Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN SISTEMAS DE INFORMACIÓN

Trabajo Fin de Grado

Entrenador Web de vulnerabilidades SQL

Autor: Rocío Recuero Santaella

Tutor/es: D. Manuel Sánchez Rubio

2015

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica

GRADO EN SISTEMAS DE INFORMACIÓN

Trabajo Fin de Grado

Entrenador Web de vulnerabilidades SQL

Autor: Rocío Recuero Santaella

Director: D. Manuel Sánchez Rubio

TRIBUNAL:

Presidente: ……

Vocal 1º: ……

Vocal 2º: D……

CALIFICACIÓN: ……..

FECHA: ……..

Entrenador Web de Vulnerabilidades SQL

Trabajo Fin de Grado

Rocío Recuero Santaella
rrecuerosantaella@gmail.com

Tutor: Manuel Sánchez Rubio

Resumen

El principal objetivo de este proyecto es el desarrollo de una aplicación que permita evidenciar la existencia de vulnerabilidades de seguridad intrínsecas a los sistemas Web, centrado específicamente en aquellas de tipo *Inyección SQL*.

La aplicación desarrollada contiene vulnerabilidades diseñadas específicamente para un propósito docente, es decir, los usuarios dispondrán de distintas funcionalidades donde poder verificar diferentes niveles y tipos de vulnerabilidad.

Para responder a las diferentes necesidades del aprendizaje, en la aplicación se incluye un "modo seguro" donde los usuarios se enfrenten al desafío de una aplicación diseñada teniendo en cuenta ciertos mínimos aspectos de *ciberseguridad*.

Abstract

This Project main purpose is develop an application able to highlight all those security vulnerabilities intrinsic to Web Systems, focused mainly on those concerning to SQL Injection.

Some designing specific vulnerabilities are part of the developed application responding a docent objective, this means, users will be allowed to check different weakness levels and types in each kind of functionalities.

In order to respond to several training needs, application has included a "safety mode" which confronts users to the challenge of an application that has been designed taking certain minimal aspects of *cyber security* in account.

Palabras clave

Web, SQL Injection, Vulnerabilidades, Ciberseguridad.

2015

Agradecimientos

Quiero agradecer ***a Jose su paciencia*** *durante estos 4 años, esos fines de semana sólo en el salón mientras yo estaba en la "bat-cueva" con "las cosas de la uni". Gracias por darme tu apoyo para tomar la decisión de embarcarme en "esto", y por tu ayuda en los malos momentos, pero sobre todo que compartas conmigo los buenos, aunque no te hagan ni pizca de ilusión mis notas sé que en el fondo te alegras… ;) ¡Tailoviuu!*

¡Y ***cómo no voy a estar agradecida a mis padres****!, ellos me han apoyado siempre, incluso cuando les pedí consejo para esta gran locura. Les agradezco la forma en que me han educado y haberme ayudado a construir esta forma de ser que tengo y de ver y actuar en la vida. A mi papi, porque sí, porque para eso soy su niña y a mi madre porque este mío es un sueño suyo hecho realidad, y porque gracias a su apoyo y gracias a ella en sí misma estoy en este punto de mi vida y en esto estoy llegando a LA META.*

Unas palabritas para mis chicos de la uah, sois lo mejor que me llevo, ¡lo que os voy a echar de menos!:

Mi ***Robot****, Irene jodía, ¡qué buena eres! Y lo digo en todos los sentidos, si eres grande en lo profesional, eres mejor en todo lo demás. Yo de pequeña quiero ser como tu, ¡qué gran descubrimiento! De verdad que te mereces todo lo bueno que te pase y ha sido un placer currar contigo (#myotherhalf)*

A ***Mary*** *por todas esas tardes que hemos pasado estudiando, …por el CRC y la diagonalización (lo tenía que decir), por todo lo que hemos currado, lo que nos hemos reído y hemos criticado, por habernos enseñado a todos una gran lección, sacarte la carrera en 4 y currando sí que tiene mérito.*

Y ***Xabi****,* ***mi medio limón****, mi compañero sobretodo en esta última parte del camino, gracias por estar al otro lado y seguir dándome fuerzas para avanzar, por ser mi bastón en momentos de flaqueza y tirar juntos del carro. Porque todos sabemos que soy insufrible y aun así te embarcaste conmigo en esto, gracias por eso, pero oye, ni tan mal, ¿no?.*

No quiero dejar fuera a ***mi tutor, Manuel****, por haberme permitido aprender con él, un poquito de redes, algo de seguridad, pero lo más importante que hemos aprendido de él no estaba en los apuntes. "… **vivir con miedo es vivir a medias**…".*

- "The ones who are crazy enough to think they can change the world, are the ones who do." -

Steve Jobs (1955 – 2011)

Índice de Contenidos

Índice de Figuras, Tablas y Diagramas

- **Figuras**

- **Tablas**

- **Diagramas**

Capítulo 1.

Resumen extendido

En el presente Trabajo Fin de Grado (TFG), se recogen todos los aspectos necesarios para el desarrollo de una aplicación Web cuyas funcionalidades sean la gestión básica de usuarios y noticias, permitiendo además monitorizar ataques de seguridad, con diferentes niveles de vulnerabilidad, para medir posteriormente la eficacia de los ataques, según su naturaleza, y origen.

La aplicación a diseñar, debe ser una aplicación web, es decir, un tipo especial de aplicación cliente/servidor, donde tanto el cliente (el navegador o explorador) como el servidor (el servidor web) y el protocolo mediante el que se comunican (*HTTP*) están estandarizados y no han de ser creados por el programador de aplicaciones.

Desde el punto de vista del usuario, este tipo de aplicaciones consisten en un enorme conjunto de documentos llamados páginas, cada una de las cuales pueden contener vínculos o enlaces con otras páginas relacionadas a modo de gran repositorio de información. En un servicio Web los clientes demandan hipertextos a los servidores por medio de direcciones bajo el esquema URL (*Universal Resource Locator*) el cual permite localizar recursos en la red, incluyendo en la red de Internet.

Más concretamente, la aplicación debe ser una Web intuitiva y fácil de usar para cualquier usuario, pero cuyo objetivo principal es garantizar la existencia de vulnerabilidades fácilmente detectables de tipo *SQL Injection,* de tal manera que permitiese a los usuarios entrenar este tipo de ataques en un entorno seguro y controlado.

La aplicación se desarrolla en lenguaje Java, puesto que éste permite el desarrollo de aplicaciones en red, distribuidas y concurrentes independientes de la plataforma, ya que el código fuente del programa se compila una única vez generando el *bytecode* asociado y éste es interpretado en tiempo de ejecución, necesitando únicamente para su correcto funcionamiento una máquina virtual Java.

Por esta característica de ser independiente de la plataforma, puede ser instalada y ejecutada en cualquier servidor de aplicaciones que sea y soporte un contenedor de Servlets y JSP, como son *Apache-Tomcat, WebSphere, Weblogic, GlassFish, etc.*, aunque en la presente memoria se describe la instalación utilizando Apache-Tomcat.

Una vez instalada en el servidor, una aplicación Web será accesible a través de cualquier navegador de la red. Se ha comprobado su correcto funcionamiento para los navegadores de internet más utilizados actualmente: *Mozilla, Internet Explorer, Google Chrome y Safari*.

La aplicación se compone de dos partes principales: la base de datos y la aplicación propiamente dicha.

La Base de Datos se ha implementado con el gestor de base de datos relacionales *MySQL*. Se ha optado por esta tecnología, ya que es una herramienta potente con licenciamiento *GNU*, ampliamente utilizada en entornos profesionales y de la administración pública, y además, porque posee la credencial de tener millones de descargas en internet.

Lo realmente destacable de la aplicación es que ha sido desarrollada, utilizando el patrón MVC. Mediante el cual se consiguen aislar procesos de obtención de los datos a mostrar al usuario, de la forma en la que estos son presentados. El hecho de separar los distintos códigos (*Java Servlets, JSP, JavaScript, CSS*…) permite un desarrollo y diseño más cómodo y organizado,

además de posibilitar un mantenimiento y desarrollo posterior de la aplicación mucho más efectivo y sencillo.

El uso del framework *Struts* provee la infraestructura básica para la implementación del patrón MVC, proporciona la integración con el modelo, la lógica de negocio se implementa basándose en clases predefinidas por el framework y la construcción de la interfaz está soportada por la utilización de un conjunto de *tags* predefinidos, buscando evitar el uso de *scriptlets* (código Java incluido en el JSP dentro de etiquetas <% %>).

Comúnmente se dice que Java es un lenguaje de programación seguro, fue diseñado para solucionar ciertas vulnerabilidades relacionadas con el acceso a memoria presentes en otros lenguajes como *C++*.

Hay ciertas características específicas del lenguaje que contribuyen de forma extraordinaria a este objetivo. No obstante, cualquier sistema de información conlleva exposición a un riesgo.

Ciertamente, cualquier elemento que muestre información puede conllevar un riesgo. Antes de la aparición de las TIC toda la información se guardaba en papel en almacenes y grandes sistemas de archivadores. Los sistemas informáticos permiten la digitalización de todo este volumen de información, ganando además de espacio físico, velocidad en el acceso a la información. Pero con ello aparecen toda una serie de implicaciones derivadas del uso de estas facilidades, ya que el sistema de información debe seguir garantizando la seguridad de la información.

Se entiende por seguridad de la información al conjunto de métodos o medidas técnicas, organizativas y legales cuyo objetivo es garantizar la confidencialidad, la integridad y la disponibilidad del sistema de información tratando otras propiedades como la responsabilidad, la autenticidad y la fiabilidad.

Toda organización debe tener en cuenta un Sistema de Gestión de la Seguridad de la información, el cual se encarga de proteger y cuidar la información frente a las amenazas a las que cualquier sistema está expuesto.

Se puede definir como amenaza a todo elemento o acción capaz de atentar contra la seguridad de la información, provocando un daño (material o inmaterial).

Las amenazas surgen a partir de la existencia de vulnerabilidades, es decir que una amenaza sólo puede existir si existe una vulnerabilidad que pueda ser aprovechada, e independientemente de que se comprometa o no la seguridad de un sistema de información.

La presencia de una amenaza es una advertencia de que puede ser inminente el daño a algún activo de la información, o bien es un indicador de que el daño se está produciendo o ya se ha producido.

La mayor parte de los ataques a los sistemas informáticos son provocados, intencionadamente o no, por las personas que en general persiguen conseguir un nivel de privilegio en el sistema que les permita realizar acciones no autorizadas sobre el mismo.

En un sistema informático hay que proteger todos los recursos que forman parte del sistema, esto es:

- Hardware, donde se incluyen todos los elementos físicos del sistema informático.
- Software, donde se encuentran todos los programas que se ejecutan sobre el hardware.
- Datos, aquellos que comprenden la información que procesa el software haciendo uso del hardware.
- Otros (consumibles, personas, infraestructuras, etc.).

De entre todos los activos, el más crítico son los datos, ya que la dificultad o imposibilidad de reponerlos conllevaría una pérdida de tiempo y dinero.

Sin llegar a entrar en detalle sobre todos los tipos de personas que pueden constituir una amenaza para el sistema, se pueden mencionar los curiosos, intrusos remunerados, crackers, terroristas, ex-empleados e incluso el personal de la propia organización.

De forma general las amenazas pueden clasificarse en 2 grandes grupos: amenazas físicas y amenazas lógicas, pudiendo ser materializadas por personas, programas específicos o catástrofes naturales (inundación, incendio, fallo eléctrico, explosión, etc.).

El objetivo de esta aplicación web es la creación de un entorno seguro y apropiado para la realización de pruebas para la verificación de la existencia de vulnerabilidades, en concreto el estudio se ha centrado en aquellas de *SQL Injection*, cuyo origen radica en el incorrecto chequeo y/o filtrado de las variables utilizadas en un programa que contiene, o genera, código SQL.

Las *inyecciones SQL* se han convertido en un problema muy común en sitios web que cuentan con base de datos. La vulnerabilidad es fácilmente detectada y fácilmente explotada, y como tal, cualquier sitio es propenso a ser objeto de un intento de ataque de este tipo.

Este tipo de intrusión normalmente es de carácter malicioso, dañino o espía y por tanto son un problema de seguridad informática, y para poder prevenirlo y evitarlo debe ser tomado en cuenta por el programador de la aplicación.

La intrusión ocurre durante la ejecución del programa vulnerable, exponiendo cualquier dato de la base de datos para ser leído o modificado por un usuario malintencionado. Los ataques por *SQL Injection* permiten a los atacantes suplantar identidad, alterar datos existentes, causar problemas de repudio como anular transacciones o cambiar balances, permiten la revelación de todos los datos del sistema, destruirlos o si no volverlos inasequibles, incluso convertirse en administradores del servidor de base de datos.

Las vulnerabilidades son difíciles de gestionar. Se descubren decenas día a día, y clasificarlas es una tarea compleja.

La mejor forma de prevenir es sin duda el conocimiento, comprender cómo y porque es posible este tipo de debilidad en nuestras aplicaciones. Todo dato enviado a nuestra aplicación por un usuario, ya sea este humano o electrónico, es susceptible de contener código SQL que podría modificar el comportamiento esperado de nuestra aplicación. Por lo tanto, cualquier información que nuestra aplicación esté esperando desde fuera, debe tomarse como potencialmente peligrosa.

Capítulo 2.

Memoria

2.1. Introducción

En un mundo como el actual hasta la más simple de las operaciones electrónicas, tanto empresariales como personales, está unida a una red. Este hecho provoca que el tema de la seguridad informática o *ciberseguridad* cobre una especial relevancia en el marco de los sistemas de información.

Los sistemas informáticos actuales, permiten la digitalización de la información, facilitando su análisis y procesamiento. Se reduce el espacio de almacenamiento y se mejora el acceso a la información siendo más eficiente, pero todas estas mejoras también plantean el problema de la protección de estos datos.

Este habitual trabajo en red genera problemas derivados de la seguridad de la información, ya que al igual que los sistemas de información van evolucionando, también lo hacen las exigencias en cuanto a la seguridad de los datos y las amenazas a las que se enfrentan los involucrados. Esto hace que, al aumentar la complejidad de los sistemas, lo haga también la manera de protegerlos y atacarlos.

Las aplicaciones en general presentan habitualmente defectos, fallos de programación, algunos de los cuales afectan a su seguridad. En el caso de las aplicaciones web en particular estos se ven incrementados a medida que aumentan las líneas de código y el número de elementos diferentes involucrados dado que es más fácil que existan errores y más difícil detectarlos y ponerles solución, ya que un fallo en cualquiera de los componentes puede afectar al resto y dejar todo el sistema en manos de un atacante.

Para profundizar en este concepto, podemos utilizar un modelo conceptual que descomponga una aplicación en varios niveles:

- Interfaz de usuario. Es la parte con la que interactúan las personas para obtener y proveer la información. En el ámbito de las aplicaciones web las tecnologías serían *HTML, JavaScript, VBScript, Silverlight, Flash*, etc.
- Procesado de datos. En esta capa se realizan las tareas que procesan de alguna forma los datos, transformando, agrupando, resumiendo la información. Aquí se enmarcan los servidores web equipados con programas generadores de contenido dinámico tales como *ASP, PHP, Java Servlets, Java Server Pages*, etc.
- Almacenamiento. Generalmente la información suele ser almacenada en un repositorio que permita su uso posterior, ya sea en un sistema de ficheros, una base de datos, cintas magnéticas (de respaldo), etc.

En este sentido, en el caso de las vulnerabilidades SQL, normalmente asociadas a las aplicaciones Web, los atacantes generalmente aprovechan errores en la programación en los distintos componentes para acceder al recurso más importante de cualquier sistema: la información, ya que a través de las deficiencias producidas a nivel de *Interfaz* se puede atravesar la capa de *Proceso* e interactuar directamente con la de *Gestión y Almacenamiento de Datos.*

De esta situación descrita parte la idea de la construcción de un sistema que permita disponer de una herramienta docente, con la que los alumnos puedan realizar ataques para verificar la existencia de vulnerabilidades SQL a diferentes niveles.

2.1.1. Objetivos

El objetivo principal del proyecto consiste en el desarrollo de una aplicación Web cuyas funcionalidades sean la gestión básica de usuarios y noticias, permitiendo además monitorizar ataques de seguridad, con diferentes niveles de vulnerabilidad, para medir posteriormente la eficacia de los ataques, según su naturaleza, y origen.

La aplicación a diseñar, será una aplicación web, que se centrará en tener ciertas vulnerabilidades, concretamente de tipo *SQL Injection* a nivel de base de datos de la aplicación, cuyo origen es el filtrado incorrecto de las variables utilizadas como parte del programa con código SQL.

Los objetivos específicos planteados son los siguientes:

- Estudiar las diferentes vulnerabilidades y amenazas a las que están expuestos los sistemas de información de forma que se pueda establecer el marco teórico objeto del presente proyecto.
- Aplicar tecnologías web para lograr una aplicación intuitiva y fácil de usar para cualquier usuario, pero garantizando la existencia de vulnerabilidades fácilmente detectables que permitan identificar acciones de mejora que permitan subsanar estas situaciones.
- Realizar un diseño de una base de datos que reúna todas las características para poder cumplir los requisitos mínimos estudiados previamente.
- Realizar la aplicación de tal forma que sea fácilmente escalable para soportar la aparición de posibles nuevas necesidades.
- Realización de pruebas de vulnerabilidad SQL recogido en los requisitos iniciales.
- Conclusiones

2.2. Base teórica

Para poder comprender completamente el contenido y objetivos del presente proyecto, es necesario establecer en primer lugar una base teórica, con el fin de que el lector y todo aquel interesado tenga los conocimientos suficientes para entender tanto el contexto y conceptos básicos de seguridad informática, como los diferentes elementos que componen el diseño y construcción de una aplicación Web basada en Java.

2.2.1. Arquitectura Cliente-Servidor

Un sistema cliente-servidor es una arquitectura software en la cual piezas separadas del mismo, que pueden correr en máquinas diferentes, trabajan de forma integrada para ejecutar una aplicación.

Esta tecnología proporciona al usuario final el acceso transparente a las aplicaciones de datos o cualquier otro recurso en múltiples plataformas.

La arquitectura cliente-servidor soporta un entorno distribuido, con una clara separación de funciones pudiendo situar las aplicaciones en la plataforma más adecuada dentro de un entorno de sistemas abiertos y heterogéneos.

En esta arquitectura, un cliente inicia el diálogo, enviando un mensaje de petición al servidor, el cual está esperando pasivamente peticiones de los clientes. El servidor realiza el

trabajo y devuelve los datos solicitados o un código de error para indicar por qué la tarea no se ha podido realizar.

Figura 1 – Arquitectura Cliente- Servidor

En una arquitectura Cliente-Servidor, éste puede interactuar con varios clientes simultáneamente, controlando el acceso a sus recursos para garantizar la seguridad. Dado que las aplicaciones cliente direccionan de manera lógica, a través de *URLs*, la localización física del servidor y sus características técnicas (hardware, sistema operativo, sistema en granja, etc.) le es indiferente.

Este tipo de sistemas permiten un despliegue independiente de cliente y servidor, los cuales pueden usar el hardware y sistema operativo más adecuados cada uno para su función.

2.2.2. Aplicación Web

Una aplicación *Web (Web-Based Application*) es un tipo especial de aplicación cliente-servidor, donde tanto el cliente (el navegador o explorador) como el servidor (el servidor web) y el protocolo mediante el que se comunican (*HTTP*) están estandarizados y no han de ser creados por el programador de aplicaciones.

Desde el punto de vista del usuario, este tipo de aplicaciones consisten en un enorme conjunto de documentos llamados páginas, cada una de las cuales pueden contener vínculos o enlaces con otras páginas relacionadas a modo de gran repositorio de información. En un servicio Web los clientes demandan *hipertextos* a los servidores por medio de direcciones bajo el esquema *URL* (*Universal Resource Locator)* el cual permite localizar recursos en la red, incluyendo en la red de Internet.

La arquitectura de las aplicaciones Web suelen presentar un esquema de tres capas:

- El primer nivel consiste en la ***capa de presentación*** que incluye no sólo el navegador, sino también el servidor Web que es el responsable de dar a los datos un formato adecuado, que sea interpretable por el navegador (p.e. lenguajes de marcado como *HTML*)
- El segundo nivel está referido normalmente a algún tipo de ***programa o script*** responsable de llevar a cabo las tareas necesarias para completar la petición realizada por el cliente, representa la ***lógica de negocio***.
- En el tercer nivel se sitúa la base de datos (o cualquier otro ***repositorio de almacenamiento de información***) la cual proporciona al segundo nivel los ***datos necesarios*** para su ejecución.

En este caso, el navegador envía peticiones a la capa intermedia, la cual ofrece servicios ayudándose de consultas y actualizaciones a la base de datos y, además proporciona una interfaz de usuario.

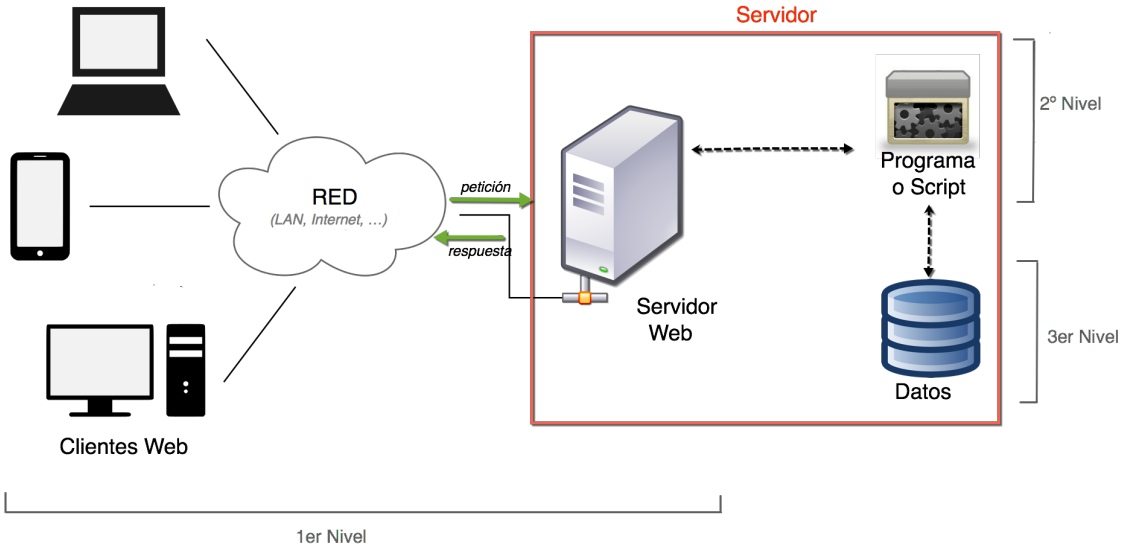


Figura 2 – Esquema Aplicación Web

2.2.2.1. *El cliente*

El cliente web es un programa con el que interacciona el usuario para solicitar a un servidor web el envío de los recursos que desea obtener mediante *HTTP*.

La parte cliente de las aplicaciones Web suele estar formada por código *HTML* que forma la página web más algo de código ejecutable en el navegador, realizado con lenguaje script (*JavaScript, JScript, jQuery o VBScript)* o mediante pequeños programas realizados en Java (*Applets)*. También se suelen utilizar *plug-ins* que permiten visualizar otros contenidos multimedia. Es decir, la misión del cliente web es interpretar las páginas *HTML* y los diferentes recursos que contienen (imágenes, sonidos, etc.).

2.2.2.2. *Servidor Web*

El servidor web es un programa que se ejecuta en el servidor y escucha el puerto TCP en espera de las conexiones *HTTP* entrantes de los clientes. Su función principal es satisfacerlas, devolviendo la información solicitada en forma de mensajes según este mismo protocolo.

2.2.2.3. *Protocolo Web: HTTP*

HTTP (*Hypertext Transfer Protocol)* es un sencillo protocolo cliente-servidor (TCP, puerto 80) que realiza los intercambios de información entre clientes y servidores. El mecanismo de intercambio entre un cliente y un servidor está basado en los mensajes. El mensaje enviado por un cliente es una petición y el enviado por el servidor es una respuesta.

Es un protocolo a nivel de aplicación para sistemas de información distribuidos, no orientado a estado y sin conexión, esto es, una vez realizado el pedido la conexión cae, para obtener una página, aunque todos los recursos estén en el mismo servidor hacen falta tantos inicios y cierres de conexión, como elementos la componen.

El esquema de direccionamiento usa el concepto de referencia dado por *URL* para indicar la fuente donde aplicarse un método (*http://host:puerto/ruta/archivo.html*), favoreciendo de este modo que resulte comprensible.

A cada petición *http* realizada le sigue una respuesta con la siguiente información: línea de estado, versión del protocolo, mensaje éxito/error e información solicitada.

2.2.2.4. Lenguajes de programación

Con la introducción de Internet y de la Web se han abierto infinidad de posibilidades en cuanto al acceso a la información desde cualquier sitio. Esto representa un desafío para los desarrolladores de aplicaciones, ya que los avances en tecnología demandan cada vez aplicaciones más rápidas, ligeras y robustas que puedan utilizarse en la Web.

En sus inicios, la tecnología *CGI* (*Common Gateway Interface*) permitía a un cliente web solicitar datos de un programa ejecutado en el servidor Web. *CGI* especifica un estándar para transferir datos entre el cliente y el programa, cuyo resultado final de la ejecución son objetos *MIME*.

Debido a las deficiencias que presentan estos sistemas en el desarrollo de aplicaciones y en la escalabilidad de las mismas se produjo el desarrollo de *APIs* y lenguajes específicos de servidor.

Los lenguajes de programación más usados en el desarrollo Web son los siguientes:

- *PHP*
- *Java J2EE*: tecnologías *Java Servlets* y *Java Server Pages (JSP)*
- *JavaScript, jQuery*
- *Perl*
- *Ruby*
- *Python*
- *ASP/ASP.NET**

  (*) No es un lenguaje de programación en sí mismo, sino una arquitectura de desarrollo web en la que se pueden usar por debajo distintos lenguajes (por ejemplo *VB.NET* o *C#* para *ASP.NET* o *VBScript/JScript* para *ASP*).

También son muy utilizados otros lenguajes o arquitecturas que no son propiamente lenguajes de programación, como *HTML o XML*.

2.2.2.5. Ventajas y desventajas

Las aplicaciones web tienen una serie de ventajas y desventajas que se detallan a continuación:

VENTAJAS:

- Ahorra tiempo: se pueden realizar tareas sencillas sin necesidad de descargar ni instalar ningún programa.

- Puesto que no es necesario realizar ninguna instalación, no ocupan espacio en nuestro disco duro.
- Actualizaciones inmediatas: como el software lo gestiona el propio desarrollador, cuando el cliente se conecta estará usando siempre la última versión.
- Multiplataforma: se pueden usar desde cualquier sistema operativo, sin problemas de compatibilidad, puesto que sólo es necesario tener un navegador.
- Portables: son independientes del dispositivo donde se utilicen (un PC de sobremesa, un portátil...), pero la reciente tendencia al acceso a través de dispositivos móviles requiere un diseño específico de los ficheros *CSS* para no dificultar el acceso de estos usuarios.
- La disponibilidad suele ser alta porque el servicio se ofrece desde múltiples localizaciones para asegurar la continuidad del mismo.
- Los virus no dañan los datos porque éstos están guardados en el servidor de la aplicación.
- Gracias a que el acceso al servicio se realiza a una única ubicación es sencillo el acceso y compartición de datos por parte de varios usuarios, evitando así duplicidad e incongruencias en la información almacenada.
- Los navegadores ofrecen cada vez más y mejores funcionalidades para crear aplicaciones web ricas (*RIAs*).

DESVENTAJAS:

- Habitualmente ofrecen menos funcionalidades que las aplicaciones de escritorio. Se debe a que las funcionalidades que se pueden realizar desde un navegador son más limitadas que las que se pueden realizar desde el sistema operativo. Pero cada vez los navegadores están más preparados para mejorar en este aspecto. La aparición de *HTML5* representa un hito en este sentido. Es posible añadir funcionalidades a estas aplicaciones gracias al uso de *Aplicaciones de Internet Ricas* (*RIA*).
- La disponibilidad depende de un tercero, el proveedor de la conexión a internet o el que provee el enlace entre el servidor de la aplicación y el cliente. Así que la disponibilidad del servicio está supeditada a agentes externos al usuario.

2.2.3. Seguridad de la información y seguridad informática

Se entiende por seguridad de la información al conjunto de métodos o medidas técnicas, organizativas y legales cuyo objetivo es garantizar la confidencialidad, la integridad y la disponibilidad del sistema de información tratando otras propiedades como la responsabilidad, la autenticidad y la fiabilidad.

Antes de la aparición de las TIC toda la información se guardaba en papel en almacenes y grandes sistemas de archivadores. Los sistemas informáticos permiten la digitalización de todo este volumen de información, ganando además de espacio físico, velocidad en el acceso a la información, pero con ello aparecen toda una serie de implicaciones derivadas del uso de estas facilidades, ya que el sistema de información debe seguir garantizando los aspectos de confidencialidad, integridad, etc. antes descritos.

Toda organización debe tener en cuenta un Sistema de Gestión de la Seguridad de la información, es decir, lo que se conoce con las siglas *SGSI*. Éste se encarga de proteger y cuidar la información, para ello debe señalarse lo que debe ser protegido y en qué grado de protección hay que hacerlo.

Según el estándar *ISO 27000,* los requisitos necesarios para establecer, implantar, mantener y mejorar un *SGSI* deben llevarse a cabo según el conocido como "*Ciclo de Deming*": *PDCA*, es decir, Planificar, Hacer, Verificar, Actuar (*Plan-Do-Check-Act)* y volver a repetir la secuencia, ya que se entiende que la seguridad es un proceso que nunca termina puesto que los riesgos nunca se eliminan.

Figura 3 – Requisitos SGSI: Ciclo *PDCA (Plan-Do-Check-Act)*

- ***Planificar (Plan)***, se basa en establecer el contexto en el que se crean las políticas de seguridad, teniendo en cuenta el análisis de riesgos, la selección de controles y el estado de aplicabilidad.
- ***Hacer (Do),*** se centra en implementar el sistema de gestión de seguridad de la información, plan de riesgos y controles.
- ***Verificar (Check)***, su función es monitorizar las actividades y hacer auditorías internas.
- ***Actuar (Act)***, consiste en realizar tareas de mantenimiento, propuestas de mejora, acciones preventivas y correctivas.

Para poder hablar de fiabilidad de un sistema de información se deben garantizar 3 aspectos fundamentales:

1) Confidencialidad. Sólo se tiene acceso a la información mediante autorización y de forma controlada.

2) Integridad. Para modificar la información hay que tener autorización. Consiste en mantener con exactitud la información tal cual fue generada, sin ser manipulada o alterada por personas o procesos no autorizados. Hay que centrarse y observar con detenimiento la integridad del origen porque puede afectar a la exactitud, credibilidad y confianza que las personas ponen en la información.

3) Disponibilidad. La información del sistema debe ser accesible mediante autorización.

Figura 4 – Bases de la seguridad informática

Englobados en el ámbito de los aspectos fundamentales que componen las bases de la seguridad informática, se encuentran los siguientes mecanismos que permiten su consecución:

- **Autenticación**: es la verificación de la identidad del usuario, sobre todo cuando entra en el sistema o la red, o accede a una base de datos. Para entrar en el sistema informático suele utilizarse un nombre de usuario y una contraseña, pero el uso de credenciales no significa que sean las más fiables, por esta razón se utilizan otras técnicas más seguras. Por ello, cuando hay que autentificarse se puede hacer de tres maneras, a través de una contraseña, por una tarjeta magnética o por las huellas digitales. Utilizando más de un método aumenta la probabilidad de que la autenticación sea correcta.
- **Autorización**: es el proceso por el cual se determina qué, cómo y cuándo un usuario autenticado puede utilizar los recursos de la organización.
- **Administración**: es la encargada de mantener, establecer y eliminar las autorizaciones de los usuarios del sistema, los recursos del sistema y las relaciones usuarios-recursos del sistema.
- **Auditoria**: este proceso es el encargado de permitir a los administradores verificar las técnicas de autenticación y autorización utilizadas y observar si se realizan según lo establecido y si cumplen los objetivos fijados por la organización, es decir, la auditoría es la continua vigilancia de los servicios de producción.
- **Registro**: es el que permite que cualquier intento de violar las reglas de seguridad establecidas quede almacenado en una base de eventos para luego analizarlo.
- **Mantenimiento de la integridad de la información**: es el conjunto de procedimientos establecidos para controlar que los archivos sufran cambios no autorizados y que la información enviada desde un punto llegue al destino inalterada. Las técnicas más conocidas para mantener la integridad de los datos son el uso de antivirus, encriptación y funciones *´hash´*.

2.2.4. Vulnerabilidades

En un sistema informático hay que proteger todos los recursos que forman parte del sistema, esto es:

- Hardware, donde se incluyen todos los elementos físicos del sistema informático.
- Software, donde se encuentran todos los programas que se ejecutan sobre el hardware.
- Datos, aquellos que comprenden la información que procesa el software haciendo uso del hardware.
- Otros (consumibles, personas, infraestructuras, etc.).

Figura 5 – Flujo de evolución del conocimiento

De entre todos los activos, el más crítico son los datos, ya que la dificultad o imposibilidad de reponerlos conllevaría una pérdida de tiempo y dinero.

No obstante, además de proteger los recursos hay que tener en cuenta todas aquellas debilidades que pueden afectar al sistema, denominadas vulnerabilidades y que pueden agruparse en:

Diseño

- Debilidad en el diseño de protocolos utilizados en las redes.
- Política de seguridad deficiente e inexistente.

Implementación

- Errores de programación.
- Existencia de "puertas traseras" en los sistemas informáticos.
- Descuido de los fabricantes.

Uso

- Mala configuración de los sistemas informáticos.
- Desconocimiento y falta de sensibilización de los usuarios y de los responsables de informática.
- Disponibilidad de herramientas que facilitan los ataques.
- Limitación gubernamental de tecnologías de seguridad.

Cualquier elemento que muestre información puede conllevar un riesgo y no todos los riesgos son idénticos. Algunos de ellos necesitan un cambio en la configuración, otros en el procedimiento y otros en el código.

Una vulnerabilidad es la capacitad y posibilidad de un sistema de responder o reaccionar a una amenaza o de recuperarse de un daño.

Las vulnerabilidades están en directa interrelación con las amenazas porque si no existe una amenaza, tampoco existe la vulnerabilidad o no tiene importancia, porque no puede ocasionar un daño.

Dependiendo del origen de la vulnerabilidad, ésta puede tener una o varias de las siguientes características:

Tipo de vulnerabilidad	Definición
Directivas de seguridad	Una directiva de seguridad es un registro de los procesos y procedimientos que debe seguir una organización para evitar realizar un seguimiento y responder a las amenazas de seguridad.
Principio de "privilegios mínimos"	Según el principio de "privilegios mínimos", un sistema sólo debería permitir el nivel de acceso necesario a un objeto protegible. Este acceso debería estar habilitado para quienes tienen una necesidad directa y sólo durante un tiempo específico.
Boletines de seguridad	Las organizaciones que no están al corriente de estos boletines ponen en peligro sus sistemas al no implementar las instrucciones de seguridad adecuadas.
El sistema no está actualizado	Microsoft publica actualizaciones de software para mejorar la seguridad de SQL Server. Si no se siguen o se aplican estas actualizaciones de software, el sistema será más vulnerable a los ataques.
Contraseñas no seguras	Las contraseñas sencillas están expuestas a los ataques por fuerza bruta o de diccionario.
Cuentas de usuario sin	Los usuarios (entidades de seguridad) a menudo cambian de puesto o dejan la organización. Si no se cambia el acceso a una cuenta de usuario, se puede seguir

auditar	teniendo acceso al sistema con el nivel de permisos anterior.
Inyección de código SQL	Fallo en la aplicación en la validación de introducción de instrucciones SQL que permite la obtención de datos dinámicos.
Contraseñas incrustadas	Algunas aplicaciones guardan las cadenas de conexión en el programa o en archivos de configuración.
Cifrado aplicado incorrectamente	El cifrado ofusca los datos o la información de conexión. No cifrar los datos cuando es necesario o hacerlo cuando no lo es, supone un riesgo y una complejidad innecesarios.
Certificados aplicados incorrectamente	Los certificados son un mecanismo para comprobar la autenticación. SQL Server puede utilizar los certificados para diversos propósitos, desde las conexiones a los datos. El uso inadecuado de los certificados autofirmados y los períodos de validación extendidos reducen la eficacia de la seguridad global.
Desbordamiento de buffer	El programa debe controlar la cantidad de datos que se copian en el buffer, si no es así puede llegar a sobrepasarse la capacidad de *buffer* y los *bytes* que sobran se almacenan en zonas de memoria paralelas.
Condición de carrera	Este tipo de vulnerabilidad únicamente ocurre cuando varios procesos acceden al mismo tiempo a un recurso compartido.
Cross Site Scripting (XSS).	Se centra en todo aquello que permita ejecutar código "*scripting*", como *VBScript* o *JavaScript*, en el contexto de otro dominio. Se pueden encontrar estos errores en aplicaciones del tipo *HTML*. Puede ser de manera indirecta cuando modifica valores que la aplicación web utiliza para pasar variables entre dos páginas sin usar sesiones o directa cuando localiza puntos débiles en la programación de los filtros.
Denegación del servicio	Permite que un recurso no esté disponible para los usuarios a través de la pérdida de la conectividad de la red.
Window Spoofing	Estas ventanas son las que hacen creer al usuario que es ganador de alguna cosa. Su objetivo es que el usuario dé información.
Error de configuración	Un problema de configuración general de software no relacionado con contraseñas o permisos. Suele ser la que provoca la mayoría de DoS aunque éstos también pueden ser provocados por otros tipos.
Validación de entrada	Fallo en la validación de la secuencia de entrada de datos introducidos en aplicaciones que afecta al flujo de control o de datos de un programa.
Salto de directorio	Utiliza la falta de seguridad de un servicio de red para moverse por los directorios hasta la raíz del volumen del sistema.
Seguimiento de enlaces	Debilidad en la protección frente a accesos no solicitados por parte de enlaces o accesos directos.
Inyección de comandos en el sistema operativo	Capacidad de un usuario que controla la entrada de comandos para realizar instrucciones que comprometan la integridad del sistema.
Inyección directa de código estático	El software permite que las entradas sean introducidas directamente en un archivo de salida que se procese más adelante como código, un archivo de la biblioteca o una plantilla.
Evaluación directa de código dinámico	En la que el software permite que las entradas sean introducidas directamente en una función que evalúa y ejecuta dinámicamente la entrada como código, generalmente en la misma lengua que usa el producto.
Inclusión remota de	Vulnerabilidad existente únicamente en páginas dinámicas escritas en PHP es debida a la inclusión de la función *include*() la cual permite el enlace de archivos

archivo PHP	situados en otros servidores, mediante los cuales se puede ejecutar código PHP en el servidor.
Error de búfer	Una aplicación incluye datos en una dirección de memoria no asignada previamente que permite la lectura o escritura de direcciones de memoria no prevista por la aplicación.
Formato de cadena	Cuando se produce a través de cadenas de formato controladas externamente, como el tipo de funciones "*printf*" en el lenguaje "*C*" que pueden conducir a provocar desbordamientos de búfer o problemas en la representación de los datos.
Desbordamiento de entero	Un desbordamiento del número entero ocurre cuando una operación aritmética procura crear un valor numérico que sea más grande del que se puede representar dentro del espacio de almacenaje disponible.
El agotamiento de entero	Consiste en que un valor se resta de otro, que es menor que el valor mínimo del número entero, y que produce un valor que no es igual que el resultado correcto.
Revelación o filtración	En este aspecto los atacantes pueden aprovechar esta vulnerabilidad para descubrir el directorio de instalación de una aplicación, la visualización de mensajes privados, etc.
Gestión de credenciales	Debilidad en el control de guardado, protección o transmisión de credenciales y contraseñas.
Permisos, privilegios y control de acceso	Fallos en la protección y gestión de permisos que permiten el control de acceso.
Fallo de autenticación	Esta vulnerabilidad se produce cuando la aplicación o el sistema no es capaz de autenticar al usuario, proceso, etc. correctamente.
Carácter criptográfico	La generación de números aleatorios para generar secuencias criptográficas, la debilidad o distintos fallos en los algoritmos de encriptación así como defectos en su implementación estarían ubicados dentro de este tipo de vulnerabilidad.
Falsificación de petición en sitios cruzados (CSRF)	Este tipo de vulnerabilidad afecta a las aplicaciones web con una estructura de invocación predecible. El tipo de ataque *CSRF* más popular se basa en el uso del marcador *HTML* <*img*>, el cual sirve para la visualización de gráficos.
Condición de carrera	Se produce cuando varios procesos tratan de acceder y manipular los mismos datos simultáneamente.
Error en la gestión de recursos	El sistema o software que adolece de este tipo de vulnerabilidad permite al atacante provocar un consumo excesivo en los recursos del sistema (disco, memoria y *CPU*).
Error de diseño	Se tratan de los errores que provocan fallos en la seguridad y que no son de implementación o configuración, sino del propio diseño del sistema o de la arquitectura del software.

