Universidad de las Ciencias Informáticas

Facultad 1

Título: Sistema para la creación de repositorios personalizados en línea

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autor: Adrián Lázaro Lara Pérez

Tutores: Ing. Alexander Martínez Fajardo

Ing. Gladys Marsi Peñalver Romero

La Habana, Cuba, Junio 2012.
"Año 54 de la Revolución"


“Vamos a ser fortísimos en la Computación, como ya lo estamos siendo en la Medicina, y no solo para beneficio de nuestro pueblo, sino de toda la Humanidad. Será también esta una poderosísima fuerza científica, económica y política del país en toda nuestra lucha por desarrollarnos”.
Fidel Castro

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al departamento Migración y Soporte de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste se firma la presente a los ____ días del mes de junio del año 2011.

Adrián Lázaro Lara Pérez

_____

Ing. Alexander Martínez Fajardo

_____

Ing. Gladys Marsi Peñalver Romero

_____

DEDICATORIA

A mi madre por darlo todo por mí incondicionalmente.

A mis padres por ser mis ejemplos en la vida.

A mis hermanos por darme aliento y confianza.

A mi novia Anneris Alonso Serrano por apoyarme en los momentos más difíciles.

RESUMEN

Las experiencias obtenidas por el departamento de Migración y Soporte durante la prestación de sus servicios a diversas empresas, le permitió detectar deficiencias que obstaculizan el desarrollo del proceso de migración; entre ellas se encuentra la existencia de problemas de conectividad que dificultan el acceso a repositorios de aplicaciones en línea y la innecesaria utilización de la mayoría de los paquetes contenidos en un repositorio; lo que conlleva a la adquisición de dispositivos de gran capacidad de almacenamiento para trasladar grandes volúmenes de aplicaciones a las empresas inmersas en el proceso, la mayoría de las veces de gran costo.

Con el propósito de erradicar estas deficiencias, el objetivo del presente trabajo de diploma y está enfocado a desarrollar un sistema que permita la creación de repositorios personalizados en línea. Para ello se analizaron aplicaciones que internacionalmente se emplean para crear pequeños repositorios personalizados. Además se realizó un estudio de las tecnologías, herramientas, lenguajes a utilizar en la construcción del sistema y la definición de los elementos necesarios para el exitoso desarrollo del mismo.

Teniendo como resultado el sistema para la creación de repositorios personalizados en línea (REPPER), el cual cumple con los requisitos establecidos por el cliente, haciendo más fácil e intuitiva la creación de repositorios personalizados. Este permite la agilización del proceso de migración a software libre que presta el departamento de Migración y Soporte a las diversas empresas.

Palabras claves: código abierto, migración, paquetes, repositorios personalizados, software libre.

Índice

INTRODUCCIÓN

A medida que surgen nuevas tecnologías en el mundo; países como Ecuador, Alemania, España, Argentina, Francia, Venezuela y otros [1], optan por la migración parcial o total a tecnologías y estándares abiertos. Para estos países la migración no es solo cambiar un Sistema Operativo (SO) por otro; efectuarla implica un mayor ahorro económico, una mayor seguridad en el tratamiento de los datos y de la información manejada por quienes la acogen. Es, por sobre todas las cosas, una vía para lograr su soberanía tecnológica.

Cuba no está ajena a los sucesos del mundo, y en abril del 2004 el Consejo de Ministros adoptó el Acuerdo 084/2004, donde indicaba al Ministerio de la Informática y las Comunicaciones (MIC) ordenar la paulatina migración a aplicaciones de código abierto y software libre en las empresas cubanas. Un año más tarde se crean los cuatro grupos que estarían al frente de dicho proceso: el Grupo de Capacitación, el Legal, el de Divulgación y el Técnico [2].

El Grupo Técnico Nacional tiene como tareas fundamentales la elaboración de la Guía Cubana para la Migración a Aplicaciones de Código Abierto, el desarrollo de la distribución cubana de GNU/Linux Nova y el establecimiento de la metodología de trabajo y los lineamientos técnicos que indican los pasos necesarios para ejecutar el proceso de migración de manera ordenada. Posee además la tarea de migrar, asesorar y garantizar el soporte para la migración nacional. Dentro de este grupo se encuentra el departamento de Migración y Soporte, perteneciente al Centro de Software Libre (CESOL) de la Universidad de las Ciencias Informáticas (UCI).

Organismos e instituciones estatales abogan por el uso del software libre, entre los cuales figuran la Aduana General de la República, la UCI, el MIC, el Ministerio de la Educación Superior (MES), el Ministerio de Cultura (MINCULT), la Empresa de Telecomunicaciones de Cuba SA (ETECSA) y la Oficina Nacional de Estadísticas e Información (ONEI); ganándose con ello no solo experiencia, sino también herramientas y personas capacitadas para las futuras tareas.

En las experiencias obtenidas con las empresas que han efectuado de una manera u otra la migración de sus aplicaciones a código abierto, el departamento de Migración y Soporte ha detectado un conjunto de deficiencias que obstaculizan el proceso de migración. La existencia de problemas en la conectividad es uno de ellos, dificultando con esto el acceso a repositorios de software en línea.

Un repositorio de software para cualquier distribución oscila aproximadamente entre los 30 y 120 GB, con tendencia a aumentar debido a la aparición de nuevas aplicaciones. Producto de estos grandes volúmenes de información, y ante la necesidad de su almacenamiento y traslado, se hace

necesaria la adquisición de dispositivos de almacenamiento extraíbles con capacidad suficiente como para realizar dichas funciones. Sin embargo, en estos repositorios existen grandes cantidades de paquetes que no serán utilizados en el proceso.

Estos dispositivos de almacenamiento extraíbles tienen un valor que oscila aproximadamente entre los 59 y 150 dólares en el mercado internacional [3], encareciendo sustancialmente el proceso de migración. Por estas razones, muchos de los usuarios y empresas se ven imposibilitados a la hora de migrar a un SO de código abierto y software libre.

Actualmente existen aplicaciones que brindan solución a algunos de los problemas planteados, pero solo pueden ser utilizadas en la máquina donde se va a construir el repositorio. Además de ser necesaria la descarga para la misma, de aquellos paquetes que conformarán el repositorio personalizado. Otras aplicaciones presentan problemas en funcionalidades o no satisfacen algunas de las requeridas, como la obtención de repositorios de código fuente de las aplicaciones. Las razones antes expuestas motivaron al departamento de Migración y Soporte al desarrollo de un sistema que sea capaz de dar solución a las deficiencias detectadas y permita brindar un servicio de creación de repositorios personalizados utilizando solamente un navegador web.

Partiendo de la situación problémica expuesta anteriormente, se llega al siguiente **problema científico**: ¿Cómo personalizar los repositorios atendiendo a las necesidades específicas de cada cliente del SO Nova? Para dar solución al mismo se define como **objetivo general**, desarrollar un sistema que permita la creación de repositorios personalizados en línea. Siendo el **objeto de estudio** de la investigación es la creación de repositorios personalizados y como **campo de acción** “la creación de repositorios personalizados para el SO Nova”.

Se pretende cumplir el objetivo general a partir los siguientes **objetivos específicos**:

1. Realizar un estudio de la bibliografía actual, tanto nacional como internacional, relacionada con los sistemas de creación de repositorios personalizados.
2. Identificar las herramientas, metodología de desarrollo de software y tecnologías que posibilitarán el desarrollo de la solución propuesta.
3. Analizar y diseñar la solución propuesta.
4. Implementar el sistema para la creación de repositorios en línea.
5. Probar el sistema desarrollado.

En esta investigación la idea a defender es que con la creación del sistema para la realización de repositorios personalizados en línea, se hará más ágil y eficiente el proceso de migración a aplicaciones de código abierto.

Los anteriores objetivos se concretan en las siguientes **tareas de investigación**:

1. Estudio del estado del arte acerca de la creación de repositorios personalizados.
2. Análisis de las herramientas, lenguajes y la metodología a utilizar en el desarrollo del sistema propuesto.
3. Definición del diseño y la arquitectura para el sistema.
4. Realización de las Historias de Usuario (HU)
5. Diseño y ejecución de pruebas funcionales a la aplicación para asegurar la calidad del producto.

Los **métodos teóricos** utilizados fueron el **Analítico – Sintético** y el **Inductivo – Deductivo**. El primero de estos, se empleó en el estudio de los mecanismos de creación de repositorios personalizados ya existentes; lo que permitió comprender su funcionamiento y la identificación de elementos que resultaron útiles para dar solución al problema en cuestión.

Por otro parte, el método Inductivo – Deductivo se utilizó para especificar la arquitectura y el uso de patrones de diseño en el desarrollo de un sistema compuesto por módulos, flexible y escalable. Además, permitió identificar las funcionalidades que poseen en común los sistemas estudiados y que definitivamente deben formar parte de la solución propuesta en esta investigación.

La técnica empleada para la recogida de la información, fue *la* **tormenta de ideas**. La misma fue realizada con especialistas del departamento Migración y Soporte, además de personas de la comunidad de software libre; donde se obtuvieron diferentes puntos de vista e ideas para la elaboración de la aplicación.

El presente trabajo de diploma está constituido por 3 capítulos, de los cuales a continuación se brinda una breve descripción:

Capítulo # 1: **Fundamentación teórica:** Se ofrecerá una visión general de los sistemas que permiten la creación de repositorios personalizados, abordando definiciones y conceptos fundamentales. Además, se describirán las herramientas, tecnologías y la metodología de desarrollo de *software* a utilizar en el desarrollo del sistema.

Capítulo # 2: **Definición del sistema propuesto:** Se definirán los requisitos funcionales (RF) y no funcionales en la Lista de Reserva de Producto (LRP). Además de reflejar en las HU las funcionalidades del sistema, con sus prototipos de interfaces gráficas de usuario no funcionales. Se plasmarán los roles que van a existir en la solución propuesta, así como la descripción de la arquitectura y el diseño.

Capítulo # 3: **Implementación y pruebas:** Se describirá el plan de releases, las tareas de ingeniería (TI) y el estándar de código a utilizar en la implementación del sistema. Además se expondrán las pruebas realizadas al mismo para ganar en calidad, lo que garantiza que el producto llegue a manos del cliente con el menor número de errores posibles.

CAPÍTULO # 1. FUNDAMENTACIÓN TEÓRICA.

En el presente capítulo se ofrecerá una visión general de los sistemas que permiten la creación de repositorios personalizados, abordando definiciones y conceptos fundamentales. Además se describirán las herramientas, tecnologías y la metodología de desarrollo de *software* a utilizar en el desarrollo del sistema.

Para una mayor comprensión de los diferentes términos que se emplearán en la investigación, se hace necesario definir los siguientes conceptos.

1.1 Conceptos asociados al dominio del problema.

Para la realización de esta investigación se hace necesario conocer los conceptos relacionados con los sistemas de paqueterías y repositorios. Estos y otros elementos guiarán el desarrollo del sistema para la creación de repositorios personalizados.

- **Paquete:** Es un conjunto de ficheros que forman una aplicación o una unión de varias aplicaciones relacionadas, normalmente formando un único fichero, con un formato propio y normalmente comprimido [4].
- **Paquetes de Código Fuente:** Son un conjunto de instrucciones a seguir por la computadora para ejecutar un programa. Generalmente los paquetes fuentes están contenidos en ficheros comprimidos en *tar.gz* o *bzip*, los que son herramientas básicas a la hora de instalar *software* no organizado, útiles además para realizar procesos de salva y restauración de archivos [5].
- **Paquetes Binarios:** Estos paquetes contienen código de máquina, no código fuente, por eso cada tipo de procesador necesita su propia versión de cada uno. Al existir varias distribuciones de GNU/Linux existen varios tipos de paquetes binarios, de los cuales, algunos son utilizados en diversas distribuciones [6].

Los paquetes de tipo **.deb** y **.rpm** son ejemplos de paquetes binarios que contienen los ficheros y librerías asociadas a una aplicación. Los **.deb** son utilizados por la distribución Debian y sus derivadas. Es un contenedor de aplicaciones que facilita en gran medida la actualización del sistema. El mismo contiene la lista de las dependencias que este necesita para su funcionamiento. Se suele nombrar de la forma nombredelpaquete_versión_plataforma.deb.

Los **.rpm** son utilizados por las distribuciones RedHat, Caldera, Madrake, SuSE y TurboLinux. Su uso está muy extendido y es posible instalar este tipo de paquetes mediante la aplicación *rpm*. Se suelen nombrar de la forma nombredelpaquete_versión_plataforma.rpm.

- **Repositorios:** Lugar donde se centralizan y organizan los paquetes a instalar en cualquier

distribución de GNU/Linux. Estos se dividen en dos tipos de repositorios: los de paquetes binarios y los de paquetes fuentes. Estos últimos se encargan de centralizar y organizar el código fuente de cada programa o aplicación.

Luego de haber abordado estos conceptos, se hace necesario el estudio de las principales aplicaciones utilizadas en la creación de repositorios personalizados de la rama de paquetería **.deb**. La razón para centrar el análisis en estos sistemas, se fundamenta en que la aplicación requerida por el cliente debe ser capaz de realizar repositorios para la distribución GNU/Linux Nova, que utiliza este tipo de paquetería.

1.2 Análisis de sistemas para la creación de repositorios personalizados.

1.2.1 Portable Software Center (PSC)

Es un programa diseñado para crear repositorios personalizados de aplicaciones. El mismo es desarrollado por el proyecto Konoha, perteneciente a la comunidad de software libre de la UCI. Entre sus principales características podrían citarse las siguientes:

- Crea los repositorios con la estructura clásica.
- El usuario puede ver información acerca de un paquete.
- Instala las aplicaciones que se encuentran en los repositorios creados por él.
- Utiliza los repositorios que se encuentran configurados en el sistema.
- Implementa filtros para que las búsquedas de aplicaciones sean más rápidas.
- Contiene un campo de búsqueda con autocompletamiento.
- Está licenciado bajo GPLv3.

Entre sus desventajas para esta investigación podrían citarse las siguientes:

- Es un sistema de escritorio, por lo que no puede ser empleado para los fines de esta investigación, es decir, para brindar un servicio conectado a Internet.
- No genera el compactado final que contiene el repositorio personalizado.
- Tiene problemas en la gestión de las dependencias de los paquetes a descargar.

- No es multiplataforma.
- El proyecto está detenido actualmente, con fecha de la última versión (25/11/2011) y los desarrolladores están trabajando en otros proyectos.

1.2.2 APTonCD

APTonCD es una herramienta con una interfaz gráfica que permite crear uno o más CD o DVD con todos los paquetes que se hayan descargado a través de *apt-get* o *aptitude*, creando un repositorio portable que se puede utilizar en otros ordenadores [7]. Entre sus principales características podrían citarse las siguientes:

- Permite realizar salvas de sus paquetes descargados mediante los gestores de paquetes *apt-get, aptitude o synaptic*.
- Utiliza los repositorios que se encuentran configurados en el sistema.
- Está licenciado bajo GPLv2.
- Genera un .iso y brinda la opción de guardarlo en un soporte de almacenamiento (CD o DVD).

