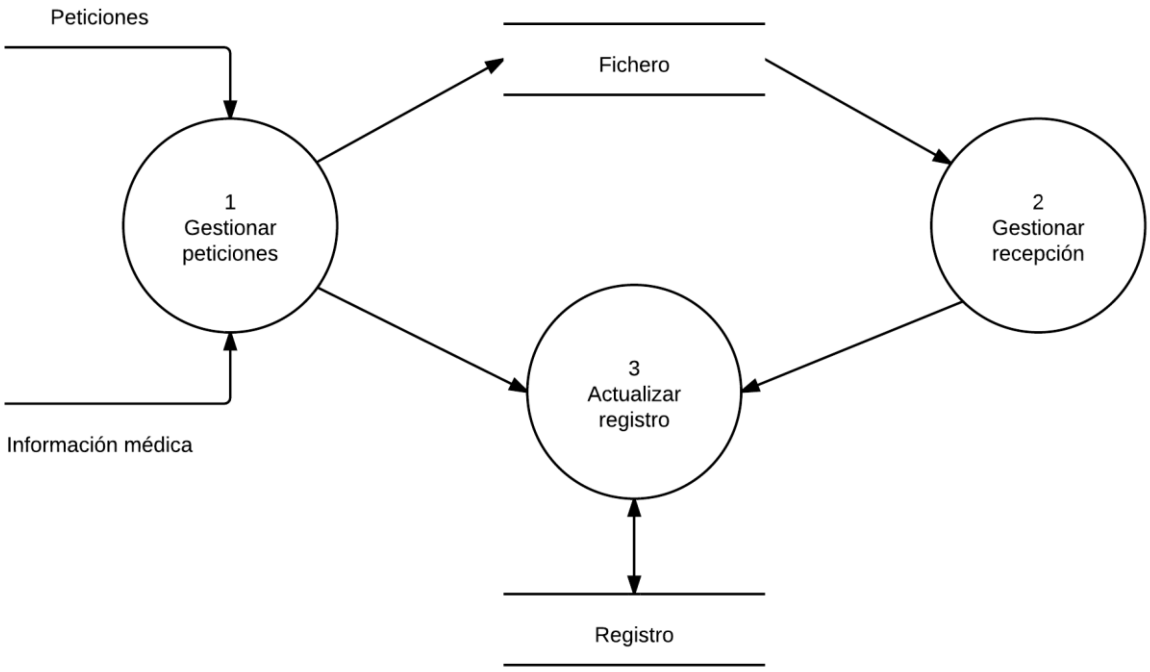

Figura 8 - Diagrama de Flujo de Datos

El diagrama de flujo de datos nos muestra de forma visual cual es el flujo de datos entre los distintos procesos, entidades externas y almacenes que conforman el sistema.

3.4.2 Descripción de componentes

La tabla que podemos observar a continuación, contiene los procesos identificados como emisores y receptores de los flujos de datos del sistema. En la descripción de los procesos se detalla el origen y el destino de la información a través de los usuarios que interactúan con el sistema.

PROCESOS	DESCRIPCIÓN
Gestionar peticiones	Un usuario, previa identificación en Ekon Salus, puede realizar una o varias peticiones de información de pacientes. Las peticiones generan ficheros HL7 que podrán ser enviados o no dependiendo del contenido del mismo y de la red.
Gestionar recepción	Un usuario, previa identificación en Ekon Salus, puede acceder a los ficheros recibidos
Actualizar registro	Un usuario administrador de sistemas puede supervisar la gestión de peticiones y recepción, así como mantener un control sobre el registro de operaciones.

Tabla 7 - Procesos del sistema

4 FASE DE DISEÑO

4.1 INTRODUCCIÓN

En la fase de diseño se explicará el diseño escogido en base a los requerimientos del proyecto. También, se hablará del diagrama UML (Lenguaje Unificado de Modelado) realizado a partir de los resultados obtenidos en la fase de análisis y de la necesidad de utilizar los patrones de diseño para solucionar problemas surgidos en el diseño de las aplicaciones.

4.2 DISEÑO DE LA INTERFAZ GRÁFICA DE USUARIO

4.2.1 Interfaz de RecibeHL7

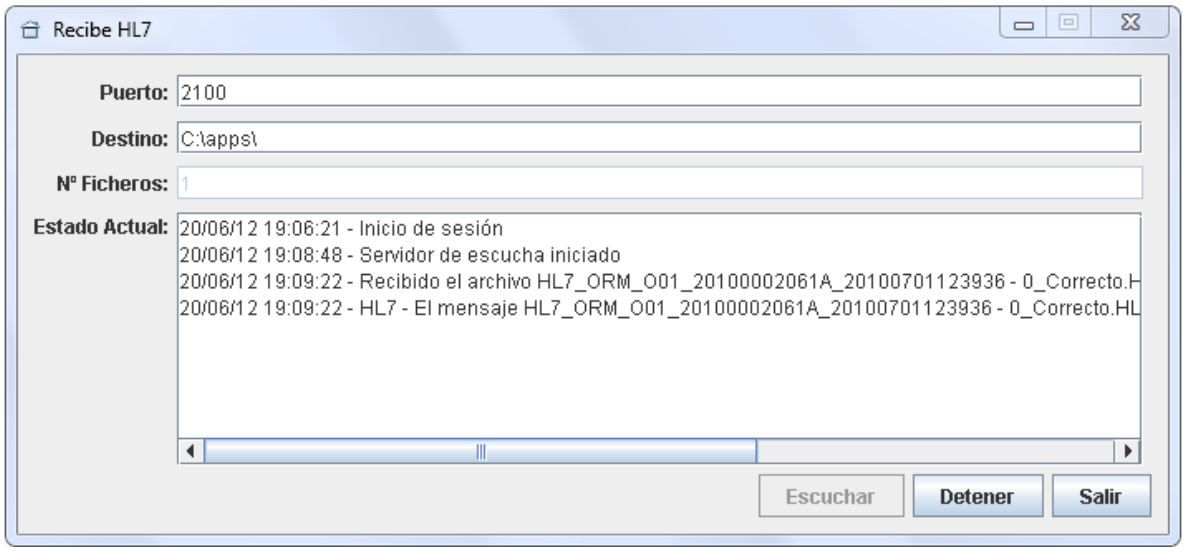

Figura 9 - Interfaz de RecibeHL7 en Java

Los principales cambios previstos en el diseño de la interfaz gráfica de usuario de la aplicación RecibeHL7, para adaptarse a los requerimientos del proyecto son:

➲ Redimensionado de la ventana principal para que los datos introducidos en los campos

de entrada y los que se muestran en el visor de sucesos se vean completos.

- Se ha insertado un visor de sucesos (estado actual) que nos informa en cada momento de:
  - El estado de la conexión.
  - La recepción de los ficheros.
  - El resultado del análisis de los archivos con los filtros HL7.
- Se ha añadido el botón "Detener" para que el administrador del sistema pueda interactuar con la aplicación de forma manual.

4.2.2 Interfaz de EnviaHL7

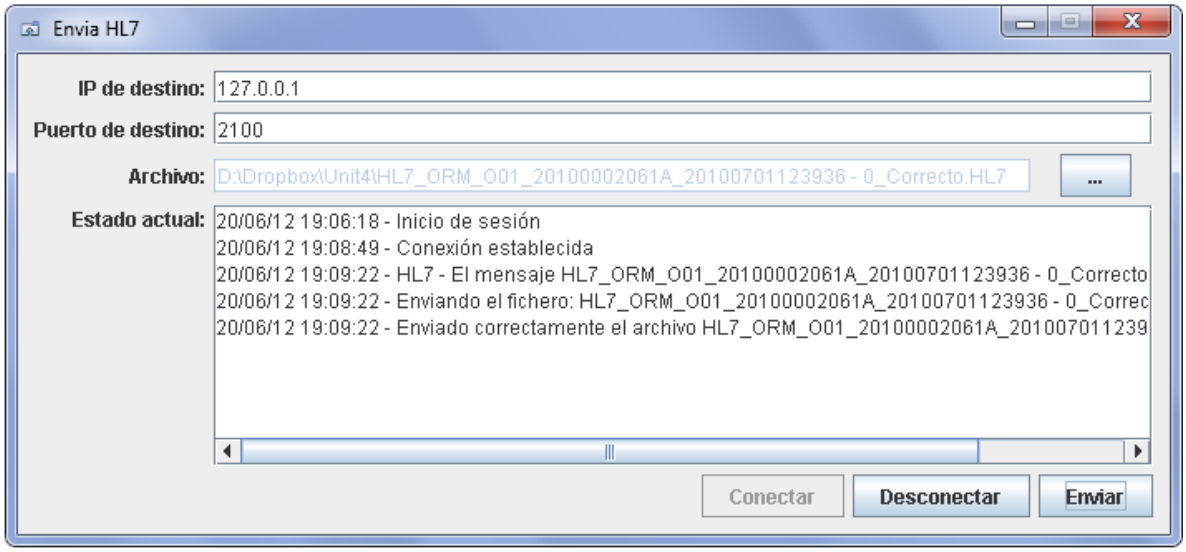

Figura 10 - Interfaz de EnviaHL7 en Java

Los principales cambios previstos en el diseño de la interfaz gráfica de usuario de la aplicación EnviaHL7, para adaptarse a los requerimientos del proyecto, son los siguientes:

- Se ha desactivado la entrada manual de texto en el campo de entrada archivo para que el usuario obligatoriamente pase por el selector de ficheros que contiene la máscara para que sólo se puedan seleccionar archivos con extensión HL7.
- Redimensionado de la ventana principal para que los datos introducidos en los campos de entrada se vean completos.
- Se ha insertado un visor de sucesos que nos informa en cada momento de:
  - El estado de la conexión.

- El envío y recepción de los ficheros.
- El resultado del análisis de los archivos con los filtros HL7.