Tabla 1 – Listado de Vulnerabilidades CWE (*Common Weakness Enumeration*)

2.2.4.1. Common Vulnerabilities and Exposures (CVE)

Las vulnerabilidades son difíciles de gestionar. Se descubren decenas día a día, y clasificarlas es una tarea compleja.

El estándar *CVE* (*Common Vulnerabilities and Exposures*), se encarga de identificar de manera unívoca cada referencia registrada sobre conocidas vulnerabilidades de seguridad.

De esta forma provee una nomenclatura común para el conocimiento público de este tipo de problemas y así facilitar la compartición de datos sobre dichas vulnerabilidades.

Fue definido y es mantenido por *The MITRE Corporation (MITRE CVE List*) con fondos de la *National Cyber Security Division* del gobierno de los Estados Unidos. Forma parte del llamado *Security Content AutomationProtocol*.

La información y nomenclatura de esta lista es usada en la *National Vulnerability Database*, el repositorio de información sobre vulnerabilidades de los Estados Unidos.

2.2.4.2. Common Vulnerability Scoring System (CVSS)

Es también un estándar que gradúa la severidad a través de fórmulas establecidas, de manera que los administradores o usuarios puedan conocer de manera objetiva la gravedad de los fallos que afectan a sus sistemas.

CVSS clasifica la facilidad de aprovechar el fallo y el impacto del problema teniendo en cuenta qué nivel de compromiso de la confidencialidad, integridad y disponibilidad de los datos se podrían obtener aprovechando la vulnerabilidad.

2.2.5. Amenazas

Se puede definir como amenaza a todo elemento o acción capaz de atentar contra la seguridad de la información, provocando un daño (material o inmaterial).

Las amenazas surgen a partir de la existencia de vulnerabilidades, es decir que una amenaza sólo puede existir si existe una vulnerabilidad que pueda ser aprovechada, e independientemente de que se comprometa o no la seguridad de un sistema de información.

La presencia de una amenaza es una advertencia de que puede ser inminente el daño a algún activo de la información, o bien es un indicador de que el daño se está produciendo o ya se ha producido.

La mayor parte de los ataques a los sistemas informáticos son provocados, intencionadamente o no, por las personas que en general persiguen conseguir un nivel de privilegio en el sistema que les permita realizar acciones no autorizadas sobre el mismo.

Podemos clasificar las personas 'atacantes' en dos grupos:

- **Activos**: su objetivo es hacer daño de alguna forma. Eliminar información, modificarla o sustraerla para su provecho.
- **Pasivos**: su objetivo es curiosear en el sistema.

Sin llegar a entrar en detalle sobre todos los tipos de personas que pueden constituir una amenaza para el sistema, cabe destacar a los curiosos, intrusos remunerados, crackers, terroristas, ex-empleados e incluso el personal de la propia organización.

De forma general las amenazas pueden clasificarse en 2 grandes grupos: *amenazas físicas* y *amenazas lógicas*, pudiendo ser materializadas por personas, programas específicos o catástrofes naturales (inundación, incendio, fallo eléctrico, explosión, etc.).

Además de esta, pueden realizarse diversas clasificaciones atendiendo a sendos criterios que se detallan a continuación como pueden ser el origen, intencionalidad, etc.

- ***Agrupación según el origen de las amenazas***

- *Amenazas naturales* (provocadas por catástrofes naturales).
- *Amenazas de agentes externos*: virus informáticos, ataques de una organización criminal, sabotajes, disturbios y conflictos sociales, intrusos en la red, robos, estafas, etc.
- *Amenazas de agentes internos*: empleados descuidados, con una formación inadecuada o descontentos, errores en la utilización de las herramientas y recursos del sistema, etc.

- ***Agrupación según la intencionalidad de las amenazas***
  - *Accidentes:* averías del hardware, fallos del software, inundación, incendio, etc.
  - *Errores:* errores de uso, de explotación, de ejecución de procedimientos, etc.
  - *Actuaciones malintencionadas:* robos, fraudes, sabotajes, intrusiones, etc.

En el flujo normal de información se habla de la fiabilidad del sistema si se garantiza la confidencialidad, integridad y disponibilidad. Esto es, que nadie no autorizado tenga acceso a la información y que los datos enviados no se modifican en el camino, siendo entregados de manera correcta al destinatario.

En este sentido, atendiendo al factor de seguridad que comprometen, puede realizarse la distinción de los siguientes tipos:

1) ***Interceptación***: acceso a la información por parte de personas no autorizadas y/o uso de privilegios no adquiridos.
   Este ataque es difícil de detectar ya que no deja huellas. Compromete la confidencialidad, ya que alguien no autorizado podría tener acceso a la información.

Figura 6 – Flujo Amenaza Interceptación

2) ***Modificación***: acceso no autorizado que cambia el entorno para su beneficio.
   En este caso se compromete la integridad y confidencialidad porque los datos pueden ser modificados en el camino y ser accesibles para alguien no autorizado.

Figura 7 – Flujo Amenaza Modificación

3) ***Interrupción***: puede provocar que un objeto del sistema se pierda, quede no utilizable o no disponible.
   No se garantiza la disponibilidad ya que puede que la recepción no sea correcta.

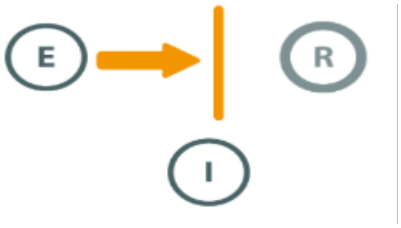


Figura 8 – Flujo Amenaza Interrupción

4) ***Fabricación***: puede considerarse como un caso concreto de modificación ya que se consigue un objeto similar al atacado de forma que no resulte sencillo distinguir entre el objeto original y el fabricado.
Este tipo de amenazas puede ser empleado en delitos de falsificación.

Figura 9 – Flujo Amenaza Fabricación

2.2.5.1. Amenazas físicas

Las amenazas físicas y ambientales afectan a las instalaciones y/o el hardware contenido en ellas.

Constituyen el primer nivel de seguridad a proteger, por este motivo evaluar y controlar permanentemente la seguridad física del edificio que alberga el CPD es la base para comenzar a integrar la seguridad como una función primordial dentro de cualquier organismo.

Tener controlado el ambiente y acceso físico permite:

- Disminuir siniestros
- Trabajar mejor manteniendo la sensación de seguridad
- Descartar falsas hipótesis si se produjeran incidentes
- Tener los medios para luchar contra accidentes

Una vez explicado el concepto de amenaza física y la importancia que suscita para el sistema de seguridad, a continuación se van a describir cuáles son las principales amenazas físicas de un sistema informático.

- ***Acceso físico***
  Hay que tener en cuenta que cuando existe acceso físico a un recurso ya no existe seguridad sobre él. Supone entonces un gran riesgo y probablemente con un impacto muy alto.

  A menudo se descuida este tipo de seguridad.

- ***Radiaciones electromagnéticas***
  Sabemos que cualquier aparato eléctrico emite radiaciones y que dichas radiaciones se pueden capturar y reproducir, si se dispone del equipamiento

adecuado. Por ejemplo, un posible atacante podría 'escuchar' los datos que circulan por el cable telefónico.

- ***Desastres naturales***
  Respecto a terremotos, el riesgo es reducido en nuestro entorno, ya que España no es una zona sísmica muy activa. Pero son fenómenos naturales que si se produjeran tendrían un gran impacto y no sólo en términos de sistemas informáticos.

- ***Desastres del entorno***
  Dentro de este grupo estarían incluidos sucesos que, sin llegar a ser desastres naturales, pueden tener un impacto igual de importante si no se disponen de las medidas de salvaguarda listas y operativas.

  Puede ocurrir un incendio o un apagón y no tener bien definidas las medidas a tomar en estas situaciones o simplemente no tener operativo el *SAI* que debería responder de forma inmediata al corte de suministro eléctrico.

2.2.5.2. Amenazas lógicas

El punto más débil de un sistema informático son las personas relacionadas en mayor o menor medida con él. Puede ser inexperiencia o falta de preparación, o sin llegar a ataques intencionados propiamente, simplemente sucesos accidentales, que en cualquier caso, hay que prevenir.

Las amenazas lógicas comprenden una serie de programas que pueden dañar el sistema informático. Y estos programas han sido creados:

- de forma intencionada para hacer daño: software malicioso o ***malware*** (**mal**icious soft**ware**)
- por error: *bugs* o agujeros.

Entre algunos de los ataques potenciales que pueden ser causados por personas, se encuentran:

- ***Ingeniería social:*** consiste en la manipulación de las personas para que voluntariamente realicen actos que normalmente no harían.
- ***ShoulderSurfing***: consiste en "espiar" físicamente a los usuarios para obtener generalmente claves de acceso al sistema.
- ***Masquerading***: consiste en suplantar la identidad de cierto usuario autorizado de un sistema informático o su entorno.
- ***Basureo***: consiste en obtener información dejada en o alrededor de un sistema informático tras la ejecución de un trabajo.
- ***Actos delictivos***: son actos tipificados claramente como delitos por las leyes, como el chantaje, el soborno o la amenaza.
- ***Atacante interno***: la mayor amenaza procede de personas que han trabajado o trabajan con los sistemas. Estos posibles atacantes internos deben disponer de los privilegios mínimos, conocimiento parcial, rotación y separación de funciones, etc.

- ***Atacante externo***: suplanta la identidad de un usuario legítimo. Si un atacante externo consigue penetrar en el sistema, ha recorrido el 80% del camino hasta conseguir un control total de un recurso.

Del mismo modo que se pueden analizar las amenazas desde un punto de vista de quién o qué lo genera, los tipos de amenazas pueden clasificarse en función de la técnica que se emplea para realizar el ataque. En la siguiente tabla se incluyen las técnicas más usuales:

Técnica de ataque	Definición
Software incorrecto	Son errores de programación (*bugs*) y los programas utilizados para aprovechar uno de estos fallos y atacar al sistema son los *exploits*.
Exploits	Son los programas que aprovechan una vulnerabilidad del sistema. Son específicos de cada sistema operativo, de la configuración del sistema y del tipo de red en la que se encuentren. Pueden haber *exploits* diferentes en función del tipo de vulnerabilidad. Es la amenaza más habitual.
Herramientas de seguridad	Puede ser utilizada para detectar y solucionar fallos en el sistema o un intruso puede utilizarlas para detectar esos mismos fallos y aprovechar para atacar el sistema.
Puertas traseras	Durante el desarrollo de aplicaciones los programadores pueden incluir 'atajos' en los sistemas de autenticación de la aplicación. Estos atajos se llaman puertas traseras, y con ellos se consigue mayor velocidad a la hora de detectar y depurar fallos. Si estas puertas traseras, una vez la aplicación ha sido finalizada, no se destruyen, se está dejando abierta una puerta de entrada rápida.
Bombas lógicas	Son partes de código que no se ejecutan hasta que se cumple una condición. Al activarse, la función que realizan no está relacionada con el programa, su objetivo es completamente diferente.
Virus	Secuencia de código que se incluye en un archivo ejecutable (llamado huésped), y cuando el archivo se ejecuta, el virus también se ejecuta, propagándose a otros programas.
Gusanos	Programa capaz de ejecutarse y propagarse por sí mismo a través de redes, y puede llevar virus o aprovechar bugs de los sistemas a los que conecta para dañarlos.
Caballos de Troya	Los caballos de Troya son instrucciones incluidas en un programa que simulan realizar tareas que se esperan de ellas, pero en realidad ejecutan funciones con el objetivo de ocultar la presencia de un atacante o para asegurarse la entrada en caso de ser descubierto.
Spyware	Programas espía que recopilan información sobre una persona o una organización sin su conocimiento. Esta información luego puede ser cedida o vendida a empresas publicitarias. Pueden recopilar información del teclado de la víctima pudiendo así conocer contraseña o nº de cuentas bancarias o pines.
Adware	Programas que abren ventanas emergentes mostrando publicidad de productos y servicios. Se suele utilizar para subvencionar la aplicación y que el usuario pueda bajarla gratis u obtener un descuento. Normalmente el usuario es consciente de ello y da su permiso.
Sniffing	Rastrear monitorizando el tráfico de una red para hacerse con información

	confidencial.
Spoofing	Técnicas de suplantación de identidad con fines dudosos, entre otras pueden encontrarse *IP*, *MAC*, tabla *ARP*, web o mail *spoofing*.
Pharming	Redirigir un nombre de dominio a otra máquina distinta falsificada o fraudulenta.
Phising	Intenta conseguir información confidencial de forma fraudulenta (conseguir contraseñas o pines bancarios) haciendo una suplantación de identidad. Para ello el estafador se hace pasar por una persona o empresa de la confianza del usuario mediante un correo electrónico oficial o mensajería instantánea, y de esta forma conseguir la información.
Spam	Recepción de mensajes no solicitados. Se suele utilizar esta técnica en los correos electrónicos, mensajería instantánea y mensajes a móviles.
Programas conejo o bacterias	Programas que no hacen nada, sólo se reproducen rápidamente hasta que el número de copias acaba con los recursos del sistema (memoria, procesador, disco, etc.).
Técnicas salami	Robo automatizado de pequeñas cantidades dinero de una gran cantidad origen. Es muy difícil su detección y se suelen utilizar para atacar en sistemas bancarios.
Password cracking	Descifrar contraseñas de sistemas y comunicaciones. Los métodos más comunes son mediante *sniffing* (observando la introducción de credenciales), por ataques de fuerza bruta (probando todas las posibilidades) o de diccionario (con un conjunto de palabras comúnmente empleadas para contraseñas).
Botnet	Conjunto de robots que se ejecutan de manera autónoma y automática en multitud de *host*, normalmente infectados, permitiendo controlar todos los ordenadores infectados de manera remota. Sus fines normalmente es rastrear información confidencial o incluso cometer actos delictivos.
Denegación del servicio (DoS)	Causar que un recurso o servicio sea inaccesible a usuarios legítimos. Una ampliación es el ataque distribuido de denegación de servicio (*DDoS*), a través de una *botnet*, siendo la técnica de ataque más usual y eficaz.

Tabla 2 – Amenazas lógicas: Técnicas de ataque más usuales

2.2.6. Inyección SQL

La inyección directa de comandos SQL es una técnica donde un atacante crea o altera comandos SQL existentes para exponer datos ocultos, sobrescribir los valiosos, o ejecutar comandos peligrosos a nivel de sistema en el equipo que hospeda la base de datos. Esto se logra a través de la práctica de tomar la entrada del usuario y combinarla con parámetros estáticos para elaborar una consulta SQL.

El primer paso para realizar un ataque por inyección de SQL es encontrar un sitio web vulnerable. Este es seguramente el proceso que consume más tiempo de todo el ataque. Cada vez más páginas web se están protegiendo de las inyecciones SQL por lo que encontrar un objetivo vulnerable puede llevar algún tiempo.

Una de las formas más sencillas de encontrar sitios vulnerables se conoce como Google *Dorking*. En este contexto, un *dork* es una consulta de búsqueda específica que encuentra sitios web que cumplen los parámetros de la consulta avanzada que introduces. Algunos

ejemplos que se pueden utilizar para encontrar sitios vulnerables de un ataque por inyección de SQL son:

- inurl:index.php?id=
- inurl:buy.php?category=
- inurl:article.php?ID=

Una forma rápida de comprobar si una aplicación es vulnerable a inyecciones de SQL podría ser enviar al intérprete una comilla simple (‘) para terminar la instrucción en ese momento y que todo lo demás sea código no ejecutable por el intérprete de SQL y esto provoque un error.

En un formulario de *login* con un campo e-mail y contraseña la consulta construida sería:

```
SELECT * FROM Usuarios WHERE Email = 'xxxxx' AND [Password] = ''';
```

Al final de la consulta todavía aparece una pequeña fracción de código ‘; que el intérprete no puede ejecutar correctamente. Lo que provocará un error en el intérprete de SQL y por extensión en la aplicación.

En este caso, el como la aplicación gestione este tipo de errores es lo de menos. Si nosotros como atacantes recibimos una pantalla de error, igualmente habremos conseguido nuestro propósito, averiguar si la aplicación es susceptible de ser atacada.

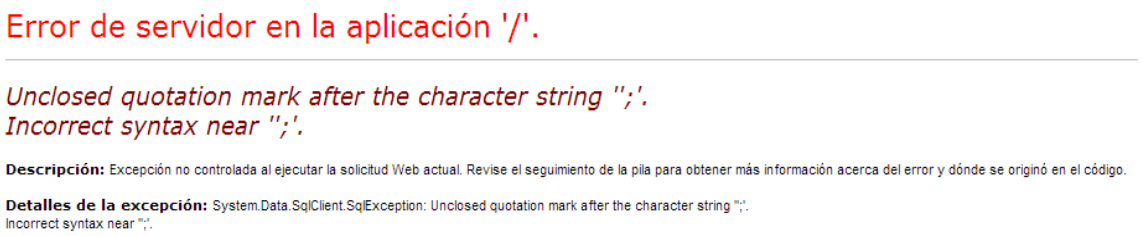


Figura 10 – Pantalla error verificación vulnerabilidad SQL

Atendiendo a su definición, el ataque a realizar se basa en modificar una sentencia SQL ya escrita y preestablecida por el diseñador de una página web. Esta sentencia no siempre será accesible y no tiene por qué ser visible. Depende de las habilidades del atacante identificar la estructura o forma de la sentencia que se realizan en este lenguaje de consulta. Además, en aquellos casos en los que el código es inyectable, hay muchos niveles de complejidad para realizar el ataque.

El proceso de inyección consiste en finalizar prematuramente una cadena de texto y anexar un nuevo comando. Como el comando insertado puede contener cadenas adicionales que se hayan anexado al mismo antes de su ejecución, el atacante debe poner fin a la cadena inyectada con una marca de comentario *"--"*, de este modo, el texto situado a continuación se omite en tiempo de ejecución.

Una forma típica de inyectar SQL en una aplicación es cuando inyectamos una instrucción siempre verdadera consiguiendo que la instrucción *WHERE* siempre se cumpla y nos devuelva siempre resultados sin conocer los datos auténticos: [`‘ or ‘1’=’1 ]`

Supongamos también que la aplicación web tiene el típico enlace “*Si no recuerdas la contraseña haz clic aquí*” en la que se pedirá un email para enviar la contraseña automáticamente por correo electrónico.

De nuevo desde la pantalla de *login* se puede modificar la dirección de correo para el acceso a la aplicación, una vez se conozca la dirección de correo de un usuario autentificado, introduciendo algo parecido a esto:

```
'; update Usuarios set email = 'miEmail@hacker.com' where email =
        'emailConocido@host.com'; --' AND [Password] = '';
```

Ahora únicamente es necesario ir al formulario donde se pide el email para enviar la contraseña e introducir el email malicioso para obtener la contraseña de este usuario.

Como se puede ver, las inyecciones SQL no sólo permiten el acceso a información confidencial o sensible, sino que permiten modificar información de la base de datos, e incluso tener acceso a información del sistema.

Cada base de datos introduce sus propios elementos, con frecuencia incompatibles con los de la competencia, esto hace que sentencias SQL que funcionan sin problema en *PostgreSQL* den lugar a errores en *ORACLE* o que instrucciones válidas en *MySQL* no lo sean en *Ms SQL Server*.

La primera de ellas es relativa a la obligatoriedad de incluir la cláusula *FROM* en sentencias *SELECT,* éste dato puede arrojar luz sobre qué gestor de base de datos se está utilizando, porque por ejemplo *ORACLE* siempre obliga a la existencia de la cláusula, *MySQL* sólo si la sentencia contiene una cláusula *WHERE* y ni *PostgreSQL* ni *SQL Server* obligan a su utilización.

A la hora de explotar una vulnerabilidad *SQL Injection* el atacante debe comenzar por obtener toda la información posible sobre la base de datos: qué motor usa, qué versión, con qué cuenta se conecta a la aplicación, qué tablas existen, etc. Esto le permite hacerse una idea de a qué se enfrenta y qué puede obtener.

Para obtener información sobre la base de datos se puede recurrir a las siguientes consultas:

Información	Base de datos	Sintaxis
Versión	ORACLE	`select version from v$instance;`
	MySQL	`select version();` `select @@version();`
	PostgreSQL	`select version();`
	SQL Server	`select @@version;`
Nombre base de datos (Instancia)	ORACLE	`select name from v$database;`
	MySQL	`select database();`
	PostgreSQL	`select current_database();`
	SQL Server	`select DB_NAME();`
Catálogo de recursos	MySQL	`select schema_name` `  from information_schema.schema;`
	PostgreSQL	`select datname, datcl` `  from pg_database;`
	SQL Server	`select name from master.sysdatabases;`

Información del esquema	ORACLE	`select table_name, tablespace_name`
		`  from all_tables;`
		`select column_name, data_type`
		`  from all_tab_columns`
		`  where table_name = '[tabla]';`

	MySQL	`select table_schema, table_name`
		`  from information_schema.tables`
		`  where table_type = 'BASE TABLE';`
		`select column_name, data_type`
		`  from information_schema.columns`
		`  where table_name = '[tabla]';`

	PostgreSQL	`select relname, nspname`
		`  from pg_class, pg_namespace`
		`  where pg_namespace.oid = relnamespace;`
		`select attnum, attname, typname`
		`  from pg_class, pg_namespace,`
		`  pg_attribute, pg_type`
		`  where pg_type.oid = attpid`
		`  and attrelid = pg_class.oid`
		`  and relname = '[tabla]'`
		`  and pg_class.relnamespace =`
		`  pg_namespace.oid;`

	SQL Server	`USE [base datos];`
		`select table_name`
		`  from information_schema.tables`
		`  where table_type = 'BASE TABLE';`
		`select column_name, data_type`
		`  from information_schema.columns`
		`  where table_name = '[tabla]';`

Tabla 3 – Consultas información base de datos

Los ataques por inyección SQL tienen una variedad de posibilidades muy amplias, pero su efectividad será mayor o menor, además de por la habilidad y conocimientos de atacante y defensor, por otros factores como pueden ser el tipo de arquitectura de la base de datos, su versión, formato de la web, interfaz que se utilice, etc. Esto hace que el ataque se vea muy mermado, pero siempre que el atacante tenga altos conocimientos se puede garantizar el ataque en la mayoría de los casos, ya que las defensas que se puedan disponer contra esta vulnerabilidad pueden tener muchas formas de ser sorteadas, desde cambios en la forma de insertar el código hasta *exploits* basados en determinadas versiones y/o tipo de sistema.

Debido a la falta de validación en la entrada de datos y a la conexión a la base de datos con privilegios de *superusuario* o de alguien con privilegios para crear usuarios, el atacante podría crear un *superusuario* en la base de datos.

Las páginas Web centradas en *PHP* son su objetivo primordial ya que, pueden ser configuradas mundialmente y a través de ellas se puede conocer información útil acerca de clientes dentro de la base de datos que se intenta atacar. Por ejemplo: *WordPress*

Pese a que pueda parecer obvio que un atacante debe tener al menos algún conocimiento de arquitecturas de bases de datos para poder realizar un ataque con éxito, la obtención de esta información suele ser muy sencilla. Por ejemplo, cuando la base de datos forma parte de un software de código abierto o disponible públicamente con una instalación predefinida, dicha información se encuentra completamente libre y utilizable.

Siempre y cuando el código SQL inyectado sea sintácticamente correcto, no será posible detectar alteraciones mediante programación. Por ello, debe validar todos los datos especificados por el usuario y revisar cuidadosamente el código que ejecute comandos SQL construidos en el servidor que utilice. A continuación se incluyen algunas prácticas recomendadas de codificación para prevenir y minimizar la vulnerabilidad del sistema:

- Validar todos los datos especificados por el usuario mediante comprobaciones de tipo, longitud, formato e intervalo. En entornos de varios niveles, todos los datos deben validarse antes de que se admitan en la zona de confianza, no creando nunca instrucciones SQL directamente a partir de datos indicados por el usuario.
- No aceptar las siguientes cadenas en campos a partir de los que puedan construirse nombres de archivo: *AUX, CLOCK$, COM1 a COM8, CON, CONFIG$, LPT1 a LPT8, NUL y PRN.*

- Si es posible, rechazar los datos que contengan los siguientes caracteres:

Carácter de entrada	Significado
;	Delimitador de consultas.
'	Delimitador de cadenas de datos de caracteres.
--	Delimitador de comentarios.
/* ... */	Delimitadores de comentarios. El servidor no evalúa el texto incluido entre /* y */.
xp_	Se utiliza al principio del nombre de procedimientos almacenados extendidos de catálogo, como **xp_cmdshell**.

Tabla 4 – Significado caracteres de entrada especiales (SQL Injection)

- Usar parámetros SQL con seguridad de tipos.
  - La colección *Parameters* de *SQL Server* proporciona comprobación de tipos y validación de longitud.
  - En lenguaje Java se puede utilizar *PreparedStatement* en lugar de *Statement*.
  - Se puede usar *parametrización* o escape de variables.
- Usar una entrada con parámetros en los procedimientos almacenados puesto que pueden ser susceptibles a una inyección de código si utilizan una entrada sin filtrar.
- Filtrar la entrada mediante la eliminación de los caracteres de escape. No obstante, debido al gran número de caracteres que pueden presentar problemas no es una defensa confiable.
- Si se utiliza una cláusula *LIKE*, los caracteres comodín seguirán necesitando utilizar caracteres de escape.

Tal y como se acaba de mencionar, en lenguaje Java se pueden usar consultas parametrizadas, *PreparedStatement,* en lugar de concatenación de variables, *Statement,* para prevenir inyecciones SQL en el sistema.

Las consultas parametrizadas no concatenan las variables a la consulta SQL sino que usan sintaxis específica para pasar a la consulta SQL un conjunto de parámetros predeterminado.

```
Connection con = (acquire Connection)
PreparedStatement pstmt = con.prepareStatement(
                    "SELECT * FROM usuarios WHERE nombre = ?");
pstmt.setString(1, nombreUsuario);
ResultSet rset = pstmt.executeQuery();
```

La consulta se construye utilizando el carácter de interrogación ? para definir la posición de cada uno de los parámetros, el valor de cada parámetro será pasado más adelante a través de un bloque de código. De esta forma la consulta primero es *precompilada* para más adelante recibir los valores de los parámetros pero sin que la consulta sea modificada.

Sin embargo una implementación incorrecta de las consultas parametrizadas puede llevar al traste la fortificación del aplicativo con un riesgo adicional asociado: la falsa sensación de seguridad.

La *Fundación OWASP* (*Open Web Application Security Project*) define una serie de herramientas, documentos y otros recursos cuyo objetivo es desarrollar, adquirir, operar y mantener aplicaciones más seguras en las que se pueda confiar. En el artículo *OWASP*

SQL Injection Prevention Cheat Sheet [1] se describen detalladamente todas las técnicas de prevención.

2.2.7. Inyección XSS

La *Inyección XSS* consiste en inyectar código *HTML* o *JavaScript* en una aplicación web, con el objetivo de que el navegador de un usuario ejecute el código inyectado cuando ve la página alterada.

Se utiliza para provocar una acción inadecuada en el navegador de un usuario y no en el servidor. Si alguien inyecta código HTML en alguna aplicación web no podrá afectar a algún servidor, ya que este nunca interpreta el código HTML, sólo son capaces de hacerlo los navegadores. Por ello, este ataque se conoce como "ataque del lado del cliente".

Figura 11 – Esquema *Cross Site Scripting* (*XSS*)

El *XSS* se trata de una vulnerabilidad que muchos desarrolladores web dejan pasar, bien por falta de planificación o por desconocimiento. Esta vulnerabilidad suele aparecer por la falta de mecanismos en el filtrado de los campos de entrada que dispone la web, permitiendo el envío de datos e incluso la ejecución de scripts completos.

El código malicioso utilizado en este tipo de ataques está compuesto por cadenas de datos: scripts completos contenidos en enlaces o ejecutados desde formularios vulnerables.

Existen tres tipos de vulnerabilidades:

- El tipo -0, la cual se utiliza para ejecutar código con los permisos de otro usuario. Esta vulnerabilidad es conocida como "*DOM-Basedcross-site.scripting*" o "*cross-site-scripting local*". El código se inyecta a través de la *URL* pero no se incluye en el código de la fuente de la página, es decir, el código únicamente se inyecta en el navegador.
- El tipo -1, donde el servidor procesa la inyección enviada por el usuario y la incluye en el código de la página.
- El tipo -2, se conoce como ataque-persistente, donde se proyecta código en páginas estáticas. Esta vulnerabilidad permite hacer ataques más peligrosos y poderosos a las aplicaciones web. La información que proporciona el usuario es almacenada en la base de datos, en el sistema de archivos o en otro lugar. Esto es lo que hace fuerte al tipo -2, porque la inyección se quedará siempre en alguna parte de la web.

[1] OWASP Injection Prevention Cheat Sheet: *https://www.owasp.org/index.php/Inyección_SQL*

Los ataques *XSS* no se limitan al robo de *cookies*, el ataque puede variar dependiendo de cada situación.

A la hora de lanzar un ataque de este tipo, los atacantes pueden utilizar varios tipos de inyección de código distinto. Veamos a continuación cuáles son los más utilizados:

- ***Inyección a un formulario***: el ataque consiste en inyectar código en un formulario que después de enviar la información al servidor, será incluido en el código fuente de alguna página, ya sea persistente o no persistente. Después el objetivo es lograr que la página sea vista por el usuario y ésta esté infectada.
- ***Inyección por medio de elementos***: Un elemento podrá ser cualquier valor establecido por el usuario o por la aplicación que viaja entre el navegador y la aplicación.
- ***Inyección por medio de recursos***: Aparte de los elementos en la *URL* y los formularios, hay otras formas en la que se puede actuar, como son las cabeceras *HTTP*. Estas cabeceras son mensajes con los que se comunican el navegador y el servidor. Aquí entran en juego las cookies, las sesiones, campo *referer, o los campos de IP* que se envían en las cabeceras de una petición, etc.

Además existen otros tipos de inyecciones que consisten en inyectar código en una aplicación Flash o en un Video.

Otros ataques XSS muy conocidos y aplicados son los siguientes:

- ***Ataque CredentialTheft o de Robo de credencial***: este ataque es el más conocido y aplicado en cuanto a vulnerabilidades XSS. Consiste en robar los métodos de autentificación de un usuario de una página para poder acceder a su cuenta sin la necesidad de saber su contraseña.
- ***Ataque Phising***: consiste en crear una misma página y hacer creer al usuario que está navegando en la página original. Una vez que el usuario se encuentra dentro de esta página se le pide información de su cuenta como nombres de usuario o contraseñas. Esta información es almacenada para que el atacante tenga o posea toda la información.

Las aplicaciones profesionales tienen filtros que validan todos los datos que introduce el usuario para evitar inyecciones. Algunos de estos filtros son los siguientes:

1. Limitar los caracteres de entrada: Limitar el tamaño de los campos de texto de los formularios utilizando la variable *maxlength* que proporciona *HTML*.
2. Sanear los datos: Seleccionar únicamente la información que interesa, eliminando las etiquetas *HTML* que pueden introducirse en una caja de texto.
3. Escapar los datos: Preparar los caracteres que deben ser representados por *HTML* para preservar su significado, evitando que el navegador evalúe el código.
4. Filtro programado en *JavaScript* donde el usuario envía la información y el *script* recibe los datos y los valida, después de ser filtrados y validados son enviados a alguna aplicación que recibe los datos y hace lo que tenga que hacer con ellos.

A continuación se incluyen un par de ejemplos de inyección *XSS*:

(1) En muchos textos de XSS aparece algo como: “Ve a algún formulario de búsqueda de una web y pon: <script>alert(); (sin el espacio), si sale una ventana es vulnerable.

(2) Cogiendo como ejemplo: *http://www.pan.senado.gob.mx*, podemos ver en la parte izquierda superior, debajo del logo del pan un pequeño campo de texto que dice

"*búsqueda*". Este es el campo vulnerable a inyección *HTML*. A continuación, se pone en el campo de búsqueda "rastan", y le damos clic en buscar, una vez cargada la página podemos observar el texto: no se encontraron resultados de buscar rastan. Por tanto, la página es vulnerable a XSS.

2.2.8. Protección

Para poder tener una visión general de los aspectos que engloba la seguridad informática, además de conocer los elementos a proteger, los tipos de amenazas y el origen de dichas amenazas, es importante hablar de las formas de protección de los sistemas.

Para proteger un sistema de información ha de realizarse un análisis mediante una auditoría de seguridad de las amenazas potenciales a las que está expuesto, así como su probabilidad de ocurrencia e impacto.

Una auditoría de seguridad comprende el análisis y gestión de sistemas para identificar y corregir las diversas vulnerabilidades que pudieran presentarse en la revisión de los equipos de trabajo, redes de comunicaciones y servidores. Los objetivos principales de una auditoría son la revisión de la seguridad en los entornos y sistemas para verificar el cumplimiento de la normativa y legislación vigente.

La auditoría debe realizarse en base a un patrón o un conjunto de buenas prácticas sugeridas, por ejemplo los estándares internacionales de mejores prácticas para la seguridad de la información tales como *COBIT* (*Objetivos de Control de las Tecnologías de la Información*) o *ISO 27002,* por ejemplo.

Las auditorías realizadas con cierta frecuencia aseguran la integridad de los controles de seguridad aplicados, estos servicios de auditoría pueden ser de distinta índole:

- ***Auditoría de seguridad interna***: se contrasta el nivel de seguridad de redes locales y corporativas de carácter interno.
- ***Auditoría de seguridad perimetral***: se estudia el perímetro de la red local o corporativa conectado a redes públicas.
- ***Test de intrusión***: se intenta acceder a los sistemas, para comprobar el nivel de resistencia a la intrusión no deseada.
- ***Análisis forense***: análisis posterior de incidentes, mediante el cual se trata de reconstruir el modo de acceso y valorar los daños ocasionados.
- ***Auditoría de código de aplicaciones***: análisis del código independientemente del lenguaje utilizado. Un ejemplo concreto se realiza en los sitios web comprobando vulnerabilidades como la *inyección SQL*, *Cross Site Scripting* (*XSS*), etc.

2.2.9. Java

Se puede definir Java a un muy alto nivel de abstracción como un lenguaje de programación orientado a objetos que fue desarrollado por *Sun Microsystems* a principios de los años 90.

Implementa los conceptos fundamentales del paradigma de Programación Orientado a Objetos: *clases, herencia, asociación, clases abstractas, polimorfismo, encapsulación, ocultamiento de información*.

El lenguaje en sí mismo toma su sintaxis de *C* y *C++* pero quitando características que contribuían a generar errores, siendo diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.

Permite el desarrollo de aplicaciones en red, distribuidas y concurrentes, independientes de la plataforma. Las primeras implementaciones tenían el lema "*write once, run anywhere*" y es que se trata de un lenguaje semi-interpretado, ya que el código fuente del programa se compila una única vez generando el *bytecode* asociado y éste es interpretado en tiempo de ejecución por la máquina virtual.

Nombre	**JAVA**
Desarrolladores	James Gosling & Sun Microsystems
Extensiones comunes	Java, class, jar
Paradigma	objetos, imperativo
Apareció en	1995
Diseñado por	Sun Microsystems (Oracle Corporation)
Última versión estable	Java Standard Edition 8 (18 de marzo de 2014)
Tipo de dato	Fuerte, Estático
Implementaciones	OpenJDK, HotSpot, etc.
Dialectos	Generic Java, Pizza
Influido por	Objective-C, C++, Smalltalk, Eiffel
Ha influido a	C#, D, J#, JavaScript, PHP, Python
Sistema operativo	Multiplataforma

Tabla 5 – Java: Información General

Es decir, Java es algo más que un lenguaje, ya que la palabra Java se refiere a dos cosas inseparables: el lenguaje que nos sirve para crear programas (*API de Java*) y la Máquina Virtual Java que sirve para ejecutarlos.