Entre sus desventajas para esta investigación podrían citarse las siguientes:

- Al igual el Portable Software Center, APTonCD es un sistema de escritorio, lo que limita su utilización en un SO específico.
- Presenta problemas a la hora de gestionar las dependencias de los paquetes a incluir en el repositorio.
- No es multiplataforma.
- Es necesario ser administrador de la estación de trabajo para tener funcionalidades avanzadas.
- El proyecto está descontinuado actualmente, con fecha de última versión (14/03/2007).

1.2.3 Keryx

Es una herramienta libre y de código abierto para manejar los paquetes en sistemas operativos basados en Debian (Ubuntu y Linux Mint) sin necesidad de conexión a Internet. Proporciona una interfaz gráfica que ayuda a instalar y actualizar *software* [8]. Entre sus principales características podrían citarse las siguientes:

- Gestiona las dependencias de los paquetes a descargar.
- Utiliza los repositorios que se encuentran configurados en el sistema.
- Es multiplataforma.
- Está licenciado bajo GPLv3.

Entre sus desventajas para esta investigación podrían citarse las siguientes:

- Es un sistema de escritorio.
- Es necesario ser administrador de la estación de trabajo para tener funcionalidades avanzadas.
- No genera el .iso o el compactado con el repositorio.
- No crea el repositorio con una estructura organizada.
- La última versión no funciona en Nova GNU/Linux y tiene fecha (01/01/2011).

1.3 Valoración del estudio realizado.

Una vez concluido el estudio de los sistemas para la creación de repositorios personalizados, se puede constatar que aunque poseen similitudes en varias de sus funcionalidades, la gestión de dependencias de los paquetes no funciona correctamente en algunos de ellos (tal es el caso de APTonCD y PSC). Se desea que la solución propuesta genere un fichero comprimido con el repositorio listo para ser utilizado, sin embargo, la única aplicación que realiza esta tarea es APTonCD, pero es un proyecto obsoleto e inactivo, por lo que no se toma como punto de partida para el desarrollo de la solución propuesta. Además de esto, ninguno de los sistemas analizados cumplen con el concepto de "*software* como servicio" requerido por el departamento de Migración y Soporte. Esto se fundamenta en que no permiten su acceso y administración a través de la red, ni la gestión centralizada de los mismos mediante la web. Otra de las exigencias del cliente, es que el sistema propuesto debe estar implementado utilizando el lenguaje de programación PHP, lo

que tampoco cumplen las aplicaciones estudiadas.

Por las razones antes expuestas se decidió desarrollar una nueva aplicación, en lugar de seleccionar una de las estudiadas. Sin embargo, se tendrán en cuenta algunas de sus características para la creación del sistema propuesto:

- El usuario puede ver información acerca de un paquete seleccionado.
- Crear los repositorios con la estructura clásica.
- Gestionar las dependencias de los paquetes a descargar.
- Organización de paquetes por secciones.
- Permitir la realización de búsquedas de paquetes en el repositorio.

A partir de la valoración anterior y la necesidad de crear un sistema que responda a las especificidades del cliente, se empleará la metodología ágil de desarrollo SXP para guiar la realización del mismo.

1.4 Metodología de desarrollo de software.

Una metodología de desarrollo de *software* no es más que un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Optimizan la gestión de recursos y garantizan la calidad del producto. Por lo que para el desarrollo del trabajo de diploma se seleccionó la metodología ágil de desarrollo SXP, debido a una decisión del proyecto.

SXP

Metodología desarrollada en la UCI, que ofrece una estrategia tecnológica a partir de la introducción de procedimientos ágiles que permiten actualizar los procesos de *software*. Esta mejora la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo y ayudando al líder del proyecto a tener un mejor control del mismo. Su particularidad es tener como parte del equipo al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto.

Consta de 4 fases:

Fase de Planificación-Definición: Se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto. Cuenta con tres tareas principales: ingeniería de sistemas o de información, planificación del proyecto y análisis de los requisitos.

Fase de Desarrollo: Se implementa un sistema listo para entregar en una serie de iteraciones. Cuenta con tres tareas fundamentales: diseño del *software*, generación de código y prueba del *software*.

Fase de Entrega: Se despliega y se pone en marcha el *software*.

Fase de Mantenimiento: Se realiza el soporte para el cliente.

SXP es ideal para proyectos de corta duración con requisitos cambiantes o no bien definidos, donde prevalezca la retroalimentación entre el cliente y el equipo de trabajo. El desarrollo con SXP se realiza en iteraciones cortas (*sprints*) a lo largo de 3 fases, dándole cumplimiento a un grupo de actividades, de las que se generan una serie de artefactos, que documentan el proceso de desarrollo, obteniendo un *release* del producto con nuevas funcionalidades [9].

Una vez conocida la metodología de desarrollo a utilizar es necesario hacer la selección de las herramientas, lenguajes, y tecnologías que se deben utilizar para el exitoso desarrollo del sistema. Teniendo en cuenta lo anteriormente planteado, a continuación se presenta un análisis sobre las mismas.

1.5 Lenguajes, tecnologías y herramientas de desarrollo utilizadas.

1.5.1 Lenguajes

Hypertext Pre-processor (PHP)

Es un lenguaje de código abierto interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Es posiblemente el lenguaje más usado para la programación de sitios web dinámicos por su sencillez y prestaciones; aún más cuando su versión más reciente, PHP 5 (v5.3), tiene incorporado un grupo de mejoras en cuanto a Programación Orientada a Objetos (POO). Este lenguaje está publicado bajo *PHP Licence* v3.01 (http://www.php.net) y considerado como software libre por la *Free Software Foundation* [10].

Características:

- Es un lenguaje multiplataforma.
- Dispone de una conexión propia a todos los sistemas de bases de datos, tales como:
  - ✔ *MySQL, PostgreSQL, mSQL, Oracle, dbm, filepro, Hyperwave, Informix, InterBase y Sybase*, entre otras.
  - ✔ Se integra a los servidores web más utilizados, Apache e *Internet Information Server*

(IIS).

- ✔ Posee alto rendimiento en la ejecución de los *scripts*.
- ✔ Posee un gran número de funciones predefinidas.

JavaScript

JavaScript es un lenguaje de *scripts* desarrollado por *Netscape* para incrementar las funcionalidades del lenguaje HTML. Es un lenguaje interpretado, es decir, no requiere compilación. Además, es orientado a eventos, ligero y permite interactuar con el navegador de manera dinámica y eficaz.

XHTML

Es una versión más estricta y limpia de HTML [11]. XHTML extiende de HTML 4.0, combinando la sintaxis de HTML con la de XML. Las principales diferencias con HTML son:

- ➢ Los elementos XHTML deben estar correctamente anidados.
- ➢ Los elementos XHTML deben estar siempre cerrados.
- ➢ Los elementos XHTML deben estar siempre en minúsculas.
- ➢ Los documentos XHTML deben tener un elemento raíz.

Cascading Style Sheets (CSS)

Es el lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML [12]. Este lenguaje se utiliza para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista, etc. El W3C es el encargado de definir los estándares a utilizar por los navegadores. Actualmente existen los estándares CSS1, CSS2.0, CSS2.1 y se trabaja en el estándar CSS3 ya soportado por algunos navegadores modernos (Mozilla Firefox, Opera y Safari). El uso de hojas de estilos permite separar los contenidos y su presentación, siendo imprescindible para crear páginas web complejas. Este lenguaje mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite su visualización en infinidad de dispositivos diferentes.

1.5.2 Tecnologías.

Asynchronous JavaScript And XML (AJAX)

AJAX se puede traducir como JavaScript asíncrono + XML. No es una tecnología en sí misma. En realidad, se trata de la unión de varias tecnologías que se desarrollan de forma autónoma y que se unen de formas nuevas y sorprendentes. El término fue anunciado por primera vez en el artículo "*Ajax: A New Approach to Web Applications*".

Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

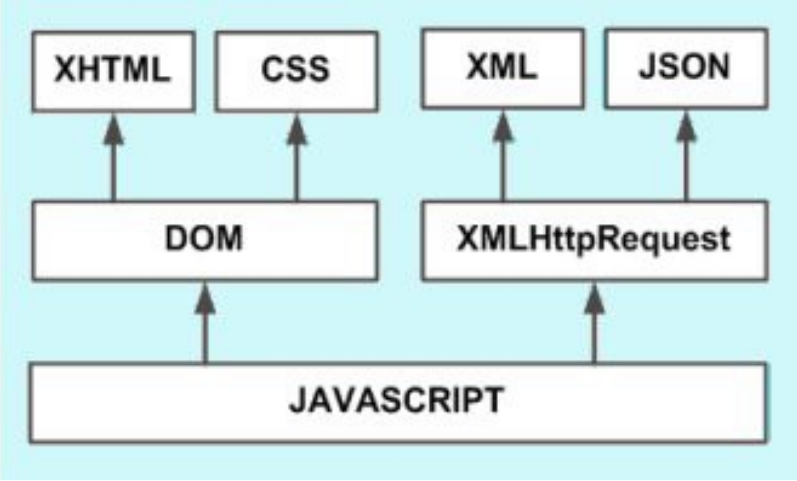

Ilustración 1: Tecnologías agrupadas bajo el concepto de AJAX

AJAX permite mejorar completamente la interacción del usuario con la aplicación, ya que evita las recargas constantes de la página y el intercambio de información con el servidor se produce en un segundo plano. Permite que las aplicaciones web sean más versátiles e interactivas, haciendo que las páginas web puedan realizar llamadas asíncronas al servidor de forma transparente para el usuario. Las tecnologías AJAX se aplican en todos los navegadores web modernos, tales como Mozilla Firefox, Internet Explorer u Opera [13].

1.5.2.1 Framework para el lenguaje Javascript

jQuery

jQuery es un *framework* de código abierto distribuido bajo las licencias *General Public License* (GPL) y MIT. El mismo implementa la programación orientada a objetos que permite su funcionamiento de igual forma en las plataformas más utilizadas. Simplifica la manera de interactuar con los documentos HTML, el control de eventos, animación, y las interacciones AJAX [14]. Es un producto estable, bien documentado y cuenta con una amplia variedad de extensiones desarrolladas sobre la base de la propia librería jQuery, lo que permite obtener sorprendentes efectos y facilidades en un sitio web.

1.5.2.2 Librerías.

jQuery UI

jQuery UI es una librería de componentes para el *framework* de jQuery. Esta le añade un conjunto de *plug-ins*, *widgets* y efectos visuales para la creación de interfaces gráficas de usuario de aplicaciones web con mejorada usabilidad. Cada componente o módulo se desarrolla de acuerdo a la filosofía de jQuery. Esta librería está dividida en 4 módulos:

Núcleo: este contiene las funciones básicas que utilizarán el resto de los módulos.

Interacciones: Añade diferentes comportamientos a los elementos:

- ***Droppable*:** Permite que el elemento responda a elementos arrastrables.
- ***Draggable*:** Hace al elemento arrastrable.
- ***Resizable*:** Permite redimensionar el elemento.
- ***Selectable*:** Permite seleccionar entre una lista de elementos.
- ***Sortable*:** Ordena una lista de elementos.
- ***Widgets*:** Es un conjunto completo de controles UI. Cada control tiene un conjunto de opciones configurables y se les pueden aplicar estilos CSS.
- ***Accordion*:** Menú con efecto acordeón.
- ***Autocomplete*:** Caja con auto completado.
- ***Button*:** Botón.
- ***Dialog*:** Ventanas con contenido.

- ***Slider*:** Elemento para elegir en un rango de valores.
- ***Tabs*:** Pestañas.
- ***Datepicker*:** Calendario gráfico.
- ***Progressbar*:** Barra de progreso.
- **Efectos:** Esta es una API para añadir transiciones animadas y facilidades para interacciones. Ejemplos de estos efectos se tienien *Hide, Animate, addClass, switchClass y toogleClass.*

Además contiene la herramienta *ThemeRoller*, que permite la creación, modificación, visualización y descarga de temas desde la misma página de la jQuery UI.

1.5.2.3 Sistemas para la abstracción de la Base de Datos.

PHP Data Objects (PDO)

La extensión PDO define una interfaz ligera para tener acceso a bases de datos en PHP. Cada controlador de base de datos que implementa está interfaz, puede exponer bases de datos específicas como funciones de extensión regular. Es necesario tener en cuenta que no se pueden realizar las funciones de base de datos utilizando dicha extensión por sí misma. Para ello se debe utilizar un controlador PDO en específico [15].

lustración 2: Funcionamiento de PDO

Este soporta varias Bases de Datos y para esto implementa **driver** para su manejo. Controladores que implementa actualmente la interfaz PDO:

- CUBRID
- MS SQL Server
- Firebird/Interbase
- IBM
- Informix
- MySQL

- MS SQL Server
- Oracle
- ODBC and DB2
- PostgreSQL
- SQLite
- 4D

ORM Doctrine

Es un ORM para PHP 5.2.3 y versiones posteriores. Entre las ventajas que brinda Doctrine se encuentran:

- **Reutilización:** La reutilización permite llamar a los métodos de un objeto de datos desde distintas partes de la aplicación e incluso desde diferentes aplicaciones.
- **Encapsulación:** La capa ORM encapsula la lógica de los datos, pudiendo hacer cambios que afecten a toda la aplicación, únicamente modificando una función.
- **Portabilidad:** La utilización de una capa de abstracción permite cambiar en mitad de un proyecto de una base de datos MySQL, a una Oracle sin dificultad. Esto es debido a que no se utiliza una sintaxis SQL para acceder a nuestro modelo, sino una sintaxis propia del ORM utilizado que es capaz de traducir a diferentes tipos de bases de datos.
- **Seguridad:** Este implementa mecanismos de seguridad que protegen nuestra aplicación de los ataques más comunes como *SQL Injections*.
- **Mantenimiento del código:** Gracias al correcto ordenamiento de la capa de datos, modificar y mantener nuestro código es una tarea sencilla.

Además de las ventajas expuestas que conlleva un ORM, uno de sus puntos fuertes es su lenguaje *Doctrine Query Language* (DQL) inspirado en el HQL de Hibernate [16]. Es una librería muy completa, que trae consigo una gama de ventajas para el desarrollador como la generación automática del modelo, la implementación de buscadores mágicos, soporte para cualquier operación en un CRUD (Crear, Listar, Actualizar y eliminar), soporte para validaciones de los datos del modelo, posee gran cantidad de documentación y se integra con los principales IDE de programación existentes. Utiliza como capa de abstracción a la base de datos PDO [17].

ADOdb

Es una librería de abstracción de base de datos para PHP. Licenciado bajo la licencia BSD-LGPL.

Es compatible con PHP 4 y PHP 5. Tiene varias ramas en su proyecto, entre las cuales se encuentra *ADOdb Lite*, que no es más que una pequeña implementación de *ADOdb,* pero con menos funcionalidades y más rápida. *ADOdb* implementa varias funcionalidades de ayuda para su mejor uso, como paginadores y un sistema de Debugging [18]. Entre las ventajas que brinda este se encuentran:

- Permite la migración de una base de datos a otra con el menor costo posible.
- Posee buena documentación.
- Usa en su mayoría las funciones estándar de SQL para todas las bases de datos.
- Posibilita el manejo de las bases de datos de: MySQL, PostgreSQL, Interbase, Firebird, Informix, Oracle, MS SQL, Foxpro, Access, ADO, Sybase, FrontBase, DB2, SAP DB, SQLite, Netezza, LDAP, y los genéricos ODBC, ODBTP.

