Facultad 1

Título: Ejecución en línea de las técnicas de agrupamiento de tarjetas y análisis de secuencia

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Fernando Rafael González Rumbaut

Tutores: Ing. Dayaisis B. Bernis Pompa

Ing. Alberto Tamayo Ramos

Declaración de autoría | 2012

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Fernando Rafael González Rumbaut

Autor

_____ _____

Ing.Dayaisis B. Bernis Pompa Ing. Alberto Tamayo Ramos

Tutora Tutor

Datos de contacto 2012

Ing. Dayaisis B. Bernis Pompa

Ingeniera en Ciencias Informáticas, graduada en 2009. Perteneció al grupo de trabajo Arquitectura y Estándares de información de la Dirección Técnica de la Universidad de las Ciencias Informáticas. Trabajó como analista principal del Proyecto Gestión de Arquitectura de Información (Abad) del Centro de Informatización Universitaria (CENIA) de la facultad 1 de la propia universidad. Actualmente desempeña el mismo rol en proyecto Identificación y Control de Acceso. Imparte en pregrado la asignatura Arquitectura de Información.

Correo: dbbernis@uci.cu

Ing. Alberto Tamayo Ramos

Profesor instructor graduado en el 2007, se desempeña como jefe del departamento de Universidad Digital del Centro de Informatización Universitaria (CENIA).

Correo: atamayor@ucic.u

Agradecimientos 2012

Hasta que se acabó el quinto curso en la UCI y como dicen voces del pasado, me he convertido en hombre, o al menos he crecido un poco. Quiero darles mis más sinceros agradecimientos a todos aquellos que me han ayudado a llegar a donde estoy hoy, no importa de qué manera, buena o mala, les doy las gracias .Agradecer a todos los que estuvieron en ese momento cuando los necesité, a mi madre por estar siempre a mi lado y darme apoyo y amor incondicional, a Alejandro por ser como un padre desde que tengo conciencia, a mi tía y a mi abuela que no por estar lejos, dejan de estar presente, a Liset por su ayuda, a Teymor, Dayron, Daniel, Relmy, Alvaro, Mario, Luis René que más que amigos han significado hermanos todo este tiempo, a Yeimy por ser luz en obscuridad y todas mis amistades en general, a Yanicet por brindarme su apoyo y la paciencia necesaria, a mi tutor Alberto por su ayuda y a todos los que me ayudaron de una manera u otra, muchas gracias.

Dedicatoria 2012

Este trabajo se lo dedico a mi madre, quien más que entregarme amor, paciencia y comprensión, me ha dedicado su vida entera.

Resumen 2012

El presente documento muestra cómo fue llevada la investigación para el desarrollo de una solución que permite la ejecución de las técnicas de agrupamiento de tarjetas y análisis de secuencia a usuarios geográficamente dispersos, describiendo los aspectos teóricos conceptuales relacionados con las técnicas de agrupamiento de tarjetas y análisis de secuencia. El desarrollo de la propuesta estuvo guiado por el proceso de desarrollo con enfoque ágil CMMI nivel 2. Se realiza un estudio a soluciones homólogas existentes, así como de los lenguajes de programación utilizados y herramientas empleadas para el desarrollo de la solución.

Palabras clave:

Agrupamiento de tarjetas, análisis de secuencia.

Introducción

Junto al desarrollo que ha tenido el campo de la informática en los últimos años, ha crecido el nivel de exigencia en cuanto a la calidad de los productos en el mercado. En este sentido, se hace necesario encontrar la manera de perfeccionar la organización y disposición de la información en las soluciones informáticas. Un factor importante para el logro de este objetivo es la arquitectura de la información (AI) aplicada a las soluciones informáticas.

La AI surge como disciplina para estudiar, analizar, y organizar la disposición y estructuración de la información en cualquier medio, de tal forma que sea consultada por el usuario de manera intuitiva (JAMES GARRET, 2002). Actualmente es una materia bastante utilizada en el mundo que va ganando popularidad y son cada vez más los profesionales de la gestión de la información y documentación que van añadiendo el título de arquitecto de información a sus habilidades (BALMASEDA BORLADO, 2011).

A medida que el proceso de AI ha ganado profundidad, se han diseñado métodos y técnicas. Dentro de este marco se encuentran las técnica de agrupamiento de tarjeta y de análisis de secuencia, que tienen como fin extraer del modelo mental de los usuarios para entender “la manera en que los usuarios esperan encontrar el contenido o una funcionalidad” delimitando como ellos agrupan y organizan la información. De esta manera, el arquitecto de información puede optimizar el diseño del producto final centrándose en el usuario y lograr una mejor interacción con el mismo. (WARFEL y SPENCER, 2004)

Tomando en cuenta la importancia que ha ganado la AI en la mejora de la calidad de las soluciones informáticas, la Universidad de las Ciencias Informáticas (UCI), encargada de formar profesionales en este campo, ha decidido adentrarse en el tema de la AI y en el perfeccionamiento de sus procesos para la producción y exportación de aplicaciones.

En el año 2010 en la UCI se crea el Centro de Informatización Universitaria (CENIA), el cual agrupa una serie de proyectos encargados de la informatización de la UCI. En febrero del 2010 se crea el proyecto ABAD dentro de dicho centro, con el objetivo de desarrollar soluciones para la gestión de los flujos de experiencia de usuario en los proyectos de desarrollo de soluciones informáticas de la entidad.

Actualmente el proyecto Abad se encuentra inmerso en el desarrollo de un paquete de herramientas para diseñar experiencia de usuarios llamado Abad. Este paquete cuenta con un *software* para la ejecución de las técnicas de agrupamiento de tarjetas y análisis de secuencia. Aunque la opinión de los especialistas es que cumple con los objetivos para el cual fue creado, se notan ciertas dificultades a la hora de ejecutar dichas técnicas, pues

para realizar una ejecución se ha de reunir a todos los usuarios que participan en un local y distribuir la herramienta de escritorio.

En caso de que los usuarios no puedan ser reunidos en un lugar específico, el arquitecto de la información debe ir hacia donde se encuentra cada uno y realizar la ejecución. Si los participantes se encuentran a una distancia considerable, que impida su presencia en la ejecución, la cantidad de ejecuciones se ve limitada afectando directamente los resultados. Todo esto trae como consecuencia un entorpecimiento y demora a la hora de definir el modelo mental del usuario por parte del arquitecto información, en ocasiones la obtención de las soluciones es limitada lo que puede traer como consecuencia menor exactitud de los resultados.

Atendiendo a la problemática planteada se define como **problema de investigación:**

¿Cómo aplicar las técnicas de agrupamiento de tarjetas y análisis de secuencia posibilitando su ejecución a usuarios geográficamente dispersos?

Se tiene como **objeto de estudio** el proceso de arquitectura de información y **como campo de acción** las técnicas de agrupamiento de tarjetas y análisis de secuencia.

Se plantea como **objetivo general:** Desarrollar una herramienta web en el paquete Abad que permita aplicar las técnicas de agrupamiento de tarjetas y análisis de secuencia a usuarios geográficamente dispersos, para apoyar la toma de decisiones del arquitecto de información en la representación de contenidos.

Objetivos Específicos:

- Identificar las características esenciales de las técnicas de agrupamiento de tarjetas y análisis de secuencia.
- Definir una propuesta de sistema.
- Implementar la propuesta a partir del análisis realizado.
- Realizar pruebas a la propuesta resultante.

Posible resultado:

Un sistema web para la ejecución de las técnicas de agrupamiento de tarjetas y análisis de secuencia que permita la realizar dichas técnicas a usuarios geográficamente dispersos, la cual que ayude en la toma de decisiones de los arquitectos de información en la representación de contenidos.

Métodos utilizados:

- **Métodos teóricos:**

  Histórico - lógico:

  Se estudió como ha sido la trayectoria y la evolución de las herramientas que ejecutan ejercicios de agrupamiento de tarjetas y análisis de secuencia para tener una visión de las competencias que la propuesta debe cumplir.

  Analítico – Sintético:

  Se llevó a cabo un análisis de la bibliografía disponible, el que permitió determinar conceptos y obtener los conocimientos necesarios para el desarrollo de diferentes funcionalidades.

- **Métodos Empíricos:**

  Entrevistas:

  Mediante entrevistas informales realizadas a los especialistas del centro se pudo determinar cuáles eran las expectativas que se debían cumplir, herramientas a utilizar, además de esclarecer dudas sobre funcionalidades que debería tener la propuesta a implementar.

El trabajo está estructurado en cuatro capítulos:

- **Capítulo 1: Fundamentación teórica**

  Este capítulo incluye una descripción teórica de las técnicas de ordenamiento de tarjetas y análisis de secuencia, un estudio del estado del arte a nivel internacional, nacional y de la Universidad de soluciones homólogas, además de un estudio sobre las metodologías y tecnologías utilizadas para el desarrollo de la propuesta.

- **Capítulo 2: Características y construcción del sistema**

  En este capítulo se incluye el modelo de dominio y los requisitos funcionales y no funcionales que debe cumplir la aplicación, así como la construcción del modelo de dominio, patrones de diseño y arquitectónicos a emplear en la solución.

- **Capítulo 3: Implementación y prueba**

  En este capítulo se expone los estándares de codificación utilizados y las pruebas realizadas al sistema.

Capítulo 1: Fundamentación teórica

1.1 Introducción al capítulo

En este capítulo se recogen los principales conceptos relacionados con la Arquitectura de la Información, agrupamiento de tarjetas y análisis de secuencia. Se hace un estudio del estado del arte sobre las herramientas que ejecutan ejercicios de agrupamiento de tarjetas y análisis de secuencia, así como las tecnologías y herramientas a utilizar para el desarrollo de la propuesta.

1.2 Conceptos sobre Arquitectura de Información

Arquitectura de la Información

La Arquitectura de la Información es la disciplina encargada del estudio, análisis, organización, disposición y estructuración de los contenidos en un espacio de información. En la actualidad es muy utilizada en el diseño de las interfaces para sitios web, aunque también es aplicada a otros escenarios fuera de la web. Estudia cómo la organización de los contenidos puede ayudar a definir el pensamiento de las personas dada la adaptabilidad que tienen estos procesos a varios aspectos de la vida, ya que pueden ser usados en ramas de la medicina, tales como la psicología y la psiquiatría. Es una materia que propone al usuario como vía fundamental para la solución de sus propios problemas, revelando sus criterios sin necesidad de que este los exprese mediante la palabra, de manera que se logre una comunicación estándar entre ambas partes (BALMASEDA BORLADO, 2011).

Presenta una amplia variedad de técnicas como la interacción con el usuario mediante reuniones, entrevistas, encuestas, diseño de escenarios y diseño participativo, las de interacción con el contexto con la evaluación de productos similares y análisis de la competencia, las técnicas matemáticas de agrupamiento de tarjetas y análisis de secuencia y técnicas de representación de información en diagramación, representación de etiquetas y prototipado (RONDA LEÓN, 2007) . La presente investigación se centra en las técnicas de agrupamiento de tarjetas y análisis de secuencia, pues son las que se necesitan comprender.

Agrupamiento de tarjetas y análisis de secuencia

Según Ronda León, son técnicas matemáticas que aplican análisis de coocurrencia para cuantificar resultados y hacer precisa la toma de decisiones. Con la aplicación de estas técnicas se logran definir grupos o crear secuencias que se correspondan con el modelo

mental de los usuarios (RONDA LEÓN, 2007). Se debe señalar que estas técnicas no son netamente matemáticas, pues para lograr aplicar un análisis de coocurrencia, primero se debe tener una interacción con los usuarios de interés, los cuales cumplen un papel fundamental pues son los principales actores y quienes brindan la información necesaria a los arquitectos de información para realizar sus respectivos análisis (MONTES DE OCA, 2004), por lo que pueden ser vistas como técnicas matemáticas de coocurrencia o de diseño centrado en el usuario participativo.

Agrupamiento de tarjetas

El agrupamiento de tarjetas es una técnica de diseño centrado en el usuario participativo, utilizado por los arquitectos de información para comprender cómo los usuarios reconocen y modelan la información (CELESTE, 2008). Puede proveer una vista interior de los modelos mentales del usuario, revelando la forma en que ellos tácticamente agrupan, ordenan y etiquetan tareas y contenidos dentro de sus mentes (WARFEL y SPENCER, 2004).

El agrupamiento de tarjetas está compuesto por tres fases: preparación, ejecución y evaluación. En la primera fase se prepara el ejercicio mediante la elaboración de las instrucciones, las tarjetas y las categorías, que serán usadas en la fase de ejecución. Estas fichas se le brindan al usuario para que agrupe según su criterio. En su última fase se evalúan las clases obtenidas mediante algoritmos de formación de agrupamientos y se obtienen recomendaciones para la organización de la información de acuerdo con los patrones de los participantes.

Se pueden identificar dos tipos de agrupamiento de tarjetas, “abierto” y “cerrado”. El agrupamiento “abierto” permite que el usuario pueda agruparlas libremente en el número de conjuntos que crea necesario, mientras que en el cerrado, los grupos o conjuntos están predefinidos y etiquetados y el usuario deberá colocar cada tarjeta en el grupo que crea le corresponda (Hassan, 2004).

El agrupamiento de tarjetas abierto tiene como objetivo descubrir qué tipo de categorías sería más correcto utilizar, mientras que el cerrado es recomendado para verificar si una clasificación de información es familiar y comprensible para el usuario (Hassan, 2004).

El ordenamiento de tarjetas presenta numerosas ventajas y algunas desventajas dentro de las que se encuentran:

Ventajas:

- Simple: el ordenamiento de tarjetas es simple para el organizador y los participantes.
- Barato: usualmente cuando se ejecuta de forma manual el costo es relativamente barato, solo se necesitan un grupo de tarjetas de 3x5'', algunas hojas, una pluma o varias etiquetas impresas y el tiempo de las personas involucradas.
- Rápida ejecución: se pueden realizar varios ordenamientos en un periodo de tiempo relativamente corto, los cuales proveerán un cúmulo significativo de datos.
- Involucra a los usuarios: porque debería ser más fácil de usar la estructura de información sugerida por el ordenamiento de tarjetas, que está basada en los modelos mentales de usuarios reales y no en las distintas o sólidas opiniones de un diseñador, arquitecto de información, o de un cliente clave.
- Provee buenos fundamentos: no es la mejor solución pero suministra buenas bases para la estructura de un sitio Web o producto (WARFEL y SPENCER, 2004).