La API de Java está compuesta de una amplia colección de componentes de software que proveen un conjunto de funciones útiles:

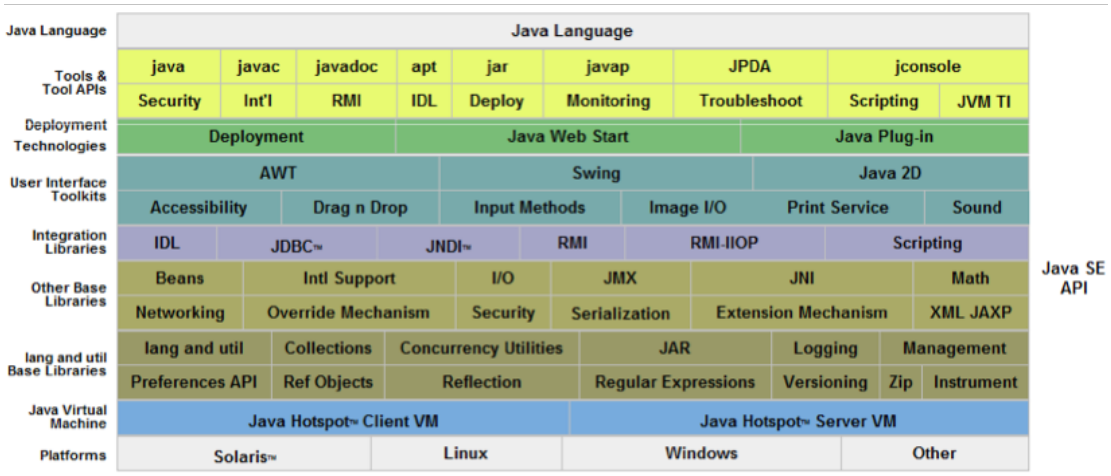


Figura 12 – Componentes API Java

Java ha experimentado numerosos cambios desde la su primera versión *JDK 1.0*, y actualmente tiene un gran número de clases y paquetes que componen la biblioteca estándar:

- **JDK 1.0** (23 de enero de 1996) — Primer lanzamiento: comunicado de prensa
- **JDK 1.1** (19 de febrero de 1997) — Principales adiciones incluidas:
  - una reestructuración intensiva del modelo de eventos *AWT* (*AbstractWindowingToolkit*)
  - clases internas (*innerclasses*)
  - *JavaBeans*
  - *JDBC* (*Java DatabaseConnectivity*), para la integración de bases de datos
  - *RMI (Remote Method Invocation*)
- **J2SE 1.2** (8 de diciembre de 1998) — Nombre clave *Playground*. Esta y las siguientes versiones fueron recogidas bajo la denominación *Java 2* y el nombre "*J2SE*" (*Java 2 Platform, Standard Edition*), reemplazó a *JDK* para distinguir la plataforma base de *J2EE* (*Java 2 Platform, Enterprise Edition*) y *J2ME* (*Java 2 Platform, Micro Edition*).
  - reflexión en la programación
  - la API gráfica (*Swing*) fue integrada en las clases básicas
  - la máquina virtual (*JVM*) fue equipada con un compilador *JIT* (Just in Time) por primera vez
  - Java *Plug-in*
  - Java *IDL*, una implementación de IDL (Lenguaje de Descripción de Interfaz) para la interoperabilidad con *CORBA*
  - Colecciones (*Collections*)
- **J2SE 1.3** (8 de mayo de 2000) — Nombre clave *Kestrel*. Los cambios más notables fueron:
  - la inclusión de la máquina virtual de *HotSpot JVM*
  - *RMI* fue cambiado para que se basara en *CORBA*
  - *JavaSound*
  - se incluyó *JNDI (Java Naming and Directory Interface)* en el paquete de bibliotecas principales (anteriormente disponible como una extensión)
  - *Java Platform Debugger Architecture (JPDA)*
- **J2SE 1.4** (6 de febrero de 2002) — Nombre Clave *Merlin*. Este fue el primer lanzamiento de la plataforma Java desarrollado bajo el Proceso de la Comunidad Java como JSR 59.
  - Palabra reservada *assert* (Especificado en JSR 41.)
  - Expresiones regulares modeladas al estilo de las expresiones regulares *Perl*
  - Encadenamiento de excepciones Permite a una excepción encapsular la excepción de bajo nivel original.
  - *non-blocking NIO* (New Input/Output)
  - *Logging API* (
  - *API I/O* para la lectura y escritura de imágenes en formatos como *JPEG* o *PNG*
  - *Parser XML* integrado y procesador *XSLT* (*JAXP*)
  - Seguridad integrada y extensiones criptográficas (*JCE, JSSE, JAAS*)
  - Java Web Start
- **J2SE 5.0** (30 de septiembre de 2004) — Nombre clave: *Tiger* (Originalmente numerado **1.5**, esta notación aún es usada internamente), añadió un número significativo de nuevas características

- Plantillas (genéricos) — provee conversión de tipos (*type safety*) en tiempo de compilación para colecciones y elimina la necesidad de la mayoría de conversión de tipos (*type casting*)
- Metadatos — también llamados anotaciones, permite a estructuras del lenguaje como las clases o los métodos, ser etiquetados con datos adicionales, que puedan ser procesados posteriormente por utilidades de proceso de metadatos
- *Autoboxing/unboxing* — Conversiones automáticas entre tipos primitivos (Como los int) y clases de envoltura primitivas (Como *Integer*). (Especificado por JSR 201.)
- Enumeraciones — la palabra reservada *enum* crea una typesafe, lista ordenada de valores (como *Día. LUNES, Día. MARTES*, *etc*.). Anteriormente, esto sólo podía ser llevado a cabo por constantes enteras o clases construidas manualmente (*enumpattern*)
- *Varargs* (número de argumentos variable) — El último parámetro de un método puede ser declarado con el nombre del tipo seguido por tres puntos (e.g. voiddrawtext(String... lines)). En la llamada al método, puede usarse cualquier número de parámetros de ese tipo, que serán almacenados en un *array* para pasarlos al método.
- Bucle *for* mejorado — La sintaxis para el bucle for se ha extendido con una sintaxis especial para iterar sobre cada miembro de un array o sobre cualquier clase que implemente *Iterable*, como la clase estándar *Collection*

- **Java SE 6** (11 de diciembre de 2006) — Nombre clave *Mustang*. Estuvo en desarrollo bajo la JSR 270. En esta versión, Sun cambió el nombre "J2SE" por Java SE y eliminó el ".0" del número de versión. Los cambios más importantes introducidos en esta versión son:
  - Incluye un nuevo marco de trabajo y APIs que hacen posible la combinación de Java con lenguajes dinámicos como *PHP, Python, Ruby y JavaScript*
  - Incluye el motor *Rhino*, de *Mozilla*, una implementación de *JavaScript* en Java
  - Incluye un cliente completo de Servicios Web y soporta las últimas especificaciones, como *JAX-WS 2.0, JAXB 2.0, STAX y JAXP*
  - Mejoras en la interfaz gráfica y en el rendimiento
- **Java SE 7** — Nombre clave *Dolphin*. En el año 2006 aún se encontraba en las primeras etapas de planificación. Su lanzamiento fue en julio de 2011
  - Soporte para *XML* dentro del propio lenguaje
  - Un nuevo concepto de *superpaquete*
  - Soporte para *closures*
  - Introducción de anotaciones estándar para detectar fallos en el software
  - Posibilidad de operar con clases *BigDecimal* usando operandos
- **Java SE 8** — lanzada en marzo de 2014. Cabe destacar:
  - Incorpora de forma completa la librería *JavaFX*
  - Diferentes mejoras en seguridad
  - Diferentes mejoras en concurrencia
  - Añade funcionalidad para programación funcional mediante expresiones *Lambda*
  - Mejora la integración de *JavaScript*
  - Nuevas API para manejo de fechas y tiempo (date - time)

2.2.9.1. *Máquina Virtual Java*

Java Virtual Machine (*JVM*) es una aplicación que interpreta y ejecuta programas escritos en el lenguaje de programación Java. Específicamente puede interpretar el *bytecode* generado al compilar en Java.

Con el compilador se convierte el código fuente que reside en archivos cuya extensión es *.java*, a un conjunto de instrucciones que recibe el nombre de *bytecodes* (archivo cuya extensión es *.class*).

La *JVM* es una de las piezas fundamentales de la plataforma Java. Básicamente se sitúa en un nivel superior al hardware del sistema sobre el que se pretende ejecutar la aplicación, y este actúa como un puente que entiende tanto el *bytecode* como el sistema sobre el que se pretende ejecutar. Así, cuando se escribe una aplicación Java, se hace pensando que será ejecutada en una máquina virtual en concreto, siendo ésta la que en última instancia convierte de código *bytecode* a código nativo del dispositivo final.

El intérprete de *JVM* ejecuta cada una de estas instrucciones en un ordenador específico (*Windows*, *Mac*, etc.). Solamente es necesario, por tanto, compilar una vez el programa, pero se interpreta cada vez que se ejecuta en un dispositivo.

Las tareas principales de la JVM son las siguientes:

- Reservar espacio en memoria para los objetos creados
- Liberar la memoria no usada (garbage collection)
- Asignar variables a registros y pilas
- Llamar al sistema huésped para ciertas funciones, como los accesos a los dispositivos
- Vigilar el cumplimiento de las normas de seguridad de las aplicaciones Java

Existen compiladores *just-in-time* que traducen el *bytecode* de Java a código máquina justo antes de la ejecución, con vistas a aumentar la velocidad.

La gran ventaja de la máquina virtual java es aportar portabilidad al lenguaje, de manera que se han creado diferentes máquinas virtuales java para diferentes arquitecturas. Para poder cambiar la arquitectura del sistema en el que se ejecuta una aplicación tan sólo es necesario disponer de dicha máquina virtual para dicho entorno.

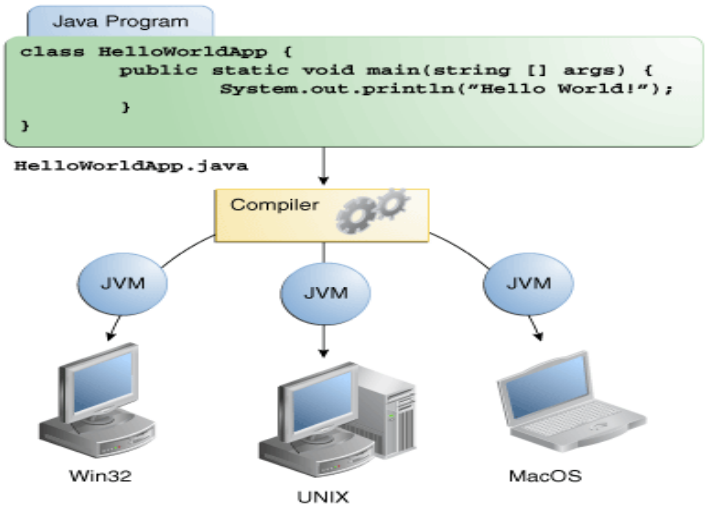


Figura 13 – Máquina virtual Java (*JVM*)

2.2.10. Seguridad en Java

Desde su creación Java ha tenido presentes los problemas de seguridad y ha definido un modelo para controlar y limitar el acceso a los recursos desde los programas y aplicaciones. El modelo de seguridad ha ido evolucionando con las distintas versiones del *Entorno de Desarrollo Java* (*JDK*), pasando de un modelo muy sencillo y restrictivo, el del *JDK 1.0*, a uno más complejo y flexible desde la aparición del *JDK 1.2*.

Java fue diseñado para ofrecer las siguientes medidas de seguridad básicas:

- **Uso de un lenguaje de programación diseñado para ser seguro**.
- **Integración de un sistema de control de permisos para los programas**. Java define un mecanismo (SANDBOX-*mecanismo del cajón de arena*) que permite controlar qué se le permite hacer a un programa y controlar cómo accede a los recursos.
- **Encriptación y uso de certificados**. Se definen mecanismos para que los programadores puedan firmar el código, de manera que los usuarios puedan verificar quién es el propietario del código y que éste no ha sido modificado después de ser firmado.

La seguridad se basa en tres componentes fundamentales del entorno de ejecución:

1) **El cargador de clases** (*ClassLoader*), que determina cómo y cuándo pueden cargar código los programas y garantiza que los componentes del sistema no han sido reemplazados.
2) **El verificador de archivos de clases** (*Class file verifier*), que garantiza que el código tiene el formato correcto, que el *bytecode* no viola las restricciones de seguridad de tipos de la JVM. Que las pilas internas no puedan desbordarse, ni por arriba, ni por abajo y que las instrucciones en *bytecode* tengan parámetros de tipos correctos.
3) **El gestor de seguridad** (*Security Manager*) controla el acceso a los recursos en tiempo de ejecución. Los recursos sobre los que tiene control son múltiples: E/S de red y ficheros, creación de cargadores de clases, manipulación de hilos de ejecución, ejecución de programas externos (del *SO*), detener la *JVM*, cargar código nativo en la máquina virtual, realizar determinadas operaciones en el entorno de ventanas o cargar ciertos tipos de clases.

2.2.10.1. Seguridad en el JDK 1.0

El modelo de seguridad original de la plataforma Java, denominado como *modelo del cajón de arena* (*sandbox model*), proporcionaba un entorno muy restringido en el que ejecutar código no fiable obtenido de la red.

Tiene dos niveles de acceso a los recursos: total, para programas locales, o muy restringido para programas remotos.

La seguridad se consigue gracias al empleo del cargador de clases, el verificador de clases y el gestor de seguridad.

Es un modelo que es demasiado restrictivo, no permite que los programas remotos hagan nada útil, por estar restringidos al *sandbox*.

2.2.10.2.Seguridad en el JDK 1.1

Se introdujo el nuevo concepto de *código remoto firmado*, que sigue garantizando la seguridad de los clientes, pero permite que código obtenido remotamente salga del cajón y tenga acceso a todos los recursos, siempre y cuando esté firmado por una entidad en la que el cliente confía.

Aunque esto mejora un poco la situación, sigue siendo un control de dos niveles: total para código local o remoto firmado y restringido para código remoto sin firma o con firmas no validadas por el cliente.

Además del código firmado, el JDK 1.1 introdujo otras mejoras de seguridad:

1. Un par de herramientas de seguridad.
   - *Jar:* no es más que un programa archivador con un formato compatible con el *zip*, que permite reunir un conjunto de clases y otros ficheros (como por ejemplo imágenes o texto) en un solo archivo, almacenado normalmente con la extensión.jar. Esto hace posible la transferencia de aplicaciones de un modo compacto, en una sola conexión, y el firmado de los programas de modo conjunto.
   - *Javakey* es el que permite el firmado de clases en los ficheros *jar*.
2. Un API para programación segura.
   El API de seguridad introdujo paquetes de clases que proporcionan funciones criptográficas a los programadores, permitiendo el desarrollo de aplicaciones que usen estas técnicas de modo estándar. El API estaba diseñada para ser extensible e incluía herramientas para: generar firmas digitales, resúmenes de mensajes, gestión de claves y el uso de listas de control de accesos (ACLs).

   Los algoritmos criptográficos se proporcionaban separados en la Extensión Criptográfica de Java (JCE), como un paquete adicional al JDK estándar, principalmente por los problemas de exportación de los EEUU.

2.2.10.3.Seguridad en Java 2

En el JDK 1.2 se introdujeron nuevas características que mejoran el soporte y el control de la seguridad:

- **Control de acceso de grano fino**. Uno de los problemas fundamentales de la seguridad en el JDK 1.1 es que el modelo sólo contempla dos niveles de permisos: acceso total o cajón de arena. Para solventar el problema el JDK 1.2 introduce un sistema de control de permisos de *grano fino* que permite dar permisos específicos a trozos de código específicos para acceder a recursos específicos en el cliente, dependiendo de la firma del código y/o el URL del que este se obtuvo.
- **Control de acceso aplicado a todo el código**. El concepto de *código firmado* es ahora aplicable a todo el código, independientemente de su procedencia (local o remoto).
- **Facilidad de configuración de políticas de seguridad**. La nueva arquitectura de seguridad permite el ajuste sencillo de los permisos de acceso usando un *fichero de políticas* (*policy file*) en el que se definen los permisos para acceder a los recursos del sistema para todo el código (local o remoto, firmado o sin firmar). Gracias a ello, el usuario puede bajar aplicaciones de la red, instalarlas y ejecutarlas asignándoles sólo los permisos que necesiten.

- **Estructura de control de acceso extensible**. En versiones anteriores del JDK cuando se deseaba crear un nuevo tipo de permiso de acceso era necesario añadir un nuevo método check a la clase del gestor de seguridad. Esta versión permite definir permisos tipo que representan un acceso a recursos del sistema y el control automático de todos los permisos de un tipo correcto, lo que repercute en que, en la mayoría de casos, se hace innecesario añadir métodos al gestor de seguridad.

Al igual que sucedió con el JDK 1.1 en el Java 2 han aparecido, o se han mejorado, las herramientas y el API de seguridad.

En Java 2 se incluyen cuatro herramientas de seguridad:

1. *jar*. Esta es básicamente la misma herramienta que apareció en el JDK 1.1 y se emplea para generar ficheros JAR.
2. *keytool*. Esta es una herramienta para la generación de pares de claves (públicas y privadas), importar y exportar certificados X.509 (versiones 1, 2 y 3), generar certificados X.509 V1 autofirmados y gestionar *keystores*.
3. *jarsigner*. Este programa se emplea para firmar ficheros JAR y verificar las firmas de ficheros JAR ya firmados. La herramienta emplea los almacenes de claves para buscar los datos que necesita para firmar ficheros (una clave privada), verificar firmas (clave pública) o verificar claves públicas (certificados). Esta herramienta junto a la anterior reemplazan al antiguo *javakey*.
4. *policytool*. Esta utilidad se emplea para crear y modificar los ficheros de configuración de políticas de seguridad del cliente de modo sencillo.

   El API de seguridad se ha mejorado con la incorporación de nuevos subpaquetes que para el soporte de certificados X.509 y permiten crear *Listas de Revocación de Certificados* (CRLs) y *Peticiones de Firmado de Certificados* (CSRs).

2.2.10.4. Comparación de los Modelos de Seguridad

Característica	JDK 1.0	JDK 1.1	Java 2 SDK
Acceso a los recursos del código local sin firma	Sin restricciones	Sin restricciones	Basado en política
Acceso a los recursos del código local firmado	No disponible	Sin restricciones si fiable o restringido por el *sandbox*	Basado en política
Acceso a los recursos del código remoto sin firma	Restringido por el *sandbox*	Restringido por el *sandbox*	Basado en política
Acceso a los recursos del código remoto firmado	No disponible	Sin restricciones si fiable o restringido por el *sandbox*	Basado en política
Servicios de firmado digital de código	No disponibles	*JCA (DSA)*	*JCA (DSA)*
Servicios criptográficos	No disponibles	*JCE 1.1*	*JCE 1.2*

Tabla 6 – Comparación modelos seguridad *JDK Java*

A continuación se detalla de forma breve las características que hacen de Java un lenguaje de programación seguro:

- **Un entorno de ejecución proporcionado por la *Máquina Virtual Java* (JVM)**. Entorno ni interpretado ni compilado; es un sistema híbrido. Los programas se compilan en un formato independiente de la máquina denominado *bytecode* y son interpretados por la Máquina Virtual, que es el único componente de Java que depende de la plataforma en la que trabajamos. La implementación de la *JVM* es la responsable de la seguridad incorporada en Java, por lo que es importante que su implementación sea adecuada.
- **Un conjunto de interfaces y arquitecturas** proporcionadas por la librería estándar de *Java* (*API de Java*). El *API de Java* es una colección de componentes software, agrupados en bibliotecas o *paquetes* de componentes relacionados, que proporcionan multitud de capacidades útiles para el diseño de interfaces gráficos, acceso a redes, gestión de E/S, etc. En lo que respecta a la seguridad el API de Java nos proporciona paquetes que implementan o dan acceso a herramientas de seguridad de alto y bajo nivel como algoritmos de encriptación, firmas electrónicas, gestión de certificados, etc.

2.2.11. Vulnerabilidades críticas entorno Java

Comúnmente se dice que Java es un lenguaje de programación seguro, fue diseñado para solucionar ciertas vulnerabilidades relacionadas con el acceso a memoria presentes en otros lenguajes como C++.

Igualmente, como se ha mencionado en el apartado anterior hay ciertas características específicas del lenguaje que contribuyen de manera extraordinaria a este objetivo, como pueden ser:

- Las referencias a *arrays* verificadas en el momento de la ejecución del programa.
- No hay manera de manipular de forma directa los punteros.
- La *JVM* gestiona automáticamente el uso de la memoria, de modo que no queden huecos.
- No se permiten realizar ciertas conversiones (*casting*) entre distintos tipos de datos.

No obstante, tal y como se ha especificado también anteriormente, cualquier sistema de información conlleva exposición a un riesgo.

A lo largo de su historia, la plataforma se ha visto expuesta a numerosas vulnerabilidades de seguridad que el fabricante ha ido solucionando periódicamente, a continuación se incluyen algunas de las que mayor impacto o repercusión han tenido en las últimas versiones estos años:

- **Vulnerabilidad crítica en Java 7 que permite ataques Zero Day**

  Descubierta en agosto de 2012 una vulnerabilidad crítica que afecta a la versión 7 de Java (update 1-6) y que permite a un atacante infectar el ordenador atacado.

  La vulnerabilidad ha estado siendo utilizada en ataques dirigidos con un *exploit* 0-day que instala un *dropper* (programa que se ha diseñado para "instalar" algún tipo de *malware* a un sistema destino), alterado para ejecutar el virus sin ser detectado por las soluciones de seguridad.

  Se clasificó como crítica y el *exploit* funciona en todas las versiones de *Internet Explorer, Firefox y Opera en Windows 7, Vista y XP*.

El ataque *zero day* es uno de los más peligrosos que existen ya que aprovecha vulnerabilidades aún desconocidas en software.

Como solución de contingencia se recomendó deshabilitar Java hasta que existiera una actualización para este fallo de seguridad.

- **Vulnerabilidad critica en Oracle Java hasta 6 Update 45 y 7 Update 21**

  Una función desconocida del componente *Networking* es afectada por esta vulnerabilidad. Mediante la manipulación de un *input* desconocido se causa una vulnerabilidad relacionada a escalamiento de privilegios.

  Esta vulnerabilidad fue solucionada como parte de las 40 actualizaciones de *Oracle Java SE Critical Patch Update Advisory* en Junio 2013, y fue clasificada por CVE (*CVE-2013-2451*). No se conocen los detalles técnicos.

  Permite a usuarios locales afectar a la confidencialidad, integridad y disponibilidad a través de vectores desconocidos relacionados con *Networking*, está relacionada con la aplicación indebida de puerto exclusivo *une* cuando se ejecuta en *Windows*, que permite a los atacantes unirse a los puertos que ya están en uso.

2.2.12. Servidor Web Apache-Tomcat

Apache-Tomcat, también conocido como simplemente *Tomcat* o *Jakarta Tomcat*, es un servidor web mutliplataforma que funciona como contenedor web actuando como motor de *Servlets* y JSP, y que se desarrolla bajo el proyecto denominado *Jakarta* perteneciente a la *Apache Software Foundation* bajo la licencia Apache 2.0 y que implementa las especificaciones de los *Servlets* y de *JavaServer Pages* (*JSP*) de Sun Microsystem.

Puesto que el producto fue desarrollado en Java, éste puede ejecutarse sobre cualquier sistema operativo, previa instalación de la máquina virtual de Java.

Puede funcionar como servidor web por sí mismo pese a que en sus inicios se pensaba que dicho servidor era recomendable usarlo en entornos de desarrollo con requisitos mínimos de velocidad. En la actualidad no existe esta percepción y por esto, es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Las versiones estables más recientes son las 6.0.30 y la 7.0.6 que implementan las especificaciones de *Servlet 2.5* y *JSP 2.1*.

A partir de la versión 4.x Tomcat fue lanzado con el contenedor de *Servlets* "*Catalina*", con el contenedor *HTTP* "*Coyote*" y un motor para *JSP* llamado "*Jasper*". Las principales características de estos tres componentes son:

- ***Catalina***

  Dicho componente implementa las especificaciones de *Servlets* y *JSP*. Para Apache Tomcat el elemento principal es una base de datos de nombres de usuarios, *password* y *roles* a estos usuarios permitiendo a Catalina integrarse donde ya existe información de autenticación como describe la especificación de *Servlets*.

- ***Coyote***

  Componente conector que admite el protocolo *HTTP 1.1* para el servidor web y que escucha en un puerto *TCP* especificado por el servidor y envía la solicitud al motor *Tomcat* para que éste procese la solicitud y envíe una respuesta al cliente.

- ***Jasper***

  Jasper analiza archivos *JSP* para compilar el código Java y, si se producen cambios, éste los vuelve a compilar. Desde la versión 5.x se usa Jasper 2 que es *JavaServer* para webs 2.0. Jasper 2 ha introducido las siguientes novedades:

  - *Recompilación* al producirse un cambio.
  - Incorpora el compilador JDT de Java.
  - Puesta en común de etiquetas JSP.

2.2.13. SQL

SQL (Structured Query Language) es un lenguaje de programación de 4ª generación vinculado con la gestión de bases de datos relacionales permitiendo la especificación de distintas clases de operaciones entre éstas.

El científico Edgar Frank Codd propuso un modelo relacional para las bases de datos y creó un sublenguaje para acceder a los datos a partir del cálculo de predicados conocido como SEQUEL, antecesor de SQL.

En esencia SQL es un lenguaje declarativo de alto nivel, basado en álgebra y cálculo relacional, orientado al manejo de conjuntos de registros (y no a registros individuales) que permite efectuar consultas con el fin de recuperar de forma sencilla información de la base de datos, así como realizar cambios en ella.

Por medio de las sentencias declarativas se indica qué se quiere hacer, siendo el sistema gestor quien interpreta la sentencia y decide cual es el mecanismo óptimo para llevarlo a cabo.

Una base de datos implica la coexistencia de múltiples tipos de lenguajes:

- El denominado *Data Definition Language* (también conocido como *DDL*) es aquél que permite modificar la estructura de los objetos contemplados por la base de datos por medio de cuatro operaciones básicas.
- SQL, por su parte, es un lenguaje que permite manipular datos (*Data Manipulation Language o DML*) que contribuye a la gestión de las bases de datos a través de consultas.

2.2.14. Servidor de Bases de Datos MySQL

MySQL es un sistema gestor de bases de datos relacionales, multi-hilo y multi-usuario. Desarrollado por *Sun MicroSystems Inc.* como software libre y distribuido por *Oracle Corporation* en un esquema de licenciamiento dual, es decir, por un lado se ofrece bajo la *GNU GPL* para cualquier uso compatible con esta licencia, pero, para aquellas empresas que quieran incorporarlo en productos privativos deben adquirir una licencia específica que les permita este uso.

Al contrario de proyectos como *Apache*, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, *MySQL* es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios.

MySQL es usado por muchos sitios web grandes y populares, tales *como Wikipedia, Google, Facebook, Twitter, Flickr, Youtube,* etc.

Existen varias interfaces de programación de aplicaciones que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos *MySQL*, incluyendo *C, C++, PASCAL, Delphi, Eiffel, Java, Lisp, PHP, Ruby, BealBasic, Harbour, FreeBasic, Tcl*, etc.; cada uno de estos utiliza una interfaz de programación de aplicaciones específica.

Entre las principales aplicaciones de MySQL se encuentra su uso en aplicaciones Web multi plataforma (*Linux/Windows-Apache-MySQL-PHP/Perl/Python*), y por herramientas de seguimiento de errores como *Bugzilla*. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL.

MySQL es un sistema gestor de acceso muy rápido en la lectura cuando utiliza el motor no transaccional *MyISAM*, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación, en los cuales se recomienda el uso del motor *InnoDB*.

2.2.14.1. Conector JDBC

Java Database Connectivity (*JDBC*) son un conjunto de clases e interfaces Java (*API*) que permiten el acceso a bases de datos relacionales permitiendo la conexión desde programas de alto nivel con el sistema gestor de base de datos para la ejecución de sentencias en lenguaje SQL.

Un único programa escrito usando el *API JDBC* será capaz de enviar sentencias SQL a la base de datos apropiada, por medio de un mecanismo estándar e independiente de la plataforma para el acceso a la mayoría de bases de datos existentes.

Con una aplicación escrita en lenguaje de programación Java tampoco es necesario escribir diferentes aplicaciones para ejecutar en diferentes plataformas. La combinación de Java y *JDBC* permite al programador escribir una sola vez y ejecutarlo en cualquier entorno.

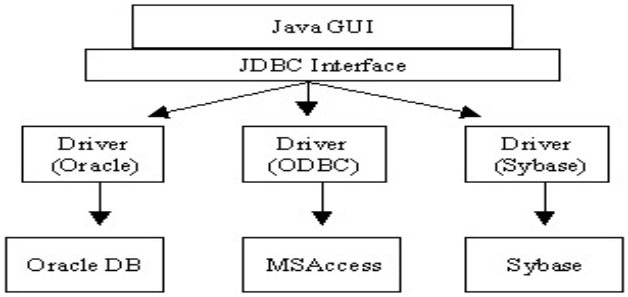


Figura 14 – Esquema Conector *JDBC*

Gracias al conector JDBC se pueden realizar de una manera muy sencilla estas 3 cosas:

- Establecer una conexión con la base de datos
- Enviar sentencias SQL
- Procesar los resultados

Cada fabricante de base de datos se encargará de proporcionar un driver *JDBC* específico para el tratamiento de las características propias de su base de datos (*triggers*, *secuencias*, tipos de datos, etc.).

Capítulo 3.

Descripción experimental

3.1. Visión General de la Aplicación

Como se ha indicado anteriormente, los requisitos básicos consisten en la creación de una aplicación web con varios módulos, a los que un usuario logado tendrá acceso en función del perfil que tenga configurado.

La aplicación debe disponer de un apartado de gestión de usuarios y otro para la gestión de noticias.

Para esto deberá disponer de varios perfiles que permitirán a los usuarios operar con las diferentes funcionalidades de la siguiente manera:

- **perfil Administrador**: debe tener acceso a todos los módulos de la aplicación, es decir, tanto a la gestión de usuarios, como a la gestión de noticias.
- **perfil Redactor**: debe tener acceso al módulo de noticias de la aplicación, para poder llevar a cabo la gestión de las noticias que sean publicadas.
- **perfil General**: debe tener acceso al módulo de noticias, únicamente en modo de consulta, es decir, tendrá acceso a la búsqueda de noticias.

En la tabla que se muestra a continuación se detallan las distintas configuraciones de cada perfil definido:

Nombre del perfil	Funcionalidad
Administrador general	Administrar toda la aplicación. Podrá realizar la gestión de usuarios (altas, modificación y baja, activación de usuarios, desactivación de usuarios) y Búsqueda de usuarios. Podrá realizar la gestión de noticias (altas, modificación y desactivación de noticias) y Búsqueda de noticias. Podrá realizar la modificación de niveles de seguridad de la aplicación
Redactor	Podrá realizar la gestión de noticias (altas, modificación y desactivación de noticias) y Búsqueda de noticias.
General	Usuario únicamente con rol de consulta. Podrá tener acceso a la Consulta de noticias.

Tabla 7 – Requisitos: Perfiles de acceso

3.2. Especificación de requisitos funcionales

En esta sección se realiza una descripción de alto nivel detallada de los servicios del sistema, dando una visión completa y consistente del comportamiento del mismo, describiendo además las interacciones que los usuarios tendrán con el software.

Cada requisito debe tener el nivel de información suficiente para poder realizar a partir de ellos el posterior diseño de cada una de las funcionalidades que deben ser soportadas. Deben ser trazables, priorizables y verificables.

A continuación se incluye el listado de requerimientos del sistema, identificados en un listado numerado para su identificación y seguimiento en apartados posteriores del presente documento.

3.2.1. RF01 – Identificación de Usuarios

Número de requisito	**RF01**
Nombre de requisito	Identificación de usuarios
Tipo	☒ Requisito ☐ Restricción
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/Opcional
Descripción	Para poder acceder al conjunto de funcionalidades del sistema, el usuario deberá realizar una identificación previa por medio de usuario y contraseña, de tal forma que el sistema pueda determinar automáticamente el perfil que tiene asociado y por lo tanto, aquellas opciones disponibles. El sistema validará si los datos introducidos en el formulario son correctos, permitiendo el acceso o en caso contrario se mostrará un mensaje de error.

Tabla 8 – Requisitos: RF01- Identificación de usuarios

3.2.2. RF02 – Registro de usuarios

Número de requisito	**RF02**
Nombre de requisito	Registro de usuarios
Tipo	☒ Requisito ☐ Restricción
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/Opcional
Descripción	Se debe dar la posibilidad a un nuevo usuario de registrarse para poder utilizar las funcionalidades que proporciona el sistema. Para ello deberá introducir su información personal así como un usuario y contraseña para su identificación en el acceso.

Tabla 9 – Requisitos: RF02 - Registro de usuarios

3.2.3. RF03 – Menú principal según perfil

Número de requisito	**RF03**
Nombre de requisito	Menú principal según perfil
Tipo	☒ Requisito ☐ Restricción
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/Opcional
Descripción	Cuando un usuario acceda al sistema después de *logarse* se le mostrará un menú en el que deben aparecer todas las opciones de menú a las que puede acceder en función del perfil que

	tiene asignado, no mostrándose aquellas para las que no tiene suficientes privilegios.

Tabla 10 – Requisitos: RF03 - Menú principal según perfil

3.2.4. RF04 - Búsqueda de usuarios

Número de requisito	**RF04**
Nombre de requisito	Búsqueda de usuarios
Tipo	☒ Requisito ☐ Restricción
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/Opcional
Descripción	Desde la pantalla principal cuando un usuario tenga perfil *Administrador* tendrá la posibilidad de acceder a un módulo en el que podrá realizar búsquedas de usuarios registrados en el sistema, pudiendo filtrar los resultados por diversos criterios, tales como el alias del usuario, o bien su nombre y/o apellidos y/o documento identificativo.

Tabla 11 – Requisitos: RF04 – Búsqueda de usuarios

3.2.5. RF05 - Listado de usuarios

Número de requisito	**RF05**
Nombre de requisito	Listado de usuarios
Tipo	☒ Requisito ☐ Restricción
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/Opcional
Descripción	Cuando el usuario introduzca uno o varios criterios para el filtrado se mostrará un listado con aquellos que estén activos en la aplicación y que sus datos coincidan con los datos introducidos en el formulario de búsqueda.

Tabla 12 – Requisitos: RF05 – Listado de usuarios

3.2.6. RF06 - Búsqueda de noticias

Número de requisito	**RF06**
Nombre de requisito	Búsqueda de noticias
Tipo	☒ Requisito ☐ Restricción
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/Opcional
Descripción	Desde la pantalla principal el usuario tendrá la posibilidad de acceder a un módulo en el que podrá realizar búsquedas de noticias, pudiendo filtrar los resultados por diversos criterios, tales como el título, entradilla y/o el texto de la noticia.

Tabla 13 – Requisitos: RF06 – Búsqueda de noticias

3.2.7. RF07 - Listado de noticias

Número de requisito	**RF07**
Nombre de requisito	Listado de noticias
Tipo	☒ Requisito ☐ Restricción
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/Opcional
Descripción	Cuando el usuario introduzca uno o varios criterios para filtrar la búsqueda, se mostrará un listado con aquellos registros cuyos datos coincidan con los criterios introducidos en el formulario de búsqueda.

Tabla 14 – Requisitos: RF07 – Listado de noticias

3.2.8. RF08 - Gestión de usuarios

Número de requisito	**RF08**
Nombre de requisito	Gestión de usuarios
Tipo	☒ Requisito ☐ Restricción
Prioridad del requisito	☐ Alta/Esencial ☒ Media/Deseado ☐ Baja/Opcional
Descripción	Los usuarios con perfil *Administrador* dispondrán en la pantalla principal, además de otras opciones, de una opción que les permitirá realizar la gestión de los usuarios del sistema, pudiendo llevar a cabo altas, modificación, baja o activación/desactivación de usuarios.

Tabla 15 – Requisitos: RF08 – Gestión de usuarios

3.2.9. RF09 - Gestión de noticias

Número de requisito	**RF09**
Nombre de requisito	Gestión de noticias
Tipo	☒ Requisito ☐ Restricción
Prioridad del requisito	☒ Alta/Esencial ☐ Media/Deseado ☐ Baja/Opcional
Descripción	Los usuarios con perfil *Administrador* o *Redactor* dispondrán en la pantalla principal, además de otras opciones, de una opción que les permitirá realizar la gestión de las noticias, pudiendo llevar a cabo el alta, modificación y desactivación de éstas.

Tabla 16 – Requisitos: RF09 – Gestión de noticias

3.2.10. RF10 - Subida de ficheros

Número de requisito	**RF10**
Nombre de requisito	Subida de ficheros
Tipo	☒ Requisito ☐ Restricción

Prioridad del requisito	☐ Alta/Esencial ☐ Media/Deseado ☒ Baja/Opcional
Descripción	Debe disponerse de la funcionalidad adicional de realizar la subida de ficheros a la base de datos.

Tabla 17 – Requisitos: RF10 – Subida de ficheros

3.2.11. RF11 - Modificar niveles de seguridad

Número de requisito	**RF11**
Nombre de requisito	Modificar niveles de seguridad
Tipo	☒ Requisito ☐ Restricción
Prioridad del requisito	☐ Alta/Esencial ☐ Media/Deseado ☒ Baja/Opcional
Descripción	Se debe dotar al sistema de un módulo para que los usuarios con perfil *Administrador* puedan realizar la gestión de los niveles de seguridad, para que según se vayan superando los diferentes tipos de requisitos de vulnerabilidad, se pueda determinar qué nivel de seguridad tiene asociado el usuario para que éste pueda seguir enfrentándose a nuevos retos.

Tabla 18 – Requisitos: RF11 - Modificar niveles de seguridad

3.3. Especificación de requisitos no funcionales

A continuación se incluye el listado de requisitos no funcionales del sistema, es decir, el conjunto de restricciones, aspectos técnicos que debe cumplir, características relevantes para la satisfacción (fiabilidad, rendimiento, calidad, etc.) e incluso normativa vigente aplicable.

CÓDIGO	DESCRIPCIÓN REQUISITO
RNF 01	La aplicación debe montarse de manera que sea posible el acoplamiento de nuevos módulos de manera sencilla, tanto en la parte de código como en la parte de base de datos
RNF 02	La base de datos a utilizar debe ser MYSQL
RNF 03	Utilización de método de conexión *JDBC* a base de datos MYSQL través de conexión sin *PreparedStatement*
RNF 04	Características de contraseñas de usuario: campo alfanumérico de 15 posiciones
RNF 05	No se realizará bloqueo de cuentas de usuario por numero de intentos
RNF 06	No será obligatorio cambio periódico de la contraseña de acceso al sistema
RNF 07	Los formularios de la aplicación deben utilizar método GET del protocolo HTTP
RNF 08	No se realizaran validaciones del contenido de los campos introducidos desde pantalla, únicamente se realizarán validaciones de obligatoriedad
RNF 09	No introducción de reglas de validación de contraseñas de los usuarios
RNF 10	Las tablas de datos no maestros deben de disponer de datos de auditoría que permitan control de cambios
RNF 11	La aplicación debe ser desarrollada en base a lo establecido por el patrón de diseño Modelo Vista Controlador
RNF 12	Registro de operativa en la aplicación a través del envío de trazas al log de todas las operaciones realizas a través de la aplicación

RNF 13	Debe disponer de Altas de usuarios sin logado inicial en la aplicación
RNF 14	Instalación inicial de la aplicación sobre servidor Apache-Tomcat
RNF 15	Desde el listado de usuarios se dará la posibilidad de realizar pruebas de inyección XSS

Tabla 19 – Requisitos No Funcionales

3.4. Diseño

3.4.1. Metodología de Desarrollo

El modelo del ciclo de vida del software en espiral fue definido por Bohem a finales de los 80, cuando promulgó un modelo desde un enfoque distinto al tradicional en *Cascada*. Tiene en cuenta el riesgo que aparece a la hora de desarrollar software, la idea fundamental es que la gestión de riesgos debe guiar el proceso de desarrollo.

La esencia del modelo es la división en cuadrantes que representan cada uno una actividad diferente:

- *Identificar objetivos, alternativas, restricciones*. El objetivo es la definición de la siguiente parte del sistema a desarrollar, es decir, especificación de los requisitos funcionales y no funcionales, se identifican las posibles opciones de diseño y sus restricciones.
- *Evaluar las alternativas mediante la identificación y resolución de riesgos.* Esta fase se centra en las áreas de incertidumbre. A partir de análisis de riesgos se derivará una estrategia (esta puede incluir prototipos, benchmarks, etc.)
- *Desarrollar y verificar el siguiente nivel del sistema.* Esta fase puede implicar bien actividades completas o bien un ciclo de ingeniería completo (diseño, construcción, pruebas e implantación).
- *Planificar las siguientes fases.* Aquí se incluyen la evaluación y planificación para el siguiente ciclo en espiral.