Criterios para la selección de los sistemas para la abstracción a la Base de Datos.

La investigación realizada trae como resultados preliminares que la extensión PDO es la indicada, teniendo como premisa que se necesita rapidez para procesar e introducir en la base de datos los más de 32000 paquetes que trae un repositorio común de Ubuntu o Debian. En pruebas realizadas a estas librerías, las consultas de inserciones desde 3000 a 53 000 registros y selecciones con la misma cantidad, PDO fue más rápido y con menos consumo de CPU, aunque no se aprecia gran ventaja sobre Doctrine. Sin embargo, el autor considera la pertinencia de la variante de Doctrine ORM, ya que permite abstraer el sistema del gestor de bases de datos que se vaya a usar.

PDO se especializa en el trabajo con los sistemas gestores de bases de datos; esto quiere decir que se incurriría en un gasto de tiempo enorme para implementar las mismas funcionalidades utilizando cada *driver* de conexión diferente, y con Doctrine las consultas se diseñan utilizando el DQL genérico para la abstracción a las bases de datos. No se pueden descartar las ventajas importantes que brinda Doctrine, como la generación de clases modelo, la protección contra ataques de inyección SQL, entre otras.

1.5.2.4 Servidor de aplicaciones web.

Apache

Apache surge a partir de un servidor web de la *National Center for Super Computing Applications* (NCSA) como corrección a errores y mejoras importantes al producto inicial. Por los resultados logrados, es fundado el Grupo Apache, impulsor del nuevo producto al que se le hicieron mejoras

en su arquitectura logrando modularidad. En 1999 los desarrolladores fundan la *Apache Software Foundation*, organización que da soporte legal al desarrollo de Apache y que lo mantiene actualmente bajo la licencia Apache Licence 2.0, convirtiéndolo en software libre. Apache en su versión 2 es una solución más flexible y escalable, ofreciendo un conjunto de mejoras notables.

Características:

- Multiplataforma y modular.
- Soporte para el protocolo HTTP 1.1.
- Sencillo, con la configuración basada en archivos fáciles de configurar.
- Soporte para CGI (Common Gateway Interface).
- Soporte para FastCGI.
- Soporte de Host Virtuales.
- Soporte de autenticación HTTP.
- Perl integrado.
- Soporte de scritps PHP.
- Soporte de servlets de Java.
- Servidor proxy integrado.
- Estado del servidor y adaptación de registros.
- Soporte de *Server Side Include* (SSI).
- Soporte de *Secured Socket Layer* (SSL).
- Módulos multiproceso.
- Portable en tiempo de ejecución [19].

1.5.2.5 Sistema gestor de base de datos.

PostgreSQL

Es un sistema de gestión de bases de datos relacional orientado a objetos de código abierto, y está publicado bajo la licencia BSD. Es multiplataforma, por lo que funciona perfectamente en los sistemas operativos más populares en la actualidad. Este tiene soporte nativo para muchos lenguajes de programación, entre ellos PHP.

Posee soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos

almacenados (en varios idiomas). Se incluye la mayoría de los tipos de datos de SQL. También soporta almacenamiento de objetos binarios grandes, como imágenes, sonidos o vídeos, y posee una buena documentación.

Es altamente escalable, tanto por la enorme cantidad de datos que puede manejar, como por el número de usuarios concurrentes que puede acomodar. Algunas características son:

- Lenguaje procedural propio denominado PL/PgSQL.
- Usa una arquitectura cliente/servidor de proceso por usuario.
- Utiliza la tecnología *Multi-Version Concurrency Control* (MVCC) para evitar el bloqueo innecesario.
- Es capaz de ajustarse al número de procesadores y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta.
- Soporte para integridad referencial.
- Soporte nativo para SSL.
- Soporte multiusuario.
- Herencia entre tablas.
- Replicación Sincrónica
- Regionalización por columna
- Gran escalabilidad.

1.5.3 Herramientas de desarrollo.

NetBeans

Es un Entorno de Desarrollo Integrado (por sus siglas en ingles IDE) gratuito de código abierto, con soporte para PHP y características visuales para el desarrollo web. Se integra muy bien con sistemas de control de versiones y sistemas gestores de bases de datos como MySQL y PostgreSQL. Funciona sobre plataformas como GNU/Linux, MAC OS, Solaris y Windows. Además de que permite depurar código en varios lenguajes, y reconoce y autocompleta lenguajes como *Java*, *PHP*, *CSS*, *Javascript*, *Python* entre otros.

Rapidsvn

Es un cliente con interfaz gráfica de usuario sencilla e intuitiva para la comunicación con

servidores Subversion. Está disponible para plataformas Windows, GNU/Linux, MAC OS y Solaris, y es distribuido bajo la licencia GNU GPLv3. Es eficiente y está escrito en C++.

Firebug

Es un complemento para el navegador Mozilla Firefox, que permite inspeccionar y editar el código Javascript, XHTML y CSS de una página web, así como analizar las peticiones y respuestas del cliente y servidor, monitorizar el tiempo de carga de las páginas, además de analizar las peticiones vía AJAX. Está distribuido bajo licencia GPL [20].

Visual Paradigm for UML (VP-UML)

Se empleará como herramienta de diseño UML. Este permite modelar los artefactos generados durante todo el desarrollo del proyecto, así como para diseñar los prototipos de interfaz de usuario. Esta herramienta, aunque no es libre constituye la mejor opción para realizar las tareas antes descritas y fue utilizada en su versión para GNU/Linux.

pgAdmin III

Es una herramienta libre para gestionar el gestor de bases de datos PostgreSQL. Está escrita en C++ usando la librería gráfica *wxWidgets* y es multiplataforma. Está diseñado para escribir consultas SQL simples, así como para desarrollar bases de datos complejas. Su interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código para el servidor, un agente para lanzar *scripts* programados, soporte para el motor de replicación Slony-I, entre otros. La conexión al servidor puede hacerse mediante conexión TCP/IP o *Unix Domain Sockets* (en plataformas *nix), y puede encriptarse mediante SSL para mayor seguridad.

CAPÍTULO # 2: DEFINICIÓN DEL SISTEMA PROPUESTO.

Una vez definida la metodología, herramientas, lenguajes y tecnologías a utilizar en la solución propuesta, se debe proceder a definir la misma. Para esto, en el presente capítulo se definirán los RF y no funcionales en la LRP. Además de reflejar en las HU las funcionalidades del sistema, con sus prototipos de interfaces gráficas de usuario no funcionales. Se plasmarán los roles que van a existir en la solución propuesta, así como la descripción de la arquitectura y el diseño.

2.1 Solución propuesta

REPPER es un sistema a partir del cual se podrá crear repositorios personalizados para GNU/Linux Nova y otras distribuciones basadas en paquetería **.deb**. Para la personalización de los mismos, el sistema permitirá que los usuarios realicen la búsqueda y selección de los paquetes que deseen, así como la adición de repositorios que no se encuentren registrados.

Este, brindará la posibilidad de utilizar proyectos realizados anteriormente por el usuario o compartidos por otros. Además, dará la opción de crear nuevos proyectos especificándole el nombre, si será compartido o no, y la exclusión o inclusión de algunos paquetes a partir de un fichero.

Con el propósito de definir las características necesarias para el desarrollo del sistema antes mencionado, a continuación se recogen en la LRP, los requisitos que este debe cumplir.

2.2 Lista de Reserva de Producto

Es una colección organizada y priorizada de los requisitos del producto. El objetivo de la misma radica en cubrir las cualidades requeridas por el *software*. Además de determinar el orden en que se le irá dando cumplimiento durante las iteraciones o sprint[1] a cada requerimiento recogido según la prioridad establecida en cada uno. También abarca las características que el producto debe poseer [21].

A continuación se enuncia en cada una de las filas de la tabla mostrada los elementos que describen cada requisito identificado. Se recoge además el nombre del desarrollador que dará cumplimiento al requisito, el número identificativo, la descripción, el tiempo estimado para su cumplimiento y el nombre del estimador respectivamente.

1 *Ciclo iterativo o período de tiempo en el que se implementa o mejora una funcionalidad del sistema. Durante un sprint el producto puede ser diseñado, implementado y probado para producir nuevos incrementos.*

Requisitos Funcionales

Prioridad	Ítem *	Descripción	Estimación	Estimado por
Muy Alta				
	1	Adicionar Usuario.	2 días	Lara
	2	Listar Usuarios.	2 días	
	3	Eliminar Usuario.	1 día	
	4	Modificar Usuario.	2 días	
	5	Mostrar Usuario.	1 día	
	6	Adicionar repositorio.	7 días	
	7	Eliminar repositorio.	1 día	
	8	Mostrar repositorio.	1 día	
	9	Modificar repositorio.	1 día	
	10	Verificar disponibilidad del repositorio.	3 días	
	11	Adicionar proyecto.	9 días	
	12	Mostrar proyecto.	2 días	
	13	Modificar proyecto.	1 día	
	14	Eliminar proyecto.	1 día	
	15	Listar proyectos	2 días	
	16	Adicionar permiso.	2 días	
	17	Mostrar permisos.	2 días	
	18	Modificar permiso.	2 días	
	19	Eliminar permiso.	1 día	
	20	Asignar permiso a rol.	7 días	
	21	Mostrar permiso de un rol.	1 día	
	22	Eliminar permiso de un rol.	2 días	
	23	Adicionar rol.	2 días	
	24	Eliminar rol.	1 día	
	25	Modificar rol.	2 días	
	26	Listar roles.	1 día	

	27	Adicionar sección personalizada.	2 días	
	28	Eliminar secciones personalizadas.	1 día	
	29	Editar sección personalizada.	1 día	
	30	Listar secciones personalizadas.	2 días	
	31	Adicionar secciones personalizadas para secciones del repositorio.	7 días	
	32	Eliminar secciones personalizadas para secciones del repositorio.	1 día	
	33	Modificar secciones personalizadas para secciones del repositorio.	2 días	
	34	Mostrar secciones personalizadas dada una distribución y su respectiva versión.	7 días	
	35	Construir repositorio de paquetes personalizado.	7 días	
	36	Mostrar repositorios personalizados.	1 día	
	37	Descargar repositorios personalizados.	1 día	
	38	Buscar dependencias dado un paquete.	7 días	
Alta				
	39	Buscar paquete.	1 semana	
	40	Mostrar paquete de una sección personalizada.	1 día	
Media				
	41	Mostrar paquetes por proyecto.	1 día	
	42	Adicionar paquete a un proyecto.	1 día	

	43	Eliminar paquete y dependencias de un proyecto.	2 días	
	44	Mostrar paquete.	1 día	
	45	Iniciar sesión.	7 días	
	46	Cerrar sesión.	1 día	
RNF (Requisitos No Funcionales)				
Lara	47	Utilizar como servidor web Apache y servidor de bases de datos Postgresql.		Lara
	48	El sistema deberá funcionar sobre el SO GNU/Linux “Nova”.		
	49	El sistema será liberado bajo la licencia establecida por la UCI.		
	50	Interfaz sencilla e intuitiva desde la cual se acceda a todas las funcionalidades del sistema.		
	51	Lenguaje de programación PHP**.**		
	52	Frameworks jQuery.		
	53	Disponer para el SO GNU/Linux de los siguientes paquetes: • apache2 • php5 • php5-curl • php5-cli • dpkg-dev		
	54	Deberá funcionar en un entorno de red con prestaciones aceptables.		

Tras haber sido definidos los RF y no funcionales de la solución propuesta, deben realizarse las HU para especificar el modo en que se efectuarán las funcionalidades del sistema.

2.3 Historias de Usuario.

Las HU son la técnica utilizada en XP para especificar los requisitos del *software*, lo que equivale a los casos de uso en el proceso unificado. Las mismas son escritas por los clientes y su construcción depende principalmente de la habilidad que tenga estos para definirlas. Son utilizadas como el único documento de requisitos que se genera en XP. Son escritas en lenguaje natural, sin un formato predeterminado, no excediendo su tamaño de unas pocas líneas de texto; estas guían la construcción de las pruebas de aceptación y son utilizadas para estimar tiempos de desarrollo.

HU Autenticar

Historia de Usuario	
Número: *01*	**Nombre Historia de Usuario:** Autenticar
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: Adrián Lázaro Lara Pérez	**Iteración Asignada:** *2*
Prioridad en Negocio: *Muy Alta*	**Puntos Estimados:** *1 semana*
Riesgo en Desarrollo: *Bajo*	**Puntos Reales:** *1 semana*
Descripción: Se manejan las sesiones de los usuarios, donde una sesión representa un usuario autenticado. Se permite iniciar una sesión, finalizar sesión, verificar si una sesión está activa y verificar si el usuario tiene permiso de realizar una acción.	

Observaciones:

- Para iniciar una sesión se requiere un usuario y una contraseña, se verificará que el usuario dado exista en la base de datos y que la contraseña almacenada coincida con la que se provee. Si no existe, se lanzará un error, en caso de existir, se deberá verificar si ya tiene una sesión iniciada, en cuyo caso se retornará el identificador de dicha sesión y el rol al que pertenece. De no tener sesión iniciada se le creará una, se almacenará la fecha de creación y el id del usuario al que representa este objeto sesión.
- Para finalizar una sesión solo se requiere eliminar el registro de la sesión en la base de datos.
- Para verificar si una sesión esta activa se deberá chequear que sea válida.

- Para verificar si un usuario tiene permiso de realizar una acción se necesita el recurso, el id de la sesión y la acción a realizar.

*Una sesión es válida si no ha transcurrido más de un tiempo X, si una sesión no es válida deberá ser eliminada automáticamente por el sistema.

*Existe un administrador que tiene acceso a todos los recursos.

Prototipo de interfaz:

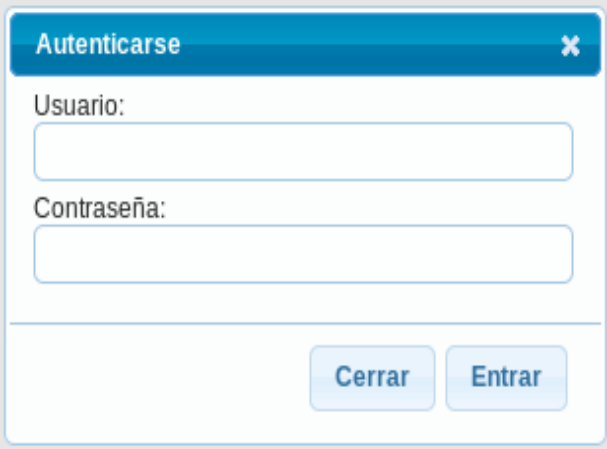

Ilustración 1: Prototipo de "Iniciar sesión".

HU Gestionar usuario

	Historia de Usuario
Número: *02*	**Nombre Historia de Usuario:** *Gestionar usuario*
Modificación de Historia de Usuario Número: *1*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** *2*
Prioridad en Negocio: Muy *Alta*	**Puntos Estimados:** *1 semana*
Riesgo en Desarrollo: *Medio*	**Puntos Reales:** *1 semana*
Descripción: Se gestionan los usuarios que interactúan con el sistema, permitiendo mostrar, adicionar, eliminar y modificar cada uno de ellos.	

