UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 3

Módulo Escalafón de integralidad para el Sistema dataFEU

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor:

Dachel Medero Oliva

Tutores:

Ing. Víctor Gabriel González Cardoso

Ing. Olga Yarisbel Rojas Grass

Ciudad de La Habana, junio de 2015
"Año 57 de la Revolución"

"Seamos realistas y hagamos lo imposible".

AGRADECIMIENTOS

Quisiera agradecerles a todos los amigos que de una forma u otra han hecho posible que este día llegara, a mi nueva familia de hermanos con los cuales he compartido desde segundo año de la carrera, a Yuly y Abelito, el negro y la negra como cariñosamente les digo, a calvo José, a mi compadre el Yasi man que tantos viajes ha dado conmigo de Madruga a la UCI, a Don Dito que me ha enseñado a jugar futbol, a Roly, a Choy o Jackie Chan como a veces le digo, que sin duda es una de las mejores personas que he conocido, a Eli la guardiana del asiático, a Rayko ese es como un hijo todo lo que sabe yo se lo he enseñado, aunque no sabe mucho, a Leo un socio al cual se le puede pedir cualquier favor, al Luiso que aunque siempre quiera darme golpe él sabe que me tiene que respetar, a Freddy que tanto tiempo estuvo compartiendo el apartamento conmigo. Agradecerle a Carlos Llama el socio que entró conmigo por primera vez a esta universidad y a mis amigos de gym.

Por último quiero agradecer a las personas a las cuales les dedico este trabajo de diploma, sin ellos nada de esto sería posible. A mis abuelas queridas, mis tías y tíos, a todos mis primos, a mis vecinos que siempre me están preguntando por cómo me va la escuela y a la familia de mi novia. En especial agradezco a mi novia Yady, “flaca te amo”, al mejor hermano del mundo y el único que tengo, chama ahora si podemos jugar futbol, pero primero termina el verde. A mi padre, que él sabe que lo quiero muchísimo y a la persona más importante de mi vida a la cual le deberían de entregar este título, este logro es más de ella que mío, a mi madre, Neli te adoro con todo mi corazón.

MI GENTE SE NOS ACABÓ EL CAFÉ.

DECLARACIÓN DE AUTORÍA

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 ___.

_____

Dachel Medero Oliva

_____ _____

Ing. Víctor Gabriel González Cardoso Ing. Olga Yarisbel Rojas Grass

RESUMEN

La Universidad de las Ciencias Informáticas, se enfatiza en el desarrollo y evaluación del concepto de integralidad como factor fundamental en el proceso de caracterización estudiantil. Este proceso para los estudiantes de quinto año culmina con la elaboración del escalafón de integralidad, el mismo se realiza de forma manual lo que provoca que su culminación se extienda en el tiempo.

Por tal motivo se decide llevar a cabo el presente trabajo de diploma que tiene como objetivo desarrollar un módulo que gestione la confección del escalafón estudiantil universitario para el Sistema dataFEU. Para la implementación de la propuesta solución se utilizan las herramientas y tecnologías definidas por el Sistema dataFEU con el fin de facilitar la integración. Por tal motivo se emplean el marco de trabajo Symfony 1.4.8 y Ext JS 3.3 empleando MySQL como servidor de base de datos, ajustándose a la soberanía tecnológica por la cual aboga Cuba. Además como metodología que guíe el desarrollo de la propuesta de solución se emplea Proceso Unificado Ágil. Se definen los principales conceptos, requisitos funcionales y no funcionales identificados a partir de las necesidades del cliente. Además, se ejecutan las pruebas para el aseguramiento de la calidad del producto.

Esta investigación arrojó como resultado una solución informática que realiza la confección del escalafón estudiantil universitario reduciendo el tiempo de culminación del proceso.

Palabras clave: caracterización estudiantil, escalafón estudiantil, integralidad, módulo.

Índice

INTRODUCCIÓN

En la actualidad el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha posibilitado en gran medida el avance de disímiles ramas de la ciencia, la tecnología y la sociedad. El aumento progresivo del empleo de las TIC se ha convertido en un factor a tener en cuenta en el desarrollo de las empresas y organizaciones, para su inclusión en un mundo tan globalizado y competitivo.

La Universidad de las Ciencias Informáticas (UCI), concebida como centro educativo de nuevo tipo por el Comandante Fidel Castro Ruz, tiene como objetivo la formación de profesionales de un amplio perfil, y para lograrlo utiliza las TIC como apoyo en el perfeccionamiento y gestión de sus actividades. En la universidad existen varias organizaciones políticas y estudiantiles y entre ellas se encuentra la Federación Estudiantil Universitaria (FEU) que es una organización de masas que agrupa a todos los estudiantes de la enseñanza superior. La FEU de la UCI realiza un gran número de procesos los cuales son difíciles de gestionar de forma manual, por lo que surge el Sistema dataFEU como sistema de gestión para la informatización de los principales procesos llevados a cabo por la FEU en la universidad. El Sistema dataFEU tiene como objetivo reducir el tiempo de procesamiento de la información, estandarizar la documentación y elevar la persistencia de los datos del proceso de evaluación estudiantil (1).

El proceso de evaluación estudiantil, es denominado proceso de integralidad para los estudiantes de primero a cuarto año y proceso de caracterización para los estudiantes de quinto año. Con el mismo se persigue fomentar el desarrollo de las capacidades, actitudes, valores y potencialidades del futuro profesional. Esto se logra durante todo el período de formación en el pregrado a partir del control y evaluación de los estudiantes en el ejercicio de sus actividades. El proceso de caracterización para los estudiantes de quinto año culmina con la creación de un escalafón estudiantil, el cual es construido desde el nivel de brigada siguiendo los indicadores aprobados por el Consejo Universitario a propuesta de la FEU (2).

El proceso de confección del escalafón estudiantil universitario (EEU) se realiza de forma manual haciéndolo susceptible a errores. Las pruebas de las actividades realizadas por los estudiantes se recogen mediante una autoevaluación que deben entregar. Después de terminada la recopilación de la información se evalúa al estudiante atendiendo a cada indicador y se utiliza la herramienta Microsoft Excel para calcular el índice de integralidad. Las evidencias que sirven de base para la confección del EEU están almacenadas en el Sistema dataFEU, pero no se utilizan directamente en este proceso. La confección del EEU culmina luego de procesar un gran volumen de información por lo que se torna lento el proceso.

Teniendo en cuenta la problemática anteriormente expuesta se plantea el siguiente **problema de la investigación**: ¿Cómo reducir el tiempo de confección del escalafón estudiantil universitario de la FEU en la UCI? Definiéndose como **objeto de estudio:** Proceso de gestión de escalafones

estudiantiles, enmarcado en el **campo de acción:** Proceso de confección del escalafón estudiantil universitario para el Sistema dataFEU de la UCI.

Para dar respuesta al problema planteado se define como **objetivo general:** desarrollar un módulo que gestione la confección del escalafón estudiantil universitario para el Sistema dataFEU. Los **objetivos específicos** que se definen para dar cumplimiento al objetivo general son:

- Definir el marco teórico de la investigación mediante el estudio y el análisis de los principales referentes teóricos para el desarrollo de la solución.
- Realizar el análisis y diseño del módulo Escalafón de integralidad para el Sistema dataFEU.
- Implementar el módulo Escalafón de integralidad para el Sistema dataFEU que permita la confección del EEU.
- Validar la propuesta de solución mediante la aplicación de técnicas, métricas y pruebas.

La validación de la investigación está sustentada en la siguiente **idea a defender:** El desarrollo del módulo Escalafón de integralidad para el Sistema dataFEU reducirá el tiempo de confección del escalafón estudiantil universitario en la UCI.

En la presente investigación científica se emplearon los siguientes métodos teóricos y empíricos:

Métodos teóricos:

- **Analítico – sintético:** se emplea para analizar la bibliografía relacionada con el tema, permitiendo la extracción de los elementos que se relacionan con el objeto de estudio. La información adquirida a través de los referentes teóricos permitieron realizar una síntesis concreta de los elementos a tener en cuenta en el desarrollo del sistema.
- **Modelación:** permite la representación de manera simplificada de la realidad del negocio, a través de modelos o diagramas que posibilitan crear abstracciones y representar los productos de trabajo de las diferentes disciplinas.

Métodos empíricos:

- **Entrevista:** posibilita la obtención de la información referente al objeto de estudio; además permite comprender y precisar la situación problemática.

La estructura del trabajo de diploma, siguiendo un orden lógico ha sido dividida en tres capítulos:

- El **primer capítulo: "Fundamentación teórica"**, se realiza el estudio del estado del arte de los sistemas informáticos relacionados con el campo de acción. Se selecciona la

metodología, las herramientas y las tecnologías a utilizar para desarrollar la propuesta de solución.

- El **segundo capítulo: "Descripción de la solución propuesta"**, comprende la descripción de los procesos de negocios, la definición de los requisitos funcionales y no funcionales, el diseño y arquitectura de la aplicación a desarrollar. Se describen los patrones de diseño y el estilo arquitectónico empleado.
- En el **tercer capítulo: "Implementación y validación del sistema"**, se analizan los resultados de la implementación del módulo propuesto, se presentan los estándares de codificación a seguir para la implementación del módulo. Además se realizan las pruebas para validar las funcionalidades del módulo y se muestran los resultados obtenidos de cada prueba.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

El presente capítulo expone el marco teórico y conceptual asociado a la problemática a resolver para lograr un mayor entendimiento de la misma. Además se detallan las tecnologías y herramientas necesarias para lograr la implementación del módulo Escalafón de integralidad, así como la metodología a utilizar en el proceso de desarrollo. Se realiza un estudio de los sistemas existentes relacionados con la elaboración de escalafones estudiantiles universitarios, con el propósito de identificar tendencias y dar cumplimiento al objetivo trazado.

1.1 Análisis de sistemas homólogos

El estudio de diferentes sistemas relacionados con el objeto de estudio permite identificar las funcionalidades existentes durante el proceso de confección de escalafones estudiantiles. Se definieron una serie de parámetros a estudiar para establecer una comparación entre los sistemas, entre los que se tienen: generación del escalafón, el acceso que poseen los estudiantes a la información y si se realiza el análisis integral.

1.1.1 Ámbito Internacional

Sistema de Información Estudiantil (SIE)

El Sistema de Información Estudiantil, es la iniciativa tecnológica más ambiciosa del Departamento de Educación de Puerto Rico. El sistema comprende un amplio repositorio de datos que almacena y maneja toda la información relacionada a los estudiantes y procesos de enseñanza en el Sistema de Educación Pública, comenzó a operar en enero del 2007. Actualmente sirve a más de cuarenta mil usuarios entre ellos maestros, directores de escuelas, personal de apoyo y administrativo. Fue desarrollado en Symfony1 y PHP 5, utilizando como servidor de base de datos MySQL 4.1 y el sistema operativo Linux.

Uno de los servicios del SIE, es el Portal de Padres, mediante el cual los padres pueden acceder a los expedientes de sus hijos por medio de una página electrónica. Entre los servicios que ofrece el nuevo portal de padres se encuentran:

- Información demográfica del estudiante
- Programa de clases
- Registro de puntuaciones
- Detalles de las evaluaciones académicas y sus calificaciones
- Informes de asistencia

El servicio Detalles de las evaluaciones académicas y sus calificaciones permite obtener el listado de estudiantes por curso escolar, ordenados por el promedio de calificaciones obtenidas hasta el momento (3). A pesar de que SIE brinda una serie de funcionalidades, entre ellas la obtención de un escalafón estudiantil, está lejos de ser útil en el entorno de la UCI ya que los estudiantes de la

universidad no solo son evaluados por sus calificaciones, sino que se tienen en cuenta un conjunto de parámetros por indicadores.

1.1.2 Ámbito Nacional

SIGENUS

El Sistema de Gestión de la Nueva Universidad (SIGENU) es un software desarrollado en el Instituto Superior Politécnico “José Antonio Echeverría” (ISPJAE) en el 2007 con el fin de ser una herramienta que permita la gestión de toda la información académica vinculada con la educación superior en Cuba. En correspondencia con su carácter nacional y la gran diversidad de sistemas de enseñanza superior, este sistema ha sido concebido de manera tal que sea capaz de brindar gran seguridad e integridad de la información, y a la vez, tan flexible que permita ser adaptado a todos los centros de educación superior del país con sus diversas particularidades y distintas maneras de realizar determinados procedimientos. El sistema permite obtener un conjunto de reportes utilizados en el mundo de la educación superior. Consta con los siguientes módulos: Codificadores, Matrícula, Control de estudiantes, Plan de estudio, Evaluaciones y Reportes.

SIGENUS fue desarrollado en el lenguaje de programación Java, está diseñado a nivel de la secretaría del centro de estudio, esto quiere decir que otros usuarios que forman parte del ámbito académico como los profesores y los estudiantes no pueden interactuar con el sistema. Aunque el sistema logra un control de las evaluaciones de los estudiantes, no genera un escalafón estudiantil. Su interfaz principal es poco amigable y necesita ser instalada en cada computadora que se va a utilizar por ser esta una aplicación de escritorio, lo que ocasiona pérdidas de tiempo y limitaciones en su uso.

Módulo gestión de ubicación laboral en la Universidad de las Ciencias Informáticas es un trabajo de diploma desarrollado por Saeret Llerena Álvarez en el año 2013. El mismo plantea como objetivo “Desarrollar una solución informática con tecnologías libres que permitiese la gestión de ubicación laboral de los egresados de la Universidad de las Ciencias Informáticas en el Sistema de Gestión Universitaria (SGU) “ (4).

Dicho sistema fue implementado con la aplicación de los lenguajes de programación PHP 5, CSS 2.1, HTML 4, JavaScript 1.2, librería JQuery y como sistema gestor de bases de datos se empleó PostgreSQL.

Dentro de las actividades que forman parte del proceso de ubicación laboral de los egresados de la Universidad se encuentra: ordenar el listado de egresados por municipio, promedio e integralidad. Después de la realización del escalafón se otorgan las ubicaciones laborales. A pesar de elaborar el EEU el mismo no se genera de forma automática, ya que se deben introducir las evaluaciones alcanzadas por cada estudiante en los distintos indicadores. Además solo accede al escalafón estudiantil el personal que trabaja directamente en el sistema, sin incluir al estudiantado universitario.

1.2 Valoración de los sistemas estudiados

El estudio de los sistemas informáticos mencionados con anterioridad, permitió establecer una comparación siguiendo una serie de parámetros, como son: interfaz, acceso estudiantil, generación de escalafón estudiantil, tipo de sistema, lenguaje de programación empleado, análisis de integralidad y multiplataforma.

Tabla 1. Comparación de los sistemas estudiados

Parámetros	Sistemas estudiados		
	SIE	SIGENUS	Módulo gestión de ubicación laboral en la UCI
Interfaz	Amigable	Poco amigable	Amigable
Generación de escalafón	Si	No	Si
Acceso estudiantil	Si	No	No
Tipo de sistema	Web	Desktop	Web
Lenguaje de programación	PHP, HTML, CSS, JavaScript	Java	PHP, HTML, CSS, JavaScript
Análisis integral	No	No	Si
Multiplataforma	Si	Si	Si

El análisis de cada uno de los sistemas permitió concluir que:

- Todos los sistemas estudiados excepto SIGENUS poseen una interfaz amigable las cuales son de fácil adaptación y entendibles para el usuario.
- En el sistema SIGENUS las funcionalidades están centralizadas a nivel de secretaría, esto quiere decir que otros usuarios que forman parte del ámbito académico como los estudiantes no pueden interactuar con él y no realiza la generación del escalafón estudiantil.
- La mayoría de los sistemas fueron desarrollados para plataforma web utilizando como lenguaje de programación PHP, HTML, CSS y JavaScript exceptuando el sistema SIGENUS que es una aplicación de escritorio desarrollada en el lenguaje Java.
- Solamente el Módulo gestión de ubicación laboral en la UCI realiza el escalafón estudiantil atendiendo a varios indicadores generando un análisis integral de los estudiantes.

A partir de la comparación establecida anteriormente se puede decir que ninguno de los sistemas estudiados está acorde en su totalidad a lo que se quiere diseñar según las necesidades expuestas por parte del cliente. El estudio permitió observar y analizar las distintas formas en que se gestionan las calificaciones académicas y actividades extra-curriculares de los estudiantes con el fin de conformar el escalafón estudiantil que facilite la toma de decisiones. Por lo tanto se define

la necesidad de desarrollar un módulo para el Sistema dataFEU que permita la confección del EEU en la UCI.

Para lograr minimizar riesgos e incrementar las posibilidades de éxito en el desarrollo de la propuesta de solución se debe utilizar una metodología de desarrollo de software, tema que se abordará en el epígrafe siguiente.

1.3 Metodología de desarrollo de software

Una metodología de desarrollo es el conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. La metodología indica cómo hay que obtener los distintos productos parciales y finales en el desarrollo del software que sigue uno o varios modelos del ciclo de vida. No existe una metodología de software universal, las características de cada proyecto y equipo de desarrollo exigen que el proceso sea configurable (5).

Metodología de desarrollo para la Actividad productiva de la UCI

La UCI se encuentra actualmente inmersa en un proceso de mejora, definiendo como metodología de desarrollo de software para sus proyectos productivos una variación del Proceso Unificado Ágil (AUP, por sus siglas en inglés) (6). La metodología AUP con sus nuevas variaciones se adapta al ciclo de vida definido para la actividad productiva de la UCI delimitando tres fases: inicio, ejecución y cierre. En la presente investigación se transitará por la fase de ejecución en la cual se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente (7).

En el proceso de desarrollo de la propuesta de solución se transitará por las siguientes disciplinas definidas por AUP en su variación para la UCI:

- **Modelado de negocio:** El modelado del negocio está destinado a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito.
- **Requisitos:** El esfuerzo principal en la disciplina requisitos es desarrollar el modelo del sistema que se va a construir. Comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.
- **Análisis y diseño:** Si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema. Además se modela el

sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales.

- **Implementación:** En la implementación, a partir de los resultados del análisis y diseño se construye el sistema.
- **Pruebas internas:** Se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible, componentes de prueba ejecutables.
- **Pruebas de aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

Una vez definida la metodología a utilizar en el proceso de desarrollo de software se establecerán las tecnologías y herramientas para la implementación de la propuesta de solución.

1.3.1 Tecnologías

Para la implementación del módulo Escalafón de integralidad es necesario seleccionar las herramientas y tecnologías a utilizar. En este caso se emplearán los lenguajes y tecnologías que se utilizaron en la implementación del Sistema dataFEU para garantizar la integración del módulo al sistema. Por ello se define como lenguajes a utilizar PHP 5.3, HTML 5, CCS 3, JavaScript 1.4, se emplea además la librería Ext JS 3.3.0 y AJAX. Como marco de trabajo que agilice el proceso de construcción del software se utilizará Symfony 1.4.8 el cual se une con los amplios beneficios de Doctrine 1.2, un Mapeador de Objetos Relacionales (ORM, por sus siglas en inglés).

Servidor web Apache 2.2

El servidor web Apache contiene la lógica para la interpretación de las peticiones de los usuarios a través de los protocolos especificados, que hace posible la interacción dinámica entre el cliente y sus acciones sobre el sistema en que esté trabajando. Apache se basa originalmente en codificación e ideas basadas en el servidor HTTP. Actualmente por su flexibilidad, rapidez, eficiencia, su constante actualización y adaptación a los nuevos protocolos es uno de los mejores servidores web utilizados en Internet, según Netcraft Survey, empresa dedicada a la realización de encuestas a nivel global y estudios sobre el tráfico en Internet. Se utiliza para proveer un alto grado de calidad y fortaleza para las implementaciones que utilizan el protocolo HTTP, ha sido desde su salida al mercado, uno de los servidores de mayor popularidad, considerado el proyecto punta del movimiento de software libre (8).

PHP 5.3

PHP es un acrónimo recursivo que significa Hypertext Pre-processor, es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es

un lenguaje de propósito general ampliamente usado y que está diseñado especialmente para el desarrollo web y puede ser embebido dentro de código HTML. Se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. El código fuente escrito en PHP es invisible al navegador y al cliente, esto hace que la programación sea segura y confiable. A esto se añaden valores como el hecho de ser un proyecto de código abierto, gratuito y multiplataforma.

Entre sus principales ventajas se puede mencionar:

- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con PostgreSQL.
- Posee una amplia documentación en su página oficial, entre la cual sobresale que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Permite las técnicas de Programación Orientada a Objeto (POO).
- Posee una biblioteca nativa de funciones sumamente amplia (9).

Symfony 1.4.8

"Es un marco de trabajo PHP de tipo full-stack construido con varios componentes independientes creados por el proyecto Symfony" (10). El mismo ha ganado popularidad por su facilidad de trabajo, potencia, solidez y rendimiento, el cual es superior notablemente a su anterior versión. Aunque utiliza la arquitectura Modelo Vista Controlador (MVC), tiene su propia forma de trabajo, con algunas variantes como: la capa de abstracción de base de datos, el controlador frontal y las acciones. Sigue las buenas prácticas y patrones de diseño para la web.

Este marco de trabajo se acopla con amplios beneficios a **Doctrine 1.2**, un Mapeador de Objetos Relacionales que crea una capa de acceso a datos separando la entrada directa a la base de datos de la lógica del sistema. Es una librería muy completa y configurable. Permite la generación automática del modelo creando las clases que representan al modelo de negocio de la aplicación. Posibilita trabajar con YAML, acrónimo recursivo que significa YAML Ain't Another Markup Language (en castellano: YAML no es otro lenguaje de marcado), que es un formato de serialización de datos legible muy usado. Posee un lenguaje propio: Doctrine Query Language (DQL), para manejar las interacciones con la base de datos (11).

JavaScript 1.8

Javascript es un lenguaje de scripts desarrollado por Netscape para incrementar las funcionalidades del lenguaje HTML. Se utiliza embebido en el código HTML y XHTML, entre las etiquetas <script> y </script>. Dentro de sus características más importantes se pueden encontrar:

- Es un lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias Javascript contenidas en una página HTML y ejecutarlas adecuadamente.

- Es un lenguaje orientado a eventos. Cuando un usuario presiona el cursor sobre un enlace o mueve el puntero sobre una imagen se produce un evento. Mediante este lenguaje se puede desarrollar Scripts que ejecuten acciones en respuesta a estos eventos.
- Es un lenguaje orientado a objetos. El modelo de objetos de Javascript está reducido y simplificado, pero incluye los elementos necesarios para que los Scripts puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador (12).

Ext JS 3.3.0

Ext JS 3.3.0 logra un equilibrio entre el cliente y el servidor al distribuir la carga de procesamiento, lo que permite que el servidor pueda atender más peticiones al mismo tiempo al tener menor carga. La gran ventaja de ExtJS es que posee un conjunto de componentes ya creados que permite su utilización de forma rápida en los sitios web. A esto se le suma el empleo de tecnologías como AJAX para mitigar la lentitud de renderizado de los componentes de Ext JS, así como de los datos adicionales que se solicitan al servidor cargándose en segundo plano sin interferir con la visualización ni el comportamiento de la página (13).

Lenguaje de Marcas de Hypertexto 5

La versión 5 de HTML (por sus siglas en inglés), es la quinta revisión de este lenguaje base de la World Wide Web, siendo esta la primera vez que HTML y XHTML se han desarrollado en paralelo. Establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Proporciona nuevas funcionalidades a través de una interfaz estandarizada y permite renderizar elementos en tercera dimensión (3D) (14).

CSS 3

CSS hace referencia a un lenguaje de hojas de estilos utilizado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas. Su aplicación más común es dar estilo a páginas web escritas en lenguaje HTML. La versión 3 provee de mejores diseños y estilos que crean un ambiente más elegante en los proyectos. La principal característica es la modularización de sus principales características (15).

UML 2.0

UML es un lenguaje de modelado para especificar o describir métodos o procesos. Se utiliza para definir un sistema, y desarrollar artefactos en el sistema, documentar y construir. Se puede aplicar en el desarrollo de software en gran variedad de formas (16).

Tiene como objetivos fundamentales:

- Visualizar: permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- Especificar: permite expresar o definir cuáles son las características de un sistema antes de su construcción.

- Construir: a partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

BPMN 2.0

La Notación de Modelado de Proceso de Negocio permite modelar, simular y, eventualmente, ejecutar procesos de negocios. Proporciona una notación gráfica para expresar procesos de negocio mediante un diagrama de proceso de Negocio. No es utilizado para modelar aplicaciones, sino procesos que corren dentro de esas aplicaciones. El modelado en BPMN se realiza mediante diagramas muy simples con un conjunto de elementos gráficos, con los que se busca que los usuarios del negocio y los desarrolladores técnicos logren entender fácilmente el flujo de actividades del proceso (17).

1.4 Herramientas

NetBeans 7.4

Es un Entorno de Desarrollo Integrado (IDE) de código abierto escrito completamente en Java, multiplataforma y está registrado bajo Licencia Común de Desarrollo y Distribución (CDDL) y Licencia Pública General de GNU (GPL). La versión 7.4 de esta herramienta brinda servicio de control de versiones. Permite crear aplicaciones web escritas en PHP 5, HTML 5, CSS 3, Javascript, AJAX, Symfony y otros frameworks PHP. Posee un potente editor y debugger integrados (18).

Visual Paradigm 8.0

Es una suite de aplicaciones, herramienta profesional que soporta el ciclo completo de desarrollo de software. Soporta el modelado visual con UML, que ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite generar documentación en html/pdf con los diagramas realizados. Permite la generación de bases de datos, conversión de diagramas entidad-relación a tablas de base de datos, mapeos de objetos y relaciones, ingeniería inversa desde gestores de bases de datos.

Proporciona una plataforma de modelado colaborativo para el trabajo en equipo. Con las características de colaboración en equipo, los miembros del equipo pueden ver y editar el mismo proyecto, o el mismo esquema, incluso de forma simultánea. Todos los cambios se almacenan en el servidor de Visual Paradigm en función de revisión (19).

Se caracteriza por:

- Ser un software libre.
- Presentar una licencia: gratuita y comercial.
- Estar disponible para múltiples plataformas (Windows, Linux).

- Permitir el uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.

MySQL 5.1.37

MySQL 5.1.37 es un Sistema Gestor de Base de Datos (SGBD) cliente/servidor, multihilo y multiusuario. Se compone de un servidor SQL, varios programas clientes y bibliotecas, herramientas administrativas y una gran variedad de interfaces de programación (APIs). Se puede obtener también como una biblioteca multihilo que se enlaza dentro de otras aplicaciones para obtener un producto más pequeño, rápido y fácil de manejar. Dentro de las ventajas se encuentra que dispone de borrados multi-tablas, mejores utilidades de administración y cuenta con un sistema de replicación multihilo en los servidores esclavos. Soporta cinco tipos de tablas y posee recuperación automática ante fallas e integridad referencial (20).

Subversion 1.6.16

Como sistema de control de versiones se selecciona Subversion en su versión 1.6.16. Es un software libre bajo una licencia de tipo Apache/ Distribución de Software Berkeley (BSD). Entre sus ventajas se encuentran que sigue el historial de los archivos y directorios a través de copias y renombrados, modificaciones atómicas, la creación de ramas y etiquetas en operaciones más eficientes enviando sólo las diferencias en ambas direcciones (21).

1.5 Patrones

Un patrón es un modelo posible a seguir para realizar determinada tarea. Los patrones surgen de la experiencia de seres humanos al tratar de lograr ciertos objetivos, estos capturan la experiencia existente y probada para promover buenas prácticas (22).

Según Christopher Alexander “Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma”. Dentro de la rama de la informática los patrones se clasifican según su escala en: patrones de arquitectura y patrones de diseño (23).

1.5.1 Patrones de diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan. Ayudan a evitar que los cambios en el sistema se realicen de una forma concreta, de manera que se afecte lo menos posible (24).

Patrones GoF

Los patrones GoF (Gang-of-Four, pandilla de los cuatro) describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad.

Permiten crear grupos de objetos que ayudan a realizar tareas complejas. Existen tres tipos de estos patrones: de creación, estructurales y de comportamiento que se relacionan a continuación (22):

- **Creacionales:** Patrones creacionales tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
- **Estructurales:** Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.
- **Comportamiento:** Los patrones de comportamiento nos ayudan a definir la comunicación e interacción entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

Patrones GRASP

Los patrones GRASP (General Responsibility Asignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Este grupo de patrones está muy relacionado con los problemas básicos del diseño (22).

- **Patrón experto:** Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele ser útil en el diseño orientado a objetos. El cumplimiento de una responsabilidad requiere a menudo información distribuida en varias clases de objetos. El patrón experto asigna responsabilidades a las clases que tienen la información necesaria para cumplir con la responsabilidad.
- **Patrón creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. Lo que define este patrón es que una instancia de un objeto la tiene que crear el objeto que tiene la información para ello. ¿Qué significa esto?, pues que si un objeto A utiliza específicamente otro B, o si B forma parte de A, o si A almacena o contiene B, o si simplemente A tiene la información necesaria para crear B, entonces A es el perfecto creador de B.
- **Patrón alta cohesión:** Mantiene la complejidad dentro de límites manejables, es decir asigna una responsabilidad de modo que la cohesión siga siendo alta. La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

- **Bajo acoplamiento:** Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Acoplamiento bajo significa que una clase no depende de muchas clases.
- **Patrón controlador:** Es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Asigna las responsabilidades de capturar los eventos del sistema a las clases.

1.6 Conclusiones parciales

El estudio realizado en el presente capítulo de los sistemas homólogos evidenció la necesidad de crear un módulo que permita la generación del EEU para el Sistema dataFEU. Debido a que el módulo Escalafón de integralidad se debe integrar al Sistema dataFEU es necesaria la utilización de los lenguajes y tecnologías utilizados en el desarrollo del Sistema dataFEU, el estudio de estos lenguajes y tecnologías permitió lograr un mayor dominio del ambiente de desarrollo con el fin de obtener un producto con la calidad requerida. Además los productos de trabajo se elaborarán a partir de lo que se define en la metodología AUP la cual servirá de guía en todo el proceso de desarrollo de la propuesta de solución.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

En el presente capítulo se modela la propuesta de solución a partir del proceso de negocio Generación del escalafón de integralidad. Se exponen las características esenciales del módulo Escalafón de integralidad quedando identificadas las funcionalidades que responden a las necesidades del cliente. Se especifican las técnicas utilizadas para la obtención de requisitos del software, la descripción de la arquitectura y los patrones de diseño empleados.

2.1 Modelado del negocio

El Modelado del negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Explica cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Identifica con claridad el espacio conceptual dentro del cual los procesos que se piensan modelar cobran sentido; el dominio puede referirse a un área, a una unidad de negocio o a un conjunto de áreas que atraviesan la organizaciones (25).

2.1.1 Descripción del proceso de negocio

El proceso de Generación del escalafón de integralidad inicia cuando el Consejo de la FEU a nivel de universidad define cada uno de los parámetros por indicadores los cuales se utilizan posteriormente para medir el desempeño de cada estudiante a lo largo de la carrera. En este momento el Consejo universitario es el encargado de revisar y aprobar los parámetros definidos por el Consejo de la FEU en cada uno de los indicadores. Posteriormente y una vez aprobadas la propuesta del Consejo de la FEU de la universidad la Comisión de la brigada solicita a cada estudiante una autoevaluación o también denominada caracterización.

El estudiante es el encargado de redactar la planilla de caracterización definida por la universidad, realizando una autovaloración de cómo ha cumplido cada uno de los indicadores contenidos en la misma. La Comisión de la brigada debe revisar la información contenida en las caracterizaciones, se discute de forma individual con cada estudiante su propuesta de evaluación según el cumplimiento de los parámetros establecidos por la universidad. La Comisión de la brigada aprueba la plantilla de caracterización en caso que no tenga errores, de lo contrario se le entrega al estudiante para su corrección. Posteriormente se elabora el escalafón estudiantil de la brigada que permitirá conformar el escalafón estudiantil de la facultad, para elaborar el escalafón se siguen una serie de pasos (ver anexo1). Al culminar la confección del escalafón de brigada esta información pasa a la Comisión de la facultad encargada de elaborar el escalafón estudiantil a nivel de facultad y posteriormente se elabora el escalafón de la facultad, dicha información pasa a nivel UCI y la Comisión central se encarga de elaborar el EEU.

A continuación se muestra el diagrama de procesos de negocio correspondiente al proceso Generación del escalafón de integralidad.

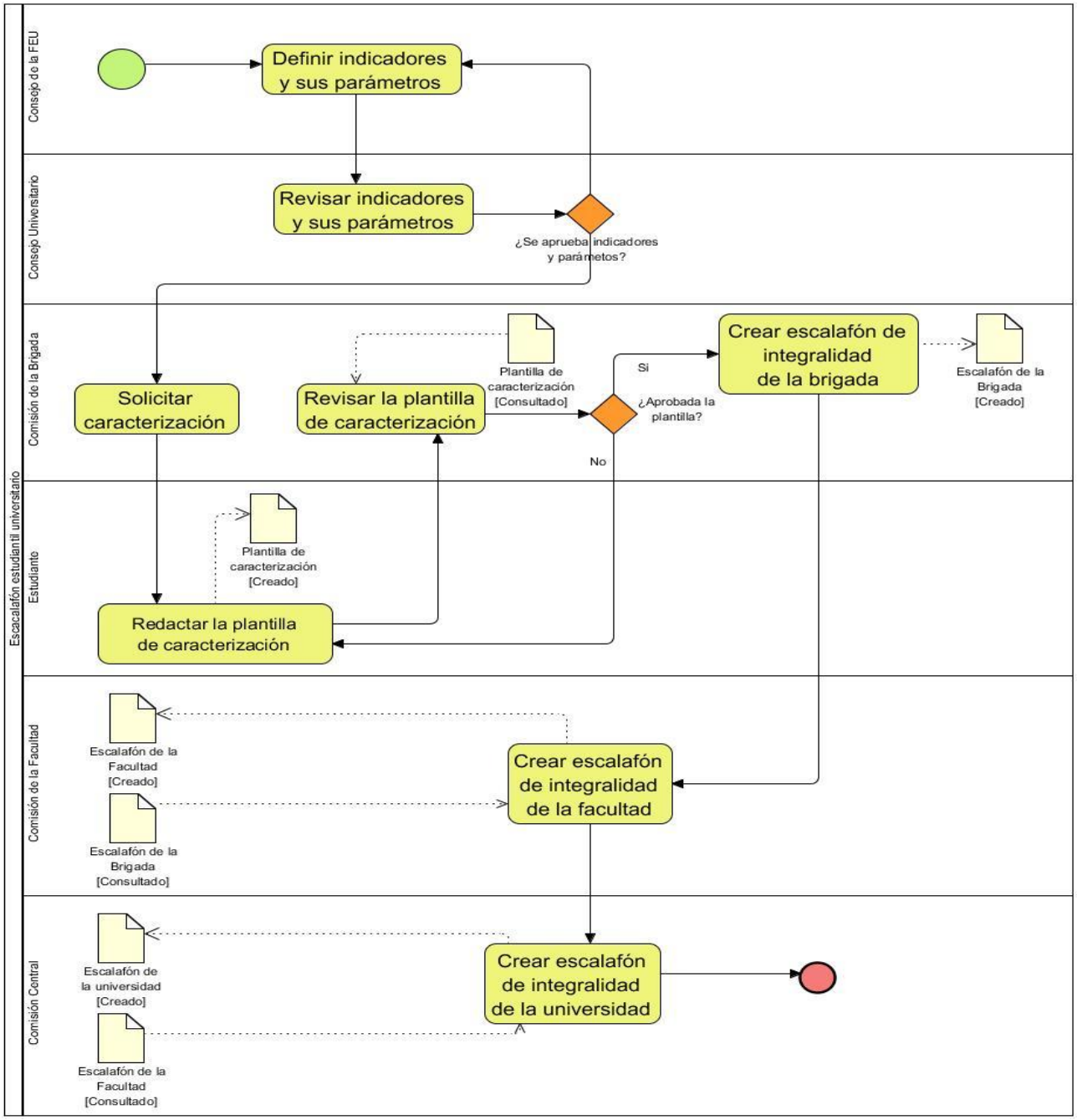


Figura 1. Diagrama del proceso Generación del escalafón de integralidad

A partir de las salidas generadas en la disciplina de Modelado del negocio se desarrollan las actividades de obtención de requisitos las cuales se llevan a cabo en la disciplina de Requisitos.

2.2 Requisitos

Esta disciplina cumple un papel primordial en el proceso de producción de software, debido a que se enfoca en un área determinante para el éxito de un proyecto: la definición de lo que se desea producir y el cumplimiento con lo que realmente se debía producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en

forma consistente, clara y compacta, el comportamiento del sistema; de esta manera, se pretende minimizar los problemas relacionados con su desarrollo (26).

En esta disciplina se mencionan las técnicas de obtención de requisitos, se definen los requisitos funcionales y no funcionales. Además se presenta la descripción de requisitos a través del producto de trabajo Descripción de requisitos presente en el expediente de proyecto.

2.2.1 Técnicas de obtención de requisitos

Para la identificación de requisitos existen varias técnicas que permiten establecer una buena comunicación entre las personas que mantienen relación con el negocio y el equipo de trabajo. Se debe destacar que ninguna de estas técnicas fue suficiente por sí sola, hubo que combinarlas para lograr un proceso de obtención de requisitos de forma satisfactoria. Entre las técnicas utilizadas se encuentran:

- **Entrevistas:** proporcionó información muy valiosa sobre el proceso de confección del EEU. Se aplicaron entrevistas a varios estudiantes pertenecientes a la dirección de la FEU en la universidad los cuales tenían un amplio dominio del proceso de caracterización integral en la UCI (ver anexo 2).
- **Tormentas de ideas (Brainstorming):** se realizaron reuniones en grupo, con la participación del cliente, con el objetivo de generar ideas respecto a las funcionalidades con que contaría el módulo.
- **Estudio de la documentación:** Tomando como base el estudio de varios tipos de documentación como manuales y reglamentos se obtuvo información con respecto a la forma de organización y el flujo del proceso de confección del EEU.

2.2.2 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, la manera en que este debe reaccionar a entradas de datos y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (27). A continuación se muestran los requisitos funcionales (RF):

Tabla 2. Listado de requisitos funcionales del módulo Escalafón de integralidad

Nº	*Nombre*
RF1	Mostrar indicador
RF2	Gestionar parámetros investigación
RF2.1	Mostrar parámetros investigación
RF2.2	Adicionar parámetro investigación
RF2.3	Eliminar parámetro investigación

RF2.4	Modificar parámetro investigación
RF3	Gestionar parámetros cultura
RF3.1	Mostrar parámetros cultura
RF3.2	Adicionar parámetro cultura
RF3.3	Eliminar parámetro cultura
RF3.4	Modificar parámetro cultura
RF4	Gestionar parámetros deporte
RF4.1	Mostrar parámetros deporte
RF4.2	Adicionar parámetro deporte
RF4.3	Eliminar parámetro deporte
RF4.4	Modificar parámetro deporte
RF5	Gestionar parámetros residencia
RF5.1	Mostrar parámetros residencia
RF5.2	Adicionar parámetro residencia
RF5.3	Eliminar parámetro residencia
RF5.4	Modificar parámetro residencia
RF6	Gestionar parámetros tsu
RF6.1	Mostrar parámetros tsu
RF6.2	Adicionar parámetro tsu
RF6.3	Eliminar parámetro tsu
RF6.4	Modificar parámetro tsu
RF7	Gestionar parámetros guardia
RF7.1	Mostrar parámetros guardia
RF7.2	Adicionar parámetro guardia
RF7.3	Eliminar parámetro guardia
RF7.4	Modificar parámetro guardia
RF8	Gestionar parámetros feu-ujc
RF8.1	Mostrar parámetros feu-ujc
RF8.2	Adicionar parámetro feu-ujc
RF8.3	Eliminar parámetro feu-ujc
RF8.4	Modificar parámetro feu-ujc
RF9	Gestionar parámetros producción
RF9.1	Mostrar parámetros producción
RF9.2	Adicionar parámetro producción
RF9.3	Eliminar parámetro producción
RF9.4	Modificar parámetro producción
RF10	Gestionar evaluación por estudiante

RF10.1	Mostrar evaluación por estudiante
RF10.2	Adicionar evaluación por estudiante
RF10.3	Eliminar evaluación por estudiante
RF10.4	Modificar evaluación por estudiante
RF11	Generar escalafón de integralidad
RF12	Mostrar escalafón por grupos
RF13	Mostrar escalafón por facultad
RF14	Mostrar escalafón por usuario
RF15	Mostrar escalafón de integralidad de la universidad
RF 16	Generar reporte de escalafón

2.2.3 Requisitos no funcionales

Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación como restricciones en el diseño o estándares de calidad. Son propiedades o cualidades que el producto debe tener (28).

Usabilidad

- El módulo debe presentar una interfaz amigable que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada.
- Las vistas deben indicar en cada momento la acción que se está realizando así como lo íconos deben estar representados por una imagen acorde a la acción que se realiza mediante el mismo.
- El sistema debe adaptarse al lenguaje y términos utilizados por los clientes en la rama abordada con vista a una mayor comprensión por su parte sobre la herramienta de trabajo.

Seguridad

- Solo se mostrarán a los usuarios aquellas acciones o informaciones a las que por su responsabilidad o rol dentro del negocio necesitan acceder.
- El servidor de aplicaciones y de base de datos deberá mantener una seguridad mediante firewall para proteger el código y la información.

Confiabilidad

- El software puede permanecer inactivo durante 10 minutos. Al cumplirse este término se debe iniciar la sesión nuevamente para poder realizar alguna acción, en caso de no reiniciar la sesión no se mostrarán correctamente las funcionalidades del sistema.

Rendimiento

- El tiempo de respuesta de cada petición será menor de 10 segundos, excepto en el caso de la generación del EEU.

Software

- Para el despliegue del software estará instalado en el servidor el sistema operativo CentOS 6.2 donde se encuentre PHP v5.3 con las librerías php5-ldap, php5-gd, php5-mcrypt, php5-xsl, php5-openssl y Apache 2.2.
- Para el uso del software se requiere una PC cliente con el navegador web Mozilla Firefox 3.6 o superior.

Hardware

- Para la explotación del cliente se requiere de una PC Pentium IV o superior, CPU 1,4 GHZ o superior, 512 MB de RAM mínimo, 1 GB RAM recomendada o superior.
- Para la explotación del servidor se requiere de un CPU Dual Core 2.0 GHZ o superior, memoria RAM de 2 GB, 160 GB HDD.

2.2.4 Descripción de requisitos funcionales

Luego de realizada la Especificación de requisitos se realizaron las descripciones de los requisitos para el módulo Escalafón de integralidad. Caracterizado fundamentalmente por presentar los requisitos de forma completa, definiéndose todas las responsabilidades del módulo. Los productos de trabajo se encuentran en el expediente de proyecto donde se describen cada uno de los requisitos funcionales del módulo. A continuación se presenta la descripción del requisito Generar escalafón de integralidad.

Tabla 3. Descripción del requisito Generar escalafón de integralidad

Precondiciones	El usuario debe estar autenticado en el sistema y tener permisos requeridos.
Flujo de eventos	
Flujo básico	
1	El usuario accede al módulo “Escalafón”.
3	El sistema muestra una interfaz con todas las funcionalidades del módulo.
2	El usuario accede a la funcionalidad “Listado Estudiantil”.
3	El sistema muestra una interfaz donde el usuario puede ver el listado estudiantil.
4	El usuario presiona la opción “Generar escalafón”.
5	El sistema muestra una interfaz donde el usuario selecciona los grupos a los que le desea calcular el escalafón.
6	El usuario selecciona los grupos.
7	El usuario presiona la opción “Generar”.
8	Concluye el requisito.
Pos-condiciones	
1	Se calculan los valores de los indicadores y el índice de integralidad de cada estudiante de los grupos seleccionados.
Flujos alternativos	
Flujo alternativo 6.a El usuario no selecciona ningún grupo	
1	El sistema notifica que debe seleccionar algún grupo.

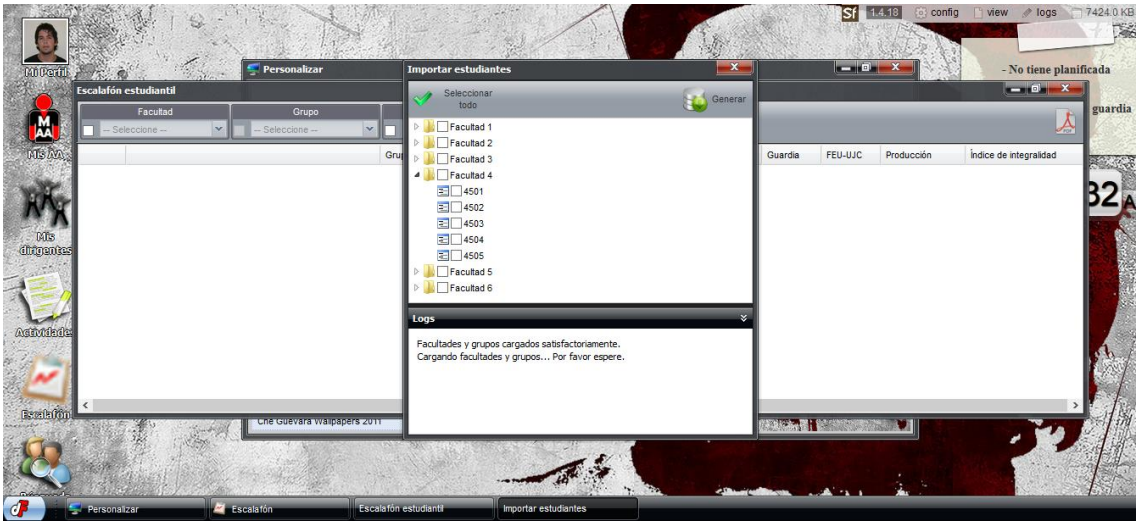


Figura 2. Prototipo de interfaz de usuario del requisito Generar escalafón de integralidad

2.2.5 Validación de requisitos funcionales

La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. El proceso de validación de requisitos comprende actividades que generalmente se realizan una vez obtenida una primera versión de la documentación de requisitos (29).

Las técnicas empleadas para la validación de los requisitos fueron las siguientes:

- **Revisiones técnicas formales:** se realizaron reuniones con el cliente para la lectura, revisión y corrección de la definición de requisitos, para de este modo obtener los posibles errores que pudiesen existir en la especificación de los requisitos del software. Se aprobaron los requisitos descritos de forma correcta, clara y consistente.
- **Construcción de prototipos:** consiste en construir una maqueta del futuro sistema de software, permiten al usuario hacerse una idea de la estructura de la interfaz del sistema; así el usuario corrige errores o añade aspectos para su completitud. Una vez concluida la descripción de los requisitos se realizaron los prototipos de interfaz de usuario utilizando la herramienta Visual Paradigm los cuales simulaban cómo debería quedar el sistema. Estos prototipos se revisaron con el cliente, quedando satisfecho con la propuesta realizada.
- **Generación de casos de prueba:** la realización de casos de prueba posibilita la verificación del cumplimiento de los requisitos funcionales y para cada requisito funcional se generó un diseño de caso de prueba para su futura comprobación.

2.3 Análisis y diseño

La fase de análisis y diseño garantiza la transformación de los productos de trabajo de los requisitos en los productos de trabajo que especifiquen el diseño del software que el proyecto va a desarrollar. Tiene como finalidad: transformar los requisitos en un diseño del sistema en creación,

evolucionar una arquitectura sólida para el sistema y adaptar el diseño para que se ajuste al entorno de implementación, con un diseño pensado para el rendimiento (30).

2.3.1 Estilo arquitectónico

La propuesta de solución está concebida como un módulo dentro del Sistema dataFEU, por esta razón se ajusta a la arquitectura definida para dicho sistema, la cual propone Cliente-Servidor. Haciendo uso del patrón Modelo-Vista-Controlador (MVC) que a su vez es el patrón de arquitectura base que utiliza el marco de trabajo Symfony1.

La arquitectura Cliente-Servidor consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Los clientes (o programas que representan entidades que necesitan servicios) y los servidores (o programas que proporcionan servicios) son objetos separados desde un punto de vista lógico y que se comunican a través de una red de comunicaciones para realizar una o varias tareas de forma conjunta. Un cliente hace una petición de un servicio y recibe la respuesta a dicha petición; un servidor recibe y procesa la petición, y devuelve la respuesta solicitada (31).

Características del cliente:

- Es quien comienza las solicitudes o peticiones, por lo que tienen un papel activo en la comunicación.
- Aguarda y recibe las respuestas del servidor.
- Por lo general, puede conectarse a varios servidores a la vez.

Características del servidor:

- Al iniciarse aguardan a que lleguen las solicitudes de los clientes, por lo que realizan un papel pasivo.
- Después de la recepción de una solicitud, la procesan y le envían la respuesta al cliente.
- Generalmente permiten conexiones desde un gran número de clientes (en ocasiones el número de peticiones puede estar limitado).

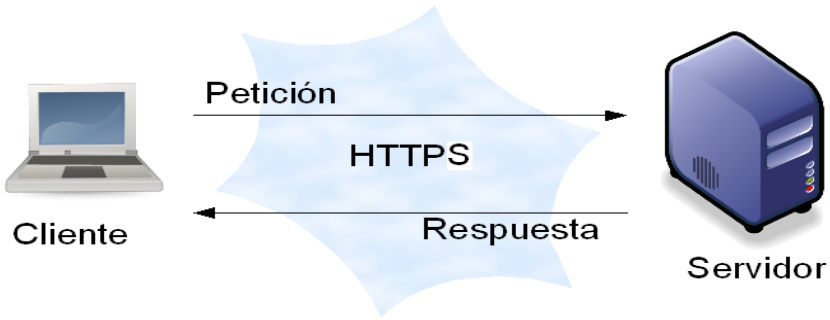


Figura 3. Arquitectura Cliente-Servidor

2.3.2 Patrón arquitectónico

Symfony1 implementa dentro de su marco de trabajo el patrón MVC, en el cual todo el proceso está dividido en 3 capas. Típicamente estas capas son el modelo, la vista y el controlador (32).

- **La vista:** el objeto que maneja la presentación visual de los datos representados por el modelo. Genera una representación visual del modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio modelo.

- **El modelo:** es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El modelo no tiene conocimiento específico de los controladores o de las vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas, y notificar a las vistas cuando cambia el modelo.

- **El controlador:** es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del modelo o por alteraciones de la vista. Interactúa con el modelo a través de una referencia al propio modelo.

Este patrón brinda una serie de ventajas como:

- Diseño modular y poco acoplado, favoreciendo la reutilización.
- Mayor cohesión ya que cada elemento del patrón está altamente especializado en su tarea.
- Las vistas proveen mayor flexibilidad y agilidad pues se pueden crear múltiples vistas de un modelo.
- Más claridad de diseño.

Figura 4. Patrón arquitectónico Modelo-Vista-Controlador

2.3.3 Diagrama de Clases del diseño

Los diagramas de clases del diseño son utilizados con el objetivo de representar las relaciones que existen entre los distintos tipos de clases. Para la realización de los siguientes diagramas se

tuvo en cuenta el marco de trabajo de Symfony 1, por lo que el mismo queda estructurado de la siguiente forma:

Vista: contiene las páginas clientes, los formularios y *java scripts*.

Modelo: contiene las clases que modelan los datos persistentes en la base de datos.

Controlador: compuesto por el controlador frontal y las clases controladoras.

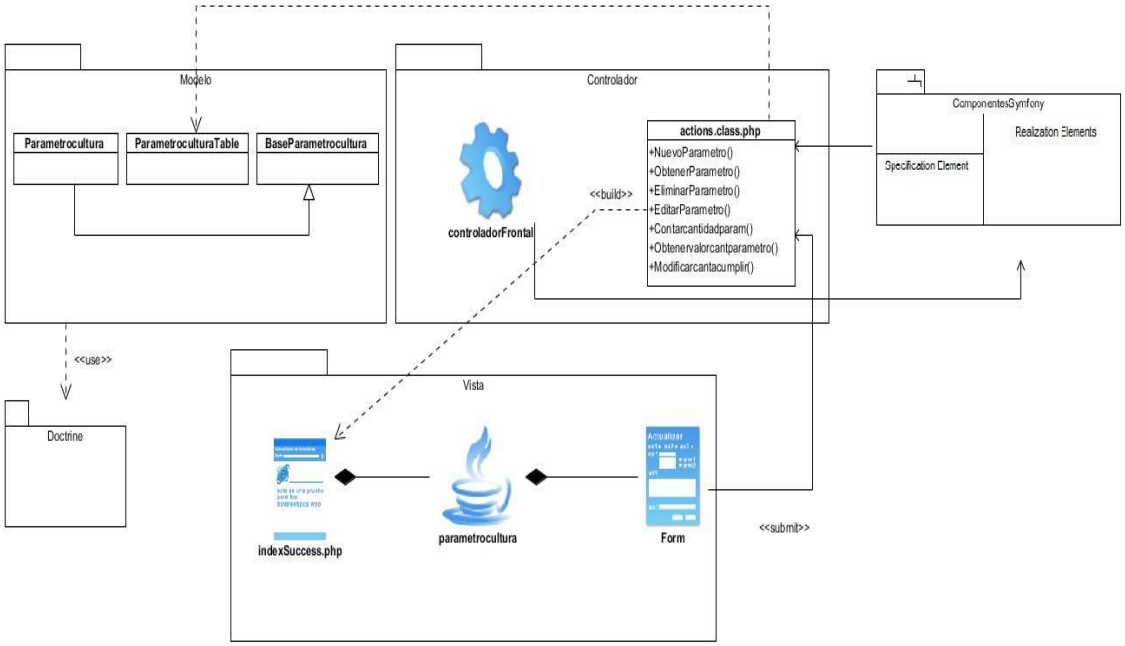


Figura 5. Diagrama de clases del diseño del requisito funcional Gestionar parámetro cultura

2.3.4 Modelo de datos

Un modelo de datos es un conjunto de conceptos, reglas y convenciones que permiten describir y en ocasiones manipular los datos del mundo real que se desee almacenar en la base de datos (33). El módulo Escalafón de integralidad requiere una base de datos donde se almacene toda la información para la gestión del proceso Generación del escalafón de integralidad. Este modelo representa la realidad a través de un esquema gráfico empleando entidades, relaciones y atributos. El modelo de datos que se muestra en la Figura 6 representa todas las entidades del módulo Escalafón de integralidad y se encuentra normalizada en tercera forma normal.

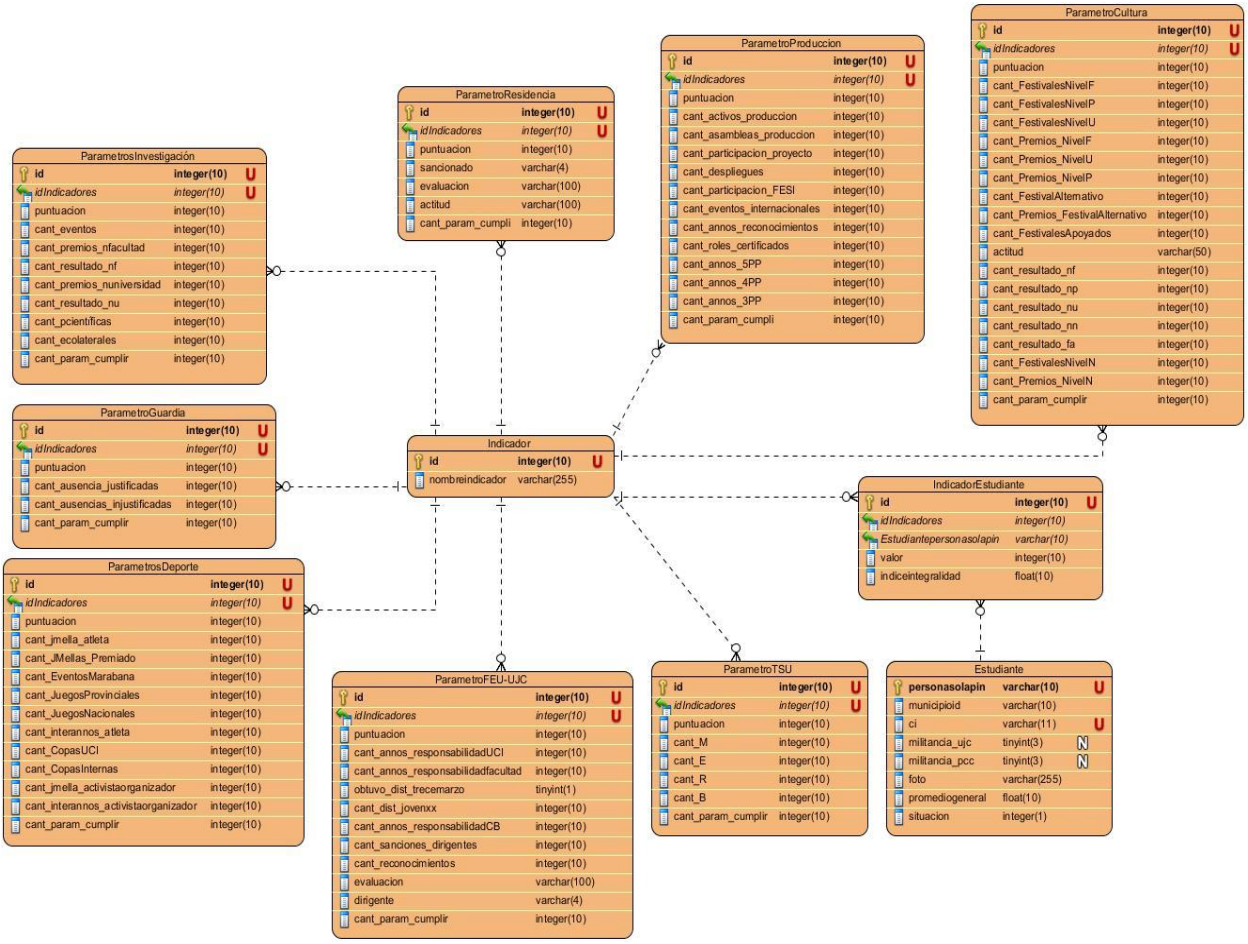


Figura 6. Modelo de datos del módulo Escalafón de integralidad

2.3.5 Patrones de diseño

Patrones GRASP

Los patrones GRASP describen los principios fundamentales del diseño de objetos para la asignación de responsabilidades. Para el diseño del módulo se tuvieron en cuenta los siguientes patrones GRASP:

- Experto: Symfony1, en el modelo, define dos tipos de clases, las encargadas de la abstracción de la base de datos y las de acceso a datos. Este marco de trabajo genera tres clases por cada tabla de la base de datos, en el caso de la tabla Indicador, se obtendrían las clases: Indicador, BaseIndicador e IndicadorTable. Las clases de abstracción de los datos serían las de tipo Base, que son las que trabajan directamente con la base de datos. Son las que tienen los atributos necesarios para obtener los registros de las tablas de la base de datos, en esa interacción directa es donde se pone de manifiesto el patrón experto.
- Creador: en la solución propuesta fue utilizado el patrón en las clases del paquete de dominio, las cuales son las encargadas de crear los objetos de tipo *query*, que permiten el acceso a la información almacenada a nivel de datos.

- Alta cohesión: una clase con mucha cohesión es cómoda ya que es bastante fácil darle mantenimiento, comprender y reutilizar. En el módulo queda evidenciado cuando cada clase solo se encarga de los eventos a los cuales se le ha asignado la responsabilidad. Un ejemplo de ello son las clases Actions, las cuales están formadas por varias funcionalidades que están estrechamente relacionadas.
- Bajo acoplamiento: se utilizó con el propósito de disminuir la dependencia entre las clases. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases. En el desarrollo de la propuesta solución se utilizó en todas las clases definiendo para cada una sus respectivos métodos con el fin de que el acoplamiento entre ellas fuera débil, para lograr la reutilización y el soporte.
- Controlador: se utilizó para que sirviera como intermediario entre cada una de las capas. Son todas aquellas clases controladoras declaradas en el módulo que son intermediarias entre la vista y los modelos. Ellas son las encargadas de ejecutar las funcionalidades para dar respuesta a la petición del cliente.

Patrones GoF

De estos patrones se emplearon para el diseño de la propuesta de solución los siguientes:

- Decorador: es un patrón estructural presente en el propio marco de trabajo Symfony1 y se puede apreciar en la clase abstracta sfView, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo nombrado layout.php es el que contiene el Layout de la página. Este archivo, conocido también como plantilla global, guarda el código HTML que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde otro punto de vista, el layout decora la plantilla.

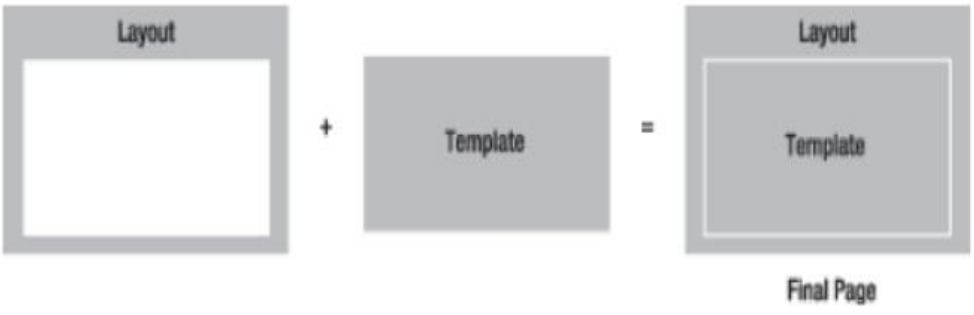


Figura 7. Ejemplo del patrón decorador

- Instancia única: garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Se puede apreciar en el controlador frontal donde hay una llamada a *sfContext* una clase del núcleo de Symfony, un objeto útil

que guarda una referencia a todos los objetos del núcleo de Symfony y puede ser accedido desde cualquier punto de la aplicación.

2.3.6 Validación del diseño

Para la validación del diseño se emplean algunas de las métricas para la medición del diseño desarrollado las cuales se concentran en las características internas de los componentes del software con medidas que pueden ayudar al desarrollador a juzgar la calidad de un diseño. Las métricas se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el desarrollo de software. Inclina sus objetivos a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo (34). Para validar el diseño se utilizaron las métricas Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC).

Tamaño Operacional de Clase (TOC)

La métrica Tamaño Operacional de Clase (TOC) está dada por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

- **Responsabilidad**: Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
- **Complejidad de implementación**: Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
- **Reutilización**: Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Fue aplicada a un total de 14 clases, para la definición de las categorías Alta, Media y Baja de los indicadores responsabilidad, complejidad de implementación y reutilización, fueron utilizados los siguientes criterios:

Tabla 4. Criterio responsabilidad

	Categoría	Criterio
Responsabilidad	Baja	Cantidad de procedimientos <= Prom
	Media	Prom < Cantidad de procedimientos <= 2*Prom
	Alta	Cantidad de procedimientos > 2*Prom

Tabla 5. Criterio complejidad implementación

	Categoría	Criterio
Complejidad implementación	Baja	Cantidad de procedimientos <= Prom
	Media	Prom < Cantidad de procedimientos <= 2*Prom
	Alta	Cantidad de procedimientos > 2*Prom

Tabla 6. Criterio reutilización

	Categoría	Criterio
Reutilización	Baja	Cantidad de procedimientos > 2*Prom
	Media	Prom < Cantidad de procedimientos < 2*Prom
	Alta	Cantidad de procedimientos <= Prom

Tabla 7. Aplicación de la métrica TOC

Clase	Cantidad de procedimientos	Responsabilidad	Complejidad	Reutilización
ParametrosinvestigacionTable	5	Baja	Baja	Alta
ParametroculturaTable	5	Baja	Baja	Alta
ParametrosdeporteTable	5	Baja	Baja	Alta
ParametroresidenciaTable	5	Baja	Baja	Alta
ParametrotsuTable	5	Baja	Baja	Alta
ParametroguardiaTable	5	Baja	Baja	Alta
ParametroproduccionTable	5	Baja	Baja	Alta
ParametrofeuujcTable	5	Baja	Baja	Alta
IndicadorTable	2	Baja	Baja	Alta
EstudianteTable	10	Media	Media	Media
PersonaTable	9	Media	Media	Media
Estudianteactions	8	Media	Media	Media
Parametroactions	9	Media	Media	Media
Indicadoractions	2	Baja	Baja	Alta

Al analizar los resultados obtenidos en cada uno de los atributos de calidad para esta métrica, se evidencia que el diseño presenta una calidad aceptable. El 71 porciento de las clases poseen una baja responsabilidad y un 29 porciento una responsabilidad media, esta característica permite que en caso de fallos en alguna de las clases el sistema no quede fuera de servicio debido que la responsabilidad está distribuida de manera equilibrada. Por otra parte el 71 porciento de las clases presentan una baja complejidad de implementación y el 29 porciento una complejidad media, esta característica permite mejorar el mantenimiento y soporte de estas clases. En el caso de la reutilización, se obtiene que el 71 porciento de las clases poseen una alta reutilización y el 29 porciento una reutilización media, lo que trae consigo que la gran mayoría de estas clases puedan ser reutilizadas nuevamente, disminuyendo el tiempo y complejidad de la implementación. En el siguiente gráfico se puede observar los resultados de la métrica.

Figura 8. Resultados gráficos de los criterios de la métrica TOC

Relaciones entre clases (RC)

Esta métrica evalúa la cantidad de relaciones existentes entre las 14 clases que forman el diseño propuesto. Para la definición de las categorías Alta, Media y Baja de los indicadores acoplamiento, complejidad de mantenimiento, reutilización y cantidad de pruebas, fueron utilizados los siguientes criterios:

Tabla 8. Criterio acoplamiento

	Categoría	Criterio
Acoplamiento	Ninguno	Cantidad de Relaciones de Uso = 0
	Ninguno	Cantidad de Relaciones de Uso = 1
	Medio	Cantidad de Relaciones de Uso = 2
	Alto	Cantidad de Relaciones de Uso > 2

Tabla 9.Criterio complejidad de mantenimiento

	Categoría	Criterio
Complejidad de Mantenimiento	Baja	Promedio de asociaciones de uso <= Prom
	Media	Prom < Promedio de asociaciones de uso <= 2*Prom
	Alta	Promedio de asociaciones de uso > 2*Prom

Tabla 10.Criterio reutilización

	Categoría	Criterio
Reutilización	Baja	Promedio de asociaciones de uso > 2*Prom
	Media	Prom < Promedio de asociaciones de uso <= 2*Prom
	Alta	Promedio de asociaciones de uso <= Prom

Tabla 11. Criterio cantidad de pruebas

	Categoría	Criterio
Cantidad de Pruebas	Baja	Promedio de asociaciones de uso <= Prom
	Media	Prom < Promedio de asociaciones de uso <= 2*Prom
	Alta	Promedio de asociaciones de uso > 2*Prom

Tabla 12. Aplicación de la métrica RC

Subsistema	Clase	CRU	Acoplamiento	CM	Reutilización	CP
Módulo Escalafón de integralidad	ParametrosinvestigacionTable	1	Bajo	Baja	Alta	Baja
	ParametroculturaTable	1	Bajo	Baja	Alta	Baja
	ParametrosdeporteTable	1	Bajo	Baja	Alta	Baja

	ParametroresidenciaTable	1	Bajo	Baja	Alta	Baja
	ParametrotsuTable	1	Bajo	Baja	Alta	Baja
	ParametroguardiaTable	1	Bajo	Baja	Alta	Baja
	ParametroproduccionTable	1	Bajo	Baja	Alta	Baja
	ParametrofeuujcTable	1	Bajo	Baja	Alta	Baja
	IndicadorTable	1	Bajo	Baja	Alta	Baja
	EstudianteTable	1	Bajo	Baja	Alta	Baja
	PersonaTable	1	Bajo	Baja	Alta	Baja
	Estudianteactions	2	Medio	Media	Media	Media
	Parametroactions	2	Medio	Media	Media	Media
	Indicadoractions	2	Medio	Media	Media	Media

Al analizar los resultados obtenidos en cada uno de los atributos de calidad para esta métrica, se evidencia que el diseño presenta una calidad aceptable. El 79 porciento de las clases poseen un bajo acoplamiento y un 21 porciento un acoplamiento medio, este resultado permite que al ocurrir algún error en una de las clases este tenga el menor impacto posible en las otras clases. Por otra parte el 79 porciento de las clases presentan una complejidad de mantenimiento baja y un 21 porciento una complejidad de mantenimiento media. En el caso de los valores de cantidad de pruebas se obtiene un 79 porciento baja y un 21 porciento media. En cuanto a la reutilización se puede apreciar que el 79 porciento de las clases poseen una alta reutilización y un 21 porciento media. En el siguiente gráfico se puede observar los resultados de la métrica.

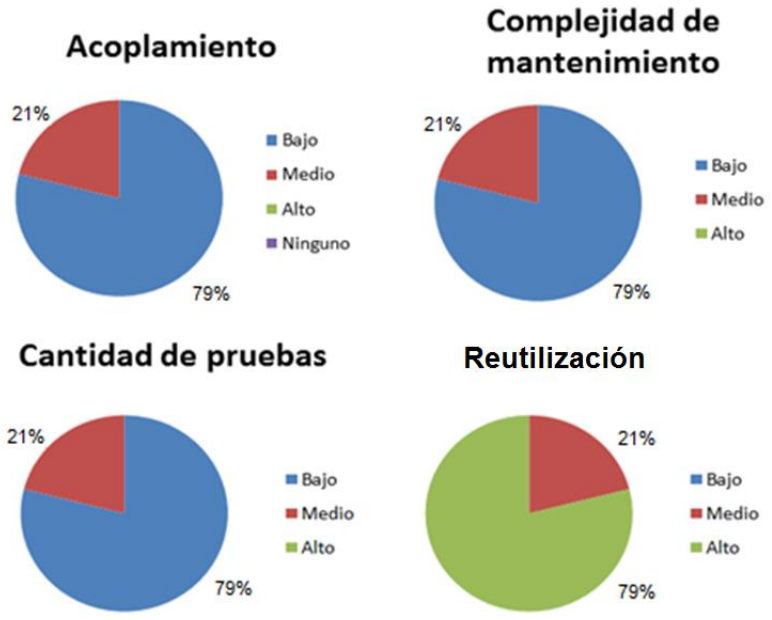


Figura 9.Resultados gráficos de los criterios de la métrica TOC

2.4 Conclusiones parciales

Después de la valoración de cómo se lleva a cabo actualmente el proceso de elaboración del EEU y de lo que se pretende lograr una vez que se informatice el proceso, se logró identificar un total de 43 requisitos funcionales y 11 no funcionales. La utilización de los patrones de diseño y el patrón arquitectónico MVC durante el desarrollo del módulo, proporciona una mayor calidad del producto. El modelo de datos elaborado permite conocer las relaciones existentes entre las diferentes tablas de la base de datos y el diagrama de clases de diseño permitió conocer la estructura y las relaciones entre las clases que se manejan en el módulo. Los resultados obtenidos de los atributos de calidad al aplicar las métricas TOC y RC evidencian que el diseño presenta una calidad aceptable.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DEL SISTEMA

Introducción

El presente capítulo abarca las disciplinas de implementación que emplea los resultados obtenidos en las disciplinas anteriores del proceso de desarrollo de software. Se realiza el diseño de la implementación del módulo propuesto realizándose los productos de trabajo diagrama de componentes y modelo de despliegue. Se presentan los estándares de codificación a seguir para la implementación del módulo. Además se realizan las pruebas para la validación de la propuesta de solución.

3.1 Implementación

Una vez obtenidos los distintos productos de trabajos generados en la disciplina de análisis y diseño se transita a la implementación de la propuesta de solución. Disciplina en la cual se pretende crear código ejecutable, o sea, código fuente que se puede compilar. Además es creada la documentación necesaria para la instalación, operación y soporte del software y es realizada la integración del sistema, es decir, componer e integrar por separado los componentes de software para que conformen un solo producto entregable (35).

3.1.1 Diagrama de componentes

En el diagrama de componentes se describen los elementos físicos del sistema y sus relaciones. Los componentes representan elementos de software incluidos en el desarrollo del módulo Escalafón de integralidad, pueden ser archivos, paquetes y bibliotecas cargadas dinámicamente.

En la vista se pueden apreciar cada una de las plantillas presentes en el módulo a las cuales se le suma la utilización de la plantilla global y de los componentes de Extjs y Ajax. En el controlador están presente cada una de las clases controladoras empleadas en el desarrollo de la propuesta de solución, también está el controlador frontal y se hace uso de los componentes del marco de trabajo symfony 1.4.8. En el modelo se representan cada una de las clases encargadas del acceso a los datos utilizadas, los componentes del mapeador de objetos relacionales Doctrine y la base de datos. A continuación se presenta el diagrama de componentes correspondiente a la propuesta de solución.

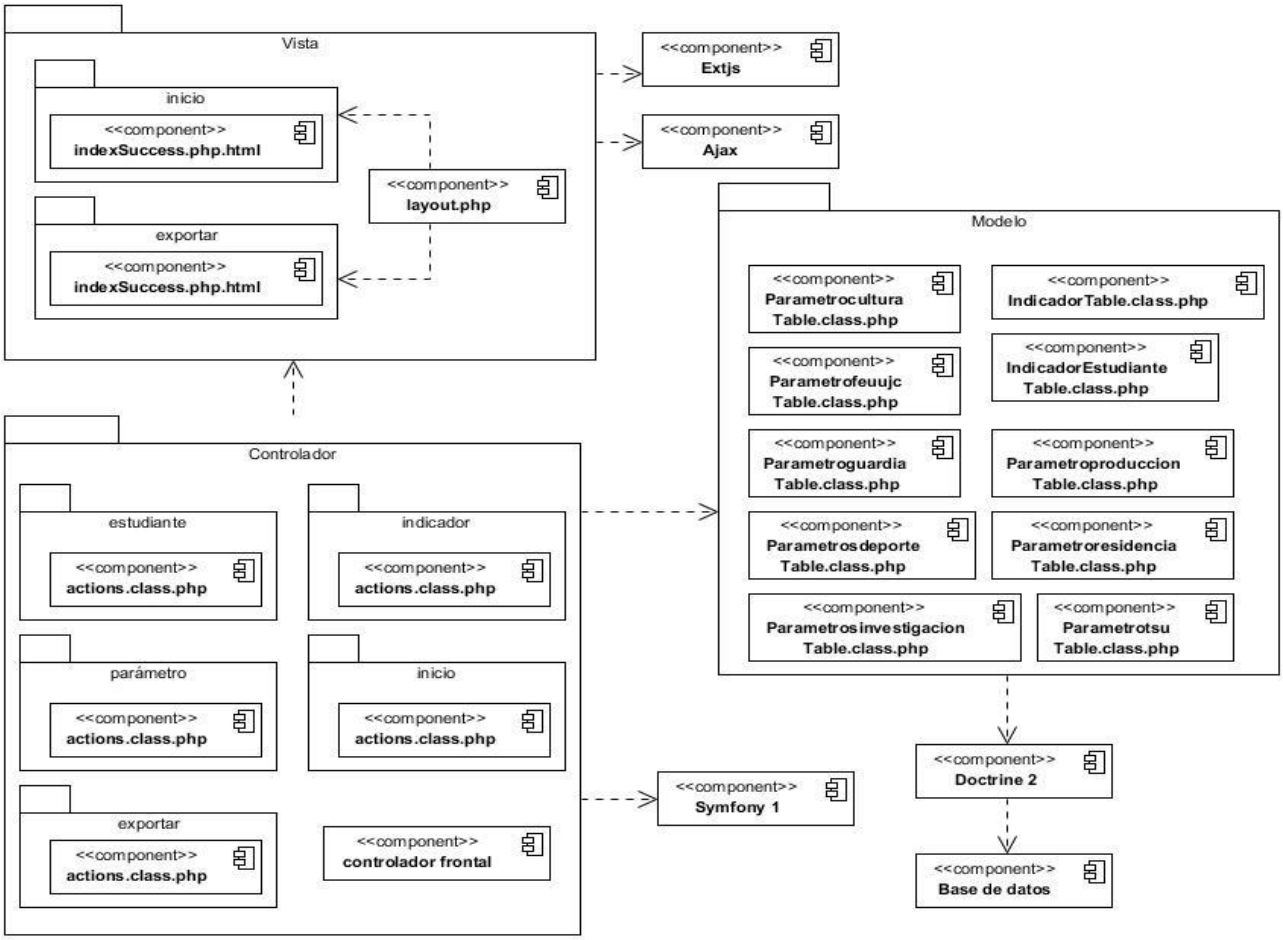


Figura 10. Diagrama de componentes

3.1.2 Modelo de despliegue

Un diagrama de despliegue es utilizado para representar la distribución física de un sistema. Muestra las relaciones físicas entre los componentes hardware y software en el sistema final a través de nodos, los cuales serían una computadora (PC), los servidores donde está alojada la aplicación y aquellos que brindan los servicios necesarios para el funcionamiento del sistema. La conexión de la PC cliente puede ser a través de la red cableada o inalámbrica. En este caso la aplicación se encuentra hospedada en un servidor web mientras que la base de datos (MySQL) se encuentra en otro servidor, la comunicación entre ambos se realiza mediante servicios. El diagrama de despliegue para el sistema de caracterización se representa en la Figura 8.

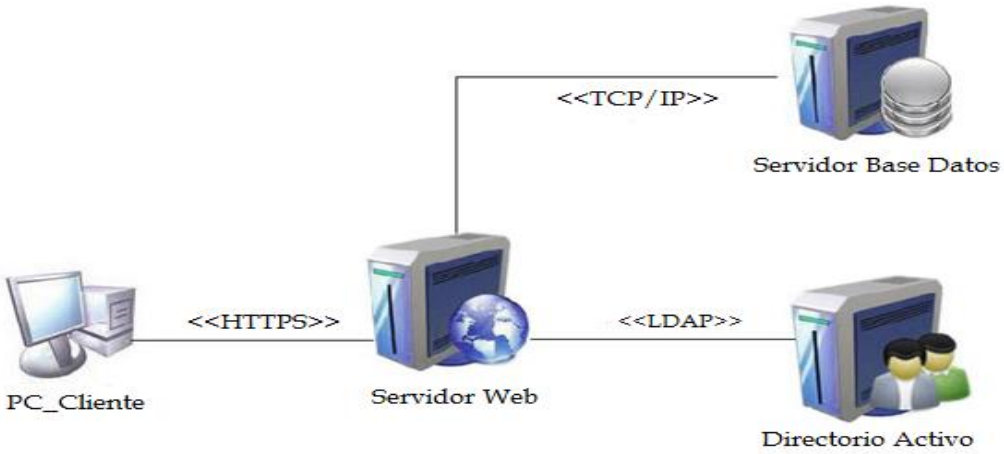


Figura 11. Modelo de despliegue

3.1.3 Estándares de código

A continuación se describe el estándar de código utilizado para el desarrollo del módulo Escalafón de integralidad, con el objetivo de mantener una uniformidad en el código se utilizaron los mismos estándares de códigos utilizados en el desarrollo del Sistema dataFEU.

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

Usar una identación de un tabulador con un equivalente a 4 espacios, para mantener integridad en las revisiones de *Subversion*. El uso de las llaves "{}" será en la misma línea y la longitud de las líneas de código es aproximadamente de 75-80 caracteres, para mantener la legibilidad del mismo.

Convención de nomenclatura

Variables: se rigen por la nomenclatura *camelCase*. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra será con mayúscula.

Clases: se rigen por la nomenclatura *Dromedarian* y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.

Funciones: 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. Los parámetros son separados por espacio luego de la coma que los separa.

Controladoras: mismo nombre de la clase que representa seguido de la palabra *Actions*.

Estructuras de control

Las estructuras de control incluyen *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. Con esto se aumenta la legibilidad del código y se disminuye la probabilidad de errores lógicos.

Si las condiciones son largas que sobrepasan el tamaño de la línea, estas se dividen en varias líneas. En el mejor de los casos cuando la condición es extensa, se puede dividir en variables y compararlas dentro de la estructura de control.

Documentación

Todos los archivos deben de tener la documentación asociada al mismo. Para esto debe de cumplirse con la documentación de tipo comentario en aras de facilitar el trabajo a la hora del completamiento de código que brinda los entornos de desarrollo actuales.

3.2 Interfaces de la aplicación

A partir de los artefactos obtenidos en las etapas previas y mediante el uso de los estándares de codificación mencionados, fue realizada la implementación de la solución, obteniendo el módulo Escalafón de integralidad. En su desarrollo fueron obtenidas interfaces gráficas que responden a los requisitos funcionales que fueron definidos antes de implementar. A continuación se muestran las interfaces antes mencionadas, otras de las interfaces se pueden ver en el anexo 3:

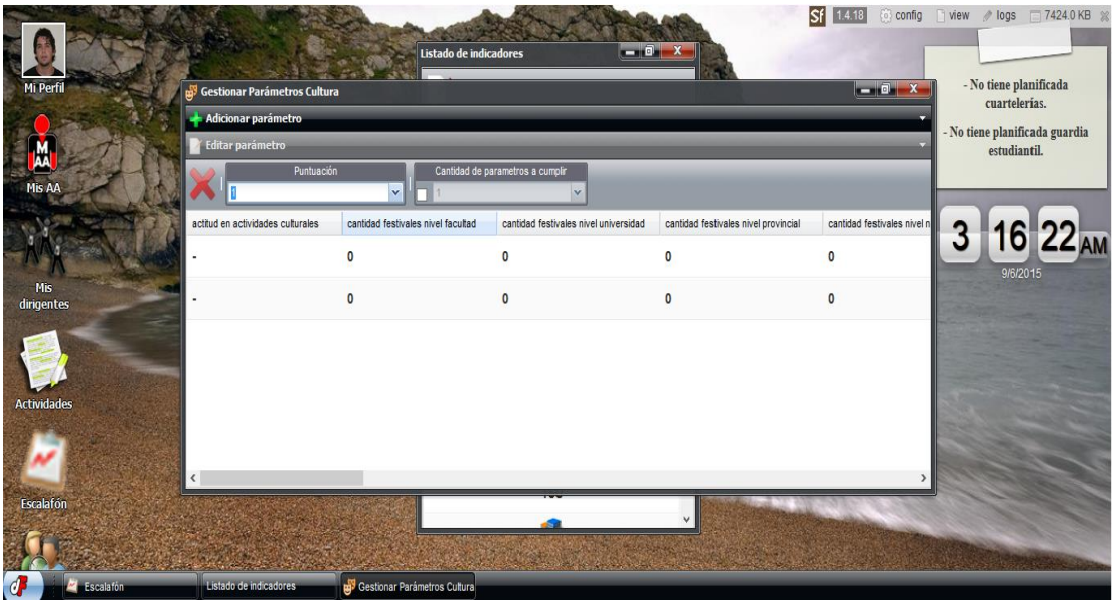


Figura 12. Interfaz de módulo Escalafón de integralidad *Gestionar parámetro cultura*

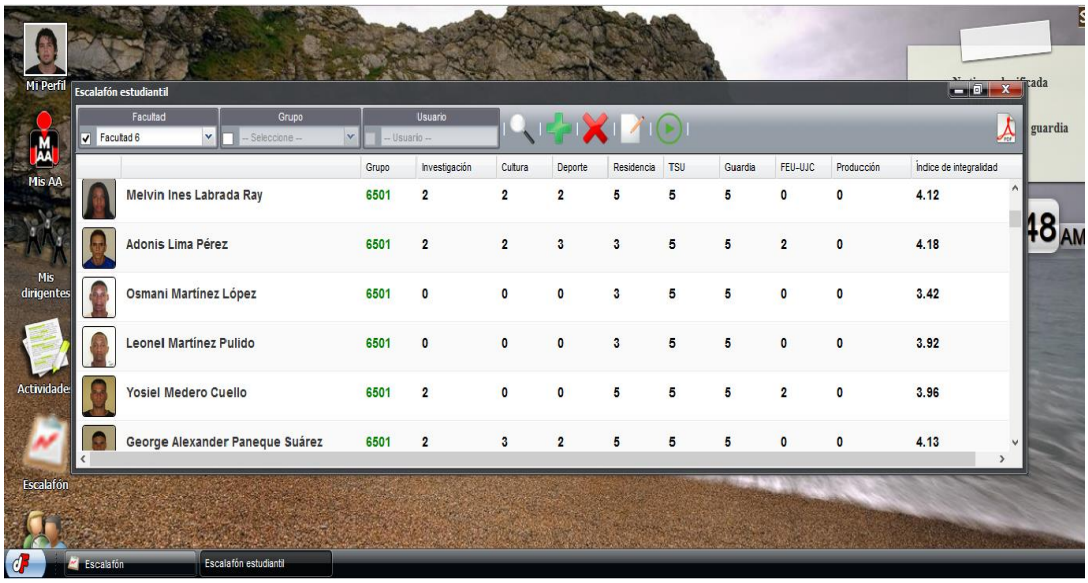


Figura 13. Interfaz del módulo Escalafón de integralidad *Escalafón de integralidad*

3.3 Pruebas

Las pruebas se consideran indispensables en el proceso de desarrollo de software para la obtención de buenos resultados. Se realizan con el objetivo de revisar que todo el software tenga el nivel de calidad requerido. Permiten detectar errores durante el funcionamiento del software, probando así la entrada y salida correcta de datos. En la ejecución de las pruebas lo que se persigue es la detección de discrepancias entre la salida obtenida y la esperada. Dicha situación se interpreta entonces como un síntoma de problemas con el software.

3.3.1 Niveles de pruebas

Un nivel de prueba es un grupo de actividades que se organizan y administran juntas dentro de la ejecución de un proceso de pruebas que tiene como objetivo verificar y validar los componentes de un producto. Las pruebas se aplican a diferentes tipos de objetivos en diferentes niveles del proceso de pruebas. Estos niveles permiten seleccionar diferentes tipos y técnicas de pruebas a realizar en cada nivel (36).

Pruebas Unitarias o de Componentes: consisten en la ejecución de actividades que le permitan verificar al desarrollador que los componentes unitarios están codificados bajo condiciones de robustez, esto es, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada. Adicionalmente, las pruebas sobre componentes unitarios, suelen denominarse pruebas de módulos o pruebas de clases, siendo la convención definida por el lenguaje de programación la que influye en el término a utilizar.

Pruebas de integración: es ejecutada para asegurar que los componentes software y hardware del producto operan adecuadamente cuando interactúan entre ellos para ejecutar un caso de uso (o transacción de negocio).Las pruebas de integración exponen la no compleción o los errores en las especificaciones de la interfaz de cada paquete software siendo integrado con los demás.

Pruebas del sistema: tienen como propósito fundamental ejercitar profundamente el sistema desarrollado, con el objetivo de verificar que se hayan integrado todos los elementos del mismo y que realizan correctamente las funciones descritas. Este tipo de pruebas estudia el producto completo para analizar defectos globales o para estudiar aspectos específicos de su comportamiento, tales como seguridad y rendimiento.

Pruebas de aceptación: la prueba de aceptación de usuario es la prueba final antes de desplegar el software en los ambientes de operación. El objetivo de la prueba de aceptación es verificar que el software está listo, que puede ser utilizado, que satisface los criterios de aceptación, y que cubre aquellas necesidades y expectativas de los clientes para los cuales el software se construyó.

3.3.2 Métodos de pruebas

Los métodos de pruebas se encargan de definir la estrategia de verificación y validación del sistema con el objetivo de demostrar que existen fallos, no la ausencia de estos. Tienen el objetivo de diseñar pruebas que descubran diferentes tipos de errores con menor tiempo y esfuerzo. Existen dos métodos de pruebas fundamentales: el método de caja negra (denominada prueba funcional) y el de caja blanca (37).

Pruebas de caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo, ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; ejecuten todos los ciclos en sus límites y con sus límites operacionales, y ejerciten las estructuras internas de datos para asegurar su validez (37).

En la prueba de caja blanca se empleó la técnica del Camino básico, a partir del cálculo de la complejidad ciclomática del algoritmo a ser analizado. Para realizarla se deben enumerar las sentencias de código y a partir de ahí elaborar el grafo de flujo de esta funcionalidad.

Se definieron una serie de pasos a seguir que se relacionan a continuación:

1. Notación del grafo de flujo: utilizando el código como base, se realiza la representación del grafo de flujo mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo.

- Nodo: cada círculo denominado nodo, representa una o más sentencias procedimentales.
- Arista: las flechas del grafo de flujo, denominadas aristas, representan el flujo de control y son análogas a las flechas del diagrama de flujo.

- Región: las áreas delimitadas por aristas y nodos se denominan regiones.

2. Complejidad ciclomática: es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa. Esto indica el límite superior para el número de pruebas que se deben realizar, para asegurar que se ejecuta cada sentencia al menos una vez. Se utilizó las siguiente forma: V (G), de un grafo de flujo G se define como: V (G) = A – N + 2, V (G) = NP + 1 y V (G) = R, donde A es el número de aristas del grafo de flujo, N es el número de nodos, NP es el número de nodos predicados y R el número de regiones.

3. Determinar un conjunto básico de caminos linealmente independientes: el valor de V (G) es el número de caminos linealmente independientes de la estructura de control del programa.

4. Obtención de casos de prueba: se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico. Se determinó aplicar ésta prueba a la funcionalidad *Obtenervalorcantparametro* encargada de obtener el valor de la variable cantparametroacumplir, de cada puntuación de los distintos indicadores. A continuación es mostrado el código de la funcionalidad seleccionada con las sentencias enumeradas y el grafo de flujo asociado al código.

```
public function executeObtenervalorcantparametro(sfWebRequest $request) {
    switch ($request->getParameter('idi')) { 1
        case 1: 2
            $resultado = ParametrosinvestigacionTable::obtenervalorcantparametro($request->getParameter('puntuacion')); 3
            break;
        case 2: 4
            $resultado = ParametroculturaTable::obtenervalorcantparametro($request->getParameter('puntuacion')); 5
            break;
        case 3: 6
            $resultado = ParametrosdeporteTable::obtenervalorcantparametro($request->getParameter('puntuacion')); 7
            break;
        case 4: 8
            $resultado = ParametroresidenciaTable::obtenervalorcantparametro($request->getParameter('puntuacion')); 9
            break;
        case 5: 10
            $resultado = ParametrotsuTable::obtenervalorcantparametro($request->getParameter('puntuacion')); 11
            break;
        case 6: 12
            $resultado = ParametroguardiaTable::obtenervalorcantparametro($request->getParameter('puntuacion')); 13
            break;
        case 7: 14
            $resultado = ParametrofeuujcTable::obtenervalorcantparametro($request->getParameter('puntuacion')); 15
            break;
        case 8: 16
            $resultado = ParametroproduccionTable::obtenervalorcantparametro($request->getParameter('puntuacion')); 17
            break;
    }
    return $this->renderText(json_encode($resultado)); 18
}
```

Figura 14. Funcionalidad Obtenervalorcantparametro

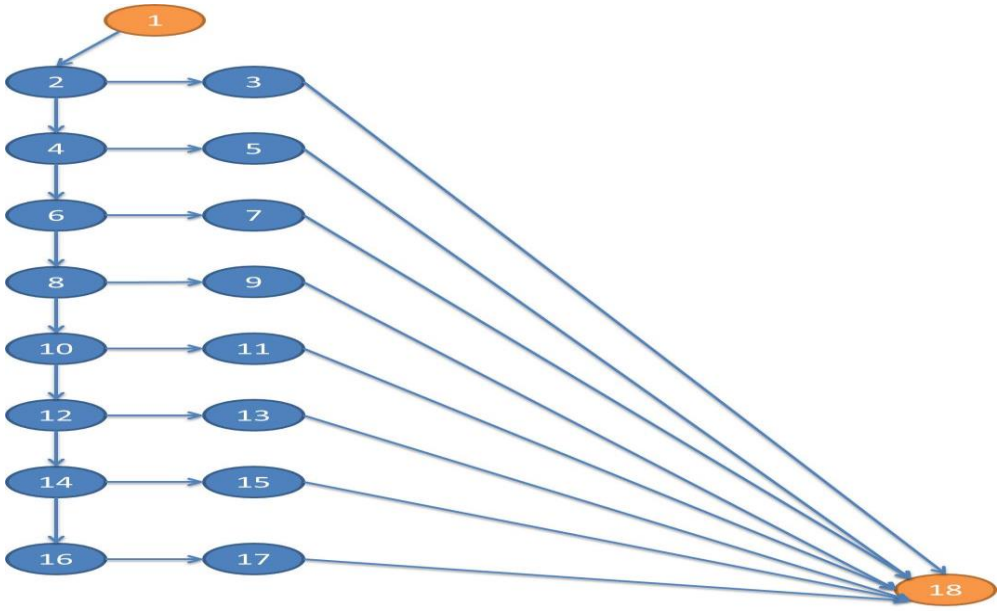


Figura 15. Grafo de Flujo asociado al método *Obtenervalorcantparametro*

Cálculo de la complejidad ciclomática:

V (G) = A – N + 2 (A: aristas, N: nodos)

V (G) = 24 – 18 + 2

V (G) = 8

V (G) = NP + 1 (NP: nodos predicados)

V (G) = 7 + 1 = 8

V (G) = R (R: regiones)

V (G) = 8

A partir del cálculo realizado se obtiene que la complejidad ciclomática de la funcionalidad es ocho, lo que significa que existen ocho posibles caminos por donde el flujo puede circular.

A continuación se muestran los caminos básicos por donde puede circular el flujo:

Camino básico uno: 1-2-3-18
Camino básico dos: 1-2-4-5-18
Camino básico tres: 1-2-4-6-7-18
Camino básico cuatro: 1-2-4-6-8-9-18
Camino básico cinco: 1-2-4-6-8-10-11-18
Camino básico seis: 1-2-4-6-8-10-12-13-18
Camino básico siete: 1-2-4-6-8-10-12-14-15-18
Camino básico ocho: 1-2-4-6-8-10-12-14-16-17-18

Cada camino independiente es un caso de prueba a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino.

Caso de prueba del camino uno:

- **Descripción**: En la interfaz Gestionar parámetros investigación selecciona una puntuación determinada.
- **Valores de entrada**: variable *idi* igual a uno y variable *puntuación* igual a cualquier valor del uno al cinco.
- **Resultados esperados**: Se modifica en la interfaz el valor de la cantidad de parámetros a cumplir para la puntuación seleccionada del indicador investigación.

Caso de prueba del camino dos:

- **Descripción**: En la interfaz Gestionar parámetros cultura selecciona una puntuación determinada.

- **Valores de entrada**: variable *idi* igual a dos y variable *puntuación* igual a cualquier valor del uno al cinco.
- **Resultados esperados**: Se modifica en la interfaz el valor de la cantidad de parámetros a cumplir para la puntuación seleccionada del indicador cultura.

Caso de prueba del camino tres:

- **Descripción**: En la interfaz Gestionar parámetros deporte selecciona una puntuación determinada.
- **Valores de entrada**: variable *idi* igual a tres y variable *puntuación* igual a cualquier valor del uno al cinco.
- **Resultados esperados**: Se modifica en la interfaz el valor de la cantidad de parámetros a cumplir para la puntuación seleccionada del indicador deporte

Caso de prueba del camino cuatro:

- **Descripción**: En la interfaz Gestionar parámetros residencia selecciona una puntuación determinada.
- **Valores de entrada**: variable *idi* igual a cuatro y variable *puntuación* igual a cualquier valor del uno al cinco.
- **Resultados esperados**: Se modifica en la interfaz el valor de la cantidad de parámetros a cumplir para la puntuación seleccionada del indicador residencia.

Caso de prueba del camino cinco:

- **Descripción**: En la interfaz Gestionar parámetros TSU selecciona una puntuación determinada.
- **Valores de entrada**: variable *idi* igual a cinco y variable *puntuación* igual a cualquier valor del uno al cinco.
- **Resultados esperados**: Se modifica en la interfaz el valor de la cantidad de parámetros a cumplir para la puntuación seleccionada del indicador TSU.

Caso de prueba del camino seis:

- **Descripción**: En la interfaz Gestionar parámetros guardia selecciona una puntuación determinada.
- **Valores de entrada**: variable *idi* igual a seis y variable *puntuación* igual a cualquier valor del uno al cinco.
- **Resultados esperados**: Se modifica en la interfaz el valor de la cantidad de parámetros a cumplir para la puntuación seleccionada del indicador guardia.

Caso de prueba del camino siete:

- **Descripción**: En la interfaz Gestionar parámetros FEU-UJC selecciona una puntuación determinada.
- **Valores de entrada**: variable *idi* igual a siete y variable *puntuación* igual a cualquier valor del uno al cinco.

- **Resultados esperados**: Se modifica en la interfaz el valor de la cantidad de parámetros a cumplir para la puntuación seleccionada del indicador FEU-UJC.

Caso de prueba del camino ocho:

- **Descripción**: En la interfaz Gestionar parámetros producción selecciona una puntuación determinada.
- **Valores de entrada**: variable *idi* igual a ocho y variable *puntuación* igual a cualquier valor del uno al cinco.
- **Resultados esperados**: Se modifica en la interfaz el valor de la cantidad de parámetros a cumplir para la puntuación seleccionada del indicador producción.

Una vez concluida la ejecución de los distintos casos de pruebas correspondientes a cada camino básico se evidenció el correcto funcionamiento del método *Obtenervalorcantparametro*. Los resultados de la aplicación de la técnica del Camino básico a los principales métodos del módulo, posibilitó optimizar el código.

Pruebas de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los detectados por los métodos de caja blanca. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación (37).

Para la realización de las pruebas de caja negra, se emplea la técnica Partición de equivalencia, cuyo objetivo fundamental es dividir el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba.

Tabla 13. Caso de prueba para validar RF Adicionar evaluación por estudiante

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1. Adicionar evaluación por estudiante.	1. El sistema debe permitir que el usuario al realizar la búsqueda de estudiantes, pueda	E.P 1.1 Flujo básico de eventos.	1. El usuario accede desde el módulo Escalafón y selecciona la opción “Escalafón estudiantil”. 2. El usuario busca a uno o varios estudiantes por facultad, facultad o

	insertarle las evaluaciones de cada indicador a un estudiante seleccionado dando clic en el botón agregar evaluaciones.		grupo, por usuario, o toda la universidad. 3. El usuario selecciona un estudiante del listado. 4. El usuario da clic en el botón “Agregar evaluación”. 5. El usuario inserta cada uno de los valores referentes a cada indicador. 6. El sistema muestra un mensaje indicando que se insertó correctamente las evaluaciones.
		E.P 1.2 Se dejan campos vacíos.	1. El usuario accede desde el módulo Escalafón y selecciona la opción “Escalafón estudiantil”. 2. El usuario busca a uno o varios estudiantes por facultad, facultad o grupo, por usuario, o toda la universidad. 3. El usuario selecciona un estudiante del listado. 4. El usuario da clic en el botón “Agregar evaluación”. 5. El usuario inserta cada uno de los valores referentes a cada indicador. 6. El sistema muestra un mensaje indicando que no se pueden dejar campos en blancos.

Los juegos de datos a probar están descritos en el producto de trabajo Diseño_de_casos_de_prueba del expediente de proyecto.

3.3.3 Resultados de las pruebas

Durante el desarrollo de las pruebas se realizaron tres iteraciones, las cuales permitieron detectar en una primera iteración un total de catorce no conformidades, de las cuales cuatro fueron errores de funcionalidad y diez de ortografía. Todas las no conformidades fueron corregidas permitiendo que el módulo pasara a una segunda iteración.

En la segunda iteración se detectaron un total de cinco no conformidades de las cuales tres fueron errores de funcionalidad y dos de ortografía. Una vez detectadas las no conformidades se erradicaron cada una de ellas para así realizar una tercera iteración en la cual no se encontraron no conformidades. Una vez concluida la eliminación de todas las no conformidades se logró que el módulo realizase todas las funcionalidades especificadas en la disciplina de requisitos de manera correcta logrando así la satisfacción del cliente, quedando como constancia la carta de aceptación del cliente (ver anexo 4).

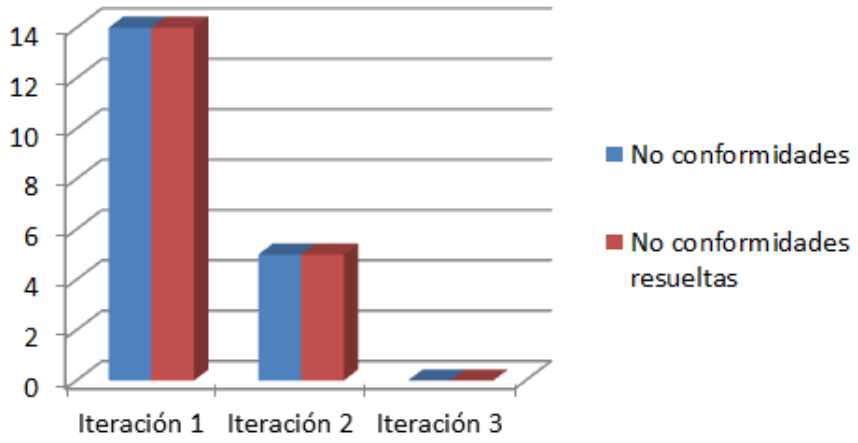


Figura 16. Resultado de las pruebas de caja negra

3.4 Impacto de la solución

Antes de la implantación del sistema, el proceso de elaboración del EEU se realizaba durante dos meses aproximadamente teniendo en cuenta los cronogramas que se registran en las actas de los Consejos de la FEU de la Universidad y los acuerdos de los Consejos de Dirección de la Universidad. Una vez concluida la implementación del módulo Escalafón de integralidad se realizaron experimentos para verificar el tiempo que se tardaba en concluir el cálculo del índice de integralidad. El experimento se realizó escogiendo cinco dirigentes de la FEU a nivel de universidad arrojando un tiempo promedio de generación del escalafón de seis minutos. De esta forma queda evidenciada la obtención de resultados satisfactorios en cuanto a la reducción del tiempo de generación del EEU.

3.5 Conclusiones parciales

En este capítulo se realizó la implementación y validación del módulo, arrojando resultados como: la elaboración del diagrama de componentes y de despliegue facilitó el proceso de implementación del módulo. La utilización de estándares de codificación permitió realizar la codificación de una manera más legible y entendible. Con la realización de las pruebas de software, se logró obtener un producto con mayor calidad una vez transitado por las distintas iteraciones, las cuales permitieron la identificación de no conformidades. Una vez concluida la implementación y prueba se pudo verificar que la solución propuesta mejora en un amplio margen el tiempo de procesamiento de la información.

CONCLUSIONES GENERALES

Con la culminación del presente trabajo de diploma se puede concluir que:

- Se realizó el marco teórico referente a la investigación donde se establecieron las tendencias marcadas en cuanto al desarrollo de escalafones estudiantiles universitarios y permitió fundamentar la necesidad de desarrollar el módulo.
- La realización del análisis y diseño permitió obtener los requisitos y modelos necesarios para la implementación del módulo Escalafón de integralidad.
- Se obtuvo el módulo para el Sistema dataFEU que permite la generación del escalafón estudiantil universitario cumpliendo con los requisitos definidos por el cliente.
- La validación de la investigación se realizó a partir de la aplicación de técnicas, métricas y pruebas que garantizaron el correcto funcionamiento de la solución desarrollada.

RECOMENDACIONES

Como recomendación del presente trabajo se tienen:

- Migrar el sistema a versiones más recientes de Symfony.
- Integrar futuras versiones a una red social, creando una plataforma más interactiva y dinámica donde también los padres de los estudiantes tengan acceso al sistema y puedan acceder a la información relacionada con sus hijos.

BIBLIOGRAFÍA

1. **González Cardoso, Victor Grabiel y Figueredo Bustamante, Eiler.** *Sistema para la gestión de los procesos de la Federación Estudiantil.* 2014.

2. **Martínez Rivera, Anigleidis y Estrada Rodríguez, Maisel Luis.** *Solución informática para la gestión del proceso de caracterización estudiantil y generación de reportes del Sistema de Caracterización Integral.* La Habana : s.n., 2012.

3. Departamento de educación de Puerto Rico. [En línea] [Citado el: 12 de febrero de 2015.] http://www.de.gobierno.pr/padres-y-estudiantes.

4. **Llerena Álvarez, Saeret.** *Solución informática para la gestión de ubicación laboral en la Universidad.* La Habana : s.n., 2013.

5. **Virrueta Méndez, Alejandra.** [En línea] 2010. [Citado el: 15 de febrero de 2015.] http://www.monografias.com/trabajos-pdf4/metodologias-de-desarrollo-software/metodologias-de-desarrollo-software.pdf.

6. **Rodríguez Sánchez, Tamara.** *Metodología de desarrollo.* La Habana : s.n., 2014.

7. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n.

8. Ciberaula. [En línea] [Citado el: 12 de 4 de 2015.] http://linux.ciberaula.com/articulo/linux_apache_intro/.

9. **Relacionados., Historia de PHP y Proyectos.** Historia de PHP y Proyectos Relacionados. [En línea] [Citado el: 22 de enero de 2015.] http://php.net/manual/es/history.php.

10. **Potencier, Fabien y Zaninotto, François.** *Symfony la guía definitiva.*

11. **Romer, Michael.** *Doctrine.* Alemania : Leanpub, 2013.

12. Introducción a JavaScript. *Introducción a JavaScript.* [En línea] [Citado el: 8 de diciembre de 2014.] http://librosweb.es/javascript/.

13. **Frederick, Shea, Ramsay, Colin y Blades, Steve.** *Learning Ext JS.* BIRMINGHAM : s.n.

14. **Colectivo de Autores.** HTML 5. [En línea] [Citado el: 25 de Enero de 2015.] http://www.w3.org/TR/html5/.

15. W3Schools. [En línea] [Citado el: 3 de Febrero de 2015.] http://www.w3schools.com/about/default.asp.

16. omg. [En línea] [Citado el: 20 de mayo de 2015.] http://www.uml.org/.

17. *Estándar para modelar procesos de negocio.* **García López, Eduardo.** 2013. ISSN 1688-6607.

18. Netbeans.org. [En línea] [Citado el: 3 de Febrero de 2015.] https://netbeans.org/.

19. **Visual Paradigm For UML.** Visual Paradigm For UML. [En línea] [Citado el: 5 de Febrero de 2015.]

20. MySQL. [En línea] [Citado el: 5 de Febrero de 2015.] http://www.mysql.com/.

21. Subversion. [En línea] [Citado el: 5 de Febrero de 2015.] http://subversion.apache.org/.

22. **Larman, Graig.** *UML y Patrones Introducci6n a1 an6lisis y diseiio orientado a objetos.* s.l. : PRENTICE HALL, 1999.

23. [En línea] [Citado el: 20 de mayo de 2015.] http://www.usolab.com/wl/2007/06/experiencias-paralelas-christo-1.php.

24. Scribd. [En línea] [Citado el: 3 de abril de 2015.] http://es.scribd.com/doc/27239031/PATRONES-DE-DISENO.

25. **Montilva, William y Sánchez Bor, Lena.** *Proceso de Modelado del Proceso de Negocios de la Organización.* 2010.

26. qvision technologies. [En línea] [Citado el: 3 de marzo de 2015.] http://www.qvision.us/es/servicios/gestion-de-requerimientos-de-software/levantamiento-de-requisitos.

27. **Sommerville, Ian.** *Ingeniería del Software. Séptima edición.* Madrid : Pearson Addison Wesley, 2005. 84-7829-074-5.

28. **Wesley, Addison.** *RUMBAUGH.* 2000.

29. **Pressman, Roger.** *Ingeniería de Software. Un enfoque práctico.* 2005.

30. ecured. [En línea] [Citado el: 12 de 4 de 2015.] http://www.ecured.cu/index.php/Flujo_de_Trabajo_An%C3%A1lisis_y_Dise%C3%B1o.

31. **Valladolid, Departamento de Informática Universidad.** Departamento de Informática Universidad Valladolid. [En línea] 2012. [Citado el: 20 de marzo de 2015.] http://www.infor.uva.es/~fdiaz/sd/2005_06/doc/SD_TE02_20060305.pdf.

32. **Seth Ladd y Darren, Davidson .** *Expert Spring MVC and Web Flow.* s.l. : Appress, 2006.

33. **Fernández Rivera, Javier.** *Modelo de datos.*

34. Desarrollo de software. [En línea] [Citado el: 15 de 5 de 2015.] https://cufmingsoftware.wordpress.com/estandares-de-diseno/.

35. *Conceptualización del proceso de implementación de software.* **Castillo, Arístides, y otros, y otros.** 3, Mérida, Venezuela : s.n., 2010 , Vol. 31. ISSN 1316-7081.

36. wordpress. [En línea] [Citado el: 3 de 5 de 2015.] https://pruebasdelsoftware.wordpress.com/.

37. *ESTRATEGIAS DE PRUEBA DEL SOFTWARE.* 2011.

ANEXOS

Anexo 1: Cálculo del escalafón de integralidad para el proceso de caracterización

Para el cálculo de escalafón de integralidad se definen una serie de indicadores a los cuales se les otorgarán valores enteros entre 0 y 5 en dependencia de las actividades realizadas por el estudiante a lo largo de la vida universitaria. Se exceptúa solamente el indicador de Formación, cuyo valor es el índice académico.

Indicadores:

- Formación
- Producción
- Investigación
- Extensión
    - Residencia
    - Guardia
    - Deporte
    - Cultura
    - TSU
- FEU – UJC

A los siguientes indicadores se les asigna otros valores, a continuación se detallan:

- Reconocimientos: El valor de este indicador se sumará en función del tipo del reconocimiento que haya obtenido el estudiante.
    - Premio Mella: Se sumarán 8 puntos al total.
    - Título de Oro: Se sumarán 4 puntos al total.
- Agravantes:
    - Medidas disciplinarias: El valor de este indicador se restará al total en función del tipo de medida en la que haya incurrido en el estudiante.
        - ✓ Menos Graves: El valor de este sub-indicador restará 8 puntos al total, por cada medida de este tipo.
        - ✓ Graves: El valor de este sub-indicador restará 12 puntos al total, por cada medida de este tipo.
        - ✓ Muy Graves: El valor de este sub-indicador restará 16 puntos al total, por cada medida de este tipo.
    - Repitente: El valor de este sub-indicador restará 12 puntos al total.
    - Arrastre: El valor de este sub-indicador restará 2 puntos al total, por cada asignatura de arrastre que haya tenido el estudiante en la carrera.

- Mundiales: El valor de este sub-indicador restará 2 puntos al total, por cada mundial que haya llevado el estudiante en la carrera.

Cálculo del valor final para el Ranking de Integralidad

El valor final de Integralidad para la ubicación en el escalafón de integralidad de la universidad se obtiene sumando el valor del indicador Formación (índice académico: índ_acad) y el valor obtenido del resto de los indicadores evaluados, al que denotaremos en adelante como: (valor_otros_ind).

A continuación se detallan los pasos para calcular el valor de los restantes indicadores evaluados (valor_otros_ind):

- Los valores enteros otorgados entre 0 y 5 a los indicadores: Producción, Investigación y FEU – UJC se suman y el valor resultante se multiplica por 5 y a este resultado se le suma, los valores enteros otorgados entre 0 y 5, a los sub - indicadores del indicador Extensión: Deporte, Guardia, Cultura, Residencia y TSU. El resultado máximo que se puede obtener es 100 puntos y el valor mínimo es 0 puntos.
  - [(valor_Producción + valor_Investigación + valor_FEU–UJC)*5] + (valor_Deporte) + (valor_Guardia) + (valor_Cultura) + (valor_Residencia) + (valor_TSU)
- A este valor resultante se le suma, el valor asignado al indicador Reconocimientos, donde el Premio Mella aporta un valor de 8 puntos y Título de Oro aporta un valor de 4 puntos. El mínimo valor de este indicador es 0 y el máximo es 12.
  - (valor_Premio_Mella) + (valor_Título_de_Oro)
- Al valor resultante se le resta el valor del indicador Agravantes, el cual está compuesto por los valores de los sub-indicadores: Medidas disciplinarias, Repitencia, Arrastres y Mundiales.
  - (valor_Agravantes) = (valor_Medidas_disciplinarias) + (valor_Repitencia) + (valor_Arrastres) + (valor_Mundiales)
  - (valor_Medidas_disciplinarias) = [(valor_Menos_Graves)*8] + [(valor_Graves)*12] + [(valor_Muy_Graves)*16]
  - (valor_Repitencia) = 12 (si el estudiante ha repetido un año académico)
  - [(valor_Arrastres) = (Cantidad_asignaturas_arrastres_carrera)*2
  - [(valor_Mundiales) = (Cantidad_de_mundiales_carrera)*2
- El valor resultante se divide entre 100, y se obtiene el valor de los restantes indicadores: (valor_otros_ind)

Anexo 2: Modelo de entrevista

MODELO DE ENTREVISTA

Nombre(s):

Apellidos:

Cargo que Ocupa:

PREGUNTAS:

1. ¿En qué consiste el proceso?
2. ¿Cuáles son las características fundamentales del proceso?
3. ¿Existe algún mecanismo para llevar a cabo el proceso?
4. ¿Qué beneficios brinda el proceso?
5. ¿Existen herramientas que ayuden a la ejecución del proceso?
6. ¿Qué limitaciones presenta el proceso actualmente?
7. ¿Qué desearía aportar al proceso para hacer el trabajo más eficiente?

Anexo 3: Interfaces del módulo Escalafón de integralidad

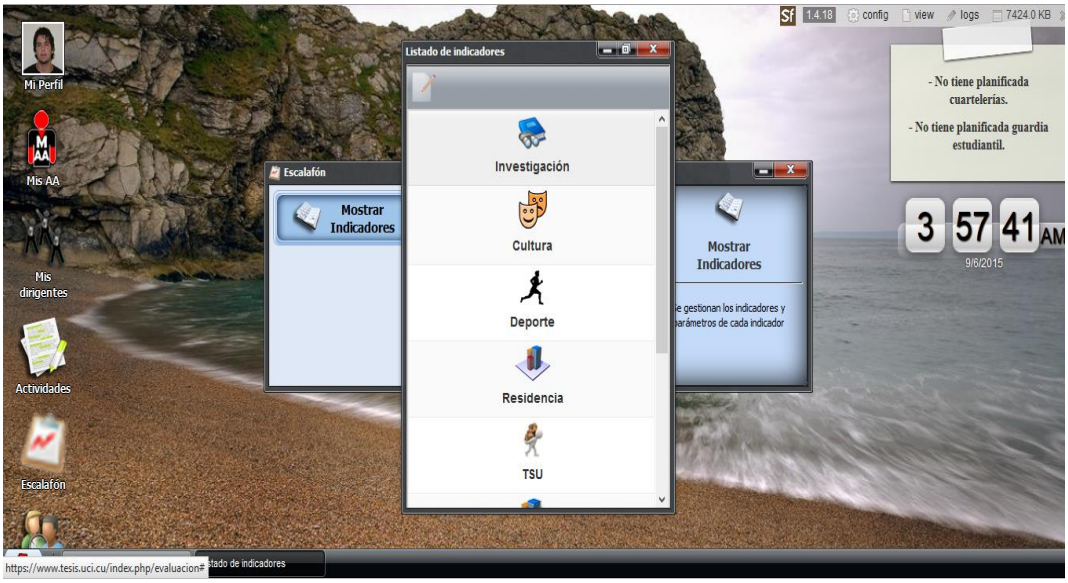
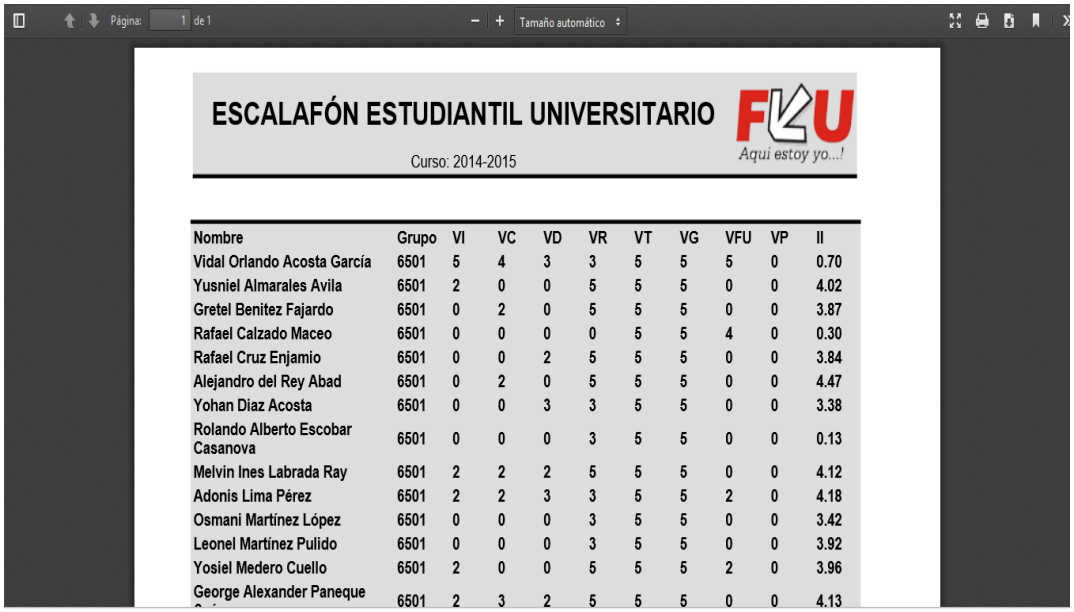


Figura 17. Interfaz del módulo Escalafón de integralidad *Listado de indicadores*



ESCALAFÓN ESTUDIANTIL UNIVERSITARIO

Curso: 2014-2015

Nombre	Grupo	VI	VC	VD	VR	VT	VG	VFU	VP	II
Vidal Orlando Acosta García	6501	5	4	3	3	5	5	5	0	0.70
Yusniel Almarales Avila	6501	2	0	0	5	5	5	0	0	4.02
Gretel Benitez Fajardo	6501	0	2	0	5	5	5	0	0	3.87
Rafael Calzado Maceo	6501	0	0	0	0	5	5	4	0	0.30
Rafael Cruz Enjamio	6501	0	0	2	5	5	5	0	0	3.84
Alejandro del Rey Abad	6501	0	2	0	5	5	5	0	0	4.47
Yohan Diaz Acosta	6501	0	0	3	3	5	5	0	0	3.38
Rolando Alberto Escobar Casanova	6501	0	0	0	3	5	5	0	0	0.13
Melvin Ines Labrada Ray	6501	2	2	2	5	5	5	0	0	4.12
Adonis Lima Pérez	6501	2	2	3	3	5	5	2	0	4.18
Osmani Martínez López	6501	0	0	0	3	5	5	0	0	3.42
Leonel Martínez Pulido	6501	0	0	0	3	5	5	0	0	3.92
Yosiel Medero Cuello	6501	2	0	0	5	5	5	2	0	3.96
George Alexander Paneque	6501	2	3	2	5	5	5	0	0	4.13

Figura 18. Interfaz del módulo Escalafón de integralidad *Escalafón.pdf*

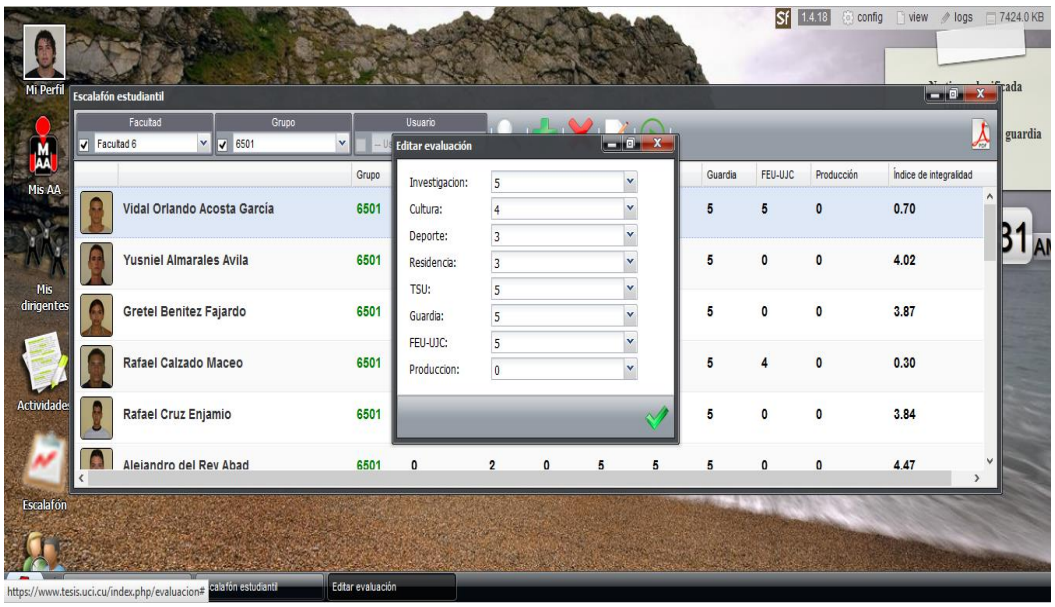
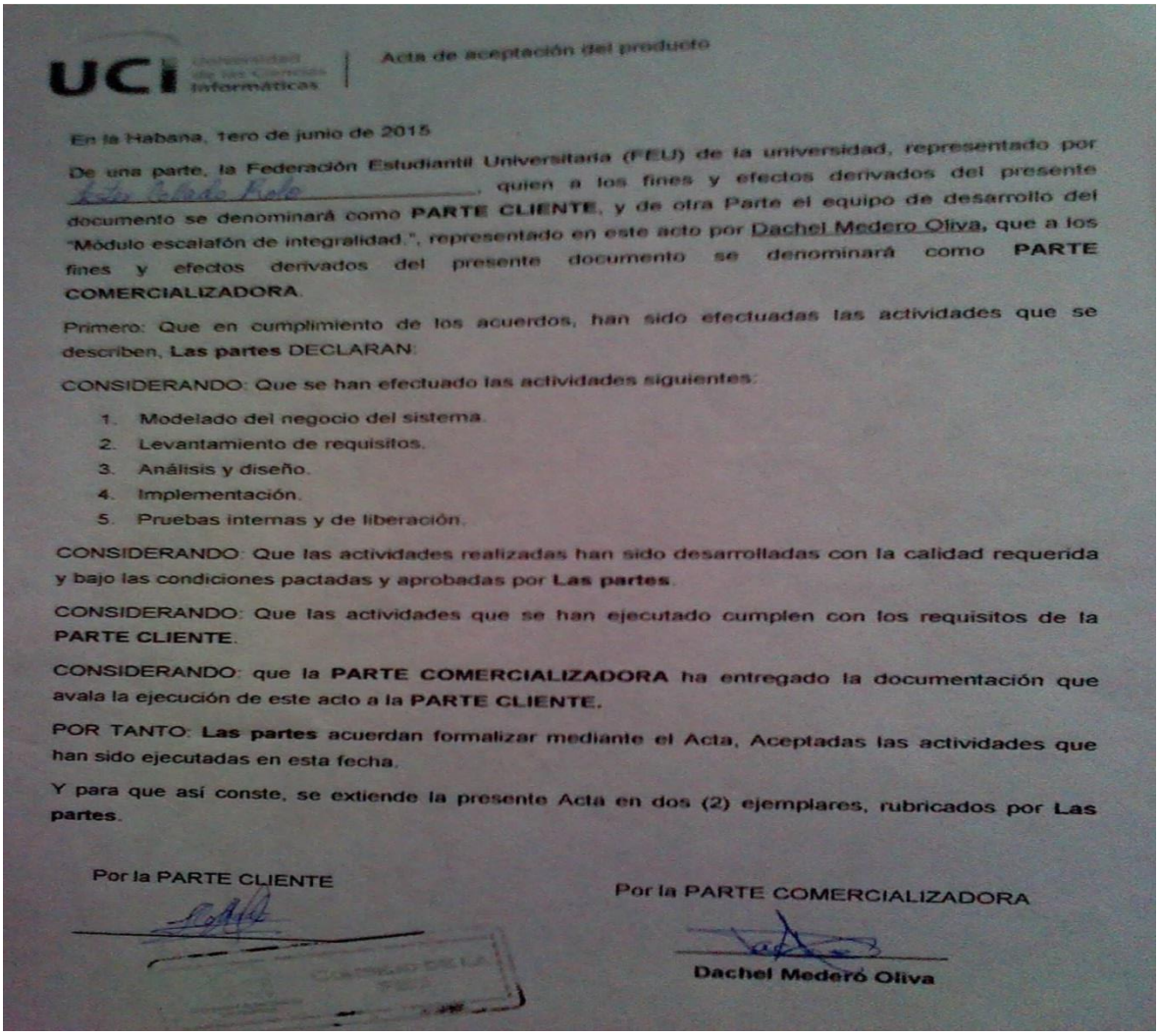


Figura 19. Interfaz del módulo Escalafón de integralidad Editar evaluación

Anexo 4: Carta de aceptación del cliente



UCI Universidad de las Ciencias Informáticas | Acta de aceptación del producto

En la Habana, 1ero de junio de 2015

De una parte, la Federación Estudiantil Universitaria (FEU) de la universidad, representado por _____, quien a los fines y efectos derivados del presente documento se denominará como **PARTE CLIENTE**, y de otra Parte el equipo de desarrollo del "Módulo escalafón de integralidad.", representado en este acto por Dachel Medero Oliva, que a los fines y efectos derivados del presente documento se denominará como **PARTE COMERCIALIZADORA**.

Primero: Que en cumplimiento de los acuerdos, han sido efectuadas las actividades que se describen, **Las partes** DECLARAN:

CONSIDERANDO: Que se han efectuado las actividades siguientes:

1. Modelado del negocio del sistema.
2. Levantamiento de requisitos.
3. Análisis y diseño.
4. Implementación.
5. Pruebas internas y de liberación.

CONSIDERANDO: Que las actividades realizadas han sido desarrolladas con la calidad requerida y bajo las condiciones pactadas y aprobadas por **Las partes**.

CONSIDERANDO: Que las actividades que se han ejecutado cumplen con los requisitos de la **PARTE CLIENTE**.

CONSIDERANDO: que la **PARTE COMERCIALIZADORA** ha entregado la documentación que avala la ejecución de este acto a la **PARTE CLIENTE.**

POR TANTO: **Las partes** acuerdan formalizar mediante el Acta, Aceptadas las actividades que han sido ejecutadas en esta fecha.

Y para que así conste, se extiende la presente Acta en dos (2) ejemplares, rubricados por **Las partes**.

Por la PARTE CLIENTE

Por la PARTE COMERCIALIZADORA

Dachel Mederó Oliva

Figura 20. Carta de aceptación del cliente