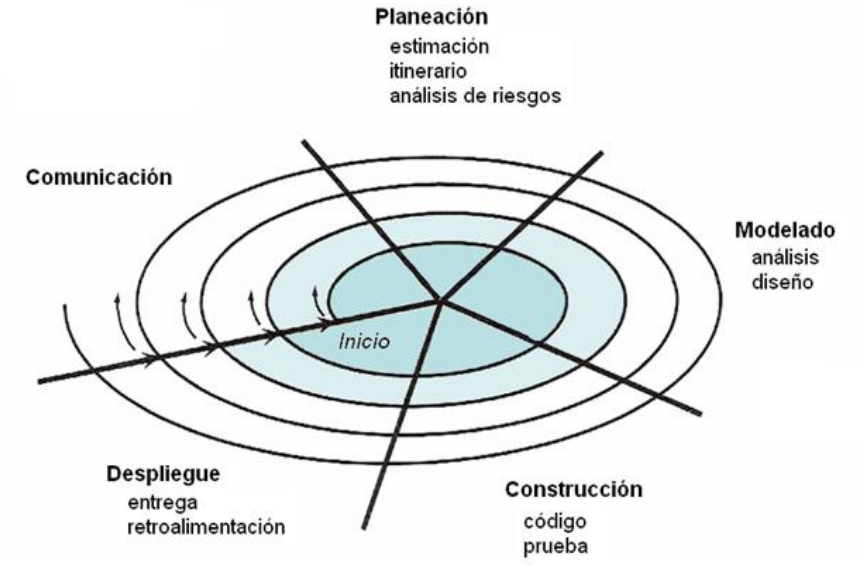


Figura 15 – Modelo de Ciclo de Vida en Espiral

El número de ciclos del modelo es específico para cada proyecto, siendo determinado por la complejidad, el volumen y los riesgos de cada uno.

Es considerado como un modelo evolutivo ya que combina el modelo clásico con el diseño de prototipos.

VENTAJAS DEL MODELO ESPIRAL

- No requiere una definición completa de los requerimientos del software a desarrollar para comenzar su funcionalidad.
- La aplicación se desarrolla de forma progresiva.
- Los riesgos y defectos son detectados a medida que el desarrollo evoluciona y no al final.
- En la terminación de un producto desde el final de la primera iteración es muy factible aprobar los requisitos.
- Se corre un riesgo menor de sufrir retrasos porque los conflictos pueden ser presentados antes y de este modo ser corregidos a tiempo.

DESVENTAJAS DEL MODELO ESPIRAL

- Existe complicación cuando se evalúan los riesgos.
- Se requiere la participación continua por parte del cliente.
- Se pierde tiempo al volver producir inicialmente una especificación completa de los requerimientos cuando se modifica o mejora el software.

La principal razón de la idoneidad de este modelo para el proyecto es que se comienza mirando las posibles alternativas de desarrollo, se opta por la de riesgo más asumible y se hace un ciclo de la espiral. En caso de querer seguir haciendo mejoras en el software, se vuelven a evaluar las distintas nuevas alternativas y riesgos y se realiza otra vuelta de la espiral, todo esto permite ir haciendo evoluciones y mejoras mientras se cumplan los plazos establecidos para la entrega final del producto para su validación.

3.4.2. Patrón de Diseño Modelo-Vista-Controlador

Los patrones de diseño de software son soluciones reutilizables a problemas que ocurren durante el desarrollo de un sistema software. Tratan de extraer la esencia del diseño para que pueda ser reutilizada en otros problemas parecidos que dicho diseño ya ha resuelto con anterioridad.

Los patrones son formas de describir mejores prácticas, buenos diseños, y captan la experiencia de tal manera que es posible que otros reutilicen esta experiencia. Proporcionan además una arquitectura uniforme que permite una fácil expansión, mantenimiento y modificación de una aplicación.

El patrón *Modelo-Vista-Controlador* (*MVC*) es uno de los más extendidos patrones de diseño de software, su principal objetivo es aislar tanto los datos de la aplicación (modelo), como el estado de la misma, del mecanismo utilizado para representar dicho estado (vista).

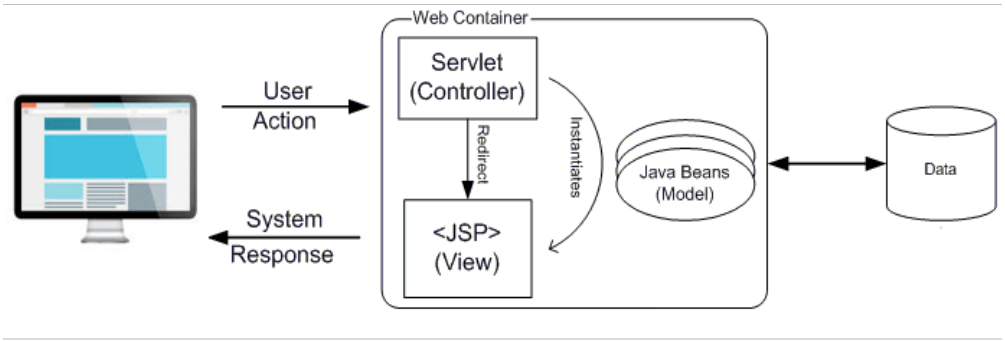


Figura 16 – Patrón de Diseño Modelo-Vista-Controlador

Los componentes pueden definirse de la siguiente manera:

- *El Modelo*: Es la representación de la información con la cual el sistema opera, es decir, gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación.
- *El Controlador*: Responde a eventos e invoca peticiones al modelo cuando se hace alguna solicitud sobre la información. También puede enviar comandos a su vista asociada si se solicita un cambio en la forma en que se presenta de modelo, por tanto se podría decir que el controlador hace de intermediario entre la vista y el modelo (*Middleware*).
- *La Vista*: Presenta el modelo en un formato adecuado para interactuar, por tanto requiere de dicho modelo la información que debe representar como salida.

3.4.2.1. Uso de Modelo de Vista Controlador en aplicaciones Web

El patrón *MVC* se utiliza frecuentemente en aplicaciones web, donde la vista es la página *HTML* y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de responder a los eventos de entrada desde la vista.

El patrón Modelo-Vista-Controlador ha sido ampliamente adaptado para diseñar e implementar aplicaciones web en los principales lenguajes de programación. Se han desarrollado multitud de *frameworks*, comerciales y no comerciales, que implementan este patrón; estos *frameworks* se diferencian básicamente en la interpretación de cómo las funciones del *Modelo-Vista-Controlador* se dividen entre cliente y servidor

Los primeros *frameworks* sobre *Modelo-Vista-Controlador* para desarrollo en la web planteaban un enfoque de cliente ligero en el que casi todas las funciones, tanto de la vista, el modelo y el controlador recaían en el servidor. En este enfoque, el cliente manda la petición de cualquier *hiperenlace* o formulario al controlador y después recibe de la vista una página completa y actualizada; tanto el modelo como el controlador y la vista están completamente alojados en el servidor. Hoy en día existen *frameworks* como *JavaScript MVC, Backbone o jQuery* que permiten que ciertos componentes del modelo se ejecuten parcial o totalmente en el cliente.

Este patrón de diseño es el utilizado para la implementación de esta aplicación, su principal beneficio es el aislamiento de cada componente de la aplicación, dotándoles de independencia, encapsulamiento de la información y robustez que favorecen la

mantenibilidad y escalabilidad, ya que cada una de las partes se encarga de un aspecto específico y no interfiere con el de las otras.

Siguiendo este patrón de diseño se puede utilizar un único componente funcional y ser representado gráficamente al usuario de muchas formas diferentes. Del mismo modo, en caso de que se desee modificar la forma en que se presenta la información, únicamente es necesario cambiar los componentes referentes a la vista, y si la modificación afectase a la forma en que se recupera la información, ésta sería transparente para las capas superiores de la aplicación. Por ejemplo, si se modificase la localización o el tipo del servidor de bases de datos, únicamente habría que corregir las clases de conexión y acceso a la base de datos, no siendo necesario modificar la lógica de la aplicación (*Servlets*) ni los componentes de presentación.

3.4.3. Framework MVC Apache-Struts

Struts es framework de la *Apache Software Foundation* para el desarrollo de aplicaciones Web bajo el patrón MVC para la plataforma *J2EE* (*Java Enterprise Edition*), esto es, una estructura jerarquizada de artefactos (clases, interfaces, etc.) y metodología de trabajo que permiten servir de base para simplificar notablemente la dificultad de implementación de una aplicación.

Por supuesto *Struts* no es el único *framework* MVC existente en el mundo J2EE. Aunque existen muchos otros, por ejemplo *Spring* o *Java Server Faces (JSF), Struts* es el más extendido con mucha diferencia.

El *framework* provee la infraestructura básica para la implementación del patrón MVC, permitiendo así que los desarrolladores puedan concentrarse en la lógica de negocio.

Struts proporciona la integración con el modelo, la lógica de negocio se implementa basándose en clases predefinidas por el framework y la construcción de la interfaz está soportada por la utilización de un conjunto de *tags* predefinidos, buscando evitar el uso de *scriptlets* (código *Java* incluido en el *JSP* dentro de etiquetas <% %>).

En este sentido, así es como implementa Struts los componentes del patrón Modelo-Vista-Controlador:

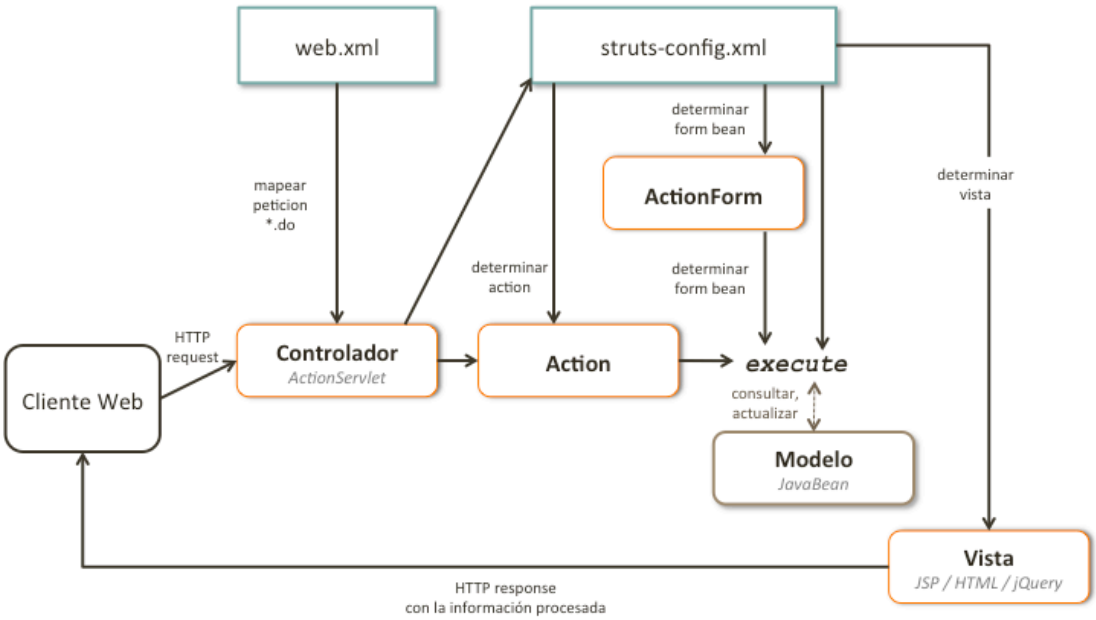


Figura 17 – Framework MVC *Apache-Struts*

- El **controlador** es un *Servlet*, de una clase incluida la librería de *Struts*. Será necesario configurar la aplicación web (a través del fichero *web.xml*) para que todas las peticiones del usuario se redirijan a este *Servlet*.
- El controlador distribuye las peticiones que recibe del usuario a la clase encargada de **ejecutar la acción**, ésta debe heredar de *Action*.
- La **vista** normalmente es implementada por JSP, no obstante, *Struts* proporciona 2 herramientas para la presentación de datos:
  - Los ***ActionForms*** son clases que capturan los datos introducidos en formularios y permiten su validación.
  - Las ***taglibs*** facilitan el trabajo con formularios, además de encargarse de mostrar los errores.
- La implementación del **modelo** al ser propio de la capa de negocio no está dentro del ámbito de *Struts*, por lo tanto, debe ser desarrollada íntegramente.

3.4.4. Estructura de la aplicación

En esta sección se describen los aspectos que resultan más relevantes para comprender la implementación realizada para el desarrollo de la aplicación.

- **Clases:** Dentro del directorio ***src*** se encuentran todas las clases *Java* del sistema, organizadas en diferentes paquetes de manera que se pueda organizar el código de tal forma que sea sencillo determinar aquellas clases e interfaces relacionadas, así como poder encontrar más fácilmente una clase dentro de la estructura.

  La división de los paquetes se ha realizado desde una perspectiva del ámbito tecnológico-funcional en el que se engloban los diferentes elementos que componen la lógica de la aplicación Web:

  - **nucleo**: contiene las clases que gestionan el acceso a base de datos (*DAOs*) y *Java Beans*, a su vez está dividido en paquetes según los objetos funcionales del sistema: noticias, niveles, menús, etc.
  - **servlets**: contiene el *Servlet* de inicio de la aplicación.

- **sistema**: contiene las clases que gestionan la conexión a la base de datos.
- **util**: contiene clases *Java* de utilidades y constantes.
- **struts**: contiene las clases de lógica de la aplicación. A su vez se encuentran divididas en subcarpetas:
  - **actions**: contiene las clases correspondientes a las acciones de *Struts,* éstas son clases *Java*, que se dispararán para implementar la lógica de negocio en respuesta de una petición HTTP del cliente que tiene asociad.
  - **forms**: contiene las clases que extienden los *ActionForms* de *Struts,* las cuales implementan los métodos get/set para cada input de la página, así como los métodos *validate y reset* de los formularios de la capa de presentación.

- **Componentes visuales:** Dentro de la carpeta **"WebContent"** del proyecto se ha establecido una estructura de directorios con los componentes tanto estáticos (HTML) como dinámicos (JSP) que componen las pantallas de la aplicación.
  - En la raíz del directorio se han incluido las páginas con los contenidos funcionales de la aplicación, es decir, las altas, modificaciones, eliminaciones, listados, etc. Prácticamente todas ellas son JSP ya que componen el contenido dinámicamente.
  - **/css:** Contiene los archivos correspondientes a las hojas de estilo que proporcionan la apariencia a las páginas de la aplicación.
  - **/img:** Contiene los ficheros con las imágenes utilizadas en las páginas de la aplicación.
  - **/js:** Contiene los ficheros con código JavaScript que se utiliza en las páginas de la aplicación.
- **WEB-INF:** En él se encuentran sendos archivos de configuración de la aplicación web tales como:
  - **web.xml.** Es el descriptor de despliegue, en él se recogen los requisitos de configuración, por ejemplo, la página principal de la aplicación, el periodo de caducidad de la sesión, el mapeo entre las *URL* y la clase *Java* asociada a cada *Servlet*.
  - **TLDs.** Contiene los archivos descriptores de las librerías de *struts*, estos archivos contienen la información de la librería en su conjunto y de cada una de las etiquetas que la componen. Es utilizado por el contenedor de JSPs a la hora de validar la utilización de las etiquetas de la librería.
  - **lib:** Aquí se recogen todas las librerías necesarias para la correcta compilación y ejecución de la aplicación Web que son utilizadas en distintas partes de la aplicación para llevar a cabo el conjunto de funcionalidades descritas.
- **build.** Este directorio contiene los archivos compilados para las clases del proyecto.
- **Otros recursos**
  - **log4j.** Los archivos necesarios para la configuración de la librería *log4j* se encuentran ubicados en la raíz del directorio de fuentes (*src*).

Esta librería permite a los desarrolladores escribir mensajes cuyo propósito es dejar constancia de una determinada transacción en tiempo de ejecución. *log4j* permite filtrar los mensajes en función de su importancia, pudiendo ser configurado el nivel de granularidad en tiempo de ejecución utilizando los ficheros de configuración.

En la siguiente imagen se muestra la estructura de directorios y paquetes que se acaba de definir en esta sección. En la columna de la izquierda se incluyen todos los paquetes de código *Java* y en la columna de la derecha los elementos *Web* de la aplicación:

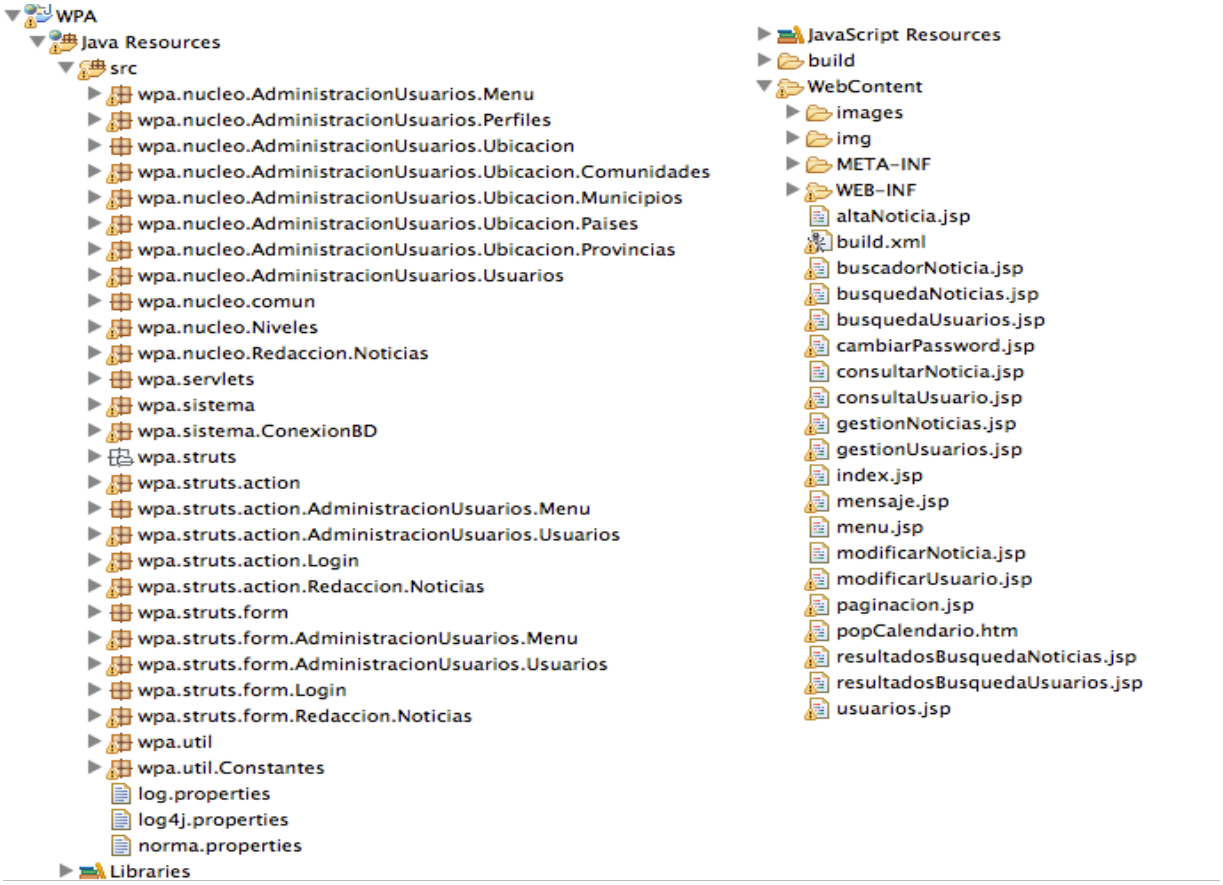


Figura 18 – Diseño: Estructura de componentes de la aplicación

3.5. Resultado Obtenido

Puesto que el objetivo principal del proyecto es el desarrollo de una aplicación Web que presente vulnerabilidades que permitan realizar ataques de inyección SQL, a continuación se incluyen algunas muestras sobre el prototipo inicial de la aplicación que permiten evidenciar la existencia de éstas.

3.5.1. Prueba Inyección 1

Se pretende realizar un ataque de inyección SQL en la pantalla de acceso a la aplicación, para intentar burlar la seguridad y poder acceder sin tener los datos de usuario y contraseña.

- **Pantalla de prueba**
  Pantalla de *login*

- **Resultado Esperado**
  Acceder a la aplicación y poder operar
- **Datos de Prueba:**
  - Campo "*Usuario*": (cualquier valor)
  - Campo "*Password*": `' OR ' '='`

Figura 19 – Datos entrada Prueba inyección 1

Como se puede ver en la imagen a continuación se ha accedido a la aplicación con un usuario con todos los permisos, puesto que aparecen todas las opciones de menú del sistema.

En esta pantalla además de comprobar el acceso, en la barra de navegación de la *URL*, se puede ver que los campos introducidos no son la contraseña del usuario si no los datos de prueba.

Figura 20 – Resultado Prueba Inyección 1

3.5.2. Prueba Inyección 2

Se pretende realizar un ataque de inyección SQL en la pantalla de acceso a la aplicación, para intentar burlar la seguridad y poder acceder sin tener los datos de usuario y contraseña.

- **Pantalla de prueba**
  Pantalla de *login*
- **Resultado Esperado**
  Acceder a la aplicación y poder operar
- **Datos de Prueba:**
  - Campo "*Usuario*": `' OR 1=1 OR password ='`
  - Campo "*Password*": *(espacio blanco)*

Figura 21 – Datos entrada Prueba inyección 2

Como se puede ver en la imagen a continuación se ha accedido a la aplicación con un usuario con todos los permisos, puesto que aparecen todas las opciones de menú del sistema.

En esta pantalla además de comprobar el acceso, en la barra de navegación de la *URL*, se puede ver que los campos introducidos no son la contraseña del usuario si no los datos de prueba.

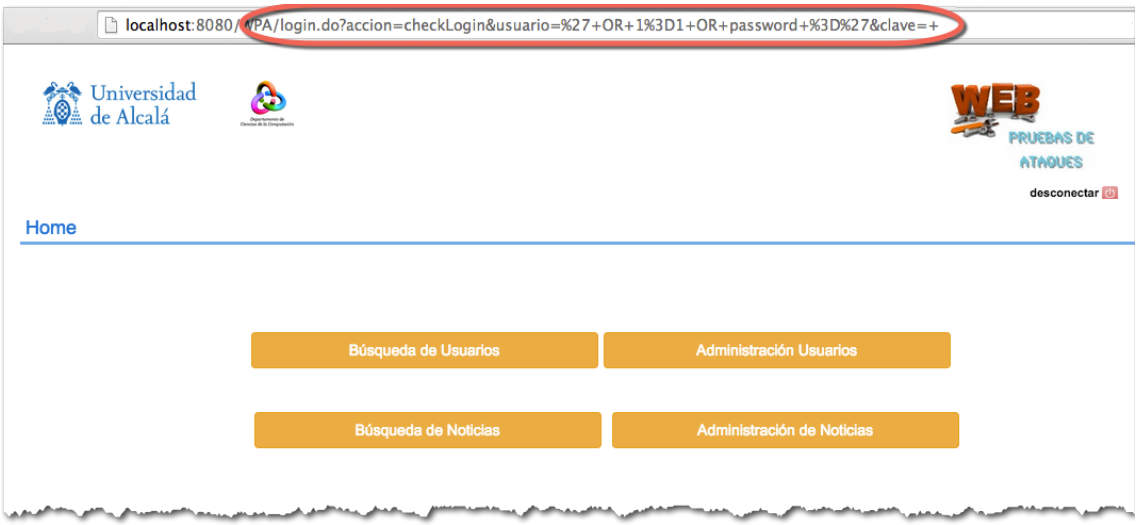


Figura 22 – Resultado Prueba Inyección 1

3.5.3. Prueba Inyección 3

Se pretende realizar un ataque de inyección SQL en la pantalla de búsqueda de noticias, que vulnere la seguridad de la aplicación, de modo que se pueda ejecutar una consulta SQL que

nos permita recuperar los alias y las contraseñas de acceso al sistema de todos los usuarios registrados.

- **Pantalla de prueba**
  Pantalla de búsqueda de noticias
- **Resultado Esperado**
  Recuperar el listado de alias y contraseñas para el acceso al sistema
- **Datos de Prueba:**
  - Campo *Título:* `SEGURIDAD' UNION SELECT 0 , USUARIO AS TITULO, PASSWORD AS TEXTO_NOTICIA, '', '', '', NULL,NULL, '', '' FROM USUARIOS --`
    Nota: Podría utilizarse cualquier campo del formulario con el mismo contenido.*

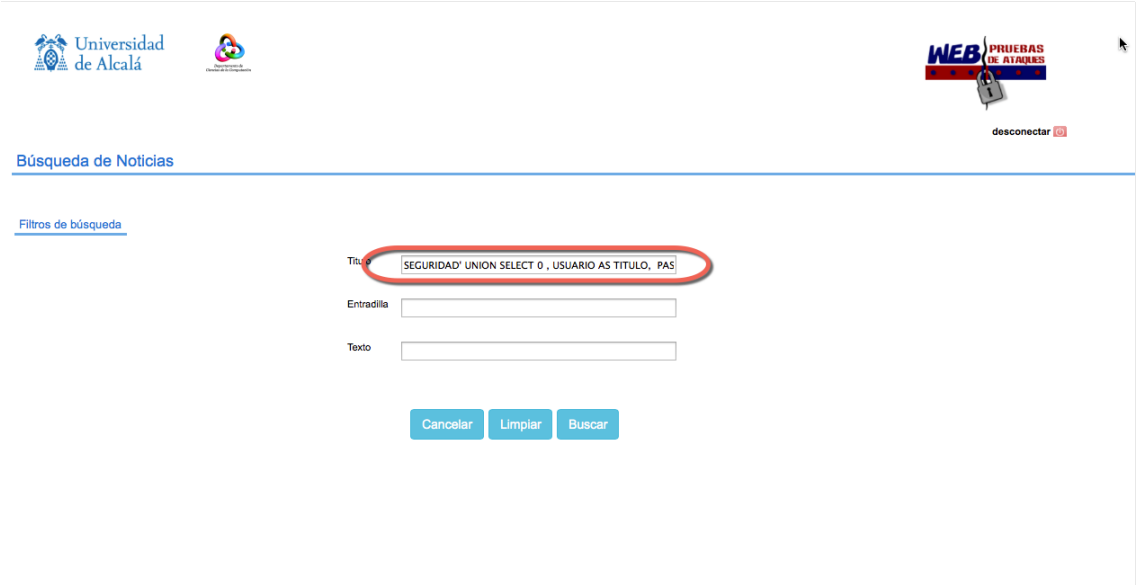


Figura 23 – Datos entrada Prueba inyección 3

Una vez escrita la consulta *SQL* anidada utilizando la cláusula *UNION* al pulsar el botón *Buscar* se ejecuta la consulta *SQL* que devolverá, además de todos los registros coincidentes de la tabla *Noticias,* aquellos resultantes de la sentencia anidada, en este caso, todos los alias y contraseñas de los usuarios del sistema.

En este caso se ha introducido un criterio para que no existan registros coincidentes en la tabla noticias y de este modo vemos en la imagen que se muestra a continuación que el listado contiene los usuarios y contraseñas del sistema.

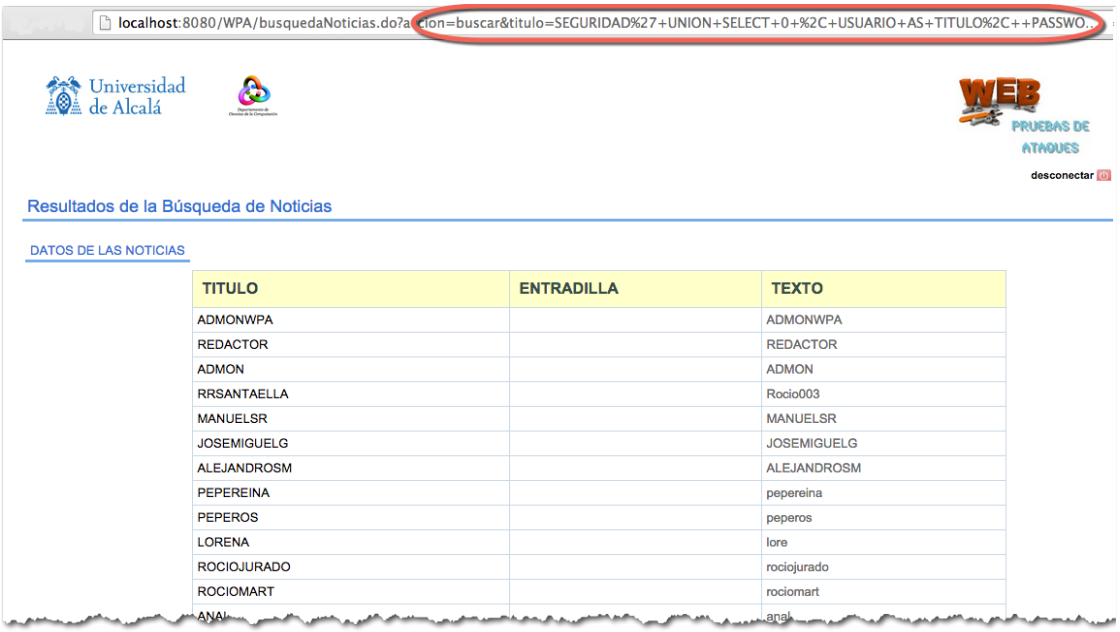


Figura 24 – Resultado Prueba Inyección 3

3.5.4. Prueba Inyección 4

Se pretende realizar un ataque de inyección SQL en la pantalla de búsqueda de noticias, que vulnere la seguridad de la aplicación, de modo que se pueda ejecutar una consulta SQL que nos permita recuperar información sobre el esquema de la base de datos.

- **Pantalla de prueba**
  Pantalla de búsqueda de noticias
- **Resultado Esperado**
  Recuperar información del esquema de la base de datos y de la tabla de usuarios.
- **Datos de Prueba:**
  - Campo *Título:* `’ UNION SELECT 0, TABLE_SCHEMA AS TITULO, TABLE_NAME AS TEXTO_NOTICIA, '', '', '', NULL, NULL, '', '' FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = ‘BASE TABLE’`

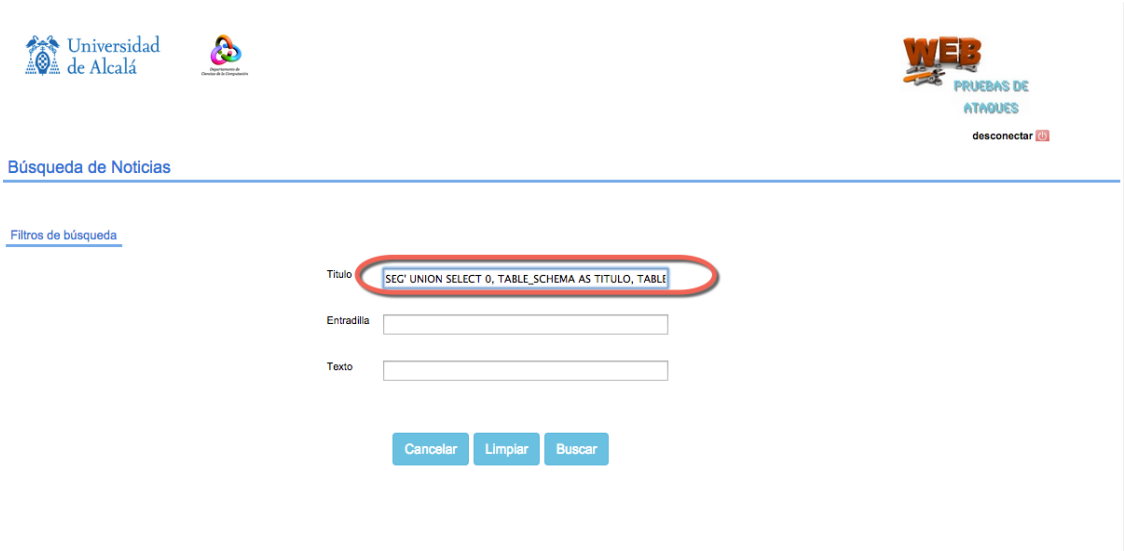
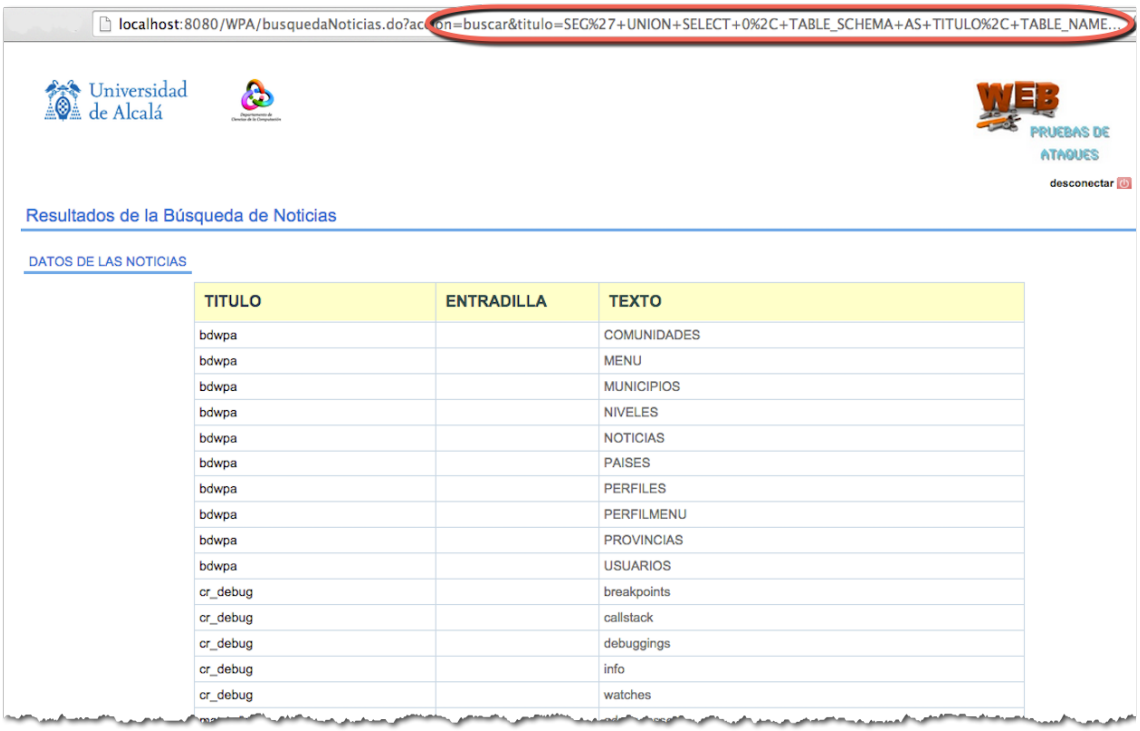


Figura 25 – Datos entrada Prueba inyección 4

Una vez escrita la consulta *SQL* anidada utilizando la cláusula *UNION* al pulsar el botón *Buscar* se ejecuta la consulta *SQL* que devolverá además de todos los registros coincidentes de la tabla Noticias aquellos resultantes de la sentencia anidada, en este caso, todos los alias y contraseñas de los usuarios del sistema.



TITULO	ENTRADILLA	TEXTO
bdwpa		COMUNIDADES
bdwpa		MENU
bdwpa		MUNICIPIOS
bdwpa		NIVELES
bdwpa		NOTICIAS
bdwpa		PAISES
bdwpa		PERFILES
bdwpa		PERFILMENU
bdwpa		PROVINCIAS
bdwpa		USUARIOS
cr_debug		breakpoints
cr_debug		callstack
cr_debug		debuggings
cr_debug		info
cr_debug		watches

Figura 26 - – Resultado Prueba Inyección 4

3.6. Conclusiones

La mayor parte de los ataques a los sistemas informáticos son provocados, intencionadamente o no, por personas que en general persiguen conseguir un nivel de privilegio en el sistema que les permita realizar acciones no autorizadas sobre el mismo.

En el presente Trabajo Fin de Grado se ha diseñado una aplicación Web intuitiva y fácil de usar para cualquier usuario, pero cuyo objetivo principal era que garantizase la existencia de vulnerabilidades fácilmente detectables de tipo *SQL Injection,* de tal manera que permitiese a los usuarios entrenar este tipo de ataques en un entorno seguro y controlado.

El origen de estas vulnerabilidades radica en el incorrecto chequeo y/o filtrado de las variables utilizadas en un programa que contiene, o genera, código SQL.

Las *inyecciones SQL* se ha convertido en un problema muy común con sitios web que cuentan con base de datos. La vulnerabilidad es fácilmente detectada y fácilmente explotada, y como tal, cualquier sitio es propenso a ser objeto de un intento de ataque de este tipo.

Este tipo de intrusión normalmente es de carácter malicioso, dañino o espía, por tanto es un problema de seguridad informática, y para poder prevenirlo y evitarlo debe ser tomado en cuenta por el programador de la aplicación. Un programa elaborado con descuido, indiferencia o con ignorancia del problema, podrá resultar ser vulnerable, y la seguridad del sistema podrá quedar eventualmente comprometida.

Existen ciertas técnicas, medidas o mecanismos que permitirán evitar uno o varios de los sistemas de ataque por *SQL Injection* desde el uso de *suites* preconstruidas o la implementación de diferentes medidas de seguridad que incorporen *filtros* que minimicen las acciones que pueden realizar los usuarios en las consultas a la base de datos.

El hecho de haber investigado acerca de las vulnerabilidades a las que están expuestos los sistemas, y en concreto las de *SQL Injection,* me ha permitido tomar consciencia del enorme problema que pueden suponer para cualquier persona u organización, ya que podría poner de manifiesto información confidencial o sensible de clientes, usuarios, pacientes, etc. que siendo utilizada con fines negativos podría ocasionar grandes perjuicios a las víctimas del ataque.

Así mismo, he podido estudiar algunas de las técnicas de prevención, constatando la importancia que tiene el hecho de que el programador Web tenga constancia de su existencia y de su aplicación a la hora de la construcción de aplicaciones, no sólo Web, sino en todas aquellas en las que haya interacción con usuarios. Y es que como desarrolladores tenemos una gran responsabilidad sobre la seguridad de las aplicaciones en las que participamos y no pensamos en ello con la debida frecuencia. Cuando desarrollamos una aplicación sobre nóminas, una aplicación sobre informes médicos, o cualquier otro tipo de información sensible, lo que está en juego no es sólo la integridad de la aplicación, detrás hay usuarios, vidas de personas.

3.6.1. Trabajo futuro

La aplicación podrá ir evolucionando para aumentar o disminuir el nivel de dificultad de la seguridad y de los diferentes tipos de ataques.

Por otro lado, se podría trabajar en la aplicación desde el punto de vista del diseño de un nivel de dificultad "aplicación segura", en el que se apliquen en la construcción técnicas y recomendaciones para la creación de aplicaciones web seguras, mencionadas algunas de ellas en otros capítulos del presente documento.