Observaciones:

- Para adicionar un usuario deben completarse los siguientes campos: nombre, apellidos, nombre del usuario, contraseña.
  - ✔ La contraseña tiene que ser introducida dos veces para verificar su igualdad. La longitud de esta debe exceder de 5 caracteres.
  - ✔ Al campo contraseña se le aplica la función resumen md5, antes de almacenarlo.
- Para actualizar se pueden modificar todos los campos anteriores excepto el nombre de usuario, y el identificador que se autogenera cuando se adiciona.
- Para mostrar los usuarios se debe tener un paginado de manera que la lista visualizada no sea demasiado extensa.

Prototipos de interfaz:

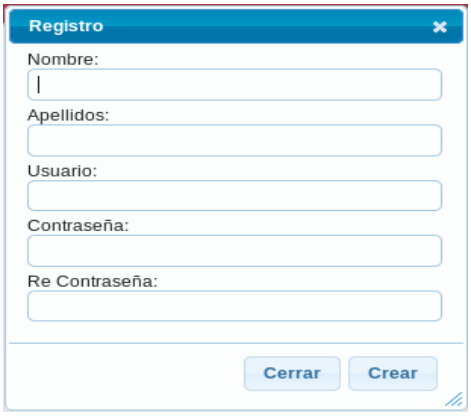

Ilustración 2: Prototipo de "Insertar Usuario".

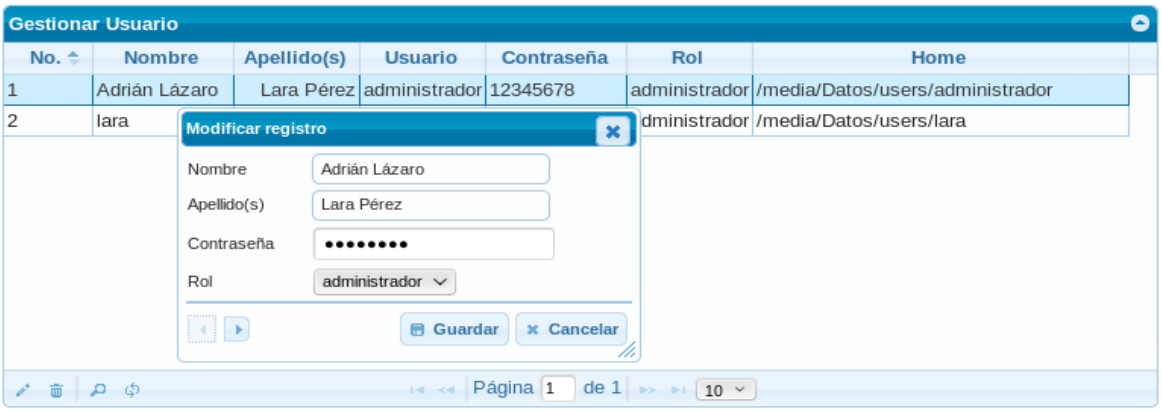

Ilustración 3: Prototipo de "Mostrar Usuario"

HU Gestionar repositorios

	Historia de Usuario
Número: *03*	**Nombre Historia de Usuario:** Gestionar repositorios.
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: Adrián Lázaro Lara Pérez	***Iteración Asignada:*** *2*
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** *2 semanas*
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2 *semanas*
Descripción: Se gestionan los repositorios binarios y fuentes del sistema, permitiendo mostrar, adicionar, eliminar y modificar cada uno de ellos.	

Observaciones:

- Para adicionar una entidad (repositorio) deben completarse los siguientes campos: nombre, distribución, versión, arquitectura, url, componentes y descripción.
- Se podrá modificar el estado, el nombre y la descripción.
- Se podrá verificar el estado de los repositorios y se clasificarán en activos e inactivos.
- Para eliminar un repositorio, se deberán marcar como inactivos los proyectos que contengan este repositorio.
- Para mostrar los repositorios se debe tener un paginado de manera que la lista visualizada no sea demasiado extensa y además debe poderse mostrar los detalles de un repositorio en específico.

Prototipo de interfaz:

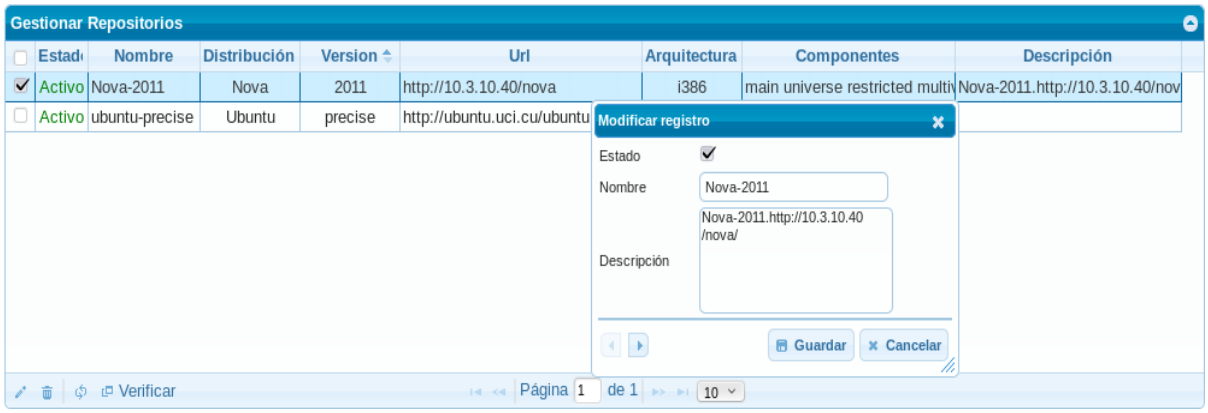

Ilustración 4: Prototipo de "Mostrar Repositorios"

Ilustración 5: Prototipo de "Adicionar Repositorio"

HU Gestionar proyecto

	Historia de Usuario
Número: *04*	**Nombre Historia de Usuario:** *Gestionar Proyecto.*
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** 2
Prioridad en Negocio: Muy *Alta*	**Puntos Estimados:** *2 semanas*
Riesgo en Desarrollo: *Muy Alto*	**Puntos Reales:** *2 semanas*

Descripción: Se gestionan los proyectos creados, permitiendo mostrar, adicionar, eliminar y modificar cada uno de ellos.

Observaciones:

- Para adicionar un proyecto debe completarse el campo nombre, se deberá escoger cual va ser el servidor base, tendrá la opción de compartir su proyecto con los demás usuarios y de excluir paquetes de su proyecto.
  - ✔ Los paquetes a excluir se generarán con el comando **dpkg-query -W -f='${Package} ${Version}\t\n' > dirguardar/paquetes.txt**
- Solo se podrá modificar el nombre y la descripción.
- Para mostrar los proyectos se debe tener un paginado de manera que la lista visualizada no sea demasiado extensa y también debe poderse mostrar los detalles de un repositorio en específico.

Prototipos de interfaz:

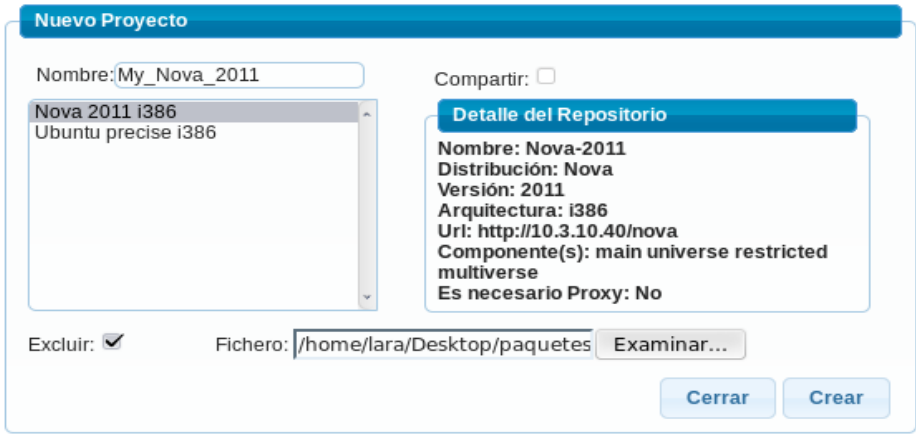

Ilustración 6: Prototipo de "Adicionar Proyecto"

Ilustración 7: Prototipo de "Mostrar Proyecto"

HU Gestionar permiso

Historia de Usuario	
Número: *05*	**Nombre Historia de Usuario:** *Gestionar permisos*
Modificación de Historia de Usuario Número: ninguna	
Usuario: Adrián Lázaro Lara Pérez	**Iteración Asignada:** *2*
Prioridad en Negocio: Alto	**Puntos Estimados:** *1 semana*
Riesgo en Desarrollo: *Bajo*	**Puntos Reales:** *1 semana*
Descripción: Se gestionan las acciones del sistema, permitiendo adicionar, eliminar y modificar cada una de ellas.	

Observaciones:

- Para adicionar una acción debe completarse el campo de nombre.
- Solo se podrá modificar el nombre.

Prototipo de interfaz:

Ilustración 8: Prototipo de "adicionar, modificar y eliminar permiso".

HU Gestionar Rol

Historia de Usuario	
Número: *06*	**Nombre Historia de Usuario:** Gestionar Rol
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** *2*
Prioridad en Negocio: Bajo	**Puntos Estimados:** *1 semana*
Riesgo en Desarrollo: *Bajo*	**Puntos Reales:** *1 semana*
Descripción: Se gestionan los roles del sistema.	

Observaciones:

- Se debe completar el campo de nombre.
- Solo se podrá modificar el campo antes especificado.

Prototipo de interfaz:

Ilustración 9: Prototipo de " adicionar, modificar y eliminar rol".

HU Asociar permisos a un rol

	Historia de Usuario
Número: *07*	**Nombre Historia de Usuario:** Asociar permisos a un rol.
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** *2*
Prioridad en Negocio: *Muy Alto*	**Puntos Estimados:** 2 *semanas*
Riesgo en Desarrollo: *Medio*	**Puntos Reales:** 1 *semana y 3 días*
Descripción: Permite especificar los permisos de cada rol.	

Observaciones:

- Se debe seleccionar un rol y añadirle los permisos correspondientes. El permiso debe ser arrastrado hasta la *ventana de permisos por rol* para adicionarlo a la l*ista de permisos permitidos por rol.*

Prototipo de interfaz:

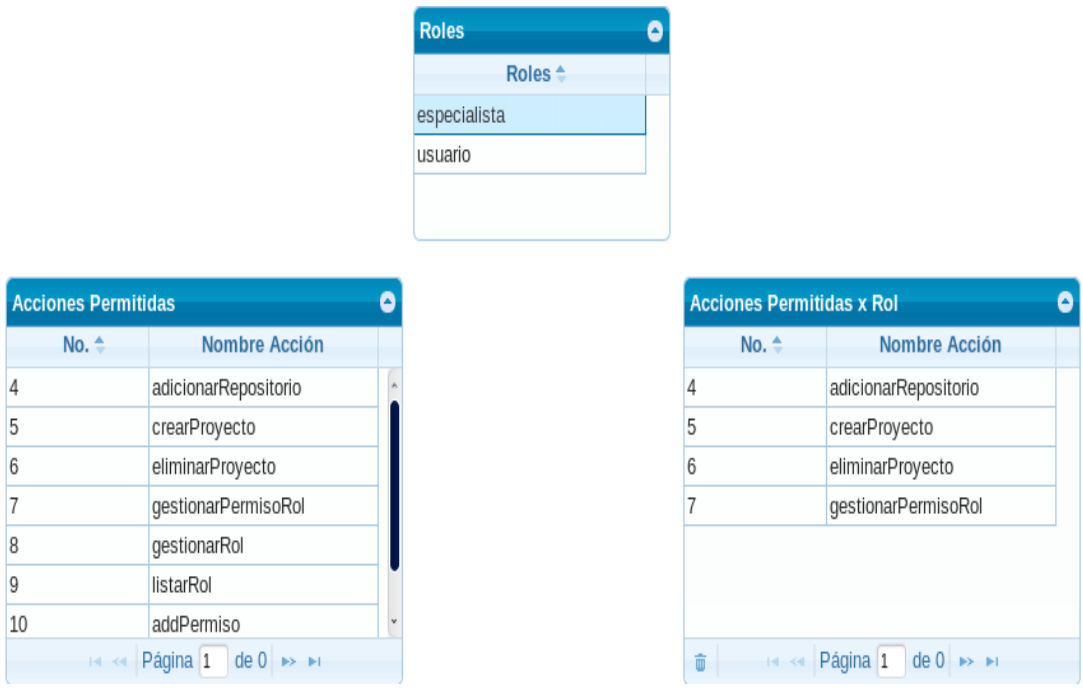

Ilustración 10: Prototipo de "asociar permisos a un rol"

HU Gestionar secciones personalizadas

Historia de Usuario	
Número: *08*	**Nombre Historia de Usuario:** Gestionar secciones personalizadas.
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** *2*
Prioridad en Negocio: *Alto*	**Puntos Estimados:** *1 semana*
Riesgo en Desarrollo: *Medio*	**Puntos Reales:** *1 semana*
Descripción: Se definen qué secciones de paquetes se le mostrarán al usuario. Ejemplos de estos tenemos *Internet, Juegos, Librerías y Otros*.	

Observaciones:

- Se debe completar el campo nombre y se escoge el tipo de sección.
- Solo se podrán modificar el nombre y el tipo de sección.
  - ✔ Los tipos pueden ser SE (Secciones especialista) y SR (Secciones repositorio).

Prototipo de interfaz:

Ilustración 11: Prototipo de " adicionar, modificar y eliminar secciones personalizadas."

HU Asociar sección personalizada

Historia de Usuario	
Número:*09*	**Nombre Historia de Usuario:** Asociar sección personalizada.
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** *2*
Prioridad en Negocio: *Alto*	**Puntos Estimados:** *2 semana*
Riesgo en Desarrollo: *Alto*	**Puntos Reales:** *1 semana y 3 días*
Descripción: Permite especificar que secciones de paquetes de un repositorio previamente añadido contendrá cada sección personalizada.	

Observaciones:

- Se selecciona la sección personalizada, y se escoge que *secciones de paquete* se le va a adicionar o eliminar.