Desventajas:

- No considera roles de usuario: el ordenamiento de tarjetas es una técnica de naturaleza centrada en el contenido. Si es utilizada sin tener presente los roles de los usuarios, puede encaminarse hacia una estructura de información que no sea la más indicada. Por ello se hace necesario un análisis de roles o necesidades de información para asegurar que el contenido a ordenar coincida con las necesidades de los usuarios y que la estructura de información resultante permita a los usuarios cumplir sus roles.
- Los resultados pueden variar: el agrupamiento de tarjetas puede proveer aleatoriamente resultados similares entre los participantes, o estos pueden ser muy diferentes entre sí.

- El análisis puede consumir bastante tiempo: de forma manual, el agrupamiento es rápido, pero el análisis de los datos puede ser dificultoso y consumir bastante tiempo, particularmente si los resultados de los participantes son muy diferentes.

- Puede ser que capture solo características superficiales: los participantes pueden no considerar sobre lo que trata el contenido o cómo lo utilizarán , lo cual puede ocasionar que ordenen las tarjetas por sus características superficiales como si fueran tipos de documentos (WARFEL y SPENCER, 2004).

Análisis de secuencia

El análisis de secuencia sirve como complemento al agrupamiento de tarjetas brindando un mayor grado de comprensión en los resultados. Es un método de diseño centrado en el usuario participativo, que los arquitectos de la información utilizan para encontrar la manera de entender mejor el orden en que los usuarios reconocen y modelan la información. Tiene un alto grado de similitud con el agrupamiento de tarjetas pero su diferencia radica en que los resultados tienen otro objetivo, formar una secuencia de elementos para ser usada en el producto como la orden de términos de una barra de navegación, de un menú desplegable, de un listado de productos a vender, etc. (WARFEL y SPENCER, 2004).

Al igual que la técnica de agrupamiento de tarjetas, el análisis de secuencia está compuesto de tres fases: preparación, ejecución y evaluación. Durante la fase de preparación se definen cuáles van a ser las etiquetas y las posiciones que se estarán usando en el ejercicio. Durante la fase de ejecución, las etiquetas son agrupadas por el usuario dentro de las posiciones, a cada etiqueta le corresponde una posición En la evaluación el arquitecto interpreta los resultados.

1.3 Análisis de sistemas homólogos

Para la realización de la presente investigación se tuvo en cuenta los trabajos de diploma: “Análisis de un sistema automatizado a través de la técnica de *CardSorting* u Ordenación de Tarjetas” (SOLA PADILLA, 2008), “Análisis y Diseño de un sistema para la aplicación de técnicas de CardSorting en la obtención de Arquitecturas de Información” (OROVIO PINO, 2009), “Implementación del módulo “Técnicas de ordenamiento de tarjetas” para la plataforma de arquitectura de la información Abad” (ESPINOSA RAMÍREZ y JIMÉNEZ MORALES, 2010) y “Herramienta de ordenamiento

de tarjetas versión 2.0 para la plataforma de arquitectura de información Abad" (BALMASEDA BORLADO, 2011).

Se realizó un estudio de varias herramientas que pudiesen dar solución a la problemática o que aportasen ideas para la construcción de un nuevo sistema. A nivel nacional no se identificó ninguna herramienta que ejecutase ejercicios de agrupamiento de tarjetas, exceptuando la que existe en el proyecto Abad.

Esta búsqueda se realizó a través de un grupo de características que se debían cumplir buscando cuales fuesen las herramientas más completas. Dichas características fueron seleccionadas atendiendo a las necesidades actuales del proyecto Abad. Se debe seleccionar una herramienta web capaz de ejecutar ejercicios de agrupamiento de tarjetas y análisis de secuencia, no privativa, capaz de generar ficheros xml (por sus siglas en ingles *eXtensible Markup language*) compatibles con la herramienta de escritorio que existe actualmente en el proyecto Abad.

Herramientas estudiadas

UzCardSort

Herramienta de código abierto con licencia MPL basado en Mozilla para la realización y análisis de ejercicios de ordenamiento de tarjetas. Puede ejecutarse en Windows, Linux, Macintosh y versiones de Mozilla.

UXsort

Herramienta creada por investigadores de experiencia de usuario de Microsoft, el uso de esta herramienta está regulada por los acuerdos de la licencia GNU.

OptimalSort

Herramienta web de licencia privativa de OptimalWorkShop que ejecuta ejercicios de agrupamiento de tarjetas tanto abiertos como cerrados.

WebSort

Herramienta web, privativa que permite la creación, edición y ejecución de ejercicios de CardSorting en la web.

GRIHO.

Herramienta de escritorio para la ejecución de ejercicios de agrupamiento de tarjetas creada por el Grupo de Investigación en Interacción Persona Ordenador de la Universidad de Lleida.

Tabla 1. Características que deben cumplir las herramientas estudiadas.

	Aplicación Web	No privativa	Agrupamiento de Tarjetas	Análisis de Secuencia	Entorno de ejecución flexible	Ficheros en formato XML	Gestión de Usuarios	Almacenamiento en BD de Soluciones
UzCardSort	X	X	X		X		X	
UxSort		X	X				X	X
WebSort	X		X				X	
OptimalSort	X		X		X		X	
GRIHO		X	X		X			

Ninguna de las herramientas estudiadas fue tomada para dar solución a la problemática existente, pues no cumplen con todas las características señaladas. Además, no ejecutan ejercicios de la técnica de análisis de secuencia ni exportan ficheros xml con la información de las soluciones contenidas, que sean compatibles con la herramienta de escritorio existente dentro del paquete Abad.

De las herramientas estudiadas se tomaron aspectos positivos tales como realizar la ejecución de los ejercicios permitiendo el agrupamiento de tarjetas mediante el arrastrar y soltar, desarrollar una gestión de usuarios que permita al arquitecto guardar y publicar sus ejercicios en la aplicación a desarrollar y permitir el almacenamiento de ejercicios y soluciones en una base de datos.

1.4 Proceso de desarrollo con enfoque ágil CMMI nivel 2

Para el desarrollo de la propuesta no se usó ninguna metodología específica, en cambio se utilizó el Proceso de desarrollo con enfoque ágil CMMI nivel 2 establecido en el proceso de mejoras en el cual se encuentra inmersa nuestra universidad.

Proceso de desarrollo de software:

Un proceso de desarrollo de software es aquel que tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente. "Define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo" mediante la definición de actividades, artefactos y roles" (P.LETELIER, 2011). Un proceso de software detallado y completo suele denominarse metodología.

El proceso de desarrollo de software no es único. No existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo. Las metodologías de desarrollo pueden clasificarse en ágiles y pesadas o tradicionales.

Enfoque ágil:

El enfoque ágil de las metodologías de desarrollo de software surge como alternativa a las metodologías formales consideradas excesivamente pesadas y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo. Dicho enfoque supera los defectos de estas, que se basan en:

- En la estrecha colaboración con los usuarios para ofrecer varias versiones del software en ráfagas cortas (*sprints*). Con cada versión, los requisitos para la próxima versión son refinados a partir de los comentarios de los usuarios.
- Valorar más el software que funciona que la documentación excesiva. Genera la documentación necesaria del desarrollo del producto.
- Mejores soluciones, menos costo y tiempo. Asegura que cada versión incorpore sólo los requerimientos de alto valor para el negocio. El enfoque ágil permite que el software sea implementado rápidamente sin perder de vista la calidad del producto que con los métodos tradicionales.
- Valorar más la respuesta al cambio que el seguimiento de un plan. Por lo que es ideal para proyectos donde los requisitos son inestables.

CMMI nivel 2:

Es el Modelo de Madurez de Capacidades Integrado (por sus siglas en inglés *Capability Maturity Model Integration*) que sirve de referencia para el crecimiento de capacidades y madurez, que se enfoca tanto en procesos de Administración como de Ingeniaría de Sistemas y Software. Fue creado por el SEI (*Software Engineer Institute*) y está compuesto por 32 áreas de proceso las que se distribuyen en 5 niveles (iniciado, gestionado, definido, gestionado cuantitativamente, en optimización) de madurez para clasificar a las empresas en función de qué áreas de proceso consiguen sus objetivos y se gestionan con principios de ingeniería(GRUPO DE CALIDAD DE SOFTWARE, 2011).

El nivel dos o administrado es alcanzado cuando existe una administración disciplinada de los proyectos, se establecen y se siguen las políticas organizacionales, los recursos son adecuados (humanos y materiales), se asigna responsabilidades y autoridades a lo largo de la vida del proyecto, los éxitos anteriores pueden ser repetidos, se siguen las prácticas aún en tiempos de estrés, la dirección tiene visibilidad de las actividades y los productos en puntos definidos. Este nivel cuenta con siete áreas de proceso: Administración de Requerimientos (REQM), Planeación del Proyecto (PP), Proceso Monitoreo y Control del Proyecto (PMC), Administración de Acuerdos con Proveedores (SAM), Medición y Análisis (MA), Aseguramiento de la Calidad de los Procesos y Productos (PPQA) y Administración de Configuración.

Entre las numerosas ventajas que trae la implantación de CMMI se destacan que posibilita la normalización y control del proceso del producto, facilita la alineación de los requisitos y los principios del modelo viabilizando a la organización en la consecución de sus metas y objetivos de negocio. Posibilita expandir el alcance y la visibilidad hacia el ciclo de vida de un producto y a las actividades de ingeniería. Mejora la rapidez y efectividad de respuesta ante exigencias del negocio y la colaboración y comunicación efectiva con implicados internos y externos.

1.5 Tecnologías y herramientas utilizadas para el desarrollo de la propuesta

Herramienta web para la ejecución de ejercicios de agrupamiento de tarjetas y análisis de secuencia.

Debido a que el proyecto Abad cuenta con una herramienta de escritorio que ejecuta ejercicios de agrupamiento de tarjetas y análisis de secuencia, creada para lugares

específicos, donde no existe un servicio de red o sea difícil acceder a estos tipos de servicio. Se valoró la posibilidad crear una solución para lugares donde si exista una red y se pueda aprovechar este recurso. Tras un estudio de las tecnologías disponibles y de las tendencias actuales de desarrollo de software se decidió que la solución a desarrollar será mediante una aplicación web.

El desarrollo de la solución mediante una aplicación web traerá consigo un grupo de ventajas, posibilitando la realización de ejercicios a un grupo mayor de participantes en menor tiempo desde diferentes localizaciones, lo que permite al arquitecto de información obtener un cúmulo mayor de soluciones de manera rápida aliviando la situación que existe respecto a los participantes que se encuentran geográficamente dispersos.

Lenguaje unificado de modelado (UML)

Es el lenguaje que se usa para visualizar, especificar, construir y documentar un sistema, de forma gráfica, característica que ayuda mucho en la comprensión de los sistemas. UML también intenta solucionar el problema de diversidad de código que en ocasiones se presenta entre los desarrolladores, pues al implementar un lenguaje de modelado común para todos los desarrollos, se crea una documentación también común, que cualquier desarrollador con conocimientos de este lenguaje será capaz de entender (LARMAN, 1999)

Lenguaje de programación

Un lenguaje de programación es el medio de la informática que permite crear programas empleando un conjunto de instrucciones, operadores y reglas de sintaxis; que se pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software del ordenador que posea. Consta de un léxico, una sintaxis y una semántica. Según el número de instrucciones que requiera para realizar una tarea específica, se clasifican en lenguaje de alto nivel y lenguaje de bajo nivel (PÉREZ GARCÍA, 2006).

Lenguajes de programación y tecnologías utilizadas del lado del servidor

PHP 5.3

Representa Preprocesador de Hipertexto (por sus siglas en inglés *Hypertext Preprocessor*). Es un lenguaje script que se ejecuta del lado del servidor en la arquitectura cliente – servidor. Muy utilizado por los desarrolladores de páginas web por ser uno de los lenguajes de programación más completo, siendo capaz de manejar un entorno que integre gestores de bases de datos como: MySQL, Microsoft SQL Server, PostgreSQL, SQLite y Oracle,

permitiendo la creación de aplicaciones web muy robustas. Tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como los soportados por UNIX, Linux, Mac OS y Windows, además puede interactuar con los servidores web más populares (ACHOUR *et al.*, 2008).

Servidor HTTP Apache 2

Es uno de los servidores web más utilizado actualmente, es de código abierto y gratuito, disponible para Windows y GNU/Linux. Apache presenta mensajes de error altamente configurables, bases de datos de autenticación y un negociador de contenido. Entre sus características fundamentales se pueden destacar (KAIR, 2003) :

- Es flexible, rápido y eficiente.
- Multiplataforma.
- Se desarrolla de forma abierta.
- Modular: puede ser adaptado a diferentes entornos, a las necesidades de los diferentes módulos de apoyo que proporciona y con la API (*Aplication Programming Interface*) de programación de módulos para el desarrollo de módulos específicos.
- Incentiva la realimentación de los usuarios obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos

SQL

El Lenguaje de Consulta Estructurada o SQL (por sus siglas en inglés *Structured Query Lenguaje*). Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas, normalizado, compatible con cualquier tipo de lenguaje (php, asp, otros) en combinación con cualquier tipo de bases de datos (PostgreSQL, Oracle, MySQL, otros) (ALVAREZ, RUBEN, 2001).

PostgreSQL 8.4

Es un Sistema Gestor de Bases de Datos (SGBD) relacionales orientadas a objetos. Es soportado en casi todos los sistemas operativos actuales Linux, Windows, Mac OS. Soportando casi toda la sintaxis SQL (incluyendo subconsultas, transacciones, tipos y funciones definidas por el usuario). Es capaz de manejar una gran cantidad de datos, permitiendo numerosos accesos simultáneos por parte de los usuarios. Presenta una documentación bien organizada, pública y libre. Tiene soporte nativo para lenguajes de

programación como PHP, Java, C, C++, Phyton además de presentar una amplia comunidad de apoyo (THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2008).