Del mismo modo, desde una perspectiva de aumentar el propósito docente de este proyecto, se puede seguir trabajando en desarrollar una infraestructura con diversas vulnerabilidades de diferentes tipos que permita a los alumnos poner en práctica los conocimientos adquiridos en un entorno seguro, ya que la realización de ataques a sistemas sin autorización expresa de su propietario podría constituir un delito.

Capítulo 4.

Diagramas

4.1. Diagramas de Casos de Uso

Analizando con mayor grado de detalle los requisitos definidos para el sistema se extraen los siguientes pasos o actividades que deberán realizarse para llevar a cabo cada una de las funcionalidades descritas.

Con el objetivo de representar de una manera fácil e intuitiva cada una de las partes de cada funcionalidad se incluye una representación gráfica de las mismas por medio de los denominados Diagramas de Casos de Uso.

4.1.1.C.U.01: *Log in* Usuario

- **Diagrama**

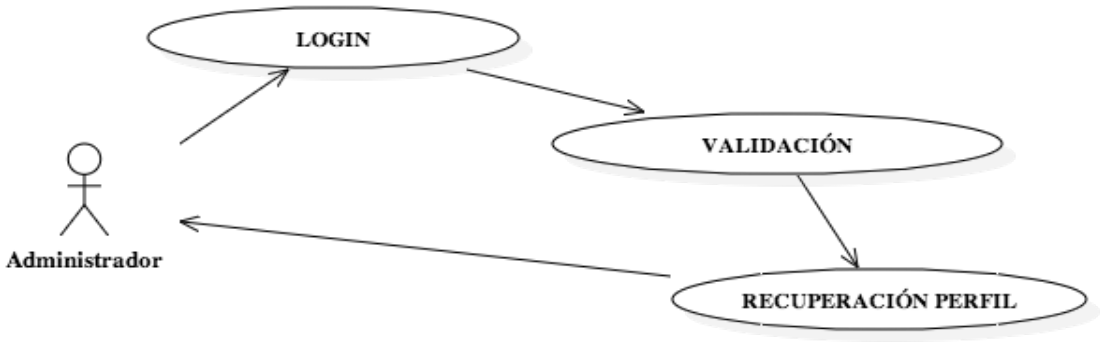


Diagrama 1 – C.U.01: Log in de usuario

- **Definición de acciones**

Definición Acciones	
Nombre	Log in
Propósito	Acceder al sistema
Precondiciones	N/E
Post-condiciones	Una vez logado, se pasa a validación en donde se determina qué tipo de acceso tendrá el usuario, según el perfil asociado.

Definición Acciones	
Nombre	Validación
Propósito	Identificar el perfil del usuario logado
Precondiciones	Acceso al sistema con el Usuario y Contraseña.
Post-condiciones	Accede al perfil específico del Usuario logado.

Definición Acciones	
Nombre	Recuperación Perfil
Propósito	Acceder a las opciones de cada perfil en función de su "rango"

4.1.2.C.U.02: Gestión

- **Diagrama**

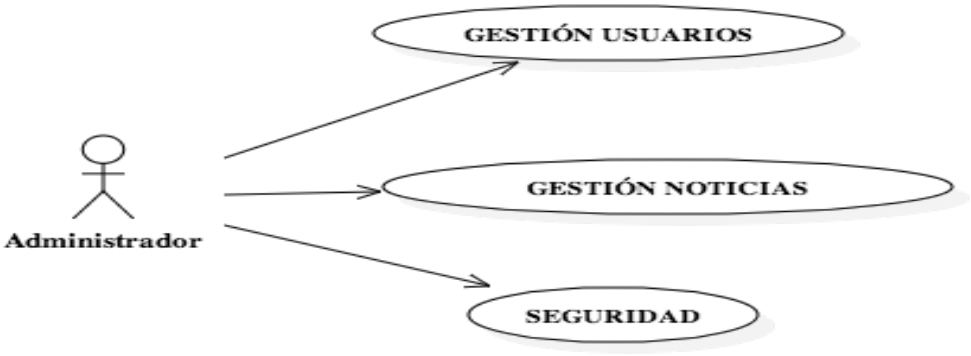


Diagrama 2 – C.U.02: Gestión, Administración

- **Definición de acciones**

Definición Acciones	
Nombre	Gestión Usuarios
Propósito	Acceder a las opciones de Gestión de Usuarios
Precondiciones	Se ha finalizado el proceso de autenticación. Se debe haber autentificado con perfil administrador.
Post-condiciones	Se accede a modificar o actualizar los datos de los usuarios.

Definición Acciones	
Nombre	Redacción Noticias
Propósito	Acceder a las opciones de redacción de noticias
Precondiciones	Se ha finalizado el proceso de autenticación. Se debe haber autentificado con perfil administrador.
Post-condiciones	Se accede a las opciones de redacción de noticias.

Definición Acciones	
Nombre	Seguridad
Propósito	Acceder a modificar el nivel de seguridad
Precondiciones	Se ha finalizado el proceso de autenticación. Se debe haber autentificado con perfil administrador.

Post-condiciones	Se accede a modificar las opciones de seguridad siempre que sea un usuario autorizado (administrador).

4.1.3.C.U.03: Búsqueda Usuarios

- **Diagrama**

Diagrama 3 – C.U.03: Búsqueda de usuario

- **Definición de acciones**

Definición Acciones	
Nombre	Búsqueda de Usuarios
Propósito	Acceder a las opciones de Búsqueda de Usuarios
Precondiciones	Se ha finalizado el proceso de autenticación. Se debe haber autentificado con perfil administrador.
Post-condiciones	Se accede a la pantalla de búsquedas de usuarios para poder realizar consultas a través de varios filtros.

4.1.4.C.U.04: Búsqueda Noticias

- **Diagrama**

Diagrama 4 – C.U.04: Búsqueda de noticias

- **Definición de acciones**

Definición Acciones	
Nombre	Búsqueda de Noticias
Propósito	Acceder a las opciones de Búsqueda de noticias
Precondiciones	Se ha finalizado el proceso de autenticación. Se debe haber autentificado con perfil administrador.
Post-condiciones	Se accede a la pantalla de búsquedas de noticias para poder realizar consultas a través de varios filtros.

4.1.5.C.U.05: Gestión Usuarios

- **Diagrama**

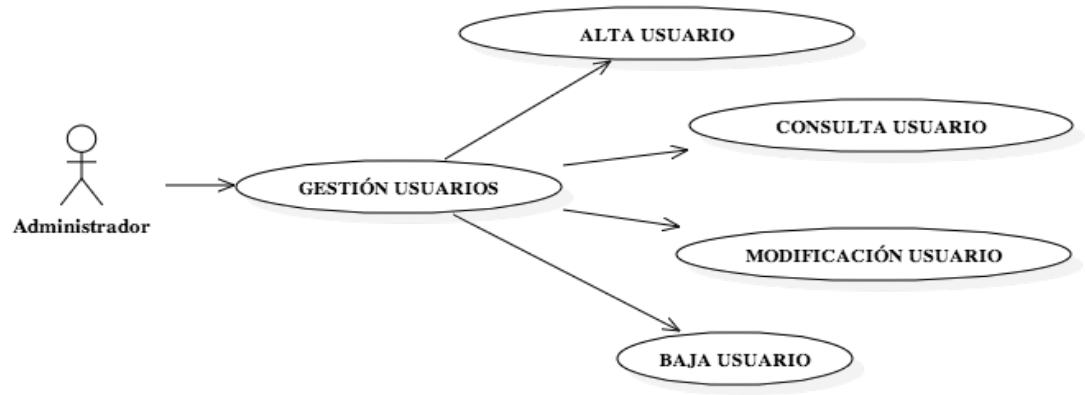


Diagrama 5 – C.U.05: Gestión de usuarios

- **Definición de acciones**

Definición Acciones	
Nombre	Alta Usuario
Propósito	Realizar el registro en el sistema de un nuevo usuario
Precondiciones	El usuario debe haberse autentificado previamente con perfil *Administrador*.
Post-condiciones	Se accede a realizar el alta de un usuario.

Definición Acciones	
Nombre	Consultar Detalle Usuario
Propósito	Realizar la consulta de los datos de un usuario determinado
Precondiciones	El usuario debe haberse autentificado previamente con perfil *Administrador*.
Post-condiciones	Se accede a visualiza la consulta de datos del usuario.

Definición Acciones	
Nombre	Modificar Usuarios
Propósito	Realizar la modificación de los datos de un usuario determinado
Precondiciones	El usuario debe haberse autentificado previamente con perfil *Administrador*..
Post-condiciones	Se accede a realizar la modificación.

Definición Acciones	
Nombre	Baja Usuarios

Propósito	Realizar la baja física del sistema de un usuario determinado
Precondiciones	El usuario debe haberse autentificado previamente con perfil *Administrador*.
Post-condiciones	Se accede a realizar la el borrado del usuario.

Definición Acciones	
Nombre	Desactivación Usuarios
Propósito	Realizar la baja lógica del sistema de un usuario determinado
Precondiciones	El usuario debe haberse autentificado previamente con perfil *Administrador*.
Post-condiciones	Se accede a realizar la desactivación del usuario.

Definición Acciones	
Nombre	Activación Usuarios
Propósito	Realizar la Activación de un usuario ya existente en el sistema
Precondiciones	Que el usuario se encuentre en estado desactivado. El usuario debe haberse autentificado previamente con perfil *Administrador*.
Post-condiciones	Se accede a realizar la activación del usuario.

4.1.6. C.U.06: Gestión Noticias

- **Diagrama**

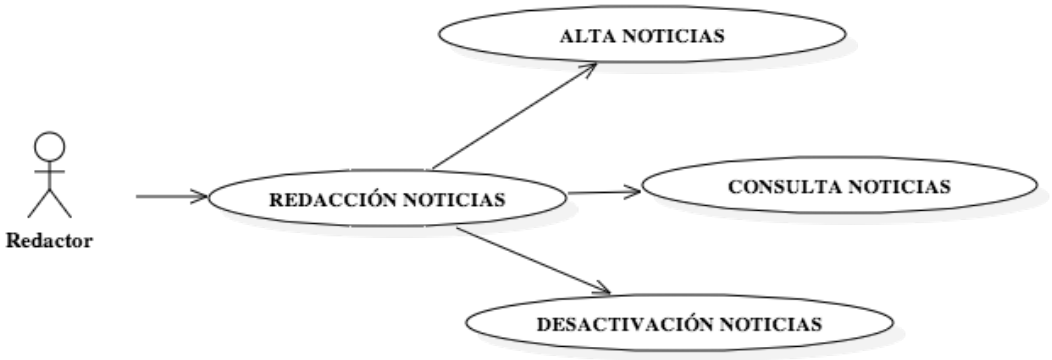


Diagrama 6 – C.U.06: Gestión Noticias

- **Definición de acciones**

Definición Acciones	
Nombre	Alta Noticia
Propósito	Realizar el registro en el sistema de una noticia

Precondiciones	El usuario debe haberse autentificado previamente con perfil *Administrador* o *Redactor*.
Post-condiciones	Se accede a realizar el alta de una nueva noticia.

Definición Acciones	
Nombre	Consultar Detalle noticia
Propósito	Realizar la consulta de los datos de una noticia determinada
Precondiciones	El usuario debe haberse autentificado previamente con perfil *Administrador* o *Redactor*.
Post-condiciones	Se accede a visualiza los datos de la noticia.

Definición Acciones	
Nombre	Modificar Noticias
Propósito	Realizar la modificación de los datos de una noticia determinada
Precondiciones	El usuario debe haberse autentificado previamente con perfil *Administrador* o *Redactor*.
Post-condiciones	Se accede a realizar la modificación de los datos de la noticia seleccionada.

Definición Acciones	
Nombre	Desactivación Noticias
Propósito	Realizar la baja lógica del sistema de una noticia determinada
Precondiciones	El usuario debe haberse autentificado previamente con perfil *Administrador* o *Redactor*.
Post-condiciones	Se accede a realizar la desactivación de la noticia. Ya no podrá ser consultada en el sistema.

4.1.7. C.U.07: Gestión Seguridad

- **Diagrama**

Diagrama 7 – C.U.07: Gestión Seguridad

- **Definición de acciones**

Definición Acciones	
Nombre	Seguridad
Propósito	Acceder a modificar el nivel de seguridad
Precondiciones	El usuario debe haberse autentificado previamente con perfil *Administrador*
Post-condiciones	Se accede a modificar el nivel de seguridad siempre que sea un usuario del perfil autorizado.

4.2. Diagrama Modelo Entidad Relación

Del análisis realizado sobre los requisitos definidos se extrae la necesidad de las entidades de datos: *Perfil, Usuario, Noticia, Nivel y Menú,* encargadas de ser el soporte para la implementación de las diversas funcionalidades del sistema.

En el siguiente diagrama, denominado Modelo Entidad-Relación, se representan de manera gráfica los diferentes componentes así como la forma en la que se relacionan entre sí:

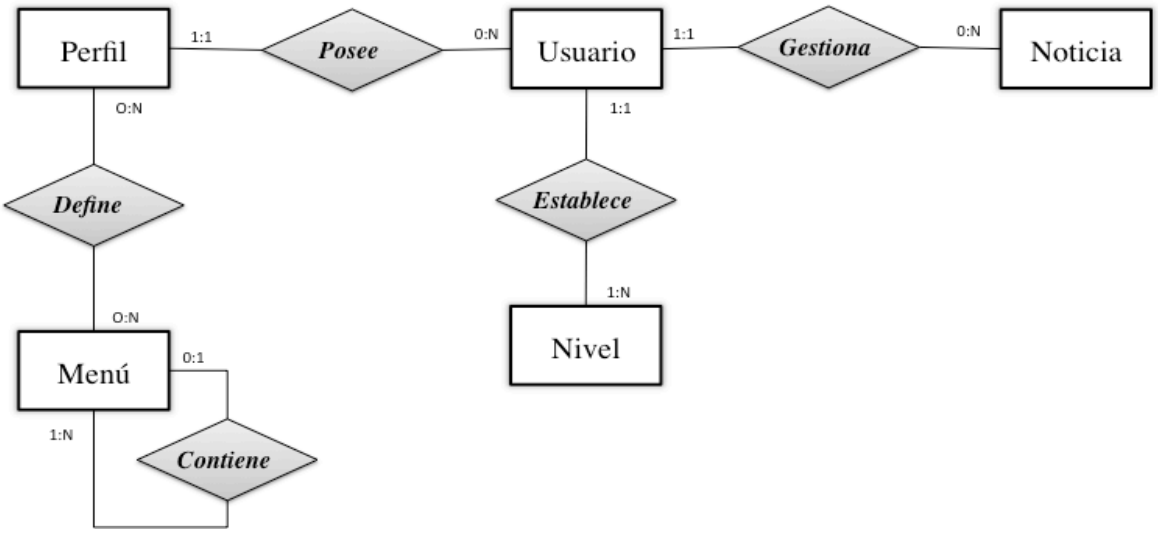


Diagrama 8 – Diagrama Modelo Entidad Relación

A continuación se describen las entidades que conforman el diagrama de forma abreviada:

- **Menú**
  Utilizada para almacenar las diferentes opciones de menú que se mostrarán en la aplicación en función del perfil que tenga el usuario.

  La información principal que debe almacenar es la referente al *nombre, URL* de acceso y *opción jerárquicamente superior*, que permitirá establecer el árbol de opciones de menú.

- **Nivel**
  Define la información que se guardará en la aplicación referente a los diferentes niveles de seguridad que pueden ser configurados por el usuario desde la aplicación.

Aquí se almacenará la información referente al nivel de seguridad tal como *identificador, nombre, indicador de si está activo* y *programa objeto del nivel.*

- **Noticia**
  Entidad responsable de almacenar toda la información referente a las diferentes noticias generadas por los usuarios del sistema.

  Para generar una nueva noticia los usuarios podrán informar *título, entradilla, texto de la noticia,* así como una *imagen asociada.*

- **Perfil**
  Esta entidad determina los diferentes niveles de visibilidad de la información que tendrán cada uno de los usuarios del sistema, estableciendo además la relación entre los usuarios y las distintas funcionalidades a las que éstos tendrán acceso.

  Para establecer los perfiles de visualización de la información se necesitan un *identificador, descripción* y *nivel de visibilidad.*

- **Usuario**
  Su utilidad es representar a alto nivel el concepto de usuario que interactúa con el sistema.

  Sus atributos principales se corresponden con los datos que determinan al usuario, tales como *usuario, contraseña, nombre*, *apellidos, fecha de nacimiento, dirección* y *datos de contacto.*

4.2.1. Modelo de Datos

A partir del análisis de entidades de alto nivel anterior se puede realizar el diseño del modelo de datos de la aplicación.

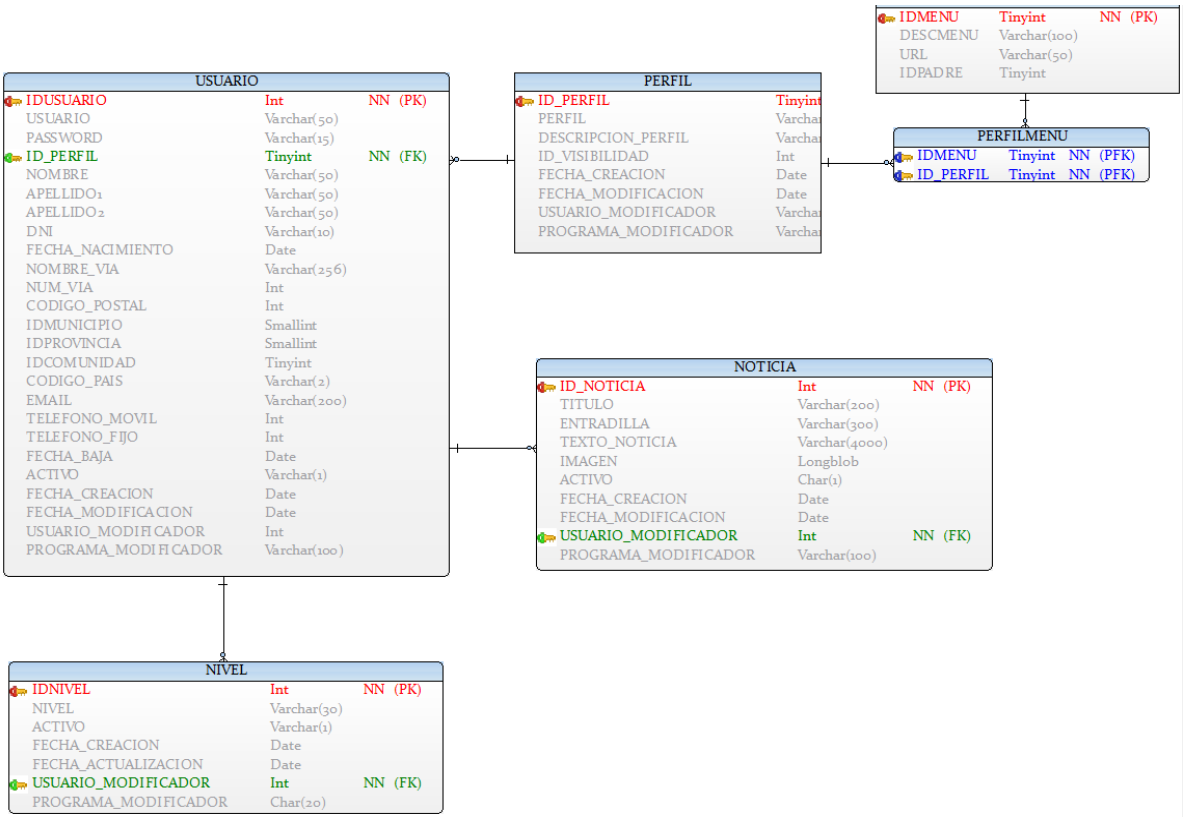


Diagrama 9 – Modelo de Datos

MENU				
Nombre del tipo de tabla	MAESTRA			
Descripción	Tabla donde se guardaran los datos referentes a las opciones de menú disponibles en la aplicación			
Nombre	**Definición**	**Tipo de dato**	**PK**	**FK**
IDMENU	Dato identificativo del menú para trabajar a nivel interno de la aplicación	Tinyint	SI	No
DESCMENU	Descripción del menú	Varchar(50)	No	No
URL	Action de la aplicación al que se redirigirá ese registro del menú	Varchar(50)	No	No
IDPADRE	Indicador de si un menú es dependiente de otro	Tinyint	No	No

Tabla 20 – Diseño: BBDD Tabla Menú

NIVEL				
Nombre del tipo de tabla	MAESTRA			
Descripción	Tabla donde se guardaran los posibles niveles de seguridad configurables en la aplicación			
Nombre	**Definición**	**Tipo de dato**	**PK**	**FK**
IDNIVEL	Dato identificativo del nivel para trabajar a nivel interno de la aplicación	Int	Yes	No
NIVEL	Descripción del nivel	Varchar(30)	No	No
ACTIVO	Indicador de si el registro estará activo o no	Char(255)	No	No
FECHA_CREACION	Dato de auditoría	Datetime	No	No
FECHA_ACTUALIZACION	Dato de auditoría	Datetime	No	No
USUARIO_MODIFICADOR	Dato de auditoría	Int	No	No
PROGRAMA_MODIFICADOR	Dato de auditoría	Varchar(100)	No	No

Tabla 21 – Diseño: BBDD Tabla Nivel

NOTICIA				
Nombre del tipo de tabla	DATOS			
Descripción	Tabla para guardar la información referente a las Noticias que son publicadas por los usuarios en la aplicación			
Nombre	**Definición**	**Tipo de dato**	**PK**	**FK**
ID_NOTICIAS	Dato identificativo de la noticia para trabajar a nivel interno de la aplicación	Int	Yes	No
TITULO	Título de la noticia	Varchar(200)	No	No
ENTRADILLA	Entradilla de la noticia	Varchar(300)	No	No
TEXTO_NOTICIA	Texto de la noticia	Varchar(4000)	No	No
IMAGEN	Fichero de imagen de la noticia	Longbob	No	No
ACTIVO	Indicador de si la noticia estará activa en la aplicación	Char(1)	No	No
FECHA_CREACION	Dato de auditoría	Datetime	No	No

FECHA_MODIFICACION	Dato de auditoría	Datetime	No	No
USUARIO_MODIFICADOR	Dato de auditoría	Int	No	No
PROGRAMA_MODIFICADOR	Dato de auditoria	Varchar(100)	No	No

Tabla 22 – Diseño: BBDD Tabla Noticia

PERFIL				
Nombre del tipo de tabla	MAESTRA			
Descripción	Datos de los Perfiles disponibles en la aplicación			
Nombre	**Definición**	**Tipo de dato**	**PK**	**FK**
ID_PERFIL	Dato identificativo del perfil para trabajar a nivel interno de la aplicación	Tinyint	SI	No
PERFIL	Perfil del usuario	Varchar(100)	No	No
DESCRIPCION_PERFIL	Descripción del perfil	Varchar(200)	No	No
FECHA_CREACION	Dato de auditoría	Datetime	No	No
FECHA_MODIFICACION	Dato de auditoría	Datetime	No	No
USUARIO_MODIFICADOR	Dato de auditoría	Varchar(50)	No	No
PROGRAMA_MODIFICADOR	Dato de auditoría	Varchar(50)	No	No

Tabla 23 – Diseño: BBDD Tabla Perfil

USUARIO				
Descripción	Tabla en la que se guardarán los datos de los usuarios con acceso a la aplicación			
Nombre del tipo de tabla	DATOS			
Nombre	**Definición**	**Tipo de dato**	**PK**	**FK**
IDUSUARIO	Dato identificativo del usuario para trabajar a nivel interno de la aplicación	Int	SI	No
USUARIO	Identificador personal del usuario	Varchar(50)	No	No
PASSWORD	Clave secreta de acceso a la aplicación	Varchar(15)	No	No
ID_PERFIL	Identificador de perfil con el que el usuario accederá a la aplicación	Tinyint	No	SI
NOMBRE	Nombre del usuario	Varchar(300)	No	No
APELLIDO1	Primer apellido del usuario	Varchar(300)	No	No
APELLIDO2	Segundo apellido del usuario	Varchar(300)	No	No
DNI	DNI del usuario	Varchar(10)	No	No
FECHA_NACIMIENTO	Fecha de nacimiento del usuario	Datetime	No	No
NOMBRE_VIA	Dato de la dirección del usuario	Varchar(255)	No	No
NUM_VIA	Dato de la dirección del usuario	Int	No	No
COGIO_POSTAL	Código postal del usuario	Int	No	No

IDMUNICIPIO	Municipio del usuario	Smallint	No	No
IDPROVINCIA	Provincia del usuario	Int	No	No
IDCOMUNIDAD	Comunidad del usuario	Tinyint	No	No
CODIGO_PAIS	País del usuario	Varchar(2)	No	No
EMAIL	Email que permitirá contactar con el usuario	Varchar(2000)	No	No
TELEFONO_MOVIL	Teléfono móvil del usuario	Int	No	No
TELEFONO_FIJO	Teléfono fijo del usuario	Int	No	No
FECHA_BAJA	Fecha de baja en la aplicación	Date	No	No
ACTIVO	Indica si el usuario está trabajando actualmente o no	Char(1)	No	No
FECHA_CREACION	Dato de auditoría	Datetime	No	No
FECHA_MODIFICACION	Dato de auditoría	Datetime	No	No
USUARIO_MODIFICADOR	Dato de auditoría	Int	No	No
PROGRAMA_MODIFICADOR	Dato de auditoría	Varchar(100)	No	No

Tabla 24 – Diseño: BBDD Tabla Usuario

PERFILMENU				
Nombre del tipo de tabla	CRUCE			
Descripción	Tabla de cruce que relaciona las diferentes opciones de menú que tendrán configurados los diferentes perfiles			
Nombre	**Definición**	**Tipo de dato**	**PK**	**FK**
Idperfil	Dato identificativo del perfil para trabajar a nivel interno de la aplicación	Tinyint	SI	SI
idmenu	Dato identificativo del menú para trabajar a nivel interno de la aplicación	Tinyint	SI	SI

Tabla 25 – Diseño: BBDD Tabla PerfilMenu

4.3. Diagrama de Clases

La estructura de clases de la capa del modelo de la aplicación debe representar una implementación del modelo de datos definido para el almacenamiento de la información en la base de datos.

Con el objetivo de representar de una manera gráfica e intuitiva la información se utiliza el conocido como Diagrama de Clases, el cual muestra la estructura estática del modelo del sistema, es decir, todo aquello que "exista" en el sistema, mostrando su estructura interna así como las relaciones entre los diferentes elementos.

El diagrama de clases permite crear el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

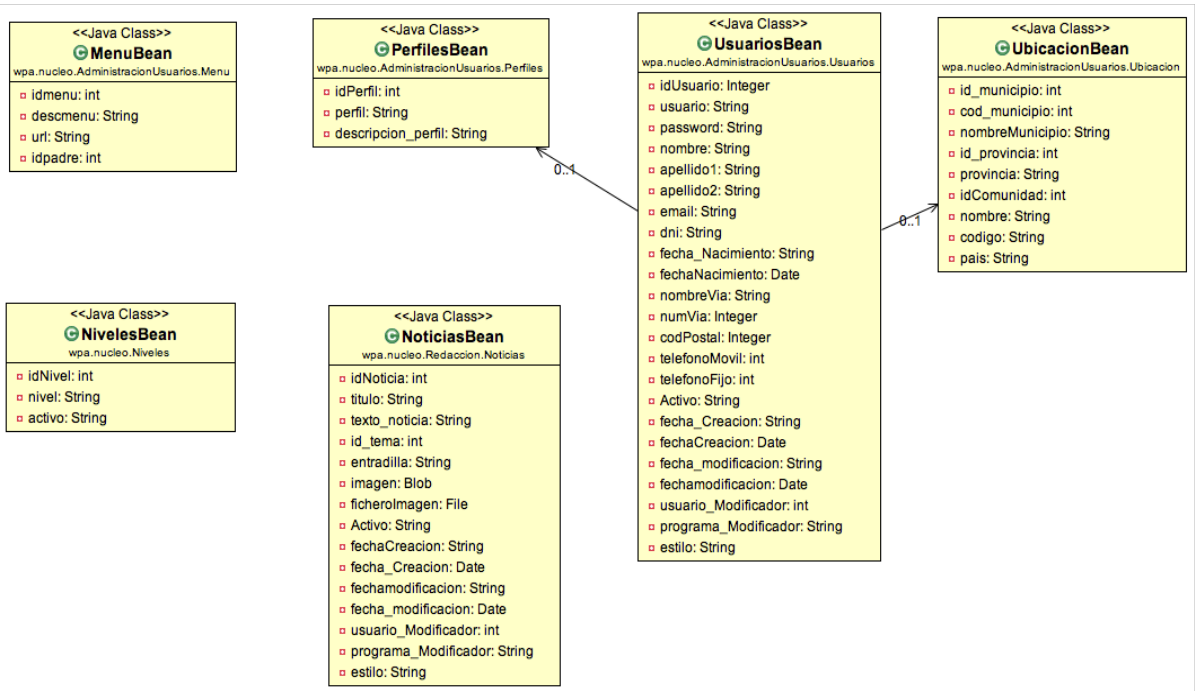


Diagrama 10 – Diagrama de clases aplicación Web

Cada uno de los elementos tiene como objetivo almacenar la información referente a una entidad del modelo de datos de manera que pueda ser utilizada de forma integrada en otras capas de la aplicación Web (lógica de negocio o capa de presentación), es decir, que se pueda trabajar bajo el paradigma de la orientación a objetos con un elemento con entidad propia que englobe un conjunto de atributos y no una serie de variables disgregadas, que sería más difícil de manipular.

Capítulo 5.

Pliego de condiciones

En este apartado se recogen todas las condiciones necesarias para poder llevar a cabo el desarrollo del proyecto especificado o bien la utilización del producto de software resultante.

5.1. Condiciones Generales

Desde el punto de vista tecnológico, todas las herramientas utilizadas para el desarrollo del producto objeto del proyecto poseen licencia *GNU GPL,* es decir, su distribución es gratuita y puede ser utilizada por cualquiera , ya que su finalidad es proteger los derechos de los usuarios finales (usar, compartir, estudiar y modificar).

A continuación se enumeran el listado de componentes y herramientas utilizado para el desarrollo de este proyecto:

- **eclipse** (versión Luna): Licenciado bajo *GNU Eclipse Public License* - v 1.0
- **MySQL** (v.5.5): *Oracle* licencia el servidor de bases de datos y las librerías bajo licenciamiento *GNU GPL versión 2* para desarrollos *Open Source*.

  Para distribución comercial *Oracle* tiene una versión comercial cuyos precios oscilan entre los 2.000 y 20.000 $ anuales en función del tipo de suscripción[2].
- **Java** (JRE 1.7): licenciado por Oracle bajo *Java Binary Code License Agreement*
- **JSP** (v.2.1): licencia *GNU GPL* versión 2
- **JavaScript**: cuenta con licencia *GNU GPL* versión 3.
- **CSS** (v.2.1): es una especificación de la W3C interpretada por la mayoría de exploradores web modernos.
- **Apache-Struts**: distribuida bajo licencia *Apache License* versión 2
- **Apache-Tomcat**: se publica bajo licencia *Apache License* versión 2
- **log4j**: distribuida bajo licencia *Apache License* versión 2

5.1.1. Normativa

La implantación y desarrollo de las TIC ha creado nuevas posibilidades de delincuencia antes impensables.

El delito informático es aquel que se refiere a actividades ilícitas realizadas por medio de dispositivos tecnológicos como ordenadores, móviles, portátiles, tabletas o de cualquier dispositivo interconectado como pueden ser las denominadas TV inteligentes (*Smart TV*), videoconsolas, coches, *wearables*, etc.

En este *Internet de las cosas interconectadas*, no sólo se incluyen los riesgos para la privacidad o el espionaje ya conocidos, sino que también incluye delitos tradicionales como

[2] Licenciamiento MySQL *Fuente: http://www.oracle.com/us/corporate/pricing/price-lists/mysql-pricelist-183985.pdf*

fraude, robo, estafa, chantaje, falsificación y malversación de caudales públicos. Los perjuicios ocasionados por este tipo de delitos son superiores a los de la delincuencia tradicional y también es mucho más difícil descubrir a los culpables.

La *Organización de Naciones Unidas* (*ONU*) reconoce los siguientes tipos de delitos informáticos:

- Fraudes cometidos mediante manipulación de ordenadores.
- Manipulación de los datos de entrada.
- Daños o modificaciones de programas o datos computarizados.

Desde el punto de vista de legislación, existen una serie de leyes nacionales y directivas de la Unión Europea que resultan en mayor o menor medida aplicables al ámbito de los sistemas de información y/o la seguridad de dicha información.

5.1.1.1. Legislación Nacional

Desde la publicación de la *Ley Orgánica de Protección de Datos de Carácter Personal* en el año 1999 hasta la *Ley de Acceso Electrónico de los Ciudadanos a los Servicios Públicos* del año 2007, hay una serie de leyes que, de una manera u otra, están relacionadas con la seguridad de la información, además de numerosas regulaciones sectoriales en diversos ámbitos: financiero, telecomunicaciones, agrario, etc.

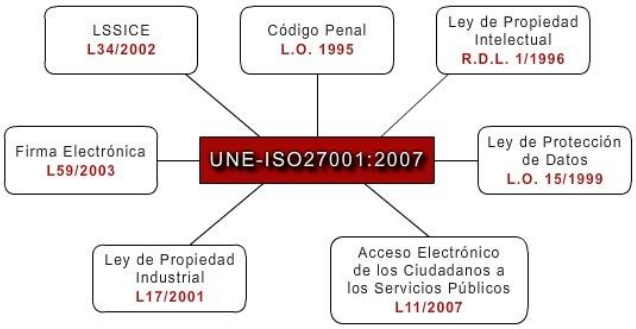


Figura 27- Legislación vigente aplicable a la seguridad de la información

- **Ley Orgánica 15/99 de Protección de Datos de Carácter Personal**

  Esta ley se complementa con el reglamento estipulado en el Real Decreto RD 1720/2007.

  El objetivo de esta Ley es garantizar y proteger, en lo concerniente al tratamiento de los datos personales (automatizados o no), las libertades públicas y los derechos fundamentales de las personas físicas y, especialmente, de su honor e intimidad personal y familiar.

  Los derechos recogidos en la LOPD son:

  Las personas de las que se almacena datos de carácter personal, tienen una serie de derechos amparados por esta ley:

  - Derecho de información: Cuando alguien proporciona sus datos debe ser informado de que van a ser almacenados.
  - Derecho de acceso, cancelación, rectificación y oposición: La persona puede ver la información que se dispone de él, puede cambiar esos datos

para que sean correctos y exactos, cancelar la información que se almacene de él y oponerse a que se almacene.

- **Ley 34/2002 de servicios de la sociedad de la información y de comercio electrónico (LSSI)**

  Esta Ley se encarga de regular las obligaciones de los prestadores de servicios y los servicios que prestan. Entre las obligaciones que estipula la Ley están:

  - Los prestadores de servicios deben facilitar sus datos de contacto.
  - Deben colaborar con las autoridades, reteniendo los datos de conexión y tráfico durante 12 meses.

  Los que albergan datos proporcionados por un cliente, no serán responsables por la información almacenada a petición del destinatario, siempre que:

  - No tengan conocimiento efectivo de que la actividad o la información almacenada es ilícita o de que lesiona bienes o derechos de un tercero susceptibles de indemnización, o
  - Si lo tienen, actúen con diligencia para retirar los datos o hacer imposible el acceso a ellos.

  Cuando transmitan información de terceros, los proveedores de servicio no tendrán responsabilidad al respecto si:

  - No modifican la información.
  - Permiten el acceso a ella sólo a los destinatarios autorizados.
  - Actualizan correctamente la información.
  - No utilizan su posición con el fin de obtener datos sobre la utilización de la información.
  - Retiran la información que hayan almacenado o hacen imposible el acceso a ella, en cuanto sepan que ha sido retirada del lugar de la red en que se encontraba, o que un tribunal u órgano administrativo competente ha ordenado retirarla o impedir que se acceda a ella.

- **R.D.L, 1/1996 Ley de Propiedad Intelectual**

  La propiedad intelectual de una obra literaria, artística o científica corresponde al autor y le da la plena disposición y el derecho exclusivo a la explotación de la obra. Las obras pueden estar expresadas en cualquier medio o soporte, tangible o intangible, actualmente conocido o que se invente en el futuro como:

  - Los libros, folletos, impresos, epistolarios, escritos, discursos y alocuciones, conferencias, informes forenses, etc.
  - Los proyectos, planos, maquetas y diseños de obras arquitectónicas y de ingeniería.
  - Los gráficos, mapas y diseños relativos a la topografía, la geografía y, en general, a la ciencia.
  - Las obras fotográficas.
  - Los programas de ordenador.

  Al amparo de esta Ley, las organizaciones protegen su conocimiento y las obliga a respetar el de las demás. El otro punto relevante en el ámbito de la seguridad de la información es la obligación de contar únicamente con software original (propietario o libre), ya que la utilización de software sin licencia sería una infracción de la Ley.

- **Ley 11/2007, de acceso electrónico de los ciudadanos a los Servicios Públicos**

  Los puntos más destacables de la Ley son:

  - Los ciudadanos verán reconocidos nuevos derechos en sus relaciones con las administraciones públicas.
  - Se creará la figura del Defensor del Usuario.
  - Los trámites y gestiones podrán hacerse desde cualquier lugar, en cualquier momento.
  - La administración será más fácil, más ágil y más eficaz.
  - Los ciudadanos pasan a tomar la iniciativa en sus relaciones con la administración.

  Contará con un Esquema Nacional de Seguridad y otro de Interoperabilidad, para que los servicios ofrecidos cuenten con un mínimo nivel de seguridad y las distintas administraciones puedan comunicarse con fluidez.

- **Reforma Código Penal (5/2010)**

  Con la publicación en el *BOE de la Ley Orgánica 5/2010*, de 22 de junio, se reformaba la *Ley Orgánica 10/1995* que aprobó el vigente *Código Penal*. En la reforma se incluyen modificaciones sobre artículos en los que se recogen nuevas tipificaciones penales para delitos referentes o cometidos con dispositivos electrónicos, incluyendo además una modificación en la denominación de los *Delitos Informáticos:*

  > *"En el marco de los denominados delitos informáticos, para cumplimentar la Decisión Marco 2005/222/JAI, de 24 de febrero de 2005, relativa a los ataques contra los sistemas de información, se ha resuelto incardinar las conductas punibles en dos apartados diferentes, al tratarse de bienes jurídicos diversos. El primero, relativo a los daños, donde quedarían incluidas las consistentes en dañar, deteriorar, alterar, suprimir o hacer inaccesibles datos o programas informáticos ajenos, así como obstaculizar o interrumpir el funcionamiento de un sistema informático ajeno. El segundo apartado se refiere al descubrimiento y revelación de secretos, donde estaría comprendido el acceso sin autorización vulnerando las medidas de seguridad a datos o programas informáticos contenidos en un sistema o en parte del mismo."*

  A continuación se describen algunas de las principales reformas aplicables introducidas:

  - ***Artículo 270:*** en relación con la Propiedad Intelectual, la novedad del texto radica en la decisión de rebajar las penas para las infracciones a pequeña escala de la misma (p.e. el conocido como top manta).
  - ***Artículo 197.3 (nuevo):*** Se tipifica la intromisión en sistemas ajenos de manera clara y expresa completando lo que ya se preveía para el apoderamiento de documentación del 197.2, lo que sucede es que aquí se castiga la mera entrada en el sistema, se produzcan o no daños o lesiones a los derechos del propietario del sistema.
  - ***Artículo 248.2:*** Este añadido penaliza la programación de aplicaciones que permitan la comisión de estafas, así como su difusión e incluso su mera tenencia, aun cuando la aplicación no haya sido utilizada. Eso sí, los programas deben reunir el requisito de estar específicamente destinados a tal fin, en la medida en que sirvan para otros fines o usos la conducta no será punible, al igual que sucede con los chips de las consolas, por ejemplo.