Prototipo de interfaz:

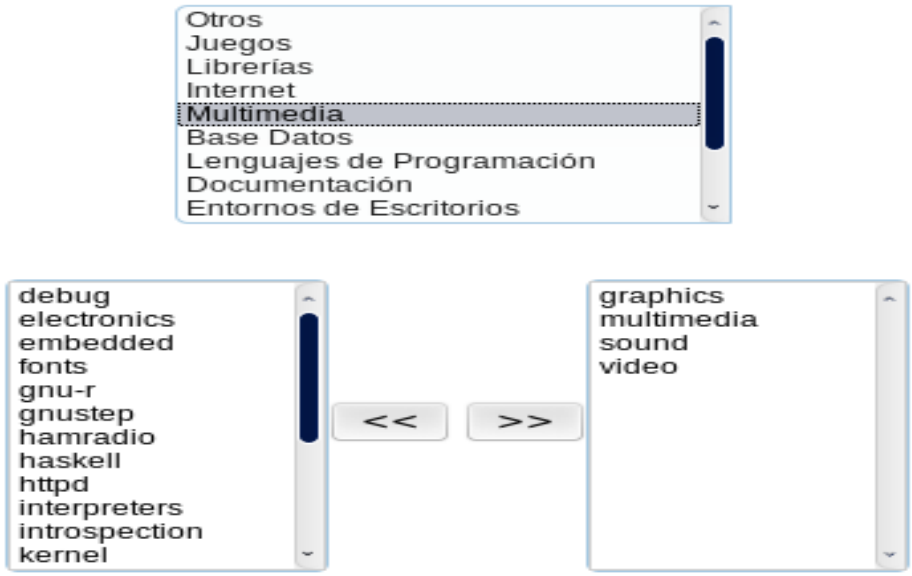

Ilustración 12: Prototipo de "Asociar sección personalizada"

HU Mostrar secciones personalizadas por repositorio

	Historia de Usuario
Número:*10*	**Nombre Historia de Usuario:** Mostrar secciones personalizadas por repositorio.
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** *2*
Prioridad en Negocio: Medio	**Puntos Estimados:** *1 semana*
Riesgo en Desarrollo: *Medio*	**Puntos Reales:** *1 semana*
Descripción: Se visualizan las secciones que contiene el repositorio escogido para crear su repositorio y se muestra la cantidad de paquetes que tiene cada sección.	
Observaciones:	
Prototipo de interfaz:  *Ilustración 13: Prototipo de "m*ostrar secciones personalizadas por repositorio*"*	

HU Generar repositorio personalizado

	Historia de Usuario
Número:*11*	**Nombre Historia de Usuario:** Generar repositorio personalizado
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** *2*
Prioridad en Negocio: *Alto*	**Puntos Estimados:** 2 semana
Riesgo en Desarrollo: *Medio*	**Puntos Reales:** *1 semana y 5 días*

Descripción: Se permite generar un fichero en un formato especificado, con el contenido del repositorio personalizado incluido listo para ser usado.
Observaciones: ➢ Se debe seleccionar un tipo de empaquetado. ➢ Se debe seleccionar para cual dispositivo de almacenamiento se va a crear.
Prototipo de interfaz:

*Ilustración 14: Prototipo de "g*enerar repositorio personalizado"

HU Manejo de paquetes de un proyecto

	Historia de Usuario
Número:*12*	**Nombre Historia de Usuario:** Manejo de paquetes de un proyecto
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** *2*
Prioridad en Negocio: Alto	**Puntos Estimados:** *1 semana*
Riesgo en Desarrollo: *Medio*	**Puntos Reales:** *1 semana*
Descripción: Se permite al usuario mostrar los detalles o eliminar paquetes de su proyecto de repositorio personalizado.	
Observaciones: ➢ Se permite ver los detalles de un paquete. ➢ Se permite eliminar un paquete con sus dependencias.	

Prototipo de interfaz:

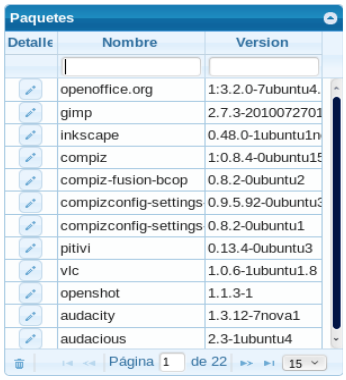

*Ilustración 16: Prototipo de "m*anejo de paquetes de un proyecto*"*

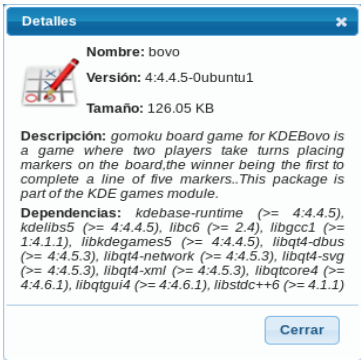

Ilustración 15: Prototipo de "metalles de un paquete"

HU Mostrar paquetes por secciones personalizadas

Historia de Usuario	
Número:*13*	**Nombre Historia de Usuario:** Mostrar paquetes por secciones personalizadas.
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** *2*
Prioridad en Negocio: Medio	**Puntos Estimados:** *1 semana*
Riesgo en Desarrollo: *Medio*	**Puntos Reales:** *2 días*
Descripción: Se muestran los paquetes de una sección personalizada.	

Observaciones:

- Se muestran los detalles de cada paquete y las diferentes versiones disponibles.
- Permite adicionar un paquete y sus dependencias al proyecto creado o cargado.

Prototipo de interfaz:

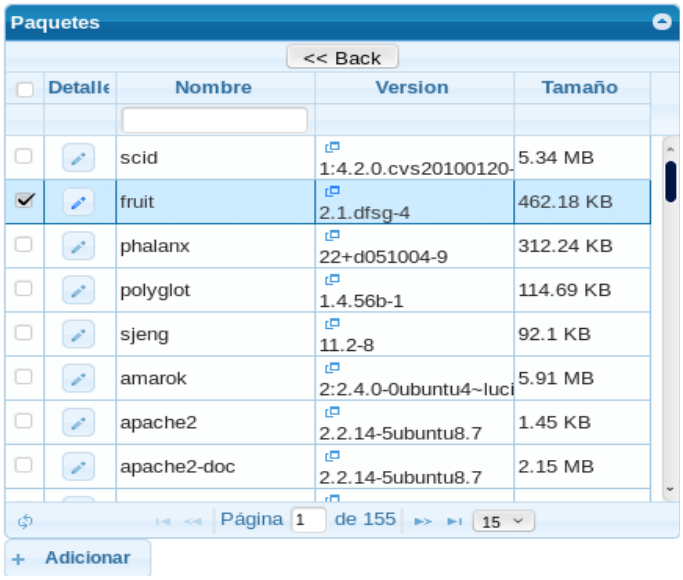

Ilustración 17: Prototipo de "Mostrar paquetes por secciones personalizadas"

HU Descargar repositorio personalizado.

Historia de Usuario	
Número:*14*	**Nombre Historia de Usuario:** Descargar repositorio personalizado.
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** *2*
Prioridad en Negocio: Medio	**Puntos Estimados:** *2 días*
Riesgo en Desarrollo: *Medio*	**Puntos Reales:** *2 días*
Descripción: Se le permite al usuario descargar los repositorios personalizados creados.	

Observaciones:

- Se muestran los repositorios personalizados creados.
- Se le permitirá al usuario eliminar los repositorios creados y descargarlos.

Prototipo de interfaz:

*Ilustración 18: Prototipo de "d*escargar repositorio personalizado*"*

HU Buscar paquetes

	Historia de Usuario
Número:*15*	**Nombre Historia de Usuario:** Buscar paquetes.
Modificación de Historia de Usuario Número: *ninguna*	
Usuario: *Adrián Lázaro Lara Pérez*	**Iteración Asignada:** *2*
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** *1 semana*
Riesgo en Desarrollo: Alta	**Puntos Reales:** *1 semana*
Descripción: Se le permite al usuario hacer búsquedas en el sistema.	

Observaciones:

- Permite hacer búsquedas.
  - ✔ Las búsquedas se hacen por nombre, descripción y versión.

Prototipo de interfaz:

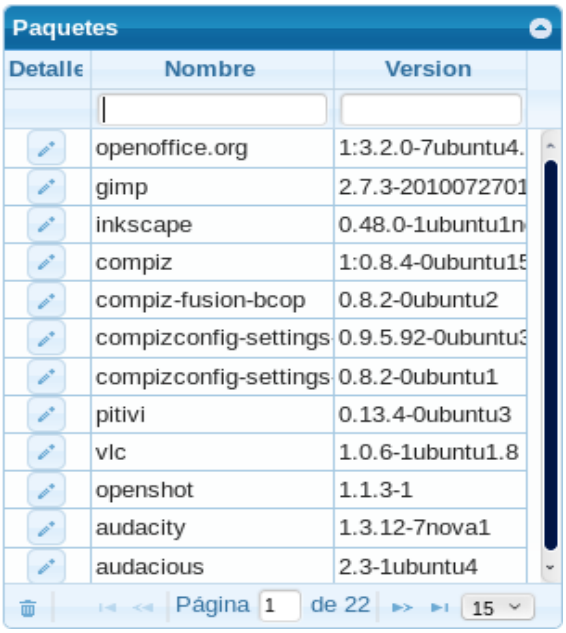

Ilustración 19: Prototipo de "buscar paquetes"

2.4 Roles del sistema

Los usuarios con acceso al sistema, estarán agrupados por roles para el desarrollo de sus funciones dentro del mismo, siendo estos: administrador, especialista y usuario. Los administradores tendrán todos los privilegios necesarios para gestionar las funcionalidades del sistema, en tanto los especialistas tendrán los mismos privilegios que el administrador, con la excepción de que no podrán gestionar roles y permisos. En el caso de los usuarios, solo tendrán permitido adicionar repositorios y crear repositorios personalizados.

Rol	Acciones permitidas
Administrador	Gestionar permisos, proyectos, roles, repositorios, secciones y usuarios. Además de poder efectuar las tareas de mantenimiento dentro del sistema.
Especialista	Puedes efectuar las mismas acciones que el administrador, excepto todas las funciones administrativas sobre permisos y roles.
Usuario	Solo podrá adicionar repositorio y crear repositorio personalizados.

2.5 Descripción de la arquitectura.

2.5.1 Estilos arquitectónicos:

Los estilos arquitectónicos son herramientas para dar forma a la arquitectura de una aplicación. Pressman los define como “una categoría de sistemas que abarca un conjunto específico de: componentes, conectores, restricciones, y modelos semánticos que permiten comprender las propiedades generales de un sistema”. Dicho de otro modo, un estilo arquitectónico pudiera entenderse como un conjunto de principios que definen a un alto nivel un determinado aspecto de un sistema.

Generalmente, en el proceso de construcción de una aplicación, suelen emplearse diversos estilos arquitectónicos de forma combinada. Esto permite una mejor solución mediante el empleo de las ventajas que proporcionan cada uno de ellos. En el caso de *REPPER,* se utiliza el estilo arquitectónico *Arquitectura en Capas (N-Capas o N-Layer)*. Aunque la arquitectura del sistema tiene elementos comunes a otros estilos, es este, el que posee un mayor protagonismo.

En Capas (N-Layer)

Este estilo se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de interactuar con otras capas y las responsabilidades la funcionalidad que implementan.

Características:

- Descomposición de los servicios de forma que la mayoría de las interacciones ocurren solo entre capas vecinas.
- Las capas de una aplicación pueden residir en la misma máquina o pueden estar distribuidas entre varios equipos.
- Los componentes de cada capa se comunican con los componentes de otras capas a través de interfaces bien conocidas.
- Cada nivel agrega las responsabilidades y abstracciones del nivel inferior.
- Separa de forma clara la funcionalidad de cada capa.
- No realiza ninguna suposición sobre los tipos de datos, métodos, propiedades y sus implementaciones.
- Muestra una vista completa del modelo y a la vez proporciona suficientes detalles para entender las relaciones entre capas.

Principios Clave:

- Cada capa contiene la funcionalidad relacionada solo con las tareas de esa capa.
- Las capas inferiores no tienen dependencias de las capas superiores.
- La comunicación entre capas está basada en una abstracción que proporciona un bajo acoplamiento entre capas.

Beneficios:

- Abstracción: los cambios se realizan a alto nivel y se puede incrementar o reducir el nivel de abstracción que se usa en cada capa del modelo.
- Aislamiento: se pueden realizar actualizaciones en el interior de las capas sin que esto afecte al resto del sistema.
- Rendimiento: se puede mejorar la escalabilidad, la tolerancia a fallos y el rendimiento, distribuyendo las capas en distintos niveles físicos.

Cuándo usarlo:

- Ya se tienen construidas capas de una aplicación anterior, que pueden reutilizarse o integrarse.
- La aplicación es compleja y el alto nivel de diseño requiere la separación para que los distintos equipos puedan concentrarse en distintas áreas de funcionalidad.
- Se quiere implementar reglas y procesos de negocio complejos o configurables [22].

La selección de dicho estilo se fundamenta en la necesidad de dividir las capas de acuerdo con su responsabilidad. Para la arquitectura del sistema propuesto se han definido 3 capas:

- **La capa de presentación** (vistas de interfaz): Es la encargada de capturar las solicitudes del cliente y mostrar los resultados de las peticiones. Se comunica con la capa de negocio.
- **La capa de negocio** (lógica del negocio): Recibe las peticiones del usuario y en ella se establecen las reglas del negocio que deben cumplirse. Se comunica con la capa de acceso a datos para el trabajo con la información y con la de presentación para recibir las solicitudes y presentar los resultados.
- **La capa de acceso a datos** (acceso a datos): Es la encargada de acceder a la información. Se comunica con la capa de negocio para recibir las solicitudes y brindarle la(s) respuesta(s) a las mismas.

Ilustración 20: Arquitectura N Layer (3 Capas).

2.5.2 Patrones de diseño:

Los patrones arquitectónicos están presentes claramente dentro de la disciplina arquitectónica, solapándose con los estilos. En tanto los patrones de diseño se encuentran aplicados más bien dentro de una solución de *software* concreta y particular. A continuación se describen los patrones de diseño a emplear en el sistema en cuestión:

- **Patrón Singleton:** Se utilizó al implementar las librerías para que existiese una única instancia de estas o un único acceso global a las mismas.
- **Patrón administración de caché:** Este patrón se utilizó para implementar la librería caché. Esta librería se utiliza para proporcionarle velocidad de respuesta al sistema, lo que se logra realizando búsquedas en la caché antes que en la base de datos.

2.6 Diagrama de clases del diseño.

REPPER un diseño de clases con el uso de estereotipos web, en el que se observan dos páginas servidoras que se relacionan con una única página cliente. En el caso de contenido lo hace a través de llamadas *AJAX*, este a su vez, incluye funcionalidades contenidas en la clase *Enrutador*, que puede considerarse como la controladora en el ámbito del sistema. La clase *Enrutador* incluye varios ficheros PHP y además se encuentra asociada directamente con *Accion*.

La clase *Accion* se encuentra en el paquete *default*, que es una representación genérica de cómo debe diseñarse cada módulo en el sistema, por tanto, se crearán tantas clases *Accion* como módulos tenga el sistema. En cada función que necesite una vista se incluirá un fichero PHP, que tendrá el nombre de la función que la requiera o la incluya; se define de esta manera para un mejor entendimiento y un fácil trabajo con el código. En la Ilustración 21 se observa el diagrama

de clases del diseño de este sistema.