➲ Se ha añadido el botón "Desconectar" para que el administrador del sistema pueda interactuar con la aplicación de forma manual.

4.3 Diagramas UML

La estructura principal de las aplicaciones EnviaHL7 y RecibeHL7 se ha diseñado a través de la realización de diagramas UML, especificando las clases y métodos que se necesitan según los requerimientos.

4.3.1 Recibe HL7

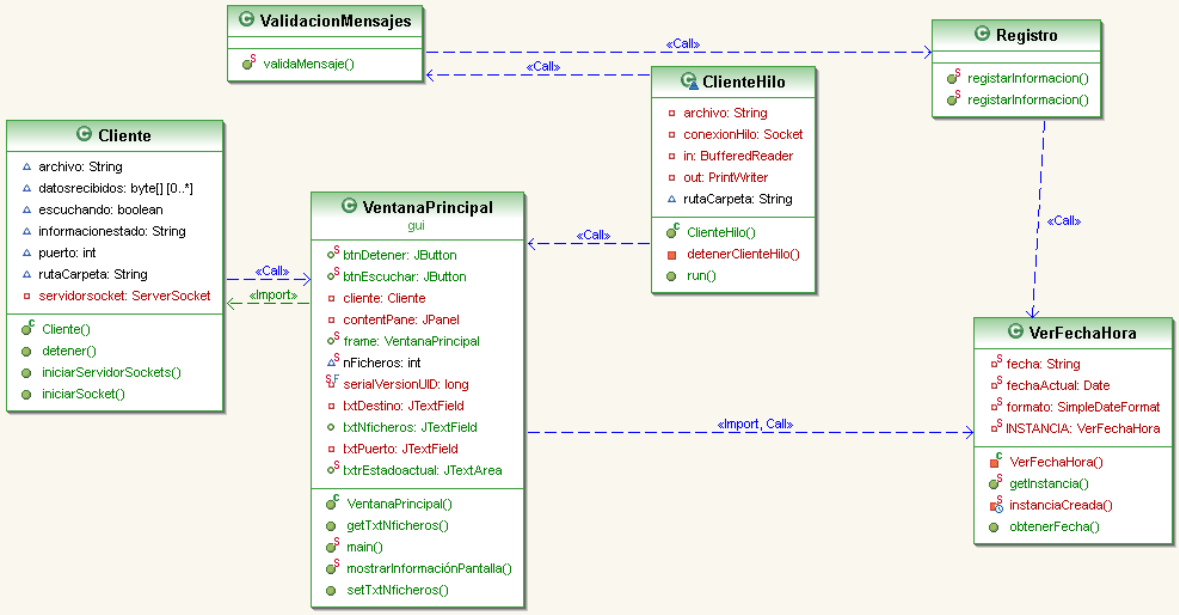

Figura 11 - Diagrama UML de Recibe HL7

En la siguiente tabla se describen las principales clases que forman parte de la aplicación RecibeHL7 y los métodos por los cuales se producen los flujos de datos.

Para poder implementar la recepción de ficheros desde distintos servidores a la vez, se ha creado una clase llamada ClienteHilo que genera hilos de ejecución simultáneos, asignando un puerto distinto a cada una de las aplicaciones EnviaHL7 que intenta

conectarse a RecibeHL7.

CLASE	DESCRIPCIÓN
Ventana Principal	Esta clase contiene toda la interfaz, de manera que la implementación principal del cliente en el sistema de mensajería sea independiente de la implementación de la interfaz.
Cliente	Es la clase principal de la aplicación RecibeHL7, en ella se creará el servidor de conexiones entrantes y creará tantos objetos de la clase ClienteHilo como sean necesarios.
ClienteHilo	La clase ClienteHilo se encarga de aceptar la conexión con la aplicación EnviaHL7 y almacenar los archivos recibidos en la ruta indicada desde la clase VentanaPrincipal.
Registro	Esta clase se encarga de guardar toda la información de los sucesos producidos en un fichero de texto plano. Se realiza la misma acción con los ficheros HL7 que contienen errores.
ValidacionMensajes	ValidacionMensajes recoge el archivo que ha sido recibido por parte de la aplicación EnviaHL7 y lo pasa por varios filtros para comprobar si contiene errores. Devuelve el resultado del análisis.
VerFechaHora	Se trata de una clase global que envía la fecha y la hora actual a todas aquellas clases que lo demandan.

Tabla 8 - Clases de la aplicación RecibeHL7

4.3.2 Envía HL7

A continuación, podemos ver el diagrama UML de la Aplicación EnviaHL7. El diagrama es similar al que hemos visto anteriormente de RecibeHL7, con la diferencia de que la clase principal en este caso es Servidor.

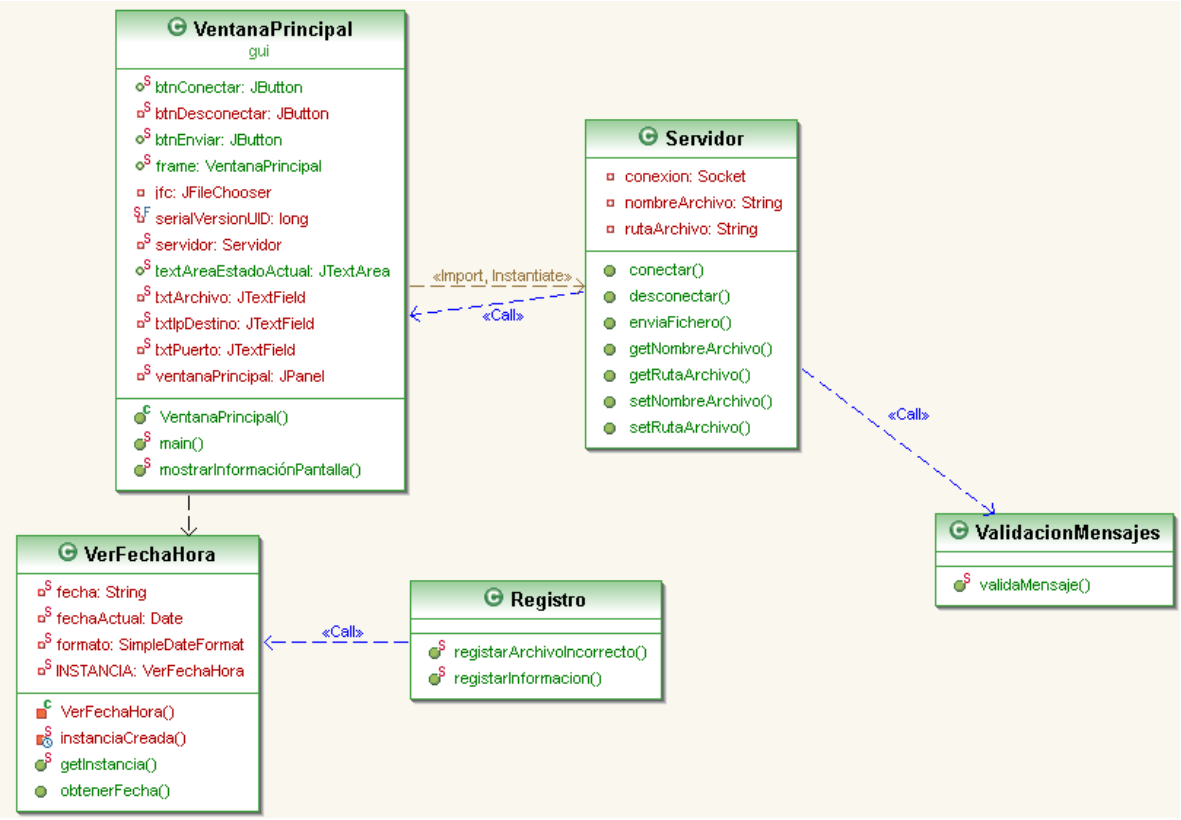

Figura 12 - Diagrama UML de Envía HL7

En la siguiente tabla, se describen las principales clases que forman parte de la aplicación EnviaHL7 y los métodos por los cuales se producen los flujos de datos.

CLASE	DESCRIPCIÓN
VentanaPrincipal	Esta clase contiene toda la interfaz, de manera que la implementación principal del servidor en el sistema de mensajería sea independiente de la implementación de la interfaz.
Servidor	Es la clase principal de la aplicación EnviaHL7, en ella se creará y se comprobará si el puerto de destino de la dirección IP introducida en VentanaPrincipal está escuchando y, en caso afirmativo, manda el archivo a su destino y cierra la conexión.
Registro	Esta clase se encarga de guardar toda la información de los sucesos producidos en un fichero de texto plano. Se realiza la

	misma acción con los ficheros HL7 que contienen errores.
ValidacionMensajes	ValidacionMensajes recibe el archivo que se ha seleccionado para ser enviado en VentanaPrincipal y lo pasa por varios filtros para comprobar si contiene errores. Devuelve el resultado del análisis.
VerFechaHora	Se trata de una clase global que envía la fecha y la hora actual a todas aquellas clases que lo demandan.

Tabla 9 - Clases de la aplicación EnviaHL7

4.4 PATRONES DE DISEÑO

4.4.1 Introducción

Los patrones de diseño son, básicamente, soluciones a los problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Una solución a un problema de diseño sólo se considera un patrón si ha demostrado ser efectivo en casos anteriores y si se puede reutilizar en diferentes problemas de diseño en circunstancias diferentes.

El uso de patrones de diseño en el desarrollo de software no es obligatorio pero es aconsejable su uso cuando nos encontramos con problemas de diseño para los cuales existe algún patrón que los soluciona.

4.4.2 Singleton

El patrón de diseño singleton (instancia única) está diseñado para garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

En el proyecto, se ha detectado que la clase VerFechaHora se instancia múltiples veces para la visualización de datos en el visor de sucesos de las aplicaciones EnviaHL7 y APPRecibe, a la vez que estas aplicaciones también instancian la clase para guardar la información en los logs de sucesos. Para evitar un posible conflicto de peticiones derivadas del multi-hilo en APPRecibe, se ha usado el patrón singleton como solución al problema.

El patrón se implementa definiendo en nuestra clase un método que crea una instancia del objeto sólo si todavía no existe ninguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado).

La llamada a los métodos pertenecientes a la clase VerFechaHora se realiza de diferentes formas dependiendo de las veces que es llamado por un método:

- *VentanaPrincipal.java*: en la clase que se implementa la IGU (Interfaz gráfica de usuario) se llama múltiples veces al método obtenerFecha de la clase VerFechaHora. En este caso la mejor opción es crear un objeto del tipo VerfechaHora llamando al método que comprueba si ya existe una instancia o no de la clase, para después llamar al método que nos devuelve la fecha y la hora desde el objeto que hemos creado en VentanaPrincipal.

  El ejemplo siguiente es una posible implementación del diseño explicado:

```
Creamos el objeto en la clase VentanaPrincipal
>> VerFechaHora fechaHora = VerFechaHora.getInstancia();
Utilizamos el objeto creado para llamar al método de la clase
>> fechaHora.obtenerFecha();
```

- Registro.java: en la clase que se implementan los métodos que guardan la información de los sucesos en ficheros de texto plano, sólo se invoca una vez el método obtenerFecha, con lo cual no es necesario crear un objeto, lo que se hará será invocar el método directamente.

  El ejemplo siguiente es una posible implementación del diseño explicado:

```
Llamamos a obtenerFecha() a través del método estático getInstancia()
>> fechaHora.getInstancia().obtenerFecha();
```

5 FASE DE IMPLEMENTACIÓN

5.1 INTRODUCCIÓN

En la fase de implementación, explicaremos el entorno de desarrollo sobre el cual se ha realizado el proyecto, así como también las soluciones open source de libre distribución que se han utilizado para la implementación de la interfaz gráfica de usuario y el análisis de ficheros HL7.

5.2 ENTORNO DE DESARROLLO

Para la implementación del software que forma parte del sistema de comunicación, se ha utilizado Eclipse por ser éste un entorno de desarrollo integrado de código abierto y multiplataforma y que tiene una compatibilidad total con el kit de desarrollo de Java (JDK).

5.2.1 Desarrollo en Java

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Java es multiplataforma, el código compilado se puede ejecutar en cualquier sistema operativo que tenga instalada la máquina virtual de Java.

El proyecto de mensajería de ficheros está dividido en tres partes claramente diferenciadas:

- El código del cliente y del servidor basado en sockets.
- La interfaz gráfica de usuario con las librerías swing de Java.
- El código que permite analizar ficheros en formato HL7.

5.3 INTERFAZ: WINDOWBUILDER

5.3.1 Swing

Para desarrollar interfaces gráficas en Java existen dos bibliotecas llamadas AWT y Swing. En el presente proyecto se ha creído oportuno utilizar la biblioteca de gráficos Swing ya que aporta varias ventajas importantes respecto a AWT:

- Al ser más dinámico que AWT, el diseño en Java puro posee menos limitaciones de plataforma.
- El desarrollo de componentes Swing es más activo dado que su uso está más extendido.
- Los componentes de Swing soportan más características de personalización.
- Es independiente de la plataforma sobre la que estemos trabajando.
- Es una arquitectura altamente particionada: los usuarios pueden proveer sus propias implementaciones modificadas para sobrescribir las implementaciones por defecto. Se pueden extender clases existentes proveyendo alternativas de implementación para elementos esenciales.
- Dado el modelo de representación programático del framework de swing, el control permite representar diferentes estilos de apariencia "look and feel" copiando el estilo de los botones, tipos de letras y diseño de la plataforma sobre la que se ejecuta (Windows, MacOS, etc).
- El usuario puede proveer su propia implementación de apariencia, que permitirá cambios uniformes en la apariencia existente en las aplicaciones Swing sin efectuar ningún cambio al código de la aplicación.

5.3.2 Window Builder

Window Builder es una herramienta muy avanzada que simplifica enormemente el desarrollo de una interfaz al permitir la adición de componentes a través de arrastrar y soltar**,** además de generar código bidireccional, es decir que puede ser modificado tanto en la vista de diseño como en la de código.

Window Builder era una herramienta de pago que se distribuía como un plugin para

Eclipse, pero hace unos años fue adquirida por Google, que pasó a distribuirla como código open source, y actualmente sigue estando en continua evolución bajo la supervisión de Google.

En las figuras 13 y 14 podremos observar la interfaz de la herramienta Windows Builder ejecutándose sobre el entorno de desarrollo eclipse:

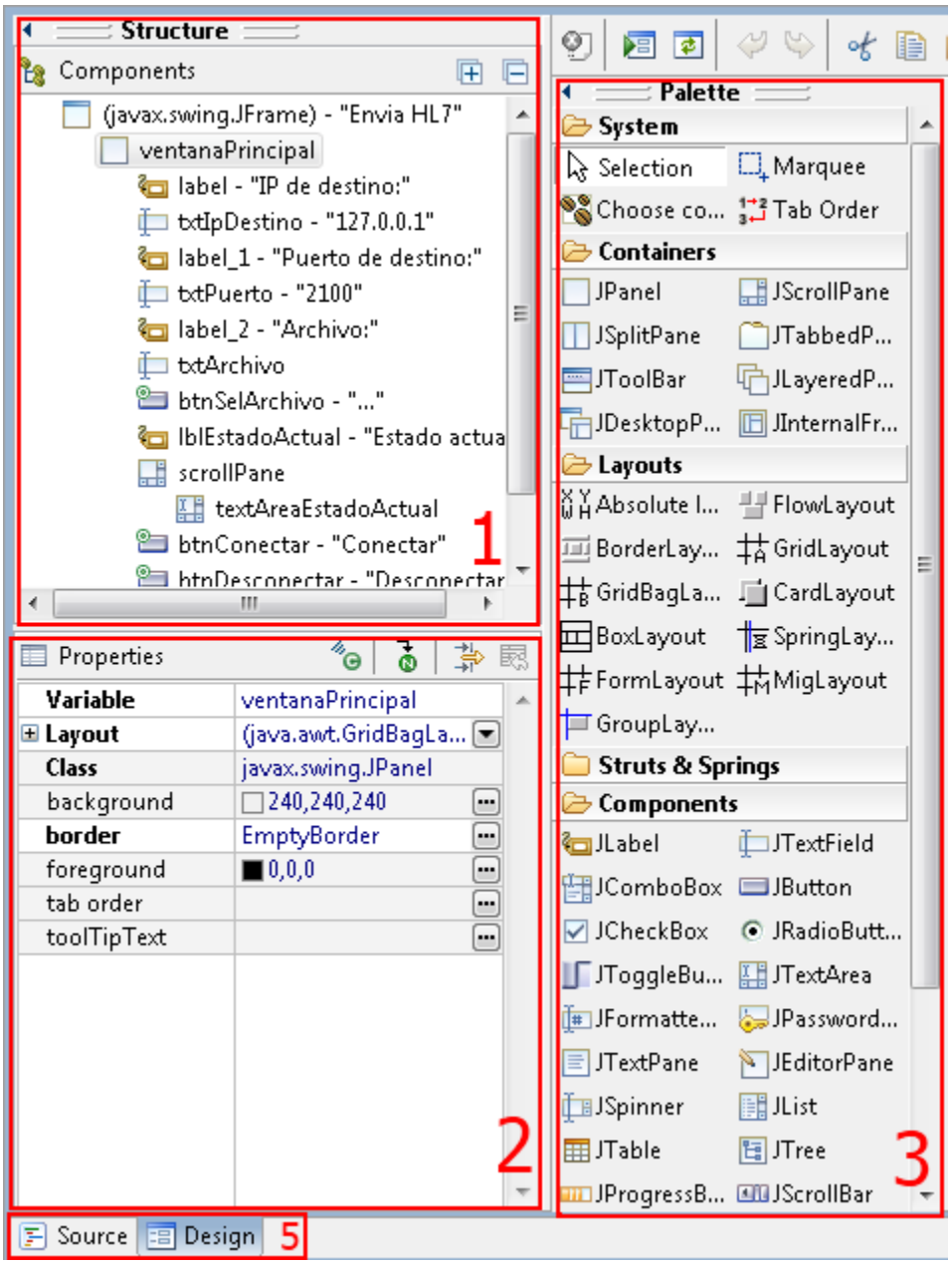

Figura 13 - Interfaz de Window Builder I

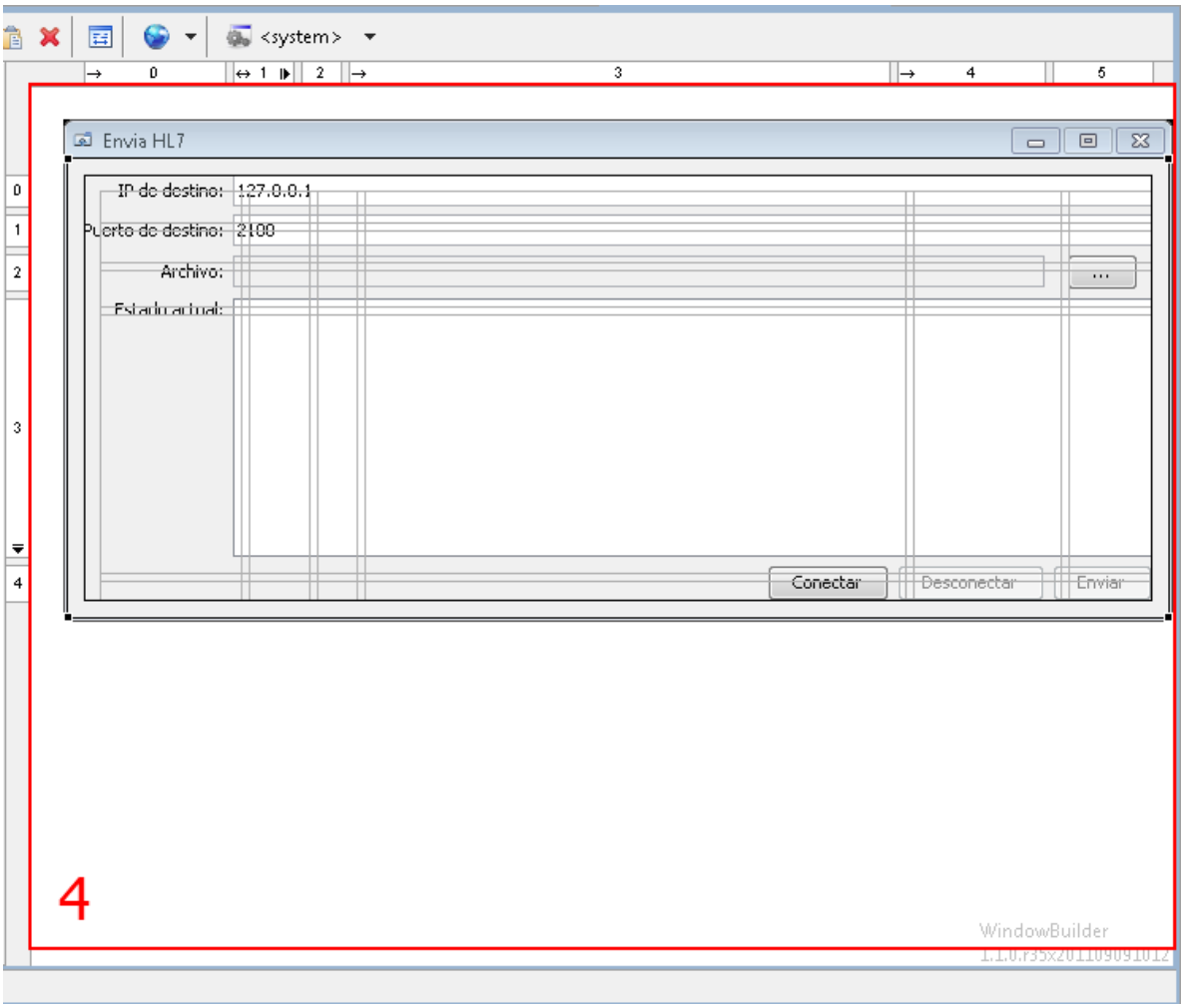

Figura 14 - Interfaz de Window Builder II

1. *Structure (Estructura):* En este cajón de la herramienta tenemos un esquema en forma de árbol en el que aparecen todos los componentes que se han insertado en la ventana y las dependencias de unos con otros.
2. *Propierties (Propiedades):* Cuando seleccionamos un componente en la ventana Estructura o Edición, aquí dispondremos de la mayoría de atributos principales del componente, pudiendo cambiar su color, texto y muchas cosas más sin tener que modificar código manualmente.
3. *Palette (Paleta):* En este panel encontramos todos los contenedores, capas y componentes disponibles en la librería Swing preparados para arrastrarlos a la ventana de edición.
4. *Ventana de Edición:* esta ventana es el eje principal de la herramienta Window Builder. Aquí podremos crear, editar y modificar todo el contenido de la pantalla, de forma que lo que vemos en ese momento será lo que se vea en el momento de

la ejecución. Además si hacemos doble clic en cualquier elemento, podremos ir al método que permite codificar acciones para ese elemento como podría ser por ejemplo la acción de un botón al ser pulsado.

5. Estas dos pestañas permiten generar código bidireccional, ya sea cambiando el código y pasando a la vista de diseño o modificando el diseño y pasando a visualizar el código.

5.4 HL7

La organización que se encarga de crear los estándares HL7 tiene varias especificaciones que indican como debe ser la estructura de los mensajes, el contenido, el tipo de datos y los caracteres de codificación que se deben usar, entre otras cosas.

5.4.1 Estructura de un mensaje HL7

Un mensaje HL7 está formado a su vez por uno o varios mensajes que se llaman segmentos. Cada uno de los segmentos tiene un número determinado de campos de datos, que deben estar debidamente codificados y delimitados por los separadores de campo.

A continuación se muestra como ejemplo un segmento perteneciente a un mensaje codificado según las especificaciones HL7:

```
MSH|^~\&|SALUS|ATIZA||2          /ONONELL|20100701123914||ORM^O01|20100701123914|P|2.5|||AL|ER
```

- Separador de campo "|": carácter utilizado para separar los campos de un segmento, ya sean estos obligatorios o no.
- Caracteres de codificación "^~\&": se utilizan para codificar el contenido de los campos pertenecientes a un segmento.
- Segmento: Cada mensaje está formado por un número de segmentos que varían dependiendo del origen de la petición. Los tres primeros caracteres son la cabecera del segmento que indican el tipo de segmento.

5.4.2 HAPI

El proyecto HAPI (HL7 Application Programming Interface) es una analizador de código abierto y orientado a objetos que acepta los mensajes codificados para las versiones de HL7 2.x. El proyecto HAPI no está afiliado a la organización HL7, simplemente desarrollan software y librerías que se ajustan a la especificación dada por la organización de estándares sanitarios.

Las librerías que se han utilizado en el proyecto permiten realizar las siguientes operaciones en los códigos que las implementen:

- Crear mensajes.
- Enviar y recibir mensajes.
- Analizar mensajes.
- Modificar segmentos y estructuras.
- Leer mensajes desde ficheros.

En cualquier caso, para el desarrollo del proyecto sólo se han necesitado las librerías que permiten leer mensajes desde ficheros y las que permiten analizar mensajes codificados.

5.5 CONTROL DE SUCESOS Y ERRORES

5.5.1 Visualización de sucesos

Uno de los objetivos del proyecto es que todo lo que suceda en las aplicaciones que forman parte del sistema de mensajería se visualice en la pantalla, de forma que el administrador del sistema posea más información y pueda reaccionar de manera más rápida frente a los problemas.

La solución que se ha implementado consiste en crear un método en la clase VentaPrincipal.java, que muestre en el contenedor que ésta posee los sucesos producidos en tiempo de ejecución.

El método se llama *mostrarInformacionPantalla* y, a continuación, mostramos y explicamos el funcionamiento de este método:

```
public static void mostrarInformaciónPantalla(String mensajeAMostrar ){
        final VerFechaHora fechaHora = VerFechaHora.getInstancia();

        if (mensajeAMostrar == "Conexión establecida" && btnConectar.isEnabled()){

                textAreaEstadoActual.setText(textAreaEstadoActual.getText()+fechaHora.obtener
                Fecha()+mensajeAMostrar+ "\n");
                Registro.registarInformacion(fechaHora.obtenerFecha()+mensajeAMostrar);
                btnConectar.setEnabled(false);
                btnDesconectar.setEnabled(true);
                btnEnviar.setEnabled(true);

        } else if (mensajeAMostrar == "Se ha cerrado la conexión con el cliente"
        || mensajeAMostrar == "Se ha perdido la conexión"){

                textAreaEstadoActual.setText(textAreaEstadoActual.getText()+fechaHora.obtener
                Fecha()+mensajeAMostrar+ "\n");
                Registro.registarInformacion(fechaHora.obtenerFecha()+mensajeAMostrar);
                btnConectar.setEnabled(true);
                btnDesconectar.setEnabled(false);
                btnEnviar.setEnabled(false);

        } else if(mensajeAMostrar == ("Enviado correctamente el archivo "
        +servidor.getNombreArchivo())){

                textAreaEstadoActual.setText(textAreaEstadoActual.getText()+fechaHora.obtener
                Fecha()+mensajeAMostrar+ "\n");
                Registro.registarInformacion(fechaHora.obtenerFecha()+mensajeAMostrar);
        }else if(mensajeAMostrar == "Enviado correctamente el archivo "){

                servidor.conectar(txtIpDestino.getText(), txtPuerto.getText());

        }else if(mensajeAMostrar == "No se ha podido establecer conexion"
        || mensajeAMostrar == "Se han introducido caracteres erroneos en el Puerto"
        || mensajeAMostrar == "La dirección IP es incorrecta"){

                textAreaEstadoActual.setText(textAreaEstadoActual.getText()+fechaHora.obtener
                Fecha()+mensajeAMostrar+ "\n");
                Registro.registarInformacion(fechaHora.obtenerFecha()+mensajeAMostrar);
                btnConectar.setEnabled(true);
                btnDesconectar.setEnabled(false);
                btnEnviar.setEnabled(false);

        }else {

        textAreaEstadoActual.setText(textAreaEstadoActual.getText()+fechaHora.obtenerFecha()
        +mensajeAMostrar+ "\n");
        Registro.registarInformacion(fechaHora.obtenerFecha()+mensajeAMostrar);

        }
}
```

El funcionamiento consiste en que el método *mostrarInformaciónPantalla*, que está declarado como público y estático, puede ser llamado desde cualquier objeto

perteneciente a las clases Cliente, ClienteHilo, Servidor y Registro, pasándole éstas como valor de entrada al método una cadena de caracteres como vemos a continuación en la invocación del método desde la clase servidor:

```
>> VentanaPrincipal.mostrarInformaciónPantalla("Conexión establecida");
```

Con el argumento de entrada, el método ejecuta una serie de secuencias "if/else if" para realizar unas acciones específicas en la interfaz dependiendo del suceso que se haya ejecutado, de manera que desde el método *mostrarInformaciónPantalla* además de visualizar en pantalla el suceso, también puede cambiar el estado del resto de elementos de la interfaz gráfica o hacer llamadas a los métodos de la clase Registro para guardar la información en los logs de sucesos.

5.5.2 Control de Errores

El control de errores en las aplicaciones del sistema de mensajería se manejan mezclando las sentencias try/catch que se utilizan en Java para la captura de errores con llamadas al método *mostrarInformaciónPantalla* que hemos explicado en el apartado anterior. La solución resulta ser simple y efectiva y permite controlar todos los errores relacionados con la comunicación entre sockets y la lectura y escritura de ficheros.

```
try {
	conexion = new Socket(InetAddress.getByName(ip),Integer.parseInt(puerto));
	VentanaPrincipal.mostrarInformaciónPantalla("Conexión establecida");

}catch (NumberFormatException e){

	VentanaPrincipal.mostrarInformaciónPantalla("Se han introducido caracteres
	erroneos en el Puerto");

}catch (UnknownHostException e) {

	VentanaPrincipal.mostrarInformaciónPantalla("La dirección IP es
	incorrecta");

}catch (IOException e) {

	VentanaPrincipal.mostrarInformaciónPantalla("No se ha podido establecer
	conexion");
}
```

En el ejemplo que acabamos de ver se puede observar el funcionamiento de las sentencias "try/catch" para capturar errores. Esto permite controlar y visualizar en la aplicación, en todo momento, el estado de la conexión y los errores generados en la introducción de datos por pantalla.

En la sentencia try se escribe el código que es susceptible de lanzar una excepción por producirse un error, que en el ejemplo son tres casos:

- *new Socket(ip, puerto):* la clase socket necesita la dirección ip y un puerto del cliente al que se va a conectar para crear el socket de comunicación. Si no se consigue crear el socket por el motivo que sea, la aplicación lanza una excepción del tipo IOException. El código la captura y ejecuta el método *mostrarInformaciónPantalla* de la clase *VentanaPrincipal*, indicando con una cadena de caracteres el problema que se ha encontrado

- *InetAddress.getByName(ip):* este método necesita como argumento una dirección ip. Si en *VentanaPrincipal* insertamos una ip incorrecta o caracteres no válidos, la aplicación lanza una excepción del tipo UnknownHostException. El código la captura y ejecuta el método *mostrarInformaciónPantalla* de la clase *VentanaPrincipal*, indicando con una cadena de caracteres el problema que se ha encontrado

- *Integer.parseInt(puerto):* este caso es exactamente igual al anterior, con la única diferencia de que la excepción que se lanza es del tipo NumberFormatException

6 FASE DE PRUEBAS

6.1 INTRODUCCIÓN.

En este capítulo se explicaran las pruebas que se han hecho para comprobar el correcto funcionamiento del sistema desarrollado. Para aumentar la calidad del producto final, las pruebas se han dividido en unitarias y de integración.

6.2 PRUEBAS UNITARIAS.

Las pruebas unitarias se utilizan para comprobar el correcto funcionamiento de distintos módulos de código. Con estas pruebas se asegurará que cada uno de los módulos funcione correctamente por separado para después poder realizar las pruebas de integración.

6.2.1 Pruebas en la Interfaz gráfica de usuario

Se han realizado pruebas en cada uno de los campos de entrada de texto y en los botones que hay en las aplicaciones *EnviaHL7* y *RecibeHL7* para asegurarnos que el código no permite la introducción de valores erróneos dependiendo del tipo de dato que necesitamos:

- Campo de texto *Puerto* y *Puerto de destino*: Se han realizado, de forma satisfactoria, las siguientes pruebas para comprobar la correcta validación de los campos de entrada:
  - Prueba 1: el código debe controlar la inserción de cadenas de texto, avisando sobre el error y dejando la interfaz en el mismo estado anterior

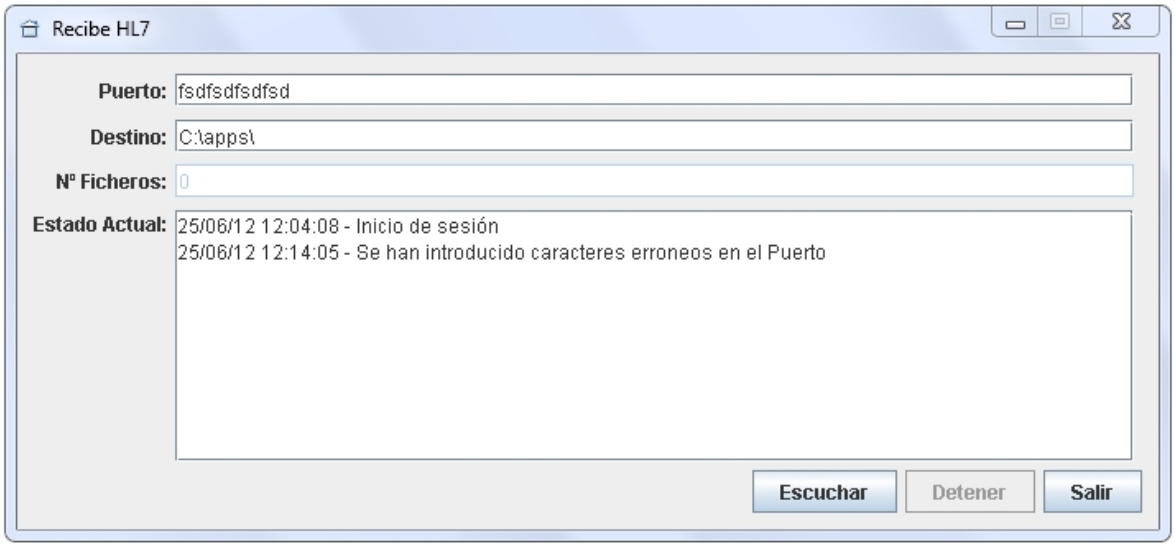

Figura 15 - Prueba 1

- Prueba 2: el código debe controlar la inserción de valores fuera del rango 0-65535, avisando sobre el error y dejando la interfaz en el mismo estado anterior

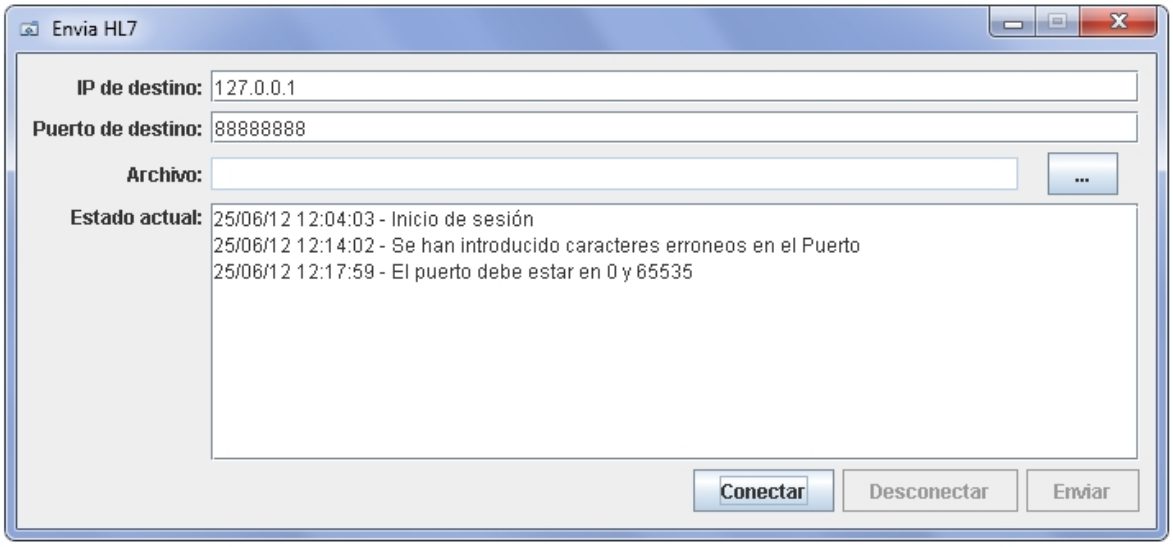

Figura 16 - Prueba 2

- Prueba 3: el código debe comprobar si el puerto seleccionado está ocupado o no, avisando en caso afirmativo y dejando la interfaz en el mismo estado anterior

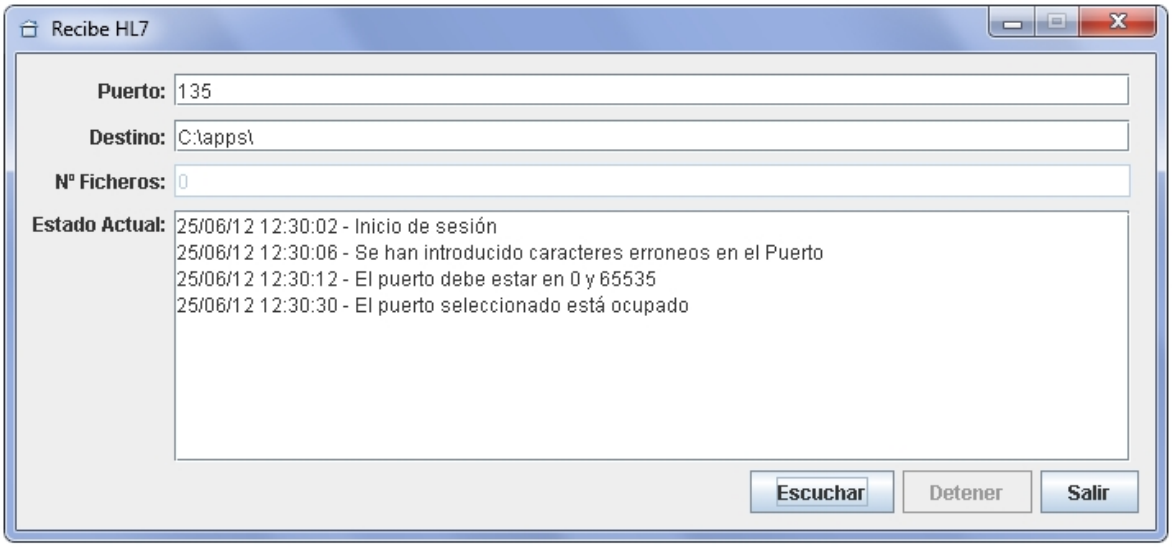

Figura 17 - Prueba 3

- Campo de texto *IP de destino*: Se han realizado, de forma satisfactoria, las siguientes pruebas para comprobar la correcta validación de los campos de entrada:

  - Prueba 4: el código debe controlar la inserción de cadenas de texto, avisando sobre el error y dejando la interfaz en el mismo estado anterior

  - Prueba 5: el código debe controlar la inserción de direcciones IP incorrectas, avisando sobre el error y dejando la interfaz en el mismo estado anterior

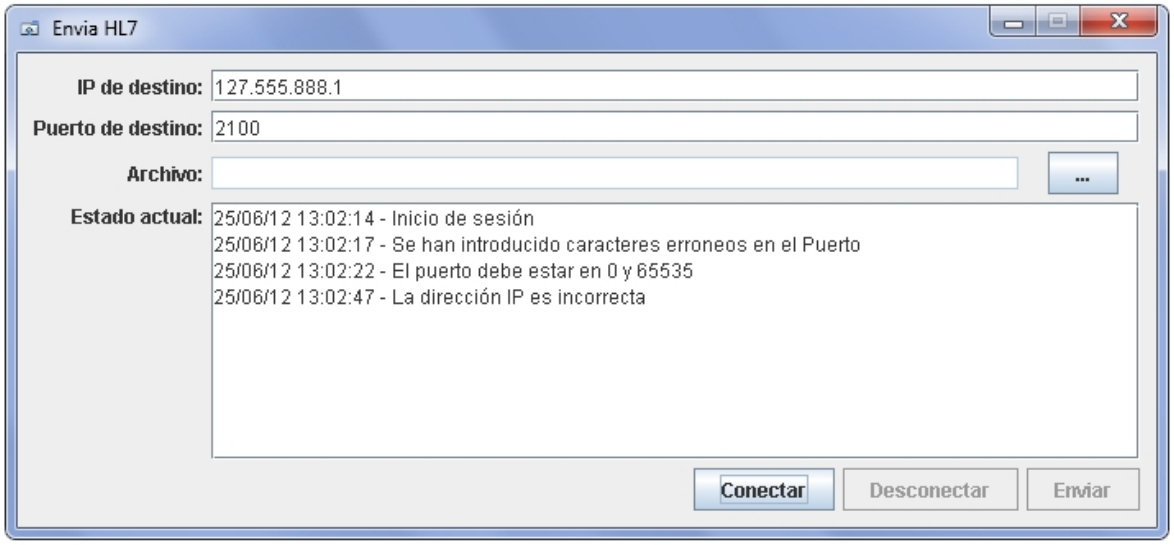

Figura 18 - Prueba 5

➲ Botón *Conectar* en *EnviaHL7*: Se han realizado, de forma satisfactoria, las siguientes pruebas para comprobar el correcto funcionamiento del botón.

- Prueba 6: El código debe controlar que el botón se deshabilita cuando se realiza la conexión con éxito
- Prueba 7: el código debe controlar que el botón se habilita si pulsamos el botón *Desconectar*
- Prueba 8: el código debe controlar que el botón se habilita si se pierde la conexión con la aplicación *RecibeHL7*

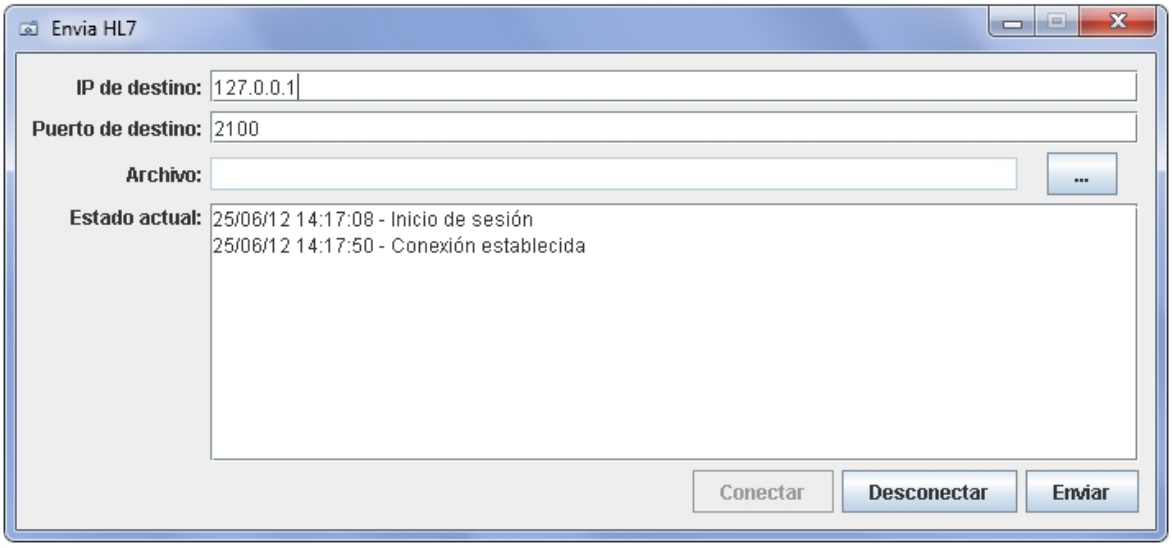

Figura 19 - Pruebas 6, 9 y 12

➲ Botón *Desconectar* en *EnviaHL7*: Se han realizado, de forma satisfactoria, las siguientes pruebas para comprobar el correcto funcionamiento del botón.

- Prueba 9: El código debe controlar que el botón se habilita cuando se realiza la conexión con éxito
- Prueba 10: el código debe controlar que el botón se deshabilita cuando se corta la conexión de forma voluntaria
- Prueba 11: el código debe controlar que el botón se deshabilita si se pierde la conexión con la aplicación *RecibeHL7*

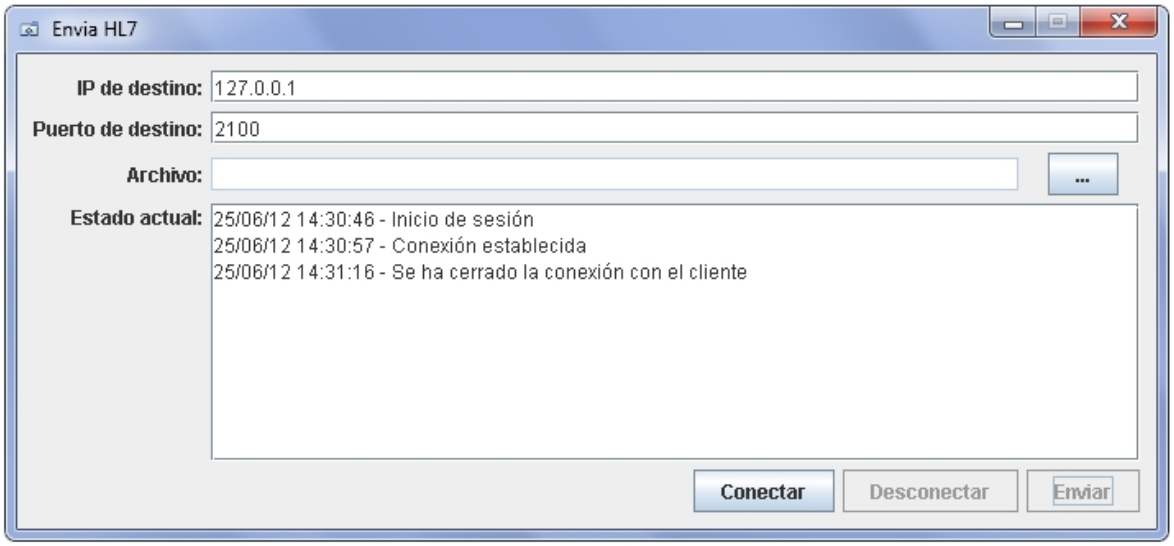

Figura 20 - Pruebas 7, 10 y 13

➲ Botón *Enviar* en *EnviaHL7*: Se han realizado, de forma satisfactoria, las siguientes pruebas para comprobar el correcto funcionamiento del botón:

- Prueba 12: El código debe controlar que el botón se habilita cuando se realiza la conexión con éxito
- Prueba 13: el código debe controlar que el botón se deshabilita si pulsamos el botón *Desconectar*
- Prueba 14: el código debe controlar que el botón se deshabilita si se pierde la conexión con la aplicación *RecibeHL7*
- Prueba 15: el código debe controlar que hayamos seleccionado un fichero cuando pulsamos *Enviar* y, en caso negativo, informar a través de la interfaz

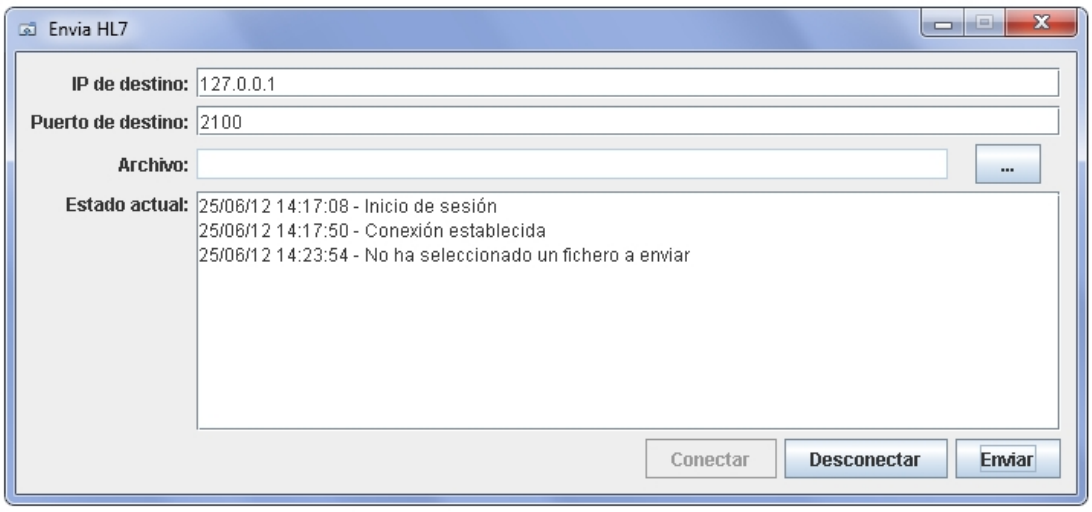

Figura 21 - Prueba 15

- Botón *Conectar* en *RecibeHL7*: Se han realizado, de forma satisfactoria, las siguientes pruebas para comprobar el correcto funcionamiento del botón:

  - Prueba 16: El código debe controlar que el botón se deshabilita cuando el socket se pone en modo escucha
  - Prueba 17: el código debe controlar que el botón se habilita si pulsamos el botón *Detener*

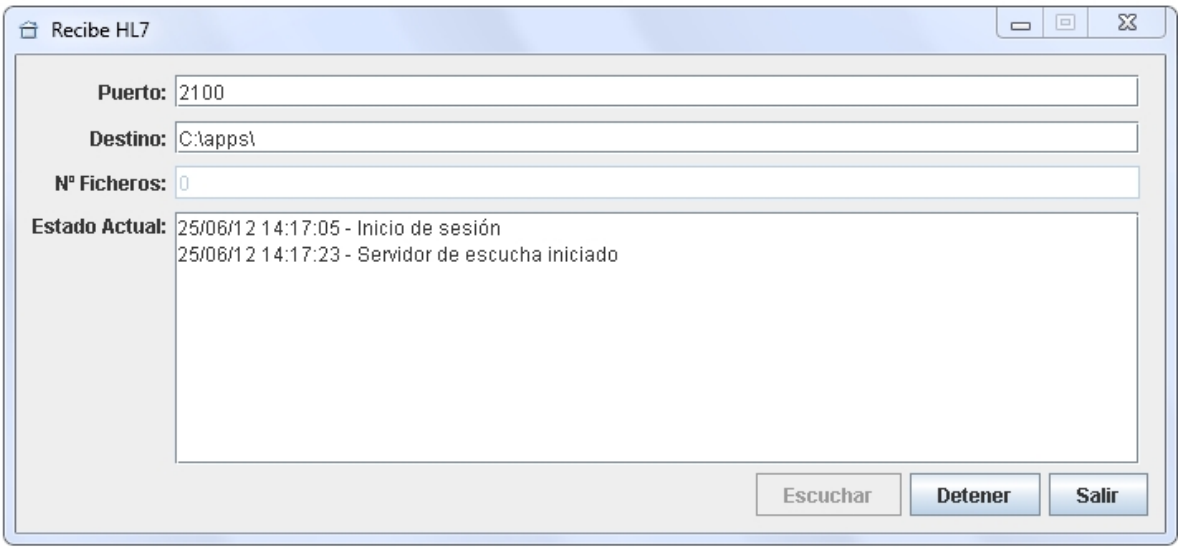

Figura 22 - Pruebas 16 y 18

- Botón *Detener* en *RecibeHL7*: Se han realizado, de forma satisfactoria, las siguientes pruebas para comprobar el correcto funcionamiento del botón.

  - Prueba 18: El código debe controlar que el botón se habilita cuando el socket se pone en modo escucha
  - Prueba 19: el código debe controlar que el botón se deshabilita si pulsamos el botón *Detener*

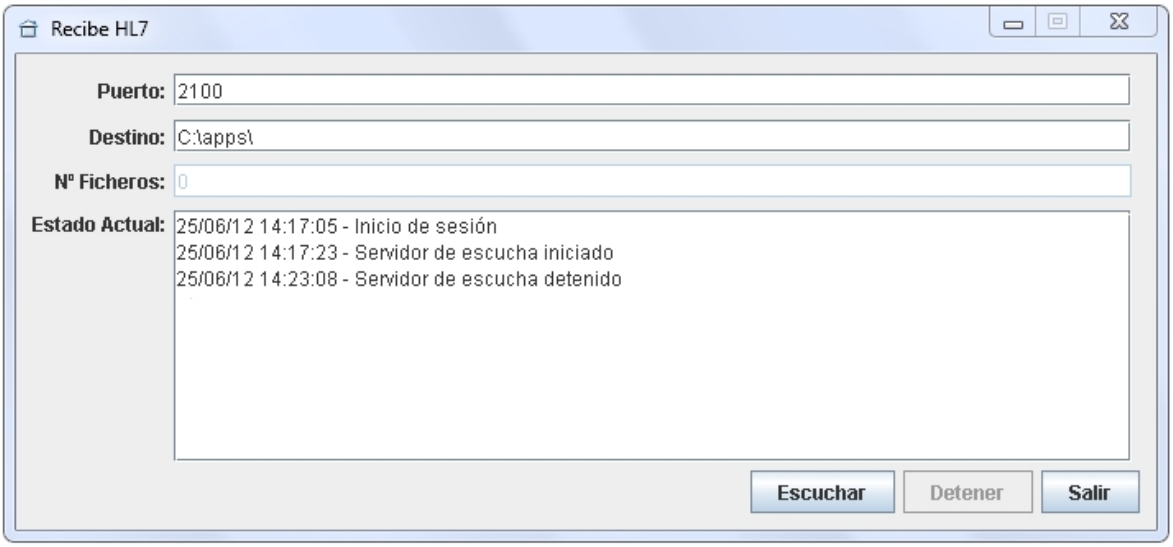

Figura 23 - Pruebas 17 y 19

➲ Botón *Salir* en *RecibeHL7*: Se han realizado, de forma satisfactoria, la siguiente prueba para comprobar el correcto funcionamiento del botón.

- Prueba 20: El código debe controlar que se cierra el socket de escucha y se cierra la aplicación cuando se pulsa el botón

6.3 PRUEBAS DE INTEGRACIÓN.

Las pruebas de integración se han realizado una vez finalizadas las pruebas unitarias, ya que así se ha podido comprobar el correcto funcionamiento de los diferentes módulos de código cuando estos interactúan juntos. Con estas pruebas se asegura que los procesos y clases que forman parte del sistema funcionan correctamente de forma conjunta.

6.3.1 Visor de sucesos

Los paneles *Estado Actual* de las aplicaciones *EnviaHL7* y *RecibeHL7,* son los centros de información de nuestro sistema de mensajería.

Están diseñados para mostrar toda la información acerca de los sucesos, errores o problemas que puedan surgir. Cada módulo de código independiente, cuando se ejecuta,

envía un mensaje que se mostrará en el visor de sucesos.

Se han necesitado muchas horas de pruebas para que los paneles Estado Actual mostrarán la información en pantalla de forma correcta, centrándose las pruebas sobre todo en tres aspectos:

- Ordenado: los mensajes siempre tienen que mostrarse en el mismo orden en el que se está ejecutando el código
- Obligatorio: No puede haber ningún suceso que no sea reflejado en el Visor de sucesos
- Fecha y hora: Todos los mensajes que se muestran tienen que contener la fecha y la hora en la que se ha generado el flujo de información

6.3.2 Conexión mediante sockets

Las pruebas realizadas en lo que respecta a la conexión mediante sockets entre las aplicaciones *EnviaHL7* y *RecibeHL7* se han centrado, en su mayoría, en cubrir los siguientes aspectos:

- La respuesta de los programas debe ser la correcta cuando se corta o pierde la conexión o cuando no se puede establecer
- Que el hilos de ejecución funcionen correctamente para que *RecibeHL7* pueda recibir de forma sincronizada archivos de múltiples aplicaciones *EnviaHL7*

6.3.3 Envío y recepción de mensajes.

El envío y recepción de ficheros se ha implementado con las clases y métodos de la biblioteca *java.io* utilizada para el manejo de los flujos de entrada y salida de información.

Así pues, las pruebas realizadas en este aspecto se han centrado en los siguientes aspectos:

- Comprobación de la creación de los canales que permitan la entrada y salida de datos.
- Comprobación de la existencia del fichero a enviar desde *EnviaHL7* y la creación del fichero en RecibeHL7.

6.3.4 Análisis del contenido de los ficheros HL7

El análisis del contenido de los ficheros HL7 desde el sistema de mensajería ha sido una de las fases de la implementación que más tiempo de dedicación ha necesitado, con lo cual la fase de pruebas dedicadas a este apartado también ha sido mayor en debido, en gran parte, al desconocimiento previo de la librería HAPI usada para el análisis de mensajería HL7.

Los puntos más importantes sobre los cuales se han realizado las pruebas son:

- Comprobar que el sistema reconoce que ficheros son incorrectos y cuáles no, evitando que el sistema reconozca ficheros incorrectos como válidos y al revés.
- Identificar y mostrar en pantalla los errores que contienen los archivos incorrectos.
- Guardar en un fichero log un lista de ficheros enviados y recibidos con errores, junto a la fecha y hora en la que se ha analizado el fichero y el error por el cual el fichero ha sido detectado como incorrecto.

7 CONCLUSIONES

7.1 ALCANCE DE OBJETIVOS.

Una vez finalizado el proyecto, es el momento de hacer balance de los objetivos iniciales conseguidos y las modificaciones y ampliaciones de objetivos que se han dado a lo largo del proyecto.

Los objetivos críticos y prioritarios del proyecto se han alcanzado de forma satisfactoria en el total de sus requisitos, dando como resultado un sistema de mensajería de ficheros HL7 robusto y portable gracias a su implementación en Java. Por otro lado, gracias al diseño realizado y a las pruebas a las que se ha sometido el software, el sistema está listo para utilizarse junto a *ekon salus*, el ERP sanitario que ofrece UNIT4 a sus clientes.

Pese a todo, de los objetivos iniciales del proyecto se descartó la inclusión de un sistema de alertas mediante widget en *ekon salus* y/o dispositivos móviles ya que, aunque a la empresa le pareció buena idea la creación de un sistema de alertas, no se llegó a un acuerdo sobre la forma y funcionamiento del mismo, con lo cual se creyó oportuno descartarlo como un objetivo del proyecto y dejar una vía abierta para un posible implementación en el futuro.

7.2 AMPLIACIONES Y MEJORAS.

Tras la finalización del proyecto se ha concluido que con los resultados obtenidos se podría estudiar la posibilidad de ampliar el sistema resultante con varias ampliaciones y mejoras.

Como se explicado en el punto anterior, a corto o medio plazo sería recomendable la implementación de un widget en *ekon salus* que informara a las partes interesadas sobre las peticiones que, al estar codificadas según el estándar HL7, contienen errores en el contenido de los mensajes. Se descarta, en un principio, que los usuarios del sistema que no sean administrador del mismo tengan acceso a todas las alertas

También se ha contemplado el volcar toda la información capturada por el sistema de mensajería y guardada en los logs de sucesos en la base de datos de *ekon salus,* para su posterior tratamiento y consulta desde el ERP sanitario.

7.3 DESVIACIONES RESPECTO LA PLANIFICACIÓN INICIAL.

En la planificación inicial se estimó una duración aproximada de 472 horas, estableciendo la fecha de fin del proyecto el día 25 del mes de abril del 2012. Respecto a esos plazos se una relación de riesgos identificados y gestionados a lo largo del proyecto, que se nombran a continuación:

- Realizar una planificación temporal del proyecto optimista
- Cambios en los objetivos iniciales del proyecto
- Modificación y ampliación de los requerimientos.

Estos riesgos han acabado sucediendo ya que las horas dedicadas en las distintas fases del proyecto han sido superiores a las planificadas en gran parte debido al estudio de las posibilidades de implementación de los estándares HL7, que, a su vez, ha repercutido en la necesidad de una mayor dedicación en la fase de codificación y pruebas.

Un riesgo que no se había identificado previamente, y que también ha afectado negativamente a la duración del proyecto, ha sido el desconocimiento por parte del alumno de los estándares HL7 y todo lo relacionado con la mensajería codificada y el análisis de la misma. La consecuencia principal ha sido la dedicación de horas extras por parte del alumno en el entendimiento de toda esa materia.

Como resultado de todo lo explicado, la duración del proyecto ha aumentado hasta un total de 544 horas y la fecha de fin se ha retrasado hasta el día 21 del mes de mayo del 2012.

En la figura 15 podemos apreciar en el diagrama de gannt como han afectado los cambios a la duración y finalización del proyecto:

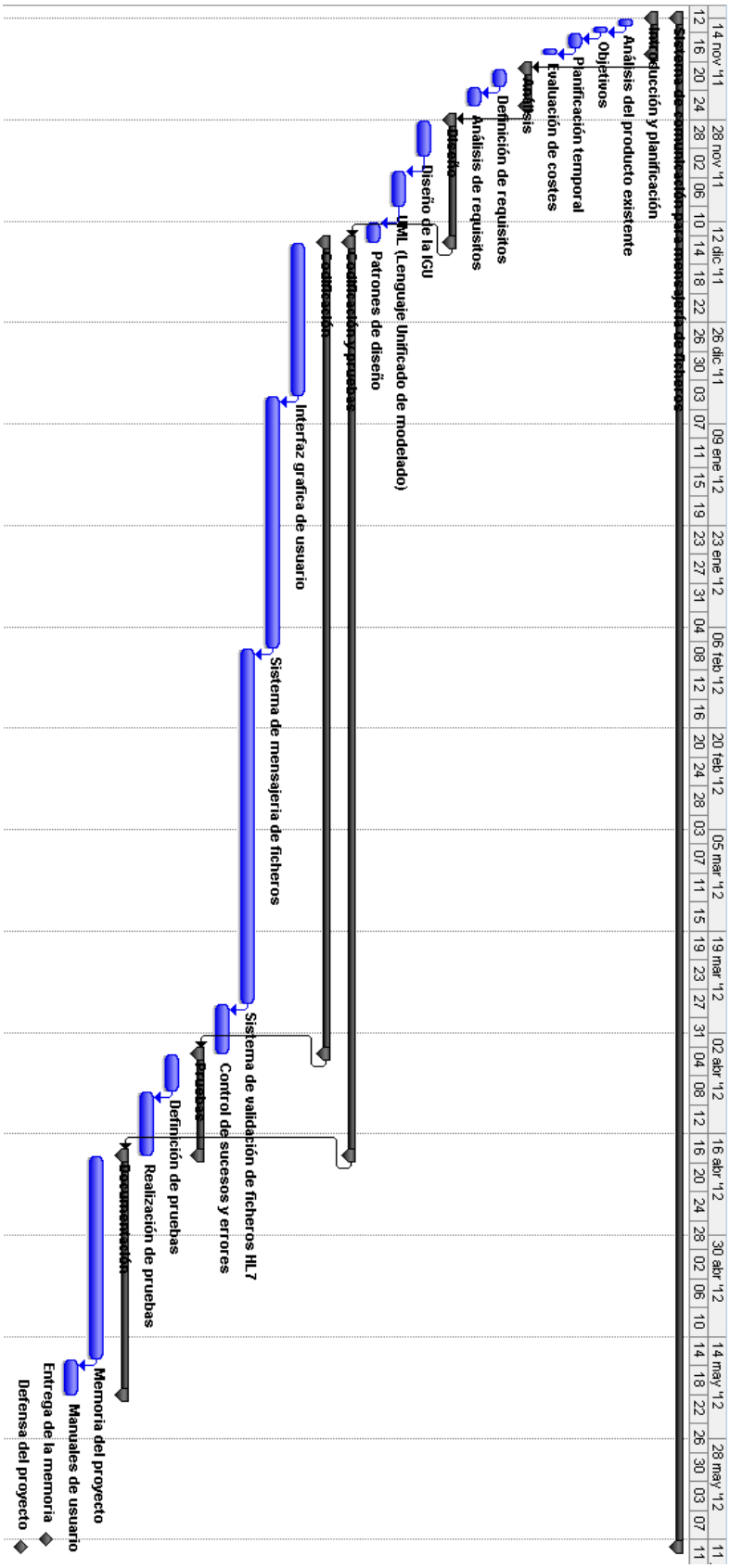

Figura 24 - Diagrama de Gantt real con desviamiento

7.4 VALORACIÓN PERSONAL.

La valoración que me merece la realización del proyecto en colaboración con la empresa UNIT4 es realmente muy positiva ya que me ha servido tanto para el ámbito educacional como profesional

A nivel educativo he tenido la oportunidad de desarrollar un proyecto que me gustara, poniendo en práctica la metodología y gestión de proyectos y pudiendo utilizar gran parte de los conocimientos adquiridos en la carrera, y de paso finalizar la carrera de forma satisfactoria.

En lo que respecta al ámbito profesional y laboral, he tenido la oportunidad de trabajar en UNIT4, una multinacional muy importante dentro del ámbito del desarrollo de software empresarial, con unas metodologías de trabajo actuales y un uso importante de las tecnologías de la información. Todo ello me ha reportado una gran adquisición de conocimientos y experiencia que seguramente me ayudarán en el futuro.

Lo más importante que he aprendido en este trabajo ha sido el desarrollar un proyecto de principio a fin, con una metodología de trabajo típica de las empresas que desarrollan software.

Al tratarse mi proyecto de un desarrollo sobre Java en el que no he tenido que interactuar con el producto al que va destinado no he podido ampliar mis bastante mis conocimientos en el desarrollo sobre la plataforma Java y la creación de interfaces graficas útiles para el usuario.

BIBLIOGRAFÍA

La bibliografía utilizada a lo largo del proyecto está dividida en documentos físicos y documentos online, siendo estos últimos los más utilizados.

- Contenido físico:
  - Eckel, Bruce. ***Piensa Java***. 3ra Edición. Lugar de edición:Prentice Hall, 2002. 1100 paginas. ISBN: 01-310-0287-2

  - Moldes,F. Javier. ***Java SE 6 – Guía práctica***. 2a edición. Lugar de edición: Anaya Multimedia, 2007. 304 paginas. I.S.B.N: 978-84-415-2288-6

- Contenido Online:
  - http://www.unit4.es/
    Página web de UNIT4, consultada para buscar información acerca de la empresa.
    Última fecha de consulta: 13/04/2012
  - http://hl7api.sourceforge.net/devbyexample.html
    Sitio con ayuda y ejemplos para implementar la librería HAPI en proyectos Java.
    Última fecha de consulta: 08/06/2012
  - https://developers.google.com/java-dev-tools/wbpro/tutorials/
    Sitio web con ayuda y tutorial de uso sobre el plugin WindowBuilder.
    Última fecha de consulta: 25/05/2012
  - http://www.hl7spain.org/
    Filial española de la empresa que crea estándares HL7. Consultada para obtener información sobre mensajería HL7.
    Última fecha de consulta: 27/04/2012
  - http://es.wikipedia.org
    Enciclopedia online, visitada para buscar información variada (HL7, patrones de diseño, Java, etc).
    Última fecha de consulta: 11/06/2012
  - http://intrabus.uab.cat/
    Intranet de la UAB usada para consultar memorias de cursos anteriores de las Ingenierías Técnicas Informáticas de Gestión y Sistemas
    Última fecha de consulta: 09/01/2012

ANEXO A - CONTENIDO DEL CD.

El CD que se adjunta con la memoria contiene los siguientes recursos:

- Memoria del proyecto en formato PDF
- Repositorio con todo el código del proyecto
- Aplicaciones del proyecto empaquetadas en formato JAR
- Exposiciones y documentos realizados en UNIT4
- Ejemplos de fichero en formato HL7 para realizar las pruebas con los ejecutables del proyecto.

_____

Firmado: Lorenzo Gama Ramos

Sabadell, **Junio** del **2012**