Lenguajes de programación tecnologías utilizadas del lado del cliente

JavaScript

Lenguaje de programación utilizado principalmente en la construcción de páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Es un lenguaje script multiplataforma orientado a objetos, pequeño y ligero que permite la creación de pequeños programas encargados de realizar acciones dentro de una página web. Todos los navegadores usados en la actualidad interpretan el código JavaScript integrado dentro de las páginas web.

JavaScript permite crear contenidos dinámicos y dar movimiento a elementos de una página web, cambiarles el color, realizar validaciones y ejecutar respuestas según las entradas del usuario a la página (ALVAREZ, MIGUEL ANGEL, 2009).

Ajax

AJAX o (*Asinchronous JavaScript And XML*) no es más que una técnica de desarrollo Web, mediante la cual se puede crear aplicaciones Web más rápidas y cómodas para el usuario. Por medio de esta técnica el cliente puede interactuar con el servidor de manera asincrónica, actualizando las páginas, sin necesidad de volver a cargarlas. Esto se traduce en un aumento significativo de la interactividad, velocidad y usabilidad en las mismas. De AJAX se ha dicho: “Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes” (JAMES GARRET, 2002).

Entre las tecnologías que conforman AJAX están:

- 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.

HTML

Siglas de *Hyper Text Markup Language* (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, para complementar el texto con objetos de imágenes. Es usado en la elaboración de interfaces (TITTEL y BURMEISTER, 2005).

CSS

Hojas de Estilo en Cascada (por sus siglas en inglés CSS) es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación, es decir, describe cómo se va a mostrar un documento HTML en pantalla. CSS permite un control preciso (fuera del marcado) del espaciado de caracteres, alineación de textos, posicionamiento de objetos en la página, características de fuente entre otros aspectos. Con la separación del estilo y el marcado los autores pueden simplificar y limpiar el HTML haciendo más accesible el documento (W3C ORGANIZATION, 2005).

Marcos de trabajos utilizados

Codeigniter 2.0.1:

Es un marco de trabajo PHP que proporciona una buena base para implementar aplicaciones web. Permite actualizar dinámicamente los contenidos. Tiene diversas librerías reutilizables además de permitir crear librerías independientes. Es compatible tanto con PHP 4 como con PHP 5. Usa el patrón modelo vista controlador (MVC) como paradigma de arquitectura de desarrollo. Existe abundante documentación y cuenta con una amplia comunidad de desarrollo. Codeigniter se creó para ser instanciado dinámicamente haciendo que los componentes y rutinas se carguen sólo cuando son invocados (COMPAÑÍA ELLISLAB INC., 2011).

Herramientas utilizadas

NetBeans 7.0.1

Es un entorno de integrado de desarrollo por sus siglas en ingles IDE, de libre uso y de código abierto que permite la extensión de aplicaciones hechas en este entorno dada su característica de que los módulos pueden ser desarrollados independientemente.

Para el desarrollo de la propuesta se utilizará el NetBeans PHP IDE 7.0.1 el cual es una versión del IDE NetBeans para el desarrollo de aplicaciones web PHP que abarca una variedad de secuencias de comandos y lenguajes de marcado. El editor de PHP es dinámico, integrado con HTML, JavaScript y CSS. Es capaz de centrarse en el código y

acelerar la lectura de mismo mediante la exclusión de directorios individuales en las propiedades del proyecto (ORACLE CORPORATION, 2011).

Visual Paradigm 6.4

Es una herramienta CASE[1] pensada para el ciclo completo del proyecto, dándole relevancia a todos los diagramas que en ella se crean. Permite dibujar todos los tipos de diagramas de clases, además de realizar código inverso, es decir, generar código desde diagramas. Tiene la capacidad de integrarse a entornos de desarrollo como el NetBeans IDE. Es una herramienta multiplataforma. También presenta como ventaja fundamental que a través de ella se pueden realizar prototipos de interfaz de usuarios que permiten tener una visión más cercana de cómo quedarían las interfaces del sistema.

pgAdmin III.

Es una herramienta gráfica para la administración del servidor de bases de datos PostgreSQL, multiplataforma, presenta una interfaz gráfica con disimiles funcionalidades que facilitan mucho el trabajo, describe en sentencia SQL cada modificación realizada en la base de datos. Presenta un editor de consultas SQL y no requiere controladores adicionales para conectarse con la base de datos del servidor (THE PGADMIN DEVELOPMENT TEAM, 2011).

1.6 Conclusiones del capítulo:

- El estudio de sistemas homólogos demostró que no existe una herramienta que satisfaga las necesidades actuales del proyecto Abad, por lo que se hace necesario el desarrollo de una nueva solución.
- El desarrollo de la propuesta será guiado por el proceso de desarrollo con enfoque ágil CMMI nivel 2.
- El desarrollo de la propuesta será mediante el uso de herramientas y tecnologías libres que no presenten ningún tipo de restricción de uso.

[1] Por sus siglas en inglés *Computer Aided Software Engineering,* (Ingeniería de software asistida por ordenador)

Capítulo 2. Características y construcción del sistema

2.1 Introducción:

En este capítulo se plantea una descripción del sistema a desarrollar, se crea el modelo de dominio, modelo físico, modelo de despliegue, se especifican los requisitos funcionales y no funcionales. Se aborda todo lo referente al modelo físico de la base de datos a utilizar en la solución, estilos y patrones arquitectónicos a emplear y el diagrama de despliegue de la solución.

2.2 Modelo de dominio

Dado que la propuesta de sistema parte de un sistema ya automatizado donde tiene sus procesos descritos, se realiza un modelo de dominio para entender el funcionamiento del negocio actual definiéndose los objetos más importantes del sistema y las interacciones que ocurren entre ellos.

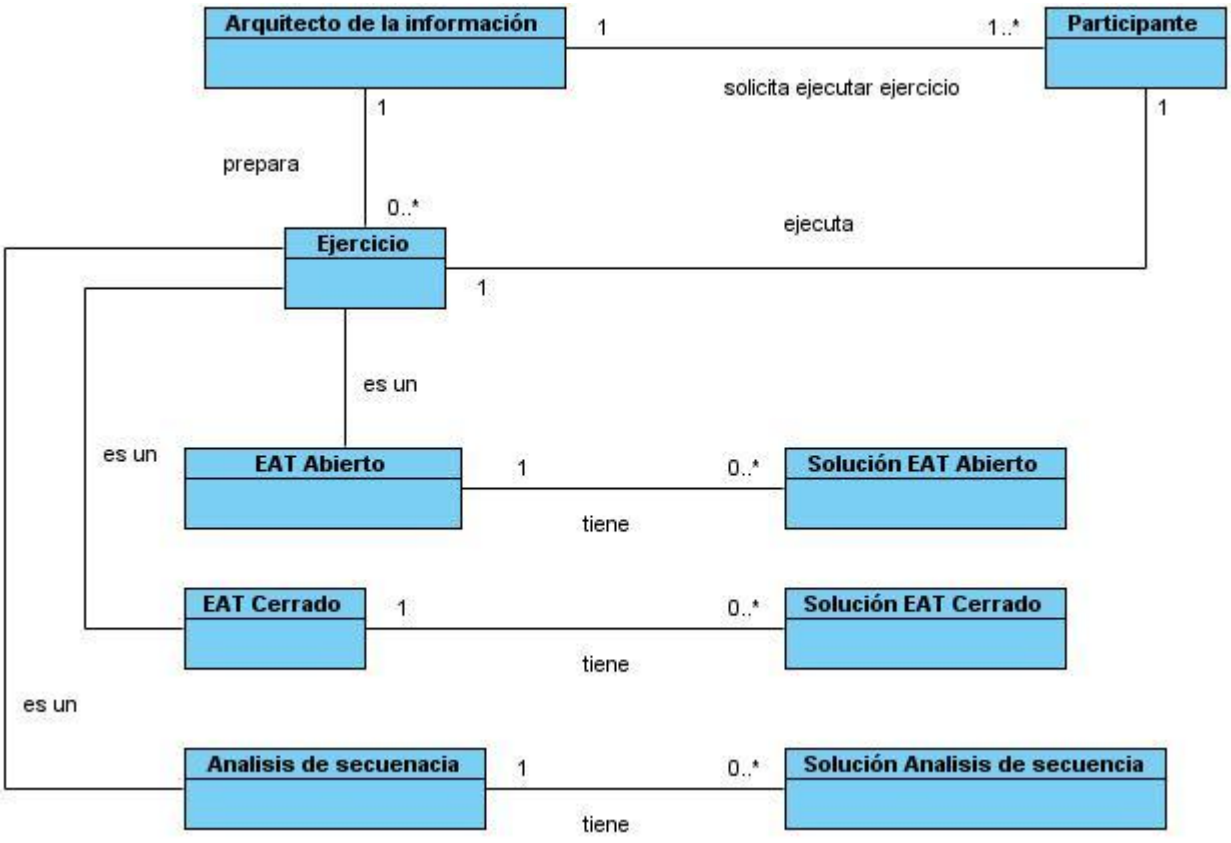


Figura 1. Modelo de dominio

Descripción de las clases del modelo de dominio.

1 **Arquitecto de la información**: Usuario encargado de la gestión de los ejercicios.

2 **Ejercicio**: Cualquier tipo de ejercicio de agrupamiento, es preparado por los arquitectos y ejecutado por los participantes.

3 **EOT Abierto**: Ejercicios de agrupamiento de tarjetas abierto.

4 **EOT Cerrado**: Ejercicio de agrupamiento de tarjetas cerrado.

5 **Análisis de secuencia**: Ejercicio de análisis de secuencia.

6 **Solución EOT Cerrado**: Solución obtenida luego de un agrupamiento de un ejercicio EOT Cerrado realizado por un participante.

7 **Solución EOT Abierto**: Solución obtenida luego de un agrupamiento de un ejercicio EOT Abierto realizado por un participante.

8 **Solución Análisis secuencia**: Solución obtenida luego de un agrupamiento de un ejercicio de Análisis Secuencia realizado por un participante.

9 **Participante**: Cualquier persona a la que se le solicite ejecutar un ejercicio de agrupamiento.

2.3 Requisitos

Los requisitos son una pieza fundamental en el desarrollo de software pues son el punto de partida para actividades como la planeación, estimación de tiempos y costos, así como la definición de costos. Permiten también verificar si se alcanzaron o no los objetivos establecidos, los mismos son un reflejo de la necesidad del cliente por lo que deben estar descritos de manera clara, correcta, libre de ambigüedades, en forma consistente y compacta (CHAVES, 2007).

Técnicas de captura de requisitos:

Para el levantamiento de requisitos de la solución a desarrollar se tuvo en cuenta la técnica de entrevista la cual se utilizó para entrevistar a las personas que se relacionan directamente con el negocio mediante entrevistas informales que permitió una mejor comprensión de las necesidades existentes. Otra técnica empleada fue el prototipado, utilizada para los casos donde los requisitos no quedaban claros o no se entendieron correctamente además de servir como simulaciones del posible producto representando aquellos aspectos que son visibles en el software para el usuario.

Requisitos Funcionales:

Los requisitos funcionales son aquellos que determinan que debe hacer el sistema, o sea, funcionalidades que debe tener el sistema una vez desarrollado.

A continuación se muestra una lista de los requisitos funcionales definidos para la realización de la solución:

RF1 Listar ejercicios de agrupamiento disponibles.

RF2 Ejecutar ejercicios de agrupamiento de tarjetas abierto.

RF3 Adicionar categoría.

RF4 Modificar categoría.

RF5 Eliminar categoría.

RF6 Eliminar todas las categorías

RF7 Terminar ejecución.

RF8 Salir ejecución.

RF9 Ejecutar ejercicios de agrupamiento de tarjetas cerrado.

RF10 Ejecutar ejercicios de análisis de secuencia.

RF11 Autenticarse en el sistema.

RF12 Gestionar usuario.

RF13 Adicionar usuario.

RF14 Modificar usuario.

RF15 Eliminar usuario.

RF16 Buscar usuario.

RF17 Gestionar ejercicio.

RF18 Importar ejercicio.

RF19 Exportar ejercicio.

RF20 Eliminar ejercicio.

RF21 Buscar ejercicio.

Requisitos no funcionales:

Los requisitos no funcionales son las características o propiedades que debe tener el sistema una vez terminado, son los que determinan la fiabilidad, usabilidad, apariencia. A continuación se muestran los requisitos no funcionales con los que debe cumplir la solución.

Usabilidad:

- **RNF1: e**l sistema podrá ser utilizado por cualquier persona con conocimientos básicos de informática.
- **RNF2** el sistema debe presentar una interfaz amigable que no represente una dificultad a la hora de su uso.
- **RNF3** el sistema debe estar disponible 24h al día.

Confiabilidad:

- **RNF4** la información manejada dentro del sistema debe estar protegida contra el acceso no autorizado.
- **RNF5** debe hacerse una copia semanal de la información contenida en la base de datos para que en caso de que ocurra algún tipo de incidente que afecte el funcionamiento permanentemente del ordenador servidor se cuente con un respaldo de información.
- **RNF6** el sistema debe contar con un grupo de políticas de accesibilidad que limite el acceso a funcionalidades en dependencia del nivel de accesibilidad que presente un usuario determinado.

Eficiencia:

- **RNF7** el sistema debe soportar una conexión simultánea de más de 60 usuarios.

Soporte:

- **RNF8** el producto debe recibir mantenimiento de forma inmediata ante cualquier fallo que afecte su funcionamiento.

Restricciones de diseño:

- **RNF9** el lenguaje de programación: PHP 5.3 o superior.
- **RNF8** el marco de trabajo base de desarrollo a utilizar: CodeIgniter 2.0.1.
- **RNF10** el IDE empleado será NetBeans 7.0.1.
- **RNF11** como servidor web se utilizará Apache 2.2.21.
- **RNF12** la arquitectura de desarrollo a utilizar será la Modelo Vista Controlador.

Interfaz:

- **RNF13** las interfaces deben presentar similitud con las interfaces de la herramienta de escritorio para la ejecución de ejercicios de agrupamiento de tarjetas y análisis de secuencia.
- **RNF14** las interfaces deben ser lo más intuitivas posibles.