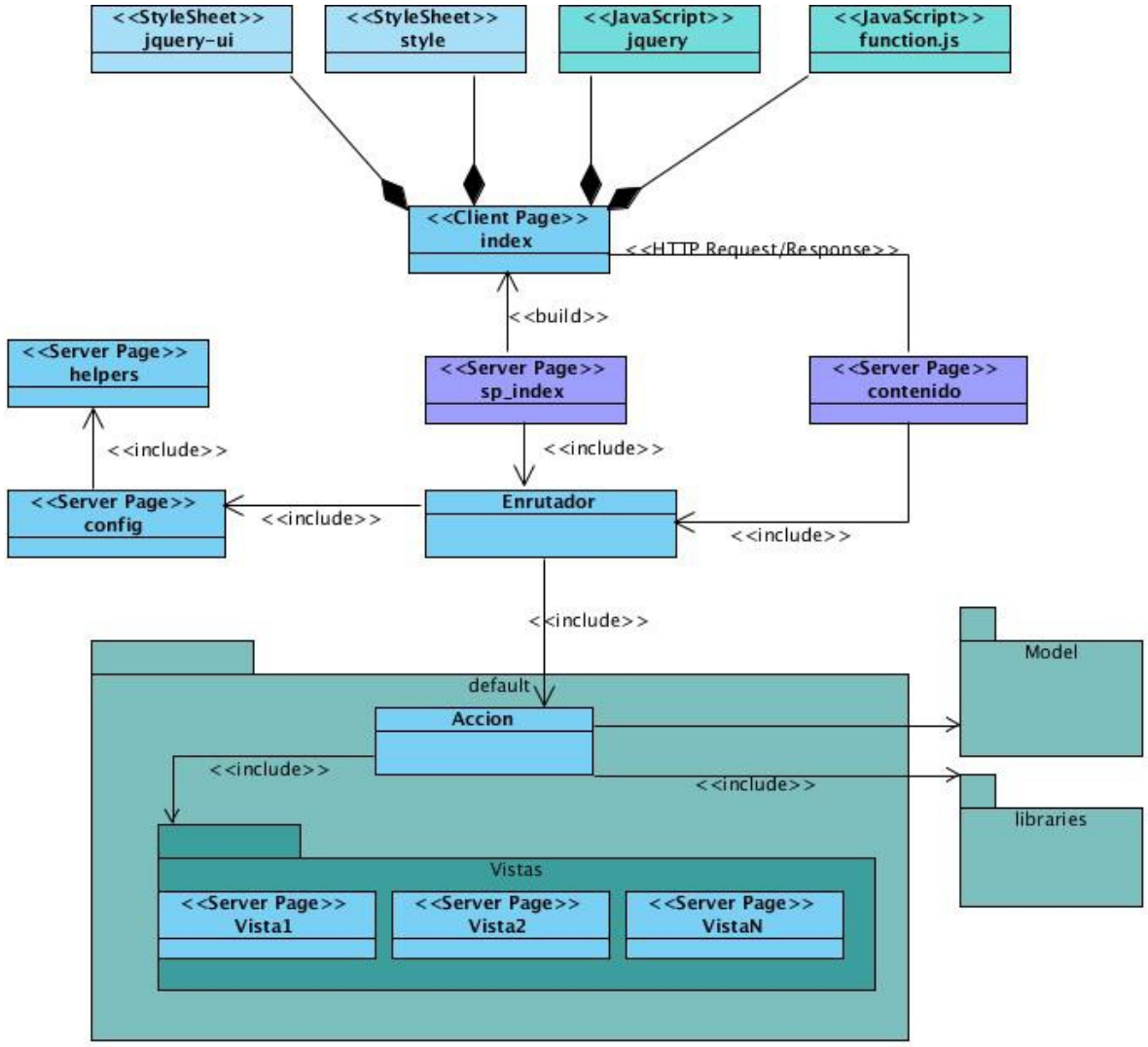

Ilustración 21: Diagrama de Clases del diseño

2.7 Diagrama de paquetes

Para lograr un mayor entendimiento del funcionamiento del sistema, se hace necesario describir su estructura física. Esto se debe a que las mejoras que se le realicen posteriormente, deberán estar acopladas a la estructura que se define, proporcionando uniformidad al sistema en general. La Ilustración 22 muestra cómo quedó organizado el sistema en términos de paquetes (que representan carpetas) y componentes.

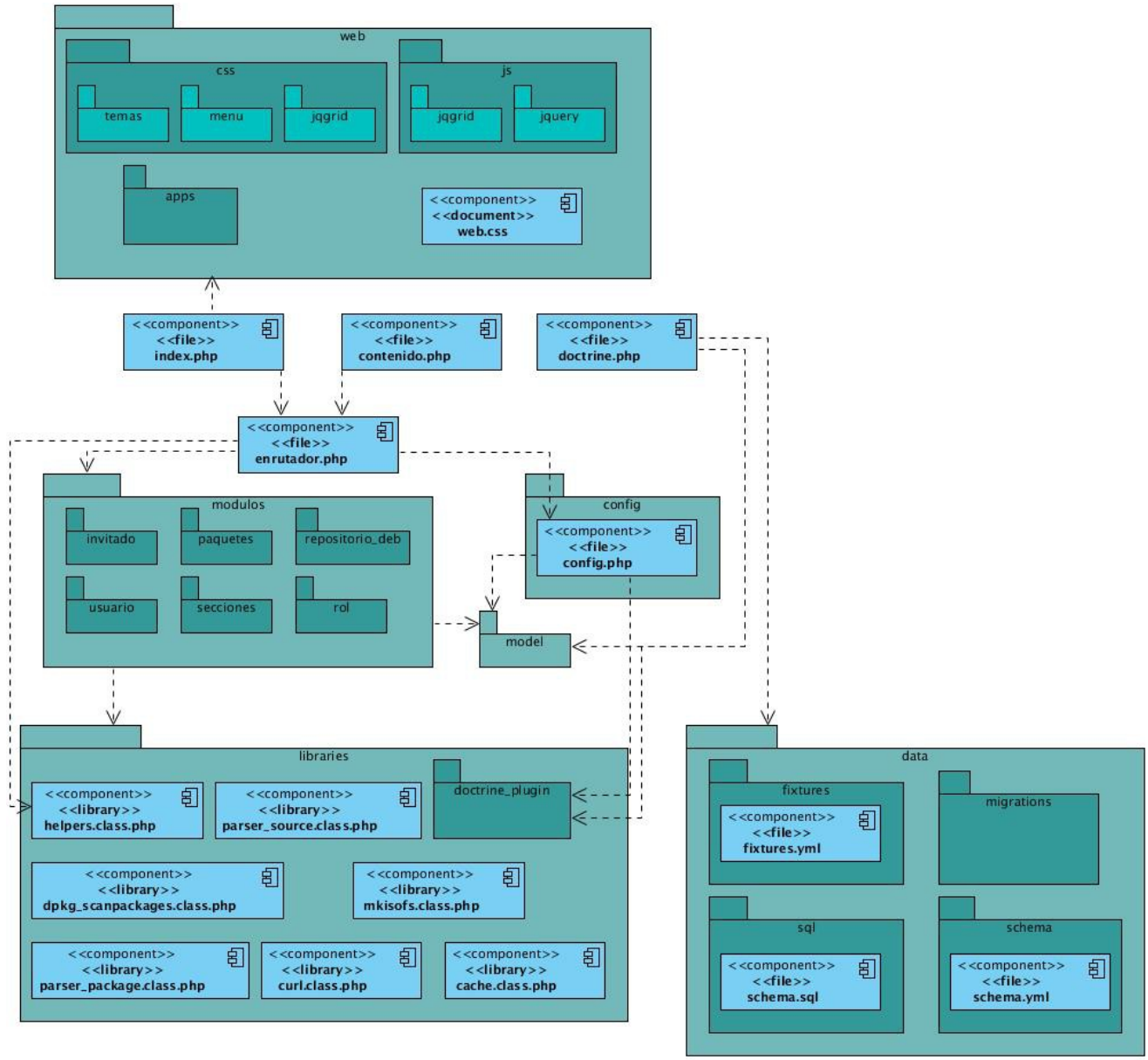

Ilustración 22: Diagrama de Paquetes

La carpeta web abarca los ficheros javascript y hojas de estilos que se emplean en los temas, el menú y en los diferentes complementos de jQuery, los que serán utilizados por la página cliente *index* mencionada en el epígrafe anterior. Las páginas servidoras "*contenido.php*" e "*index.php*" dependen del fichero "*enrutador.php*". Este representa a la clase del mismo nombre, la que a su vez dependerá de los diferentes módulos que existan en la carpeta del mismo nombre y del fichero "*config.php*" que contiene la configuración del sistema.

CAPÍTULO # 3: IMPLEMENTACIÓN Y PRUEBAS.

Para el correcto desarrollo de la solución propuesta, en el presente capítulo se describirá el plan de releases, las TI y el estándar de código a utilizar en la implementación del sistema. Además se expondrán las pruebas realizadas al mismo para ganar en calidad, lo que garantiza que el producto llegue a manos del cliente con el menor número de errores posibles.

Una vez definido el sistema propuesto, se hace necesaria la realización del plan de releases con el propósito de describir en que iteración se va a realizar cada HU, según su prioridad.

3.1 Plan de Releases

De acuerdo a las funcionalidades descritas en las HU y en concordancia con la prioridad asignada para su realización, se planificaron 5 iteraciones que recogen el desarrollo de la solución propuesta.

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total
1	Darle cumplimiento a las HU que tienen la máxima prioridad	*03, 04*	4 semanas
2	Desarrollo de las HU de muy alta prioridad	01, 02, 07	6 semanas
3	Desarrollo de las HU de alta prioridad	05, 08, 09, 11, 12,15	7 semanas
4	Desarrollo de las HU que tienen prioridad media e integrar los componentes desarrollados en la iteración anterior.	10, 13, 14	2 semanas
5	Desarrollo de las HU de prioridad baja e integrar los componentes desarrollados en iteraciones anterior	06	1 semana

3.2 Tareas de Ingeniería

TI asociadas a la HU Autenticar

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 01
Nombre Tarea: Iniciar sesión.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0,1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se personalizan los validadores de formularios y los mensajes de validación. Se utiliza el componente visual de la librería jQuery UI para mostrar el formulario en forma de diálogo.	

TI asociadas a la HU Gestionar usuario

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 02
Nombre Tarea: Registrar usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se personalizan los validadores de formularios y los mensajes de validación. Se utiliza el componente visual de la librería jQuery UI para mostrar el formulario en forma de diálogo.	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** 02
Nombre Tarea: Listar usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se hace uso del plugin de jQuery jqgrid para mostrar la lista de usuarios. Se instala el plugin y se configuran las acciones que se van a permitir, así como el paginador. Se implementan las funcionalidades de listarUsuarios.	

	Tarea de Ingeniería
Número Tarea: 3	**Número Historia de Usuario:** 02
Nombre Tarea: Modificar usuario.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se personalizan los mensajes de validación. Se implementan listas de roles para su selección y se integran al plugin.	

TI asociadas a la HU Gestionar repositorios

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 03
Nombre Tarea: Listar repositorios.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se hace uso del plugin de jQuery jqgrid para mostrar la lista de repositorios. Se instala el plugin y se configuran las acciones que se van a permitir, así como el paginador. Se implementan las funcionalidades de listarRepositorio.	

Tarea de Ingeniería	
Número Tarea: 2	Número Historia de Usuario: 03
Nombre Tarea: Adicionar repositorio.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se personalizan los validadores de formularios y los mensajes de validación. Se utiliza el componente visual de la librería jQuery UI para mostrar el formulario en forma de diálogo.	

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 03
Nombre Tarea: Eliminar repositorio.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se personalizan los validadores de formularios y los mensajes de validación. Se implementa el método encargado de eliminar un repositorio.	

Tarea de Ingeniería	
Número Tarea: 4	**Número Historia de Usuario:** 03
Nombre Tarea: Verificar repositorio.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se personalizan los validadores de formularios y los mensajes de validación. Se implementa la funcionalidad verificarRepositorio.	

TI asociadas a la HU Gestionar proyecto

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 04
Nombre Tarea: Listar proyectos.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se hace uso del plugin de jQuery jqgrid para mostrar la lista de proyectos. Se instala el plugin y se configuran las acciones que se van a permitir, así como el paginador. Se implementan las funcionalidades de listarProyectos y gestionarProyectos.	

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 04
Nombre Tarea: Adicionar proyecto.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se personaliza el formulario de adicionar proyecto, añadiendo validadores de formularios y los mensajes de validación.	

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 04
Nombre Tarea: Eliminar proyecto.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se personalizan los validadores de formularios y los mensajes de validación.	

TI asociadas a la HU Gestionar permiso

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 05
Nombre Tarea: Listar permisos.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se hace uso del plugin de jQuery jqgrid para mostrar la lista de proyectos. Se instala el plugin y se configuran las acciones que se van a permitir, así como el paginador. Se implementan las funcionalidades de listarPermisos.	

TI asociadas a la HU Gestionar Rol

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 06
Nombre Tarea: Listar rol.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se hace uso del plugin de jQuery jqgrid para mostrar la lista de proyectos. Se instala el plugin y se configuran las acciones que se van a permitir, así como el paginador. Se implementan las funcionalidades de listarRoles.	

TI asociadas a la HU Gestionar Rol

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 07
Nombre Tarea: Asociar permiso a un rol	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	

Descripción: Se hace uso del plugin de jQuery jqgrid para mostrar la lista de proyectos. Se instala el plugin y se configura para mostrar múltiples tablas. Se hacen las configuraciones para que soporte el *drag and drop* y las acciones que va a permitir, así como el paginador.

TI asociadas a la HU Gestionar secciones personalizadas

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 08
Nombre Tarea: Listar secciones personalizadas	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se hace uso del plugin de jQuery jqgrid para mostrar la lista de proyectos. Se instala el plugin y se configuran las acciones que se van a permitir, así como el paginador. Se implementan las funcionalidades de listarSecciones y gestionarSecciones.	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** 08
Nombre Tarea: Modificar secciones personalizadas.	
Tipo de Tarea : Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se personalizan los mensajes de validación. Se implementa la lista de tipos de secciones y se integran al plugin.	

TI asociadas a la HU Asociar sección personalizada

	Tarea de Ingeniería
Número Tarea: 1	Número Historia de Usuario: 09
Nombre Tarea: Asociar sección personalizada.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	

Descripción: Se personalizan los validadores de formularios y los mensajes de validación.

TI asociadas a la HU Generar repositorio personalizado

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 11
Nombre Tarea: Generar repositorio personalizado.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se personalizan los validadores de formularios y los mensajes de validación. Se utiliza el componente visual de la librería jQuery UI para mostrar el formulario en forma de diálogo. Verificar que se encuentren instaladas las aplicaciones *mkisofs* y *dpkg-dev*, de no ser así, se deben instalar en el sistema. Se implementan las clases necesarias para crear el *Package.gz* y el comprimido. Se hace necesario implementar una función que permita la escritura de la información del repositorio.	

TI asociadas a la HU Manejo de paquetes de un proyecto

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 12
Nombre Tarea: Listar paquetes	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se hace uso del plugin de jQuery jqgrid para mostrar la lista de proyectos. Se instala el plugin y se configuran las acciones que se van a permitir, así como el paginador. Se implementan las funcionalidades de listarPaquetes.	

	Tarea de Ingeniería
Número Tarea: 2	**Número Historia de Usuario:** 12
Nombre Tarea: Detalles del paquete	

Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0,1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se utiliza el componente visual de la librería jQuery UI para mostrar el formulario en forma de diálogo.	

TI asociadas a la HU Mostrar paquetes por secciones personalizadas

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 13
Nombre Tarea: Listar paquetes	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se hace uso del plugin de jQuery jqgrid para mostrar la lista de proyectos. Se instala el plugin y se configuran las acciones que se van a permitir, así como el paginador. Se implementan las funcionalidades de listarPaquetesSeccion.	