- ***Artículo 264:*** Este delito castiga tanto la destrucción de información como los ataques que impiden que un sistema pueda funcionar correctamente, tipo *DoS*.

5.1.1.2. Legislación Europea

En materia de seguridad informática existen dos normativas legales relevantes:

- La **Directiva 2009/136/CE** modifica la Directiva 2002/22/CE relativa al servicio universal y los derechos de los usuarios en relación con las redes y los servicios de comunicaciones electrónicas, la Directiva 2002/58/CE relativa al tratamiento de los datos personales y a la protección de la intimidad en el sector de las comunicaciones electrónicas y el Reglamento (CE) nº 2006/2004 sobre la cooperación en materia de protección de los consumidores.
- La **Directiva 2009/140/CE** modifica la Directiva 2002/21/CE relativa a un marco regulador común de las redes y los servicios de comunicaciones electrónicas, la Directiva 2002/19/CE relativa al acceso a las redes de comunicaciones electrónicas y recursos asociados, y a su interconexión, y la Directiva 2002/20/CE relativa a la autorización de redes y servicios de comunicaciones electrónicas.

5.1.1.3. Estándares internacionales

La serie de normas ISO/IEC 27000 son estándares de seguridad publicados por la *Organización Internacional para la Estandarización* (*ISO*) y la *Comisión Electrotécnica Internacional* (*IEC*).

La serie contiene las mejores prácticas recomendadas en Seguridad de la información para desarrollar, implementar y mantener Especificaciones para los Sistemas de Gestión de la Seguridad de la Información (SGSI). La mayoría de estas normas se encuentran en preparación e incluyen:

Familia de normas 27000	
Norma ISO/IEC	**Título**
ISO 27000	Gestión de la Seguridad de la Información: Fundamentos y vocabulario.
ISO 27001	Especificaciones para un **SGSI**.
ISO 27002	Código de Buenas Prácticas.
ISO 27003	Guía de Implantación de un **SGSI**.
ISO 27004	Sistema de Métricas e Indicadores.
ISO 27005	Guía de Análisis y Gestión de Riesgos.
ISO 27006	Especificaciones para Organismos Certificadores de **SGSI**.
ISO 27007	Guía para auditar un **SGSI**.
ISO 2701X	Guías sectoriales.
ISO 27XXX	Futuras normas.

Figura 28 – Familia normas ISO 27000

De entre todas las normas de la familia ISO 27000 cabe mencionar especialmente las siguientes:

- ***UNE-ISO 27001.*** Esta norma es la definición de los procesos de gestión de la seguridad, por lo tanto, es una especificación para un SGSI y, en este momento, es la única norma Certificable, dentro de la familia ISO 27000.
- ***ISO 27002.*** Previamente BS 7799 Parte 1 y la norma ISO/IEC 17799, es un código de buenas prácticas en el que se recoge un catálogo de los controles de seguridad y una guía para la implantación de un SGSI.

5.2. Condiciones de Materiales y Equipos[3]

En cuanto a condiciones materiales, todo el trabajo puede ser desarrollado utilizando un ordenador sin requerimientos hardware especiales y lo bueno de ser una aplicación web realizada en *Java* junto con base de datos *MySQL* es que permite su desarrollo bajo cualquier plataforma.

Atendiendo a las restricciones dadas por los fabricantes en sus páginas Web se van a establecer una serie de requerimientos hardware mínimos, en función del sistema operativo:

		Windows	**Linux**	**Mac OSX**	**Solaris**
Java JRE 1.7	**Versión**	▪ Windows 8 (*escritorio*) ▪ Windows 7 ▪ W Vista SP2 ▪ W Server 2008 ▪ W Server 2012 (64 bits) ▪ Windows XP (*no soportada 04/2014*)	▪ Linux 5.5+ ▪ Linux 6.x ▪ Linux 7.x (64b) ▪ Red Hat Linux 5.5+, 6.x ▪ Ubuntu Linux 10.04 y superior ▪ SUSE Linux Enterprise Server 10 SP2, 11.x	▪ OS X 10.7.3+ ▪ OS X 10.8.3+ ▪ OS X 10.9+	▪ Solaris 11 ▪ Solaris 11 Express ▪ Solaris 10 Update 9+
	Arquitectura CPU	32 bits 64 bits	64 bits	64 bits	64 bits 32 bits (*11 Expr, 10u9+*)
	Procesador	Mínimo Pentium 2 a 266 MHz	--	Mac con Intel	
	Memoria	128 MB	--	2 GB (mín.Mac)	
	Espacio en disco	124 MB JRE 2 MB Java Update	--		
	Explorador Web	▪ IExplorer 7 o sup. ▪ Firefox 3.6 o sup ▪ Chrome	▪ Firefox 3.6 o sup ▪ Chrome	▪ Safari 5.1.3 o sup ▪ Firefox ▪ Chrome	▪ Firefox 3.6 o sup

[3] Requerimientos Hardware. *Fuente*: http://www.oracle.com/technetwork/java/javase/config-417990.html
http://www.mysql.com/support/supportedplatforms/database.html
http://www.oracle.com/technetwork/java/javaee/documentation/javaee7sdk-readme-1957703.html

MySQL	**Versión**	▪ Windows 8 ▪ Windows 7 ▪ Windows Vista ▪ W Server 2008 ▪ W Server 2012 ▪ W Server 2003 ▪ Windows XP	▪ Linux 4, 5, 6.x, 7.x ▪ Red Hat Enterp. Linux 5, 6.x ▪ Ubuntu Linux 12,14 ▪ SUSE Linux Enterp. 10, 11.x ▪ Debian Linux 6x, 5x	▪ OS X 10.6 + ▪ OS X 10.5 *Versión 5.6:* ▪ OS X 10.10 ▪ OS X 10.9 ▪ OS X 10.8 ▪ OS X 10.7 ▪ OS X 10.6 (*no soportada*)	▪ Solaris 11 ▪ Solaris 10 update 8+ (32b)
	Memoria*	576 MB	576 MB	576 MB	576 MB
	Disco*	*	*	*	*
Tomcat (J2EE)	**Memoria**	2 GB	1 GB	1 GB	1 GB
	Disco	500 MB	500 MB	500 MB	500 MB

Tabla 26 – Requerimientos mínimos hardware

* *El tamaño requerido para memoria y espacio en disco necesario para el servidor MySQL dependen en gran medida del volumen de datos con los que trabaje la aplicación. Respecto al espacio en disco, los datos facilitados son los correspondientes al requerimiento para instalación y despliegue de la base de datos del proyecto.*

De manera opcional, al ser una aplicación Web, puede configurarse de modo distribuido, es decir, en distintas máquinas físicas, conectadas entre sí, cuyo único requerimiento es que exista una conexión de red (*LAN* por ejemplo) entre ellas.

Del mismo modo, los usuarios de la aplicación deben conectarse a ella vía peticiones http por medio de un explorador Web, por tanto no hay necesidad de estar en la misma máquina física, igualmente necesitan estar bajo la misma subred o bien se debe haber publicado acceso a través de Internet al servidor de aplicaciones.

5.3. Condiciones de Ejecución

Las condiciones para la ejecución de la aplicación son muy simples ya que como se ha comentado anteriormente las herramientas utilizadas son multiplataforma.

Al tratarse de una aplicación Web desarrollada con *Java* puede ser ejecutada bajo cualquier plataforma sin necesidad más que de la existencia de una *Java Virtual Machine* (JVM).

Del mismo modo para los servidores de base de datos y Web, *MySQL* y *Apache-Tomcat* respectivamente, el fabricante ofrece distribuciones compatibles con los sistemas operativos más comercializados. Por lo tanto la condición principal es la instalación y configuración de los servidores, así como el despliegue de la aplicación.

Adicionalmente, desde una perspectiva de usuario, el ordenador desde el que se desee acceder a la aplicación debe disponer de un explorador Web compatible como pueden ser *Google Chrome, Mozilla Firefox, Internet Explorer, Opera, Safari, etc*.

Capítulo 6.

Manual de Instalación

A continuación se detallan los pasos necesarios para la instalación de la aplicación desde 2 perspectivas diferentes, la primera de ellas se trata del despliegue del producto final en un servidor para que pueda ser explotado por usuarios que se conecten a él. La otra, la correspondiente a la configuración del entorno de desarrollo integrado.

6.1. Despliegue del producto

1) Máquina Virtual Java

Uno de los requisitos principales para el despliegue de una aplicación Java en un servidor *Web* es la existencia de una *máquina virtual Java (JVM)*, como se ha mencionado anteriormente encargada de traducir el *bytecode* del lenguaje con el sistema en el que se quiere ejecutar.

El primer paso es acceder a la página Web del lenguaje, a la sección de descargas (http://www.java.com/es/download/manual.jsp) para poder descargar el archivo instalable correspondiente al software de *Java*. En esta página se permite al usuario seleccionar la versión del Sistema Operativo destino así como la versión del producto, en nuestro caso *v8 u65*.

Una vez descargado el paquete, para proceder a su instalación es suficiente con seguir las instrucciones del proceso de instalación en el que habrá que aceptar el acuerdo de licencia y las condiciones y posteriormente se muestra una pantalla donde poder establecer el directorio de instalación:

Figura 29 – Instalación producto. Paso 1: *JVM* selección directorio

Pulsando el botón *Siguiente* se inicia el proceso de instalación, que una vez finalizado muestra un mensaje de confirmación de que se ha realizado correctamente.

Figura 30 – Instalación producto. Paso 2: *JVM* Resumen instalación

2) Instalación servidor aplicaciones (*Apache-Tomcat*)

También debe haber sido descargado el ejecutable desde la página del fabricante (http://tomcat.apache.org/), en este caso la versión 7.0.55. Esta versión de *Apache Tomcat* soporta las especificaciones de las tecnologías *Java Servlet* y *JavaServer Pages* (JSP) en sus versiones 3.0 y 2.2 respectivamente.

De entre las distintas opciones permitidas para Windows, escogemos la del instalador de servicio Windows (*"Windows Service Installer"*).

En primer lugar, se muestra la pantalla de bienvenida, en la que se ofrece un mensaje al usuario con recomendaciones e información del programa que se va a instalar, a continuación es necesario aceptar las condiciones de licenciamiento para poder continuar.

La siguiente pantalla que se muestra corresponde a la selección del tipo de Instalación, ya que el usuario puede querer una instalación completa del servidor y todos los complementos, tales como la documentación, o bien seleccionar únicamente aquellos que necesita. En este caso, la opción *"Normal"* es la más apropiada, manteniendo las opciones seleccionadas por defecto, que además del servicio, incluye la documentación y el gestor de aplicaciones.

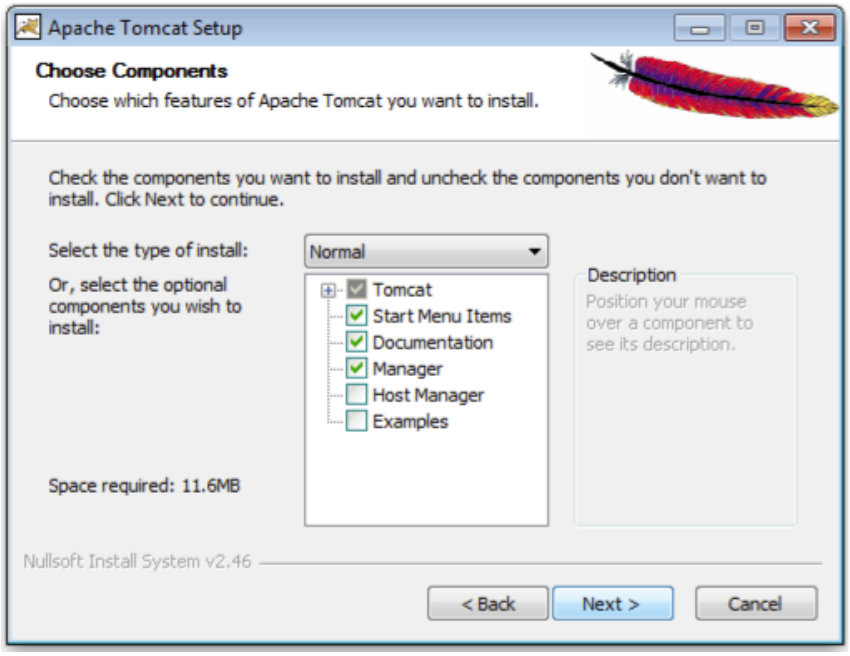


Figura 31 – Instalación producto. Paso 3: *Apache Tomcat* – Selección tipo instalación

A continuación, hay que especificar las opciones de configuración del servidor Web, es decir, puerto en el que recibirá las peticiones *http* en su versión 1.1 (8080 por defecto), usuario/contraseña del administrador y nombre para el servicio *Windows*.

Como se ha mencionado anteriormente, *Apache Tomcat* depende directamente de un compilador java, por ello en el proceso de instalación, se buscará la variable de sistema **JAVA_HOME** que indique donde está la ubicación de la máquina virtual Java que se instaló en el paso anterior. (se trata de la ubicación por defecto: *C:\Program Files\Java\jre7).*

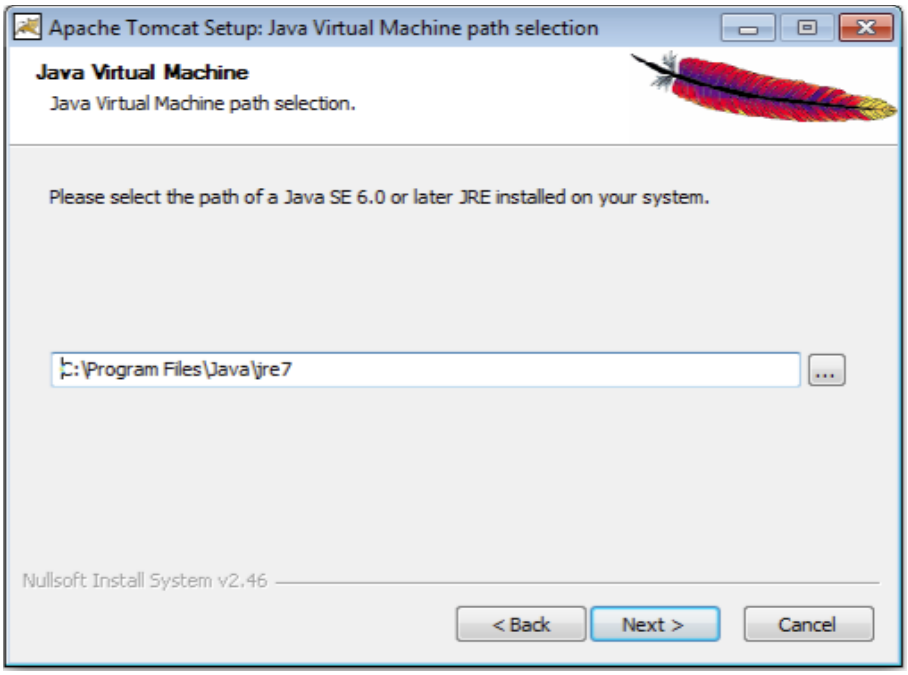


Figura 32 – Instalación producto. Paso 4: *Apache Tomcat* – Selección directorio JVM

En la siguiente pantalla que se muestra, el asistente de instalación da la posibilidad al usuario de modificar el directorio de instalación del programa, así como información del espacio requerido vs espacio disponible en el sistema del usuario.

Posteriormente el asistente inicia la instalación del software, en caso de que no haya problemas durante la instalación se muestra una última pantalla que indica que se ha completado la instalación dando la posibilidad al usuario de ejecutar el servicio.

Figura 33 – Instalación producto. Paso 5: *Apache Tomcat* – Resumen Instalación

Tras pulsar el botón *"Finish"* se ejecuta el servicio de arranque del servidor.

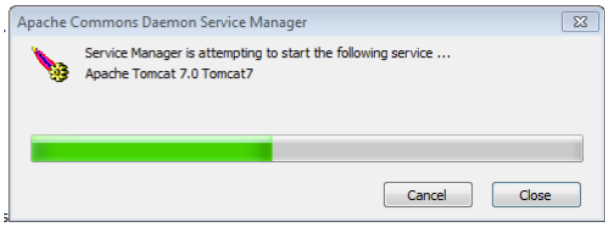


Figura 34 – Instalación producto. Paso 6: *Apache Tomcat* – Arranque Servicio

Una vez desaparece la barra de progreso correspondiente al estado del arranque del servicio, podemos comprobar desde cualquier cliente, que en el ámbito Web son los navegadores (*Mozilla, Google Chrome, IExplorer*, etc.), que el servidor se ha instalado correctamente, y por tanto está arrancado y escuchando, a la espera de peticiones, en el puerto indicado en la instalación (8080).

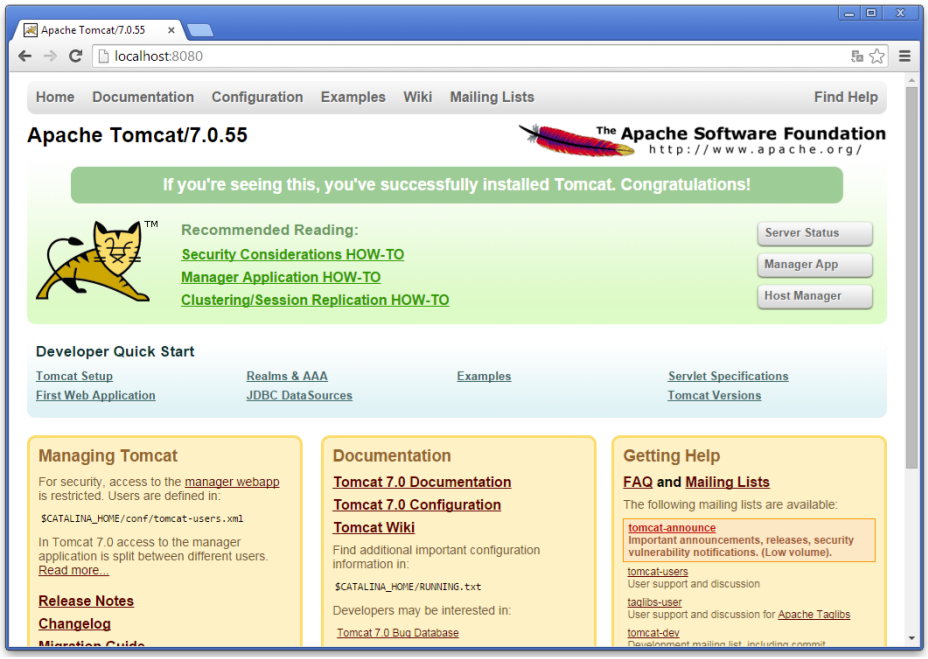


Figura 35 – Instalación producto. Paso 7: *Apache Tomcat* – Verificación

3) Despliegue aplicación

Apache Tomcat incorpora ciertas utilidades que permiten la gestión y administración de aplicaciones desde una funcionalidad denominada *Manager App*. Algunas de las opciones más interesantes que proporciona son:

- Desplegar aplicaciones desde un fichero *WAR* o desde una ruta del sistema de ficheros del servidor.
- Listar las aplicaciones web desplegadas, así como las sesiones que están activas para cada una de esas aplicaciones.
- Parar o detener una aplicación.
- Parar una aplicación pero no replegarla (eliminarla del servidor).

Para poder acceder al mismo, se puede seleccionar el enlace que se proporciona en la página principal del servidor web, o bien escribir directamente la dirección http://localhost:8080/manager/html en el navegador. Una vez acreditado el acceso (con el usuario y contraseña establecidos en la instalación) se muestra la siguiente vista:

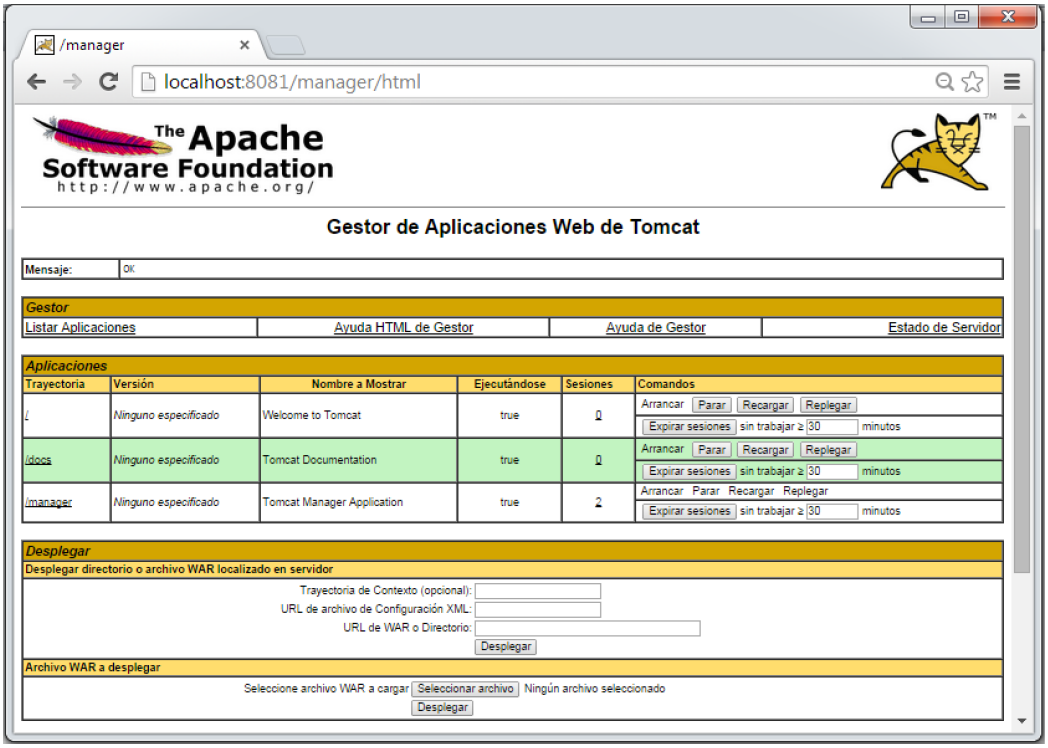


Figura 36 – Instalación producto. Paso 8: *Manager App* – Página Principal

Para poder realizar el despliegue de una aplicación web es necesario disponer de un archivo con extensión *WAR* (*Web Archive*), esto es, un fichero comprimido que contiene todos los elementos que juntos componen una aplicación web: fuentes compilados (*.class*), *JSP's*, *XML*, librerías de etiquetas (*Tags*), páginas web estáticas (en formato *HTML*, *XHTML*), Archivos *JavaScript*, *CSS's* , etc.

Una vez generado el archivo, o bien utilizando el archivo que se facilita adjunto a la presente memoria, se puede realizar el despliegue. En la parte inferior de la pantalla principal desde la opción "*Seleccionar un archivo*", se permite seleccionar un *WAR* disponible en el sistema de archivos y posteriormente pulsar el botón *"Desplegar"*.

Si no ocurre ningún problema se debe recargar la pantalla incluyendo la nueva aplicación desplegada al final del listado, tal y como se muestra a continuación:

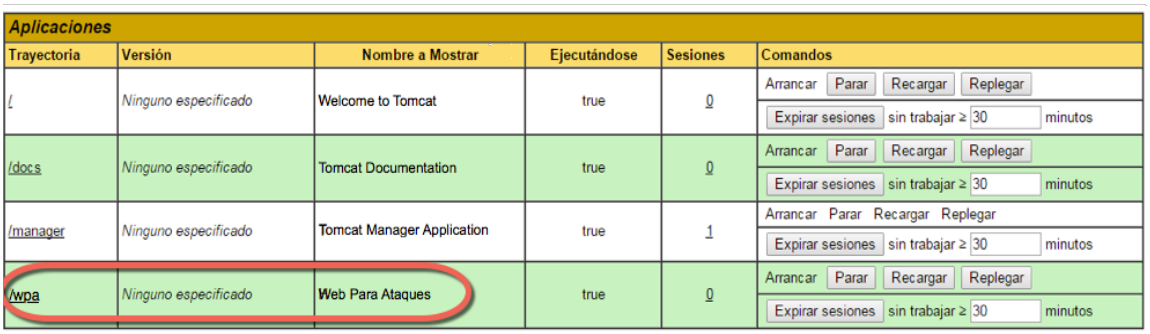


Figura 37 – Instalación producto. Paso 9: *Manager App* – Aplicación desplegada

4) Instalación Sistema Gestor de Bases de Datos (SGBD) – MySQL

Al igual que es necesario instalar en el servidor de aplicaciones el código fuente correspondiente a la aplicación, también lo es instalar la base de datos que sirva como repositorio de toda la información generada en ella.

Como se ha mencionado anteriormente, el sistema de bases de datos seleccionado para tal efecto es *MySQL*, en su versión 5.5. Para realizar la instalación el primer paso es descargar el software de la página web del fabricante (http://dev.mysql.com/downloads/mysql/5.5.html#downloads) y seleccionar el Sistema Operativo y el formato, en el caso de *Windows*, *MSI Installer* o *Zip*.

Seleccionando la opción *MSI Installer*, una vez descargado el archivo se debe proceder a la instalación siguiendo las instrucciones del asistente. El proceso de instalación es muy simple y prácticamente no requiere intervención por parte del usuario.

Una vez aceptados los términos y la licencia GNU, el siguiente paso es seleccionar el tipo de instalación, en este caso la recomendada por el fabricante para la mayoría de usuarios es "*Typical*".

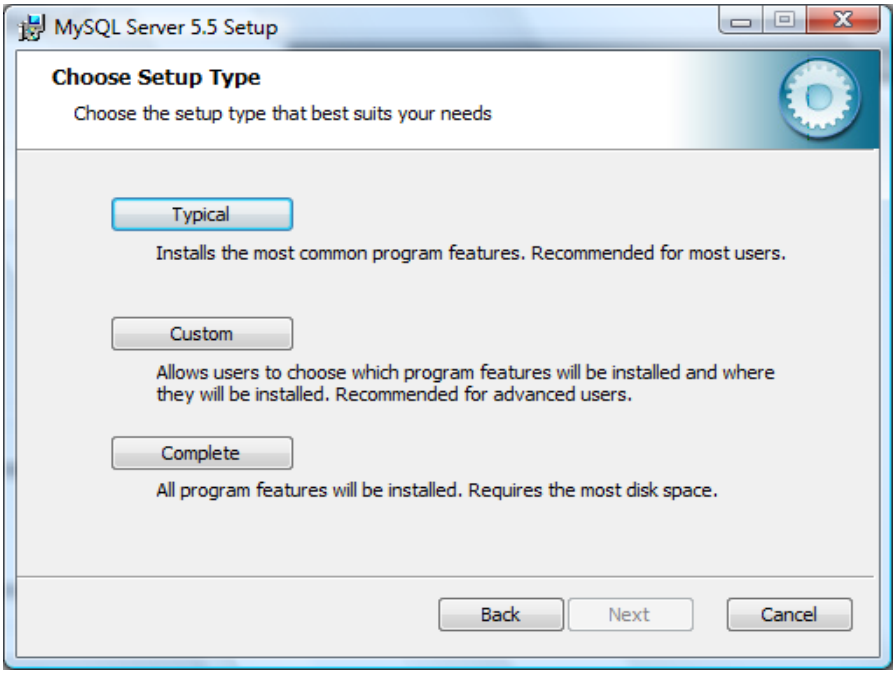


Figura 38 – Instalación producto. Paso 10: Instalación *MySQL* – Selección Tipo

Siguiendo las instrucciones del asistente se realizará la instalación del software, cuando ésta haya finalizado debe mostrarse al usuario una pantalla de confirmación, en la que por defecto estará seleccionada la opción "*Launch the MySQL Instance Configuration Wizard*" mediante la cual se permite continuar con el proceso de configuración del servidor.

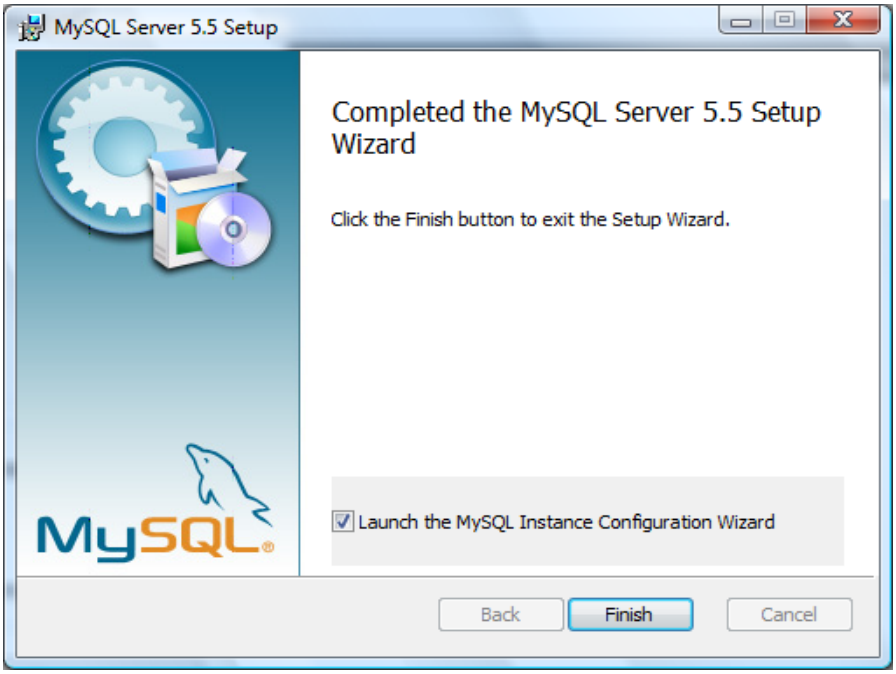


Figura 39 – Instalación producto. Paso 10: Instalación *MySQL* – Mensaje confirmación

Una vez instalado *MySQL*, la siguiente fase es la configuración del servidor en sí mismo, de nuevo utilizando el asistente de configuración es muy sencillo.

En primer lugar es necesario seleccionar el tipo de configuración a aplicar, en este caso *Detailed Configuration* que permite al usuario optimizar la configuración del servidor *MySQL*.

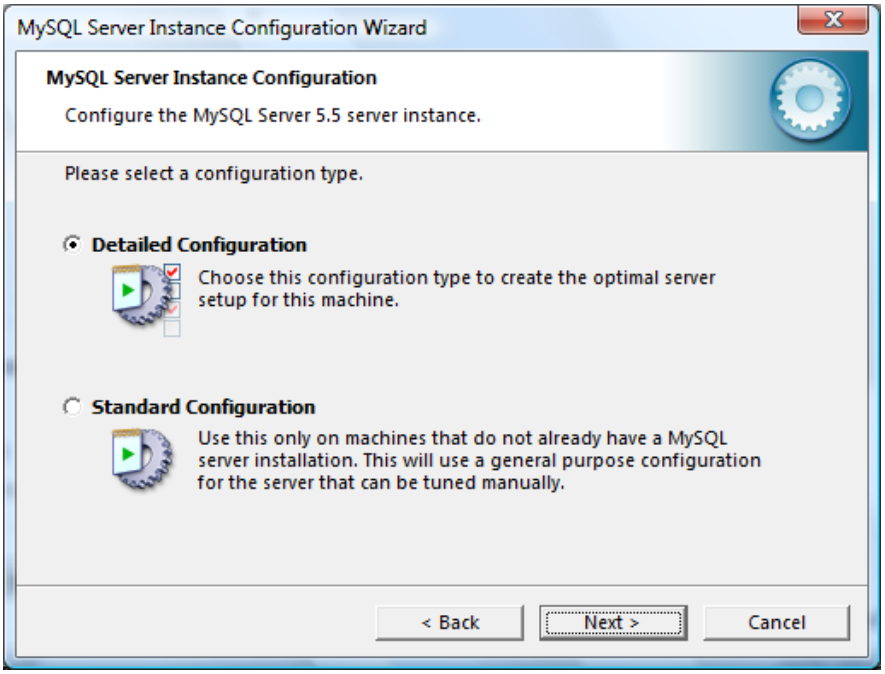


Figura 40 – Instalación producto. Paso 11: Configuración *MySQL*

A continuación se puede configurar el tipo de instancia del servidor, dependiendo del uso que se le vaya a dar es necesario elegir una opción u otra, cada una con sus propios requerimientos de memoria. La opción *Developer Machine*, para desarrolladores, la más

apta para un uso de propósito general y la que menos recursos consume. Para compartir servicios en la máquina, probablemente *Server Machine* sea la mejor elección o, si va a estar dedicada exclusivamente como servidor SQL, se puede optar por *Dedicated MySQL Server Machine*, que asignará la totalidad de los recursos a esta función.

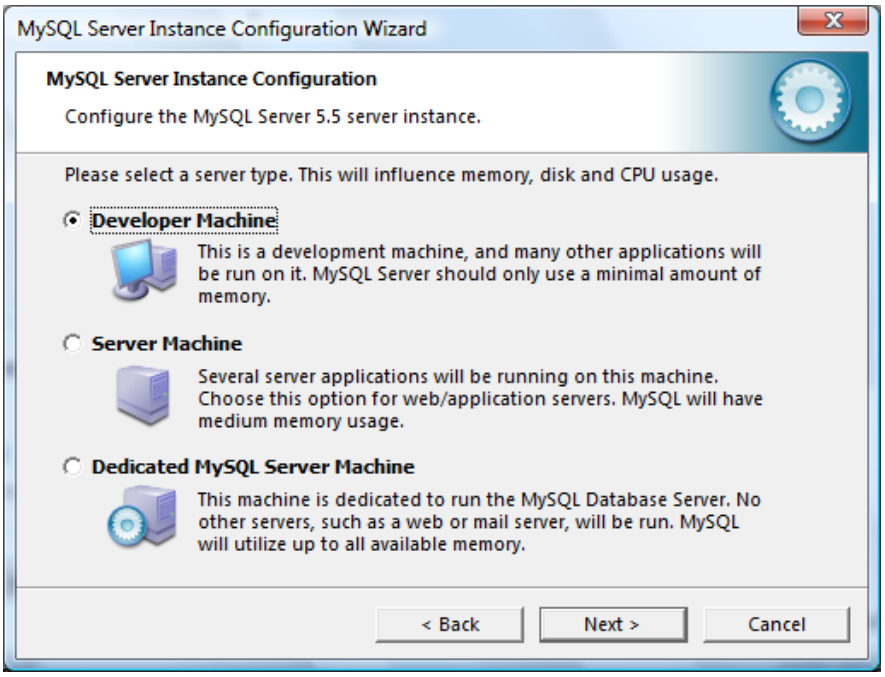


Figura 41 – Instalación producto. Paso 12: Configuración *MySQL*. Tipo de servidor

De nuevo, para un uso de propósito general, se recomienda la opción por defecto, *Multifunctional Database*.

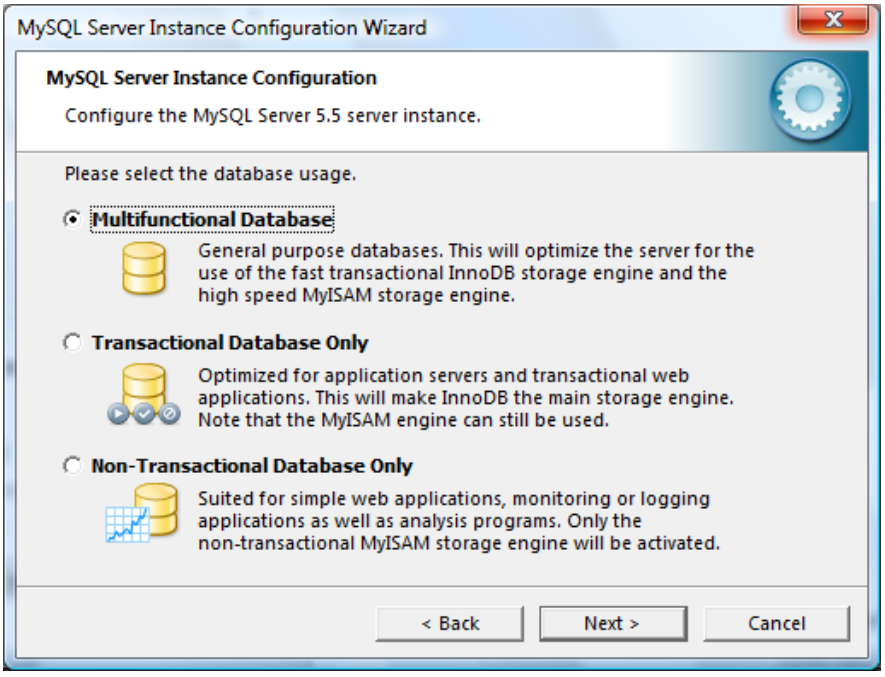


Figura 42 – Instalación producto. Paso 13: Configuración *MySQL*. Tipo de base de datos

En la siguiente pantalla se puede seleccionar el directorio para la instalación de *InnoDB*, se trata del motor subyacente que dota de toda la potencia y seguridad a *MySQL*. Su

funcionamiento requiere de unas tablas e índices cuya ubicación se puede configurar, en este caso se selecciona la opción por defecto.

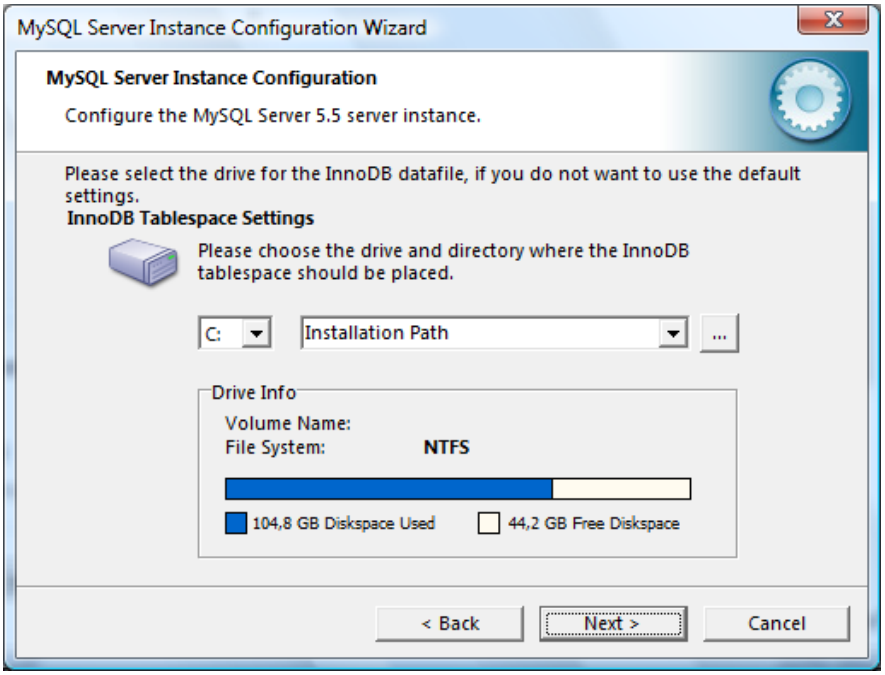


Figura 43 – Instalación producto. Paso 14: Configuración *MySQL*. Directorio *InnoDB*

En el siguiente paso se permite optimizar el funcionamiento del servidor, en previsión del número de usos concurrentes. La opción por defecto, *Decision Support (DSS) / OLAP* será probablemente la que más convenga en este caso, ya que según el fabricante se asumen 20 conexiones concurrentes.