Requisitos de Licencia.

- **RNF15** el sistema desarrollado cumplirá con todas las características de un software libre.

Requisitos de Hardware:

- **RNF16** los ordenadores cliente y servidor deberán tener tarjeta de red.
- **RNF17** el servidor de BD debe tener un procesador de 2.8GHz, 2GB de RAM y un disco duro de 40GB.
- **RNF18** los ordenadores cliente deberán tener un procesador de 1.8GHz mínimo, 256MB de RAM de 200MB de espacio libre en el disco duro.

2.4 Modelo Físico de la Base de Datos

El modelo físico de la base de datos es una descripción de cómo va a quedar implementada la base de datos, en él se incluye la descripción de cómo se almacenan los datos en las tablas y las relaciones que existen entre ellas.

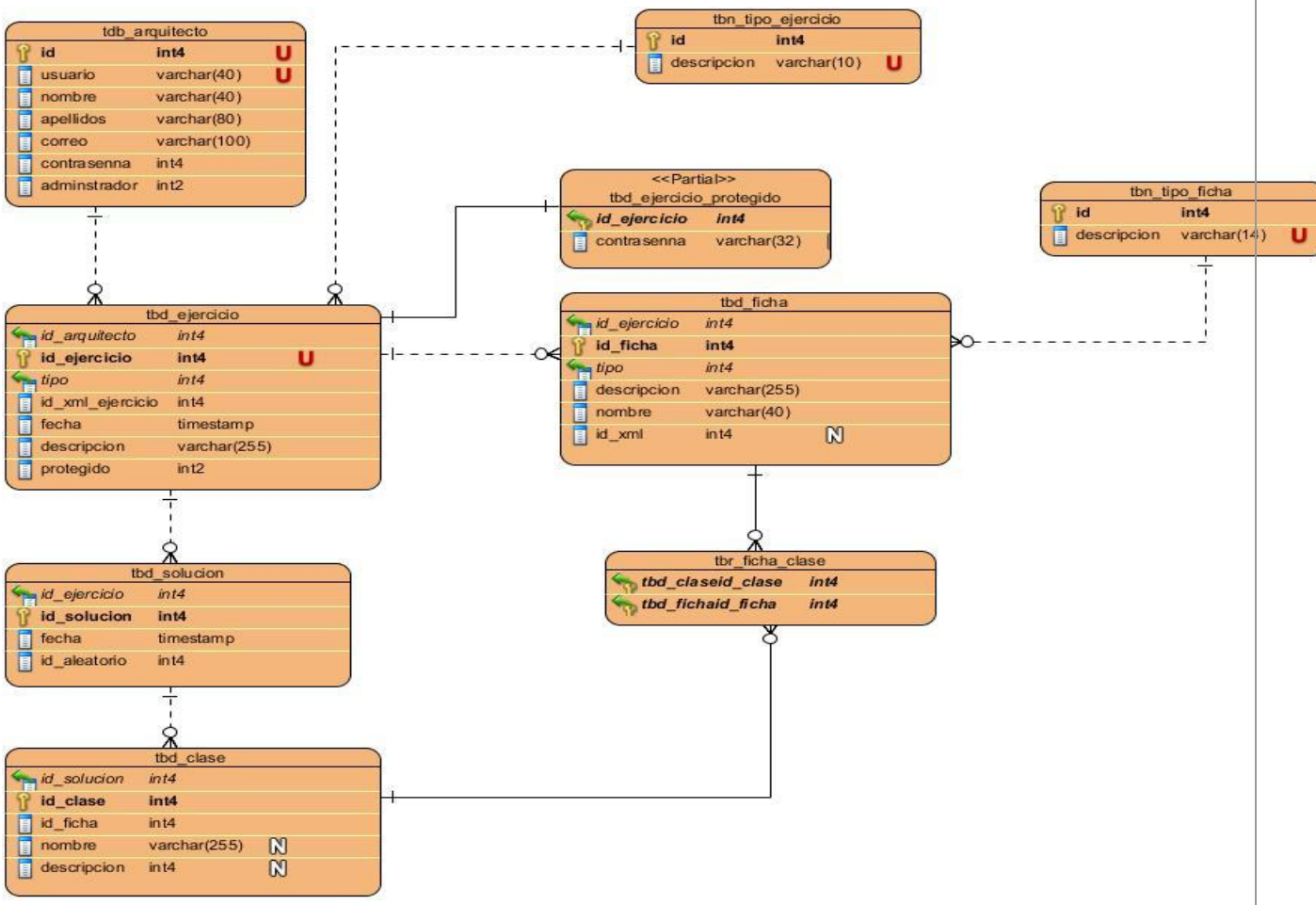


Figura 2. Modelo físico de la base de datos

Diccionario de Datos

En el diccionario de datos se describen las entidades persistentes en un proceso atendiendo a las diferentes características que presenta además de dar una breve descripción de la entidad.

Tabla 2. Descripción de la entidad tbd_arquitecto.

Nombre de la entidad		tbd_arquitecto			
Descripción de la entidad		Contiende los datos de los arquitecto que están registrados en el sistema			
Nombre del	Descripción	Tipo	Puede	Restricciones	Criterio de Selección

atributo			ser nulo	Clases válidas	Clases no válidas	Múltiple	Única
id	Identificador	serial	no	Valores numéricos positivos			si
usuario	Usuario que tiene el arquitecto en el sistema	Character varying	no	Cadenas de caracteres			si
nombre	Nombre del arquitecto	Character varying	no	Cadenas de caracteres		si	
apellidos	Apellidos del arquitecto	Character varying	no	Cadenas de caracteres		si	
correo	Correo del arquitecto	Character varying	no	Cadenas de caracteres			si
contraseña	Contraseña del arquitecto	Character varying	no	Cadenas de caracteres		si	
administrador	Indica si es arquitecto o administrador	boolean	no	True,false, t,f.		si	

2.5 Patrones de diseño

Patrones GRASP

Los patrones GRASP (*General Responsability Assignment Software Patterns*, en español Patrones de Software para la Asignación de Responsabilidades) describen los principios fundamentales de la asignación de responsabilidades a objetos (LARMAN, 1999). En el desarrollo de la aplicación se utilizaron cinco de los nueve patrones GRASP:

- El patrón Experto soluciona el problema de asignar una responsabilidad de forma general, al recomendar para esta función a la clase que maneje la información necesaria. La clase gestionar_usuario_mdl es la encargada de manejar todos los aspectos referentes al negocio de los datos de los usuarios del sistema.

- El patrón Bajo Acoplamiento recomienda asignar las responsabilidades de tal forma que se le brinde soporte para una mayor reutilización y poca dependencia. La clase gestion_ejercicio_cc presenta un bajo acoplamiento pues para realizar sus funcionalidades depende únicamente de la clase gestion_ejercicio_mdl

- El patrón Alta Cohesión aconseja mantener la clase lo más cohesionada posible para controlar la complejidad de la misma. La clase gestion_usuario_cc realiza una función única dentro del sistema, como su nombre indica, es la encargada de la gestión de usuario y ninguna otra clase realiza sus funcionalidades dentro del sistema.

- El patrón Controlador ayuda a decidir quién se encarga de administrar un evento del sistema. La clase ejecucion_ejercicio_cc es el encargado de controlar las funcionalidades de ejecución durante la ejecución de un ejercicio.

Patrones GoF (Gang of Four).

- **Fachada:** el patrón Fachada propone crear una clase que unifique las funciones de un conjunto de clases y que esta sea la encargada de colaborar con el sistema (LARMAN, 1999), se requiere de una interfaz común, unificada para un conjunto de implementaciones o interfaces dispares, como en un subsistema. Podría no ser conveniente acoplarla con muchos elementos del subsistema, o la implementación del subsistema podría cambiar. Para solucionar este problema, defina un único punto de conexión con el subsistema, un objeto fachada que envuelva el

subsistema. Este objeto fachada presenta una única interfaz y es responsable de colaborar con los componentes del subsistema**.**

2.6 Arquitectura del sistema.

Estilo arquitectónico.

Para el desarrollo de la solución se empleara la arquitectura **Cliente – Servidor** que es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes (CSI, 2010).

Patrón arquitectónico Modelo Vista Controlador (MVC):

Es un patrón de arquitectura que se utiliza frecuentemente para desarrollar aplicaciones web. Se utiliza separando tres componentes: el modelo, representa la información con la que trabaja la aplicación, es decir, la lógica del negocio, la vista, que es la capa de presentación que interactúa con al usuario y el controlador, que es el encargado de procesar las interacciones del usuario y actualizar la vista (Labrada, 2011).

Modelo de Diseño.

El marco de trabajo Codeigniter utiliza el patrón arquitectónico MVC y propone un modelo de diseño por el cual deben regirse las aplicaciones que se desarrollan con él.

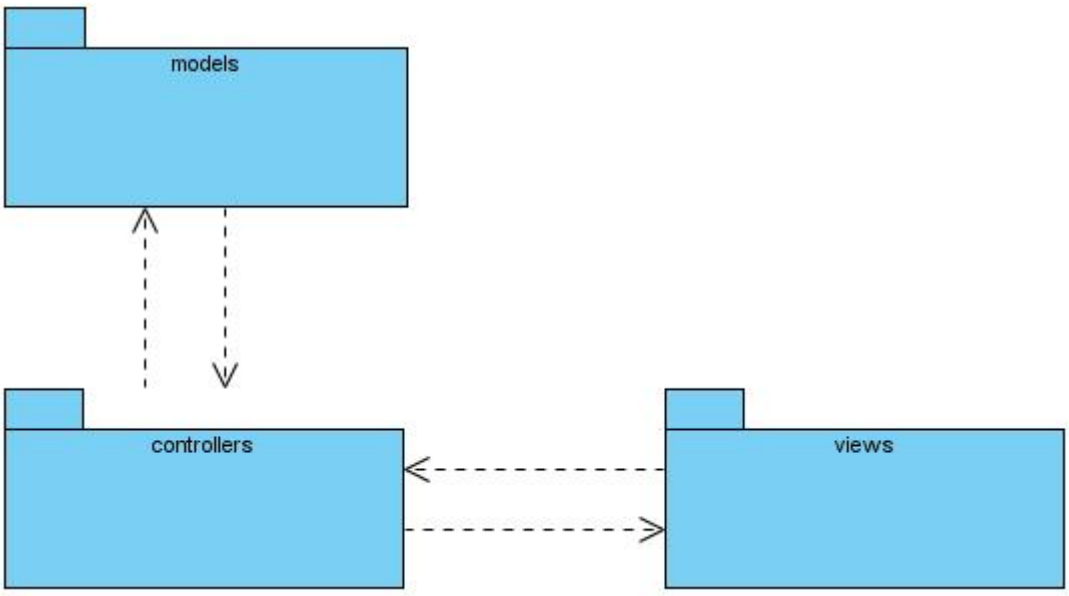


Figura 3. Modelo de diseño de la aplicación

Dentro del paquete views (vistas) se van a encontrar todas las páginas que contengan código HTML y que realicen la función de interfaces e interactúen los usuarios.

Dentro del paquete controllers (controladores) se encuentran todas las clases encargadas de dar respuesta a las interacciones del usuario con la aplicación y de actualizar la vista en dependencia de la respuesta.

Dentro del paquete models (modelos) se encuentran las clases encargadas de la lógica del negocio ya que son los encargados de manipular todos los datos del negocio y de dar respuesta a las peticiones de las clases controladoras.

2.7 Modelo de despliegue

"El modelo de despliegue es un modelo de objetos que describe la organización física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo" (JACOBSON *et al.*, 2000). El siguiente diagrama describe la instancia de los nodos en que se ejecuta la aplicación una vez instalada, que estará compuesto por un ordenador cliente donde el usuario va a acceder a la aplicación instalada en el ordenador servidor, a través del protocolo HTTP, que a su vez va a consultar los datos referentes al negocio con el servidor web a través de del protocolo TCP/IP.

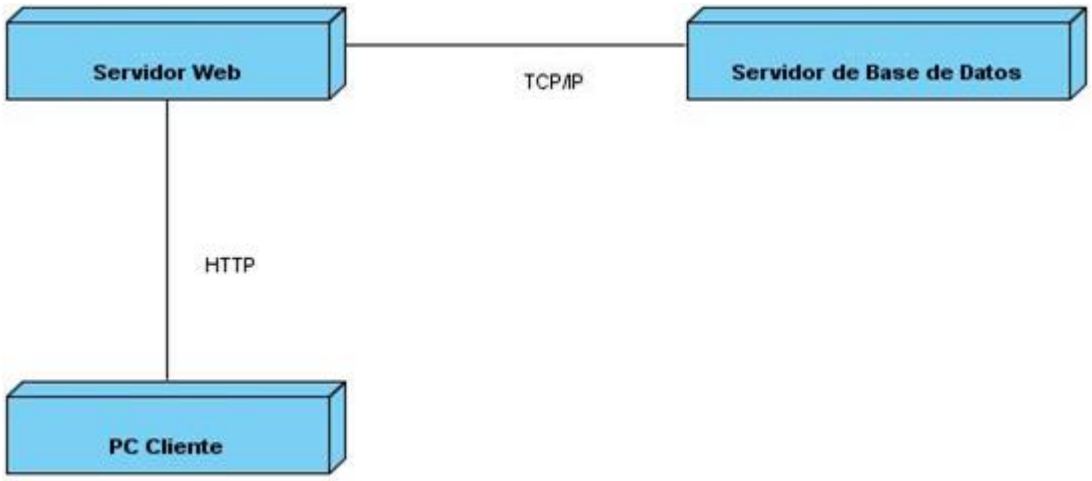


Figura 4. Modelo de despliegue

2.8 Conclusiones del capítulo:

- Con la creación del modelo de dominio se logró un mejor entendimiento sobre el funcionamiento del negocio.
- La captura de requisitos facilitó el entendimiento de las necesidades que presenta el proyecto Abad.
- El uso de patrones de diseño garantiza la realización de un código legible, reutilizable, cohesionado lo que incrementa la calidad del producto final.
- El modelo de despliegue muestra cómo queda distribuido el sistema una vez que se encuentre instalado.