TI asociadas a la HU Buscar paquetes

	Tarea de Ingeniería
Número Tarea: 1	**Número Historia de Usuario:** 15
Nombre Tarea: Buscar paquetes.	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.1
Fecha Inicio:	**Fecha Fin:**
Programador Responsable: Adrián Lázaro Lara Pérez	
Descripción: Se hace uso del plugin de jQuery jqgrid. Se configura el plugin para mostrar las barras de búsqueda. Se implementa la funcionalidad para hacer las búsquedas.	

3.3 Estándar de codificación utilizado

Una de las buenas prácticas propuesta por las metodologías ágiles es la adopción de un estándar de codificación por parte del equipo de desarrollo, asegurando que el código exprese claramente

el propósito del mismo y agilice el proceso de refactorización. No se trata de una imposición, sino de proveerle organización al código, alta calidad, y una cantidad baja de *bugs*, incluso cuando existan diversos usuarios contribuyendo en el desarrollo del producto.

Para dar cumplimiento a la solución propuesta se escogió el estándar recomendado por la empresa Zend, salvo por algunas *excepciones* y adicionándose otras reglas (**Ver anexos I**).

Luego de implementada la solución propuesta, se describen en detalles las pruebas asociadas a las HU, con el fin de verificar que el sistema cumpla con el funcionamiento esperado, permitiendo así su aceptación por parte del cliente.

3.4 Pruebas

En el proceso de desarrollo de *software* suelen existir inconsistencias sobre los requisitos que debe tener el producto para dar cumplimiento a las exigencias del cliente. En ese sentido, las pruebas que se le realizan al *software* permiten ganar en seguridad y calidad. Estas permiten encontrar deficiencias previas a la entrega del sistema, de modo que una vez liberado el producto, posea el menor número de errores posibles. Para la realización pruebas a REPPER, se utiliza el tipo de prueba unitaria y de aceptación, mediante los métodos de caja blanca y .caja negra

Estas pruebas se documentan mediante el artefacto denominado Caso de Prueba de Aceptación, en la que el desarrollador, el cliente y el probador comprueban y validan las funcionalidades del sistema a partir de las HU implementadas, para finalmente decidir la liberación del producto. En este epígrafe se muestran los Casos de Prueba de Aceptación más significativos.

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-02-01	**Nombre Historia de Usuario:** Gestionar usuario
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la Prueba: Prueba a la funcionalidad adicionar usuario, completando todos los datos correctamente.	
Condiciones de Ejecución: Acceder a la interfaz gestionar usuario o al registrar usuario.	
Entrada / Pasos de ejecución: 1. Seleccionar la opción adicionar usuario. 2. Completar los campos con los siguientes valores: ◦ **Nombre:** Adrián Lázaro ◦ **Apellido (s):** Lara Pérez	

◦ **Contraseña:** 123456789 ◦ **Re Contraseña:** 123456789 3. Se presiona el botón Crear.
Resultado Esperado: Debe aparecer la notificación de que los datos fueron insertados correctamente, el usuario agregado debe aparecer en la lista de usuarios con los datos especificados, debe poderse autenticar en el sistema utilizando el usuario y su respectiva contraseña.
Evaluación de la Prueba: *Satisfactoria.*

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-02-02	**Nombre Historia de Usuario:** Gestionar usuario
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la Prueba: Prueba a la funcionalidad editar usuario, completando todos los datos correctamente.	
Condiciones de Ejecución: Haberse autenticado con un usuario que tenga permitido realizar esta acción.	
Entrada / Pasos de ejecución: *1. Seleccionar el usuario a editar y luego presionar la opción editar.* *2. Modificar los siguientes campos:* ✔ ***Nombre:*** *prueba1* ✔ ***Apellido (s):*** *prueba1 prueba1* ✔ ***Contraseña:****prueba123* ✔ ***Rol:*** *usuario* *3. Se presiona el botón Guardar.*	
Resultado Esperado: Se actualiza la lista de usuarios y se evidenciarán los cambios. El usuario no podrá autenticarse en el sitio con la contraseña anterior.	
Evaluación de la Prueba: *Satisfactoria*	

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-03-01	**Nombre Historia de Usuario:** Gestionar Repositorios
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la Prueba: Prueba a la funcionalidad adicionar repositorio, completando todos los datos correctamente.	

Condiciones de Ejecución: Haberse autenticado con un usuario que tenga permitido realizar esta acción. Acceder a la interfaz gestionar repositorio o adicionar repositorio.

Entrada / Pasos de ejecución:

1. Seleccionar la opción adicionar repositorio.
2. Completar los campos con los siguientes valores:
   - ✔ **Nombre:** Nova-2011
   - ✔ **Seleccionar la distribución:** Nova
   - ✔ **Versión:** 2011
   - ✔ **Arquitectura:** i386
   - ✔ **Url:** http://novarepo.uci.cu/nova/
   - ✔ **Componente (s) :** main universe restricted multiverse
   - ✔ **Descripción:** Nova-2011. http://novarepo.uci.cu/nova/.
3. Se comprueba la conexión.
4. Se presiona el botón Crear.

Resultado Esperado: Debe aparecer la notificación de que los datos fueron insertados correctamente. Se adiciona a la lista de repositorios marcado como activo y aparece en la lista de repositorios para crear proyectos.

Evaluación de la Prueba: Satisfactoria

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-03-02	**Nombre Historia de Usuario:** Gestionar Repositorios

Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón

Descripción de la Prueba: Prueba a la funcionalidad editar repositorio, completando todos los datos correctamente.

Condiciones de Ejecución: Haberse autenticado con un usuario que tenga permitido realizar esta acción. Acceder a la interfaz gestionar repositorio.

Entrada / Pasos de ejecución:

1. *Seleccionar el repositorio a editar y luego presionar la opción editar.*
2. *Completar los campos con los siguientes valores:*
   - ✔ *Estado: No (desactivar checkbox)*
   - ✔ *Nombre:Ubuntu-oneiric*
   - ✔ *Descripción: Repositorio de ubuntu*
3. *Se presiona el botón Guardar.*

Resultado Esperado: Se actualiza la lista de usuarios y se evidenciarán los cambios. Se marca

como inactivo el repositorio y no aparece en la lista de repositorios para crear proyectos.
Evaluación de la Prueba: Satisfactoria

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-04-01	**Nombre Historia de Usuario:** Gestionar proyecto
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la Prueba: Prueba a la funcionalidad adicionar proyecto, completando todos los datos correctamente.	
Condiciones de Ejecución: Haberse autenticado con un usuario que tenga permitido realizar esta acción. Acceder a la interfaz ver proyectos o adicionar proyecto.	
Entrada / Pasos de ejecución: 1. Seleccionar la opción adicionar proyecto 2. .Completar los campos con los siguientes valores: ✔ **Nombre:** My_repositorio_prueba ✔ **Compartir:** Si ✔ **Seleccionar Repositorio:** Nova 2011 i386 ✔ **Descripción:** Repositorio de caso de prueba de aceptación 3. Se presiona el botón Crear.	
Resultado Esperado: Debe aparecer la notificación de que el proyecto fue creado exitosamente, lo debe redirigir a la vista de secciones. Se debe adicionar a la lista de proyectos, marcado como activo y mostrarse a todos los usuarios.	
Evaluación de la Prueba: Satisfactoria	

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-04-02	**Nombre Historia de Usuario:** Gestionar proyecto
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la Prueba: Prueba a la funcionalidad editar proyecto, completando todos los datos correctamente.	
Condiciones de Ejecución: Haberse autenticado con un usuario que tenga permitido realizar esta acción. Acceder a la interfaz gestionar repositorio.	
Entrada / Pasos de ejecución: 1. Seleccionar el proyecto a editar y luego presionar la opción editar. 2. Completar los campos con los siguientes valores: ✔ **Nombre:** Repositorio Prueba.	

✔ **Descripción:** Descripción repositorio prueba. 3. Se presiona el botón Guardar.
Resultado Esperado: Se actualiza la lista de usuarios y se evidenciarán los cambios.
Evaluación de la Prueba: Satisfactoria

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-05-01	**Nombre Historia de Usuario:** *Gestionar permisos*
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la Prueba: Prueba a la funcionalidad adicionar permiso, completando todos los datos correctamente.	
Condiciones de Ejecución: Haberse autenticado con un usuario que tenga permitido realizar esta acción. Acceder a la interfaz gestionar permiso.	
Entrada / Pasos de ejecución: 1. Seleccionar la opción adicionar permiso 2. .Completar los campos con los siguientes valores: ✔ **Nombre Acción:** adicionarRepositorio 3. Se presiona el botón Guardar.	
Resultado Esperado: Se actualiza la lista de permisos y se evidenciarán los cambios. El nuevo permiso aparecerá en la lista de permisos.	
Evaluación de la Prueba: Satisfactoria	

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-05-02	**Nombre Historia de Usuario:** *Gestionar permisos*
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la Prueba: Prueba a la funcionalidad editar permiso, completando todos los datos correctamente.	
Condiciones de Ejecución: Haberse autenticado con un usuario que tenga permitido realizar esta acción. Acceder a la interfaz gestionar permiso.	
Entrada / Pasos de ejecución: 1. Seleccionar el permiso a editar y luego presionar la opción editar. 2. Completar los campos con los siguientes valores: ✔ **Nombre Acción:** addRepositorio 3. Se presiona el botón Guardar.	
Resultado Esperado: Se actualiza la lista de permisos y se evidenciarán los cambios.	

Evaluación de la Prueba: Satisfactoria

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-07-01	**Nombre Historia de Usuario:** Asociar permisos a un rol
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la Prueba: Prueba a la funcionalidad asociar permisos a un rol, completando todos los datos correctamente.	
Condiciones de Ejecución: Haberse autenticado con un usuario que tenga permitido realizar esta acción. Acceder a la interfaz asociar permisos.	
Entrada / Pasos de ejecución: 1. Seleccionar el rol al que se le va a dar los permisos: usuario 2. Seleccionar el permiso que se le va a dar: addRepositorio, vistaAddRepositorio 3. Se arrastra a las acciones permitidas por rol.	
Resultado Esperado: El usuario que pertenezca al rol usuario podrá añadir repositorios al sistema.	
Evaluación de la Prueba: Satisfactoria	

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-08-01	**Nombre Historia de Usuario:** Gestionar secciones personalizadas
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la Prueba: Prueba a la funcionalidad adicionar sección personalizada, completando todos los datos correctamente.	
Condiciones de Ejecución: Haberse autenticado con un usuario que tenga permitido realizar esta acción. Acceder a la interfaz gestionar secciones.	
Entrada / Pasos de ejecución: 1. Seleccionar la opción adicionar sección 2. Completar los campos con los siguientes valores: ✔ **Tipo :** SR[2] ✔ **Nombre sección:** Juegos 3. Se presiona el botón Guardar.	
Resultado Esperado: Se actualiza la lista de secciones personalizadas y se evidenciarán los cambios.	
Evaluación de la Prueba: Satisfactoria	

2 Secciones de repositorio

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-08-02	**Nombre Historia de Usuario:** Gestionar secciones personalizadas
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la Prueba: Prueba a la funcionalidad adicionar sección personalizada, completando todos los datos correctamente.	
Condiciones de Ejecución: Haberse autenticado con un usuario que tenga permitido realizar esta acción. Acceder a la interfaz gestionar secciones.	
Entrada / Pasos de ejecución: 1. Seleccionar la sección personalizada a editar, y luego presionar la opción editar. 2. Completar los campos con los siguientes valores: ✔ **Tipo:** SR ✔ **Nombre sección:** Internet 3. Se presiona el botón Guardar.	
Resultado Esperado: Se actualiza la lista de secciones personalizadas y se evidenciarán los cambios.	
Evaluación de la Prueba: Satisfactoria	

Caso de Prueba de Aceptación	
Código Caso de Prueba: REPPER-09-01	**Nombre Historia de Usuario:** Asociar sección personalizada
Nombre de la persona que realiza la prueba: Yoandy Pérez Villazón	
Descripción de la Prueba: Prueba a la funcionalidad asociar sección personalizada, completando todos los datos correctamente.	
Condiciones de Ejecución: Haberse autenticado con un usuario que tenga permitido realizar esta acción. Acceder a la interfaz asociar sección.	
Entrada / Pasos de ejecución: 1. Seleccionar sección personalizada que se desea asociar: Juegos 2. Seleccionar sección de paquete que se le va a asociar: games 3. Se presiona el botón (>>) para asociar *Juegos* con *games*.	
Resultado Esperado: Se actualizan las listas y se evidenciarán los cambios. Al crear un proyecto se mostrarán las secciones de ese repositorio, al seleccionar la sección Juegos se deben mostrar los paquetes que pertenecen a esta.	
Evaluación de la Prueba: Satisfactoria	

Resultados de las pruebas

Las pruebas diseñadas se realizaron a lo largo de todo el ciclo de desarrollo del sistema. Como resultado, se obtuvo un producto que cumple las especificaciones del cliente. La ejecución de estas contribuyó a facilitar el desempeño de los desarrolladores, sirviendo de apoyo para la realización de las HU y para localizar los errores que a simple vista no son detectados cuando se está implementando. Esto permitió la corrección de deficiencias antes de la liberación del sistema, posibilitando que se hiciera una mejor distribución del tiempo de trabajo. De esta forma, se obtiene una herramienta cuya funcionalidad ha sido comprobada antes de ponerse en manos de los usuarios finales, lo que garantiza la confiabilidad en la aplicación y la obtención de resultados satisfactorios durante su uso.

3.5 Impacto de la solución propuesta

Toda investigación o producto que tenga como objetivo solventar un problema o necesidad, debe tener un impacto, ya sea social, político o económico. La reacción que este produzca y su aporte al desarrollo de la humanidad demuestran su importancia. Al término de la presente investigación, se obtuvo una aplicación web libre, que automatiza algunas de las tareas realizadas en el departamento de Migración y Soporte, permitiendo añadir un nuevo servicio a la lista de los que ya provee dicho departamento.

El sistema propuesto es útil para cualquier persona que necesite crear un repositorio personalizado. Entre sus ventajas se encuentra la agilización del proceso de migración en las empresas y organismos que llevan a cabo el mismo. También causará decremento del tiempo y personal destinado a la creación de repositorios para las empresas a las que se les brinda servicio. Este, implica un gran ahorro económico, ya que no se necesitarán grandes dispositivos de almacenamiento para el traslado de los repositorios creados. El sistema es competitivo frente a otras aplicaciones, ya que el usuario solamente necesita de la disponibilidad de un navegador web, y no es necesario tener instalado un sistema GNU/Linux con los repositorios previamente configurados.

Conclusiones

En la presente investigación se realizó un estudio del estado del arte de los sistemas para la creación de repositorios personalizados, permitiendo sintetizar y documentar sus principales características y desventajas. Este estudio permitió determinar la necesidad de desarrollar un sistema para la creación de repositorios en línea que cumpliera con las especificidades del cliente, y de esta manera erradicar una de las principales deficiencias que existen en el proceso de migración a software libre y código abierto, llevado a cabo en nuestro país.

