Intranet Clínica

Jaime Buera Agraz
ETIG / ETIS

Consultor: Jose Juan Rodriguez

Fecha: 10/01/2012

Agradecimientos

Realizar este proyecto me ha supuesto un gran esfuerzo, pero no solo realizar en si el proyecto, si no el conseguir llegar hasta aquí. Es difícil cuantificar el esfuerzo de tantos años de trabajo, pero la verdad es que el sacar esto mientras uno trabaja se hace realmente duro.

Por eso quiero agradecer a todos los amigos que han estado ahí en los momentos difíciles dando su apoyo incondicional, su tiempo para escuchar y sus ánimos para levantar la moral.

Gracias también a toda mi familia, en especial a mi madre Josefina y a mi novia Laura, que son las personas que más fuerza me han dado para seguir hacia delante en los momentos más difíciles.

Y por último agradecer a mi tutor José Juan por ayudarme y guiarme durante todo el proyecto.

A todos Gracias.

Resumen

La idea principal de este proyecto nace a raiz de unas necesidades marcadas por los facultativos en su entorno de trabajo.
Dichas necesidades surgen por la desaparicion del papel y la posisbilidad de explotar las nuevas tecnologías en el ambito médico. El poder consultar la historia clinica de un paciente en cuestión de segundos hace que la intranet clínica sea un herramienta de consulta muy potente, a la vez de sencilla y dinámica.

La intranet clínica es diseñada de forma que el facultativo puede tener toda la información necesaria para poder diagnosticar o por lo menos ayudar a un diagnóstico con toda la información que pueda consultar.

El proyecto en si solo será la base o el comienzo de una estructura que irá creciendo y mejorandose a medida que los usuarios manifiesten sus necesidades, así como la introducción de nuevos modulos agregando nueva información a la historia clínica del paciente.

Así pues en esta primera fase veremos los 5 capítulos más importantes que son la história, radiología, laboratorio, anatomía patológica y los informes médicos. Toda esta información que se muestra del paciente viene precedida por un buscador de pacientes efeicaz y eficiente en el entorno de trabajo del facultativo.

Indice

Indice de figuras

1. Introducción

1.1 Justificación y aportación

La intranet clínica representa una herramienta de consulta, es decir, servirá para dar apoyo al facultativo a la hora de poder tomar decisiones de carácter médico.

La necesidad de tener una intranet clínica surge principalmente para tener una respuesta rápida ante el paciente a la hora de visitar y diagnosticar. Esta necesidad conlleva la sustitución de la historia en papel a una historia electrónica, con lo cual obtenemos una herramienta de fácil manejo y consulta instantánea.

1.2 Objetivo del proyecto

El objetivo principal del proyecto es conseguir una herramienta eficaz y eficiente capaz de servir toda la información necesaria para realizar o ayudar al usuario (en este caso el médico) a tomar decisiones de caracter médico.
La herramienta será de apoyo para diagnosticar y favorecer los tiempos de demora a la hora de realizar las consultas, emergencias, etc... Por lo tanto, podemos ofrecer en muchos aspectos una mejora sustancial en todos los cuadros clínicos.

1.3 Método utilizado

Para conseguir una estimación y una planificación ajustada utilizaremos el ciclo de vida en cascada para desarrollar el sistema de intranet clínica. La utilización del ciclo de vida en cascada nos proporcionará la capacidad de análisis para el desarrollo de la herramienta.

En el llamado modelo en cascada, es el enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.

Un ejemplo de una metodología de desarrollo en cascada es:

1. Análisis de requisitos
2. Diseño del Sistema
3. Diseño del Programa
4. Codificación
5. Pruebas
6. Implantación
7. Mantenimiento

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costes del desarrollo. La palabra *cascada* sugiere, mediante la metáfora de la fuerza de la gravedad, el esfuerzo necesario para introducir un cambio en las fases más avanzadas de un proyecto.

1.4 Planificación del proyecto

Para la planificación utilizaremos la herramienta Microsoft Project planificaremos la estimación del tiempo con las tareas a realizar desde el inicio. El desarrollo será en función de los 3 meses tendiendo en cuenta el esfuerzo en cada etapa del desarrollo.

En caso de que la planificación sea con un desarrollo complicado se utilizará la herramienta COCOMO II para estimar el esfuerzo y poder distribuir así los recursos de forma óptima.

Para la utilización de la herramienta será necesario utilizar los navegadores IE7-8 o firefox 1.2 (en adelante).

La herramienta correrá bajo Apache 2.0 para servir páginas estáticas y tomcat 5.5 para servir las páginas dinámicas (servlets).

Utilizaremos java JDK 1.6 en adelante para poder generar o modificar código para la puesta en marcha.

La BB.DD será MySQL5 pero en función de la carga o robustez de los datos se tendrá en cuenta una migración a Informix u Oracle.
Las páginas serán optimizadas y servidas en JSP, siguiendo el patrón de struts 2.

Consideramos que las actividades necesarias para llevar a cabo el proyecto son las siguientes:

Descomposición estructural de actividades		
Código de actividad	*Actividades nivel 1*	*Actividades nivel 2*
01	Inicio del proyecto	
02	Gestión del proyecto	
03	Construcción del software	
03.01		Análisis
03.02		Diseño
03.03		Programación y Pruebas

		unitarias
03.04		Pruebas
04	Formación de usuarios	
05	Puesta en producción	
06	Final del proyecto	

Descomposición en actividades. Figura 1

La actividad 03.03 se agrupa mediante dos actividades y que más adelante se explicará su desarrollo para facilitar el desarrollo de las mismas.

Teniendo definidas las actividades que se van a realizar en el proyecto, vamos a desglosar el tiempo estimado para cada actividad en jornadas de trabajo. La estimación máxima es de 3 meses (aprox) desde el inicio, por lo tanto, procedemos a desglosar las etapas de la siguiente manera:

Duración estimada de las actividades		
Código de actividad	*Actividad*	*Jornadas*
01	Inicio del proyecto	--
02	Gestión del proyecto	5
03	Construcción del software	--
03.01	Análisis	10
03.02	Diseño	15
03.03	Programación y Pruebas unitarias	35
03.04	Pruebas	9
04	Formación de usuarios	1
05	Puesta en producción	1
06	Final del proyecto	--
TOTAL		**76**

Duración proyecto.Figura 2.

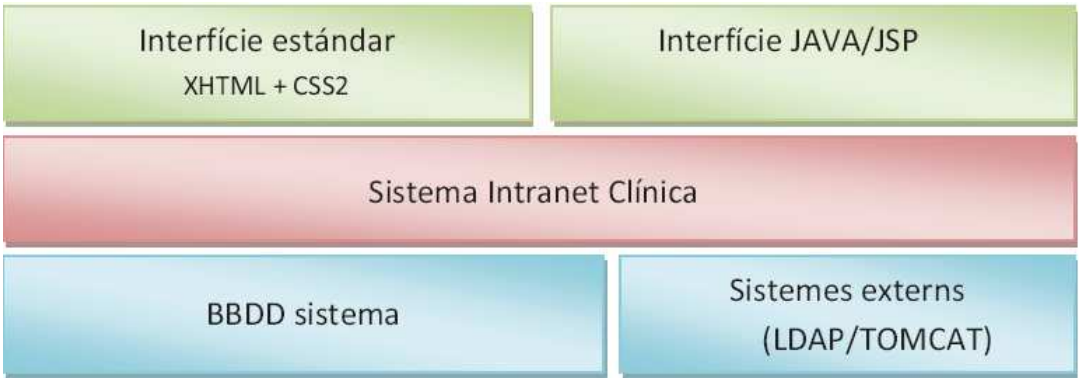
NOTA: La actividad 03.03 su duración será de 35 jornadas, pero el desarrollo de la misma será llevada a cabo mediante agrupaciones de entre 7 y 10 días. Con esto conseguimos agrupar las jornadas en sesiones para mantener la fiabilidad de la actividad.
Utilizaremos Microsoft Project para hacer la planificación temporal del proyecto teniendo en cuenta las precedencias entre las actividades establecidas.

		Nombre de tarea	Duration	Start	Finish	Predecessors	Resource Names
1		**Inicio proyecto**	**1 day**	**Mon 03/10/11**	**Mon 03/10/11**		
2		**Gestión proyecto**	**6 days**	**Tue 04/10/11**	**Tue 11/10/11**		**gestión**
3		Administración aplicación	3 days	Tue 04/10/11	Thu 06/10/11		gestión
4		Consulta necesidades	2 days	Fri 07/10/11	Mon 10/10/11	3	gestión
5		Seguimiento	1 day	Tue 11/10/11	Tue 11/10/11	4	gestión
6		**Construcción software**	**68 days**	**Wed 12/10/11**	**Fri 13/01/12**		
7		Análisis	10 days	Wed 12/10/11	Tue 25/10/11		analista
8		Diseño	15 days	Wed 26/10/11	Tue 15/11/11	7	analisis programador
9		Porgramación y pruebas	35 days	Wed 16/11/11	Tue 03/01/12	8	analisis programador
10		Pruebas	8 days	Wed 04/01/12	Fri 13/01/12	9	analista
11		**Formación usuario**	**2 days**	**Mon 16/01/12**	**Tue 17/01/12**		
12		Usuarios	2 days	Mon 16/01/12	Tue 17/01/12		analista
13		**Puesta en producción**	**1 day**	**Wed 18/01/12**	**Wed 18/01/12**		**Tecnico de sistemas**
14		**Fin proyecto**	**1 day**	**Thu 19/01/12**	**Thu 19/01/12**	13	

Planificación temporal. Figura 3.