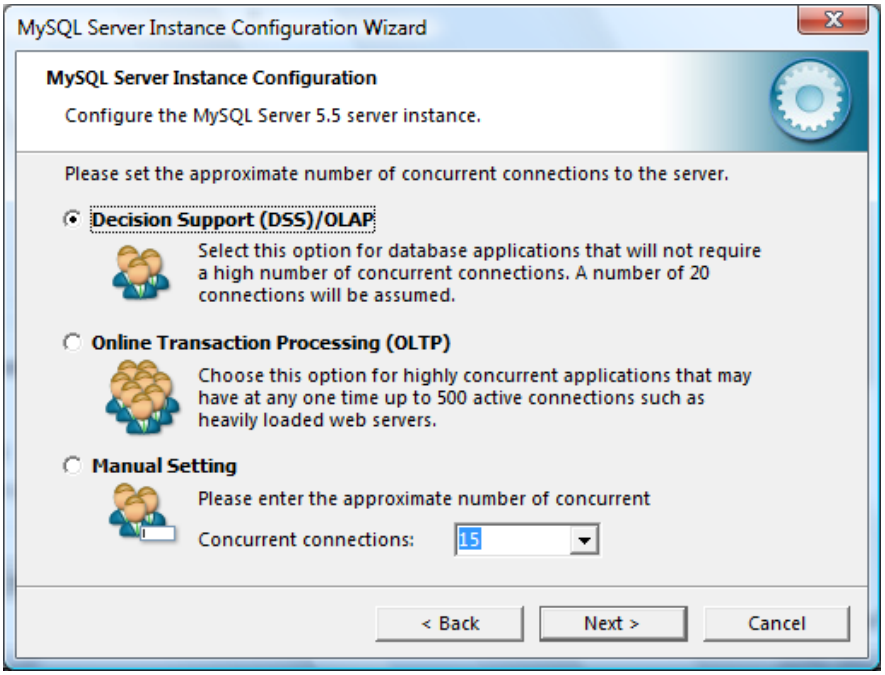


Figura 44 – Instalación producto. Paso 15: Configuración *MySQL*. Conexiones al servidor

En la siguiente pantalla y sucesivas se permite seleccionar el juego de caracteres (*Latin1, Unicode, etc.*) a utilizar, así como las opciones de conexión.

Dejar estas opciones tal como vienen por defecto es la opción más adecuada para un uso de propósito general o de aprendizaje. Aceptar conexiones TCP permitirá conectarse al servidor desde otras máquinas (o desde la misma, simulando un acceso web típico).

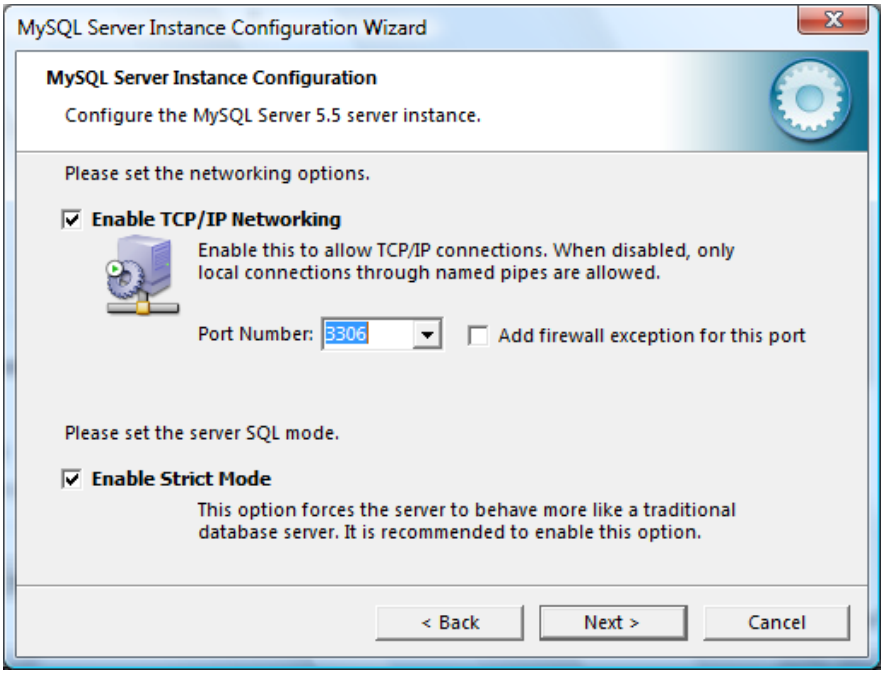


Figura 45 – Instalación producto. Paso 16: Configuración *MySQL*. Conectividad

Instalar *MySQL* como un servicio de *Windows* es la opción más limpia, pudiendo elegir si el motor de la base de datos debe arrancar por defecto o hacerlo manualmente.

Además, se debe seleccionar la 2ª opción para que los ejecutables estén en la variable *PATH*, para poder invocar a *MySQL* desde cualquier lugar en la línea de comandos.

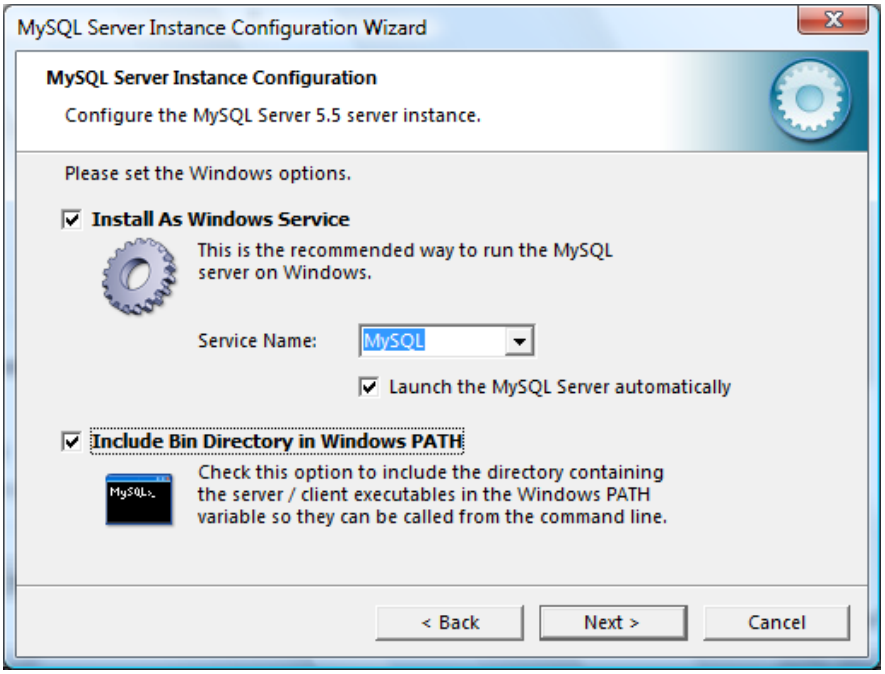


Figura 46 – Instalación producto. Paso 17: Configuración *MySQL*. Opciones de Windows

El último paso para proceder a la modificación del fichero de configuración es asignar una contraseña al usuario *root*. Esto siempre es lo más seguro, se puede indicar que el éste pueda acceder desde una máquina diferente a esta, aunque eso no sea una buena práctica de seguridad.

En este caso, para que funcionen correctamente las conexiones desde la aplicación Web, y las diferentes características respecto a vulnerabilidad que la aplicación debe tener, no se ha establecido contraseña para el usuario.

Una vez pulsado el botón *"Ejecutar"*, se genera el fichero de instalación y se arranca el servicio, mostrando una vez más un mensaje de confirmación al usuario.

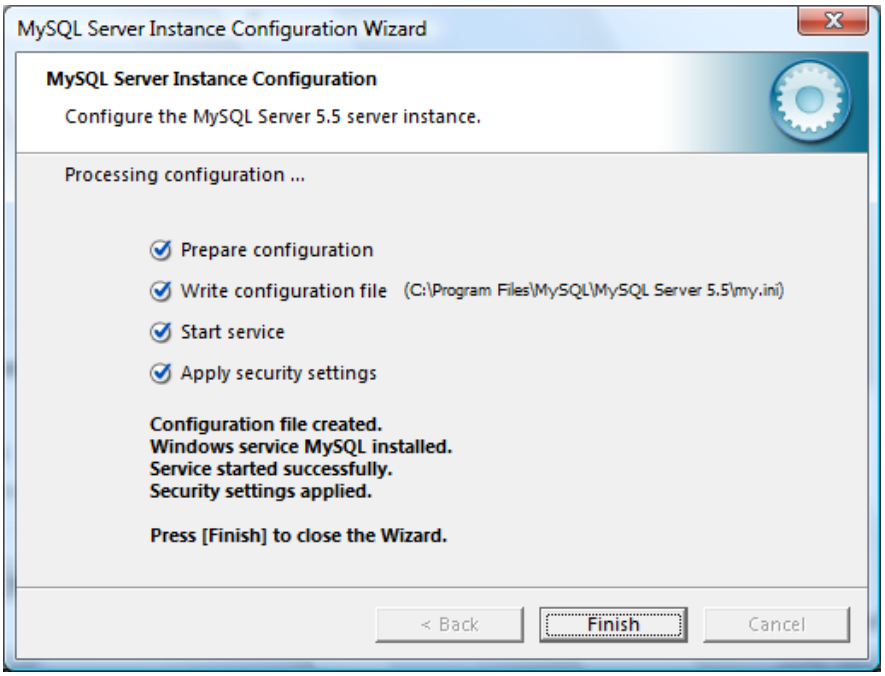


Figura 47 – Instalación producto. Paso 18: Configuración *MySQL*. Mensaje confirmación

5) Instalación de la base de datos

Una vez instalado el motor de base de datos el último paso que hay que hacer es crear la estructura completa de la base de datos (tablas, índices, restricciones de integridad, etc) y cargar las tablas maestras con la información necesaria para garantizar el correcto funcionamiento de la aplicación.

Para la conexión al servidor de base de datos hay muchas herramientas gráficas, tales como *NAVICAT* o *phpMyAdmin,* etc., para los cuales hay muchos tutoriales en la Web que detallan su instalación estándar.

A continuación se van a detallar las instrucciones para la configuración de la base de datos por medio de la línea de comandos.

MySQL tiene un programa, que se llama con el mismo nombre de la base de datos (*mysql*) que sirve para gestionar la base datos por línea de comandos. Ese programa, en una instalación de *Windows* se encuentra en un directorio como *C:\Archivos de programa\MySQL\MySQL Server 4.1\bin*.

El primer paso una vez en el editor de comandos es realizar la conexión, indicando la dirección del servidor, usuario y contraseña para el acceso:

```
> mysql -h localhost -u root
```

(si hubiésemos asignado una contraseña al usuario habría que añadir `-ppassword`*, no dejando espacio en blanco entre -p y la clave)*

Una vez dentro, se disponen de todas las sentencias de *MySQL* para el trabajo con la base de datos y el lenguaje SQL.

A continuación, para crear la base de datos y su estructura asociada se debe ejecutar el siguiente comando, indicando el directorio en el que se encuentra archivo por lotes (.sql) que contiene dicha información (puede encontrarse en el CD adjunto).

```
mysql> [directorio]source crear_bdwpa.sql
```

Una vez finalizada la creación del esquema de la base de datos, para cerrar la conexión con el servidor, simplemente hay que escribir "quit" desde el prompt.

Llegados a este punto puede empezar a ser utilizada la aplicación, accediendo con cualquier explorador Web tal y como se detalla más adelante en el manual de usuario (cap. 7).

6.2. Instalación en entorno de desarrollo

Algunos de los pasos de la configuración del entorno de desarrollo son iguales que en el apartado anterior, es decir, para crear un entorno de desarrollo Java es necesario tener instalada la máquina virtual, y para poder ejecutar la aplicación en local es necesario acceder a una base de datos *MySQL,* por lo que también es necesario instalar el motor de la base de datos y configurarlo como se detalla en el apartado anterior.

El único paso potencialmente omisible del proceso anterior es la instalación y configuración del servidor *Web Apache-Tomcat*, ya que *IDEs* como *eclipse, NetBeans,* etc. normalmente suelen incorporar *plugins* que permiten hacer una instalación integrada del servidor y gestionar las aplicaciones y los despliegues desde el propio *IDE*[4].

En este caso se van a detallar los pasos necesarios para la configuración del proyecto en *eclipse*, el cual puede descargarse en su versión para J2EE (aplicaciones Web) desde la página web https://eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/lunasr2 y seleccionar la versión del Sistema Operativo.

Una vez descargado el archivo comprimido (*ZIP*) basta con descomprimirlo en un directorio local y pulsar el archivo ejecutable que hay en él, *eclipse* no necesita instalación, con lo que se podría empezar a utilizar.

Si bien es necesario descargar los componentes del servidor de aplicaciones, de la página web de *Apache* (http://tomcat.apache.org/download-70.cgi), en este caso en formato *ZIP* es suficiente, y al igual que en el paso anterior, descomprimirlo en algún directorio local.

Para añadir el servidor *Web* en *eclipse* es necesario mostrar la vista “Servidores”, en el menú superior *Window* > *Show View* > *Servers.*

[4] Si bien los *IDEs* incorporan la posibilidad de gestionar el servidor de manera integrada, se puede gestionar el despliegue de la aplicación como se ha descrito en el apartado anterior, generando el *WAR* con un archivo *.bat* o desde el propio *IDE* (eclipse por ejemplo) y desplegándolo manualmente en el servidor instalado.

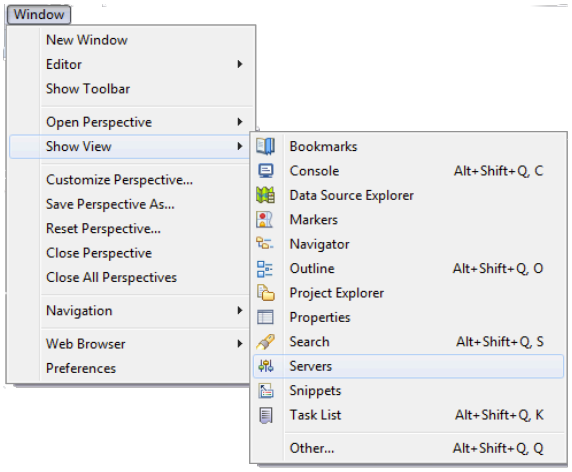


Figura 48 – Instalación entorno desarrollo. Paso 1: Configurar servidor Web

Desde la perspectiva de eclipse en la que esté el usuario se abre en la parte inferior de la pantalla una nueva pestaña "*Servers*", con un enlace informativo de que no hay ninguno creado.

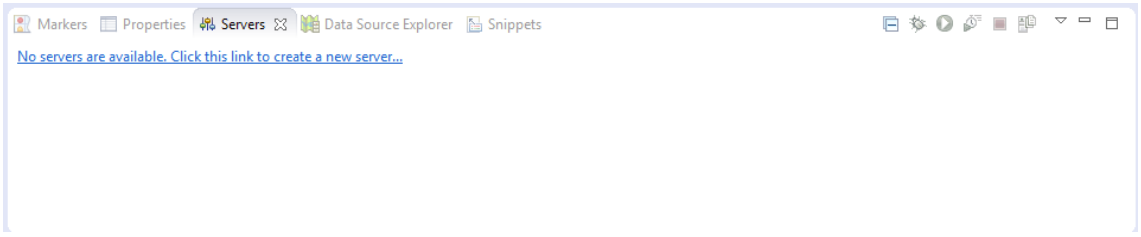


Figura 49 – Instalación entorno desarrollo. Paso 2: Vista de servidores

Al pulsar el enlace aparece una ventana emergente donde se da la posibilidad al usuario de seleccionar la tecnología del servidor de aplicaciones, así como la versión, en este caso *Tomcat 7.0,* dejando las demás opciones de la pantalla con los valores por defecto.

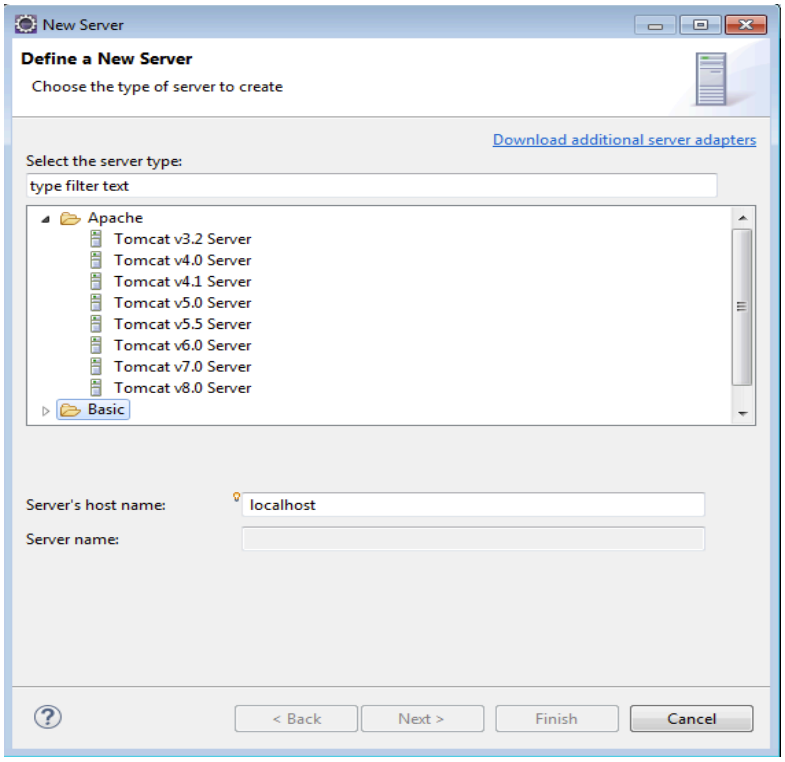


Figura 50 – Instalación entorno desarrollo. Paso 3: Tecnología y versión del servidor

El siguiente paso es indicarle al asistente dónde se encuentra el directorio con los datos del servidor, o bien descargar e instalar uno. Del mismo modo, se debe indicar a qué versión de Java van a corresponder las aplicaciones desplegadas. Se da la posibilidad al usuario de seleccionar cualquiera de las versiones de JVM instaladas en el ordenador.

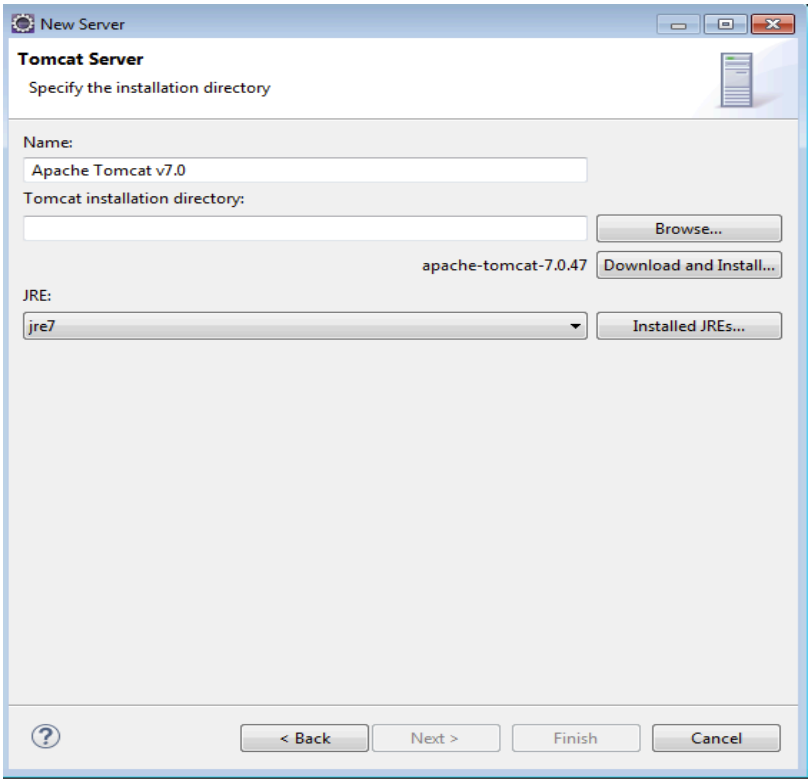


Figura 51 – Instalación entorno de desarrollo. Paso 4: Selección JVM y directorio del servidor

Al pulsar el botón *Finish* se inicia la configuración del servidor, al acabar la instalación, en la pestaña *Servers* inicial ahora aparece el nuevo servidor, y en la parte superior derecha de dicha pestaña las distintas opciones para la gestión de dicho servidor (*arranque, parada, depuración, perfilado*, etc.)

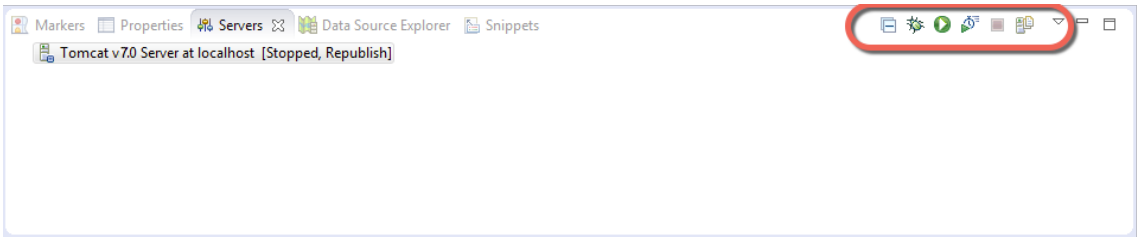


Figura 52 – Instalación entorno de desarrollo. Paso 5. Opciones gestión servidor

A continuación, se incluye el detalle de los pasos necesarios para configurar *eclipse* para poder desplegar el proyecto. Una vez copiados los fuentes del proyecto (pueden encontrarse en el CD adjunto) a un directorio local, éste debe ser importado en el área de trabajo de *eclipse* (*File > Import*).

En la ventana emergente que se muestra hay que seleccionar la opción *General > Existing Projects into workspace*.

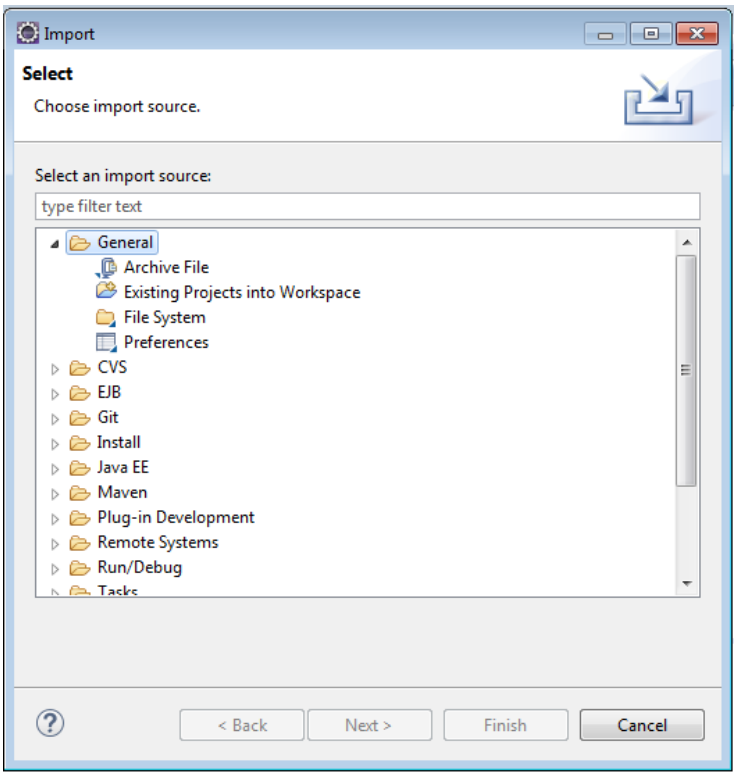


Figura 53 – Instalación entorno de desarrollo. Paso 6. Importación *eclipse*

A continuación el asistente para la importación dará la posibilidad de escoger el directorio en el cual se encuentra el proyecto, el resto de opciones se dejan tal y como aparecen por defecto.

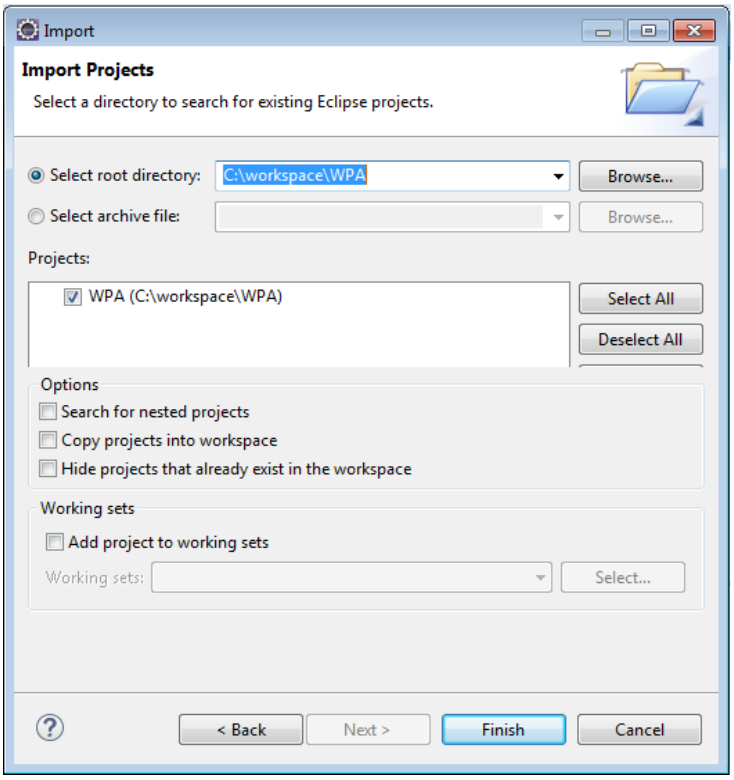


Figura 54 – Instalación entorno de desarrollo. Paso 7. Selección directorio importación

Al pulsar el botón *Finish* el sistema inicia el proceso de importación, al finalizar, en el explorador de proyectos de *eclipse* se muestra un nuevo proyecto con nombre *WPA*.

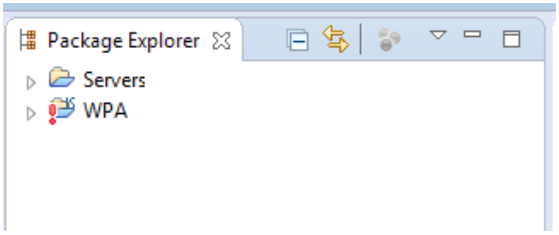


Figura 55 – Instalación entorno de desarrollo. Paso 7. Explorador Proyectos

Si bien es cierto que el proyecto no debería presentar errores de compilación, podrían producirse si la versión del *JRE* no es la misma que la especificada en el presente documento. Los errores de compilación se representan con un signo de admiración '!' de color rojo sobre el icono del proyecto, además, en la pestaña de errores de *eclipse* se incluye un listado con la información sobre los errores producidos.

Para solucionar estos problemas, hay que pulsar con el botón derecho del ratón sobre el nombre del proyecto y en el menú desplegable seleccionar la opción *Properties,* en la ventana emergente que se abre se selecciona la opción "*Java Build Path*".

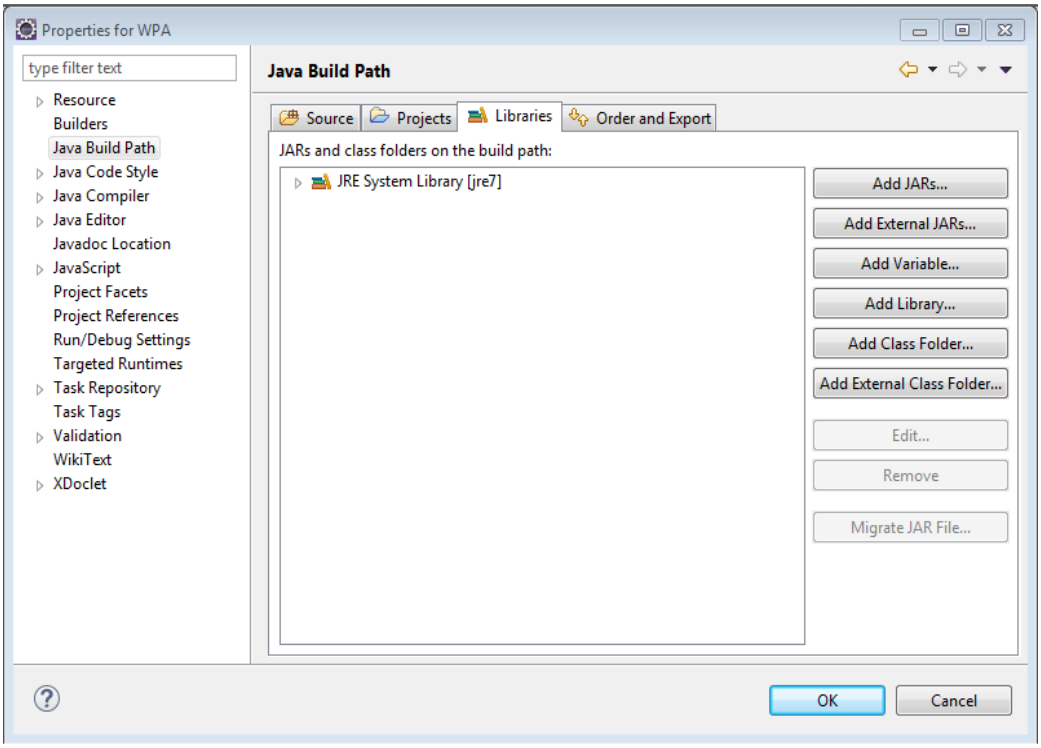


Figura 56 – Instalación entorno de desarrollo. Paso 8. Configuración propiedades

En esta opción se puede modificar la librería Java utilizada (*JRE),* para ello hay que pulsar el botón *"Add Library…"* y seleccionar la opción *JRE System Library* y a continuación seleccionar una de las opciones que se dan al usuario (por defecto, *Workspace default JRE*).

Otra de las acciones que se pueden realizar desde esta pantalla es configurar las librerías (*JAR)* necesarias para la correcta compilación y ejecución del proyecto, dichas librerías están incluidas en el directorio *lib* del proyecto, por lo tanto habrá que seleccionarlos ahí.

Capítulo 7.

Manual de Usuario

En esta sección y las siguientes se incluye un manual que pretende facilitar el uso de la aplicación, familiarizando al usuario con las pantallas que la componen y explicando el propósito de los diferentes componentes y la funcionalidad desarrollada en cada una.

En primer lugar, para acceder a la aplicación únicamente se necesita un explorador web como pueden ser *Internet Explorer, Chorme, Mozilla Firefox, Safari, etc*., una vez abierto, se debe introducir la siguiente URL en la barra de direcciones:

```
http://ip_maquina_servidora:puerto/wpa
```

Dónde:

- ***ip_maquina_servidora***: será la dirección IP de la máquina en la que se realizó la instalación de la aplicación, o en su defecto el nombre o alias de la maquina.
- ***puerto***: puerto de escucha del servidor de aplicaciones en la máquina en la que se instaló la aplicación. Por defecto para el servidor Tomcat este puerto es el 8080.

En el caso de tener la instalación corriendo sobre la maquina local del usuario la *URL* seria:

http://localhost:8080/wpa ó http://127.0.0.1:8080/wpa

En caso de que la URL introducida sea la correcta y que el servidor de aplicaciones esté arrancado y accesible desde la máquina del usuario, se mostrará la pantalla de inicio (*login*) en el navegador, cuya apariencia es similar a la imagen que se muestra a continuación:

Figura 57 – Pantalla de inicio WPA

7.1. Pantalla de Login

Esta pantalla permite a los usuarios introducir sus credenciales para el acceso a la aplicación o bien acceder a la pantalla de registro para realizar el alta de un nuevo usuario con sus datos que le permita el acceso a la aplicación.

Figura 58 – Pantalla login de usuarios

Para el acceso a la aplicación el usuario deberá introducir los datos solicitados en la pantalla, esto es, sus credenciales, compuestas por un alias de usuario y una contraseña, una vez introducidos al pulsar el botón "Entrar". Si el usuario es validado, accederá a la aplicación.

De manera obligatoria, para poder acceder a la aplicación, el usuario deberá introducir texto en los dos campos de la pantalla.

La aplicación realizará esta validación de manera que si no se introducen texto en ambos se mostrará un mensaje de error. En la parte inferior de la pantalla aparecerá el mensaje "Usuario o contraseña erróneos", tal y como se muestra en la siguiente imagen:

Figura 59 – Pantalla de login de usuarios: error validación

Pulsando el botón "Registro Usuarios", accederá a la pantalla de alta de usuarios donde podrá realizar el registro de un usuario con perfil general (remitirse al apartado Perfiles).

- **Descripción Física de la pantalla**
  - *Parte superior*
    Se sitúan los logos de la aplicación en la parte superior derecha, y los logos oficiales de la Universidad de Alcalá y del departamento de ciencias de la computación de la escuela de Politécnica de esta Universidad.
  - *Parte central*
    Dispone de dos campos de texto para introducir el usuario y la contraseña secreta del usuario. Debajo dispone de dos botones "Entrar" y "Registro Usuarios", para realizar diferentes acciones en la pantalla.
  - Correspondencia con caso de uso: *LOGIN USUARIO*

7.2. Página Principal (Home)

A la página que se muestra una vez los usuarios se hayan logado en el sistema se la denomina página principal o *home*. En ella están a disposición de los usuarios las diferentes opciones en función del perfil del usuario que se haya logado:

- ***Principal rol Administrador***

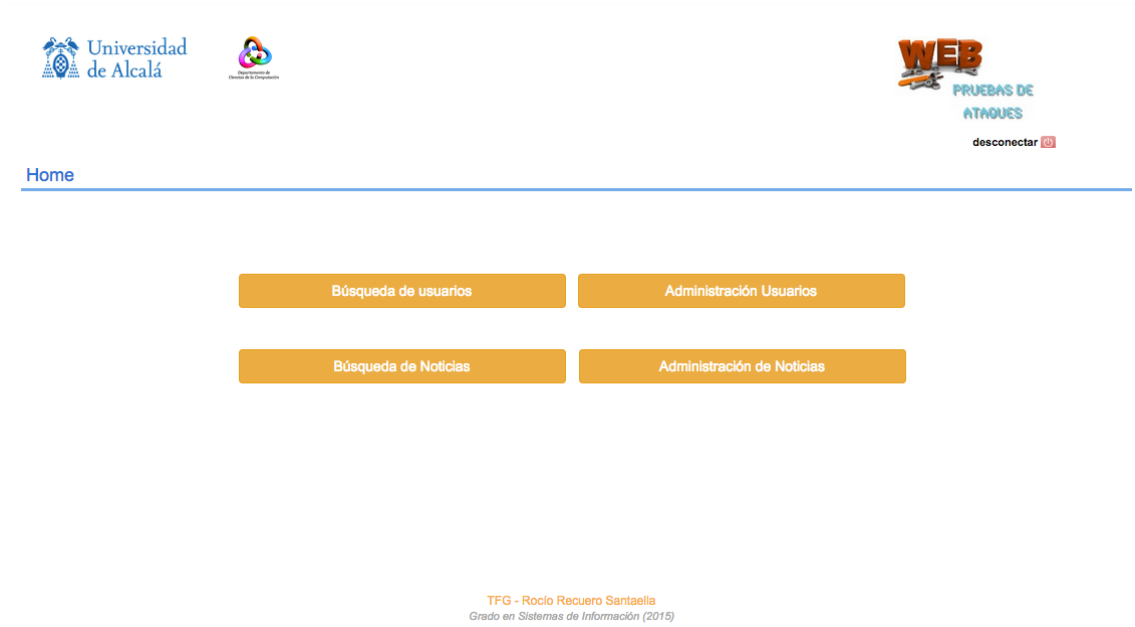


Figura 60 – Página Principal rol *Administrador*

- ***Principal rol General***

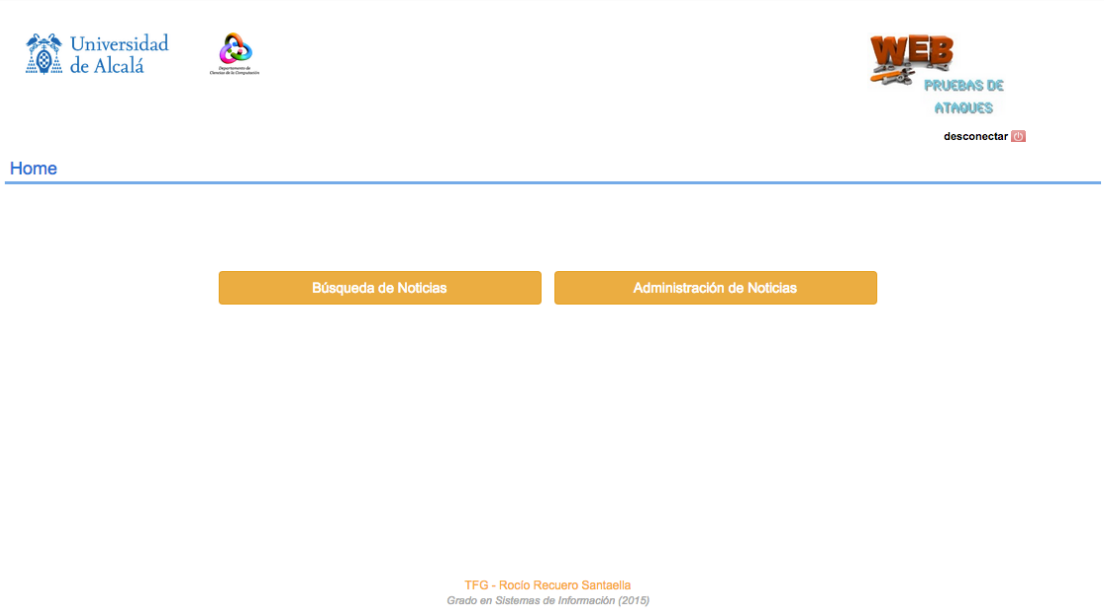


Figura 61 – Página Principal rol *Administrador*

- ***Principal rol Redactor***

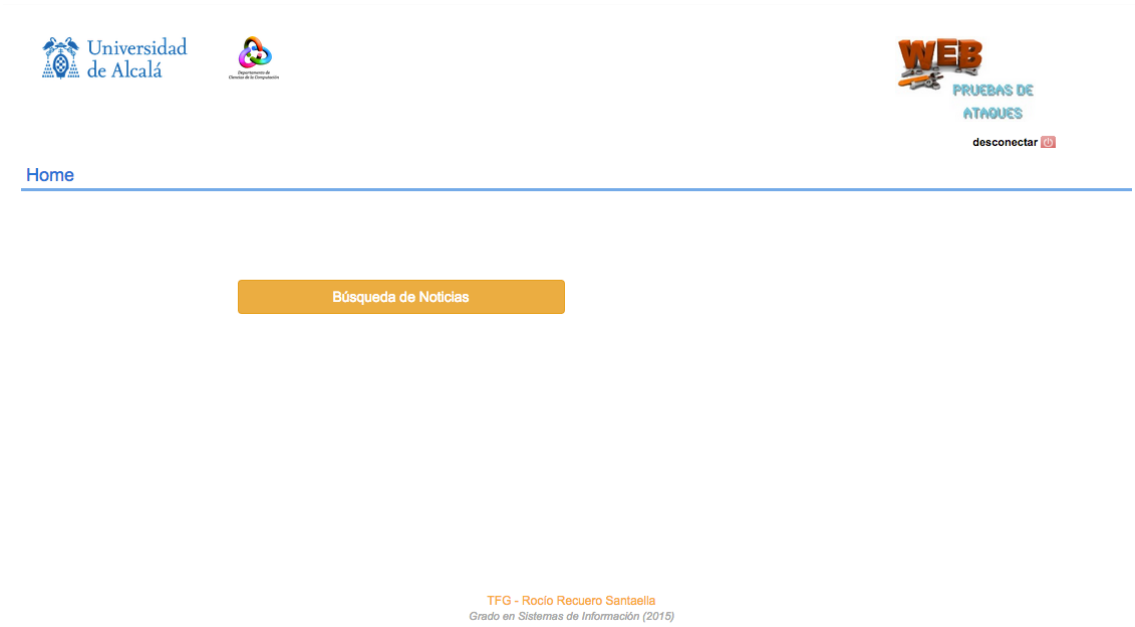


Figura 62 – Página Principal rol *Administrador*

El objetivo principal de esta pantalla es gestionar el acceso de los usuarios a las diferentes funcionalidades que tienen disponibles, a través de cada uno de los elementos que componen el menú:

- ***Búsqueda de Usuarios***: pulsando este botón el usuario accederá a la pantalla de Búsqueda de usuarios. Esta funcionalidad estará disponible únicamente para el perfil administrador.
- ***Administración de Usuarios***: pulsando este botón el usuario accederá a la pantalla de gestión de usuarios. Esta funcionalidad estará disponible únicamente para el perfil administrador.
- ***Búsqueda de Noticias***: pulsando este botón el usuario accederá a la pantalla de Búsqueda de noticias. Esta funcionalidad estará disponible para el perfil administrador, para el perfil Redactor y para el perfil general.
- ***Administración de Noticias***: pulsando este botón el usuario accederá a la pantalla de Gestión de noticias. Esta funcionalidad estará disponible para el perfil administrador y para el perfil Redactor

- **Descripción Física de la pantalla**
  - *Parte superior*
    Se sitúan los logos de la aplicación en la parte superior derecha, y los logos oficiales de la Universidad de Alcalá y del departamento de ciencias de la computación de la escuela de Politécnica de esta Universidad.
  - *Parte central*

    Primera fila: botones para acciones relacionados con Usuarios
    Botones “Búsqueda de Usuarios” y “Administración de Usuarios”

    Segunda fila: botones para acciones relacionados con Noticias
    Botones “Búsqueda de Noticias” y “Administración de Noticias”

  - Correspondencia con caso de uso: *GESTIÓN ADMINISTRACIÓN*

7.3. Búsqueda de usuarios

A continuación, se muestra la pantalla de Búsqueda de usuarios, a la que se tendrá acceso desde la pantalla Principal, descrita en el apartado anterior, disponible para usuarios con perfil Administrador.