Se identificaron las principales herramientas, tecnologías, y lenguajes de programación usados en la construcción de aplicaciones web dinámicas. El proceso de desarrollo de dicho sistema fue guiado por la metodología ágil SXP.

Finalmente, se obtiene un producto completamente libre, a través de un ciclo de desarrollo ágil, permitiendo responder rápidamente a cambios en los requisitos y logrando niveles aceptables de confiabilidad y calidad en el producto obtenido, a partir del diseño y ejecución de un conjunto de pruebas realizadas al culminar cada nueva funcionalidad.

Al concluir las pruebas, se obtiene un producto con el menor número de errores posibles, que cumple con el concepto de *software* como servicio y permite la creación de repositorios personalizados en línea. Contribuyendo así a la automatización de una de las principales tareas llevadas a cabo por el departamento de Migración y Soporte en la migración a aplicaciones de código abierto en el país.

Recomendaciones

Al término de la presente investigación, se recomienda:

- Reimplementar la clase que realiza la operación de parseo del Package.gz utilizando programación paralela.
- Implementar servicios web para que pueda ser integrada a la Plataforma de Migración.

Referencias bibliográficas.

[1]. Ger. Software libre y gobierno Los países más Open Source | WebAyunate. 2010 2010. [cited 6 October 2011]. Available from world wide web: <http://www.webayunate.com/software-libre-gobierno-paises-que-usan-open-source-linux/>.

[2]. Yoandy Pérez Villazon. Guía cubana para la migración a Software Libre. May 2008. [cited 24 September 2011]. Available from world wide web: <ftp://ftp.cult.cu/softwarelibre/documentacion/Guia%20cubana%200.32.pdf>.

[3].Anon. Newegg.com Hard Drives. [cited 15 September 2011]. Available from world wide web: <http://www.newegg.com/Store/Category.aspx?Category=15&name=Hard-Drives>.

[4] Josep Jorba Esteve, y Remo Suppi Boldrito. Herramientas de gestión de paquetes. En Administración avanzada de GNU/Linux, marzo 2004, pp 124. Eureca Media, SL Available from world wide web: <http://curso-sobre.berlios.de/introsobre>.

[5] Josep Jorba Esteve, y Remo Suppi Boldrito. Herramientas de gestión de paquetes. En Administración avanzada de GNU/Linux, marzo 2004, pp 125-128. Eureca Media, SL Available from world wide web: <http://curso-sobre.berlios.de/introsobre>.

[6] Josep Jorba Esteve, y Remo Suppi Boldrito. Herramientas de gestión de paquetes. En Administración avanzada de GNU/Linux, marzo 2004, pp 130-133. Eureca Media, SL Available from world wide web: <http://curso-sobre.berlios.de/introsobre>.

[7] Comunidad de Software Libre. APTonCD. [citado 27 mayo 2012]. Available from world wide web: <http://aptoncd.sourceforge.net/>.

[8]. Comunidad de Software Libre. Keryx Project - Offline Package Management Made Easy. [citado 27 mayo 2012]. Available from world wide web: <http://keryxproject.org/>.

[9]. Gladys Marsi Peñalver Romero, y Abel Meneses Abad. SXP, metodología ágil para proyectos de software libre. 2009.

[10]. Comunidad de PHP. PHP: Historia de PHP - Manual. December 2011. [cited 5 December 2011]. Available from world wide web: <http://www.php.net/manual/es/history.php.php>.

[11]. Anon. Guía Breve de XHTML. February 2008. [cited 5 December 2011]. Available from world wide web: <http://www.w3c.es/divulgacion/guiasbreves/XHTML>.

[12]. Javier Eguíluz Pérez. Introducción a CSS. *Introducción a CSS*, pp 5,6,7. diciembre 2008.

[13]. Javier Eguíluz Pérez. Introducción a AJAX. *Introducción a AJAX*, pp 5,6,7. junio 2008.

[14]. Anon. jQuery: The Write Less, Do More, JavaScript Library. [cited 10 February 2012]. Available from world wide web: <http://jquery.com/>.

[15]. Comunidad de PHP. PHP: Introducción Manual. February 2012. [cited 10 February 2012]. Available from world wide web: <http://www.php.net/manual/es/intro.pdo.php>.

[16]. Anon. Doctrine PHP Object Persistence Libraries and More. [cited 10 February 2012]. Available from world wide web: <http://www.doctrine-project.org/>.

[17]. Fabien Potencier. Practical symfony | Día 3: El Modelo De Datos | symfony | Web PHP Framework. [cited 10 February 2012]. Available from world wide web: <http://www.symfony-project.org/jobeet/1_2/Doctrine/es/03>.

[18]. John Lim. ADODB Manual. May 2010. [cited 10 February 2012]. Available from world wide web: <http://phplens.com/lens/adodb/docs-adodb.htm>.

[19]. Mohammed J, Kabir. La biblia Server Apache. July 2004.

[20]. Anon. Firebug. [cited 9 December 2011]. Available from world wide web: <**http://getfirebug.com/**>.

[21] Idem [9]

[22] Cesar de la Torre, Unai Zorrilla, y Miguel Ángel Ramos Javier Calvarro. ESTILOS ARQUITECTURALES. *Guía de Arquitectura N-Capas orientada al dominio con .NET 4.0*, pp 17,18 Available from world wide web: <http://www.amazon.com/Arquitectura-N-Capas-orientada-Dominio-Spanish/dp/8493669636>.

Bibliografía.

Anon. Doctrine PHP Object Persistence Libraries and More. [cited 10 February 2012]. Available from world wide web: <http://www.doctrine-project.org/>.

Anon. Firebug. [cited 9 December 2011]. Available from world wide web: <http://getfirebug.com/>.

Anon. Guía Breve de XHTML. February 2008. [cited 5 December 2011]. Available from world wide web: <http://www.w3c.es/divulgacion/guiasbreves/XHTML>.

Anon. jQuery: The Write Less, Do More, JavaScript Library. [cited 10 February 2012]. Available from world wide web: <http://jquery.com/>.

Anon. Newegg.com - Hard Drives. [cited 15 September 2011]. Available from world wide web: <http://www.newegg.com/Store/Category.aspx?Category=15&name=Hard-Drives>.

Cesar de la Torre, Unai Zorrilla, y Miguel Ángel Ramos Javier Calvarro. ESTILOS ARQUITECTURALES. Guía de Arquitectura N-Capas orientada al dominio con .NET 4.0, pp 17,18 Available from world wide web: <http://www.amazon.com/Arquitectura-N-Capas-orientada-Dominio-Spanish/dp/8493669636>.

Comunidad de PHP. PHP: Historia de PHP - Manual. December 2011. [cited 5 December 2011]. Available from world wide web: <http://www.php.net/manual/es/history.php.php>.

Comunidad de PHP. PHP: Introducción Manual. February 2012. [cited 10 February 2012]. Available from world wide web: <http://www.php.net/manual/es/intro.pdo.php>.

Dr. Delfín Felipe Hernández Rico. Comportamiento del riesgo reproductivo preconcepcional - Revista Electrónica de PortalesMedicos.com. March 2011. [cited 26 September 2011]. Available from world wide web:<http://www.portalesmedicos.com/publicaciones/articles/3071/8/Comportamiento-del-riesgo-reproductivo-preconcepcional>.

Fabien Potencier. Practical symfony | Día 3: El Modelo De Datos | symfony | Web PHP Framework. [cited 10 February 2012]. Available from world wide web: <http://www.symfony-project.org/jobeet/1_2/Doctrine/es/03>.

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerald, y Michael Stal. Pattern-Oriented.Software Architecture. A Systemof Patterns. 2001.

Ger. Software libre y gobierno - Los países más Open Source | WebAyunate. August 2010. [cited 6

October 2011]. Available from world wide web: <http://www.webayunate.com/software-libre-gobierno-paises-que-usan-open-source-linux/>.

Gnome.org. Guías de la interfaz humana de GNOME 2.2.2. Guías de la interfaz humana de GNOME. [citado 20 mayo 2012]. Available from world wide web: <http://developer.gnome.org/hig-book/3.0/>.

Javier Eguíluz Pérez. Introducción a AJAX. Introducción a AJAX, pp 5,6,7. junio 2008.

Javier Eguíluz Pérez. Introducción a CSS. Introducción a CSS, pp 5,6,7. diciembre 2008.

John Lim. ADODB Manual. May 2010. [cited 10 February 2012]. Available from world wide web: <http://phplens.com/lens/adodb/docs-adodb.htm>.

Josep Jorba Esteve, y Remo Suppi Boldrito. Herramientas de gestión de paquetes. En Administración avanzada de GNU/Linux, marzo 2004, pp 123, 133, Eureca Media, SL Available from world wide web: <http://curso-sobre.berlios.de/introsobre>.

M.Sc. Migdely Ochoa Avila, M.Sc. Lourdes Pérez Iglesias, and Lic. Moisés Martínez Ramírez. Revista Electrónica “ Ciencias Holguín.” July 2006. [cited 26 September 2011]. Available from world wide web: <http://www.ciencias.holguin.cu/2006/junio/articulos/ARTI6.htm>.

Mohammed J, Kabir. La biblia Server Apache. July 2004.

Oscar Bernal. Tipos de paquetes en linux/unix. October 2011. [cited 26 September 2011]. Available from world wide web: <http://www.oscarbernal.net/index.php?/content/view/36/20/>.

REA. Real Academia Española. [citado 2 junio 2012]. Available from world wide web: <http://www.rae.es/rae.html>.

Yoandy Pérez Villazon. Guía cubana para la migración a Software Libre. May 2008. [cited 24 September 2011]. Available from world wide web: <ftp://ftp.cult.cu/softwarelibre/documentacion/Guia%20cubana%200.32.pdf>.

Anexos.

Anexo I Estándar de codificación utilizado

1. Los archivos que contienen solo código PHP los *tags* de demarcación (“<?”) no estarán permitidos, ya que no son requeridos por PHP y omitirlos nos previene de accidentes ocasionados por un espacio en blanco.

2. La identación consistirá en 4 espacios en blanco y la tabulación no estará permitida:

```
<?php
class sfFoo {
    public function bar() {
      sfCafe::hacer();
   }
}
```

3. El largo máximo ideal de una línea de código es de 80 caracteres, esto nos permite una cómoda lectura del mismo. El máximo permitido por PHP es de 120 caracteres.

4. Las llaves siempre irán en su propia línea.

5. No ponga espacios después de abrir un paréntesis ni antes de cerrarlo:

```
<?php
if ($miVariable == getRequestValue($name))    // correcto
if ( $miVariable == getRequestValue($name) )  // incorrecto
```

6. Todo código PHP debe estar delimitado por los tags estándares:

```
<?php      ?>  //correcto
<?         ?> // incorrecto
```

7. Se usará la notación camelCase.

```
function hacerCafe() // correcto
function HacerCafe() // incorrecto
function hacer_cafe() // incorrecto
```

8. Use llaves para indicar el cuerpo de las estructuras de control, sin tener en cuenta el número de sentencias que éstas contengan.

9. En el cuerpo de las funciones, la sentencia **return** debe tener una línea en blanco a priori para incrementar la legibilidad:

```
<?php
function hacerCafe() {
    if (false !== estaDurmiendo() && false !== tieneSuficienteCafeinaPorHoy())   {
        puedeHacerCafe();

        return 1;
    }  else  {
      noPuedeHacerCafe();
    }
  }
```

10. Todo comentario debe estar en sus propias líneas y con el siguiente formato:

```
<?php
// primero un espacio, no necesita etiqueta de cierre.
```

11. Evite la evaluación de variables dentro de una cadena, en cambio use la concatenación:

```
<?php
$string = 'algo';
$nuevoString = "$string es imponente!";  // mal
$nuevoString = $string.' es imponente!'; // bien
$nuevoString = sprintf('%s es imponente', $string);
// para mensajes de excepción y cadenas con muchas sustituciones.
```

12. Use minúsculas para constantes tipadas nativas de PHP: false, true y null. Lo mismo para array(). Todo lo contrario, siempre use cadenas en mayúsculas para la definición de sus propias constantes, como define('MI_CONSTANTE', 'foo/bar'). Mejor, trate de usar siempre las constantes de clases:

```
<?php
class sfCafe {
    const TIENE_AZUCAR = true;
}
var_dump(sfCafe::TIENE_AZUCAR);
```

13. Cuando compare una variable con un string, ponga primero el string y luego use un tipo de prueba siempre y cuando sea aplicable:

```
<?php
if (“cadena” === $variable)
```

14. Los nombres de las tablas de la base de datos serán lo más descriptivo posible, iniciarán con el prefijo t, y para el caso de nombres compuestos se usará guión bajo:

    - ✔ tpaquete
    - ✔ trepositorio
    - ✔ trepositorio_paquetes

15. Los nombres de los atributos de las tablas de la base de datos serán lo más descriptivos posibles, usando guión bajo para nombres compuestos por más de una palabra y siempre en minúscula:

    - ✔ id_paquete
    - ✔ id_repositorio
    - ✔ id_repositorio_paquete

Glosario de términos

Apt-get: Es una herramienta que es usada mediante la línea de comandos y realizan funciones tales como la instalación y actualización de paquetes de software.

Código Abierto: Es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones morales y/o filosóficas las cuales destacan en el llamado software libre.

Librería o Biblioteca: Es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular. Algunos programas ejecutables pueden ser a la vez programas independientes y bibliotecas, pero la mayoría de éstas no son ejecutables. Ejecutables y bibliotecas hacen referencias (llamadas enlaces) entre sí a través de un proceso conocido como enlace, que por lo general es realizado por un software denominado enlazador.

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en Windows en un procesador x86, en GNU/Linux en un procesador x86, y en Mac OS X en uno x86 (solo para equipos Apple) o en un PowerPC.

ORM (Object Relational Mapping): Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

Packages.gz: Es un fichero donde se encuentra toda la información de todos los paquetes de un repositorio. Toda la información es listada y usada por los Administradores de Paquetes del sistema tales como *dselect* o *aptitude*.

Plugin: Es una aplicación que se integra con otra para aportarle una función nueva y generalmente muy especifica.

Programación paralela: Es el uso de varios procesadores trabajando en conjunto para dar solución a una tarea en común.

Release: Se refiere a una versión funcional de un producto software.

Scripts: Es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano. Los script son casi siempre interpretados, pero no todo programa interpretado es considerado un script. El uso habitual de los **scripts** es realizar diversas tareas como combinar componentes, interactuar con el SO o con el usuario.

Software libre: Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido libremente. Según la Free Software Fundation (FSF), el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar el software y distribuirlo modificado.

Software: Conjunto de programas, instrucciones y reglas para ejecutar ciertas tareas en una computadora u ordenador.

Synaptic: Esta herramienta pueden listar todos los paquetes que se han instalado y los paquetes que están disponibles en los repositorios que se han configurado. También permite realizar búsquedas a los paquetes de los repositorios. Además de proporcionar la instalación y actualización de software.

UML (Unified Modeling Language): Lenguaje Unificado de Modelado, es un lenguaje de modelado de sistemas de software. Permite la especificación, visualización, construcción y documentación de elementos de la Ingeniería del Software.

W3C (World Wide Web Consortium): Organismo internacional que tiene por fin establecer normas para el desarrollo y uso de la web.