1.5 Productos obtenidos.

Para tener una visión de conjunto clara respecto de qué hacer para desarrollar el sistema, a continuación presentamos algunos elementos que nos permitirán tomar contacto con la magnitud del proyecto. Para ello, utilizaremos herramientas propias de la ingeniería del software, y mostraremos cómo será la base de datos que utilizará el sistema, cuál será el diagrama de contexto y como habrá que configurar los perfiles de seguridad del nuevo sistema.
La estructura de la aplicación será en base a la tecnología JAVA/JSP.
El esquema que define la simplicidad del sistema respecto al usuario final es:



Esquema sistema. Figura 4.

1.6 Descripción de capitulos

Los capitulos que se han realizado forman parte de todo el proceso que ha ocasionado el desarrollo de la aplicación, es decir, se ha detallado cualquier configuración o instalación que forma que se pueda seguir en cualquier punto en que se encuentre de forma técnica o presencial.
La descripción de los capitulos es la siguiente:

Capítulo1: La introducción forma parte en todos los proyectos para describir brevemente en que consiste.

Capítulo2: En todo proyecto es necesario identificar las necesidades y que solución o como podemos obtener en definitiva un desarrollo.
Las necesidades dan lugar a obtener todo la estructura necesaria para poder desarrollar y obtener el producto final.

Capítulo3: La estructua del hardware será un paso importante en la aplicación. Si una aplicación no tiene una base correcta y bien configurada, los problemas de rendimiento pueden ser graves y ocasionar un abandono de la aplicación. Por lo tanto, en este capítulo definiremos una estructura de hardware y una configuración para que el rendimiento sea el correcto.

Capítulo4: Todo hardware da una base a un sistema operativo para llevar o desplegar un entorno de trabajo. En este capítulo describimos brevemente el S.O seleccionado para la aplicación.

Capítulo5: Las aplicaciones necesarias para poder desplegar la aplicación y que funcione sin problemas, vendrás definidas en este capítulo. Es importante tener claro desde un principio cuales son las necesidades básicas, y a raiz de obtener una necesidad seleccionamos que software puede ser el adecuado para su implantación.

Capítulo6: Todo hace referencia al anterior capítulo, ya que trata sobre la configuración del software seleccionado para el desarrollo y puesta en producción de la aplicación.

Capítulo7: En este capítulo pasamos a definir la estructura interna, es decir, ver la composición de las capas, configuración de archivos de sistema, rutas e información de las classes. Con esto conseguimos una descripción adecuada de como funciona internamente la aplicación a nivel de configuración.

Capítulo8: La validación de los usuarios es una parte importante en lo que a la seguridad se refiere. Cada usuario dado de alta necesitará un permiso que será el que dicte si puede acceder o no a la aplicación y que contenido puede ver. Se definene una clase de roles para obtener restricciones a nivel de aplicación.

Capitulo9: La aplicación lleva toda una serie de tablas en MySQL para poder guardar toda la información sobre la historia del paciente y el rastro de la navegabilidad.
Estas tablas serán las que alimenten a la plaicación para mostrar los datos de consulta.

Capitulo10: Muetsra en entorno de trabajo para el desarrollo de la aplicación, en definitiva, las herramientas usadas para realizar el trabajo.

Capitulo 2. Definición de las necesidades

2.1 Estudio inicial

La intranet clínica representa una herramienta de consulta, es decir, servirá para dar apoyo al facultativo a la hora de poder tomar decisiones de carácter médico.

La necesidad de tener una intranet clínica surge principalmente para tener una respuesta rápida ante el paciente a la hora de visitar y diagnosticar. Esta necesidad conlleva la sustitución de la historia en papel a una historia electrónica, con lo cual obtenemos una herramienta de fácil manejo y consulta instantánea.

Si bien existe una necesidad para consultar el historial médico, hay que tener en cuenta que existe una ley orgánica sobre la protección de datos, por lo tanto, todo acceso se restringirá únicamente al profesional, quedando guardada la traza de toda consulta efectuada. Esta traza será almacenada y disponible como mínimo 2 años como dicta la ley.

2.2 Estructura global

El desarrollo de la herramienta mostrará parte de la historia clínica. Se creará la base en función de las necesidades más urgentes para tomar decisiones de carácter médico, quedando a disposición para ir creando nuevos desarrollos e ir completando la historia hasta mostrar en su totalidad cualquier dato clínico relacionado con el paciente.

En la estructura inicial podremos encontrar 3 pantallas:

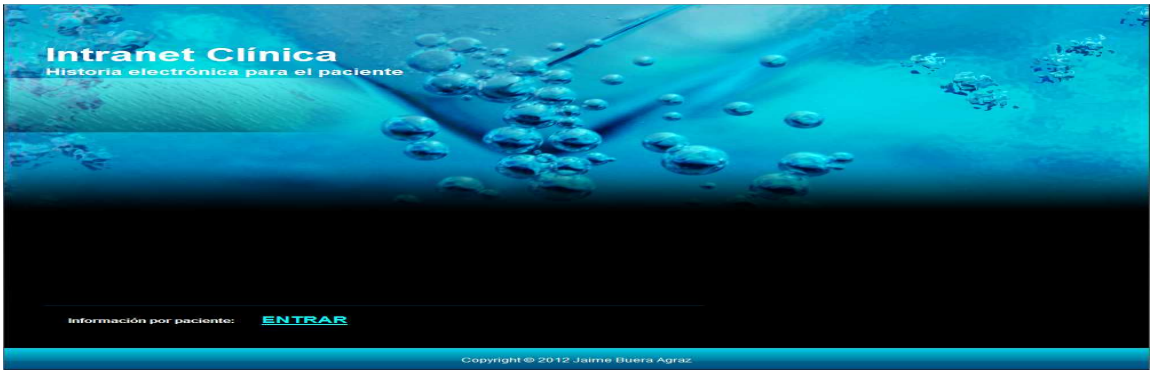
- La primera pantalla será el buscador del paciente, pudiendo realizar la búsqueda por número de historia, nombre, apellidos, DNI y número de la seguridad social.
- La segunda pantalla será la muestra de la búsqueda, donde aparecerá los datos del paciente con nombre y apellidos, dirección, teléfono y número de historia.
- La tercera pantalla será la que muestre el historial clínico en forma de carpeta. La carpeta llevará adosada una serie de pestañas indicando la naturaleza de las pruebas. Las pestañas serán:

- Historial
- Radiología
- Laboratorio
- Informes
- Anatomía Patológica

Toda la estructura de pestañas será ampliable con nuevos desarrollos para el futuro.

2.3 Funcionamiento aplicación

La pagina principal es una pantalla la cual utilizaremos para realizar búsqueda de pacientes. La pantalla constará de un formulario con los campos nombre, apellidos, DNI, número de historia y número de tarjeta sanitaria.



Página inicial. Figura5

Buscador. Figura 6

Una vez que se lanza la búsqueda accederemos a la pantalla donde aparecerán los pacientes que coincidan con la búsqueda. Se seleccionará el interesado mediante un enlace el cual nos llevará a la historia del paciente.

Busqueda de paciente. Figura 7

La historia del paciente consta de 5 pestañas, nombradas anteriormente. Cada pestaña es accesible desde cualquier pestaña, por lo que la navegabilidad y comodidad a la hora de seleccionar información es bastante accesible.

Todas las apestañas muestran los episodios correspondientes a cada servicio, es decir, la pestaña radiología mostrara los informes y radiografías realizadas, la pestaña laboratorio las analíticas realizadas, la pestaña informes mostrara todos los informes realizados, etc…

Pestaña historial. Figura 8

Pestaña radiología. Figura 9

Pestaña Anatomía patológica. Figura 10

Pestaña Laboratorio. Figura 11

Pestaña Informes. Figura 12

Por lo tanto, en cuestión de minutos podemos acceder a toda la historia de un paciente, evitando así papeleos y desorden administrativo.

Todos los usuarios deberán estar dados de alta para poder utilizar la aplicación. Se asignarán roles para poder fraccionar la información de modo que no pueda ser visualizada para aquel especialista que no deba.

En todas las páginas existirá una cabecera que mostrará información del usuario, realizar conexiones y desconexiones y mejorar la navegabilidad para volver a realizar búsquedas.

Capitulo 3. Estructura del Hardware.

El servidor es una parte importante del proyecto, ya que en su estructura y componente se aloja el software que dará servicio a toda la estructura. Si bien inicialmente la carga de usuarios será baja, con la implementación y desarrollo de nuevas aplicaciones web esta demanda irá creciendo.
Por lo tanto, necesitamos una máquina la cual pueda soportar cargas puntuales bastantes elevadas.
En el siguiente punto describimos un servidor tipo que se ajusta a las necesidades y que garantiza un servicio óptimo.

3.1 Componentes

El desglose del servidor se realiza teniendo en cuenta que el retardo en mostrar la información puede suponer para el usuario que abandone la herramienta y el proyecto sea un fracaso por culpa del hardware.

Principalmente, el software que más necesidad tiene de utilizar hardware óptimo es Tomcat, Java y MySQL.

Partiendo de estas necesidades construimos el servidor con los siguientes componentes:

- **Procesador**: 2x AMD Opteron 6128, 8C, 2.0GHz, 8x512K L2/12M L3 Cache, 80W ACP, DDR3-1333MHz.
- **Placa Base**: PowerEdge R715 Rack Chassis for Up to 6x 2.5" HDDs
- **Memoria**: 8GB Memory for 2 CPUs, DDR3, 1333MHz (8x1GB Single Ranked RDIMMs)
- **Raid:** C4 8/12HD - R0 for PERC H700, 1-8 or 1-12 SAS/SATA/SSD HDDs based on chassis
- **Controladora RAID:** PERC H200 Integrated RAID Controller
- **Disco**: 3 discos de 600GB, SAS 6Gbps, 2.5-in, 10K RPM Hard Drive (Hot Plug)
- **Tarjeta red:** Intel® Gigabit ET Dual Port Server Adapter

- **Fuente alimentación:** Power Supply, Redundant (2 PSU), 750W

Los demás componentes no son nombrados ya que carecen de valor en el rendimiento del servidor.

Capitulo 4. Sistema operativo

El sistema operativo influye sensiblemente a la hora de proporcionar un servicio de calidad. Es importante que el sistema operativo no se degrade con facilidad y que además su rendimiento sea bueno en comunión con el software que utilicemos.

El sistema operativo que se utilizará será Debian, ya que aparte de ser un software libre tiene una ventaja y es que al no instalar un entorno gráfico el rendimiento del S.O es bastante elevado y el consumo de recursos es infinitamente más bajo que cualquier S.O con entorno gráfico. A parte de todo esto podemos encontrar casi todo el software de aplicaciones para este sistema operativo en modo stable, por lo que da tranquilidad a la hora de efectuar cualquier instalación.

El único requisito imprescindible es tener actualizado el repositorio de librerías para poder actualizar en cualquier momento cualquier paquete necesario para el correcto funcionamiento de servidor.

Capitulo5. Software del núcleo base

El software que utilizaremos para que nuestra aplicación funcione correctamente y el rendimiento sea óptimo, será el que describimos a continuación:

5.1 Tomcat 5.5

Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

Tomcat es un servidor Web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones. Incluye el compilador Jasper, que compila JSPs

convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el Servidor Apache Web
Tomcat puede funcionar como servidor Web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor Web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Resumiendo, Tomcat será el encargado de servirnos todas las páginas dinámicas provenientes de los JSPs.

5.2 MySql 5.0

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.

Por un lado se ofrece bajo la GNU para cualquier uso compatible con esta licencia. Está desarrollado en su mayor parte en ANSI C

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado.

MySQL será el encargado de almacenar toda la información que necesitemos que sea almacenada para posteriormente ser mostrada desde la Web.

5.3 Apache 2.0

El servidor HTTP Apache es un servidor Web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o

explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

Por lo tanto, Apache será bastante útil para utilizarlo como servidor http o HTTPS si queremos implementar el protocolo seguro bajo certificado y obtener así una seguridad casi absoluta con el cifrado del HTTPS.

5.4 Java JDK 6.0

Java Development Kit o (JDK), es un software que provee herramientas de desarrollo para la creación de programas en java.

Con el JDK obtenemos las herramientas necesarias para poder desarrollar servlets, JSPs, etc… y poder así mostrar el contenido de la información que extraeremos de las bases de datos.

6. Análisis

Vamos a analizar la aplicación desde el punto de vista del esquema de actores y sobre el diagrama de casos de uso.
También aportaremos una breve descripción del esquema para detallar su funcionamiento.

6.1 Actores

El esquema principal de la aplicación viene dada por dos actores, uno que será el usuario y por otro lado el administrador. Hay que destacar que un usuario puede o no tener registro de entrada.

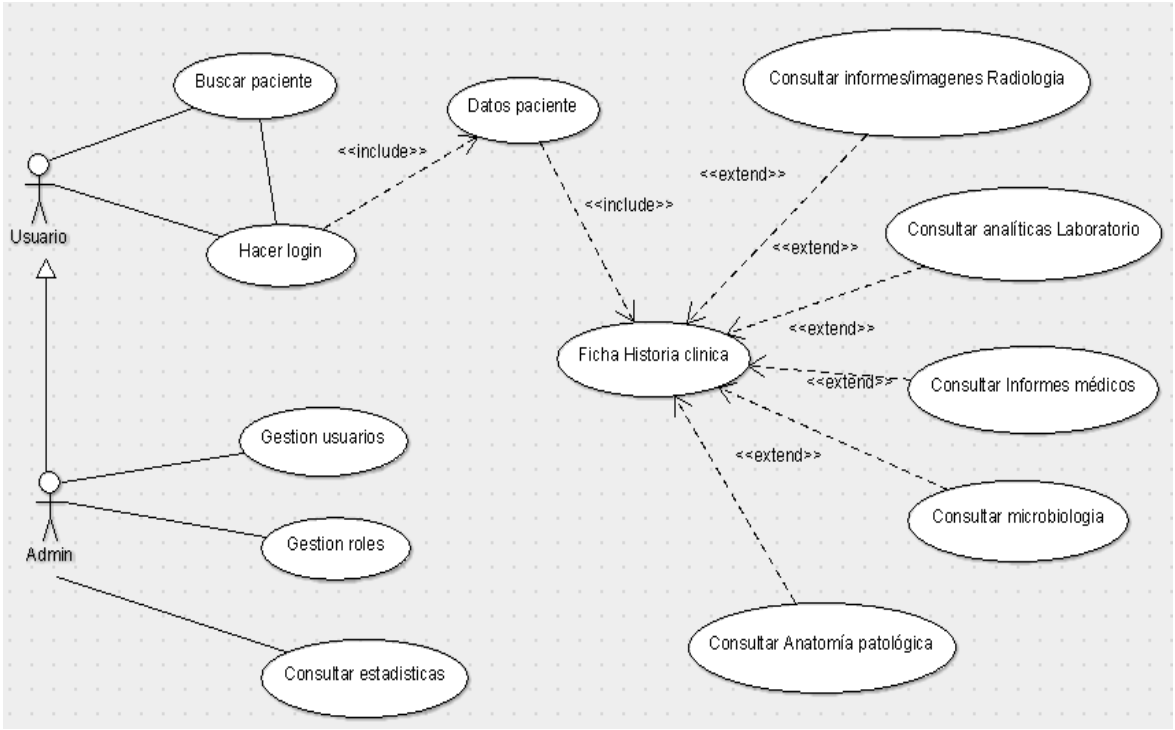
Usuario NO registrado: El usuario no registrado podrá acceder a la pantalla principal, pudiendo así mostrar los datos de búsqueda.

Usuario Registrado: Será el usuario que una vez efectuada una búsqueda acceda a la pantalla donde accederá a la información del paciente o a la información clínica.

Administrador: El administrador se encargará de proporcionar los registros a los usuarios alta/baja.
Podrá insertar/eliminar/modificar los roles de acceso a las distintas pestañas.
Gestionará los datos sobre estadísticas de acceso, rastro, contenido, etc…

6.2 Diagrama de casos de uso

El diagrama de casos de uso es el siguiente:



Casos de uso. Figura 13

6.3 Descripción funcionamiento del diagrama de casos de uso

Login

Tenemos al usuario, que podrá acceder a la aplicación por dos vías, una haciendo login directamente o entrando a la pantalla de búsqueda de paciente y será logueado en cuanto realice la búsqueda. Si el usuario intenta acceder a la búsqueda y la identificación es incorrecta obtendrá un mensaje de error y de nuevo optará a realizar un nuevo intento.

Búsqueda de paciente

La búsqueda de paciente consiste en un formulario que recoge los campos de nombre, apellidos, dni, número de historia del paciente.
La búsqueda hace un cribado dependiendo del campo a utilizar, es decir, podemos tener un listado de pacientes en función del campo seleccionado.

Una vez realizada la búsqueda accederá a la pantalla de datos paciente, que será la pantalla donde aparecerán los pacientes que coincidan con los criterios de búsqueda.

Historia clínica

Cuando el paciente ha sido localizado, accederemos a los datos clínicos mediante un enlace colocado sobre el número de historia, el cual nos llevará a la pantalla de historia clínica.
La pestaña historial es la primera pestaña en mostrarse. En esta pestaña obtenemos todos los episodios del paciente que se han ido generando a los largo de su vida.

Radiología

Es la pestaña donde obtendremos los datos radiológicos del paciente. Nos mostrará información acerca de la imagen radiológica y del informe.

Laboratorio

La pestaña laboratorio muestra todas las pruebas de análisis clínicos mostrando la naturaleza de la petición. También obtenemos la descripción si proviene de una urgencia o bajo demanda.

Anatomía Patológica.

Al igual que la anteriores pestañas está es la encargada de mostrar las pruebas que se realizan en la sección de anatomía patológica. Al igual que el laboratorio mostrará información de la naturaleza de la prueba así como los resultados de las mismas.

Informes médicos

La pestaña de informes médicos mostrará información acerca de los informes que se realizan en las diversas consultas del hospital.

Microbiología.

Es el mismo caso que el laboratorio salvo que se mostraran las petición de microbiología.

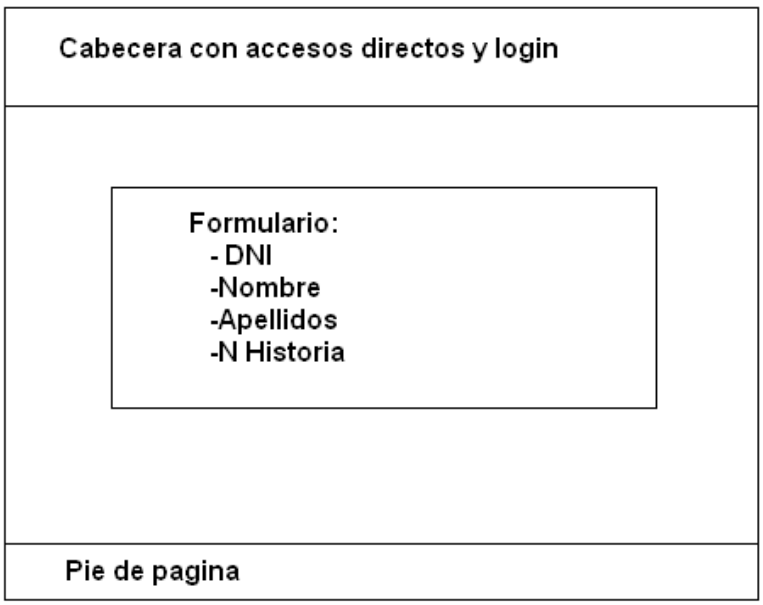
Todas las pestañas son accesibles desde historial y desde cualquier pestaña.

Todos los usuario estarán gestionados por medio del administrador, quien será el que gestione las altas/bajas (gestión usuarios), gestionará los permisos para acceder a la aplicación (gestión roles) y será el encargado de poder consultar las estadísticas de consulta como accesos a pestañas, rastro de la actividad, gestor de consultas, etc...

NOTA: A modo de navegabilidad, la aplicación llevará una cabecera donde podremos acceder a distintas pantallas de la aplicación y a poder hacer login o desloguearse.

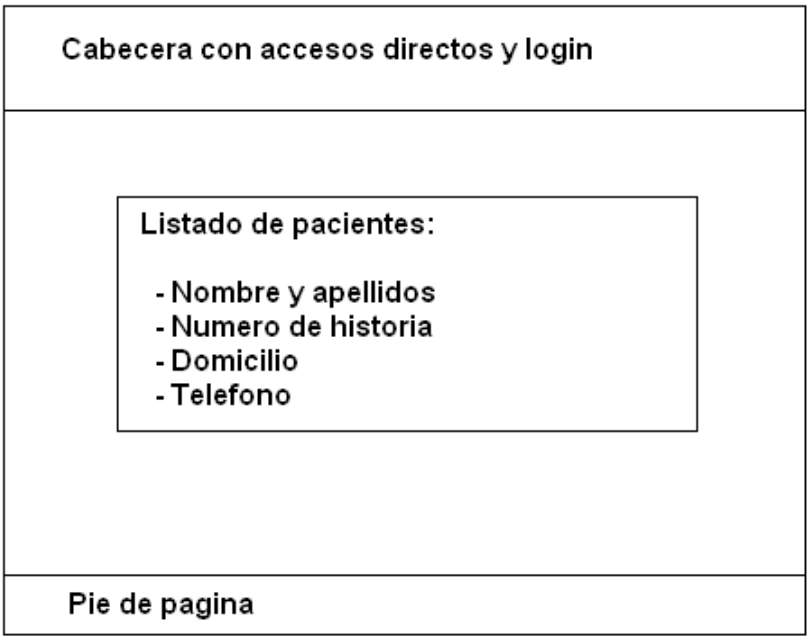
6.4 Entorno visual aplicación.

La primera pantalla es el formulario de búsqueda de paciente:



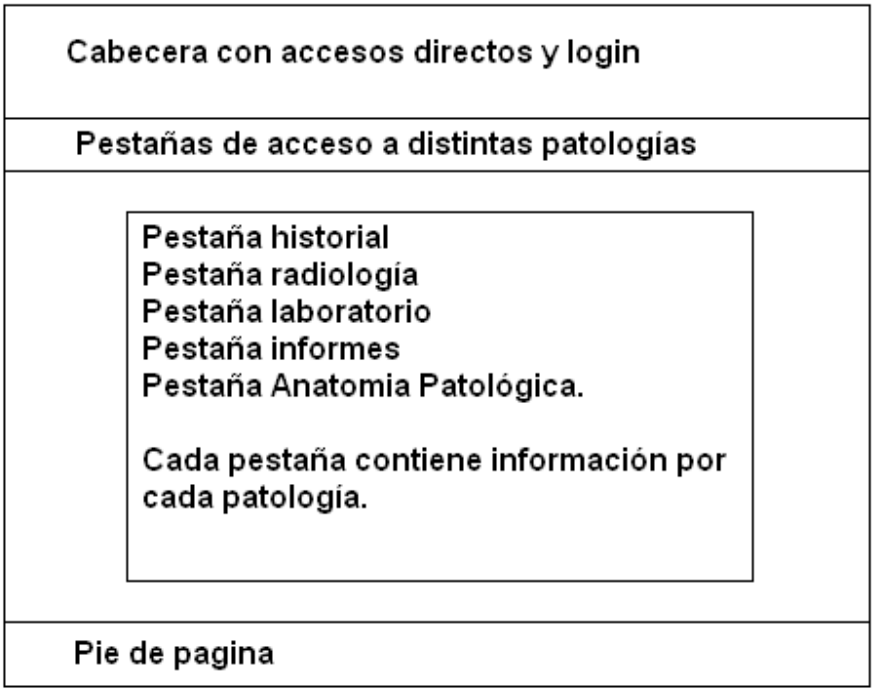
Entorno visual busqueda: Figura 14

Una vez que hemos hecho login y el usuario tiene permiso podemos acceder a las demás pantallas. La siguiente sería la información que realiza la búsqueda de paciente:



Entorno visual listado: Figura 15

Accediendo a la historia clínica mediante el enlace en el número de historia accedemos al historial médico.



Entorno visual historial: Figura 16

Todas las pestañas contendrán información detallada de cada episodio o pestaña que se refiera.

7. Diseño

7.1 Arquitectura aplicación

La aplicación será creada con tecnología J2EE. Esta tecnología nos permite utilizar un modelo como el que detallamos a continuación:

Programación eficiente

Para conseguir productividad es importante que los equipos de desarrollo tengan una forma estándar de construir múltiples aplicaciones en diversas capas (cliente, servidor web, etc.).
En cada capa necesitaremos diversas herramientas, por ejemplo en la capa cliente tenemos applets, aplicaciones Java, etc…
En la capa web tenemos servlets, páginas JSP, etc. Con JEE tenemos una tecnología estándar, un único modelo de aplicaciones, que incluye diversas herramientas; en contraposición al desarrollo tradicional con HTML, Javascript, CGI, servidor web, etc. que implicaba numerosos modelos para la creación de contenidos dinámicos, con los lógicos inconvenientes para la integración.

Extensibilidad

En un contexto de crecimiento de número de usuarios es precisa la gestión de recursos, como conexiones a bases de datos, transacciones o balanceo de carga. Además los equipos de desarrollo deben aplicar un estándar que les permita abstraerse de la implementación del servidor, con aplicaciones que puedan ejecutarse en múltiples servidores, desde un simple servidor hasta una arquitectura de alta disponibilidad y balanceo de carga entre diversas máquinas.

Integración

Los equipos de ingeniera precisan estándares que favorezcan la integración entre diversas capas de software.

La plataforma JEE implica una forma de implementar y desplegar aplicaciones empresariales. La plataforma se ha abierto a numerosos fabricantes de software para conseguir satisfacer una amplia variedad de requisitos empresariales.

La arquitectura JEE implica un modelo de aplicaciones distribuidas en diversas capas o niveles. La capa cliente admite diversas tipos de clientes (HTML, Applet, aplicaciones Java, etc.). la capa intermedia contiene subcapas (el contenedor web y el contenedor EJB). La tercera capa dentro de esta visión sintética es la de de aplicaciones como ERP, EIS, bases de datos, etc.
Como se puede ver un concepto clave de la arquitectura es el de contenedor, que dicho de forma genérica no es más que un entorno de ejecución estandarizado que ofrece unos servicios por medio de componentes. Los componentes externos al contenedor tienen una forma estándar de acceder a los servicios de dicho contenedor, con independencia del fabricante.
Utilizaremos un tipo de arquitectura de diseño MODELO – VISTA – CONTROLADOR también denominado MVC.

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón de llamada y retorno MVC (según CMU), se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.
Pasamos a describir la arquitectura:

Modelo

Esta es la representación específica de la información con la cual el sistema opera. En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

Vista

Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador

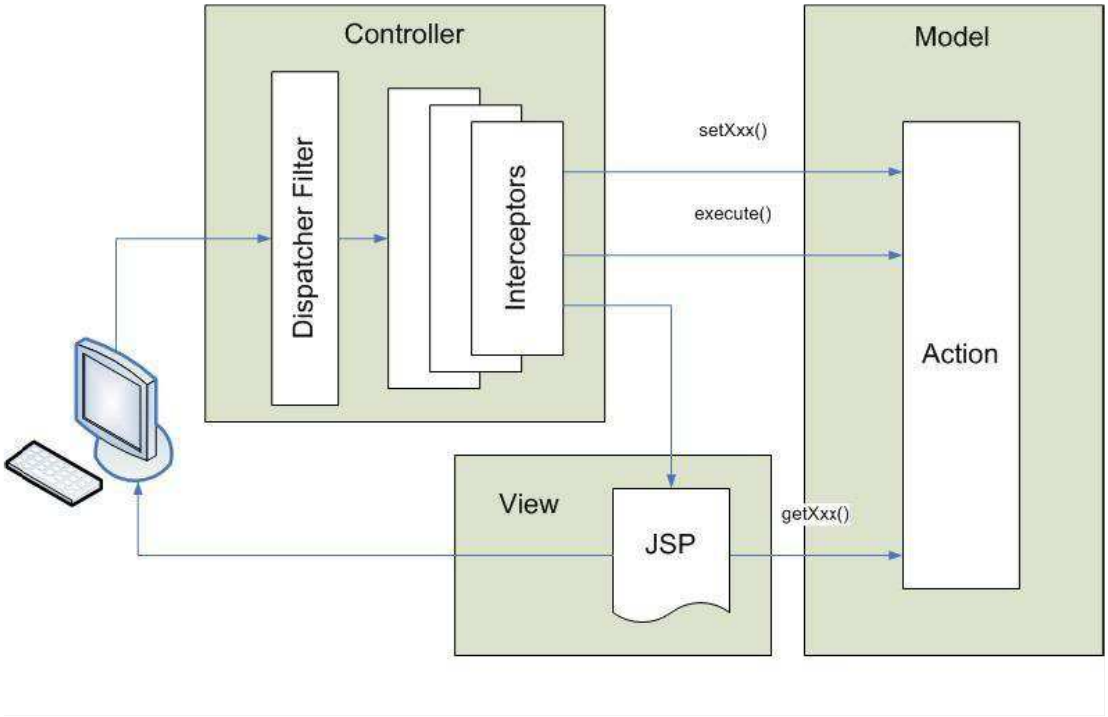
Este responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista.

Muchos de los sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos: en líneas generales del MVC corresponde al modelo.
La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre Vista y su correspondiente Controlador de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

Para facilitar el desarrollo de la capa de Vista utilizaremos STRUTS 2 la cual detallamos a continuación.

7.2 Framework Struts 2

Struts 2 es un framework de presentación, basado en el patron MVC. Se distingue su amplia capacidad de configuración y extensibilidad. Permite el uso de plugins de componentes e integración con otros frameworks.



Struts 2: Figura 17

Los principales componentes de Struts2 son:

- DispatcherFilter

- Interceptors
- Actions
- Results

DispatcherFilter

Es el punto de entrada del framework. A partir de el se lanza la ejecución del procesamiento para cada request que involucre al framework. Sus principales responsabilidades son:

- Ejecutar los Actions (handlers para los request)
- Comenzar la ejecución de la cadena de interceptors (interceptors chain) asociados al request
- Limpiar el ActionContext (para evitar memory leaks). El ActionContext es el contexto sobre el cual se ejecuta un Action.

Interceptors

Son clases que siguen el patrón interceptor, realizan tareas muy similares a los advice de AOP. Se encargan de interceptar la invocación a un Action. Permiten realizar operaciones antes y después de la invocación de un Action. También permiten evitar que un Action se ejecuto. Nos sirve para realizar cierta funcionalidad que queremos aplicar a un conjunto de Actions.
Struts2 trae definidos un conjunto de interceptors por defecto, que le permite realizar un conjunto de acciones sobre los Actions. el Request y Response.

Actions

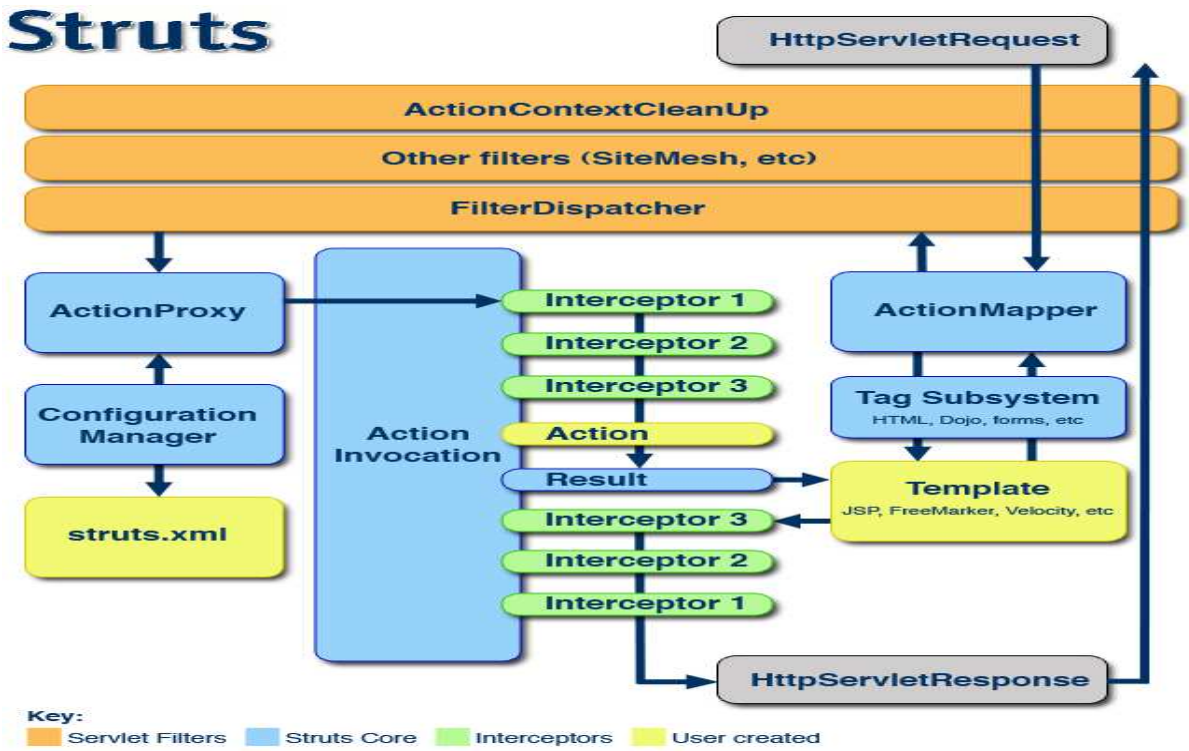
Los Actions serán lo encargados de ejecutar la lógica necesaria para manejar un request determinado. A diferencia de versiones anteriores de struts, los Actions no están obligados a implementar o heredar de una interfaz o clase ya definida. Pueden ser POJOs. Igualmente, struts2 posee una interfaz Action. Esta interfaz permite definir el método por defecto que se ejecutará sobre el Action para que no tengamos que definirlo por otro medio (Existen varias formas de indicarle a Struts2 el método a invocar sobre un action). También existe una implementación de esta interfaz, denominada ActionSupport, que nos provee funcionalidad de gran utilidad necesaria por un conjunto de Interceptores para poder manipular el Action a invocar.

Results

Los Results son Objetos que encapsulan el resultado de un Action. Los Actions de la aplicación simplemente devolverán un String en sus métodos. Un Action puede devolver diferentes resultados luego de su ejecución. Cada uno de estos

resultados se corresponde con un Result, previamente configurado en Struts2. La configuración de cada Result determina principalmente: el tipo de vista a mostrar (JSP, Velocity Templates, FreeMarker, entre otros), el recurso asociado a dicha vista, el nombre asociado.

La siguiente gráfica muestra una petición en http diferenciando cada capa.



Http struts 2: Figura 18

El funcionamiento es el siguiente:

1. Llega un Request a la aplicación.

2. El Request es interpretador por el DispatcherFilter y determina que Action y que conjunto de Interceptors invocar.

3. Cada Interceptor ejecuta sus acciones previas a la ejecución del método de Action a invocar

4. Es invocado el método del Action.

5. Cada Interceptor ejecuta sus acciones posteriores a la ejecución del método de Action a invocar

6. Se examina el resultado obtenido del Action y se determina el Result correspondiente.

7. Mediante el Result determinado se genera la vista, y según la configuración definida sobre el se invoca el proceso de generación de la vista.

8. La vista generada retorna al cliente.

Struts2 posee varios archivos de configuración:

struts.xml:

Es el principal archivo de configuración del framework. Aquí definimos los ActionMapping de nuestra aplicación, su división en Package, la registración de los Interceptors, la asignación de los Interceptors a los Package, entre otras cosas.

struts-default.xml:

Este archivo define una configuración básica de un Package del cual es conveniente que el resto de nuestro Packages hereden para obtener los beneficios del framework. Podemos redefinir esta configuración ubicando un archivo propio con el mismo nombre en el classpath.

struts.properties:

Este es un archivo de properties que contiene un conjunto de pares key - valor que nos permiten definir un conjunto de parámetros del framework. Este archivo es cargado por defecto por el framework (obteniendolo de struts-core.jar), pero podemos redefinir los parámetros ubicando el mismo en el classpath de la aplicación.

7.3 Rutas desarrollo aplicación

La aplicación será desplegada en el webapps de Tomcat y se llamará Intranet. Será un war que se desplegará automáticamente para ser utilizado sin más. En el war irán las clases con las librerías, imágenes, scripts, estilos y plantillas necesarias para el correcto funcionamiento. Por lo tanto, partiendo del directorio de Tomcat la ruta principal será webapps/Intranet. Partiendo de la ruta principal tendremos las siguientes rutas :

- Plantillas: Intranet/plantillas
- Imágenes: Intranet/imágenes

- Scripts, estilos: Intranet/estilos
- Librerías: Intranet/WEB-INF/lib
- Pestañas: Intranet/carpeta
- Herramientas: Intranet/herramientas
- Busqueda: Intranet/paciente
- Página inicial: Intranet/portada

7.4 Información de las clases

Las clases irán alojadas en el contenedor de la Intranet en:

- Intranet/WEB-INF/classes

La estructura será la siguiente:

- *classes/paciente*: Será donde se alojen las clases referentes a la busqueda del paciente y las conexion con el JDB.
- *classes/base*: Será donde se alojen las clases secundarias para operaciones.
- *classes/carpeta*: Directorio donde están las clases referentes a las pestañas.
- *classes/validar:* Ubicación para el login y el logout.
- *clases/utilidades:* Classes encargadas de utilizar las operaciones de crear,modficar y borrar datos.

7.5 Acceso a datos

La base de datos será desarrollada bajo MySQL. Existen 2 bases de datos. La base de datos llamada intranet se encargará de contener todo tipo de datos clínicos así como el de estadísticas, tráfico, etc… La segunda base de datos contiene la validación de usuarios con los permisos para acceso a las zonas restringidas.

La estructura es la siguiente:

```
--
-- Base de datos: `intranet`
--

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `anatomia`
--
```

```
CREATE TABLE IF NOT EXISTS `anatomia` (
  `numerohc` int(11) NOT NULL,
  `fecha` date NOT NULL,
  `servicio` varchar(50) NOT NULL,
  `medico` int(11) NOT NULL,
  `tipo_prueba` varchar(150) NOT NULL,
  `estado` varchar(20) NOT NULL,
  `nest` int(11) NOT NULL,
  PRIMARY KEY  (`numerohc`),
  UNIQUE KEY `nest` (`nest`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------


-- Estructura de tabla para la tabla `anatomia_inf`
--

CREATE TABLE IF NOT EXISTS `anatomia_inf` (
  `numerohc` int(11) NOT NULL,
  `micro` varchar(250) NOT NULL,
  `macro` varchar(250) NOT NULL,
  `nest` varchar(40) NOT NULL,
  `datos_clinicos` varchar(250) NOT NULL,
  `diagnostico` varchar(250) NOT NULL,
  PRIMARY KEY  (`numerohc`),
  UNIQUE KEY `nest` (`nest`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `estadistica`
--

CREATE TABLE IF NOT EXISTS `estadistica` (
  `usuario` varchar(20) NOT NULL,
  `ip` varchar(20) NOT NULL,
  `fecha` date NOT NULL,
  `hora` time NOT NULL,
  `servidor` varchar(100) NOT NULL,
  `conexion` varchar(500) NOT NULL,
  `clase` varchar(500) NOT NULL,
```

```
  KEY `usuario` (`usuario`),
  KEY `fecha` (`fecha`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;


-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `historial`
--

CREATE TABLE IF NOT EXISTS `historial` (
  `numerohc` int(11) NOT NULL,
  `episodio` varchar(20) NOT NULL,
  `fecha_registro` date NOT NULL,
  `servicio` varchar(50) NOT NULL,
  `fecha_alta` date NOT NULL,
  `prestacion` varchar(50) NOT NULL,
  KEY `numerohc` (`numerohc`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
--
-- Estructura de tabla para la tabla `informes`
--

CREATE TABLE IF NOT EXISTS `informes` (
  `numerohc` int(11) NOT NULL,
  `fecha` date NOT NULL,
  `servicio` varchar(50) NOT NULL,
  `medico` varchar(50) NOT NULL,
  `tipo` varchar(40) NOT NULL,
  `estado` varchar(30) NOT NULL,
  `hat` int(11) NOT NULL,
  PRIMARY KEY  (`numerohc`),
  UNIQUE KEY `hat` (`hat`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `informes_inf`
--

CREATE TABLE IF NOT EXISTS `informes_inf` (
  `hat` int(11) NOT NULL,
```

```
  `informe` longtext NOT NULL,
  UNIQUE KEY `hat` (`hat`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `laboratorio`
--

CREATE TABLE IF NOT EXISTS `laboratorio` (
  `numerohc` int(11) NOT NULL,
  `fecha` date NOT NULL,
  `servicio` varchar(50) NOT NULL,
  `medico` varchar(50) NOT NULL,
  `contenido` varchar(250) NOT NULL,
  `estado` varchar(20) NOT NULL,
  `peticion` varchar(20) NOT NULL,
  `custom` int(11) NOT NULL,
  PRIMARY KEY  (`numerohc`),
  UNIQUE KEY `custom` (`custom`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `labo_result`
--

CREATE TABLE IF NOT EXISTS `labo_result` (
  `custom` int(11) NOT NULL,
  `parametro` varchar(50) NOT NULL,
  `resultado` varchar(100) NOT NULL,
  `unidades` varchar(50) NOT NULL,
  `rinf` varchar(50) NOT NULL,
  `rsup` varchar(50) NOT NULL,
  KEY `custom` (`custom`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `medicos`
--
```

```
CREATE TABLE IF NOT EXISTS `medicos` (
  `codigo` varchar(20) NOT NULL,
  `nombre` varchar(100) NOT NULL,
  `apellidos` varchar(250) NOT NULL,
  PRIMARY KEY  (`codigo`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `pacientes`
--

CREATE TABLE IF NOT EXISTS `pacientes` (
  `numerohc` int(11) NOT NULL,
  `tis` varchar(20) NOT NULL,
  `nombre` varchar(100) NOT NULL,
  `apellid1` varchar(100) NOT NULL,
  `apellid2` varchar(100) NOT NULL,
  `direccion` varchar(250) NOT NULL,
  `dni` varchar(20) NOT NULL,
  `telefono` varchar(50) NOT NULL,
  `fechanac` date NOT NULL,
  `edad` int(11) NOT NULL,
  `sexo` varchar(10) NOT NULL,
  `numeross1` varchar(50) NOT NULL,
  `numeross2` varchar(50) NOT NULL,
  `poblacion` varchar(200) NOT NULL,
  PRIMARY KEY  (`numerohc`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `prestaciones`
--

CREATE TABLE IF NOT EXISTS `prestaciones` (
  `codigo` varchar(50) NOT NULL,
  `descripcion_presta` varchar(200) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------
```

```
--
-- Estructura de tabla para la tabla `radiologia`
--

CREATE TABLE IF NOT EXISTS `radiologia` (
  `numerohc` int(11) NOT NULL,
  `fecha_registro` date NOT NULL,
  `servicio` varchar(50) NOT NULL,
  `prestacion` varchar(50) NOT NULL,
  `imagen` varchar(50) NOT NULL,
  `informe` int(50) NOT NULL,
  `estado` varchar(2) NOT NULL,
  `medico` int(11) NOT NULL,
  PRIMARY KEY  (`numerohc`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `radiologia_inf`
--

CREATE TABLE IF NOT EXISTS `radiologia_inf` (
  `numerohc` int(11) NOT NULL,
  `fecha` date NOT NULL,
  `inf_datos` varchar(250) NOT NULL,
  `informe` int(11) NOT NULL,
  PRIMARY KEY  (`numerohc`),
  UNIQUE KEY `informe` (`informe`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `registro`
--

CREATE TABLE IF NOT EXISTS `registro` (
  `ip` varchar(14) NOT NULL default '',
  `fecha` varchar(30) default NULL,
  `clase` varchar(255) NOT NULL default '',
  `variable` varchar(2000) NOT NULL,
  `usuario` varchar(30) NOT NULL
```

```
) ENGINE=MyISAM DEFAULT CHARSET=latin1 PACK_KEYS=0;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `servicios`
--

CREATE TABLE IF NOT EXISTS `servicios` (
  `gfh` varchar(50) NOT NULL,
  `descripcion_servi` varchar(200) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `sesion`
--

CREATE TABLE IF NOT EXISTS `sesion` (
  `sesion` varchar(50) NOT NULL default '',
  `usuario` varchar(30) NOT NULL default '',
  `ip` varchar(50) NOT NULL default ''
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `visitas`
--

CREATE TABLE IF NOT EXISTS `visitas` (
  `aÃ±o` int(11) NOT NULL default '0',
  `fecha` date NOT NULL default '0000-00-00'
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

La base de datos de validación es:

```
--
-- Base de datos: `autorizacion`
--

-- --------------------------------------------------------
```

```
--
-- Estructura de tabla para la tabla `usuarios`
--

CREATE TABLE IF NOT EXISTS `usuarios` (
  `nif` varchar(15) default NULL,
  `cip` int(11) NOT NULL default '0',
  `cias` varchar(20) NOT NULL,
  `usuario` varchar(30) NOT NULL default '',
  `password` varchar(100) NOT NULL default '',
  `dircorreo` varchar(50) default NULL,
  `usugraba` varchar(15) NOT NULL default '',
  `fechamodi` timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP,
  `fechagra` timestamp NOT NULL default '0000-00-00 00:00:00',
  `descri` varchar(50) default NULL,
  `Centro` varchar(50) default NULL,
  `Servicio` varchar(50) default NULL,
  `estamento` varchar(50) NOT NULL default '',
  `numcolegiado` varchar(20) NOT NULL default '',
  `activo` char(1) NOT NULL default '',
  `fechabaja` date NOT NULL default '0000-00-00',
  PRIMARY KEY  (`usuario`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;


--
-- Volcar la base de datos para la tabla `usuarios`
--

INSERT INTO `usuarios` (`nif`, `cip`, `cias`, `usuario`, `password`, `dircorreo`,
`usugraba`, `fechamodi`, `fechagra`, `descri`, `Centro`, `Servicio`, `estamento`,
`numcolegiado`, `activo`, `fechabaja`) VALUES
(NULL, 0, '-', 'jaime', 'jaime', NULL, 'jaime', '2011-05-30 14:14:02', '0000-00-00
00:00:00', 'root', 'root', 'inf', '', '', '', '0000-00-00');

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `usu_permiso`
--

CREATE TABLE IF NOT EXISTS `usu_permiso` (
```

```
  `usuario` varchar(30) NOT NULL,
  `nombre_permiso` varchar(15) NOT NULL default '',
  PRIMARY KEY  (`usuario`,`nombre_permiso`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Volcar la base de datos para la tabla `usu_permiso`
--

INSERT INTO `usu_permiso` (`usuario`, `nombre_permiso`) VALUES
('jaime', 'admin'),
('jaime', 'manager'),
('jaime', 'role1'),
('jaime', 'tomcat');
```

Las BB.DD serán compuestas por cada pestaña, por lo tanto, tendremos una BB.DD por cada pestaña. La compasión será la siguiente:

- Pestaña Historial: BB.DD historial
- Pestaña Radiología : BB.DD radiologia y radilogia_inf
- Pestaña Anatomía patológica: BB.DD anatomia y anatomia_inf
- Pestaña Laboratorio: BB.DD laboratorio y labo_result
- Pestaña Informes: BB.DD informes y informes_inf
- Datos paciente: BB.DD pacientes
- Rastro: BB.DD rastro
- Autorización: BB.DD autorizacion

Cada pestaña tendrá su propia BB.DD y tendremos otra BB.DD donde contendrá toda la información del paciente.

Como debemos almacenar por ley como mínimo dos años toda la actividad de consulta, crearemos una BB.DD llamada rastro para almacenar todas las rutas consultadas.

Y por último, la BB.DD autorización donde alojaremos los usuarios dados de alta.

Las tablas forman parte de las BB.DD que hemos nombrado anteriormente. Cada BB.DD contendrá una tabla de registro clínico.

Todas pestañas tendrán su BB.DD y una tabla de registro para almacenar toda la actividad por separado. La excepción serán las BB.DD de Rastro y la de autorización, donde habrá varias tablas para poder registrar la información de forma organizada.

La BB.DD de autorización contendrá dos tablas, una para el registro del usuario y otra la que contendrá los permisos/roles de seguridad.

La BB.DD de Rastro llevará una tabla de registro para guardas las trazas de consulta y otra tabla estadística donde registraremos la actividad desglosada para obtener estadísticas de consulta.

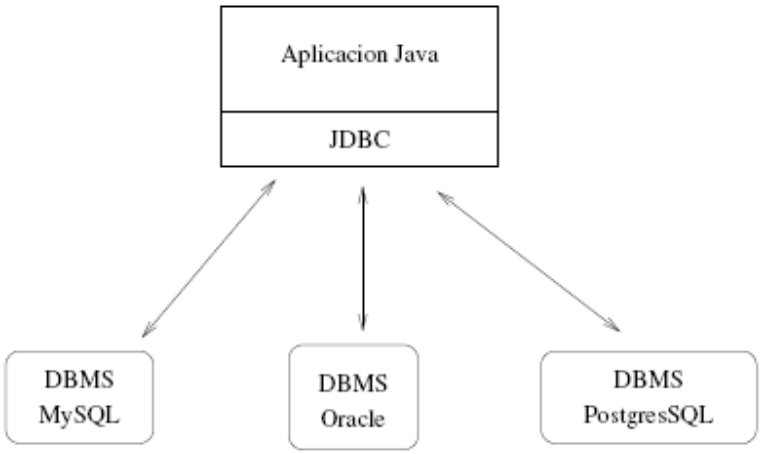
Modelo de acceso JDBC

El acceso a BB.DD será mediante conexión JDBC. Utilizaremos la librería standard de MySQL, la cual nos permitirá realizar SQLs sin ningún problema de compatibilidad.
La versión será la utilizada por MySQL , nombrada anteriormente.
Todas las conexiones estarán definidas en el fichero de configuración de Tomcat (Server.xml). Utilizaremos el pool de conexiones para realizar las conexiones, permitiendo así llevar una mejor gestión de cuantas conexiones se efectúen. Por lo tanto, todos los servlets estarán programados para la liberación de recursos mediante el pool.

Para poder acceder a la base de datos necesitaremos tener un conector específico para MySQL. El Driver que utilizamos es MySQL java connector que será el encargado de:

- Conexión con las fuente de datos
- Realizar peticiones y actualizaciones
- Manejar los resultados de las consultas

El esquema básico es el siguiente:



JDBC: Figura 19

7.6 Diseño de la Base de Datos

Una vez que sabemos las tablas y su contenido, así como la conexión a las mismas, mostramos el diseño en el siguiente diagrama:

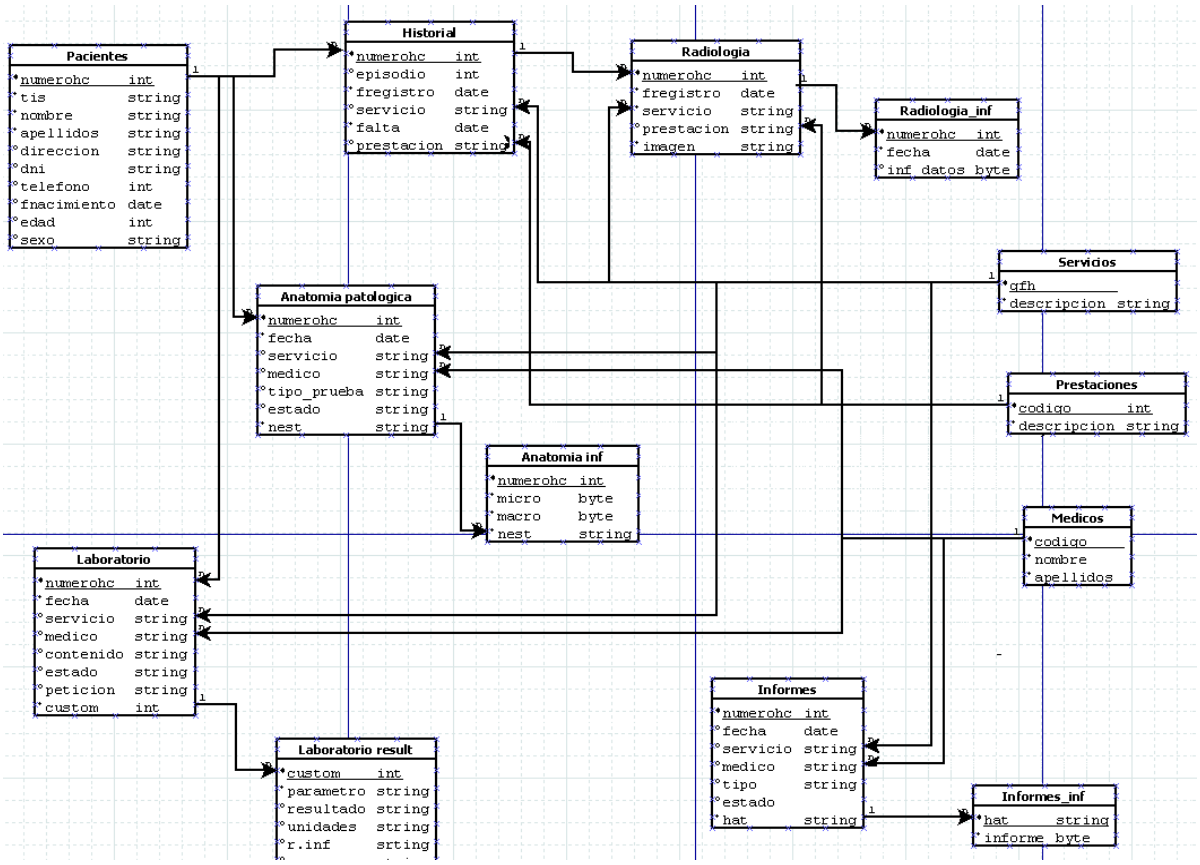


Diagrama bb.dd: Figura 20

8. Validación de usuario

En el apartado anterior hemos descrito que la validación de usuario sería mediante el pool de conexiones. Ahora vamos a describir como será la asignación del role al usuario.

8.1 Estructura Usuarios

Los usuarios serán dados de alta con una información básica para poder tener constancia de su persona. Los datos que utilizaremos para dar de alta al usuario serán:

- Nombre
- Apellidos
- Servicio
- DNI
- Usuario
- Password

Existira un usuario administrador que podrá acceder a la zona de herramientas para poder gestionar la aplicación. Las tareas del administrador serán:

Lista Usuario/role

Se utiliza para poder listar todos los usuarios que están dados de alta en el sistema. Como acción secundaria, podemos dar de baja eliminando al usuario.

Crear Usuario

Hay un formulario de registro de usuarios el cual necesitaremos para poder añadir nuevos usuarios al sistema. El formulario tiene unos campos obligatorios para poder realizar el registro.

Cambio password

Como acción complementaria existe un apartado de cambio de password para poder agilizar la gestión de las claves. Es importante este apartado ya que en la práctica habitual es muy común la perdida del password.

9. Diagrama de clases

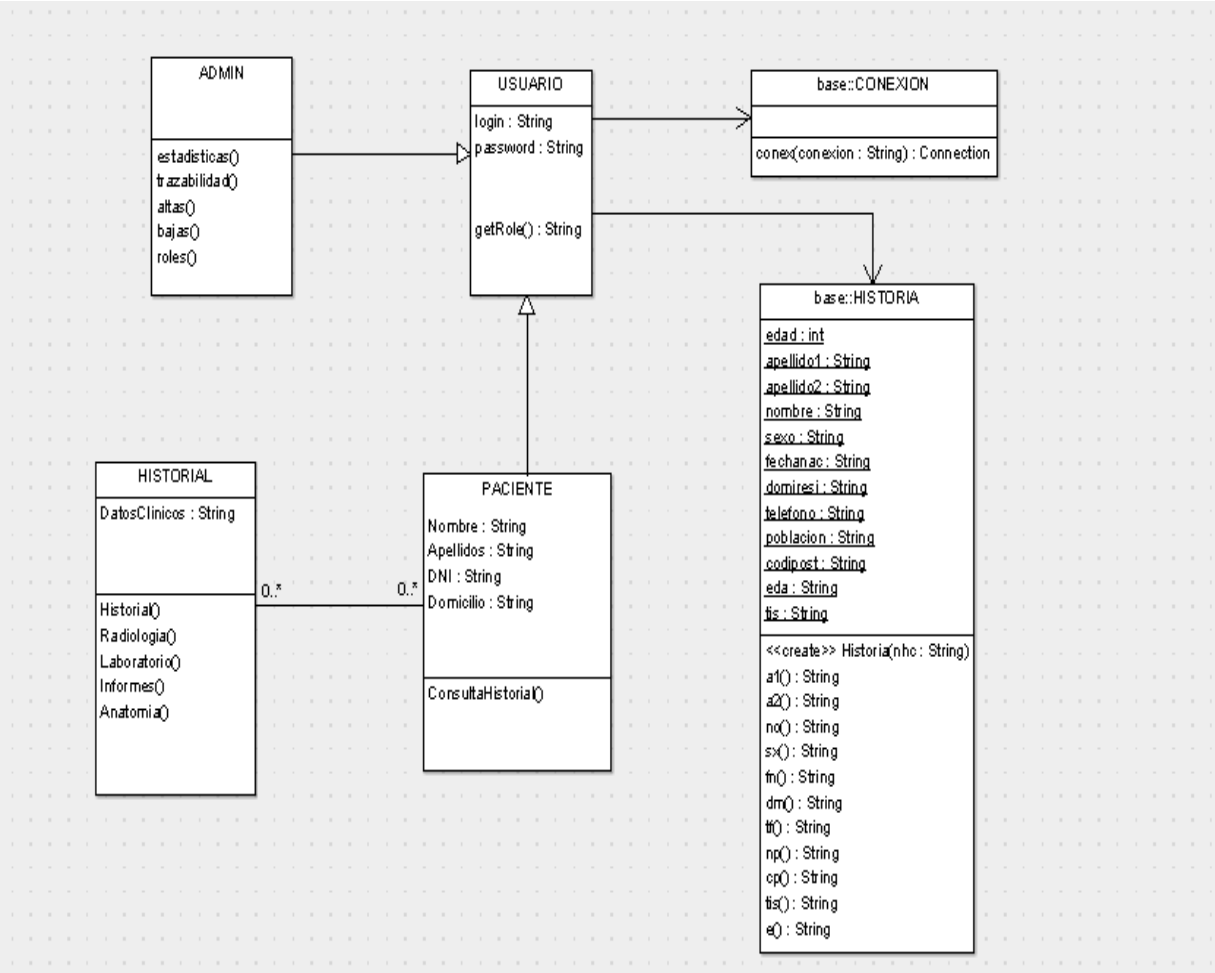


Diagrama clases: Figura 21

10. Herramienta de programación

Las herramientas de trabajo para realizar el proyecto son:

- Entorno de desarrollo utilizaré Netbeans 7.1
- Navegador para la visualización Internet Explorer 7 o superior

- Capa de presentación utilizaré JSP, HTML, CSS
- La comunicación con la capa de negocio será mediante Framework STRUTS 2.
- Como gestor de base de datos utilizaré MySQL con conexión JDBC.
- Para generar documentos utilizaré:

    - Microsoft Word 2003
    - Power point 2003
    - UML Draw

Y por último, necesitaremos el API de Java para poder consultar las clases y packages necesarios para el desarrollo:

http://download.oracle.com/javase/1.5.0/docs/api/

Capitulo 11. Valoración económica

En este capítulo se hace una valoración básica haciendo referencia, a grandes rasgos, las operaciones, software y hardware más importantes del proyecto en general. Se detalla en la siguiente tabla la valoración económica:

Actividad	***Importe (€)***
Inicio del proyecto	0
Gestión del proyecto	1500
Construcción del software	2100
Análisis	600
Diseño	2400
Programación y Pruebas unitarias	3500
Pruebas	500
Formación de usuarios	300
Puesta en producción	200
Final del proyecto	0
TOTAL	**11.100 €**

Presupuesto. Figura 22

La valoración economica solo muestra el aspecto desde el inicio del proyecto hasta el final, sin tener encuenta aspectos como el hardware o el fotware que se puediera utilizar en el caso de mantener propiamente la herramienta.

Capitulo 12. Conclusiones

Con la realización de este proyecto se ha comprobado la eficacia de la herramienta a la hora de obtener información de la historia de un paciente. Se ha sustituido el papel por la agilidad y ordenación de los datos para llevar a cabo un diagnostico o simplemente para aportar información al usuario final.

Existen muchos casos en los cuales existe un desorden sobre la historia de un paciente. Un paciente puede tener muchísimas hojas sobre episodios de hospitalización, radiología, urgencias, anatomía patológica, analíticas, etc… y difícil ordenar y agrupar cuando toda esa información está en papel. No solo eso, si no que puede causar un gran retraso en cualquier consulta o incluso en urgencias cuando hay que ordenar y leer toda esa información para que el especialista pueda saber con exactitud que cuadro clínico puede estar presentando el paciente.

Por eso, el conseguir mostrar en una sola pantalla, ordenado por fechas y episodios, conseguimos una mayor rapidez a la hora de tomar decisiones y una mejora a la hora de realizar un diagnóstico prematuro.

A medida que vas trabajando sobre el proyecto te vas dando cuenta sobre las mejoras que puedes realizar sobre la planificación inicial. Para realizar el proyecto me he apoyado en facultativos para valorar la información que podría ser útil. A medida que desarrollas el trabajo puedes observar que la información se puede mostrar con el mismo contenido pero con un desarrollo distinto para ser más eficaz. Por eso mismo, después de haber recorrido todo el camino, la verdad es que podría cambiar bastantes aspectos del proyecto haciendo que la herramienta fuera más interactiva con el usuario.

La experiencia o conclusión que puedo sacar es que las tecnologías actuales hacen más fácil el trabajo y sobre todo, dependiendo del campo en que se trabaja, puede llevar a cabo una misión tan delicada como la de incluso salvar vidas en determinados casos. El seguimiento de un paciente realmente puede ser efectivo si el especialista reúne toda la información necesaria en cualquier momento bajo una ordenación y sencillez, haciendo que el diagnóstico sea totalmente acertado.

Otra conclusión que puedo observar que ha sido significativo es que la herramienta es muy eficaz sobre todo en las zonas rurales, donde el acceso a los hospitales o centros de salud es más dificultosa por el medio en el que viven. Podemos encontrar pacientes rodeados de montañas y poder acceder a todo su historial sin tener que moverse del centro de salud más cercano gracias a la intranet clínica. Con esto conseguimos que el paciente no se tenga que desplazar

al hospital y pueda ser atendido por el centro de salud más cercano con todo el historial del mismo.

Para concluir, solo quiero hacer referencia a la planificación inicial. He visto y comprobado que una buena planificación puede ser la clave del éxito de cualquier proyecto ya que es la que realmente organiza todas las fases del proyecto haciendo que un proyecto llegue a buen puerto y sobre todo con el cumplimiento de las fechas que es un tema importante en el ámbito empresarial.

Capitulo 13. Glosario

Buscadores: Son servidores Web que tienen acceso a una extensa base de datos sobre recursos disponibles en la propia Web. El usuario conecta con un buscador e indica unas pocas palabras representativas del tema sobre el que está buscando información y que se utilizan como clave de búsqueda. Como resultado de la búsqueda se muestra al usuario una lista con enlaces a páginas Web en cuya descripción o contenidos aparecen las palabras claves suministradas.

Autentificación: Acción de verificar la identidad de una persona o proceso.

Cabecera: Información acerca de un documento Web o un mensaje de correo que se encuentra al principio del documento o mensaje. La información que contiene una cabecera puede hacer referencia al autor, o el generador del texto.

Conexión: Circuito virtual de transporte que se establece entre dos programas de aplicación con fines comunicativos.

Contraseña: Palabra o cadena de caracteres, normalmente secreta, para acceder a través de una barrera. Se usa como herramienta de seguridad para identificar usuarios de una aplicación, archivo, o red. Puede tener la forma de una palabra o frase de carácter alfanumérico, y se usa para prevenir accesos no autorizados a información confidencial.

CPU: Central Process Unit. Unidad Central de Proceso. El "cerebro" funcional del ordenador. Ejecuta instrucciones. Elemento que realiza la suma y sustracción real de los ceros y unos esencial para el cálculo.

Formulario: Fragmento de código HTML que permite la interactividad del usuario con una página Web mediante el sistema denominado CGI.

Login: Nombre que se usa para acceder a un sistema de ordenadores. Acción de entrar en un sistema de ordenadores.

Navegador: Aplicado normalmente a programas usados para conectarse al servicio WWW.

Nombre de usuario: El nombre que utiliza el usuario cuando accede a otro ordenador.

Password: Ver contraseña.

Servidor: Cualquier ordenador de una red que proporciona servicios a otras estaciones de la red.

URL: Uniform Resource Locator. "Localizador Uniforme de Recursos". Denominación que no solo representa una dirección de Internet sino que apunta a un recurso concreto dentro de esa dirección.

Username: Ver nombre de usuario.

S.O: Sistema operativo.

Base de datos: Sistema de almacenamiento de datos muy flexible que te permite utilizar la información en función de diversos criterios.

Hardware: (soporte físico): Nombre con el que se designa a los componentes físicos de los sistemas informáticos: unidad central del ordenador, periféricos, cables, conectores...

Java: Lenguaje de programación, desarrollado por Sun Microsystem, que también se utiliza para incluir pequeños programas (applets) en las páginas Web, que se activan al cargar la página. Permiten introducir más interacción en las páginas Web. El sistema ActiveX de Microsoft funciona de forma parecida.

JavaScript: Lenguaje complementario de Java que se utiliza para insertar "scripts" con funcionalidades diversas en las páginas Web. Forman parte del lenguaje HTML dinámico; son programas que se insertan totalmente en las páginas Web y se ejecutan en los ordenadores clientes. Permiten añadir

Sistema operativo (S.O): Es el conjunto de programas que nos permiten comunicarnos con el ordenador y ordenarle la ejecución de determinadas tareas: ver lo que hay en un disco, copiar y transferir datos, ejecutar programas...

WWW *(World Wide Web)*: La telaraña mundial. Gran sistema de información en Internet formado por todas las páginas web alojadas en los servidores de la Red y relacionadas entre si mediante enlaces o hipervínculos. Cada página tiene una dirección a través de la cual se puede acceder a ella con un programa navegador (Netscape, Internet Explorer...) desde cualquier ordenador conectado a Internet

Capitulo 14. Bibliografía

General

Google: http://www.google.es
Wikipedia: http://www.wikipedia.org

Java

API: http://download.oracle.com/javase/1.5.0/docs/api/
Documentación java: http://www.java2s.com/

Configuración Server

Tomcat**: http://tomcat.apache.org/**

CSS

Tutorial CSS: http://www.usabilidad.tv/hojas_de_estilos_css/css.asp

Bases de datos

MySQL: http://www.mysql.com/

Capitulo 15. Anexos

No existen documentos anexos al respecto.