Capítulo 3. Implementación y prueba

3.1 Introducción:

En este capítulo se reflejan los estándares de codificación empleados, aspectos que se tuvieron en cuenta a la hora de implementar tales como el tratamiento de errores y aspectos de seguridad así como la realización de pruebas.

3.2 Estándares de codificación empleados.

Un estándar de codificación comprende los aspectos de la generación del código y repercute directamente en la legibilidad y en la calidad del mismo. Para el desarrollo de la propuesta se decidió usar estándares de codificación sólidos haciendo uso de las buenas prácticas de programación basándose en los establecidos por el centro CENIA (Centro de Informatización Universitaria) en el documento CENIA_Estandar_de_codigo del expediente de proyecto en su versión 1.0.

Identación, llaves de apertura y cierre, y tamaño de las líneas.

Usar una identación sin tabulaciones, con un equivalente a 4 espacios, para mantener integridad en las revisiones svn. El uso de las llaves será en una nueva línea. La longitud de las líneas de código es aproximadamente de 75-80 caracteres. Para mantener la legibilidad del código.

Ejemplo:

```
1 ....$a = $b;
2
3 ....function ejemplo()
4 ....{
5     ....//BI
6 ....}
```

Figura 5. Ejemplo de utilización de llaves (GARCÍA VIDAL, 2010) .

Variables

Se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.

Ejemplo:

```
1 ....$variable
2 ....$variableNombreCompuesto
```

Figura 6. Ejemplo de declaración de variables (VIDAL, 2010).

Constantes

Siempre debe ser todo en mayúsculas, con caracteres de subrayado "_" para separar palabras en caso de nombres compuestos.

Ejemplo:

```
1 ....define(CONSTANTE,valor);
2 ....define(CONSTANTE_COMPUESTO,valor);
```

Figura 7. Ejemplo de declaración de constantes (VIDAL, 2010).

Funciones

Se rigen por la nomenclatura camelCase. Los parámetros son separados por espacio luego de la coma que los separa

```
1 ....function funcion($parametro1,.$parametro2)
2 ....{
3     ....//BI
4 ....}
5
6 ....function funcionNombreCompuesto($parametro1,.$parametro2)
7 ....{
8     ....//BI
```

Figura 8. Ejemplo de declaración de funciones (VIDAL, 2010).

Estructuras de control

Se incluye if, for, foreach, while, switch, entre las estructuras de control y los paréntesis debe de existir un espacio. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales. Esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos.

Ejemplo:

```
1 ....if.(condicion)
2 ....{
3     ....//BI
4 ....}
5 ....elseif(condicion)
6 ....{
7     ....//BI
8 ....}
9 ....else
```

```
10....{
11    ....//BI
12....}
13
14....switch.(valor)
15....{
16    ....case valor1:
17        ....//BI para valor1
18    ....break;
19    ....case valor2:
20        ....//BI para valor2
21    ....break;
22    ....default:
23        ....//BI por defecto
24....}
```

Figura 9. Ejemplo de estructuras de control (VIDAL, 2010).

Si las condiciones son muy largas que sobrepasan el tamaño de la línea, estas se dividen en varias líneas

Ejemplo:

```
1 ....if.(condicion1
2 ....|| condicion2)
3 ....||(condicion3
4 ....&& condicion4))
5 ....{
6     ....//BI
7 ....}
```

Figura 10. Ejemplo de declaración de condiciones (VIDAL, 2010).

En el mejor de los casos cuando la condición es muy extensa, se puede dividir esta en variables y compararlas dentro de la estructura de control.

Ejemplo:

```
1 ....$variableCondicion1 = condicion1 || condicion2;
2 ....$variableCondicion2 = condicion3 && condicion4;
3
4 ....if.($variableCondicion1 || $variableCondicion2)
5 ....{
6     ....//BI
7 ....}
```

Figura 11. Ejemplo de declaración de condiciones (VIDAL, 2010).

Documentación

Todos los archivos deben tener la documentación asociada al mismo. Para esto debe cumplir con el siguiente bloque al principio de cada clase.

Ejemplo de Clase:

```
1 /**
2 *Breve descripción de la clase
3 *
4 *PHP versión #
5 *
6 *@category Categoría de la clase implementada "Libreria,
7 * Controladora, Modelo"
8 *@package  Nombre del paquete o módulo al que pertence
9 *@author  Nombre y Apellidos del autor y  correo electrónico
10*/
```

Figura 12. Ejemplo de documentación de una clase (VIDAL, 2010).

Ejemplo de función:

```
1 /**
2 *Breve descripción de la clase
3 *
4 *PHP versión #
5 *
6 *@category Categoría de la clase implementada "Libreria,
7 * Controladora, Modelo"
8 *@package  Nombre del paquete o módulo al que pertence
9 *@author  Nombre y Apellidos del autor y  correo electrónico
10*/
```

Figura 13. Ejemplo de documentación de una función (VIDAL, 2010).

Buenas prácticas

Los valores booleanos y nulos siempre se escriben con mayúscula, para facilitar la legibilidad del código se deja una línea antes de las estructuras de control y definición de las funciones.

Ejemplo:

```
1 ....$variableBooleana = FALSE;
2 ....$variableNula = NULL;
3 ....
4 ....if(condicion)
5 ....{
6     ....//BI
7 ....}
```

Figura 14. Ejemplo de buenas prácticas (VIDAL, 2010).

3.3 Tratamiento de errores

La posibilidad de que se introduzcan o aparezcan errores durante el desarrollo del software o en su funcionamiento es prácticamente inevitable, ya sean introducidos por el desarrollador durante la elaboración del producto o producidos por el usuario, intencional o inconscientemente, lo que puede atentar directamente a la disponibilidad y correcto funcionamiento de la aplicación.

Para dar solución a ello se procura que todos los datos introducidos sean correctamente validados antes de ser procesados y en caso de ocurrencia de algún tipo de error en los datos, se muestra la información necesaria al usuario que esta interactuando con la aplicación.

3.4 Seguridad

El aspecto de la seguridad es de vital importancia en una aplicación web, al igual que una aplicación no se puede declarar libre de errores tampoco se puede declarar impenetrable a ataques producidos hacia ella.

Para ello se ha de garantizar la correcta disponibilidad de la información en dependencia del usuario que esté interactuando con la aplicación. Todos los datos que son introducidos en la aplicación deben ser correctamente validados además de que son filtrados contra ataques

XSS(*Cross Site Scripting*) con un filtro que el marco de trabajo CodeIgniter posibilita, el cual es habilitado para que se ejecute automáticamente cada vez que se encuentren datos POST o COOKIE (COMPAÑÍA ELLISLAB INC., 2011).

Toda la introducción de datos a la base de datos es filtrada contra inyecciones a través de la función $this->db->escape_str($cadena), la cual escapa los datos pasados a ella independientemente del tipo que sea (COMPAÑÍA ELLISLAB INC., 2011). Se controla el tamaño de las cadenas que son enviadas al servidor. No se permite la introducción de caracteres extraños a servidor. El acceso a la aplicación se realiza mediante el control de acceso basado en roles (RBAC) lo que permite controlar cuáles usuarios tienen acceso a recursos basados en el rol que ocupan. El acceso a los recursos se restringe a los usuarios que han sido autorizados para asumir el rol asociado.

3.5 Pruebas

El desarrollo de un software implica una serie de actividades de producción las cuales no quedan libres del fallo humano, por lo que la aparición de errores está presente durante todo el ciclo de vida del software. La realización de pruebas es de vital importancia, pues garantiza que el producto final llegue a manos del cliente con un correcto funcionamiento y una alta calidad. En este capítulo se aborda el tema referente a la realización de pruebas a la solución obtenida.

Pruebas de Software

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa. Estas involucran las operaciones del sistema evaluando los resultados bajo condiciones controladas, lo que hace que la realización de pruebas a los programas constituya un factor de vital importancia. Antes de liberar el producto es necesario tener la certeza de que este se encuentra libre de errores. Aunque se considera que no se puede estar totalmente seguro de esto, mientras más pruebas se le realicen al programa más cerca se podrá estar de lograr un producto con la calidad esperada por los usuarios. Esta etapa de pruebas implica los siguientes aspectos:

- Verificar la interacción de componentes.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.

- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.

- Diseñar pruebas que sistemáticamente saquen a la luz diferentes tipos de errores, con la menor cantidad de tiempo y esfuerzo.

Pruebas de Unidad

Es la escala más pequeña de las pruebas, está basada en la funcionalidad de los módulos del programa, como funciones, procedimientos y módulos de clases. En ciertos sistemas también se verifican o se prueban los drivers y el diseño de la arquitectura (PRESSMAN, 2002).

Los casos de pruebas que se diseñen para este nivel de pruebas, deben descubrir errores como:

- Comparaciones entre tipos de datos distintos.

- Operadores lógicos o procedencia incorrecta.

- Igualdad esperada cuando los errores de precisión la hacen poco probable.

- Variables o comparaciones incorrectas.

- Fallo de salida cuando se encuentra una iteración divergente.

Se probaron diez fragmentos de código de diversas clases, con el objetivo de involucrar la mayor diversidad de clases posible. Luego de probadas las unidades, se detectó que todas funcionaban correctamente, lo que no quiere decir que todas las funciones del sistema lo hagan, para asegurarse de ello es necesario realizar otras pruebas que abarquen más código. A continuación se muestra uno de los fragmentos de código probados mediante el método de camino básico.

```
$ruta_temp = $_FILES['ruta']['tmp_name'];
$rutaNombre = $_FILES['ruta']['name'];
$protegido = FALSE;
$contrasenna = "";
if ($this->input->post('protegido', TRUE))
{
    $prot = $this->input->post('protegido', TRUE);
    if ($prot === 'on')
    {
        $protegido = TRUE;
        if ($this->input->post('contrasenna', TRUE))
        {
            $contrasenna = $this->input->post('contrasenna', TRUE);
        }
        else
        {
            echo json_encode(array(
                "error" => TRUE,
                "mensaje" => 'Contraseña Incorrecta'
            ));
            die();
        }
    }
```

Figura 15. Fragmento de código estudiado para realizar pruebas unitarias

Luego de analizar el código se procede a la creación del grafo, los nodos son generados a partir de las secuencias, condiciones y ciclos.

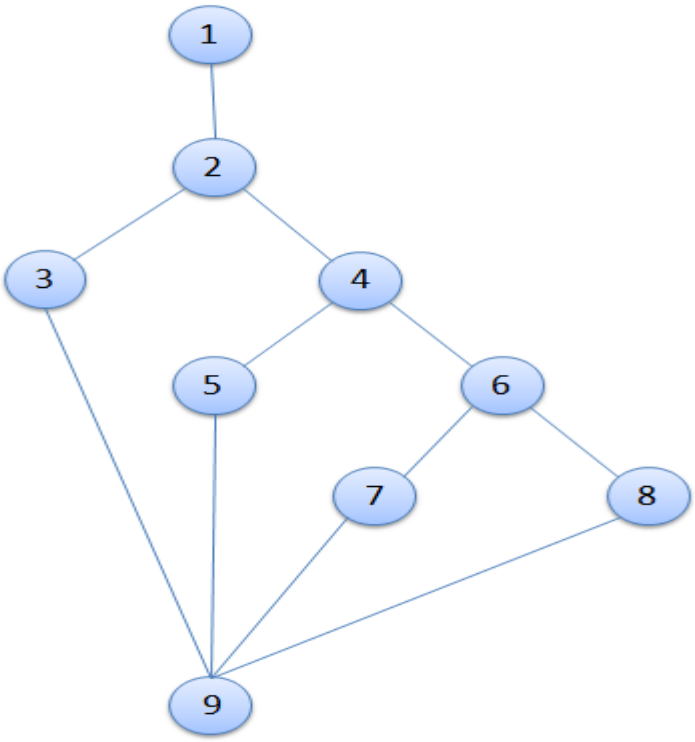


Figura 16. Grafo de camino mínimo del fragmento de código seleccionado

Este grafo se usa para calcular la complejidad ciclomática del módulo a probar. Para ello son usadas tres vías distintas, cada una de ellas se define por una fórmula, que independientemente de cuál de ellas se use, el resultado siempre debe ser el mismo.

Primera fórmula:

V (G) = (A - N) + 2 (donde A es el número aristas del grafo y N el número de nodos.)

V (G) = (11-9) + 2

V (G) = 4

Segunda fórmula:

V (G) = P + 1 (donde P es la cantidad de nodos desde los cuales se pueden tomar dos caminos)

V (G) = 3 + 1

V (G) = 4

Tercera fórmula:

V (G) = R (donde R es la cantidad de regiones)

V (G) = 4

Luego de la aplicación de las tres fórmulas se puede apreciar que el resultado obtenido en todos los casos es el mismo, cuatro. Esto sirve para determinar que la cantidad de caminos básicos que pueden ser tomados son cuatro en total, es decir el procedimiento puede tomar por cuatro caminos distintos.

Caminos que pueden ser tomados: 1,2,3,9; 1,2,4,5,9; 1,2,4,6,7,9; 1,2,4,6,8,9

El siguiente paso es probar cada uno de los caminos, asegurando que los datos de entrada necesarios para ejecutar el módulo son correctos, para luego obtener, al término de cada prueba, una respuesta como resultado a la correcta evaluación de la función.

A partir del resultado obtenido en la prueba de camino mínimo se procede a realizar una descripción de casos de prueba con los posibles caminos del grafo resultante:

Tabla 3. Pruebas de camino mínimo.

Escenario	Descripción	Variable 1 "ruta"	Variable 2 "protegido"	Variable 3 "contraseña"	Variable 4 "repetir contraseña"	Respuesta del sistema	Flujo central
EC 1.1 Importar fichero correctamente	Se adiciona un nuevo ejercicio de agrupamiento a través de un fichero xml al sistema	V pruebaEOT.xml	V true	V ishtar	V ishtar	El sistema añade el ejercicio al sistema	1.El usuario se autentica en el sistema como arquitecto. 2. El usuario selecciona la opción importar. 3.El sistema muestra un formulario con los campos necesario para importar el fichero. 4. El usuario completa correctamente los campos necesarios y envía los datos al sistema. 5. El sistema guarda los datos del ejercicio enviado.
		V AS.xml	V false				
EC 1.2 Importar fichero con campos vacíos o incorrectos	No se adiciona un nuevo ejercicio al sistema, mostrando un mensaje indicando cual es el error	I (vacío)	V false	V (vacío)	V (vacío)	El sistema muestra un mensaje indicando donde se encuentra el error y no se añade ningún dato al sistema	1.El usuario se autentica en el sistema como arquitecto. 2. El usuario selecciona la opción importar. 3.El sistema muestra un formulario con los campos necesario para importar el fichero. 4.El usuario completa los datos necesarios de manera incorrecta. 5.El sistema verifica los datos y no guarda los datos en el sistema.
		V Abierto.xml	V true	V ashgar	I (vacío)		
		V Abierto.xml	V true	I (vacío)	I (vacío)		

Pruebas de Aceptación

Las pruebas de aceptación constituyen pruebas de caja negra que se centran en el cumplimiento de los requisitos definidos para el sistema una vez concluido. Por lo tanto, estas pruebas constituyen la validación de la aplicación por parte del usuario final y según Pressman plantea lo siguiente:

“La validación del software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requisitos”. Un plan de prueba traza la clase de pruebas

que se han de llevar a cabo, y un procedimiento de prueba define los casos de prueba específicos en un intento por descubrir errores de acuerdo con los requisitos.

Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen todos los requisitos funcionales, que se alcanzan todos los de rendimiento, que la documentación es correcta e inteligible y que se alcanzan otros requisitos (por ejemplo, portabilidad, compatibilidad, recuperación de errores, facilidad de mantenimiento)" (PRESSMAN, 2002). Dentro de las pruebas de aceptación no formales, realizadas en software a la medida del cliente se encuentran las pruebas alfa y beta. Estas pruebas son muy convenientes cuando se desarrollan aplicaciones que van a ser utilizadas por muchos clientes, con el objetivo de descubrir errores que parezca que sólo el usuario final puede descubrir.

Al sistema resultante se le realizaron 3 iteraciones, una primera donde, de 21 requisitos funcionales evaluados, se detectaron 16 no conformidades de las cuales las 16 fueron resueltas. En una segunda iteración realizada a la misma cantidad de requisitos, se detectaron 14 no conformidades las cuales fueron resueltas satisfactoriamente. Se realizó una tercera iteración para verificar que se arreglaron las no conformidades anteriores, además de buscar nuevas no conformidades se detectaron 11 no conformidades a las que se les dio solución inmediata. La realización de estas iteraciones permitió lograr una mejora en el desempeño de la aplicación, pues se mejoró la interacción producto cliente además de mejora el rendimiento del sistema.

3.6 Conclusiones del capítulo:

- La utilización de estándares de códigos garantiza un código de alta calidad haciendo uso de las buenas prácticas de programación.
- El correcto tratamiento de errores y de aspectos de seguridad garantiza un correcto funcionamiento de la aplicación.
- La realización de pruebas permite la detección de errores no detectados en la fase de desarrollo, lo cual permite localizarlos y rectificarlos mejorando el desempeño de la solución.

Conclusiones

La presente investigación ha permitido el cumplimiento del objetivo trazado, desarrollar una solución web para el paquete Abad que permita la ejecución de ejercicios de las técnicas de agrupamiento de tarjetas y análisis de secuencia posibilitando aplicar dichas técnicas a usuarios geográficamente dispersos por lo que se concluye que:

- El estudio de herramientas homólogas demostró que no existe actualmente una herramienta web que permita la ejecución de la técnica de análisis de secuencia lo cual se implementó en la aplicación siendo esto un aporte importante para la plataforma Abad.
- La aplicación desarrollada permite la ejecución de las técnicas de agrupamiento de tarjetas y análisis de secuencia a usuarios geográficamente dispersos, además de importar y generar ficheros xml totalmente compatibles con la herramienta de escritorio que existe actualmente en el proyecto Abad.
- La implementación de la aplicación se realizó utilizando estándares de códigos, permitiendo un claro entendimiento y buena legibilidad del código haciendo uso de buenas prácticas de programación.
- La realización de pruebas permitió detectar no conformidades y deficiencias, las cuales, al ser eliminadas, garantizan un mejor desempeño de la aplicación.

Recomendaciones

En vista de mejorar el funcionamiento y rendimiento de la aplicación desarrollada se recomienda:

- Desarrollar una funcionalidad que permita el envío, mediante correo electrónico, de una dirección URL
- Continuar la realización de pruebas de seguridad a fin de encontrar posibles vulnerabilidades no detectadas.

Bibliografía referenciada

ACHOUR, M.; BETZ, F.*, et al. PHP Manual* [Página Web: Comunidad de Desarrolladores]. The PHP Group, [Consultado el: febrero de 2012]. Disponible en: http://php.net/manual/es/intro-whatis.php.

ALVAREZ, M. A. *Qué es jQuery, para qué sirve y qué ventajas tiene el utilizar este framework Javascript* [Página Web: Foro Temático]. desarrolloweb.com, [Consultado el: noviembre de 2011]. Disponible en: http://www.desarrolloweb.com/articulos/introduccion-jquery.html.

ALVAREZ, R. *Qué es y para qué sirve el SQL* [Página Web: Foro Temático]. desarrolloweb.com, [Consultado el: diciembre de 2010]. Disponible en: http://www.desarrolloweb.com/articulos/262.php.

BALMASEDA BORLADO, F. *Herramienta de ordenamiento de tarjetas versión 2.0 para la plataforma de arquitectura de la información Abad.* Tutor: Rodríguez, Y. A. y Pompa, D. B. B. Grado, Facultad 1. Universidad de las Ciencias de la Informática, 2011.

CELESTE, L. P. A Modified Delphi Approach to a New Card Sorting Methodology. *Journal of Usability Studies*, 2008, vol. 4, nº 1, p. 7-30. Disponible en: http://www.upassoc.org/upa_publications/jus/2008november/paul1.html.

CHAVES, M. A. La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software. *Revista InterSedes*, 2007, vol. VI, nº 10, p. 13. Disponible en: http://www.latindex.ucr.ac.cr/intersedes10/10-art_11.pdf. ISSN 1409-4746.

COMPAÑÍA ELLISLAB INC. *CodeIgniter User Guide Version 2.1.0* [Guía del Usuario]. EllisLab, 2011, Disponible en: http://codeigniter.com/user_guide/.

ESPINOSA RAMÍREZ, J. G. y JIMÉNEZ MORALES, Y. *Implementación del modulo "Técnica de ordenamiento de tarjetas" para la plataforma de arquitectura de la información "Abad". .* Tutor: Rodríguez., Y. A. y Almenares, K. P. Grado, Facultad 9. Universidad de las Ciencias Informáticas, 2010.

GARCÍA VIDAL, Y. *CENIA_Estandar_de_codigo-v1.0.* [Guía]. La Habana: Centro CENIA. Facultad 1. Universidad de las Ciencias Informáticas, 2010,

GRUPO DE CALIDAD DE SOFTWARE. *Programa de mejora de los procesos productivos en la universidad de las ciencias informáticas.* [Guía]. CALISOFT, 2011,

JACOBSON, I.; BOCH, G.*, et al. El Proceso Unificado de Desarrollo de Software.* Madrid Addison Wesley, 2000.

JAMES GARRET, J. *The Elements of User Experience.* 2 ed. New York: New Riders Publishing, 2002.

KAIR, M. J. *La Biblia de Servidor Apache 2.* Madrid: Anaya Multimedia, 2003. ISBN 84-415-146.

LARMAN, C. *UML y Patrones: Introducción al análisis y programación orientada a objetos.* México: Prentice Hall, 1999. ISBN 970-17-026.

MONTES DE OCA, A. S. D. B. *Arquitectura de información y usabilidad: nociones básicas para los profesionales de la información.* Ciudad de La Habana: Dirección de Información Científico- Técnica, 2004,

ORACLE CORPORATION. *NetBeans IDE 7.1 Features* [Página web: Comunidad de Desarrollo]. NetBeans, [Consultado el: diciembre de 2011]. Disponible en: http://netbeans.org/features/php/.

OROVIO PINO, J. M. *Análisis y Diseño de un sistema para la aplicación de técnicas de Card Sorting en la obtención de Arquitecturas de información.* Tutor: Garay, S. G. Grado, Facultad 7. Universidad de las Ciencias Informáticas, 2009.

P.LETELIER. *Rational Unified Process (RUP).* Universidad Politécnica de Valencia., 2011, Disponible en: https://pid.dsic.upv.es.

PÉREZ GARCÍA, F. U. *Tipo de lenguaje de programación* [Página Web: Publicación Digital]. La Revista Informatica, [Consultado el: diciembre de 2011]. Disponible en: http://www.larevistainformatica.com/tipo-lenguaje-programacion.htm.

PRESSMAN, R. S. *Ingeniería del software, un enfoque práctico.* Madrid: Madrid McGraw-Hill 2002. 601 p.

RONDA LEÓN, R. Revisión de técnicas de arquitectura de información. *No Solo Usabilidad journal*, 2007, vol. 6, nº Disponible en: http://www.nosolousabilidad.com/articulos/tecnicas_ai.htm. ISSN 1886-8592.

SOLA PADILLA, Y. *Análisis de un sistema automatizado a través de técnica de Card Sorting u Ordenación de Tarjetas.* Universidad de las Ciencias Informáticas, 2008.

THE PGADMIN DEVELOPMENT TEAM. *pgAdmin PostgresSQL Tools.* [Cliente de bases de datos]. 1.14.0 ed. 2011,

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL 8.3.9 Documentation.* United State: University of California, 2008.

TITTEL, E. y BURMEISTER, M. C. *HTML 4 for dummies.* Indianapolis: Wiley Publishing, 2005. 68 p. ISBN 978-0-7645-8917-1.

VIDAL, Y. G. *CENIA_Estandar_de_codigo-v1.0.* CENIA, 2010,

W3C ORGANIZATION. *Características de Accesibilidad de CSS* [Página Web: Publicación web]. Usabilidad-Web.com [Consultado el: diciembre de 2011]. Disponible en: http://www.usabilidad-web.com/articulos-usabilidad/accesibilidad/accesibilidad-css.html.

WARFEL, T. y SPENCER, D. *Card sorting: a definitive guide* [Página Web: Blog colectivo]. Boxes and Arrows, [Consultado el: diciembre de 2011]. Disponible en: http://www.boxesandarrows.com/view/card_sorting_a_definitive_guide.

Bibliografía consultada

BEIGHLEY, L. *Head First SQL.* O'Reilly Media, Inc, 2007. 586 p. ISBN 0-596-52684-9.

BLANCHARD, J. *Applied jQuery DEVELOP AND DESIGN.* Berkeley: Peachpit Press, 2012. ISBN 978-0-321-77256-5.

CASTLEDINE, E. y SHARKIE, C. *jQuery: Novice to Ninja.* Cambridge Street Collingwood: SitePoint Pty. Ltd., 2010. 431 p. ISBN 978-0-9805768-5-6.

CONVERSE, T.; PARK, J.*, et al. PHP5 and MySQL® Bible.* Indianapolis: Wiley Publishing, Inc, 2004. 1083 p. ISBN 0-7645-5746-7.

CROCKFORD, D. *JavaScript: The Good Parts.* Sebastopol,: O'Reilly Media, Inc., 2008. 262 p. ISBN 978-0-596-51774-8.

FLANAGAN, D. *jQuery Pocket Reference.* United States of America: O'Reilly Media, Inc, 2011. 158 p. ISBN 978-1-449-39722-7.

MACCAW, A. *JavaScript Web Aplications.* United States of America: O'Reilly Media, Inc, 2011. 280 p. ISBN 978-1-449-30351-8.

MYER, T. *Professional CodeIgniter.* Indianapolis,: Wiley Publishing, Inc, 2008. ISBN 978-0-470-28245-8.

SNYDER, C. y SOUTHWELL, M. *Pro PHP Security.* New York: Appres, 2005. 529 p. ISBN 1-59059-508-4.

STINSON, B. *PostgreSQL Essential Reference.* New Riders Publishing, 2001. 400 p. ISBN 0-7357-1121-6.

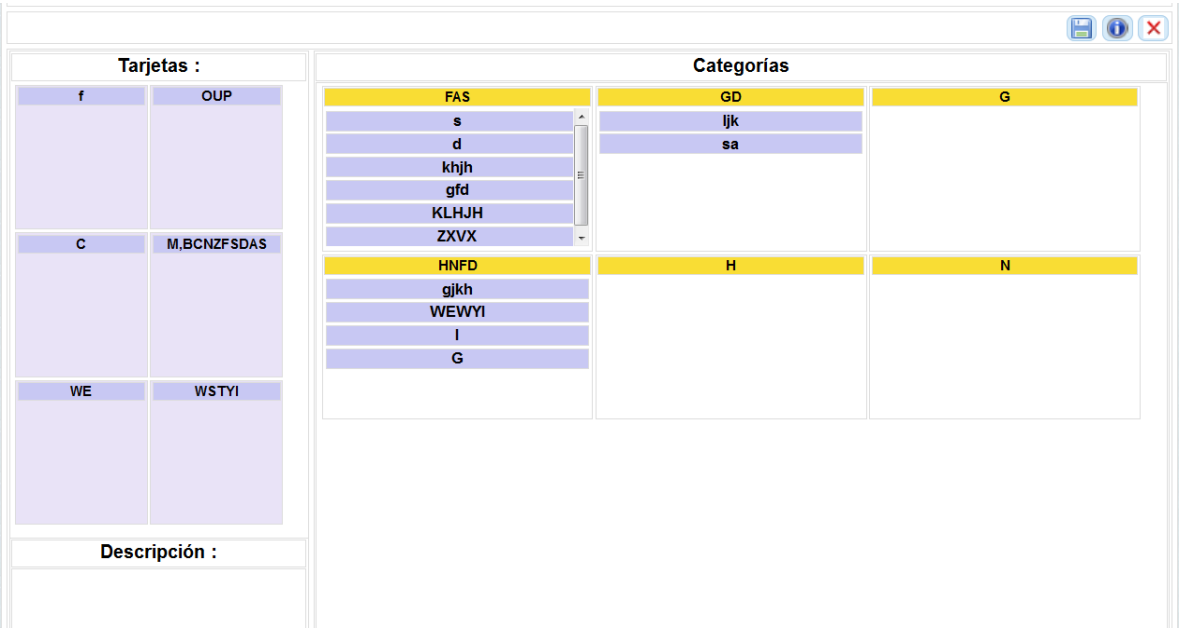
Glosario de términos

- **AI**: Arquitectura de la Información.
- **APACHE**: es un servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, Windows, otras).
- **Arquitectura Cliente/Servidor**: es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en elementos independientes que cooperan entre sí para intercambiar información, servicios o recursos
- **CMMI**: Modelo de Madurez de Capacidades Integrado (por sus siglas en inglés, *Capability Maturity Model Integration*).
- **EOT**: Ejercicios de agrupamiento de tarjetas.
- **HTML**: *HyperText Markup Language.* Lenguaje usado para escribir documentos para servidores World Wide Web. Es una aplicación de la ISO Standard 8879:1986.
- **HTTP:** *HyperText Transfer Protocol.* Protocolo de Transferencia de Hipertextos. Modo de comunicación para solicitar páginas Web
- **MVC:** Modelo Vista Controlador.
- **PHP**: *Hypertext Preprocessor*. Es un ambiente script del lado del servidor que permite crear y ejecutar aplicaciones Web dinámicas e interactivas. Con PHP se pueden combinar páginas HTML y scripts. Con el objetivo de crear aplicaciones potentes.
- **PostgreSQL**: es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) libre.
- **Software**: Programas de sistema, utilerías o aplicaciones expresados en un lenguaje de máquina.
- **SQL**: *Structured Query Language*. Es un lenguaje declarativo de acceso a bases de datos que permite especificar diversos tipos de operaciones sobre las mismas. Aúna características del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar información de interés de una base de datos.

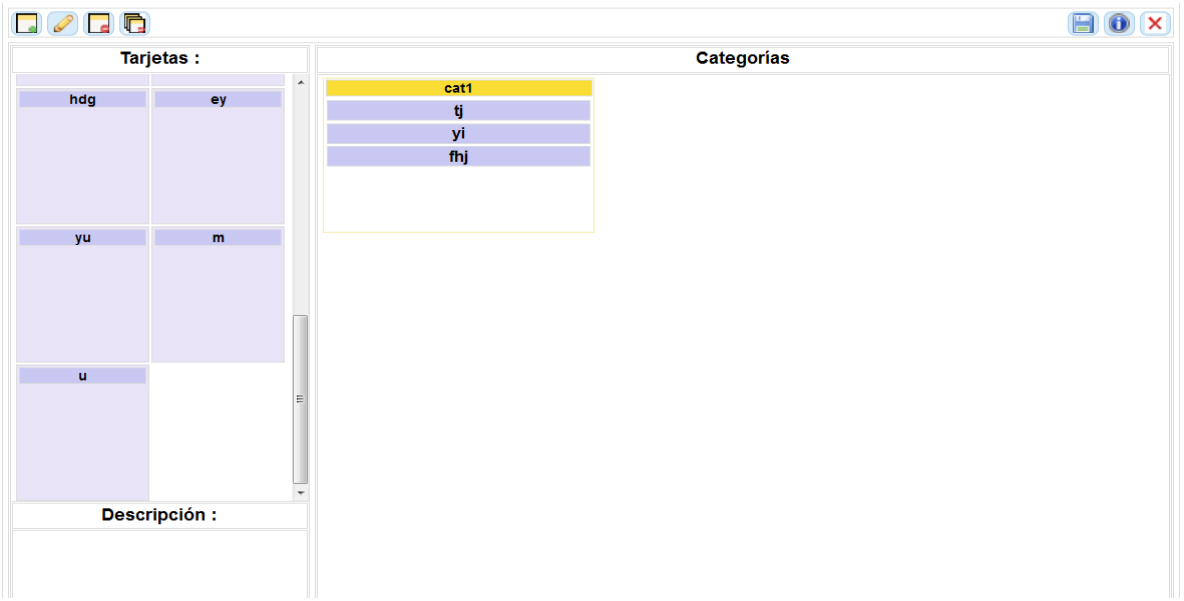
- **UML**: Lenguaje unificado de modelado.
- **WEB**: Red de documentos HTML intercomunicados y distribuidos entre servidores del mundo entero.
- **XML**: *Extensible Markup Language*. Es un lenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium. Orientado principalmente al almacenamiento, procesamiento y transmisión de mensajes

Anexo 1. Interfaz de inicio donde se muestran los ejercicios diponibles.

Anexo 2. Interfaz de ejecución de ejercicios de ordenamiento de tarjetas cerrado.



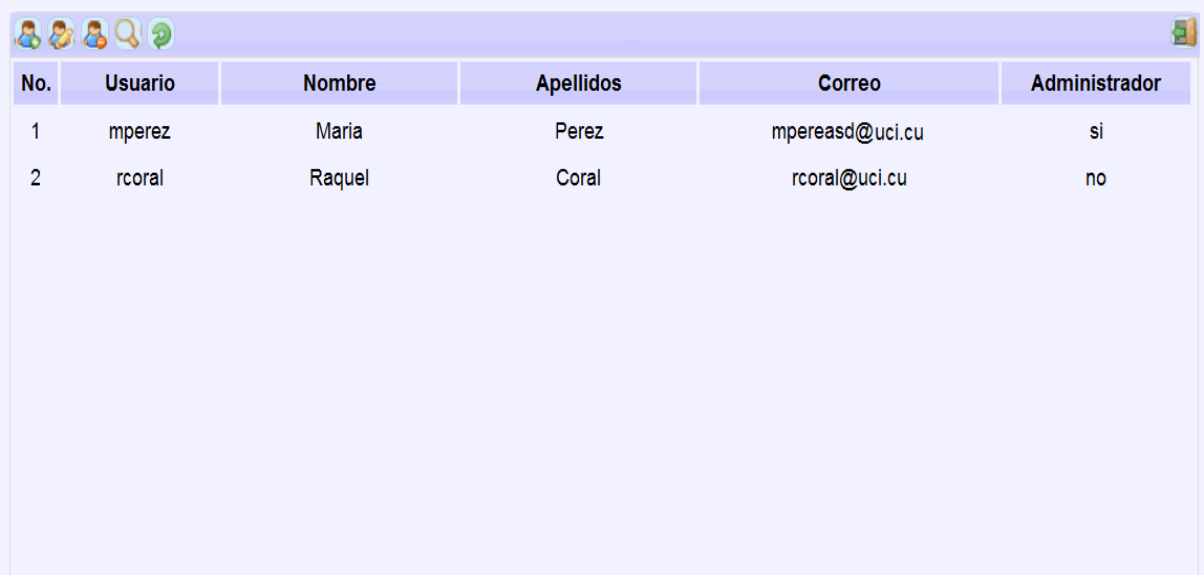
Anexo 3. Interfaz de ejecución de ejercicios de agrupamiento de tarjetas abierto.



Anexo 4. Interfaz de gestión de ejercicio.

No.	Nombre	Fecha	Tipo	Cantidad de Soluciones
1	abierto1	Sunday, April 29, 2012 3:34:36 PM EDT	Agrupamiento de Tarjetas Abierto	4
2	abierto2	Sunday, April 29, 2012 3:35:23 PM EDT	Agrupamiento de Tarjetas Abierto	0
3	abierto4	Sunday, April 29, 2012 3:42:30 PM EDT	Agrupamiento de Tarjetas Abierto	0
4	abierto3	Sunday, April 29, 2012 3:40:46 PM EDT	Agrupamiento de Tarjetas Abierto	0
5	abierto5	Sunday, April 29, 2012 3:41:24 PM EDT	Agrupamiento de Tarjetas Cerrado	2
6	abierto6	Sunday, April 29, 2012 3:43:16 PM EDT	Agrupamiento de Tarjetas Abierto	0
7	abierto8	Sunday, April 29, 2012 3:46:43 PM EDT	Agrupamiento de Tarjetas Abierto	0
8	bierto7	Sunday, April 29, 2012 3:44:59 PM EDT	Agrupamiento de Tarjetas Abierto	0
9	cerrado1	Sunday, April 29, 2012 3:02:46 PM EDT	Agrupamiento de Tarjetas Cerrado	0
10	cerrado3	Sunday, April 29, 2012 3:26:54 PM EDT	Agrupamiento de Tarjetas Cerrado	0
11	cerrado4	Sunday, April 29, 2012 3:28:43 PM EDT	Agrupamiento de Tarjetas Cerrado	0
12	cerrado5	Sunday, April 29, 2012 3:29:42 PM EDT	Agrupamiento de Tarjetas Cerrado	0

Anexo 5. Interfaz de gestión de usuario.



No.	Usuario	Nombre	Apellidos	Correo	Administrador
1	mperez	Maria	Perez	mpereasd@uci.cu	si
2	rcoral	Raquel	Coral	rcoral@uci.cu	no

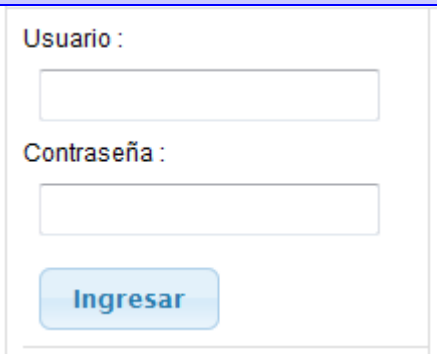
Anexo 6. Especificación de requisitos para listar ejercicios de agrupamiento disponibles

Nº	***Nombre***	***Pre-condiciones***	***Complejidad***	***Prioridad para cliente***
RF1	Listar ejercicios de agrupamiento disponibles.	El usuario debe encontrarse en la página de inicio	Media	Alta
	Descripción			
	1- El sistema muestra las listas de los ejercicios disponibles que existen en sistema. 2- El listado se muestra en dos paneles diferentes, uno para los ejercicios de agrupamiento de tarjetas y otro para los ejercicios de análisis de secuencia. 3- Los ejercicios públicos se muestran a través de una pestaña denominada Públicos y los protegidos se muestran al seleccionar la pestaña Protegidos 4- De cada ejercicio se muestra: a. Número de ejercicio. • Nombre del ejercicio. • Usuario del arquitecto lo creo. • En caso de ser un ejercicio de agrupamiento de tarjetas se pone el tipo. • Fecha en que fue creado el ejercicio. 5- Si el usuario elige ejecutar ejercicio ir a **RF8**			

	Prototipo		
			
	Campos	***Tipos de Datos***	***Reglas o Restricciones***
	Observaciones		

Anexo 7. Especificación de requisitos para autenticarse en el sistema

Nº	***Nombre***	***Pre-condiciones***	***Complejidad***	***Prioridad para cliente***
RF2	Autenticarse en el sistema	El usuario debe estar registrado en el sistema	Media	Alta
	Descripción			
	1- El sistema formulario de autenticación con los campos: • Usuario. • Contraseña. 2- Al presionar el botón ingresar el sistema analiza los datos enviados: • En caso de que los datos estén correctos si es administrador se muestra la página gestionar usuario ir a **RF2,** en caso de ser arquitecto se muestra la página gestionar ejercicio ira a **RF5**.			

	• Si los datos incorrectos se muestra un mensaje indicando que los datos son incorrectos.		
	Prototipo		
			
	Campos	***Tipos de Datos***	***Reglas o Restricciones***
	Usuario	Input(texto)	Cadena de caracteres con letras solamente
	Contraseña	Input(pastor)	Cualquier secuencia de caracteres
	Observaciones		

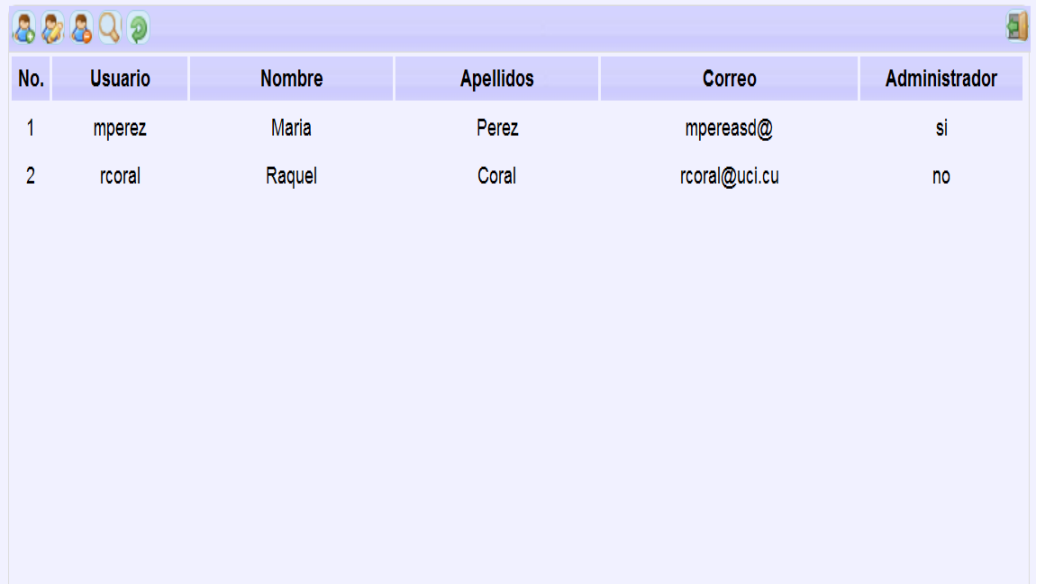
Anexo 8. Especificación de requisitos para gestionar usuario.

Nº	***Nombre***	***Pre-condiciones***	***Complejidad***	***Prioridad para cliente***
RF2	Gestionar Usuario	El usuario debe estar registrado en el sistema como Administrador	Media	Alta
	Descripción			
	• El sistema muestra una lista con todos los usuarios registrados en el sistema. • De cada usuario se muestra: a. Usuario b. Nombre. c. Apellidos. d. Correo. e. Administrador. • Si el usuario selecciona la opción “Adicionar” ir a **RF2.2** • Si el usuario selecciona la opción “Modificar”:			

- Si no hay un usuario seleccionado muestra un mensaje indicando que se debe seleccionar un usuario.
- Si existe un usuario seleccionado ir a **RF2.3**

- Si el usuario selecciona la opción “Eliminar” usuario:
  a. Si no hay un usuario seleccionado muestra un mensaje indicando que debe haber un usuario seleccionado.
  b. Si existe un usuario seleccionado lo el sistema lo elimina.
- Si el usuario selecciona la opción “Salir” se cierra la sesión y se muestra la página principal

Prototipo



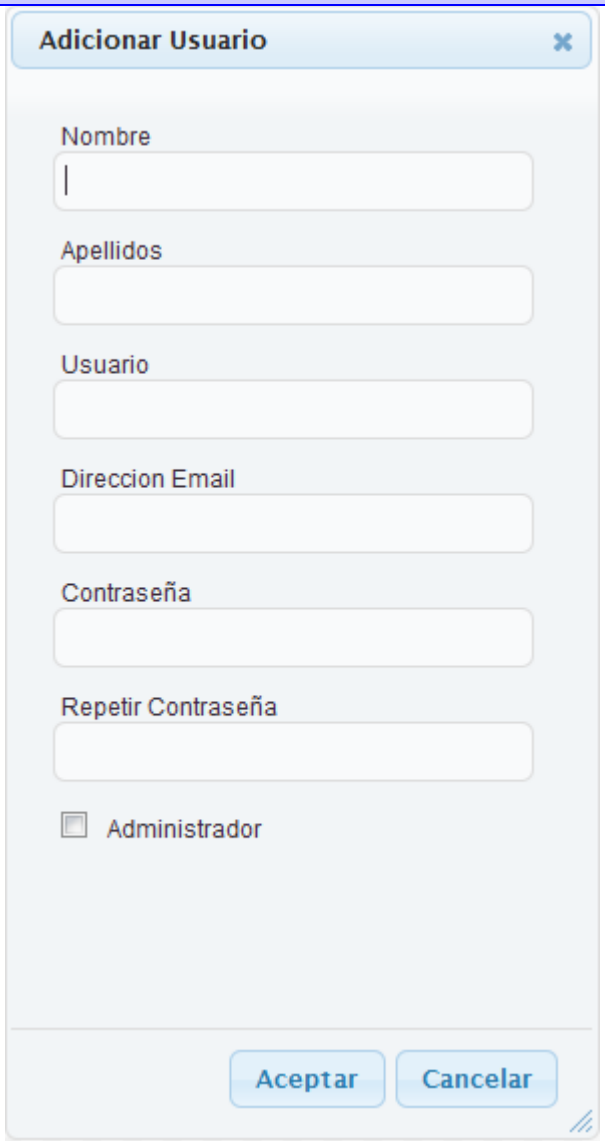
Campos	***Tipos de Datos***	***Reglas o Restricciones***
Observaciones		

Anexo 9. Especificación de requisitos para adicionar usuario.

Nº	***Nombre***	***Pre-condiciones***	***Complejidad***	***Prioridad para cliente***
RF2.1	Adicionar Usuario	El usuario debe estar registrado en el sistema como Administrador	Media	Alta
	Descripción			
	• El sistema muestra un formulario donde se introducirán los datos del nuevo usuario.			

- Los campos que muestra el formulario:
  a. Nombre
  f. Apellidos.
  g. Usuario.
  h. Correo.
  i. Contraseña.
  j. Repetir Contraseña.
  k. Administrador.
- El usuario selecciona la opción “Aceptar” los datos son enviados al sistema:
  a. Si los datos son correctos se registra el nuevo usuario en el sistema.
  b. Si los datos son incorrectos se muestra un mensaje especificando que existen datos incorrectos.

Prototipo

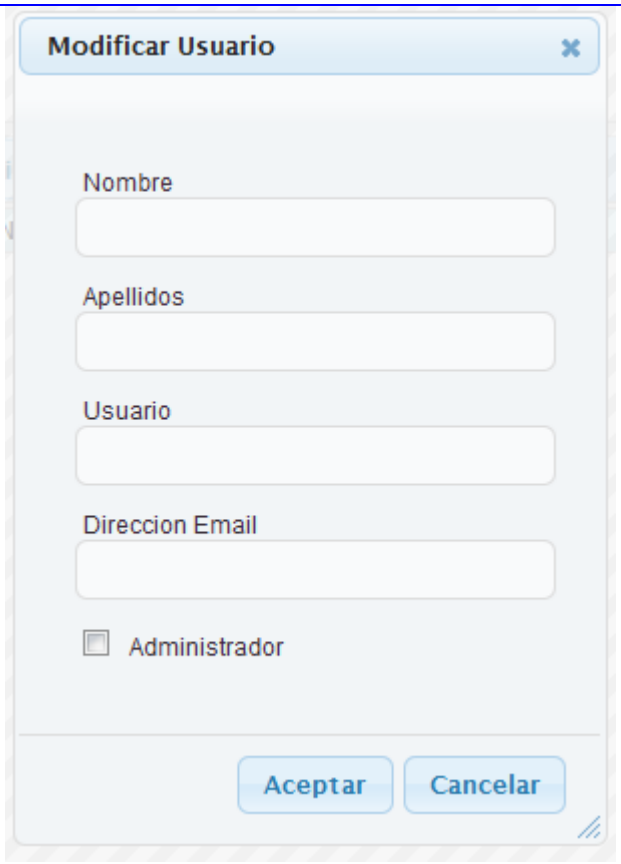


Campos	***Tipos de Datos***	***Reglas o Restricciones***

	Nombre	Input(text)	Solo letras
	Apellidos	Input(text)	Solo letras
	Usuario	Input(text)	Solo letras
	Dirección Email	Input(email)	Dirección de correo
	Contraseña	Input(password)	Cualquier secuencia de caracteres
	Repetir Contraseña	Input(password)	Cualquier secuencia de caracteres
	Administrador	Checkbox	
	Observaciones		

Anexo 10. Especificación de requisitos para modificar usuario.

Nº	***Nombre***	***Pre-condiciones***	***Complejidad***	***Prioridad para cliente***
RF2.2	Modificar Usuario	El usuario debe estar registrado en el sistema como Administrador	Media	Alta
	Descripción			
	1- El sistema muestra un formulario donde se introducirán los datos a modificar del usuario seleccionado. 2- Los campos que muestra el formulario: a. Nombre b. Apellidos. c. Usuario. d. Correo. e. Administrador. 3- El usuario selecciona la opción “Aceptar” los datos son enviados al sistema: a. Si los datos son correctos se modifican los datos del usuario seleccionado. b. Si los datos son incorrectos se muestra un mensaje especificando que existen datos incorrectos.			
	Prototipo			



Campos	***Tipos de Datos***	***Reglas o Restricciones***
Nombre	Input(text)	Solo letras
Apellidos	Input(text)	Solo letras
Usuario	Input(text)	Solo letras
Correo	Input(email)	Dirección de correo
Administrador	checkbox	
Observaciones		

Anexo 11.Especificacion de requisitos para gestionar ejercicio.

Nº	***Nombre***	***Pre-condiciones***	***Complejidad***	***Prioridad para cliente***
RF3	Gestionar Ejercicio	El usuario debe estar registrado en el sistema como arquitecto	Media	Alta

Descripción

1 El sistema muestra una lista con todos los ejercicios que el arquitecto ha importado.
2 De cada ejercicio se muestra:
   a. Nombre
   b. Fecha.
   c. Tipo.
   d. Cantidad de Soluciones.
3 Si el usuario selecciona la opción “Importar” ir a **RF3.1**
4 Si el usuario selecciona la opción “Exportar”:
   - Si no hay un usuario seleccionado muestra un mensaje indicando que se debe seleccionar un usuario.
   - Si existe un usuario seleccionado ir a **RF3.2**
5 Si el usuario selecciona la opción “Eliminar” usuario:
   c. Si no hay un ejercicio seleccionado muestra un mensaje indicando que debe haber un usuario seleccionado.
   d. Si existe un ejercicio seleccionado lo el sistema lo elimina.
   c. Si el usuario selecciona la opción “Salir” se cierra la sesión y se muestra la página principal

Prototipo

No.	Nombre	Fecha	Tipo	Cantidad de Soluciones
1	abierto1	Sunday, April 29, 2012 3:34:36 PM EDT	Agrupamiento de Tarjetas Abierto	4
2	abierto2	Sunday, April 29, 2012 3:35:23 PM EDT	Agrupamiento de Tarjetas Abierto	0
3	abierto4	Sunday, April 29, 2012 3:42:30 PM EDT	Agrupamiento de Tarjetas Abierto	0
4	abierto3	Sunday, April 29, 2012 3:40:46 PM EDT	Agrupamiento de Tarjetas Abierto	0
5	abierto5	Sunday, April 29, 2012 3:41:24 PM EDT	Agrupamiento de Tarjetas Cerrado	2
6	abierto6	Sunday, April 29, 2012 3:43:16 PM EDT	Agrupamiento de Tarjetas Abierto	0
7	abierto8	Sunday, April 29, 2012 3:46:43 PM EDT	Agrupamiento de Tarjetas Abierto	0
8	bierto7	Sunday, April 29, 2012 3:44:59 PM EDT	Agrupamiento de Tarjetas Abierto	0
9	cerrado1	Sunday, April 29, 2012 3:02:46 PM EDT	Agrupamiento de Tarjetas Cerrado	0
10	cerrado3	Sunday, April 29, 2012 3:26:54 PM EDT	Agrupamiento de Tarjetas Cerrado	0
11	cerrado4	Sunday, April 29, 2012 3:28:43 PM EDT	Agrupamiento de Tarjetas Cerrado	0
12	cerrado5	Sunday, April 29, 2012 3:29:42 PM EDT	Agrupamiento de Tarjetas Cerrado	0

Campos	***Tipos de Datos***	***Reglas o Restricciones***

Observaciones	

Anexo 12. Especificación de requisitos para importar ejercicio.

<table>
<tr><th>Nº</th><th>Nombre</th><th>Pre-condiciones</th><th>Complejidad</th><th>Prioridad para cliente</th></tr>
<tr><td>RF3.1</td><td>Importar Ejercicio</td><td>El usuario debe estar registrado en el sistema como arquitecto</td><td>Media</td><td>Alta</td></tr>
<tr><td></td><td colspan="4">Descripción</td></tr>
<tr><td></td><td colspan="4">1- El sistema muestra un formulario donde se mostraran los datos a llenar necesarios para importar un fichero que contiene el ejercicio.
2- En el formulario se muestra:
• Ruta
• Protegido.
• Contraseña.
• Repetir Contraseña.
• Forma a mostrar
3- Si el usuario selecciona “Proteger” se habilitaran los campos para introducir la contraseña del ejercicio.
4- La opción de “Forma a mostrar ” puede ser “En parilla”, “Aleatorio” o “Apiladas”
5- Si el usuario selecciona la opción “Subir” se envían los datos al servidor:
a. En caso ser validos se importan los datos del fichero y se añade un nuevo ejercicio.
b. Si existe algún tipo de dato incorrecto se muestra un mensaje indicando que hay datos incorrectos</td></tr>
<tr><td></td><td colspan="4">Prototipo</td></tr>
</table>

Campos	***Tipos de Datos***	***Reglas o Restricciones***
Ruta	Input(file)	
Proteger	checkbox	
Contraseña	Input(password)	Cualquier cadena de caracteres
Repetir Contraseña	Input(password)	Cualquier cadena de caracteres
Forma a mostrar	radiobutton	Se muestran tres radiobutton , uno indicando la forma de Parilla, otro la de Aleatorio y otro la de Apiladas
Observaciones		

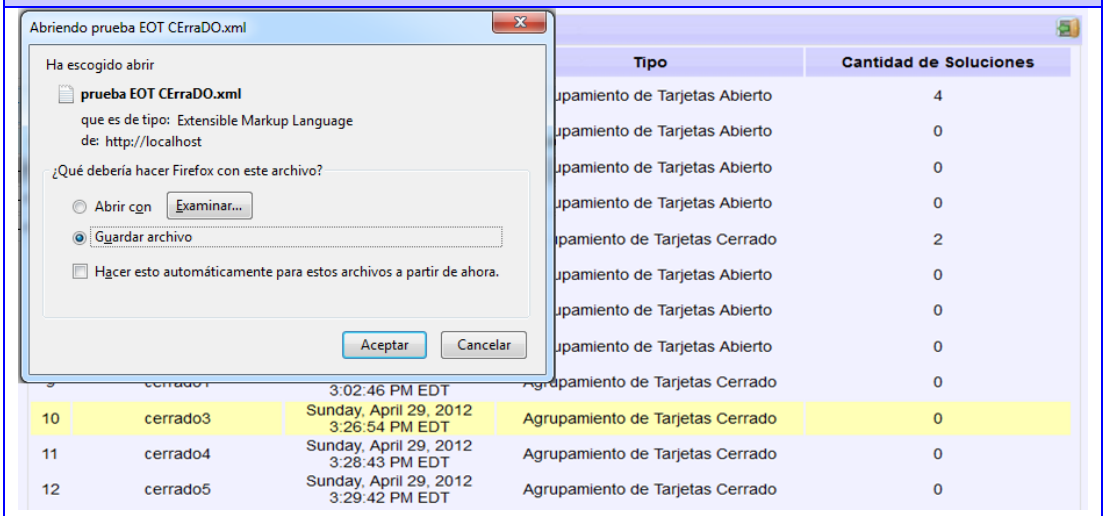
Anexo 13. Especificación de requisito exportar ejercicio.

Nº	Nombre	Pre-condiciones	Complejidad	Prioridad para cliente
RF19	Exportar Ejercicio	El usuario debe estar registrado en el sistema como arquitecto	Media	Alta

Descripción

1- Si no existe un ejercicio seleccionado muestra un mensaje indicando que debe seleccionar el ejercicio a exportar.
2- Si existe un ejercicio seleccionado lo exporta en un fichero formato XML .

Prototipo



Campos	Tipos de Datos	Reglas o Restricciones

Observaciones	