UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA UNIVERSITARIA DE INFORMÁTICA

MASTER EN INGENIERÍA WEB

- PROYECTO FIN DE MÁSTER -

“Seguridad en la programación web de un prototipo de aplicación para la gestión hospitalaria”

Autor: David Gómez Pedrero

Fecha: Junio, 2012

ANEXO I: Propuesta de Proyecto Fin de Máster

Titulación	**Máster Universitario en Ingeniería Web**
Título del TFM	"Seguridad en la programación web de un prototipo de aplicación para la gestión hospitalaria"
Tutor coordinador	Juan Alberto de Frutos Velasco
Otros tutores	----------
Alumnos	David Gómez Pedrero
Objetivos	El desarrollo de un prototipo para la gestión hospitalaria, enfocado en la aplicación de técnicas seguridad web para minimizar varias de las amenazas conocidas.
Descripción	Se pretende implementar técnicas de seguridad web para gestionar: • Autenticación y autorización de usuarios • Validación de los datos • Seguridad con los ficheros y para minimizar los riegos provenientes de las siguientes amenazas: • Cross Site Scripting (XSS) • SQL Injection • Robo de sesiones
Aprobación de la propuesta	V° B°, Tutor Juan Alberto de Frutos Velasco Fdo. V° B°, Comisión Académica de Postgrado Fdo. SERGIO ARÉVALO VIÑUALES
Aprobación para la defensa	Fdo.
Calificación del TFM	Presidente — Fdo.
	Vocal — Fdo.
	Secretario — Fdo.

UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA UNIVERSITARIA DE INFORMÁTICA
MADRID

ÍNDICE DE CONTENIDOS

ANEXOS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 DESCRIPCIÓN GENERAL DEL PROYECTO

La razón para realizar el presente Proyecto Fin de Máster (*PFM*) parte de la idea de desarrollar el prototipo de una aplicación para el tratamiento de información, flexible en su concepción e implementación para diferentes entornos de gestión, a la vez que segura, para minimizar la vulnerabilidades existentes y reducir las posibilidades de éxito de los ataques web más populares actualmente, tales como:

- Cross Site Scripting (XSS)
- SQL Injection
- Robo de sesiones

Para tal fin, se han aplicado varias técnicas de seguridad web para la gestión de:

- Autenticación y autorización de usuarios
- Validación de los datos
- Seguridad con los ficheros
- Uso de certificados SSL

Durante el presente proyecto se ha hecho especial hincapié en el tema de la seguridad en las aplicaciones web, aplicando diferentes técnicas y métodos, estudiadas y aprendidas durante el desarrollo del máster y del proyecto, para mitigar en la mayor cantidad posible las principales amenazas existentes actualmente.

1.2 OBJETIVOS

El objetivo principal del presente Proyecto Fin de Máster (PFM) ha sido la realización de una aplicación web segura, de fácil desarrollo y mantenimiento, flexible en su crecimiento y adaptable a cualquier entorno de gestión, sin renunciar a cualquier otro entorno que requiera una especificación mayor.

Como consecuencia de lo anteriormente expuesto, se han documentado las fases de análisis, diseño e implementación necesarias para la realización de la aplicación, del modo más claro, legible y objetivo posible con la intención de que el tribunal, o cualquier otro lector interesado en el tema, ajeno o no al mismo, sea capaz de comprenderlo sin problema alguno.

De igual manera, se han comentado las características principales de varios de los ataques web más utilizados, así como de los métodos y técnicas para mitigar sus efectos y del énfasis en su aplicación para desarrollar una aplicación práctica y segura.

2 INTRODUCCIÓN A LA SEGURIDAD EN APLICACIONES WEB

Nos encontramos actualmente en la era digital; todo nuestro entorno depende de dispositivos electrónicos que en su gran mayoría están conectados entre sí a través de Internet. Cada vez más notamos la masificación de accesos a la red, ya sea por el aumento en las posibilidades de conexión o quizá por la necesidad de estar conectado para poder realizar tareas tan básicas como solicitar una cita médica o hacer una transacción bancaria, lo que conlleva a una gestión de datos que requiere un tratamiento seguro y confiable para todo aquel que hace uso de cualquier servicio en la web.

Según [Garfinkel, 2002], la seguridad web se define como el conjunto de procedimientos, prácticas y tecnologías que aseguran la fiabilidad y el comportamiento esperado de servidores web, navegadores, aplicaciones y de toda la infraestructura de internet que se encuentra involucrada.

Tomando como punto de partida esta definición y considerando el enfoque de éste proyecto, éste capítulo tiene como objetivos:

- Describir los conceptos básicos de seguridad de la información.
- Describir los tipos de amenazas para aplicaciones web.

2.1 Principios básicos de la Seguridad de la Información

Antes de describir los principios básicos de la Seguridad de la Información, es importante acotar el proyecto de acuerdo a los tipos de Seguridad que existen. En general la seguridad se puede clasificar en dos grandes tipos:

- Seguridad Física: Se refiere a la protección del Hardware y de los soportes de datos, así como a la de los edificios e instalaciones que los albergan. Contempla las situaciones de incendios, sabotajes, robos, catástrofes naturales, etc.

- Seguridad Lógica: Se refiere a la seguridad de uso del software, a la protección de los datos, procesos y programas, así como la del ordenador y autorizado acceso de los usuarios a la información.

De acuerdo a esta división y considerando el objetivo de este proyecto, el desarrollo del mismo irá enfocado a la Seguridad Lógica, por lo que se deben perseguir los siguientes objetivos:

- Restringir accesos a programas y archivos.
- Asegurar que los usuarios no puedan modificar programas y/o archivos sobre los que no deben tener acceso.
- Asegurar que la información transmitida sea recibida por el destinatario seleccionado y no otro.
- Asegurar que la información recibida sea la misma que fue transmitida.

Basándonos en el concepto de Seguridad de la Información, específicamente sobre la Fiabilidad, es importante considerar 3 factores básicos, de acuerdo a la norma ISO 27001. [ISO27001, 2009]

- Confidencialidad: El grado en el cual se restringe el acceso a la información y se definen grupos de personas dándoles autorización para acceder a ella.
- Integridad: El grado en el que la información se mantiene actualizada y sin errores. Para su cumplimiento, la información debe ser correcta y completa.
- Disponibilidad: Es la medida tanto la información y los sistemas que la gestionan se encuentran disponibles para los usuarios, en el momento en que la organización lo requiera. Sus características son: Oportunidad, Continuidad y robustez.

Para proteger un sistema y poder cumplir con los principios básicos de la seguridad de la información, específicamente del tipo lógico, es necesario saber exactamente contra qué tipo de amenazas debe ser blindado y así poder definir e implementar las mejores medidas.

2.1.1 Tipos de Amenazas

Una amenaza puede ser definida como todo elemento o acción capaz de atentar contra la seguridad de la información. Estas surgen a partir de las vulnerabilidades, es decir, sólo cuando una vulnerabilidad puede ser aprovechada, esta acción se convierte en amenaza.

Como tipos de amenazas que podrían afectar a un sistema de información podemos encontrar las siguientes:

- Interrupción: Sucede cuando se logra que un objeto se pierda, quede inutilizable o no disponible.
- Interceptación: Se presenta cuando un objeto no autorizado consigue acceder a un objeto del sistema.
- Modificación: Además de interceptar el objeto, este ataque logra modificarlo. Puede incluir además la destrucción del objeto
- Fabricación: Se presenta cuando además de lograr modificar el objeto, consigue un comportamiento similar al objeto inicial, de manera que no es fácil diferenciarlos entre sí.

Las amenazas también pueden ser clasificadas de acuerdo a su origen [ISO27001, 2009]. Estás pueden ser:

- Humanas
    - Intencionales: En este grupo encontramos amenazas que pueden ser provenientes de ex – empleados de compañías que actúan por venganza, así como empleados no promovidos, hackers, crackers, entre otros. Una de las técnicas más utilizadas especialmente por empleados o ex – empleados es la ingeniería social.
    - No intencional: Accidentes provocados por el uso no adecuado de funciones del sistema, el uso de dispositivos externos con virus, etc. Suelen presentarse cuando no se han tomado medidas de seguridad adecuadas o no han sido aplicadas correctamente.

- No Humanas: Se presentan cuando ocurren eventos inesperados que afectan la infraestructura base de un sistema.

2.1.2 Tipos de Ataques

2.1.2.1 Autenticación

Autenticación es el método mediante el cual un usuario valida su identidad en un sistema usando al menos uno de los siguientes mecanismos: “Algo que se tiene”, “Algo que se conoce” ó “Algo que se es”.

Este apartado pretende recopilar los ataques enfocados a quebrantar la validación de usuarios en un sistema web, explotando o engañando el proceso de autenticación.

2.1.2.1.1 Fuerza bruta

El ataque de fuerza bruta, es el procedimiento utilizado para sobrepasar los métodos de seguridad implementados para acceder a un sistema, haciendo uso de herramientas que con la ayuda de diccionarios de palabras (incluyendo combinaciones de caracteres especiales, mayúsculas y minúsculas, cadenas alfa-numéricas entre otras) hacen pruebas aleatorias (prueba y error) hasta conseguir obtener un usuario y contraseña válidos. [INET, 18]

2.1.2.1.2 Uso de Exploits

Consiste en explotar agujeros en los algoritmos de encriptación utilizados, en la administración de las claves, o simplemente aprovechar un error en los programas utilizados.

Los programas para explotar estos "agujeros" reciben el nombre de Exploits y lo que hacen es aprovechar la debilidad, fallo o error hallado en el sistema (hardware o software) para ingresar al mismo. [INET, 19]

2.1.2.2 Autorización

Este factor de seguridad contempla los niveles de accesos asignados a cada uno de los usuarios una vez estos han superado el proceso de autenticación. Determina exactamente a qué opciones del sistema tendrán acceso los usuarios y en muchos casos, determina además qué tipo de operaciones o transacciones podrán usar. Algunos de los tipos de ataque que pueden afectar su correcto funcionamiento son:

2.1.2.2.1 Cross-Site Scripting

Cross-Site Scripting es el nombre que recibe una vulnerabilidad que radica en la pobre verificación por parte de los sitios web de las cadenas de entrada enviadas por los usuarios a través de formularios, o directamente a través del URL. Estas cadenas, en el caso de ser maliciosas, podrían llegar a contener scripts completos. Cuando esta entrada se le muestra dinámicamente a un usuario dentro de una página web, en caso de contener un script, éste se ejecutará en el navegador del usuario dentro del contexto de seguridad de la página web visitada. Como consecuencia, podrá realizar en el ordenador del usuario todas las acciones que le sean permitidas a ese sitio web, como por ejemplo interceptar entradas del usuario víctima o leer sus cookies. [INET, 20]

Este ataque tiene una particularidad y es que el cliente puede ser quien lo ejecuta sin ser consciente de ello, siguiendo algún link, botón o cualquier funcionalidad que contiene el código malicioso incluido por el atacante.

Algunos de los tipos de ataque podrían ser: [de Frutos, 2012]

- Robo de Sesiones:
  Una de las vulnerabilidades que puede ser aprovechada por este tipo de ataque es la falta o mala gestión de sesiones en una aplicación web. Un ID de sesión podría ser recuperado haciendo uso de un script malicioso que consiga acceder a la cookie que lo contenga y, dependiendo entonces del tiempo en que ésta sesión expire, podrá darle la posibilidad al atacante de entrar al sistema. Gráficamente éste ataque se puede ver así: [INET-21]

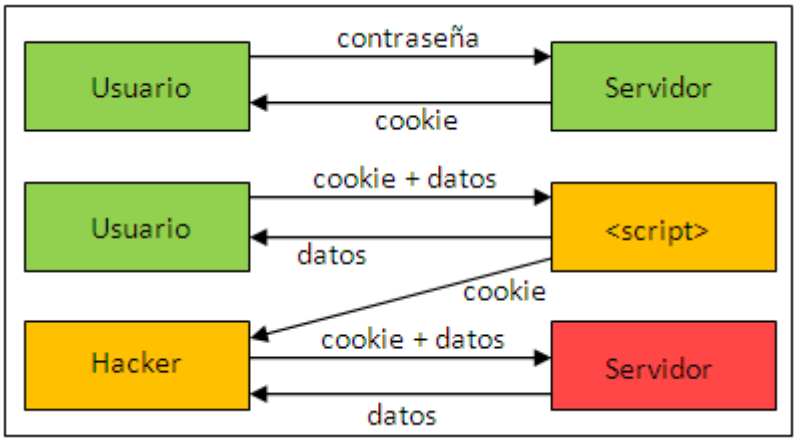

Figura 1. Robo de sesiones

- Phishing:

  Haciendo uso de XSS (Cross-Site Scripting), un atacante puede obtener datos sensibles de usuarios de una aplicación web aplicando "Phishing", técnica que consiste en proveer links a usuarios direccionándolos a sitios maliciosos con la apariencia de un sitio de confianza, engañando así al usuario para conseguir sus credenciales de acceso entre otros.

2.1.2.2.2 Fijación de Sesión

Este tipo de ataque, genera un identificador genuino (que no está asociado a ningún usuario por el momento) en el portal web afectado para, a continuación, tratar que la víctima se autentique en el portal con él. De este modo, el atacante obtiene un identificador de un usuario autenticado que puede utilizar para realizar acciones en el portal afectado en nombre de la víctima. [INET, 21]

Para llevarlo a cabo, el atacante también puede aprovechar la vulnerabilidad de encontrar en la aplicación web que el identificador de sesión sea enviado como parámetro a través del método GET. [de Frutos, 2012]

2.1.2.2.3 Ataques semánticos

Son aquellos provocados por la poca o nula validación de datos de entrada en una aplicación web. Los datos pueden provenir de dos fuentes: Un usuario o una entidad remota. A su vez se pueden clasificar en: Datos filtrados o corruptos.

Es importante considerar siempre los datos de entrada ya sea provenientes de un usuario o de una entidad remota como "datos corruptos", así se asegura que todos los datos de entrada sean filtrados y validados. [INET, 23]

Si no existe la suficiente validación y filtrado de datos de entrada, la aplicación quedaría expuesta a sufrir ataques semánticos, los cuales se refieren a dejar abierta la posibilidad de afectar el comportamiento del sistema por modificaciones que podrían ser introducidas a través de los inputs o datos de entrada (métodos GET, POST, URL's, campos de formularios, etc).

2.1.2.2.4 SQL Injection

Otra de las vulnerabilidades a considerar en aplicaciones web, es el SQL Injection, el cual, según la OWASP consiste en la inserción o "inyección" de una sentencia SQL a través de cualquier entrada de datos desde el cliente.

Un ataque exitoso de SQL Injection puede leer datos bastante sensibles en la base de datos, modificarlos a través de sentencias de inserción, actualización o borrado ó incluso ejecutar tareas de administración sobre el DBMS como podrían ser apagar el administrador de bases de datos, entre otros. [INET, 24]

Una de las formas de SQL Injection, quizá la principal consiste en aprovechar los campos de datos en cliente que se refieren a variables en la aplicación cuyos valores hacen parte de una sentencia SQL, concatenando así sus valores a éstas y ejecutándolas. Un ejemplo podría ser el siguiente:

SELECT * FROM `tabla` WHERE `user` = '$_POST[usuario]' && `pass` = '$_POST[pass]'

Si los valores introducidos en los campos del formulario son:

Login: user

Password: ‘|| 1 = ‘1

La sentencia SQL quedaría así:

SELECT * FROM `tabla` WHERE `user` = 'user' && `pass` = '' || 1 = '1'

Otro ejemplo podría ser:

pass'; drop table tabla—

En este caso, la siguiente consulta la ensambla el script:

SELECT * FROM tabla WHERE ‘user’ = 'pass';drop table tabla--'

En la sentencia anterior el (;) significaba que finaliza la sentencia select y se ejecuta una nueva sentencia, esta vez un “drop table” indicando al final que cualquier cosa que siga a este “drop table” es un comentario, invalidando cualquier otro comando que esté ubicado en el script después de esta sentencia.

2.2 SSL

Protocolo diseñado para dar seguridad al intercambio de datos entre dos aplicaciones, principalmente entre un servidor web y un navegador.

A nivel de arquitectura, el protocolo SSL es insertado a nivel de TCP/IP y el protocolo a alto nivel HTTP, para el cual fue realmente diseñado así:

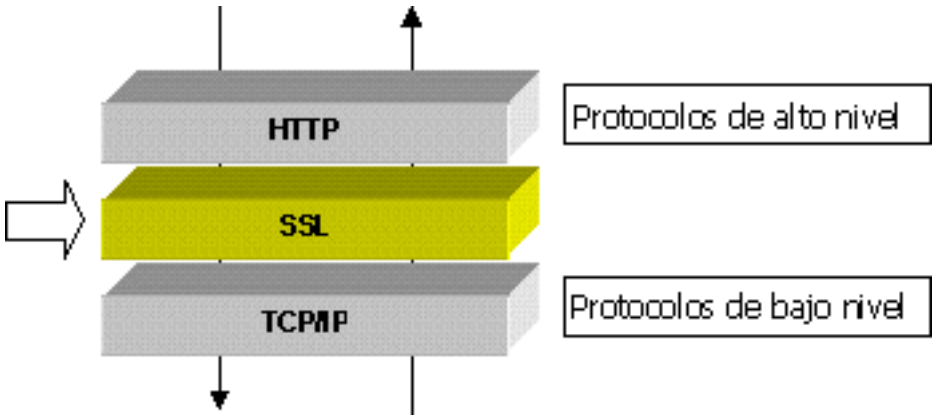

Figura 2. SSL

Este protocolo está diseñado para autenticar la identidad del emisor y receptor, así como la confidencialidad e integridad de la información intercambiada:

- Autenticación: la identidad del emisor y receptor son confirmadas.
- Confidencialidad: los datos enviados se encriptan de manera que una tercera persona pueda entender el mensaje.
- Integridad: los datos recibidos no han sido alterados, por accidente o fraudulentamente.

SSL utiliza una técnica para desecriptar llaves públicas basada en un par de llaves asimétricas para encriptar y desencriptar: una llave pública y una llave privada.

La llave privada se utiliza para encriptar los datos, es conservada por el emisor (el sitio Web). La llave pública se utiliza para desencriptar la información y se envía a los receptores (navegadores Web) por medio de un certificado. Cuando utiliza SSL en Internet, el certificado se entrega por medio de una autoridad de certificación. El sitio Web paga a la autoridad certificadores por entregar un certificado que garantice la autenticación del servidor y que contenga la llave pública que permita el intercambio de datos en modo seguro.

3 ESPECIFICACIÓN DE REQUISITOS SOFTWARE

3.1 INTRODUCCIÓN

Como se menciona en [Pressman, 1998], la especificación de los requisitos del software implica la culminación de la tarea del análisis de la aplicación. Dicha especificación se logra estableciendo una completa descripción de las clases que colaboran, su función y el comportamiento de la aplicación, de forma que se tenga una idea clara y concisa de lo que hay que realizar, quedando satisfechas las necesidades de quien demanda la construcción del sistema.

Este apartado cumple tres objetivos importantes:

- Describir lo que requiere el usuario.
- Establecer una base para la creación de un diseño de software.
- Definir un conjunto de requisitos que se puedan validar una vez que se ha construido el software.

De esta manera, se ha tratado de establecer las bases para un buen diseño de la aplicación, documentando la funcionalidad del software al definir las clases principales que componen la aplicación, así como sus atributos y métodos, además de las relaciones existentes entre ellas.

La estructura del presente documento se ha realizado de acuerdo a varios de los puntos del Software Engineering Standards Committee of the IEEE Computer Society; standard IEEE Std 830-1998, junto con las directrices del método Orientado a Objetos propuestas por Larman [Larman, 1999].

3.1.1 Propósito

El propósito de esta sección es definir el conjunto de Especificaciones de Requisitos Software que debe cumplir la aplicación, estableciendo los pasos necesarios para satisfacer las necesidades de los usuarios con respecto a la realidad modelada por el sistema implementado.

A partir de la especificación de requisitos estaremos en condiciones de establecer un diseño que se ajuste a los requerimientos expuestos, después de realizar el estudio de los componentes necesarios.

3.1.1.1 Audiencia a la que va dirigido

Dirigido al tribunal de evaluación del Máster en Ingeniería Web y a todos aquellos que sientan la necesidad o curiosidad por la aplicación de mecanismos y técnicas de seguridad en el desarrollo y explotación de aplicaciones web.

3.1.2 Alcance

Se pretende eliminar o, en su defecto, minimizar los principales riegos existentes en aplicaciones web desarrollados con tecnología PHP en el lado del servidor y de Javascript en el lado de los clientes.

Las principales amenazas tratadas en el presente Proyecto Fin de Máster han sido:

- Cross Site Scripting (XSS)
- SQL Injection
- Robo de sesiones

Las técnicas y mecanismos de seguridad web aplicadas han sido:

- Autenticación y autorización de usuarios
- Validación de los datos
- Seguridad con los ficheros
- Certificados Digitales

3.1.2.1 Funcionalidades

La aplicación permitirá:

- Listar, filtrar, crear, modificar y eliminar usuarios del sistema.
- Listar, filtrar, crear, modificar y eliminar pacientes del sistema.
- Listar, filtrar, crear, modificar y eliminar admisiones creadas para pacientes existentes.

Para conseguir llevar a cabo estas funcionalidades, la aplicación debe ser capaz de:

- Comunicarse correctamente con el gestor de base de datos para efectuar las peticiones realizadas por el usuario.
- Presentarse una interfaz adecuada, independientemente de navegador y del sistema operativo desde el que acceda el usuario.
- Evaluar los permisos definidos para el usuario autenticado para presentar la información y opciones para las cuales tiene derechos.

3.1.2.2 Beneficios, objetivos y metas

Por todo lo anteriormente expuesto, el presente proyecto tiene como principal objetivo la seguridad de las aplicaciones web frente a distintas amenazas. Para ello, se han aplicado diversos métodos y técnicas de seguridad para tal fin, conocimientos adquiridos durante la realización del Máster en Ingeniería Web y el desarrollo del Proyecto Fin de Máster.

Los beneficios de la seguridad web son claros si tenemos en cuenta que, cuanto más sensible e importante es la información con la que se trata, más imprescindible se hace garantizar la integridad, confidencialidad y disponibilidad de la misma.

3.2 DEFINICIONES Y ACRÓNIMOS

3.2.1 Definiciones

Cross Site Scripting [INET-01]

Es un tipo de inseguridad informática o agujero de seguridad típico de las aplicaciones Web, que permite a una tercera parte inyectar código JavaScript en páginas web vistas por el usuario, evitando medidas de control como la Política del mismo origen. También conocido como XSS.

Flujo de Trabajo [INET-05]

Es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas. Generalmente los problemas de flujo de trabajo se modelan con redes de Petri.

Secure Socket Layer [INET-06]

Proporciona autenticación y privacidad de la información entre extremos sobre Internet mediante el uso de criptografía. Habitualmente, sólo el servidor es autenticado (es decir, se garantiza su identidad) mientras que el cliente se mantiene sin autenticar.

SSL implica una serie de fases básicas:

- Negociar entre las partes el algoritmo que se usará en la comunicación.
- Intercambio de claves públicas y autenticación basada en certificados digitales.
- Cifrado del tráfico basado en cifrado simétrico.

XAMPP [INET-07]

Se trata de un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre está formado por los acrónimos: **X** (para cualquiera de los diferentes sistemas operativos), **A**pache, **M**ySQL, **P**HP, **P**erl.

.HTACCESS [INET-08]

Se trata de un fichero especial (hypertext access), también conocido como *archivo de configuración distribuida*, popularizado por el Servidor HTTP Apache el cual permite definir diferentes directivas de configuración para cada directorio (con sus respectivos subdirectorios) sin necesidad de editar el archivo de configuración principal de Apache *php.ini*.

3.2.2 Acrónimos

PHP	Hypertext Pre-processor
XSS	Cross Site Scripting
SSL	Secure Socket Layer
PFM	Proyecto Fin de Master
MVC	Modelo Vista Controlador
AC	Autoridad de Certificación
CA	Certification Authority
EC	Entidad de Certificación
CE	Certification Entity
CRUD	Create, Read, Update, Delete
XAMPP	X, Apache, MySQL, PHP, Perl
MIW	Máster en Ingeniería Web
RFI	Remote File Inclusion
GPC	Get, Post, Cookie
EGPS	Entorno, GET, POST, Cookie, Server
CSRF	Cross-Site Request Forgery
SID	Sesión Identity
URL	Uniform resource locator
IDE	Integrated Development Environment

3.3 DESCRIPCIÓN GLOBAL

3.3.1 Perspectiva del Producto.

La aplicación ha sido desarrollada aplicando uno de los patrones de arquitectura de software más frecuentemente utilizado para aplicaciones web: Modelo Vista Controlador (MVC). Este patrón se caracteriza por separar el desarrollo de una aplicación en varios componentes claramente identificables:

- Modelo = Capa de acceso a datos
- Vista = Capa de presentación o Interfaz de usuario
- Controlador = Capa lógica de negocio.

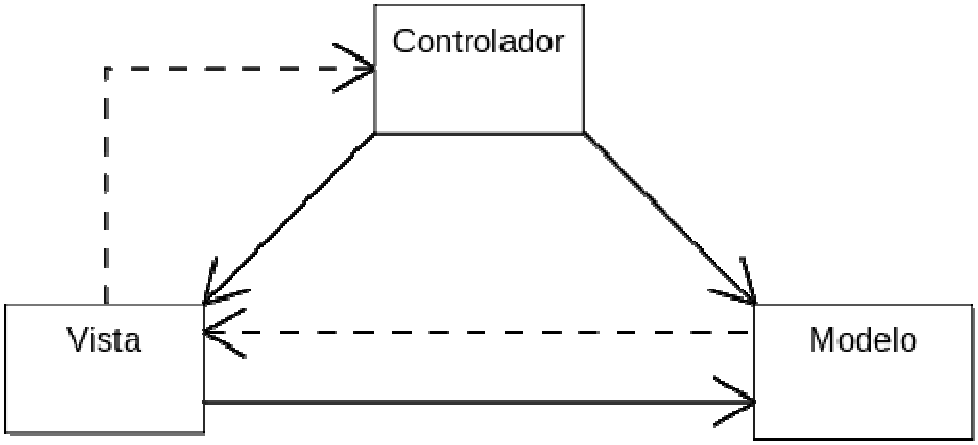

Figura 3. Modelo Vista Controlador (MVC)

La capa Modelo contiene la información con la cual trabaja el sistema, atendiendo las peticiones realizadas desde la capa Controlador. Se trata de la unidad de información mínima con la que trabaja la aplicación y representa el modelo de la aplicación.

La capa Vista representa la interfaz de usuario o el aspecto del modelo. Este componente se encarga de presentar la información enviada desde la capa Controlador.

La capa Controlador es la encargada de responder las peticiones del usuario y de transmitírselas a la capa Modelo e, incluso, a la capa Vista.

3.3.2 Funciones del Prototipo

A continuación, se muestra un listado de funcionalidades básicas que ofrecerá el prototipo de aplicación web para la gestión hospitalaria:

- Autenticación: acceso al sistema
- Gestión de las autorizaciones: cambio de perfil
- Indicadores definidos por la relación módulo-perfil, con el objetivo de gestionar la inserción, modificación y eliminación de registros.
- Alta, baja, modificación y consulta de Pacientes.
- Alta, baja, modificación y consulta de Usuarios del sistema.
- Alta, baja, modificación y consulta de Admisiones de pacientes.
- Adjuntar documentos a las fichas de los Pacientes.
- Adjuntar documentos a las fichas de los Usuarios.
- Adjuntar documentos a las fichas de las Admisiones.

3.3.3 Requisitos Funcionales

A continuación, se describen los requisitos que ha de cumplir la aplicación para alcanzar con éxito las funcionalidades deseadas:

Req(01)	La aplicación debe permitir ser usado solo por usuarios registrados.
Req(02)	La aplicación debe ser capaz de recuperar de la base de datos el listado los Pacientes.
Req(03)	La aplicación debe permitir dar de alta Pacientes, uno a uno, de manera individual.

Req(04)	La aplicación debe permitir modificar los datos de los Pacientes, uno a uno, de manera individual.
Req(05)	La aplicación debe permitir eliminar los Pacientes, uno a uno, de manera individual.
Req(06)	La aplicación debe permitir filtrar el listado de Pacientes.
Req(07)	La aplicación debe ser capaz de recuperar de la base de datos el listado los Usuarios del sistema.
Req(08)	La aplicación debe permitir dar de alta Usuarios del sistema, uno a uno, de manera individual.
Req(09)	La aplicación debe permitir modificar los datos de los Usuarios del sistema, uno a uno, de manera individual.
Req(10)	La aplicación debe permitir eliminar los Usuarios del sistema, uno a uno, de manera individual.
Req(11)	La aplicación debe permitir filtrar el listado de Usuarios del sistema.
Req(12)	La aplicación debe ser capaz de recuperar de la base de datos el listado los Admisiones.
Req(13)	La aplicación debe permitir dar de alta Admisiones, una a una, de manera individual.
Req(14)	La aplicación debe permitir modificar los datos de las Admisiones, una a una, de manera individual.
Req(15)	La aplicación debe permitir eliminar las Admisiones, una a una, de manera individual.
Req(16)	La aplicación debe permitir filtrar el listado de Admisiones.
Req(17)	La aplicación debe permitir adjuntar documentos a los Pacientes, uno a uno, de manera individual.
Req(18)	La aplicación debe permitir adjuntar documentos a los Usuarios del sistema, uno a uno, de manera individual.
Req(19)	La aplicación debe permitir adjuntar documentos a las Admisiones, una a una, de manera individual.

Req(20)	La aplicación debe mostrar mensajes del estado en que se encuentre en cada operación, lo más claros posibles.
Req(21)	La aplicación debe dar información clara y aportar soluciones constructivas ante los posibles errores que se produzcan.
Req(22)	El usuario podrá salir de la aplicación en cualquier momento, solicitando confirmación para llevar a cabo la petición.
Req(23)	Habrá una opción de salir/cancelar, para las pantallas emergentes, con la que se abandonarán las mismas sin que con ello se cierre la aplicación completamente.
Req(24)	El usuario, una vez autenticado en la aplicación, podrá acceder a cualquiera de los módulos para los que tenga autorización, sin necesidad de volver a introducir los datos de acceso, incluso si se abren nuevas pestañas, siempre y cuando pertenezcan al navegador en el que se autenticó previamente (y mantiene la sesión abierta).
Req(25)	La aplicación mostrará ayuda contextual adicional (“*ToolTips”*) al situar el puntero del ratón sobre los botones y módulos disponibles, con la intención de informar a los usuarios de la finalidad del elemento sobre el que se encuentra posicionado.
Req(26)	La aplicación debe avisar, de manera visual, remarcando el marco de los elementos de texto, así como con mensajes contextuales, de la obligatoriedad de un campo o campos determinados.

3.3.4 Requisitos de Interfaz

3.3.4.1 Interfaz con el Usuario.

Las salidas del prototipo irán destinadas a los usuario de la aplicación y serán ofrecidas a través de la pantalla, haciendo uso de cualquiera de los navegadores más utilizados actualmente: Mozilla Firefox, Chrome y Opera. (Nota: el navegador Internet Explorer presenta un tipo de incompatibilidad con la librería gráfica ExtJS que queda pendiente de ser solventada).

Para comunicarse con el navegador, los usuarios utilizarán el ratón y el teclado.

3.3.5 Requisitos No Funcionales.

3.3.5.1 Requisitos Hardware.

En cuanto a hardware se refiere, con el fin de que la herramienta sea lo más accesible posible, bastará con un ordenador personal estándar del mercado (PC), tanto para el servidor como para el cliente.

Los requisitos no funcionales que se han de cumplir son:

Req-NF(01)	Fácil de aprender.
Req-NF(02)	Fácil de usar.
Req-NF(03)	Uso fácil de recordar.
Req-NF(04)	Navegación sencilla.
Req-NF(05)	Colores adecuados.
Req-NF(06)	Usable.
Req-NF(07)	Entorno gráfico agradable.
Req-NF(08)	El sistema debe cumplir las disposiciones recogidas en la Ley Orgánica de Datos Personales y en el Reglamento de medidas de seguridad.
Req-NF(09)	El sistema debe tardar menos de 5 segundos en mostrar los resultados del módulo en el que se encuentren los usuarios.
Req-NF(10)	La base de datos que contiene toda la información de los patrones debe estar montada sobre alguna de las versiones del sistema operativo Linux.
Req-NF(11)	La aplicación debe poder ser accesible desde los navegadores más utilizados actualmente por los usuarios: Internet Explorer, Firefox Mozilla, Chrome y Safari.
Req-NF(12)	La aplicación debe poder ser accesible, al menos, desde todas las versiones y/o distribuciones de los siguientes sistemas operativo: Windows, Linux y Mac OS.
Req-NF(13)	La aplicación se instalará en el servidor sobre el sistema operativo Windows 7 home.
Req-NF(14)	La aplicación se integrará con la Base de Datos MySQL.
Req-NF(15)	La aplicación se integrará con el Servidor Web Apache.

3.3.5.2 Requisitos Software.

La aplicación deberá funcionar en ordenadores con cualquier sistema operativo.

3.4 CASOS DE USO

3.4.1 Definición de los Casos de Uso.

Según definió Jacobson [Jacobson, 1992], un "*Caso de Uso"* es un documento narrativo que describe la secuencia de eventos de un actor (agente externo) que usa un sistema para completar un proceso, es decir, una forma de usar la función que ofrece el sistema. [Ferré, UML]

Los Casos de Uso [Jacobson, 1992] describen, bajo la forma de acciones y reacciones, el comportamiento de un sistema desde el punto de vista del usuario, permiten definir los límites del sistema y las relaciones entre el sistema y el entorno. Por lo tanto, actúan como descripciones de la funcionalidad del sistema, independientes de la implementación. [INET, 02]

3.4.2 Diagrama de casos de uso.

En esta sección se muestran los Diagramas de Casos de Uso que permiten visualizar el conjunto de actores y funcionalidades disponibles identificadas para la aplicación. Este tipo de diagramas nos permite capturar información sobre cómo trabaja nuestra herramienta y está orientada a la captura de requisitos.

Actores:

- Usuario: engloba todos los perfiles definidos en la aplicación y tiene acceso a los casos de uso básicos tales como "Entrar al sistema", "Salir del sistema" y "Cambiar de perfil"

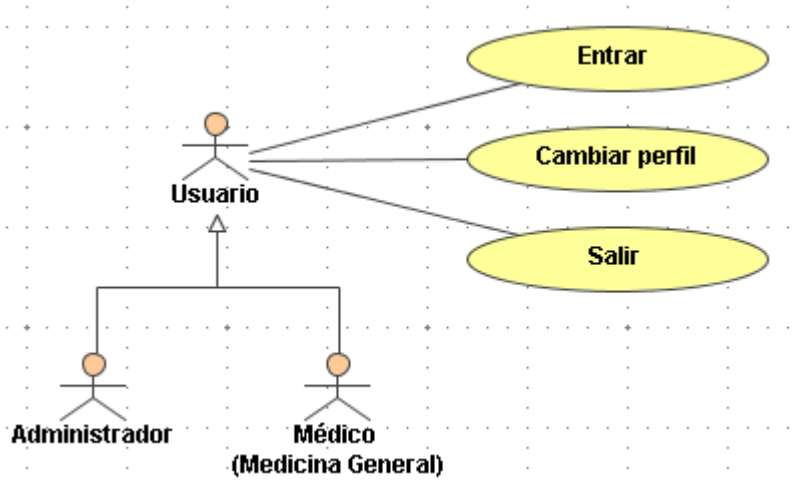

Figura 4. Diagrama de Casos de Uso – Usuario

- Médico (Medicina General): hereda los Casos de Uso descritos para el "Usuario", así como los relacionados con la gestión de Pacientes, excepto "Eliminar paciente", y los relacionados con las Admisiones.

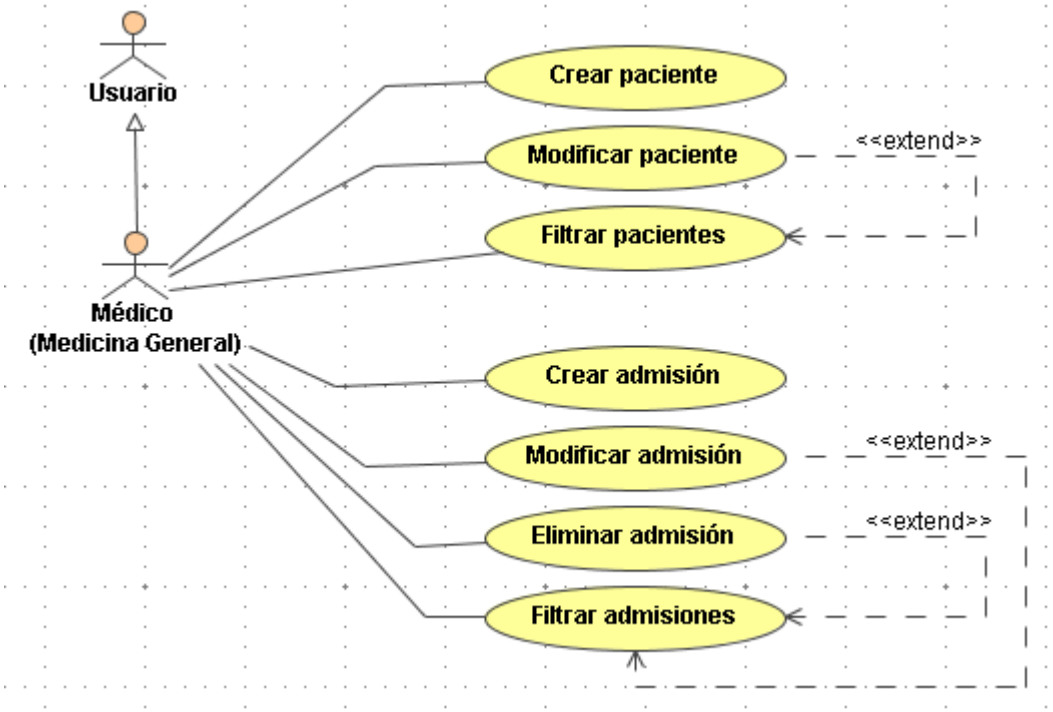

Figura 5. Diagrama de Casos de Uso – Médico (Medicina general)

- Administrador: tiene acceso a todos los Casos de Uso, incluidos los heredados por el "Usuario"

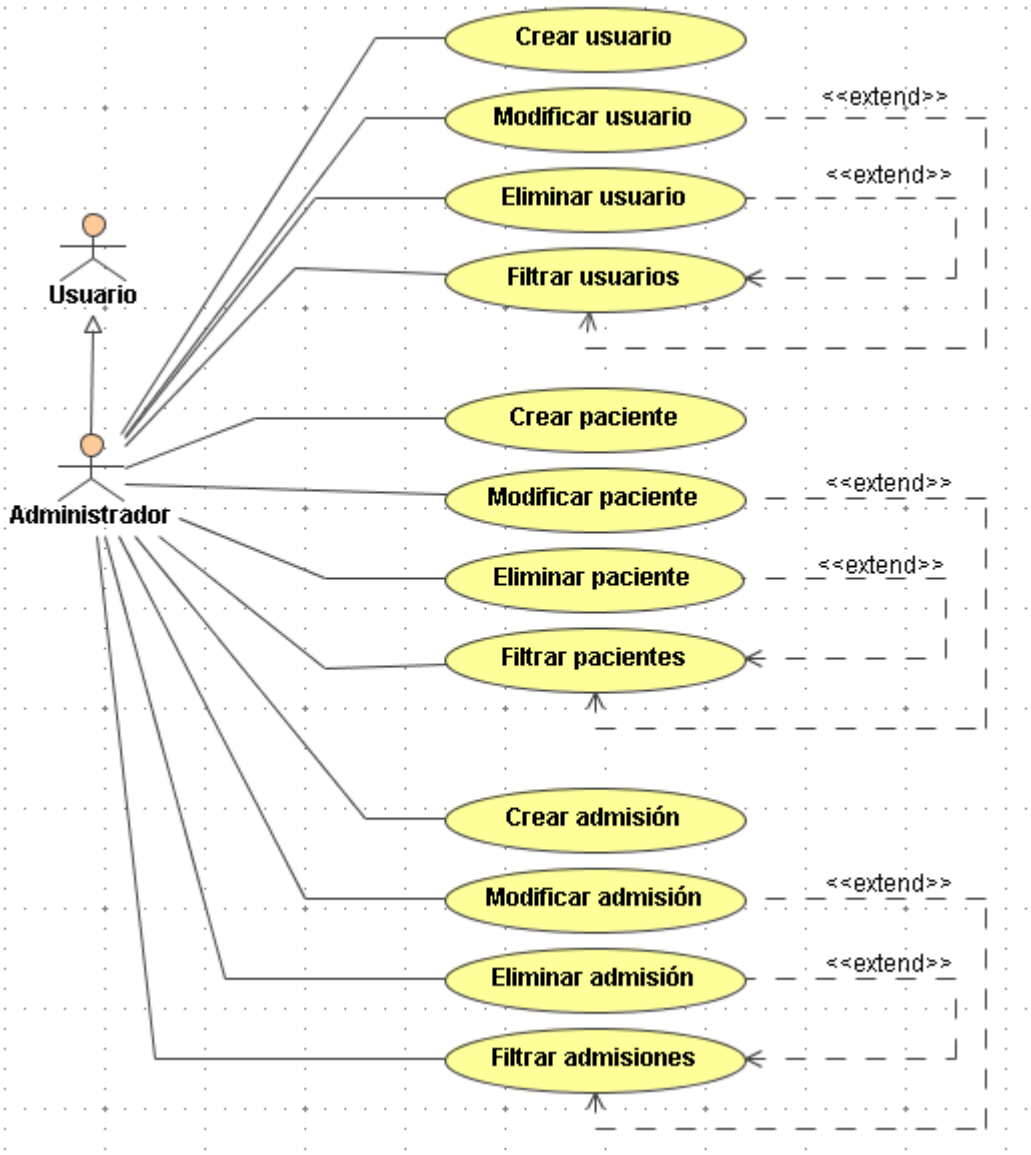

Figura 6. Diagrama de Casos de Uso – Administrador

3.4.3 Casos de Uso de Alto Nivel

Esta sección nos ofrece una visión abstracta de la funcionalidad de la herramienta mediante la descripción de los "Casos de Uso de Alto Nivel", lo cual nos permite tener una idea general y resumida de lo que se puede realizar con la herramienta.

3.4.3.1 Entrada al sistema

Caso de Uso: Entrar
Actor/es: Usuario (todos)
Tipo: Primario
Descripción: El sistema le pedirá al usuario el nombre de *Usuario* y *Contraseña* con el que desea acceder a la aplicación. El sistema comprobará que los valores introducidos son correctos. Al tercer intento incorrecto, la cuenta del usuario quedará bloqueada.

3.4.3.2 Salir

Caso de Uso: Salir
Actor/es: Usuario (todos)
Tipo: Primario
Descripción: El usuario termina su sesión en el sistema, presentándosele la pantalla para realizar, nuevamente, la autenticación a este.

3.4.3.3 Cambiar perfil

Caso de Uso: Cambiar perfil
Actor/es: Usuario (todos)
Tipo: Primario
Descripción: El usuario seleccionará un perfil de entre los disponibles y el sistema actualizará la información mostrada dependiendo de las autorizaciones que posea el perfil elegido. Puede mantenerse en el módulo desde el que se realizó la petición de cambio o cargar el módulo definido por defecto, en caso de que el usuario no tenga derechos para acceder al módulo actual con el nuevo perfil.

3.4.3.4 Usuarios

Caso de Uso: Crear usuario
Actor/es: Administrador
Tipo: Primario
Descripción: El administrador solicita la creación de un nuevo usuario, introduce los datos requeridos por el sistema, que son validados por este, comprueba que el identificador de usuario no se encuentre duplicado y crea el nuevo registro.

Caso de Uso: Modificar usuario
Actor/es: Administrador
Tipo: Primario
Descripción: El administrador selecciona un usuario de entre los disponibles y modifica uno o varios de los datos de este. El sistema valida los datos introducidos, comprueba que el identificador de usuario no se encuentre duplicado y actualiza el registro.

Caso de Uso: Eliminar usuario
Actor/es: Administrador
Tipo: Primario
Descripción: El administrador selecciona un usuario de entre los disponibles y solicita su eliminación. El sistema se encarga de quitarlo de la aplicación.

Caso de Uso: Filtrar usuarios
Actor/es: Administrador
Tipo: Primario
Descripción: El administrador solicita filtrar el listado existente de usuarios introduciendo un conjunto de valores, los cuales son evaluados por el sistema para devolver el listado resultante.

3.4.3.5 Pacientes

Caso de Uso: Crear paciente
Actor/es: Administrador, Médico (Medicina general)
Tipo: Primario
Descripción: El usuario solicita la creación de un nuevo paciente, introduce los datos requeridos por el sistema, que son validados por este, comprueba que el identificador de usuario no se encuentre duplicado y crea el nuevo registro.

Caso de Uso: Modificar paciente
Actor/es: Administrador, Médico (Medicina general)
Tipo: Primario
Descripción: El usuario selecciona un paciente de entre los disponibles y modifica uno o varios de los datos de este. El sistema valida los datos introducidos, comprueba que el identificador de usuario no se encuentre duplicado y actualiza el registro.

Caso de Uso: Eliminar paciente
Actor/es: Administrador
Tipo: Primario
Descripción: El administrador selecciona un paciente de entre los disponibles y solicita su eliminación. El sistema se encarga de quitarlo de la aplicación.

Caso de Uso: Filtrar pacientes
Actor/es: Administrador, Médico (Medicina general)
Tipo: Primario
Descripción: El usuario solicita filtrar el listado existente de pacientes introduciendo un conjunto de valores, los cuales son evaluados por el sistema para devolver el listado resultante.

3.4.3.6 Admisiones

Caso de Uso: Crear admisión
Actor/es: Administrador, Médico (Medicina general)
Tipo: Primario
Descripción: El usuario solicita la creación de una nueva admisión, introduce los datos requeridos por el sistema, que son validados por este, y lo añade.

Caso de Uso: Modificar admisión
Actor/es: Administrador, Médico (Medicina general)
Tipo: Primario
Descripción: El usuario selecciona una admisión de entre las disponibles y modifica uno o varios de los datos de esta. El sistema los valida y actualiza el registro.

Caso de Uso: Eliminar admisión
Actor/es: Administrador
Tipo: Primario
Descripción: El administrador selecciona una admisión de entre las disponibles y solicita su eliminación. El sistema se encarga de quitarla de la aplicación.

Caso de Uso: Filtrar admisiones
Actor/es: Administrador, Médico (Medicina general)
Tipo: Primario
Descripción: El usuario solicita filtrar el listado existente de admisiones introduciendo un conjunto de valores, los cuales son evaluados por el sistema para devolver el listado resultante.

4 TÉCNICAS Y MECANISMOS DE SEGURIDAD WEB APLICADOS

Con la intención de proteger la aplicación de los posibles ataques maliciosos, se ajustaron, en tiempo de ejecución, varias directivas existentes en el fichero de configuración *php.ini*, perteneciente al servidor web Apache, y se aplicaron diversas técnicas y métodos durante la codificación del prototipo.

Nota: existe una alternativa para definir diferentes directivas de seguridad, como puede ser la creación del fichero *.htaccess*. Desafortunadamente, no todas las directivas tienen pueden ser incluidas en este fichero.

4.1 DIRECTIVAS DE SEGURIDAD

A continuación se muestra el listado de directivas ajustadas y funciones aplicadas, así como unos breves comentarios con las razones que motivaron sus usos:

4.1.1 php.ini

4.1.1.1 Configuración en tiempo de ejecución

- error_reporting = E_ALL [INET, 10], [INET, 11]

  Con esta directiva a "E_ALL" se establece el nivel de notificación de errores. Este valor permite mostrar todos los errores y advertencias soportados. Estos mensajes se registraran en un archivo de texto de errores de registro especificado en "error_log".

- display_errors = 0 [INET, 10]

  Con esta directiva a "0" se establece que los errores no serán impresos por pantalla como parte de la salida y, por lo tanto, quedarán ocultos al usuario.

- session.use_trans_sid = 0 [INET, 15]

  Con esta directiva a "0" se evita la propagación del identificador de sesión del usuario (SID) por la barra de direcciones.

- display_startup_errors = 0 [INET, 10]

  Con esta directiva a "0" se establece que los errores que ocurren durante la secuencia de arranque de PHP no se muestren por pantalla y, por lo tanto, quedarán ocultos al usuario.

- log_errors = 1 [INET, 10]

  Con esta directiva a "1" indicamos que los mensajes de error del script serán registrados en el fichero de texto de registro establecida en la directiva "error_log".

- error_log = [ruta]/miwErrLog.txt

  Esta directiva contiene la ruta en la cual se encuentra el archivo donde los errores del script serán registrados.

- expose_php = 0

  Con esta directiva a "0" se previene que las cabeceras que se envían junto a las páginas muestren si se está ejecutando código PHP y/o su versión.

- session.cookie_secure = 1 [INET, 15]

  Con esta directiva a "1" se especifica que las cookies serán enviadas **sólo** sobre conexiones seguras.

- magic_quotes_sybase = 0 [INET, 13]

  Esta directiva tiene valor "0" para evitar interferencias con la directiva *magic_quotes_gpc*. En caso contrario de que tuviera definido el valor "On", se ignoraría completamente el valor de la directiva *magic_quotes_gpc*, haciendo que ni las ' (comillas simples), " (comillas dobles), \ (barra invertida) ni NULL's fueran escapadas con una barra invertida de forma automática, ya que se recomienda escapar caracteres según se requiera.

- session.referer_check = https://localhost/pfm/ [INET, 15]

  Esta directiva contiene como valor una cadena que sirve de base para comprobar las peticiones de entrada a través de la variable HTTP_REFERER enviada por los clientes. En el caso de que el valor de esta directiva no se encuentre en la referencia enviada por el cliente, el identificador de sesión embebido será marcado como no válido.
  Esta directiva sirve para mitigar ataques de *Cross-Site Request Forgery (CSRF)* y de robos de sesiones.

- session.cookie_httponly = 1 [INET, 15]

  Esta directiva con valor a "1" sirve para declarar a las cookies accesibles sólo a través del protocolo HTTP. Esto significa que las cookies **no** serán accesibles por lenguajes de script, tales como JavaScript, ayudando a mitigar los ataques de robos de identidad.
  Nota: no está soportado por todos los navegadores

- session.use_only_cookies = 1 [INET, 15]

  Con esta directiva a "1" se indica que **solo** se utilizarán cookies para almacenar el identificador de sesión en la parte del cliente. Este ajuste previene los ataques relacionados con el paso de identificadores de sesión por la URL (método GET), como por ejemplo el ataque por *fijación de sesión*.

- session.cookie_lifetime = 1200 [INET, 15]

  El valor asignado a esta directiva especifica el tiempo de vida en segundos de las cookies enviadas al navegador, cuyo el valor corresponde al siguiente cálculo: [60 segundos]*[20 minutos] = 1200 segundos)

  La marca de tiempo de caducidad se establece relativa a la hora del servidor, la cúal no es necesariamente la misma que la hora del navegador del cliente.

4.1.1.2 Configuración del fichero php.ini

- allow_url_fopen = 0 [INET, 09]

  Con esta directiva a "0" se previenen los posibles ataques por *inclusión remota de archivos* (RFI), haciendo que ficheros pasados por URL externas puedan ser utilizados en las funciones *include(), include_once(), require() y/o require_once()*. De este modo, las funciones mencionadas solo permitirán la llamada de archivos que residan en el directorio web en la cual se encuentre alojada la aplicación web.
  Así pues, se desea enfatizar en la importancia de esta directiva.

- magic_quotes_gpc = 1 [INET, 12]

  El alcance de esta directiva afecta a las operaciones GPC (Get/Post/Cookie) y establece que todas las ' (comillas simples), " (comillas dobles), \ (barra invertida) y NULL's sean escapadas con una barra invertida de forma automática.

4.1.2 Configuración del fichero .htaccess

- register_globals = 0 [INET, 14]

  Con esta directiva a "0", no se registran las variables EGPS (Entorno, GET, POST, Cookie, Server) como globales.
  php_flag register_globals off

4.2 FUNCIONES

- session_name() [INET, 16]

  Esta función se utiliza para obtener o establecer el nombre de la sesión actual.
  Con ella, cambiamos el nombre predeterminado "PHPSESSID" por "MIW_PFM_[nuestra dirección IP, sin separadores]

4.3 SSL

Uno de los muchos riesgos existentes en Internet, relacionado con la transferencia de información, son las escuchas no autorizadas en el medio de comunicación empleado (Eavesdropping).

Para mitigar este riesgo, se activó el cifrado de la información (SSL) del XAMPP y se generó un certificado SSL auto-firmado por el servidor de aplicaciones web Apache utilizado por este, solo con validez práctico/académica. [INET, 17]

Los pasos que se siguieron para activar el servicio de cifrado de información SSL y para generar el certificado fueron:

1. Para activar el servicio SSL de Apache se editó el fichero ***php.ini***, gracias a la siguiente directiva:

   extension=php_openssl.dll

2. Para obtener el certificado digital, utilizo el propio servidor Apache. En primer lugar, se abrió una ventana de comandos MS-DOS y se ejecutó el comando ***makecert*** existente en el directorio ***/xampp/apache***.

3. Se introdujo la contraseña que servirá para desencriptar la clave privada generada para el certificado.

4. Se rellenaron algunos de los siguientes campos informativos solicitados por el proceso, tales como: Estado o Provincia, Localidad, Organización, Sección o Unidad, DNS o dirección IP, email, etc... Para el campo del DNS se insertó la dirección IP en la que se encuentra el servidor web empleado para hacer funcionar la aplicación y que es la misma que la utilizada por el navegador cuando se accede a las páginas de la aplicación.

5. Se finalizó el proceso y se generó el certificado SSL auto-firmado.

Debido a que el certificado es auto-firmado (no está firmado por una Autoridad de Certificados (CA) bien conocida), cuando se navega entre las páginas protegidas de la aplicación se recibe un mensaje de advertencia. Para eliminar esta advertencia, se debe importar el certificado recién creado como una CA de confianza en los navegadores empleados.

4.4 MÉTODOS Y TÉCNICAS DE PROGRAMACIÓN APLICADAS

A continuación se muestra un listado de los aspectos de seguridad web más importantes que se tuvieron en cuenta y de los métodos y técnicas que se aplicaron para cumplirlos:

4.4.1 Autenticación de usuarios al sistema

Los usuarios, para poder acceder al sistema, deben introducir un identificador de usuario y contraseña validos.

Para salvaguardar la integridad de las contraseñas existentes en la base de datos y evitar que puedan recuperarse en texto claro, se hizo uso de los algoritmos hash *MD5* y *SHA,* cuyo objetivo es obtener las huellas de cada una de las contraseñas y cuya principal propiedad es la unidireccionalidad, y de un algoritmo "*Salt*" aplicado al hash para dificultar encontrar las contraseña.

El valor "*Salt*" almacena el hash sin cifrar y se utilizará más adelante para la comprobación de contraseñas.

El esquema de la implementación del algoritmo queda como sigue a continuación:

(1) salt = MD5 ([contraseña introducida] + "¡~#|%·=(76~:")
(2) resultado = MD5 (salt + [contraseña introducida] + salt)

(1) Al valor de la contraseña original se le suma la cadena de texto **"¡~#|%·=(76~:"** y, al resultado de dicha combinación, se le aplica el algoritmo MD5, obteniendo como resultado final el valor de la variable *SALT*.
(2) El valor obtenido para la variable SALT en el punto (1) es combinado, por ambos extremos, con el valor de la contraseña original. Al resultado de dicha combinación se le aplica, nuevamente, el algoritmo MD5 generando el valor final para la contraseña introducida.

Una vez el usuario ha introducido un identificador de usuario y contraseñas válidos, el servidor PHP generará un identificador de seguridad (SID) para el usuario autenticado, y tendrá vigencia durante las siguientes 20 minutos o hasta que el usuario cierre el navegador. El valor del SID generado es único e identificará al usuario durante el uso de la aplicación.

Además, se implementó un control adicional en el servidor web para asegurar la navegación del usuario autenticado y prevenir el robo de sesión:

- Una vez que el usuario se autentica en el sistema, el servidor web actualiza las siguientes variables de sesión que se usarán para validar al usuario durante la navegación entre los módulos del sistema:

  $_SESSION['USER_AUTHENTICATED'] = 1

  Si, en algún momento durante la navegación entre las páginas de la aplicación, el valor de esta variable de sesión es diferente a "1", el servidor cancelará automáticamente la sesión del usuario y expulsará al supuesto intruso del sistema, mostrando de nuevo la pantalla de petición de usuario y contraseña.

4.4.2 Autorizaciones de los usuarios dentro del sistema

El sistema se diseño e implementó para que la gestión de las autorizaciones de los usuarios fuera manejada por medio del concepto de perfiles.

A nivel global para todos los usuarios, los perfiles ofrecen derechos de acceso a los módulos de la aplicación para los que se les ha dado permisos así como derechos sobre las operaciones sobre los registros disponibles en estos módulos. Es decir, los usuarios que acceden con el mismo perfil, tienen derechos para acceder a los mismos módulos del sistema para los que su perfil tiene autorización y tienen disponibles las mismas operaciones sobre los datos, las cuales, se limitan actualmente a la creación, modificación y eliminación de registros.

4.4.3 Validación de los datos

La validación de datos incluye evaluar sintáctica y/o semánticamente cualquier dato de entrada en una aplicación. Por el alcance definido para este proyecto, en cuanto a que se pretende implementar un prototipo de aplicación para la gestión hospitalaria, se incluyó como punto de partida la validación sintáctica de los siguientes campos:

- Documento de Identidad: específicamente para el tipo de documento DNI, validando que el dato introducido cumpla la siguiente estructura:
  NNNNNNNNNL – N: Número y L: Consonante.
- Email: El valor de este campo debe cumplir con las siguientes características:
  /^([A-Za-z0-9_\-\.])+\@([A-Za-z0-9_\-\.])+\.([A-Za-z]{2,4})$/;
- Número de teléfono: Este campo debe ser un número de 9 dígitos no negativo.

Estas validaciones se gestionan internamente a través de la librería gráfica ExtJS [INET, 27], librería utilizada para desarrollar la interfaz de usuario de la aplicación, cuyas reglas pueden ser configurables según las necesidades de cada uno de los formularios de la aplicación y de la información que manejen.

Por otro lado se incluyó como de validación semántica la comprobación de que el campo "Fecha de Nacimiento" del asegurado no podría dar como resultado una edad mayor a 150 años, como tampoco se podrían registrar asegurados con fecha de nacimiento posterior al día de registro.

Cabe aclarar que la implementación de validaciones a otros datos de entrada podría ser valorada por desarrollos futuros sobre este prototipo.

Además de lo mencionado anteriormente y ya buscando asegurar la transferencia de datos, se utilizó el método POST durante la navegación entre las páginas de la aplicación. Aunque esto no evita que un atacante pueda manipular los datos durante su transferencia (para lo cual podría hacer uso, por ejemplo, de un proxy como "Burp Suite"), si añade una cierta dificultad con la intención de mitigar ataques que puedan ser originados en los datos de entrada.

Finalmente, se implementaron las siguientes funciones a nivel de servidor web: [de Frutos, 2012]

- "*mysql_real_escape_string*", para mitigar ataques de SQL Injection.

  Ejemplo: los datos introducidos en el formulario de creación de nuevos pacientes (nombre, apellidos, fecha de nacimiento, etc…) son enviados al servidor y este, por medio de la función PHP "*mysql_real_escape_string*", antepone una barra invertida o backslash a los siguientes caracteres::

  ‘, “, \n, \, \r, *x00 (null)* **y *x1a* **(sub)**

  - Código PHP perteneciente al fichero "Controller.php" del módulo "Pacientes":

```
$patient = new Patient();
$result = $patient->addRecord(
    mysql_real_escape_string(htmlspecialchars($_POST['patientFirstname'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientLastname'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientDateOfBirth'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientDocType'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientDocIdentification'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientGender'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientInsuranceID'])),
    mysql_real_escape_string(htmlspecialchars($_POST['PatientEmail'])),
    (int)$_POST['PatientTelephoneNumber']
);
```

- "*htmlspecialchars*", para mitigar ataques de SQL Injection y XSS.

  Ejemplo: los datos introducidos en el formulario de edición de Usuarios (nombre, apellidos, cuenta, contraseña, etc…) son enviados al servidor y este, por medio de la función PHP "*htmlspecialchars*", realiza la conversión de los siguientes caracteres por entidades HTML:

  '&' se convierte en '&'
  '"' se convierte en '"'
  ''' se convierte en '''
  '<' se convierte en '<'
  '>' se convierte en '>'

  - Código PHP perteneciente al fichero "Controller.php" del módulo "Usuarios":

```
$user = new USR();
$result b= $user->updateRecord(
    (int)$_POST['UserID'],
    mysql_real_escape_string(htmlspecialchars($_POST['UserPublicName'])),
    mysql_real_escape_string(htmlspecialchars($_POST['UserPublicLastname'])),
    mysql_real_escape_string(htmlspecialchars($_POST['UserLoginname'])),
    mysql_real_escape_string(htmlspecialchars($_POST['UserPassword'])),
    mysql_real_escape_string(htmlspecialchars($_POST['UserActive'])),
    mysql_real_escape_string(htmlspecialchars($_POST['UserDefaultModule'])),
    mysql_real_escape_string(htmlspecialchars($_POST['UserDefaultProfile'])),
    mysql_real_escape_string(htmlspecialchars($_POST['UserFlagMedical']))
);
```

- Las características de estas funciones se encuentran comentadas con mayor detalle en los apartados "Cross site Scripting" y "SQL Injection" descritos posteriormente.

4.4.4 Seguridad con los ficheros

Para prevenir el acceso, invocación y/o ejecución de ficheros alojados en el servidor web se aplicaron los siguientes mecanismos de seguridad: [de Frutos, 2012]

- Para los ficheros subidos por medio de la aplicación, se definió una ubicación para alojarlos que se encuentra fuera del directorio raíz de la aplicación, es decir, se situaron en un directorio cuyo acceso no se encuentra disponible por URL, pero al que sí tiene acceso la aplicación.

- Se prohibió la subida de ficheros al servidor de documentos con las siguientes extensiones: *EXE, PHP, JS, ASP, ASPX, CLASS, JAVA, BAT y SQL.*

- Se evitó la carga de archivos externos a través de las funciones *include* y/o *require* de PHP, eliminando la posibilidad de ejecución de ficheros peligrosos o dañinos por parte de un atacante.

- Se hace uso del método POST para así reducir la posibilidad de ataques de carga y/o ejecución de ficheros, como también el acceso a otros niveles o directorios no permitidos, utilizando "*Directory traversal"* (\..\..\), el cual permite acceder a cualquier directorio de nivel superior sin ningún control.

- No se aplicaron mecanismos para el ocultamiento de ficheros dado que las únicas extensiones utilizadas por la aplicación fueron "php", "js" y "css". Aún así, no se descarta su aplicación en el futuro si fuese necesario, como por ejemplo, para ocultar extensiones "inc".

4.5 TIPOS DE AMENAZAS MITIGADAS

El siguiente listado muestra varias de las principales amenazas web existentes y las técnicas y métodos de seguridad aplicados para evitarlos o mitigar sus efectos:

4.5.1 Cross-Site Scripting (XSS)

Para mitigar ataques XSS se aplicaron los siguientes mecanismos de seguridad: [de Frutos, 2012]

- El uso del método POST para aportar dificultad en los ataques XSS.

- Se aplicó el método "*htmlspecialchars*" en el servidor web, para sustituir los caracteres <, >, ‘, “, & por entidades HTML (&lt, &gt, &#039, &quot, &amp).

- Se configuraron las directivas de seguridad "*session.use_only_cookies*", "*session.cookie_httponly*", "*session.cookie_secure*" y "*session.cookie_lifetime*" (para más detalles, ver las directivas descritas en el punto 4.1.1 del presente documento).

  Ejemplo:

  ini_set('session.use_only_cookies', 1);
  ini_set('session.cookie_httponly', 1);
  ini_set('session.cookie_secure', 1);
  ini_set('session.cookie_lifetime',1200); *[1200 segundos = 20 minutos]*

4.5.2 SQL Injection

Para mitigar ataques SQL Injection se aplicaron los siguientes mecanismos de seguridad: [de Frutos, 2012]

- Nuevamente, el uso del método POST aporta dificultad para los ataques XSS.

- Se utilizó la función "*mysql_real_escape_string*" de PHP para escapar los caracteres especiales en una cadena de texto, evitando que puedan ser usadas como una sentencia SQL.

- Se aplicó la conversión explícitamente por medio de la función "*int*" de PHP para asegurarnos del envío de datos numéricos enteros (de igual manera, se podría haber utilizado la función "*intval*" de PHP).

  Ejemplo: el "identificador" y el "número de teléfono" introducidos en el formulario de edición Pacientes son convertidos, de manera explícita en el servidor, a valores enteros por medio de la función PHP "*int*".

  - Código PHP:

```
$patient = new Patient();
$result = $patient->updateRecord(
    (int)$_POST['patientID'],
    mysql_real_escape_string(htmlspecialchars($_POST['patientFirstname'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientLastname'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientDateOfBirth'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientDocType'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientDocIdentification'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientGender'])),
    mysql_real_escape_string(htmlspecialchars($_POST['patientInsuranceID'])),
    mysql_real_escape_string(htmlspecialchars($_POST['PatientEmail'])),
    (int)$_POST['PatientTelephoneNumber']
);
```

  - Algunos ejemplos de la conversión explicita de estos valores de entrada a valores enteros son:

    (int)(3.5) => 3
    (int)("texto") => 0

- Para evitar el acceso a la base de datos con el usuario "*root*", creado por defecto por XAMPP para el gestor de la base de datos MySQL, se generó un nuevo usuario para la base de datos "MIW" llamado "*miwuser*", con privilegios, únicamente para realizar consultas, inserciones, modificaciones y borrado de datos, prohibiendo la ejecución del resto de operaciones tales como modificación de las tablas de la base de datos, consulta sobre su estructura, etc...

- Para evitar ofrecer información a los usuarios, a través del navegador, en caso de producirse un error en tiempo de producción, que pueda dar pistas sobre cierta estructura de la base de datos y/o de consultas implementadas en el servidor web, se configuraron las siguientes directivas de seguridad: "*display_errors*", "*error_reporting*", "*log_errors*" y "*error_log*" (para más detalles, ver las directivas descritas en el punto 4.1.1 del presente documento).

4.5.3 Robo de sesiones

Para mitigar el robo de sesiones se aplicaron los siguientes mecanismos de seguridad: [de Frutos, 2012]

- Se hace uso del método POST para aporta dificultad en los ataques de robo de sesiones, ya que utilizando el método GET, se deja abierta la posibilidad de que un atacante acceda al identificador de sesión fácilmente, ya sea a través del historial del navegador, o simplemente siendo visible en la URL, mitigando también la posibilidad de ataques por fijación de sesión ("*Sesión fixation*").

- Se cambió el nombre *PHPSESSID* del identificador SID por otro generado dinámicamente por la aplicación, en tiempo de ejecución, a través de la función "*session_name*". Con esto se pretende reducir el riesgo de captura de una variable conocida por un atacante y por ende un ataque de este tipo.

- Se hizo uso de un certificado SSL para asegurar la transferencia cifrada de información entre el cliente y el servidor.

- Se hizo uso del método "*session_destroy*" de PHP, para eliminar la sesión creada anteriormente, cuando el usuario cierra su sesión o cuando la aplicación encuentra una situación anómala que deba ser contenida y controlada.

- Aunque se ha usado el método POST para el desarrollo de este proyecto, se configuraron las directivas de seguridad "*session.use_trans_sid*", "*session.use_only_cookies*" y "session.cookie_lifetime", para evitar la propagación del SID por la barra de direcciones, mitigando así un ataque por fijación de sesión en caso de que en desarrollos futuros se requiera el uso del método GET (para más detalles, ver las directivas descritas en el punto 4.1.1 del presente documento).

- Se mantuvieron los valores por defecto para las siguientes directivas de seguridad que se encargan de configurar el "garbage" del servidor web: "*session.gc_divisor*", "*session.gc_probability*" y "*session.gc_maxlifetime*".

- Se realizó una comprobación adicional con las variables "*HTTP_USER_AGENT*" y "*REMOTE_ADDR*" con el objetivo de verificar la autenticidad del usuario:

  $_SESSION['HTTP_USER_AGENT'] = $_SERVER['HTTP_USER_AGENT']
  $_SESSION['REMOTE_ADDR'] = $_SERVER['REMOTE_ADDR']

  Si los valores almacenados en las correspondientes variables de la sesión del usuario ($_SESSION) no coinciden con las almacenadas en las variables del servidor ($_SERVER), la aplicación cerrará la sesión del usuario y lo expulsará del sistema.

5 ANÁLISIS

Esta sección muestra el comportamiento de la herramienta, observado como una caja negra, de manera visual mediante el Modelo Conceptual. Se trata de un Diagrama de Estructura Estática que consiste en identificar y representar los conceptos reales, no de componentes software, que conforman el dominio del proyecto y su objetivo es aumentar la comprensión del problema

5.1 MODELO CONCEPTUAL

En la siguiente representación del *Modelo Conceptual* se exponen los conceptos, observados del mundo real, que tienen relevancia dentro de los límites o dominio del presente proyecto.

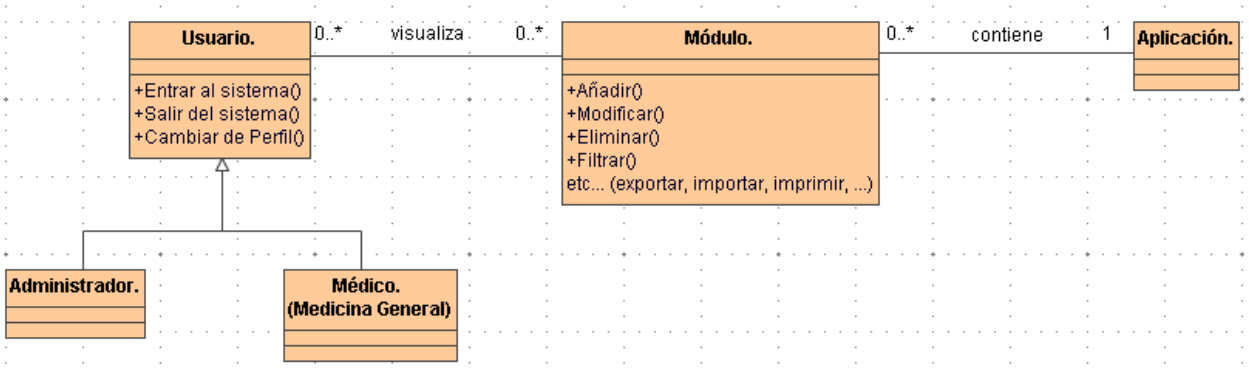

Figura 7. Modelo Conceptual

De la figura anterior se puede deducir que un usuario puede acceder a varios Módulos y hacer uso de las operaciones disponibles en ellos. Es importante tener en cuenta que los perfiles definen los derechos de visualización de los módulos. Una vez dentro de un módulo, el usuario tendrá disponibles un listado de acciones que dependen de los derechos de ejecución que se le hayan asignado, de manera específica, a los perfiles por cada módulo.

6 DISEÑO

Esta sección comprende el *Modelo o Diseño de la Herramienta Software* enfocado a la necesidad que se pretende aportar, aplicando algunas de las normas básicas de *Usabilidad* [Dix-HCI, 2003], y teniendo en cuenta las tecnologías de las que se dispusieron para alcanzar el éxito del proyecto.

El Usuario necesita tener conexión a internet y un navegador web. No se requiere de un equipo especialmente potente.

Para el objeto del *Proyecto de Fin de Máster (PFM)*, tanto la base de datos como el servidor de aplicaciones web "Apache" se han instalado y configurado sobre un sistema operativo Windows 7 pero sería aconsejable y recomendable, por cuestiones de seguridad y de costes, realizar su instalación y configuración sobre un sistema operativo Linux.

Se emplea el diseño por capas *MVC (modelo, vista, controlador)* con el objetivo de dividir la funcionalidad de la aplicación, aportando una alta cohesión, un bajo acoplamiento y ofreciendo facilidad a la hora de evolucionar la solución propuesta.

A continuación, se muestra un ejemplo del diseño de comunicación establecido, a nivel interno:

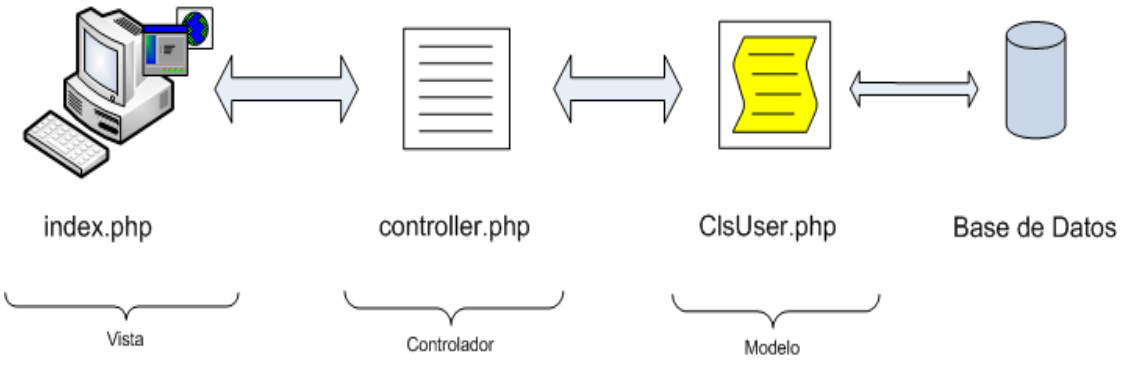

Figura 8. MVC – Modelo, Vista, Controlador

La figura anterior muestra como el fichero ***index.php***, responsable de la interfaz de usuario, de la presentación de la información y de la recepción de las acciones de los usuarios, transmite las peticiones de estos al fichero ***controller.php***, el cual se encarga de orquestar las llamadas necesarias a todas las clases implicadas para satisfacerlas. En la figura anterior se aprecia como el fichero controller.php se comunica con el fichero ***ClsUser.php***, el cual se responsabiliza de las operaciones sobre la base de datos (CRUD = Create, Read, Update, Delete). Este último responderá las peticiones recibidas desde el fichero controller.php y, a su vez, este responderá las peticiones realizadas por el fichero index.php, según sea el caso y la naturaleza de dichas solicitudes.

6.1 DIAGRAMA DE CLASES DE DISEÑO

En el siguiente *Diagrama de Clases* se captura, especifica y muestra el vocabulario del proyecto (elementos, relaciones y estructura), de una manera clara y sencilla, fundamento del comportamiento de la herramienta software.

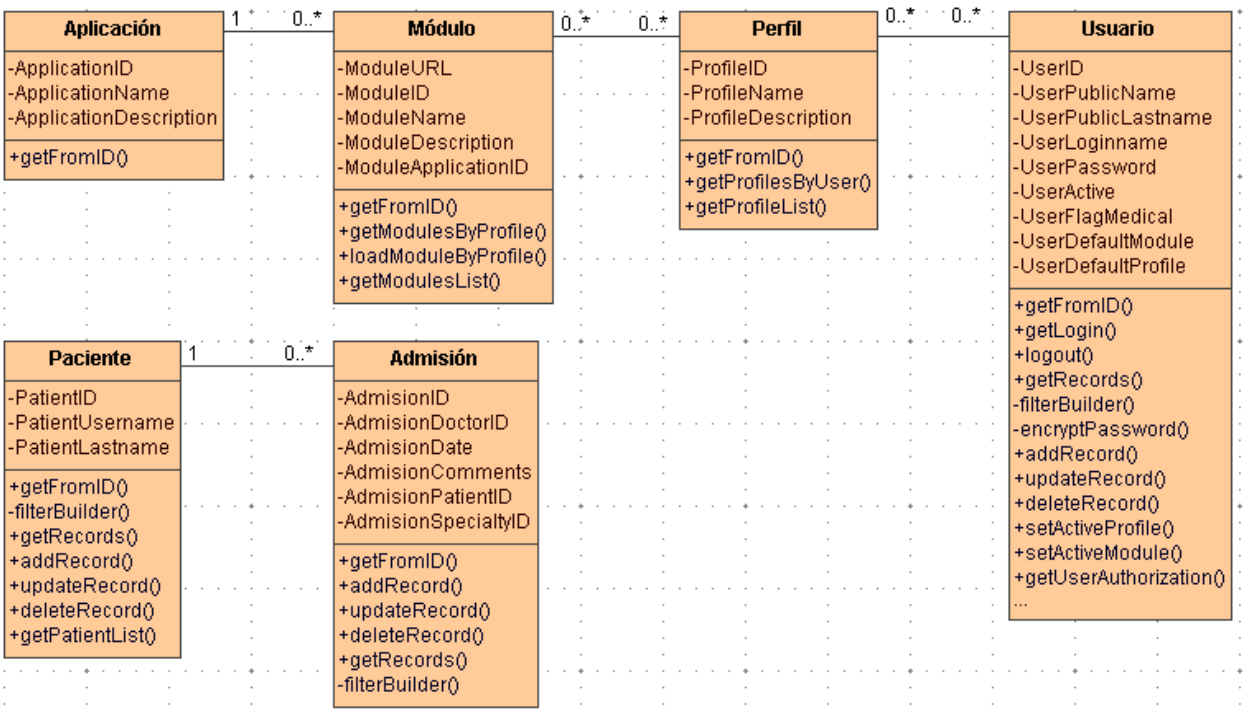

Figura 9. Diagrama de Clases

De la figura anterior se puede interpretar que un usuario puede tener asignados varios perfiles, los cuales pueden tener derecho para visualizar varios módulos. Estos módulos pertenecen a una única aplicación.

Por otro lado, un paciente puede tener asignadas varias admisiones. Estas admisiones podrán ser gestionadas por los usuarios que cuenten con los perfiles y derechos de ejecución mínimos indispensables para el módulo en cuestión.

6.2 DIAGRAMAS DE ESTADOS

Los diagramas de estado muestran el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación en respuesta a eventos (por ejemplo, mensajes recibidos, tiempo rebasado o errores), junto con sus respuestas y acciones. También ilustran qué eventos pueden cambiar el estado de los objetos de la clase. [INET, 03]

Estos diagramas nos sirven para formalizar y visualizar la secuencia de acciones del actor y reacciones del sistema a través de varios elementos básicos:

- Transiciones: acciones del actor o actores y reacciones del sistema de un Caso de Uso
- Estados: nombre descriptivo de la situación alcanzada al finalizar una transición.

6.2.1 Diagrama de Estado: Caso de Uso "Entrar"

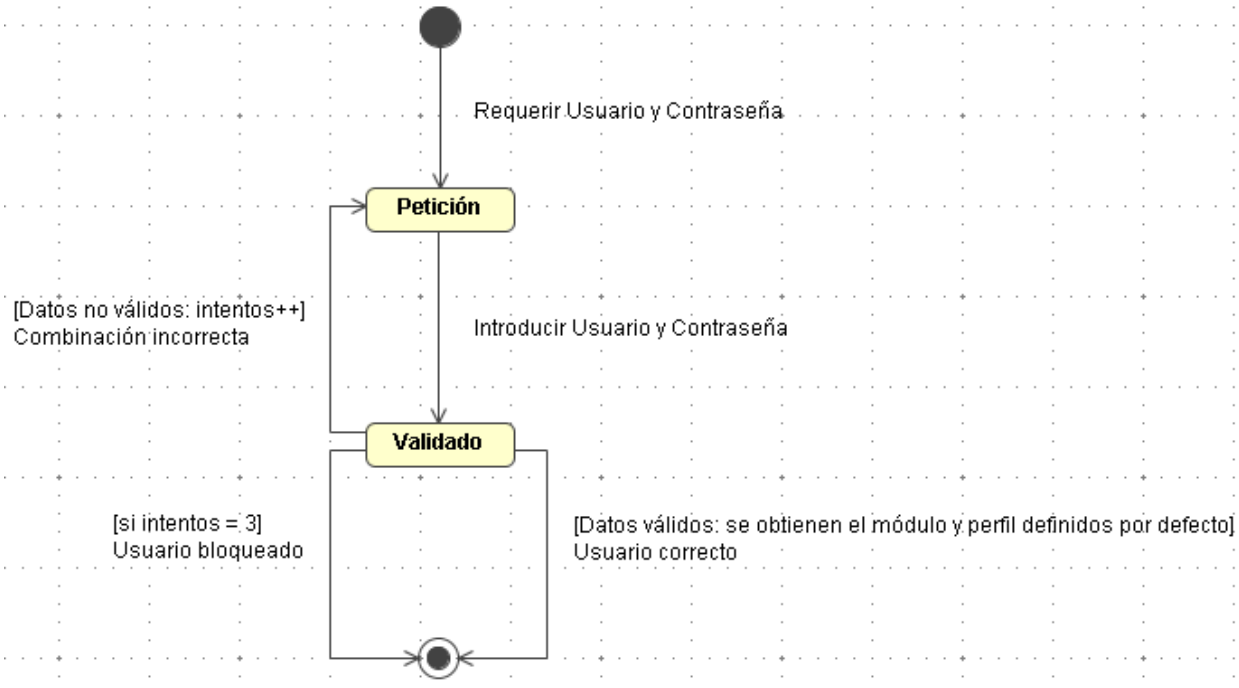

Figura 10. Diagrama de Estado: Entrar

6.2.2 Diagrama de Estado: Caso de Uso "Salir"

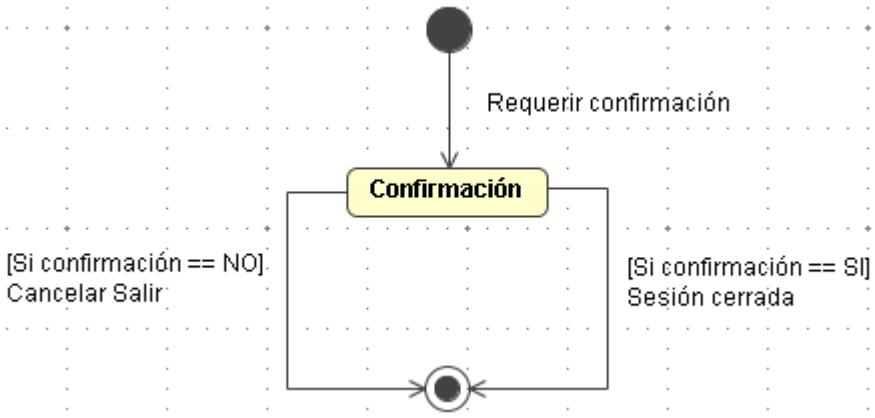

Figura 11. Diagrama de Estado: Salir

6.2.3 Diagrama de Estado: Caso de Uso "Cambiar perfil"

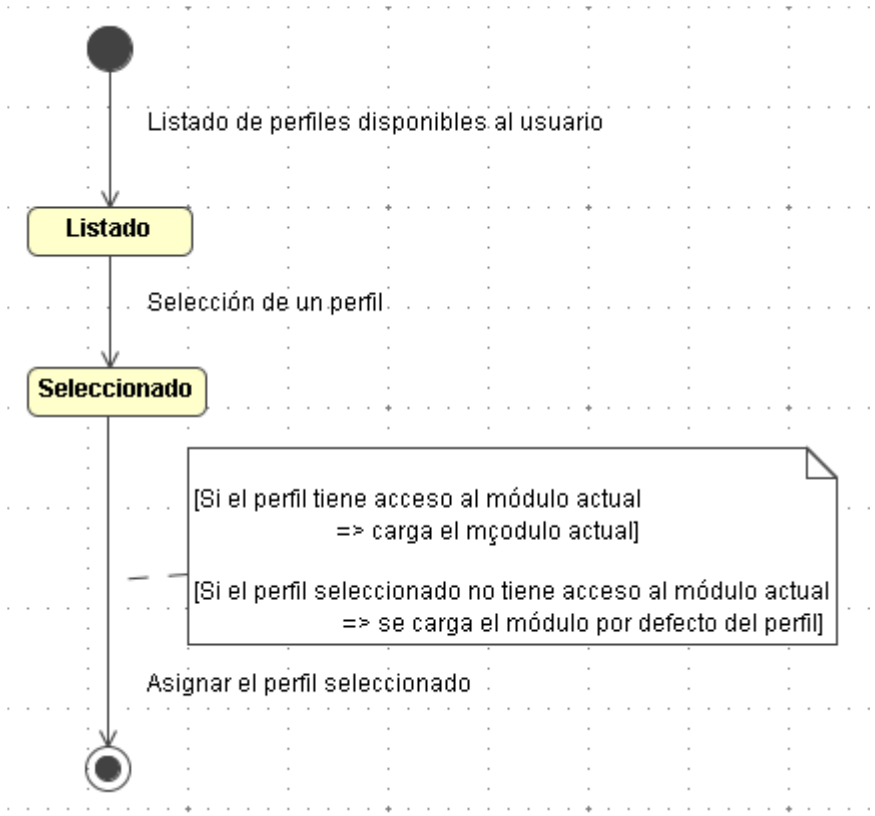

Figura 12. Diagrama de Estado: Cambiar perfil

6.2.4 Diagrama de Estado: Caso de Uso "Crear paciente"

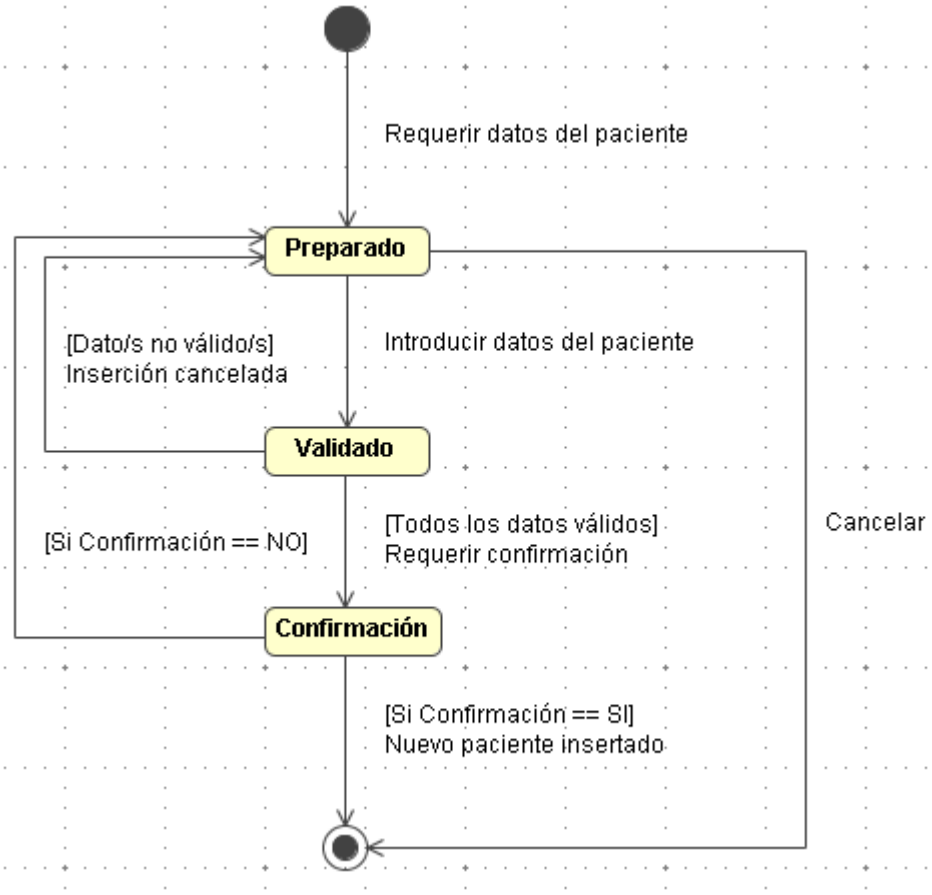

Figura 13. Diagrama de Estado: Crear paciente

6.2.5 Diagrama de Estado: Caso de Uso "Modificar paciente"

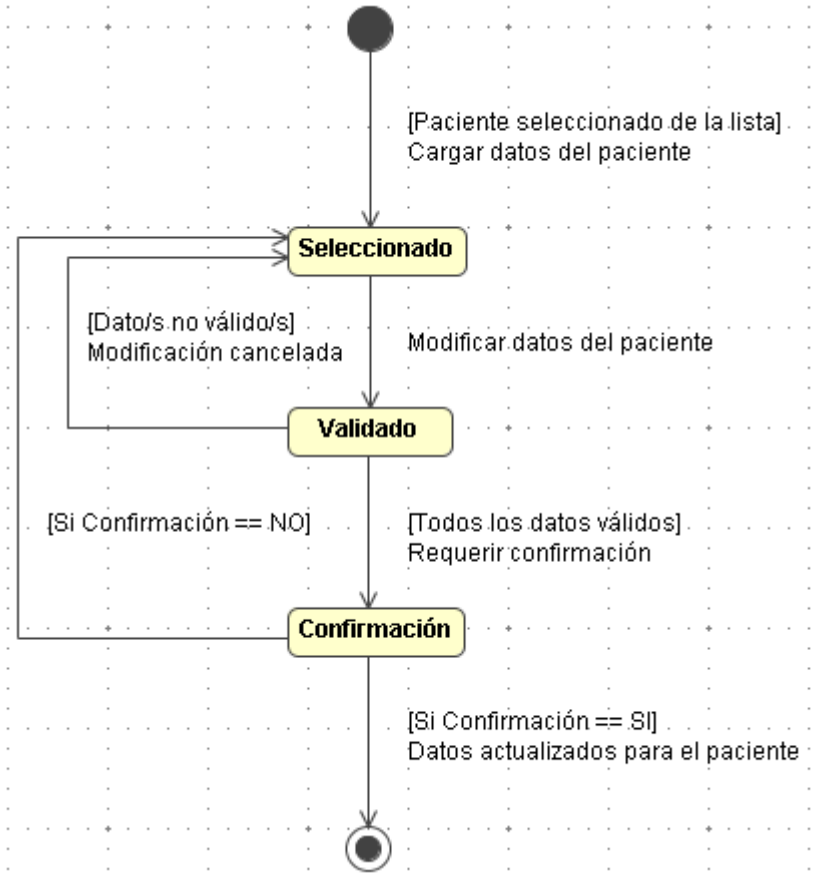

Figura 14. Diagrama de Estado: Modificar paciente

6.2.6 Diagrama de Estado: Caso de Uso "Eliminar paciente"

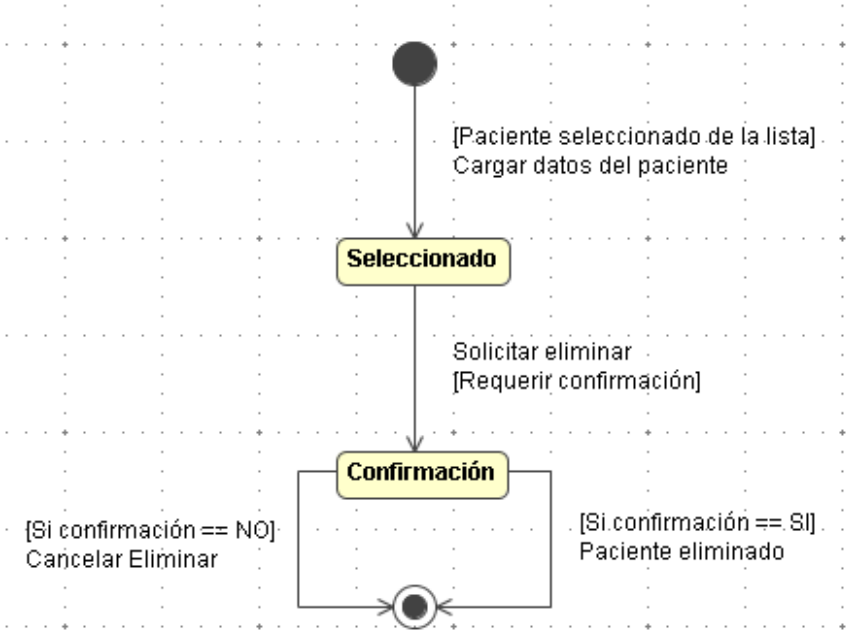

Figura 15. Diagrama de Estado: Eliminar paciente

6.2.7 Diagrama de Estado: Caso de Uso "Filtrar pacientes"

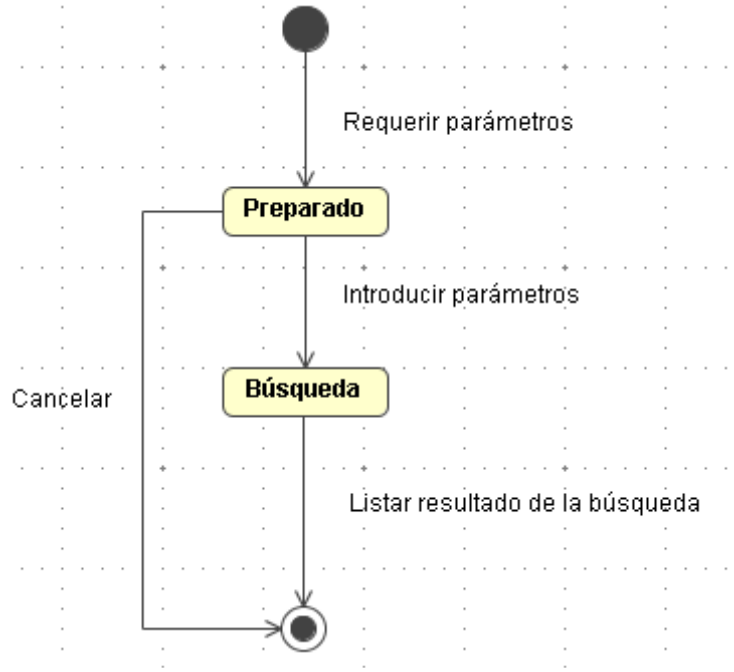

Figura 16. Diagrama de Estado: Filtrar pacientes

6.2.8 Diagrama de Estado: Caso de Uso "Crear usuario"

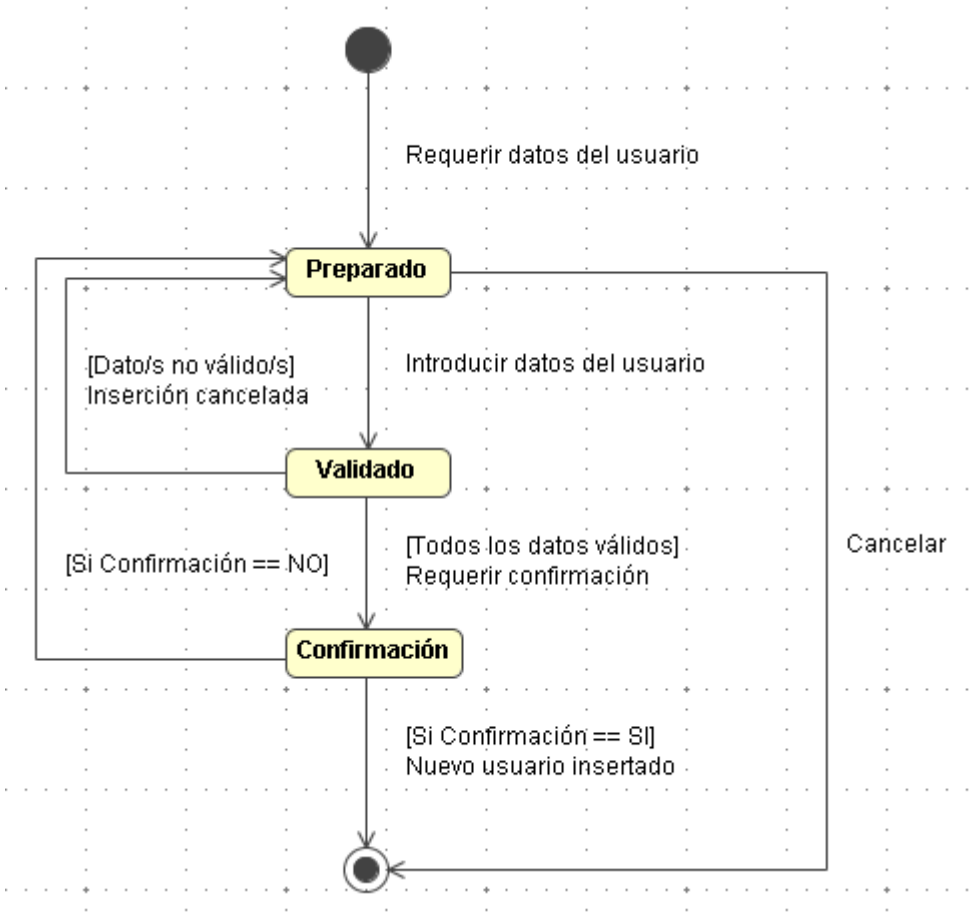

Figura 17. Diagrama de Estado: Crear usuario

6.2.9 Diagrama de Estado: Caso de Uso "Modificar usuario"

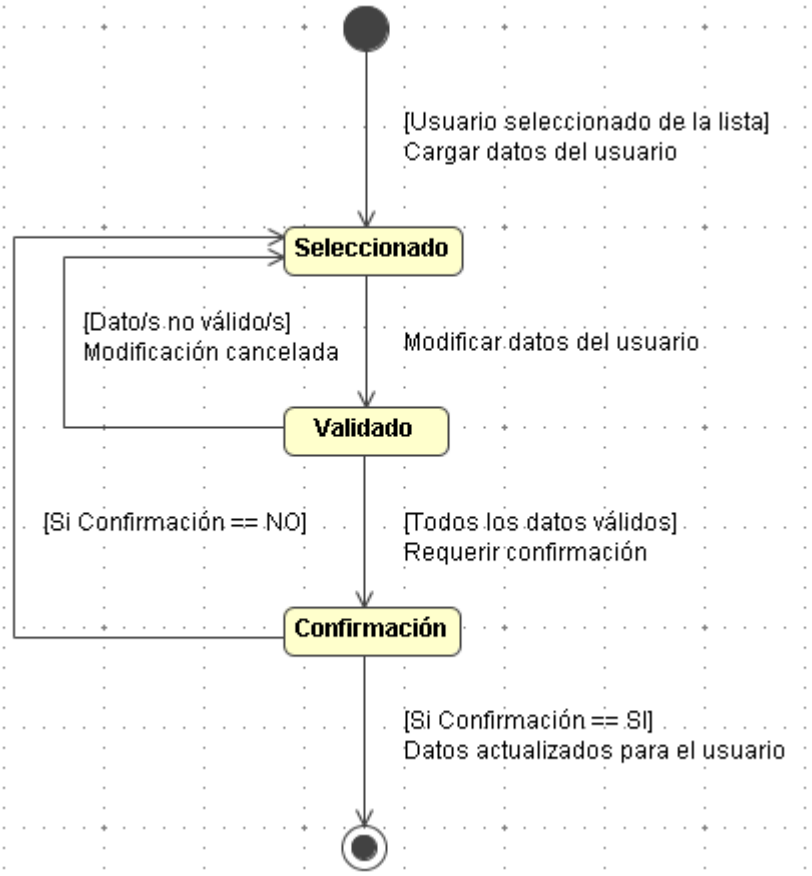

Figura 18. Diagrama de Estado: Modificar usuario

6.2.10 Diagrama de Estado: Caso de Uso "Eliminar usuario"

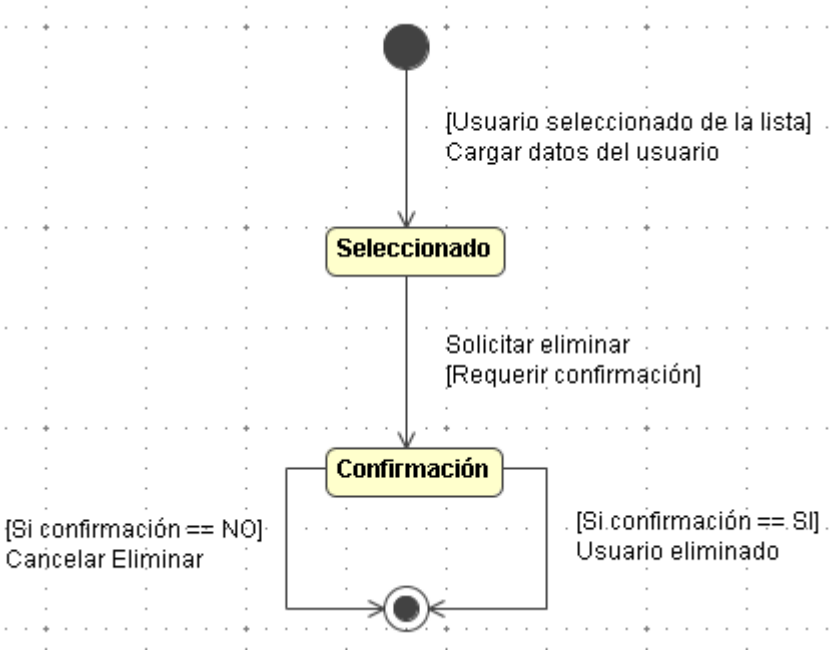

Figura 19. Diagrama de Estado: Modificar usuario

6.2.11 Diagrama de Estado: Caso de Uso "Filtrar usuarios"

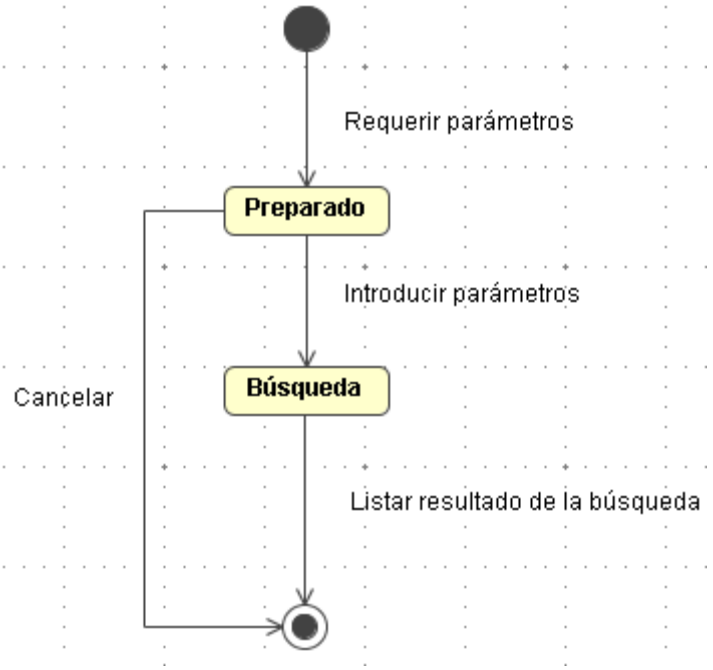

Figura 20. Diagrama de Estado: Filtrar usuarios

6.2.12 Diagrama de Estado: Caso de Uso "Crear admisión"

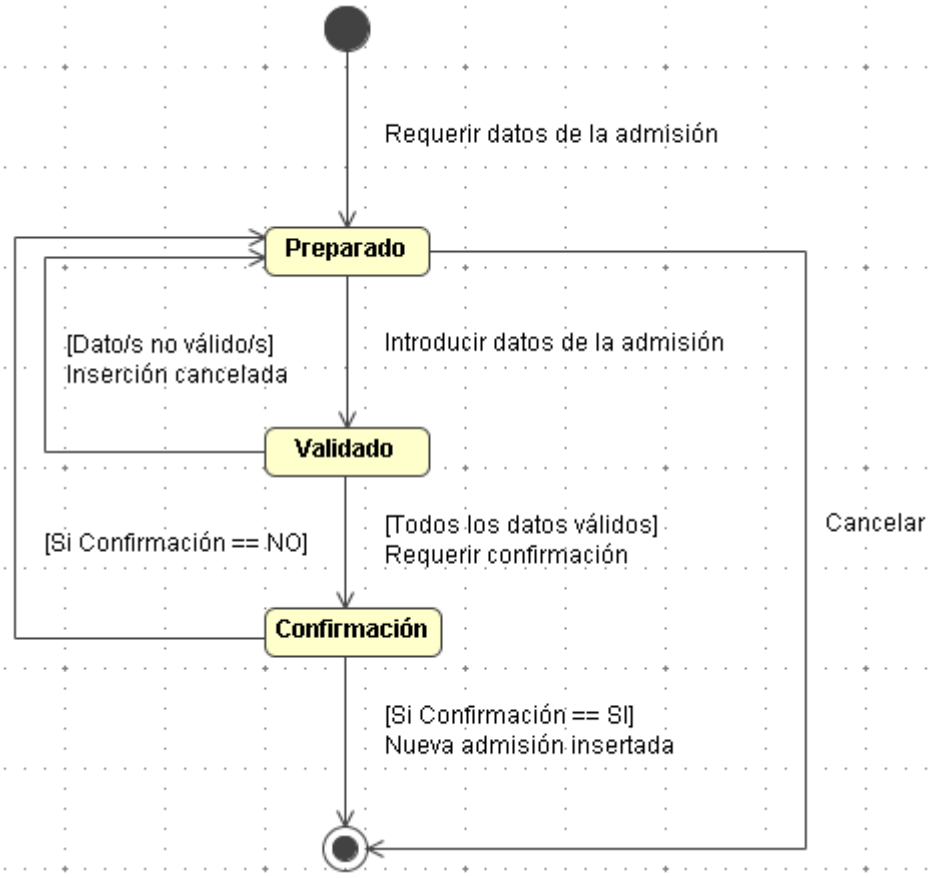

Figura 21. Diagrama de Estado: Crear admisión

6.2.13 Diagrama de Estado: Caso de Uso "Modificar admisión"

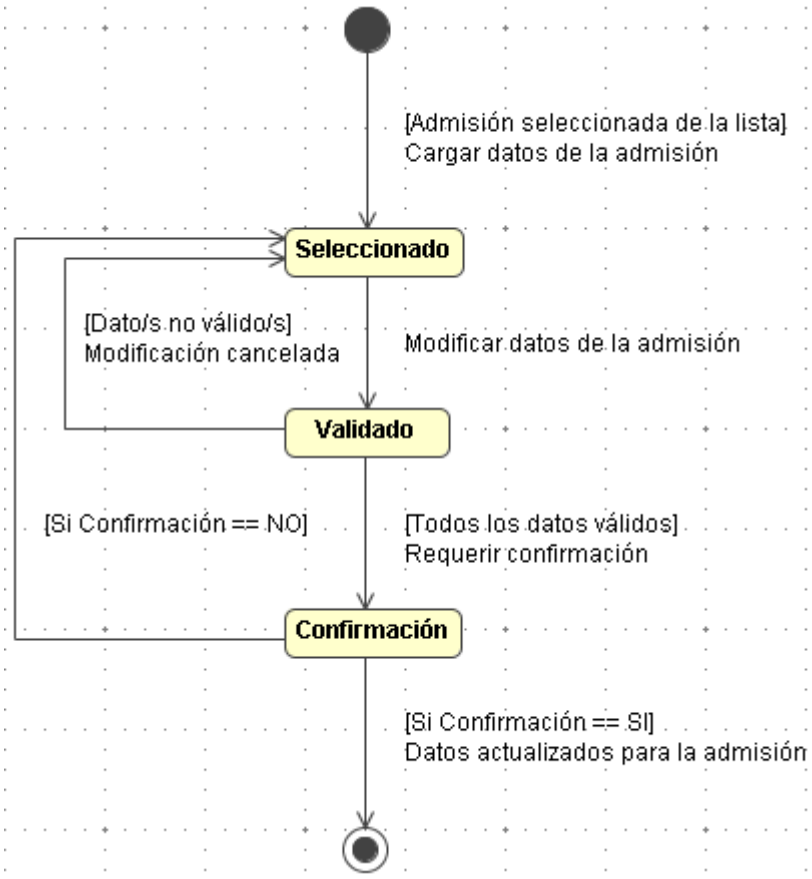

Figura 22. Diagrama de Estado: Modificar admisión

6.2.14 Diagrama de Estado: Caso de Uso "Eliminar admisión"

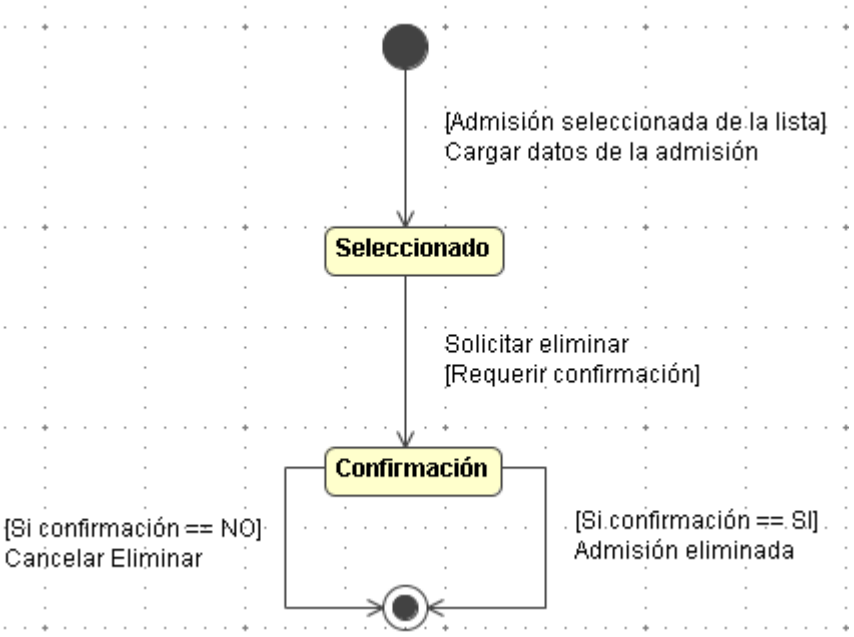

Figura 23. Diagrama de Estado: Modificar admisión

6.2.15 Diagrama de Estado: Caso de Uso "Filtrar admisiones"

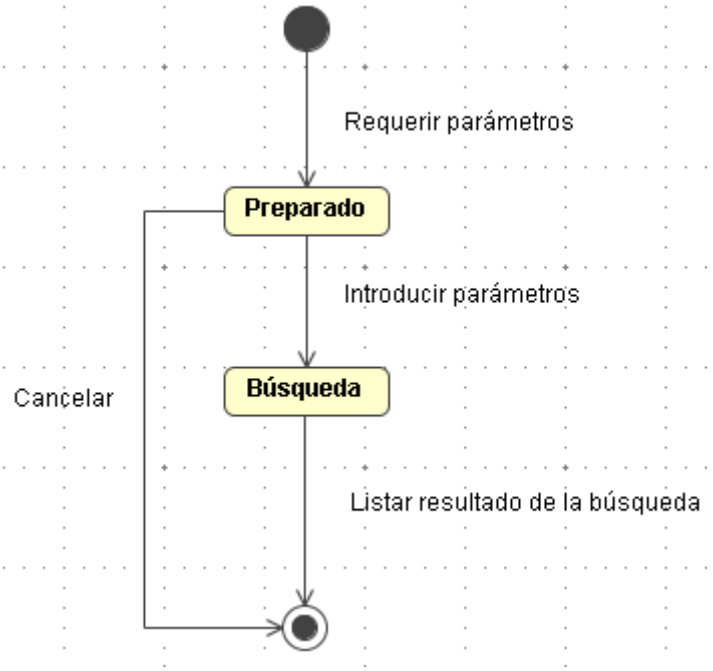

Figura 24. Diagrama de Estado: Filtrar admisión

6.3 ENTORNO DEL DISEÑO – ESQUEMA DE COMUNICACIÓN

La herramienta se ha diseñado de tal manera que todos los elementos se puedan comunicar de manera orquestada y acorde a la funcionalidad establecida para ellos.

Visualizando la siguiente figura podemos hacernos una idea clara de la relación existente entre los diversos componentes que forman parte del proyecto, de una manera o de otra, así como del rol individual que desempeñan, con respecto el conjunto del sistema.

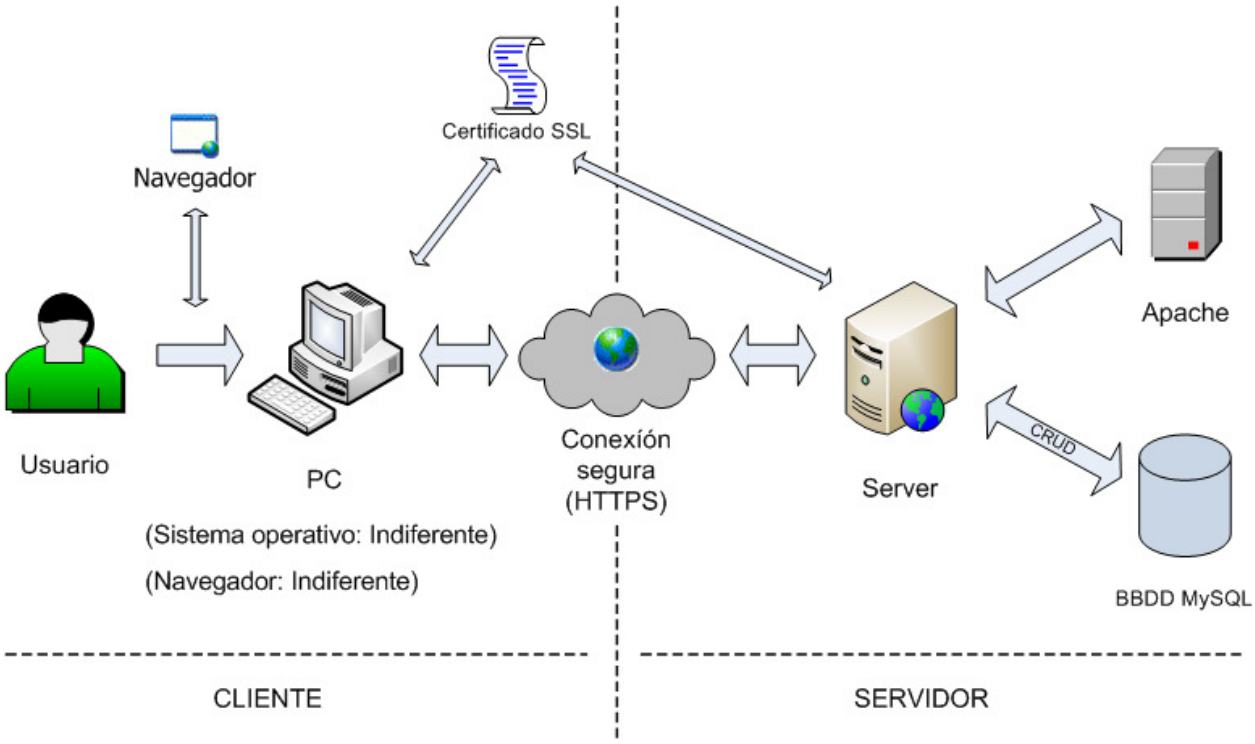

Figura 25. Diseño: Esquema de Comunicación

En la figura anterior podemos observar la agrupación de los elementos en dos conjuntos claramente diferenciados:

- Conjunto "Cliente": formado todos por los usuarios y equipos informáticos desde los cuales se accede a la aplicación, utilizando el protocolo de comunicación seguro HTTPS y un certificado SSL auto-firmado, con independencia del navegador y del sistema operativo que se utilice.

Para la implementación de la interfaz de usuario se hizo uso de la versión 4.0.7 de la librería gráfica ExtJS, desarrollada en javascript. [INET, 27]

- Conjunto "Servidor": formado por el servidor donde se aloja el servidor de aplicaciones web "Apache" y el gestor de bases de datos "MySQL", donde se pueden realizar diversas operaciones de consulta y gestión de la información (CRUD).

6.4 MODELO DE DATOS

El modelo de datos está orientado a describir la Base de Datos, el cual permite describir dos aspectos importantes: [INET, 04]

- Las estructuras de datos de la base de datos, que especifica los tipos de datos que existentes en la base de datos y la forma en la que se relacionan.
- Las restricciones de integridad, que son el conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada.

A continuación, se presenta el diagrama con las tablas de la base de datos que contienen toda la información necesaria para la carga de las aplicaciones y sus módulos. Esta información está condicionada por los derechos de visualización asignados a los perfiles disponibles para el usuario autenticado:

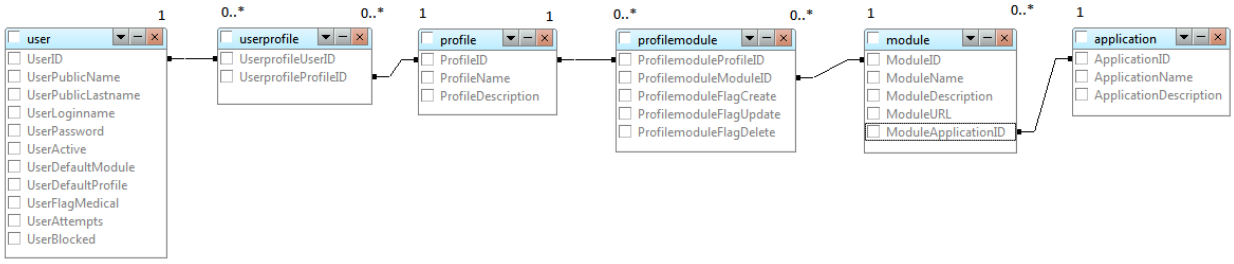

Figura 26. Modelo de Datos: Tablas Usuario-Perfil-Módulo-Aplicación

Nota: En la tabla "user" de la figura anterior, quedó pendiente la visualización de los siguientes campos: "UserAttempts" y "UserBlocked".

Por otra parte, se definieron varias tablas con el cometido de almacenar toda la información referente a los pacientes y a sus admisiones:

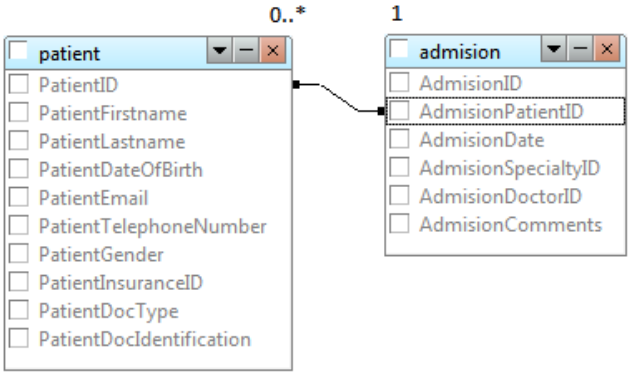

Figura 27. Modelo de Datos: Tablas Paciente-Admisión

Diseño de la base de datos completa:

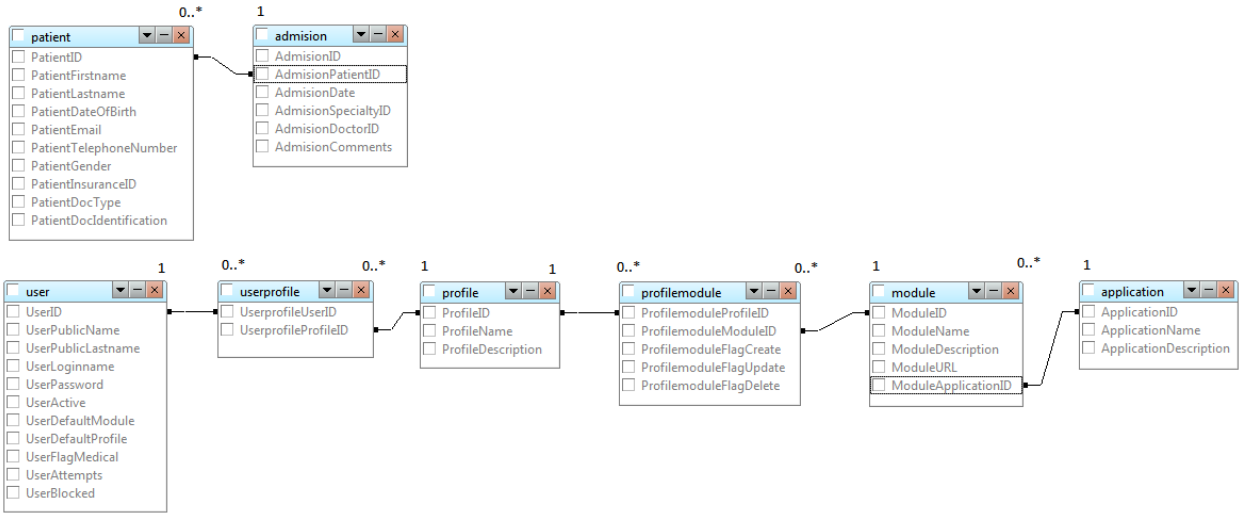

Figura 28. Modelo de Datos: Base de datos completa

Para consultar la información de la base de datos en detalle, ver el Anexo A.

7 ENTORNO DE DESARROLLO Y PRODUCCIÓN

7.1 ENTORNO DE DESARROLLO (IDE)

Para la implementación de este proyecto, se utilizó el entorno de desarrollo integrado Eclipse, el cual ofrece como ventajas:

- Licencia de software libre (EPL) [INET-28]
- Interfaz de usuario amigable
- Multilenguaje: Característica importante en la integración de código PHP y Javascript.
- Adaptabilidad de acuerdo a las necesidades de desarrollo, permitiendo la instalación de plugins para ampliar funcionalidades.

7.2 LIBRERÍA GRÁFICA DE APOYO

Para soportar la interfaz de usuario, se utilizó la librería gráfica ExtJS Versión 4.07 de Sencha, la cual dispone de: [INET, 26]

- Componentes UI del alto desarrollo y personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias Open source y comerciales.

ExtJS usa un motor de render con los siguientes beneficios:

- Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.

- Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.

- Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

7.3 SERVIDOR XAMPP

Por la integración de plataformas que ofrece y una interfaz amigable, se ha utilizado para la implementación de este proyecto el Servidor Independiente Multiplataforma XAMP.

Este servidor incluye:

- El servidor web Apache.
- Los sistemas gestores bases de datos MySQL y SQLite con sus respectivas interfaces de administración.
- Servidores de FTP como ProFTPD óFileZilla FTP Server.
- OpenSSL: Paquete de herramientas de administración y bibliotecas relacionadas con la criptografía, que suministran funciones criptográficas a otros paquetes como OpenSSH y navegadores web (para acceso seguro a sitios HTTPS).

Existen versiones para Linux, Windows, MacOS X y Solaris, cuyos paquetes difieren según la disponibilidad de los diversos programas en cada plataforma. Cabe aclarar que la versión utilizada para este proyecto Xampp para Windows.

8 CONCLUSIONES Y TRABAJO FUTURO

8.1 CONCLUSIONES

La realización del presente Proyecto Fin de Máster supuso el estudio, aprendizaje y comprensión de las más habituales y diversas amenazas existentes en el desarrollo de aplicaciones de sistemas o servicios web. Esto llevó a la necesidad de aplicar diversas técnicas y mecanismos para minimizar los riegos existentes, con la intención de evitar sus efectos o, al menos, obligar a sus ejecutores a tener que invertir más tiempo y esfuerzo para alcanzar sus objetivos.

Así pues:

- Se aplicaron algunas de los métodos y técnicas de la Ingeniería del Software para realizar el análisis, diseño, implementación y documentación apropiados de la herramienta en cuestión.

- Debido a las características del proyecto, fue necesario el estudio de varios lenguajes de programación de los que se pueden agrupar en dos entornos claramente diferenciables:

  - Entorno del Cliente: estudio, comprensión y aprendizaje de la librería gráfica ExtJS, creada en JavaScript, y que sirvió para implementar la interfaz del usuario de una manera atractiva y manejable para el desarrollo y mantenimiento de la aplicación.

  - Entorno del Servidor: estudio, comprensión y aprendizaje de los mecanismos de seguridad disponibles para el lenguaje de programación PHP.

- Se aplicó la programación por capas MVC, obteniendo como ventajas principales la facilidad y flexibilidad de la estructura del código, la separación clara de “poderes” (datos, implementación e interfaz), un mayor nivel de seguridad y una gran ayuda para el desarrollo y mantenimiento presente y futura de la aplicación.

- Se analizaron las principales características de dos de las herramientas de desarrollo (IDE) más populares en el mercado (Eclipse y NetBeans), con el fin de poder identificar cual podría adaptarse mejor al propósito del presente PFM. Finalmente, se seleccionó la primera de ellas gracias a su facilidad de uso, de instalación, configuración y por tratarse de software libre.

En definitiva, el presente proyecto ha requerido de un provechoso y educativo periodo de tiempo para la adquisición de los nuevos conocimientos, tanto técnicos como conceptuales, imprescindibles para llevar a buen término los objetivos propuestos, experiencia que ha sido muy útil para adquirir habilidades en la implementación de mecanismos y técnicas de seguridad web, así como para potenciar la capacidad de adaptación ante nuevos retos profesionales.

8.2 TRABAJO FUTURO

De entre las posibles mejoras y ajustes que se podrían realizar sobre la aplicación, me gustaría agruparlas en dos niveles:

- A nivel de la funcionalidad:

  - La implementación de funcionalidades para el manejo de históricos de operaciones, por aplicaciones, módulos, perfiles, usuarios,… para que puedan ser usados como base a futuros estudios estadísticos para la mejora de los procesos.

  - La implementación de *Flujos de trabajo* o *WorkFlows*, para afinar los procesos existentes en los módulos, teniendo en cuenta a los perfiles y/o cualquier otro aspecto o indicador que se considere necesario.

  - El análisis, diseño e implementación de una interfaz para sistemas móviles y tabletas, con un funcionamiento más ligero y/o complementario al disponible en la versión para PC.

- Posibilidad de realizar algunas operaciones "off-line", almacenando los datos en modo local, y aplicando los mecanismos de seguridad y cifrado más adecuados para salvaguardar la integridad y la privacidad de estos. La sincronización con el servidor podría darse con dos posibilidades, siempre y cuando la aplicación tenga establecida la conexión con el servidor: de manera automática, nada más reconectar la aplicación con el servidor o, de manera manual, a petición del usuario.

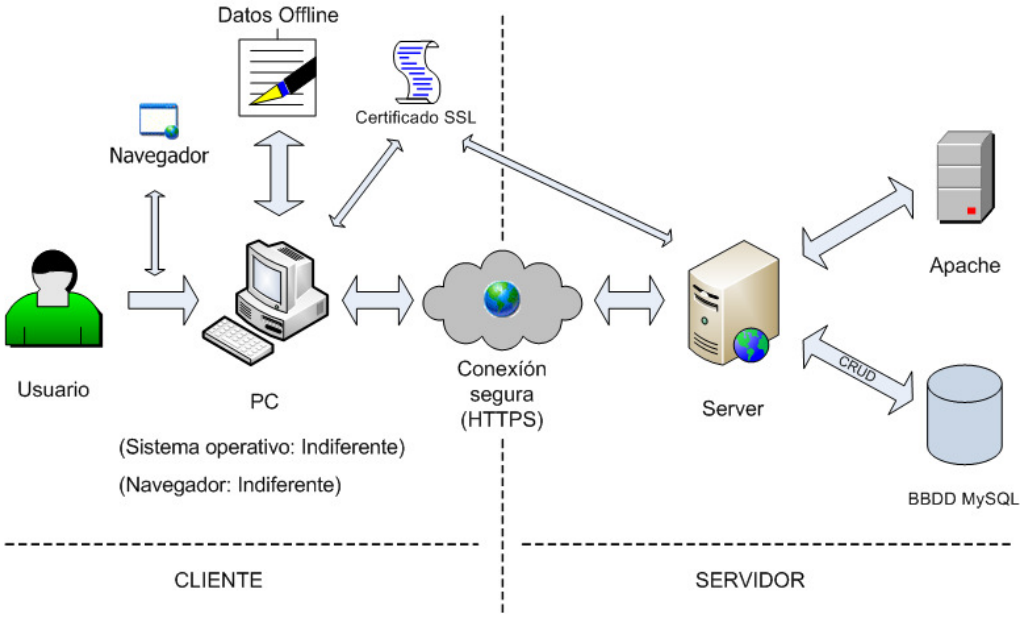

Figura 29. Trabajo futuro: Esquema de Comunicación

- La inclusión de nuevas aplicaciones, módulos, perfiles, acordes a las necesidades definidas por nuevos requisitos funcionales.

- Afinación de los derechos de acceso y de operaciones, para permitir la posibilidad de asignación de diferentes niveles de derechos de manera individual para los usuarios del sistema. Es decir, un usuario U1, con perfil P1, podría tener disponibles las operaciones de creación, modificación y eliminación de los datos de un módulo M1, mientras otro usuario U2, con idéntico perfil P1, podría no tener disponibles ninguna de las operaciones sobre los datos para el mismo módulo M1.

- La asignación univoca a los registros de documentos subidos por medio de la aplicación, así como su visualización y/o gestión.

- Internacionalización de los textos de la aplicación.

- A nivel de la seguridad web:

  - Sustituir el actual certificado digital de seguridad (SSL) auto-firmado, que no es de confianza por ser generado por el propio servidor, por otro certificado digital firmado por alguna de las Entidades de Certificación (EC) o Autoridades de Certificación (AC) externa, para obtener una sistema de clave pública/privada seguro.

  - Uso de una tarjeta inteligente para la autenticación en la aplicación y/o la realización de operaciones. Para ello, es recomendable contar con un certificado digital de seguridad válido.

  - Estudiar posibles mejoras y ajustes para el almacenamiento de los datos más sensibles e importantes almacenados en la base de datos.

  - Estudiar la posibilidad de migrar la aplicación a un servidor con sistema operativo Linux, realizando un estudio de rendimiento, seguridad, pragmatismo y costes, manteniendo como servidor de aplicaciones web al servidor HTTP Apache.

  - Estudiar las posibilidades existentes para ofuscar "dinámicamente" el código PHP generado por la aplicación, así como las alternativas para ofuscar el código JavaScript a nivel de aplicación.

  - Estudiar la conveniencia de cifrar el código fuente de la aplicación, en el servidor donde este sea alojado, así como la posibilidad de ofuscar el código de salida presentado por los navegadores.

- Configurar un Firewall o cortafuegos para bloquear el acceso de las peticiones no autorizadas, permitiendo, al mismo tiempo, las comunicaciones autorizadas, limitando, cifrando y/o descifrando el tráfico entre los diferentes ámbitos sobre la base de un conjunto de normas y otros criterios. [INET, 25]

- Configurar el gestor de base de datos MySQL desde el punto de vista de la seguridad, llevando a cabo los siguientes ajustes: [de Frutos, 2012]

  - Eliminar los usuarios anónimos.
  - Eliminar cualquier base de datos de ejemplo o test.
  - Cambiar la contraseña del usuario "*pma*", el cual es utilizado por la aplicación *phpMyAdmin* de MySQL.
  - Crear usuarios que adoptarán el principio del *mínimo privilegio*, asignando los privilegios imprescindibles para realizar sus labores.

9 BIBLIOGRAFÍA

9.1 MANUALES

[IEEE-830, 1998] "IEEE-STD-830-1998 : Especificaciones de los requisitos del software"

[Jacobson, 1992] "Object-Oriented Software Engineering: A Use Case Driven Approach"
Jacobson I, Christenson M, Johnson P, Overgaard G.
Addison-Wesley. Reading MA (1992)
(ISBN: 0201544350)

[Ferré, UML] "Desarrollo Orientado a Objetos con UML"
Xavier Ferré Grau (Facultad de Informática – UPM)
María Isabel Sánchez Segura (Escuela Politécnica Superior – UC3M)

[Larman, 1999] "UML y Patrones: Introducción al análisis y diseño orientado a objetos y al proceso unificado"
C. Larman. Prentice Hall, 1999.
(ISBN: 8420534382)

[Dix-HCI, 2003] "Human-Computer Interaction"
Alan Dix, Janet E. Finlay, Gregory D. Abowd, Russell and Russell Beale. Ed, Prentice Hall. 3rd Edition. (2003)
(ISBN: 8428325456)

[de Frutos, 2012] Módulo "Seguridad en la Programación Web" (SPW)
Juan Alberto de Frutos Velasco.
Máster en Ingeniería Web (2011- 2012)
Universidad Politécnica de Madrid

[Garfinkel, 2002] "Web Security, Privacy & Commerce"
Simson Garfinkel.
Ed. O'Reilly, 2nd Edition, January 2002
(ISBN: 0-596-00045-6)

[ISO27001, 2009] "The Basics of information Security – A practical handbook"
March 2009
(ISBN: 978-90-813341-1-2)

[Snyder, 2010] "Pro PHP Security"
Chris Snyder, Michael Southwell.
2nd Edition, Apress, 2010
(ISBN: 1-59059-508-4)

[Shiflett, 2005] "Essential PHP Security"
Chris Shiflett
O'Really, 2005
(ISBN: 0-596-00656-X)

[Alshanetsky, 2010] "PHP | Architect's Guide to Security Security"
Ilia Alshanetsky, 2005
(ISBN: 0-9738621-0-6)

9.2 ENLACES WEB

[INET, 01] *Cross Site Scripting*.
http://es.wikipedia.org/wiki/Cross-site_scripting

[INET, 02] *Glosario de Términos (varios)*
http://e-amanecer.com/licenciatura/glosario.html

[INET, 03] *Diagramas de Estado*
http://es.wikipedia.org/wiki/Diagrama_de_estados

[INET, 04] *Modelo de Datos*
http://es.wikipedia.org/wiki/Modelo_de_datos

[INET, 05] *Flujos de Trabajo*
http://es.wikipedia.org/wiki/Flujo_de_trabajo

[INET, 06] *Secure Sockets Layer*
http://es.wikipedia.org/wiki/Transport_Layer_Security

[INET, 07] *XAMPP*
http://es.wikipedia.org/wiki/XAMPP

[INET, 08] *.htaccess*
http://es.wikipedia.org/wiki/Htaccess

[INET, 09] *Apache - allow_url_open*
http://php.net/manual/es/filesystem.configuration.php

[INET, 10] *Apache - display_errors*
Apache - display_startup_errors
Apache - log_errors
Apache - error_log
http://php.net/manual/es/errorfunc.configuration.php

[INET, 11] *PHP - constantes predefinidas*
http://www.php.net/manual/es/errorfunc.constants.php

[INET, 12] *Apache - magic_quotes_gpc*
http://php.net/manual/es/info.configuration.php

[INET, 13] *Apache* - magic_quotes_sybase
http://php.net/manual/es/sybase.configuration.php

[INET, 14] *Apache* - register_globals
http://php.net/manual/es/ini.core.php

[INET, 15] *Apache - session.referer_check*
Apache - session.cookie_httponly
Apache - session.use_only_cookies
Apache - session.use_trans_sid
Apache - session.cookie_secure
Apache - session.cookie_lifetime
http://es.php.net/manual/es/session.configuration.php#ini.session.referer-check

[INET, 16] PHP - session_name()
http://php.net/manual/es/function.session-name.php

[INET, 17] XAMPP, SSL – certificado autofirmado
http://tecnoloxiaxa.blogspot.com.es/2009/05/xampp-ssl-cifrar-la-transmision-de-las.html

[INET, 18] *Ataque de fuerza bruta*
https://www.owasp.org/index.php/Brute_force_attack

[INET, 19] *Utilización de Exploits*
http://www.hackxcrack.es/forum/index.php?topic=1378.0

[INET, 20] *Cross-Site Scripting*
http://www.instisec.com/publico/xss.asp

[INET, 21] *¿Qué es y cómo opera un ataque de Cross-Site Scripting (XSS)?*
http://www.seguridad.unam.mx/documento/?id=35

[INET, 22] *Fijación de sesión*
http://cert.inteco.es/cert/Notas_Actualidad/gestion_sesiones_web_iv_ataques_fijacion_sesion_eavesdropping_20111216?postAction=getLatestInfo

[INET, 23] *Seguridad Web. Formularios Web y URL's (Parte 1)*
http://www.capitanseo.es/2011/04/seguridad-web-formularios-web-y-urls-parte-1/

[INET, 24] *SQL Injection*
https://www.owasp.org/index.php/SQL_Injection

[INET, 25] *Firewall*
http://es.wikipedia.org/wiki/Cortafuegos_(inform%C3%A1tica)

[INET, 26] *ExtJS, lo bueno, lo feo y lo malo*
http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/

[INET, 27] *Sencha - ExtJS*
http://www.sencha.com/products/extjs/

[INET, 28] *Eclipse Public License - EPL*
http://gestiweb.com/?q=content/346-eclipse-public-licence-epl/

ANEXO A. BASE DE DATOS

admision

Field Name	Field Type	Default	AllowNull	PriKey	Extra
AdmisionID	int(11) FIRST		NO	YES	
AdmisionPatientID	int(11) COMMENT 'Foreign key fro	-1	NO	NO	
AdmisionDate	date AFTER `AdmisionPatientID`		NO	NO	
AdmisionSpecialtyID	int(11) AFTER `AdmisionDate`	-1	NO	NO	
AdmisionDoctorID	int(11) AFTER `AdmisionSpecialtyI	-1	NO	NO	
AdmisionComments	varchar(1000) AFTER `AdmisionD		YES	NO	

application

Field Name	Field Type	Default	AllowNull	PriKey	Extra
ApplicationID	int(11) FIRST		NO	YES	
ApplicationName	varchar(100) AFTER `ApplicationI		NO	NO	
ApplicationDescription	varchar(255) AFTER `ApplicationN		YES	NO	

module

Field Name	Field Type	Default	AllowNull	PriKey	Extra
ModuleID	int(11) FIRST	-1	NO	YES	
ModuleName	varchar(100) AFTER `ModuleID`		NO	NO	
ModuleDescription	varchar(255) AFTER `ModuleNam		YES	NO	
ModuleURL	varchar(255) AFTER `ModuleDesc		NO	NO	
ModuleApplicationID	int(11) COMMENT 'Foreign key fro	-1	YES	NO	

patient

Field Name	Field Type	Default	AllowNull	PriKey	Extra
PatientID	int(11) FIRST	-1	NO	YES	
PatientFirstname	varchar(255) AFTER `PatientID`		NO	NO	
PatientLastname	varchar(255) AFTER `PatientFirstr		NO	NO	
PatientDateOfBirth	date AFTER `PatientLastname`		NO	NO	
PatientEmail	varchar(100) AFTER `PatientDate		NO	NO	
PatientTelephoneNumber	varchar(9) AFTER `PatientEmail`		NO	NO	
PatientGender	char(255) AFTER `PatientTelepho		NO	NO	
PatientInsuranceID	int(11) AFTER `PatientGender`	-1	YES	NO	
PatientDocType	int(11) AFTER `PatientInsuranceID		NO	NO	
PatientDocIdentification	varchar(255) AFTER `PatientDocT		NO	NO	

profile

Field Name	Field Type	Default	AllowNull	PriKey	Extra
ProfileID	int(11) FIRST	-1	NO	YES	
ProfileName	varchar(100) AFTER `ProfileID`		NO	NO	
ProfileDescription	varchar(255) AFTER `ProfileName		YES	NO	

profilemodule

Field Name	Field Type	Default	AllowNull	PriKey	Extra
ProfilemoduleProfileID	int(11) COMMENT 'Foreign key fro	-1	NO	YES	
ProfilemoduleModuleID	int(11) COMMENT 'Foreign key fro	-1	NO	YES	
ProfilemoduleFlagCreate	int(1) AFTER `ProfilemoduleModul	0	YES	NO	
ProfilemoduleFlagUpdate	int(1) AFTER `ProfilemoduleFlagCr	0	YES	NO	
ProfilemoduleFlagDelete	int(1) AFTER `ProfilemoduleFlagUp	0	YES	NO	

user

Field Name	Field Type	Default	AllowNull	PriKey	Extra
UserID	int(11) FIRST	-1	NO	YES	
UserPublicName	varchar(100) AFTER `UserID`		YES	NO	
UserPublicLastname	varchar(100) AFTER `UserPublicN		YES	NO	
UserLoginname	varchar(100) AFTER `UserPublicLa		NO	NO	
UserPassword	varchar(256) AFTER `UserLoginna		NO	NO	
UserActive	int(1) AFTER `UserPassword`	0	NO	NO	
UserDefaultModule	int(11) COMMENT 'Foreign key fro	-1	YES	NO	
UserDefaultProfile	int(11) COMMENT 'Foreign key fro	-1	YES	NO	
UserFlagMedical	int(1) AFTER `UserDefaultProfile`	0	YES	NO	
UserAttempts	int(1) AFTER `UserFlagMedical`	0	NO	NO	
UserBlocked	int(1) AFTER `UserAttempts`	0	NO	NO	

userprofile

Field Name	Field Type	Default	AllowNull	PriKey	Extra
UserprofileUserID	int(11) COMMENT 'Foreign key fro	-1	NO	YES	
UserprofileProfileID	int(11) COMMENT 'Foreign key fro	-1	NO	YES	

Figura 30. Base de datos - Descripción

ANEXO B. INTERFAZ DE USUARIO

1 Página de inicio: Index.php

En esta página se solicitarán un identificador de usuario y contraseñas válidas para acceder al sistema.

Figura 31. Autenticación - Index.php

A continuación se muestran los posibles resultados al intento de autenticación al sistema:

- En el caso de autenticación correcta, la aplicación cargará el perfil y el módulo establecidos por defecto para el usuario.

- En el caso en el que se introduzcan un identificador de usuario válido pero una contraseña incorrectas, el contador de intentos se incrementará en una unidad y se mostrará el siguiente mensaje:

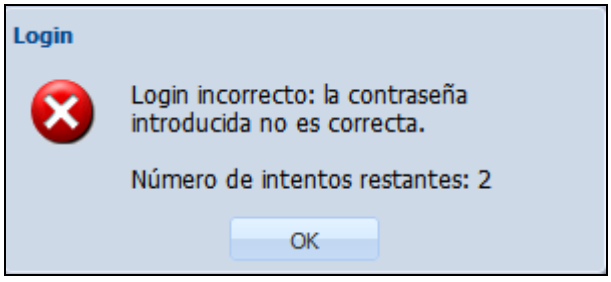

Figura 32. Login – Contraseña incorrecta

Nota: El número máximo de intentos es 3.

- En el caso en el que se introduzca un identificador de usuario válido, independientemente de si se introdujo una contraseña válida o no, y se haya superando el número máximo de intentos, se mostrará el siguiente mensaje:

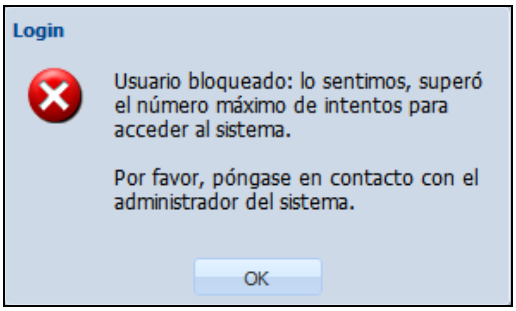

Figura 33. Login – Usuario bloqueado

- En el caso en el que se introduzcan un identificador de usuario y contraseñas válidos, pero la cuenta no se encuentre activada por el administrador, se mostrará el siguiente mensaje:

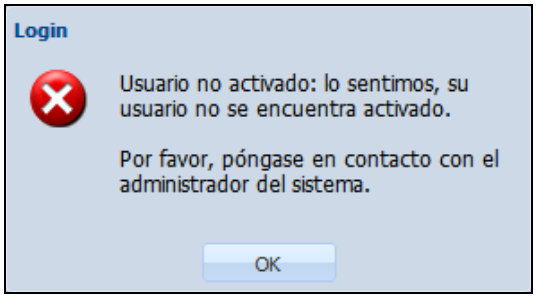

Figura 34. Login – Usuario no activado

- Por último, se contempló la posibilidad de una autenticación correcta, con la cuenta activa adecuadamente, pero sin autorizaciones para ninguno de los módulos de la aplicación. En este caso, se mostrará el siguiente mensaje:

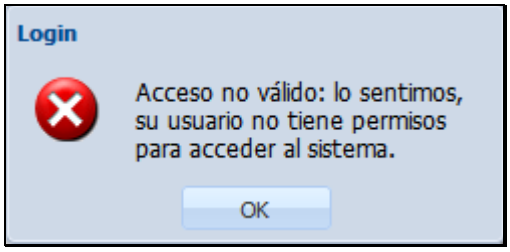

Figura 35. Login – Acceso no válido

2 Acceso a la aplicación

Una vez que el usuario se ha autenticado correctamente, el sistema mostrará el módulo y perfil definido por defecto para el usuario.

La siguiente figura muestra la interfaz de usuario del módulo "*Usuarios*", cuya disposición es, actualmente, idéntica para el resto de los módulos, salvo que se quiera diseñar otros módulos con una interfaz diferente y especifica.

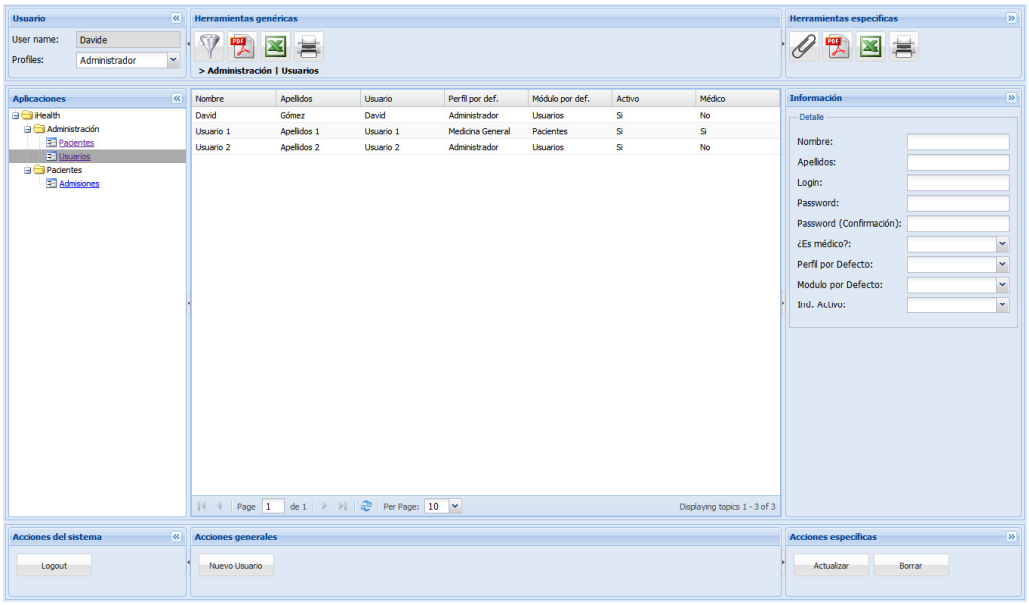

Figura 36. Aplicación "Administración", Módulo "Usuarios"

La disposición de los diversos componentes existentes en los módulos se encuentra agrupada por "paneles" y es la siguiente:

	Columna A	Columna B	Columna C
Fila 1	Panel A1	Panel B1	Panel C1
Fila 2	Panel A2	Panel B2	Panel C2
Fila 3	Panel A3	Panel B3	Panel C3

Figura 37. Aplicación – Estructura básica

- COLUMNAS:

  - *Columna A*: Elementos relacionados con la navegación de las páginas y autorizaciones disponibles para el usuario logeado (módulos y perfiles)
  - *Columna B*: Registros especificos al módulo cargado, así como acciones sobre el conjunto de estos.
  - *Columna C*: Detalle de los registros seleccionados previamente de la columna B, así como acciones disponibles sobre estos registros especificos.

- PANELES:

  - *Panel A1*: Datos del usuario logeado y perfiles disponibles.
  - *Panel A2*: Arbol de navegación disponible, dependiente del perfil y de los permisos disponibles.
  - *Panel A3*: Accion de logout de la aplicación.

  - Panel B1: Acciones adicionales sobre el conjuto de los datos cargados: filtro, exportar datos a Excel o a PDF, imprimir datos.
  - Panel B2: Listado de registros mostrados en el módulo.

- Panel B3: Acciones de nivel genérico que impactan sobre la base de datos: añadir un nuevo registro al listado.

- *Panel C1*: Acciones adicionales sobre un elemento en particular, previamente seleccionado del panel B2.
- *Panel C2*: Detalle de los datos del elemento seleccionado previamente del panel B2.
- *Panel C3*: Acciones a nivel especifico del registro seleccionado del panel B2 que impactan sobre la base de datos: actualizar datos, eliminar registro.