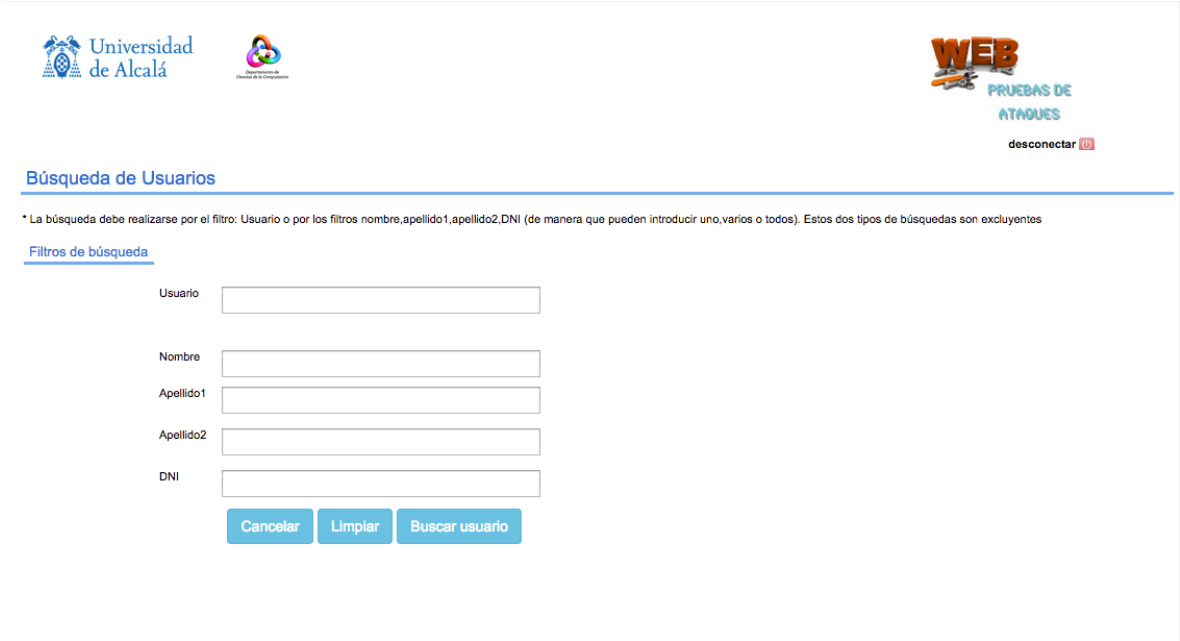


Figura 63 – Pantalla Búsqueda Usuarios

Como se indica en la pantalla, las búsquedas son excluyentes, es decir, se podrán realizar búsquedas por usuario o por el resto de los campos de pantalla, pero no simultáneamente por todos los campos.

Al pulsar el botón *Buscar,* se realizará la búsqueda de usuarios según los filtros introducidos en la pantalla. Previamente se realizará la validación de los campos introducidos, de manera que si no se introduce texto en alguno de los campos se mostrará un mensaje de error. La aplicación abrirá una ventana emergente con siguiente mensaje "Deberá realizar la búsqueda por Usuario o por el resto de filtros de la pantalla", tal y como se muestra en la siguiente imagen:

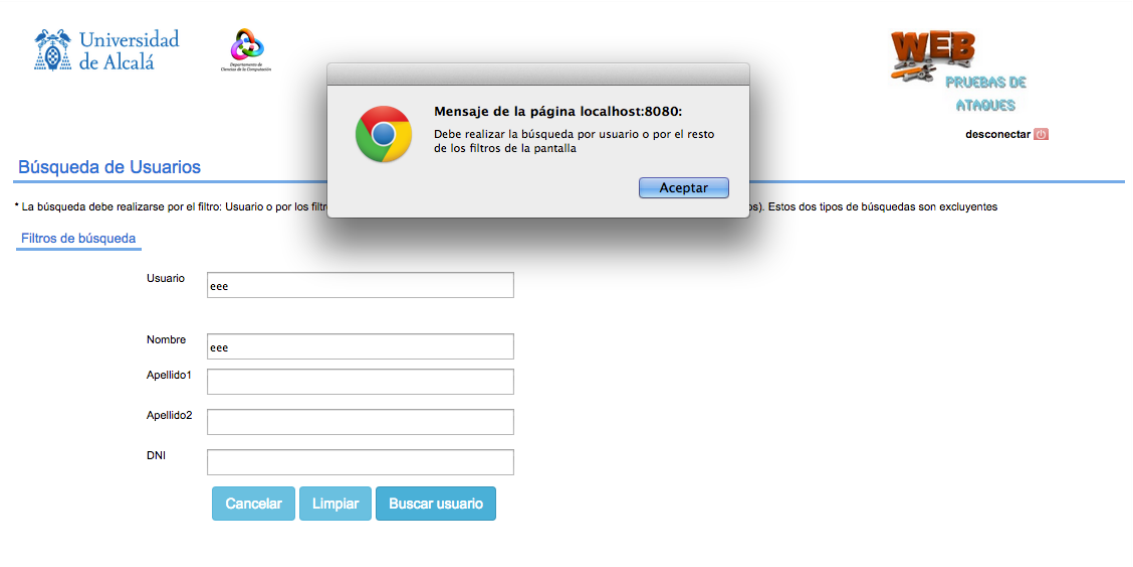


Figura 64– Pantalla Búsqueda Usuarios: Mensaje validación

Las búsquedas que realiza el sistema serán por aproximación, buscando aquellos registros del sistema que contengan la cadena introducida por el usuario en los campos del formulario, sin distinguir entre mayúsculas y minúsculas.

Adicionalmente, pulsando el botón *Cancelar* se podrá regresar a la *Home*.

El botón *Limpiar* borrará el contenido de todos los campos de filtro del formulario.

- **Descripción Física de la pantalla**
  - *Parte superior*
    Se sitúan los logos de la aplicación en la parte superior derecha, y los logos oficiales de la Universidad de Alcalá y del departamento de ciencias de la computación de la escuela de Politécnica de esta Universidad.
  - *Parte central*

    Primera fila: campo de texto para filtrar por el alias del Usuario

    Segunda fila: campo de texto para filtrar por el nombre del Usuario

    Tercera y cuarta fila: campos de texto para filtrar los apellidos del Usuario

    Quinta fila: campo de texto para filtrar por el nombre del Usuario DNI

    Botones “Cancelar”, “Limpiar” y “Buscar”
  - Correspondencia con caso de uso: *BUSQUEDA USUARIOS*

7.4. Resultados búsqueda usuarios

Cuando un usuario introduzca una serie de criterios en la pantalla de búsqueda y pulse el botón “Buscar”, el sistema procederá a recuperar aquellos usuarios cuya información coincida con los criterios introducidos.

Las búsquedas se realizarán por aproximación, buscando aquellos registros cuyo campo contenga la cadena de caracteres introducida por el usuario y no por coincidencia exacta, sin tener en cuenta mayúsculas y minúsculas.

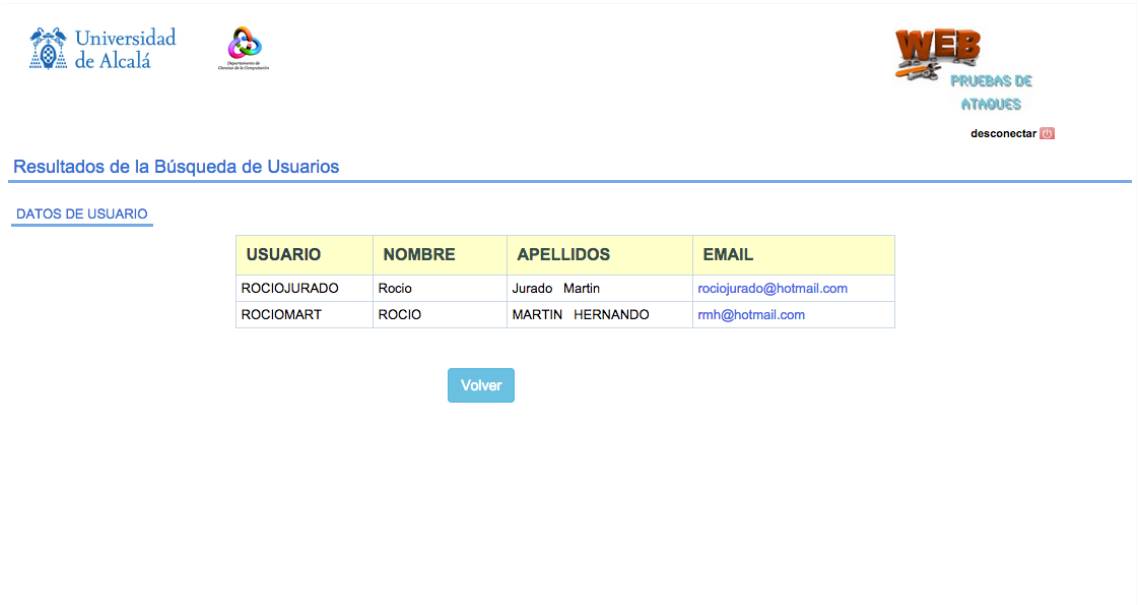


Figura 65 – Pantalla Resultados búsqueda de usuarios

En el caso de que no se haya encontrado ningún resultado, la aplicación devolverá un mensaje informativo al usuario con el siguiente texto "No se han encontrado resultados con los filtros introducidos en la pantalla de Búsqueda", tal y como se muestra en la siguiente imagen:

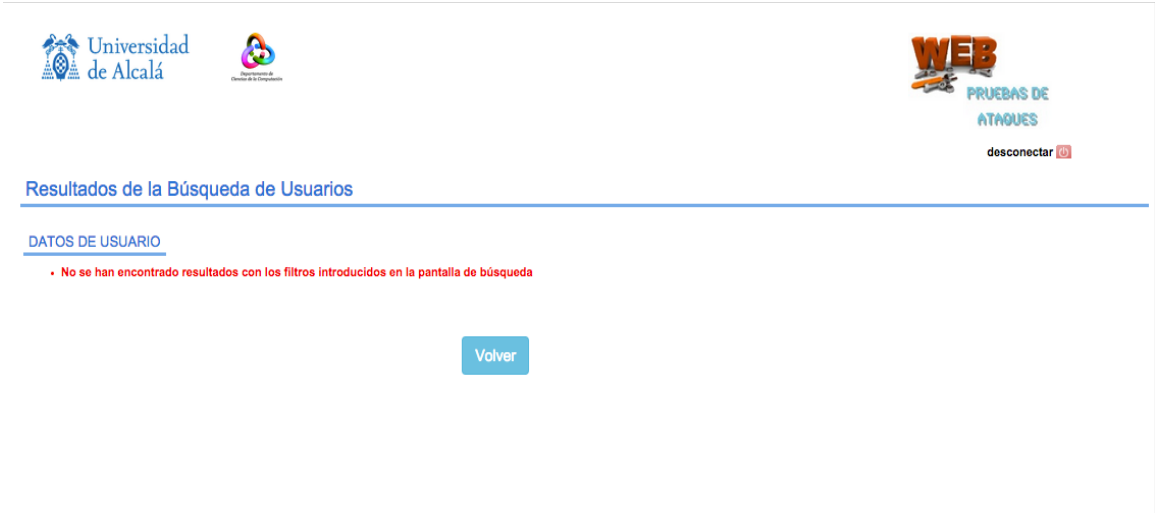


Figura 66 – Pantalla Resultados búsqueda de usuarios: sin coincidencias

El botón *Volver* permite navegar de nuevo hasta el formulario de búsqueda de usuarios.

- **Descripción Física de la pantalla**
  - *Parte superior*
    Se sitúan los logos de la aplicación en la parte superior derecha, y los logos oficiales de la Universidad de Alcalá y del departamento de ciencias de la computación de la escuela de Politécnica de esta Universidad.
  - *Parte central*

    Tabla con los resultados de la búsqueda devuelta.

    La tabla dispone de cuatro columnas: Usuario, nombre, apellidos y email.

    Botones "Volver"

- Correspondencia con caso de uso: *BUSQUEDA USUARIOS*

7.5. Búsqueda de noticias

Desde esta sección se dará la posibilidad al usuario de realizar búsquedas de las noticias registradas en el sistema que puedan resultarle de su interés. Se podrá acceder a ella desde la pantalla Principal, descrita en el apartado anterior, estando disponible para usuarios con todos los perfiles.

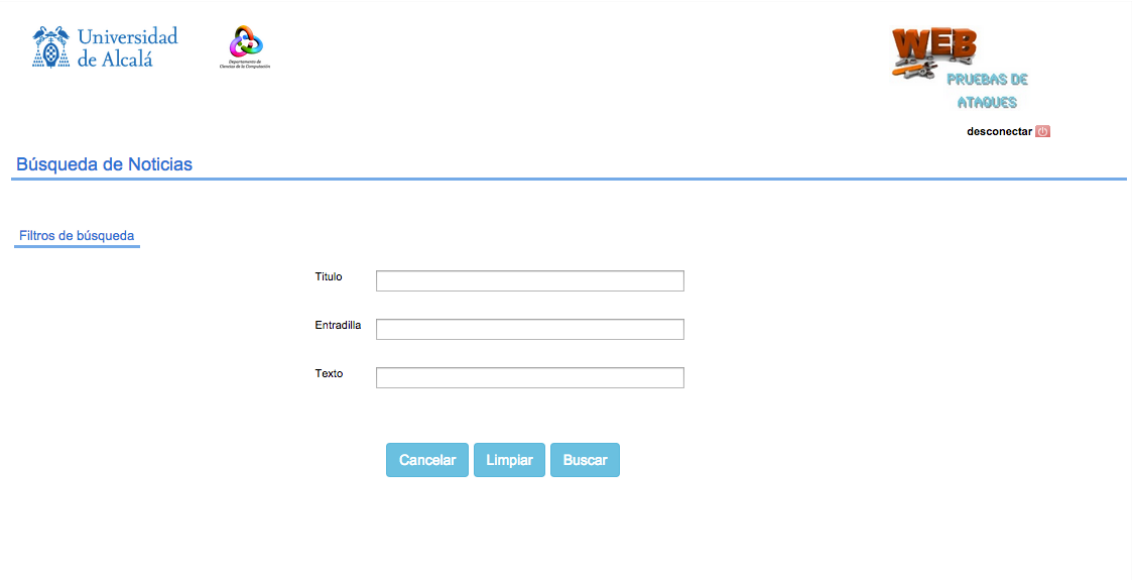


Figura 67 – Pantalla Búsqueda de noticias

Al pulsar el botón *Buscar,* se realizará la búsqueda de aquellas noticias que coincida con los los filtros introducidos en la pantalla. Las búsquedas que realiza el sistema serán por aproximación, buscando aquellos registros del sistema que contengan la cadena introducida por el usuario en los campos del formulario, sin distinguir entre mayúsculas y minúsculas.

Adicionalmente, pulsando el botón *Cancelar* se podrá regresar a la *Home*.

El botón *Limpiar* borrará el contenido de todos los campos de filtro del formulario.

- **Descripción Física de la pantalla**
  - *Parte superior*
    Se sitúan los logos de la aplicación en la parte superior derecha, y los logos oficiales de la Universidad de Alcalá y del departamento de ciencias de la computación de la escuela de Politécnica de esta Universidad.
  - *Parte central*

    Primera fila: campo de texto para filtrar por el Título de la noticia

    Segunda fila: campo de texto para filtrar por la entradilla de la noticia

    Tercera: campos de texto para filtrar el texto de la noticia

    Botones “Cancelar”, “Limpiar” y “Buscar”
  - Correspondencia con caso de uso: BUSQUEDA NOTICIAS

7.6. Resultados búsqueda noticias

Una vez introducidos los criterios de búsqueda en el formulario, el sistema procederá a recuperar aquellas noticias cuya información coincida con los parámetros de búsqueda introducidos.

Las búsquedas se realizarán por aproximación, buscando aquellos registros cuyo campo contenga la cadena de caracteres introducida por el usuario y no por coincidencia exacta, sin tener en cuenta mayúsculas y minúsculas.

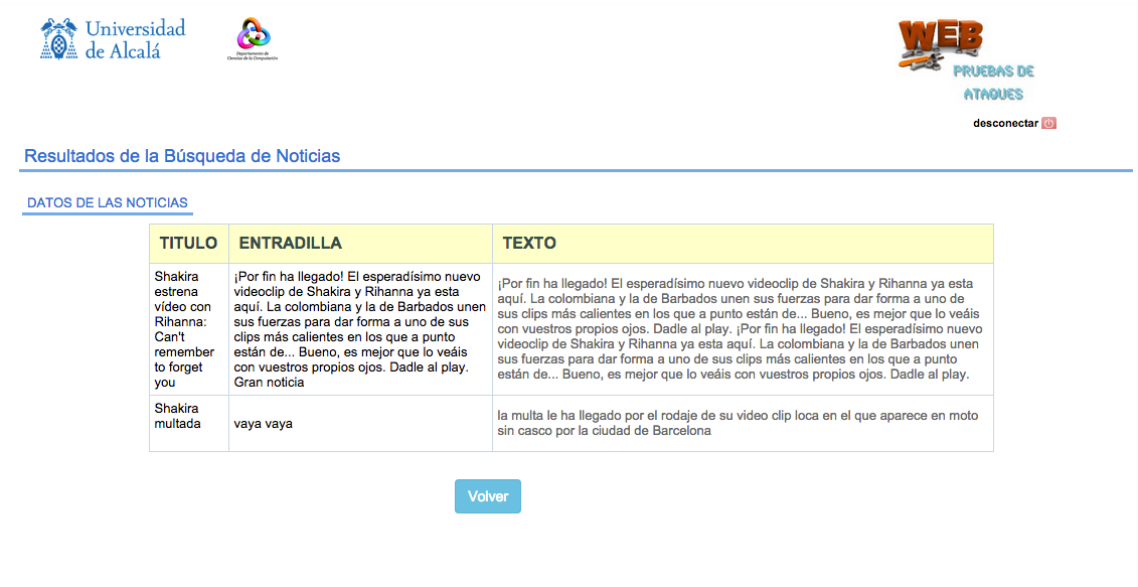


Figura 68 – Pantalla Resultados búsqueda de noticias

En el caso de que no se haya encontrado ningún resultado, la aplicación devolverá un mensaje informativo al usuario con el siguiente texto "No se han encontrado resultados con los filtros introducidos en la pantalla de Búsqueda", tal y como se muestra en la siguiente imagen:

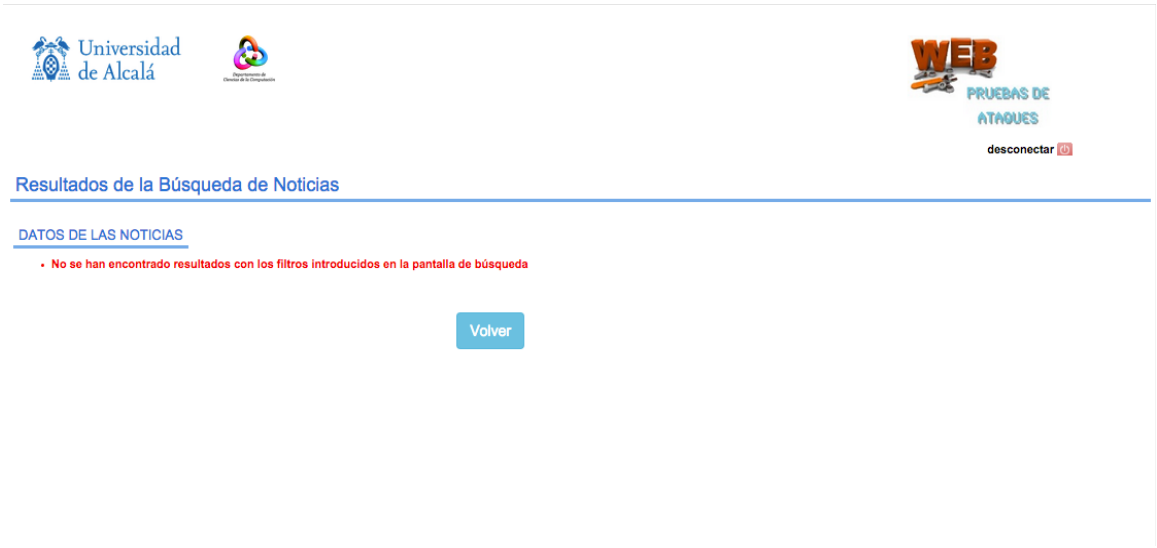


Figura 69 – Pantalla Resultados búsqueda de noticias: sin coincidencias

- **Descripción Física de la pantalla**
  - *Parte superior*

Se sitúan los logos de la aplicación en la parte superior derecha, y los logos oficiales de la Universidad de Alcalá y del departamento de ciencias de la computación de la escuela de Politécnica de esta Universidad.

- *Parte central*

  Tabla con los resultados de la búsqueda devuelta.

  La tabla dispone de cuatro columnas: Titulo, entradilla y texto.

  Botones "Volver"

- Correspondencia con caso de uso: *BUSQUEDA NOTICIAS*

7.7. Registro de usuarios

El sistema posee una funcionalidad de registro de usuarios para todas aquellas personas que no dispongan de credenciales para el acceso a la aplicación y deseen poder acceder.

Desde la pantalla inicial, en la que se solicitan al usuario los datos para el acceso, se dará la posibilidad de acceder a la página de registro, mediante la opción "*Registro de usuario"*.

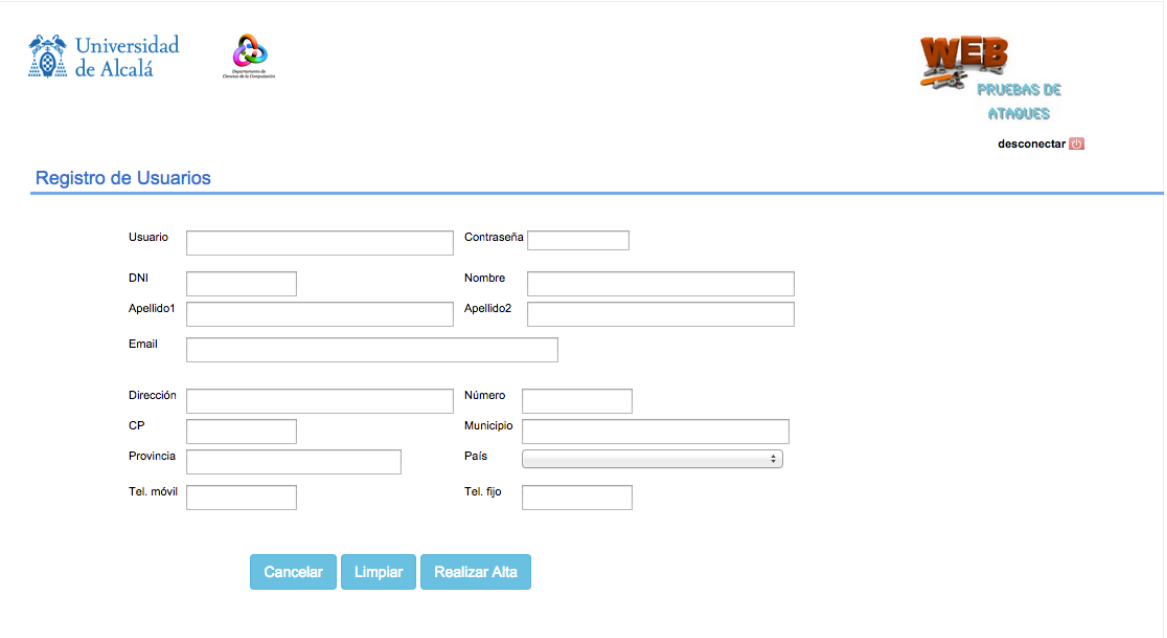


Figura 70 – Pantalla Registro de usuarios

Para realizar el registro del nuevo usuario en el sistema se deberán introducir todos los campos obligatorios, esto es:

- Usuario,
- Contraseña
- Nombre
- Primer apellido
- Segundo apellido
- E-mail

Una vez introducidos, se debe pulsar el botón *Realizar Alta*, en ese momento la aplicación validará que se hayan introducido todos correctamente para realizar la operación.

De manera que si no se introduce texto en alguno de los campos indicados la aplicación devolverá un mensaje informativo en la parte inferior de la pantalla.

Dependiendo del campo no introducido el mensaje variará, incluyendo el nombre del mismo en el mensaje tal y como se muestra a continuación:

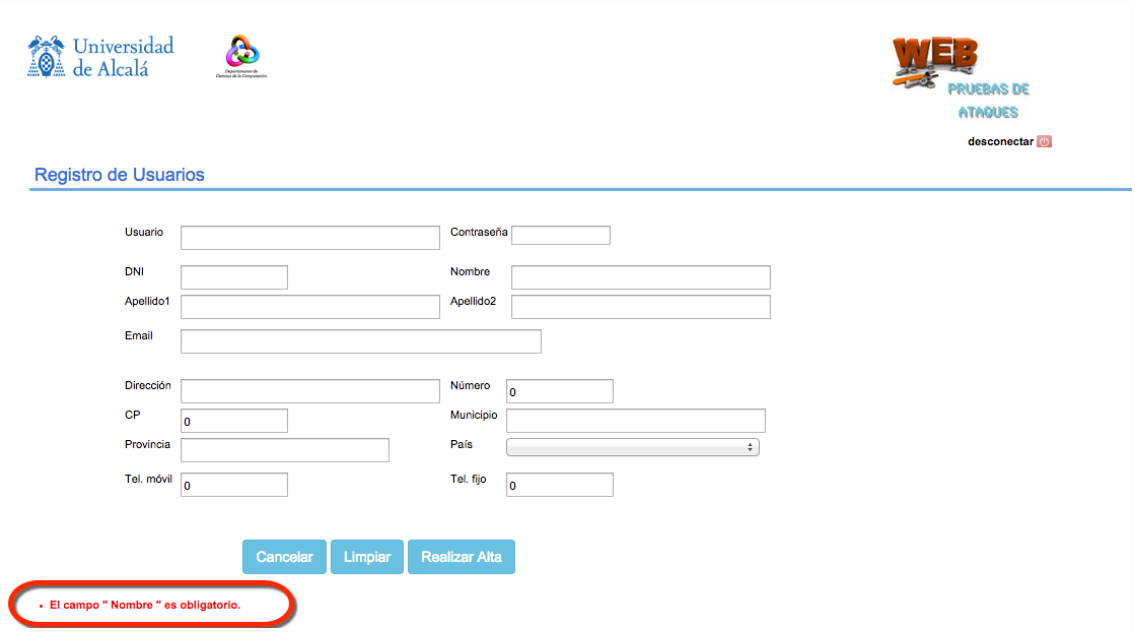


Figura 71 – Pantalla Registro de usuario: error validación

Por el contrario, en caso de que el sistema pueda realizar correctamente el alta del nuevo usuario, se mostrará un mensaje informativo con el texto "La operación se realizó correctamente" en la parte inferior de la pantalla:

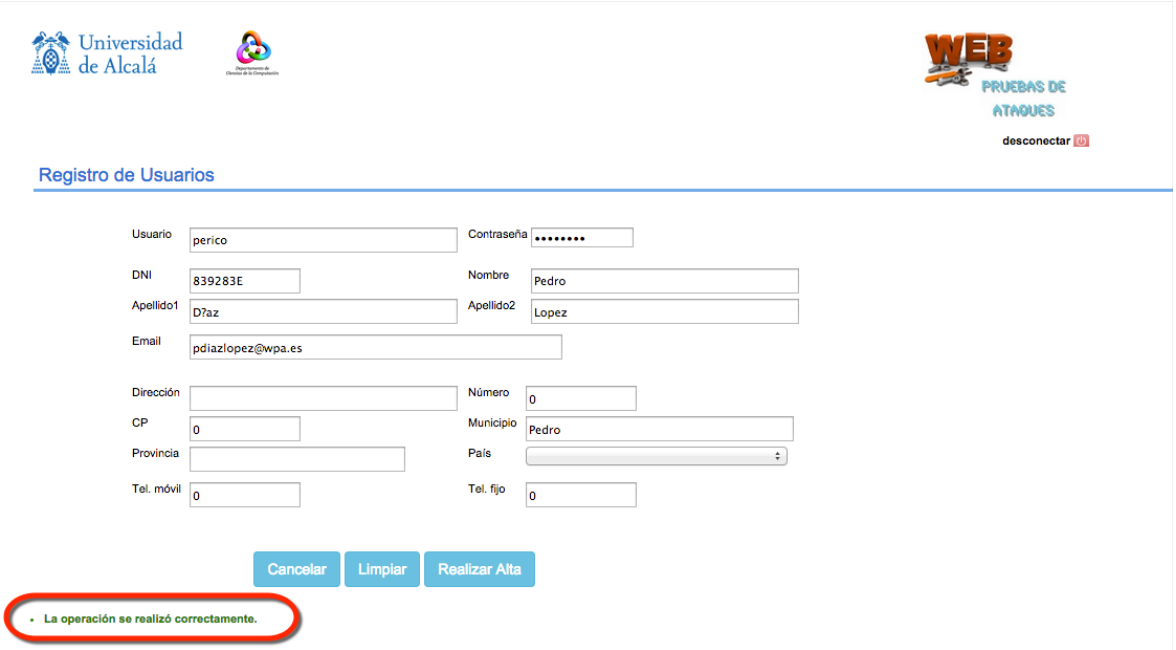


Figura 72 – Pantalla Registro de usuario: alta OK

Adicionalmente, pulsando el botón *Cancelar* se podrá regresar a la pantalla de *Login*.

El botón *Limpiar* borrará el contenido de todos los campos del formulario.

- **Descripción Física de la pantalla**
  - *Parte superior*

    Se sitúan los logos de la aplicación en la parte superior derecha, y los logos oficiales de la Universidad de Alcalá y del departamento de ciencias de la computación de la escuela de Politécnica de esta Universidad.

  - *Parte central*

    Primera fila:

    - Campo de texto para introducir el alias del Usuario
    - Campo de texto para introducir la contraseña del Usuario

    Segunda fila:

    - Campo de texto para introducir el *DNI* del Usuario
    - Campo de texto para introducir el nombre del Usuario

    Tercera fila:

    - Campo de texto para introducir el apellido1 del Usuario
    - Campo de texto para introducir el apellido2 del Usuario

    Cuarta fila:

    - Campo de texto para introducir el email del Usuario

    Quinta fila:

    - Campo de texto para introducir el nombre de la vía del Usuario
    - Campo de texto para introducir el numero de la vía del Usuario

    Sexta fila:

    - Campo de texto para introducir el código postal de la vía del Usuario
    - Campo de texto para introducir el municipio del Usuario

    Séptima fila:

    - Campo de texto para introducir la provincia del Usuario
    - Lista despegable para seleccionar el país del Usuario

    Octava fila:

    - Campo de texto para introducir el teléfono móvil del Usuario
    - Campo de texto para introducir el teléfono fijo del Usuario

    Botones “Home”, “Limpiar” y “Guardar”

  - Correspondencia con caso de uso: *ALTA USUARIOS*

Bibliografía

- Clarke, J. (2012). *SQL Injection Attacks and Defense*. Waltham (MA): Syngress.

- Franco García, A. (2000). *Programación en lenguaje Java*. Mayo 2015, [publicación electrónica] de Universidad del País Vasco. Sitio Web: http://www.sc.ehu.es/sbweb/fisica/curso.htm

- Gamma, E., Helm, R., Johnson R. & Vlissides, J. (1994). *Diseño de patrones: Elementos de reutilización de Software Orientado a Objetos*. Indianapolis (IN): Pearson Education

- Luján Mora, S. (2002). *Programación de aplicaciones web: historia, principios básicos y clientes web*. Alicante: Editorial Club Universitario

- Mifsud, E.. (2012). *MONOGRÁFICO: Introducción a la seguridad informática*. Junio 2015, [publicación electrónica] de Observatorio Tecnológico - Mº de Educación, Cultura y Deporte Sitio web: http://recursostic.educacion.es/observatorio/web/es/software/software-general/1040-introduccion-a-la-seguridad-informatica

- Mikoluk, K. (2013). *Tutorial de Inyección SQL para Principiantes*. Mayo 2015, de Udemy [publicación electrónica]. https://www.udemy.com/blog/tutorial-de-inyeccion-sql-para-principiantes

- Orasmas Bustillos R. (2010). *Curso básico de JDBC*. Mayo 2015, de Slideshare [publicación electrónica]. http://es.slideshare.net/roramas/curso-bsico-de-jdbc

- Pacheco Azpilicueta, A.M. (2008). *Struts MVC en Java*. Mayo 2015, [publicación electrónica] http://www.slideshare.net/siis/struts-en-java

- Rando, E. & Alonso, C. (2012). *Hacking de Aplicaciones Web: SQL Injection* . Madrid: 0xWord.

- Sanchez, S., Sicilia, M.A. & Rodríguez, D. (2011). *Ingeniería del Software. Un enfoque desde la guía SWEBOK*. Madrid: Garceta Grupo Editorial.

- Talens-Oliag S. (1999). *Seguridad en Java*. Instituto Tecnológico de Informática. Mayo 2015, [publicación electrónica] http://www.uv.es/~sto/cursos/seguridad.java/html/sjava-32.html

- Apache Software Foundation. *Apache Tomcat*. Recuperado Mayo de 2015 de: http://apachefoundation.wikispaces.com/Apache+Tomcat

- ArquitecturaJava.com. (2013). *SQL Injection y Consultas Parametrizadas*. Junio 2015 [publicación electrónica] http://www.arquitecturajava.com/sql-injection-y-consultas-preparadas-24/

- Definicion.de (s.f.). *Definición de SQL - Qué es, Significado y Concepto*. Recuperado Mayo 2015 de: http://definicion.de/sql/

- Foro Portal Hacker (s.f.). *Tutorial Inyección XSS*. Junio 2015, [publicación electrónica] http://xschukixs.blogspot.com.es/p/inyeccion-html.html

- Gits Informática. *Legislación.* Recuperado Mayo de 2015 de: http://www.gitsinformatica.com/legislacion.html

- Wikispaces Universidad UFPS. *Normas de las Políticas de Seguridad Informática.* Recuperado Junio de 2015 de: https://seguridadinformaticaufps.wikispaces.com/Normas,+estandares,+Leyes+y+demas+de+las+politicas+de+seguridad.+1150204-159-250-214

- Hostalia Whitepapers. (s.f.) *¿Qué es y cómo funciona un ataque Cross-Site Scripting?*. Junio 2015, [publicación electrónica] http://pressroom.hostalia.com/wp-content/themes/hostalia_pressroom/images/cross-site-scripting-wp-hostalia.pdf

- Java.com (s.f.). *¿Qué es Java?* Recuperado Junio de 2015 de: http://www.java.com/es/download/faq/whatis_java.xml

- MySQL 5.1 *Reference Manual. 2.3.4 Installing MySQL on Microsoft Windows Using an MSI Package*. Mayo 2015 [publicación electrónica]. https://dev.mysql.com/doc/refman/5.1/en/windows-using-installer.html

- National Institute of Standards and Technology (s.f.) CWE – *Common Weakness Enumeration*. Recuperado Junio de 2015 de: https://nvd.nist.gov/cwe.cfm#cwes

- Wikipedia (2015). *MySQL*. Recuperado Junio 2015 de: http://es.wikipedia.org/wiki/MySQL

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá