**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**

**Facultad 3**


# *Sistema Informático de Gestión para las estrategias educativas de la Facultad 3.*

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

**Autor:** Miguel Piñeda Londres

**Tutor:** Ing. Mailen Edith Escobar Pompa

La Habana. Junio, 2015

## Declaración de autoría

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

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

_____

Miguel Piñeda Londres

Autor

_____

Ing. Mailen Edith Escobar Pompa

Tutor

## Agradecimientos

*A mis padres, por todo el apoyo incondicional que me han brindado durante el transcurso de la carrera y en todo momento. Gracias a ustedes me he convertido en mejor persona, profesionalmente y de grandes valores que me permiten incorporarme positivamente a la sociedad.*

*A mi hermana, por estar presente siempre cuando he necesitado su ayuda y por demostrarme que puedo contar con ella en todo momento.*

*A mi tía Tuti, por ser mi segunda madre y estar siempre atenta a las cosas que me suceden.*

*A mi familia en general que siempre se preocupa por mí, gracias a todos.*

*A mi tutora Mailen, mi compañera de guerrilla, por toda la ayuda y dedicación en la tesis.*

*Al teniente coronel Alberto Bouli, por ser un gran amigo y compañero de lucha durante toda la carrera.*

*A mi profesora de música Tamara, por demostrarme que podría ser un buen cantante y hoy se ven los frutos del trabajo de los dos.*

*A mis antiguos compañeros de aula y a los ahora presentes.*

*A mi compañero de cuarto Macías, por ser un gran amigo y profesor de programación, a él le debo los conocimientos que pude adquirir durante el desarrollo de la tesis y a su novia Dayana por soportarme todo este curso.*

*A la profesora Dariela, por ser una gran amiga, por los consejos y la ayuda que me brindó incondicionalmente.*

*A la Universidad de las Ciencias Informáticas le agradezco las oportunidades de conocimiento y preparación que supe aprovechar. Hoy, no solo me gradúo como ingeniero en ciencias informáticas. Gracias a la universidad puedo decir, que me siento preparado política e ideológicamente para enfrentar los problemas que presenta la sociedad de nuestros días.*

## Dedicatoria

Les dedico este trabajo a mis padres que además, es el fruto de sus enseñanzas. De manera muy especial a mi madre, que sin ella sería muy difícil hacer realidad todo lo que he logrado hoy.

## Resumen

Las estrategias educativas en la Universidad de las Ciencias Informáticas tienen la misión de coordinar acciones con el objetivo de formar ingenieros altamente calificados, con conocimientos técnicos que le permitan incidir de manera positiva en la sociedad. Actualmente el proceso de gestión de las estrategias educativas en la Facultad 3 se realiza de forma manual, lo que dificulta el control y seguimiento del mismo. Para darle solución a este problema la presente investigación se planteó como objetivo el desarrollo de un software que realice la gestión de este proceso. Para su cumplimiento se realizó un estudio detallado del proceso a informatizar, se definieron las metodologías, tecnologías y herramientas necesarias para desarrollar la solución, se describe la solución propuesta y la validación de la misma. Se obtuvo como resultado una aplicación informática que permite la creación, gestión, seguimiento y control de las estrategias educativas en la Facultad 3.


**Palabras claves**: estrategias educativas, gestión de información, información estudiantil.


## Abstract

The educational strategies in the University of Informatics Sciences have the mission of coordinating actions with the objective of training highly qualified engineers, with technical knowledge that will allow them to influence society in a positive manner. Currently, the management process of the educational strategies in Faculty 3 comes about in a manual form, which makes its control and tracing more difficult. To provide a solution to this problem, the present investigation implements itself as an objective in the development of a software that fulfills the management of this process. For its fulfillment a detailed study was made about the computerizing process. The necessary methodologies, technologies, and tools were defined to develop a solution. The proposed solution is described, as well as its validation. The obtained result was an informatics application that allows the creation, management, tracing, and control of the educational strategies in Faculty 3.


**Key words:** educational strategies, information management, student information.

Tabla de contenidos

## Introducción

La actividad educativa es una de las más complejas del ser humano. Es por ello que la operatividad de distintos modelos educativos, dentro de los paradigmas imperantes del momento, empleando distintos recursos didácticos que produzcan aprendizaje significativo y activo del estudiante, es uno de los temas de investigación que más atención se le brinda en la actualidad. Dentro de estos modelos se trazan estrategias educativas, las cuales deben estar bien planificadas para llevar a cabo un mejor control y organización de las actividades que se realizan en los centros educacionales. Estas estrategias funcionan sobre la base de la Gestión de la información. Pueden ser monitoreadas y controladas, estar sujetas a cambios después de ser analizadas por los especialistas y ayudan al proceso de enseñanza y aprendizaje debido al conocimiento que se genera de cada estudiante como individuo y como grupo en el momento que se integran para las actividades docentes y de extensión.

Las estrategias educativas en la Universidad de las Ciencias Informáticas (UCI) tienen la misión de coordinar acciones con el objetivo de formar ingenieros altamente calificados, con vastos conocimientos técnicos, que le permitan incidir de manera positiva en la sociedad. Poseedores de un pensamiento crítico y una amplia cultura general integral, comprometidos con la Revolución y sus valores (Peña, 2014).

En la Facultad 3 de la Universidad de las Ciencias Informáticas el proceso de gestión de las estrategias educativas presenta los siguientes problemas:

- Las estrategias educativas se realizan mediante un formato establecido por la universidad y son guardadas por el profesor guía de la brigada, no existiendo un lugar centralizado donde, tanto la brigada como el resto del colectivo pedagógico, puedan consultar los datos referentes a las mismas.
- La notificación de las actividades es realizada por el profesor guía y en algunos casos por los responsables de la actividad, lo que conlleva a que muchas veces la actividad no se realice o se haga fuera de la fecha planificada debido a que no se convocó a la misma en tiempo.
- La información que generan las estrategias educativas para cada nivel (brigada, año, facultad), son gestionadas manualmente por los responsables (profesores principales del año y profesores guías), dificultándose la creación de reportes que permitan dar una vista global o parcial del proceso en toda la facultad, afectándose además, el control y seguimiento del proceso.

A raíz de la problemática antes expuesta, se plantea el siguiente **problema a resolver:** La forma en que actualmente se gestionan las estrategias educativas de la Facultad 3 incide negativamente en el adecuado control y seguimiento de las mismas.

El problema planteado se enmarca en el **objeto de estudio**: Gestión de las estrategias educativas en la Universidad de las Ciencias Informáticas.

Para darle solución al problema identificado se define el siguiente **objetivo general:** Desarrollar un sistema informático que permita gestionar las estrategias educativas de la Facultad 3 mejorando el control y seguimiento de las mismas.

En la investigación se concibe como **campo de acción**: Gestión de las estrategias educativas en la Facultad 3.

Para dar cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos**:

- Realizar el marco teórico referente al objeto de estudio de la investigación sentando las bases para el desarrollo del sistema propuesto.
- Desarrollar el sistema informático propuesto de acuerdo a la metodología y tecnologías seleccionadas.
- Validar el correcto funcionamiento de la aplicación haciendo uso de las pruebas internas, de liberación y aceptación, así como las variables propuestas en la investigación.

Se plantea como **idea a defender:** El desarrollo de un sistema informático para la gestión de las estrategias educativas de la Facultad 3 contribuirá a elevar el control y seguimiento de las mismas.

**Tareas de la investigación:**

- Análisis de los principales conceptos asociados a las estrategias educativas.
- Análisis de los sistemas informáticos de gestión asociados a los procesos de creación y control de las estrategias educativas en la UCI.
- Análisis de las metodologías y tecnologías para el desarrollo de software identificando las adecuadas para realizar el sistema propuesto.
- Modelación del negocio.
- Análisis de los requisitos.

- Validación de los requisitos identificados para el sistema.
- Diseño del sistema informático.
- Validación del diseño realizado.
- Implementación de las funcionalidades del sistema informático haciendo uso de las tecnologías seleccionadas.
- Validación del sistema informático mediante pruebas de software.
- Validación de las variables de la investigación.

Para el desarrollo del presente trabajo se hacen uso de los siguientes **métodos científicos de investigación**:

**Métodos teóricos:**

- **Analítico–sintético:** este método se emplea para el análisis de la bibliografía relacionada con el tema, de los sistemas homólogos existentes; el estudio de las características, ventajas y desventajas de dichos sistemas, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.

- **Modelación:** este método permite la realización de los artefactos que propone la metodología, para comprender mejor el proceso desde el punto de vista informático.

**Métodos empíricos:**

- **Entrevista:** este método permite conocer las causas y qué soluciones se le dan actualmente a los problemas sobre el registro de información para la caracterización de los estudiantes y si existen normas que rigen dicho registro.

El documento de tesis consta de una introducción, tres capítulos, las conclusiones, recomendaciones, las bibliografías utilizadas y el cuerpo de anexos. A continuación se describe la estructura capitular del documento.

**Capítulo 1. Fundamentación teórica**

En este capítulo se lleva a cabo un estudio para el entendimiento del proceso de gestión de las estrategias educativas en la Facultad 3. Se realiza un análisis de algunos de los principales sistemas que gestionan información asociada a los procesos de creación y control de las estrategias educativas en la UCI, con el

objetivo de identificar sus principales características. Además, se justifica el empleo de cada una de las herramientas, metodologías, lenguajes de desarrollo, marcos de trabajo y tecnologías empleadas en la investigación.

**Capítulo 2. Modelado del negocio, requisitos y diseño**

En este capítulo se describe el análisis y diseño de la solución. Se modela el negocio y se definen los requisitos funcionales y no funcionales del sistema a desarrollar. Además, se describe la arquitectura, los patrones de diseño y se muestran los artefactos definidos para las disciplinas modelado del negocio, requisitos, análisis y diseño, las cuales son propuestas por la metodología escogida para el desarrollo de la investigación.

**Capítulo 3. Implementación y prueba**

En este capítulo se presentan los estándares de diseño y codificación a seguir para la implementación del sistema. Se muestra el diagrama de despliegue y se definen las pruebas de caja negra y caja blanca realizadas al software en aras de detectar errores o defectos relacionados con su funcionalidad.

# Capítulo1. Fundamentación teórica.

## Introducción

El presente capítulo contiene los conceptos asociados a las estrategias educativas, así como las características de este proceso. Presenta un estudio de otros sistemas que gestionan información asociada a los procesos de creación y control de las estrategias educativas en la UCI, con el objetivo de identificar sus principales características. Además, se aborda la selección de cada una de las herramientas, metodologías, lenguajes de desarrollo, marcos de trabajo, gestor de base de datos y servidor web utilizados para el análisis, diseño e implementación del proceso de gestión de las estrategias educativas en la Facultad 3.

## 1.1. Características de las estrategias educativas

Los estudios realizados y la práctica educativa en el proceso de formación integral de los estudiantes universitarios han demostrado que en el funcionamiento de la estructura organizativa de la universidad, las mayores insuficiencias se concentran en el eslabón de base. Entre las razones que explican una más débil articulación de estas relaciones en la base se encuentran: Falta de sistematicidad en la comunicación y el diálogo en la base, débil integración de las estructuras institucionales con todos los factores para canalizar las responsabilidades en el proceso de formación y el resto de las actividades que se generan en la vida universitaria, falta de entrenamiento en la elaboración de una estrategia compartida con los estudiantes y sus organizaciones, que conduzcan al cumplimiento de los objetivos que se deben alcanzar por la institución en este nivel (Ministerio de Educación Superior, 2014).

Estas insuficiencias indican la necesidad de trabajar de forma intencionada en el eslabón de base y en especial en la estrategia educativa de la comunidad universitaria en el año académico.

La estrategia educativa incluye dos aspectos claves (Ministerio de Educación Superior, 2014):

- Las actividades metodológicas del claustro de profesores en el año académico.
- Las dimensiones educativas en el proceso de formación integral de los estudiantes.

Las actividades metodológicas del claustro de profesores y las dimensiones educativas en el proceso de formación integral de los estudiantes se elaboran a partir de los objetivos del año académico.

Por tanto, las estrategias educativas hacen referencia a un conjunto de actividades, en el entorno educativo, diseñadas para lograr de forma eficaz y eficiente la consecución de los objetivos educativos esperados. Desde el enfoque constructivista esto consistirá en el desarrollo de competencias por parte de los estudiantes. (Revista Pedagogía Universitaria, 2007)

### 1.1.1. Dimensiones educativas en el proceso de formación.

La caracterización estudiantil constituye la vía para profundizar en el conocimiento de las cualidades personales de los estudiantes y las características principales del grupo. Tiene como objetivo fundamental que el estudiante establezca un grado de compromiso con el desarrollo de cualidades afines a los requerimientos y exigencias de su futura profesión y está orientada a profundizar en las cualidades intelectuales, la motivación, las cualidades morales, los hábitos y habilidades para el desempeño profesional y el desarrollo sociopolítico (Benítez, 2000).

Las dimensiones educativas en el proceso de formación de los estudiantes se organizan sobre la base de una caracterización de los estudiantes. Los resultados de la caracterización constituyen un diagnóstico del grupo de estudiantes y de cada estudiante en particular (Benítez, 2000).

Uno de los aspectos claves que debe seguir el colectivo de profesores del año académico a partir de las nuevas exigencias que emanan de la estrategia educativa lo constituye su contribución a los valores que debe caracterizar al egresado de la carrera. Los resultados del diagnóstico contribuyen a una mayor precisión de estos valores y de los modos de actuación del grupo de estudiantes en el año académico (Benítez, 2000).

Por consiguiente, después de concluido el diagnóstico, se definen los valores y modos de actuación que se deben alcanzar en el año académico, así como las necesidades educativas que emanan del trabajo de la universidad, sus facultades y el territorio, de esta forma, se adecuan y enriquecen los objetivos educativos e instructivos del año académico. Dichos objetivos sirven de base para planificar el contenido de las dimensiones: Curricular, extensión universitaria y actividades sociopolíticas. Para la evaluación del cumplimiento de las dimensiones se elaboran los criterios de medida y las acciones que se deben realizar (Benítez, 2000).

La estrategia del año académico se elabora y se discute con el grupo de estudiantes para enriquecerla e involucrar protagónicamente a los educandos en su cumplimiento y permite de esta manera, que tanto el

colectivo pedagógico como los estudiantes, incorporen una cultura de trabajo cooperada en función de lo objetivos compartidos. De esta forma, los estudiantes quedan incorporados en el trabajo por una formación académica y una preparación integral más completa para alcanzar mejor desarrollo de su personalidad en la educación superior (Benítez, 2000).

### 1.1.2. Acciones educativas individuales en la estrategia educativa.

Las acciones educativas individuales en la estrategia educativa están dirigidas al fortalecimiento de la atención personalizada al estudiante y contienen las principales tareas que, en el plano de la dimensión curricular, la extensión universitaria y las actividades sociopolíticas, realiza el educando, orientadas al proceso de transformación de su personalidad en aras de alcanzar una cultura general integral (Benítez, 2000).

Los aspectos generales para la elaboración de las acciones educativas individuales son (Benítez, 2000):

- Contiene acciones que reflejan tareas de impacto en la universidad, en el territorio y en los marcos de la sociedad.
- Incluye acciones concretas que realiza el estudiante en la residencia estudiantil como un espacio legítimo para su formación.
- Compromiso individual de cada estudiante de los resultados académicos que debe alcanzar en cada asignatura en el semestre.
- Compromiso de su participación en exámenes de premios.
- Participar en las estrategias curriculares de preparación jurídica, preparación económica, preparación para la defensa, informatización, medio ambiente, Historia de Cuba e Idioma.
- Participación en la investigación con resultados presentados en la Jornada Científica Estudiantil.
- Participación en las diferentes manifestaciones culturales y deportivas en los festivales de aficionados y en los juegos deportivos.
- Participación como promotor cultural en el interior de la universidad y en la comunidad (proyectos comunitarios).
- Participación protagónica en las acciones de prevención y combate frontal contra las manifestaciones de fraude, indisciplinas, ilegalidades, corrupción y consumo de drogas.
- Participación y cumplimiento de la guardia estudiantil.
- Participación en las movilizaciones políticas convocadas por las organizaciones y por la institución.
- Participación y resultados alcanzados en la preparación política ideológica que incluye el dominio de la actualidad nacional e internacional.

- Responsabilidades individuales en la realización de las acciones que se convocan en el grupo de estudiantes.

Las acciones educativas individuales deben tener como base para su elaboración los resultados del diagnóstico. De esta forma, las acciones planificadas, además de reflejar los objetivos educativos e instructivos, los valores y modos de actuación que se deben alcanzar, expresan aspiraciones, motivaciones y las necesidades educativas de cada estudiante, las cuales se corresponden con los objetivos educativos del año académico.

### 1.1.3. Proceso de evaluación de los estudiantes.

El proceso de evaluación es una de las tareas más complejas en la educación superior. Para alcanzar una efectiva evaluación de la formación del estudiante, se requiere valorar en qué medida se cumple la estrategia educativa en general y en particular lo que a cada estudiante corresponde (Benítez, 2000).

En el proceso de evaluación de los estudiantes, se sigue al educando y también al grupo, ya que la evaluación se realiza a cada estudiante y al grupo en su conjunto al concluir un determinado período y los resultados alcanzados se comparan con los objetivos previstos para ese grupo y para cada estudiante en particular. Este método de trabajo refuerza en el proceso de evaluación la atención personalizada a los estudiantes (Benítez, 2000).

### 1.1.4. Gestión de las estrategias educativas en la Facultad 3.

La gestión de la información es el proceso mediatizado por un conjunto de actividades que permiten la obtención de información, lo más pertinente, relevante y económica posible, para ser usada en el desarrollo y el éxito de una organización. Genera nuevos conocimientos que apoyan la toma de decisiones en las organizaciones (Barroso, 2009).

La gestión de la información que se genera en la confección y seguimiento de las estrategias educativas en la Facultad 3 se realiza de forma manual. Los profesores principales conforman la estrategia educativa para el año. La misma es adaptada y discutida en cada brigada, desglosándose las actividades a realizar, por cada una de las dimensiones, política, curricular y extracurricular, en el grupo, además de la responsabilidad de los estudiantes en la participación de estas actividades. Estos resultados quedan plasmados en documentos digitales que son controlados por los profesores guías a nivel de brigada y por los profesores principales a nivel de año, lo que dificulta el control y disponibilidad de la información que genera el proceso.

## 1.2. Análisis de sistemas homólogos

A continuación se realiza un análisis valorativo, de algunos sistemas que gestionan información asociada a los procesos de creación y control de las estrategias educativas en la UCI. Para el análisis se utilizaron 9 indicadores, que deben estar presentes en la posible solución, en base a darle respuesta a la problemática planteada y a algunas características que debe presentar dicha solución. Además, se revisan las funcionalidades y características que puedan ayudar a la elaboración de una posible propuesta de solución.

**Sistema de Caracterización Integral del Graduado (SCIG)**

El sistema consiste en un sitio web desarrollado por el Centro de Innovación de la Calidad de Educación (CICE) en la UCI, con el objetivo de automatizar el proceso de registro para contribuir a la caracterización integral de los estudiantes de quinto año (Rodríguez, 2012).

**Entre las actividades que realiza el sistema se encuentran** (Rodríguez, 2012)**:**

- La asignación de la persona encargada de realizar la caracterización (profesor guía o tutor supervisor evaluador), según el ciclo de instrucción en que se encuentre el estudiante.
- La generación de la planilla de caracterización integral a partir del previo registro de los indicadores a medir en la caracterización. Con la opción de exportar la planilla en formato PDF para su impresión, envío por correo o almacenamiento en un archivo.
- La generación de la evaluación final en la caracterización (bien, regular o mal) según la trayectoria del estudiante en el curso.
- La gestión de la información en forma de reportes dinámicos que muestran la trayectoria de los estudiantes durante la carrera, para el monitoreo de su formación por parte de los directivos docentes.

**Algunas de las características que presenta el sistema** (Rodríguez, 2012)**:**

*Seguridad:*

- La seguridad está a nivel de roles, con el fin de mantener la integridad de los datos en función del acceso de cada uno de ellos.

- Políticas de seguridad por usuarios y roles: el sistema cuenta con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo, en dependencia del nivel de autorización que presente un usuario determinado.

*Software:*

- PC Cliente: sistema operativo GNU/Linux.
- PC servidor de base de datos: el servidor debe contar con el sistema operativo Linux Ubuntu 11.10 o superior y el sistema gestor de base de datos PostgreSQL 8.4 o superior.
- PC servidor web: el sistema debe estar montado sobre Apache 2.1 o superior

**Sistema para la informatización del proceso de Gestión de la caracterización estudiantil en la Facultad 3 (SIPGCE)**

El sistema consiste en un sitio web resultado de una tesis de diploma (Barbán, 2014), con el objetivo de automatizar el proceso de gestión de las caracterizaciones de los estudiantes de la Facultad 3, además de detectar posibles problemas que puedan surgir dentro de los grupos.

**Entre las actividades que realiza el sistema se encuentran** (Barbán, 2014)**:**

- La gestión del colectivo pedagógico de los años, que permite asignar un profesor guía a cada grupo.
- La gestión de la caracterización de los estudiantes, ingresando los datos docentes, extensionistas y personales de cada uno.
- La generación de la planilla de caracterización de los estudiantes, con la opción de exportar a formato PDF para su impresión, envío por correo o almacenamiento en un archivo.
- La generación de reportes, para obtener las caracterizaciones de los estudiantes, de acuerdo a diferentes criterios de búsquedas relacionados con los datos que poseen los estudiantes.

**Algunas de las características que presenta el sistema** (Barbán, 2014)**:**

*Seguridad*

- El sistema realiza la autenticación, a través del servicio de directorios que se encuentra en la UCI, usando el Protocolo Ligero de Acceso a Directorios (LDAP).

- La seguridad está a nivel de roles, con el fin de mantener la integridad de los datos en función del acceso de cada uno de ellos.

*Software:*

- PC servidor de base de datos: el servidor debe contar con el sistema gestor de base de datos PostgreSQL 9.2 o superior.
- PC servidor web: el sistema debe estar montado sobre Apache 2.2.6 o superior.

**Análisis de los sistemas estudiados**

En la

Tabla **1** se realiza una comparación de los sistemas expuestos anteriormente de acuerdo a indicadores con los que se debe cumplir para darle solución a la problemática planteada.

Tabla 1 Resumen del análisis de sistemas homólogos. Fuente: elaboración propia.

| **Indicadores** | **SIPGCE** | **SCIG** |
|---|---|---|
| Gestión de la caracterización del colectivo pedagógico | No | No |
| Gestión de la caracterización de los estudiantes | Sí | No |
| Gestión de las acciones de la Estrategia Educativa por grupo | Sí | No |
| Gestión de las acciones de la Estrategia Educativa por año | No | No |
| Gestión de las actividades metodológicas del claustro de profesores por año | No | No |
| Basado en la tecnologías libres | Sí | Sí |
| Manejo efectivo de la seguridad de la información | Sí | Sí |
| Envío de notificaciones a cada involucrado en las actividades próximas a realizarse | No | No |
| Basado en tecnologías web | Sí | Sí |

La investigación realizada arrojó como resultado que los dos sistemas estudiados no realizan algunas de las actividades necesarias para llevar a cabo la gestión de las estrategias educativas en la Facultad 3, las cuáles son: gestión de la caracterización del colectivo pedagógico, gestión de las actividades metodológicas del claustro de profesores por año, gestión de las acciones por año y envío de notificaciones a cada involucrado en las actividades próximas a realizarse. Aunque estos sistemas cubran aspectos comunes, gestionan y planifican dependiendo de sus características propias; no obstante, pueden ser de gran utilidad algunas de las funcionalidades que manejan estos sistemas como la caracterización de los estudiantes. Estos elementos evidencian la necesidad de elaborar un sistema informático que le dé solución a las problemáticas planteadas en el proceso de gestión de las estrategias educativas en la facultad 3.

## 1.3. Metodología de desarrollo de software

Según (Gacitúa Bustos, 2003) una metodología impone un proceso de forma disciplinada sobre el desarrollo de software con el objetivo de hacerlo más predecible y eficiente. Una metodología define una representación que permite facilitar la manipulación de modelos y la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema. Para la elaboración de la solución propuesta se empleará la metodología de desarrollo para la actividad productiva de la UCI, basado en buenas prácticas y principios de la metodología Proceso Unificado Ágil (AUP).

### 1.3.1. AUP

El Proceso Unificado Ágil de Scott Ambler AUP (*Agile Unified Process, por sus siglas en inglés*) es una versión simplificada del Proceso Unificado de Desarrollo (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (Ambysoft Inc, 2006).

AUP propone cuatro fases (Inicio, Elaboración, Construcción y Transición) y se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, que se nombra Ejecución y se agrega una fase de Cierre (Sánchez, 2014).

A continuación se realiza una descripción de cada una de las fases de la metodología de desarrollo para la actividad productiva de la UCI, para una mayor comprensión de las mismas.

1. **Inicio:** durante la fase de Inicio se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto (Sánchez, 2014).
2. **Ejecución:** esta fase agrupa las restantes 3 fases de AUP (Elaboración, Construcción y Transición). En ella 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, se 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 entregados al cliente, además, en la transición se capacita a los usuarios finales sobre la utilización del software (Sánchez, 2014).
3. **Cierre:** durante esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (Sánchez, 2014).

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno). Se decidió para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional (Sánchez, 2014).

A continuación se realiza una descripción de cada una de las disciplinas de la metodología de desarrollo para la actividad productiva de la UCI. Además, se hace referencia a los artefactos que serán generados por cada una de estas, para una mayor comprensión de las mismas.

1. **Modelado de negocio:** esta disciplina está destinada 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 que se va a desarrollar cumplirá su propósito (Sánchez, 2014). En esta disciplina se generarán como salidas los artefactos: Modelo conceptual, Reglas de negocio, Diagrama de proceso de negocio y Descripción de procesos de negocio.
2. **Requisitos:** destinada a desarrollar un modelo del sistema que se va a construir, también comprende la administración y gestión de los requisitos funcionales y no funcionales del producto

(Sánchez, 2014). En esta disciplina se generarán como salidas los artefactos: Descripción de requisitos, Especificación de requisitos y Salidas del sistema. Una vez concluida dicha disciplina y antes de comenzar la otra se debe realizar la validación de los requisitos, como buena práctica que le proporciona al ingeniero de software una medida de la calidad con que realizó las tareas de la disciplina. Para la validación de los requisitos se emplearán las siguientes técnicas: prototipado de interfaces de usuario y revisión técnica formal.

3. **Análisis y diseño:** en esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa y una descripción que sea fácil de mantener. Se modela el sistema, su forma y arquitectura para que soporte todos los requisitos, incluyendo los requisitos no funcionales (Sánchez, 2014). En esta disciplina se generarán como salidas los artefactos: Diagrama de clases del diseño, Diagrama de paquetes y Modelo de datos. Una vez concluida dicha disciplina y antes de comenzar la otra se debe realizar la validación del diseño. Para medir el diseño se emplearán las métricas: tamaño operacional de clases y relaciones entre clases, inspiradas en el estudio de la calidad del diseño orientado a objetos.
4. **Implementación:** se construye el sistema a partir de los resultados del Análisis y diseño (Sánchez, 2014). En esta disciplina se generará como salida el artefacto Diagrama de despliegue.
5. **Pruebas internas:** se verifica el resultado de la implementación. Se desarrollan artefactos de prueba como: Diseños de casos de prueba, Listas de chequeo y si es posible Componentes de prueba ejecutables para automatizar las pruebas (Sánchez, 2014). En esta disciplina se generará como salida el artefacto Diseño de casos de pruebas.
6. **Pruebas de liberación:** son pruebas diseñadas y ejecutadas por una entidad externa certificadora de la calidad, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación (Sánchez, 2014). Para la realización de estas pruebas, se utilizará como entrada el artefacto Diseño de casos de pruebas y serán ejecutadas por el Centro de Informatización de Entidades (CEIGE), obteniéndose el Acta de liberación del producto que se va a realizar.
7. **Pruebas de Aceptación:** es la prueba final que se realiza antes del despliegue del sistema con el objetivo de verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales fue construido (Sánchez, 2014). En esta disciplina se generará el Acta de aceptación por parte de los clientes: Vicedecana de Formación, asesora docente metodológica y profesores principales de los diferentes años académicos en la Facultad 3.

8. **Despliegue:** consiste en la instalación, configuración, adecuación, puesta en marcha de soluciones informáticas y entrenamiento al personal del cliente (Sánchez, 2014).

Todas las disciplinas antes definidas (desde Modelado de negocio hasta Despliegue) se desarrollan en la fase de Ejecución, de ahí que en la misma se realicen iteraciones y se obtengan resultados incrementales (Sánchez, 2014). En la Figura 1 se muestran las fases de la metodología de desarrollo para la actividad productiva de la UCI.

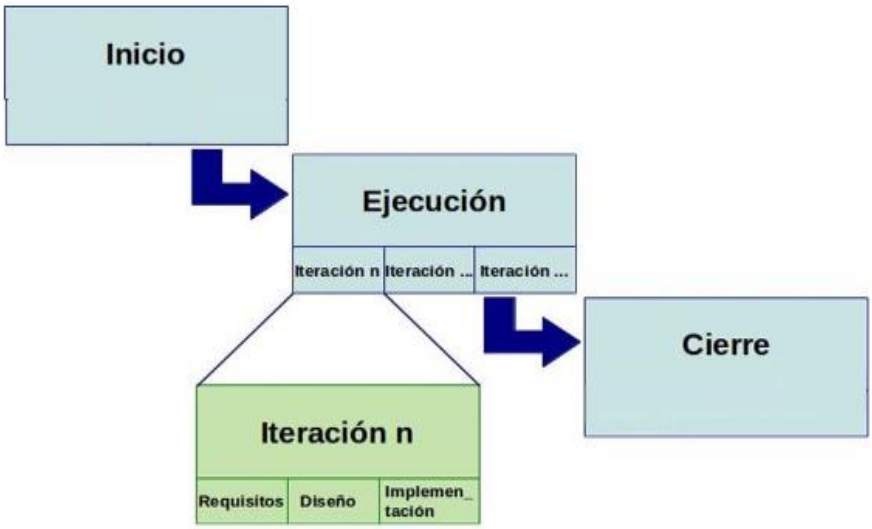



Figura 1 Fases de la metodología de desarrollo para la actividad productiva de la UCI. Fuente: **(Sánchez, 2014)**.

### 1.4. Lenguaje Unificado de Modelado

UML (*Unified Modeling Language, por sus siglas en inglés*) como notación orientada a objetos cuenta con una notación estándar y semánticas esenciales para el modelado de sistemas, se empleará con el fin de modelar, especificar y documentar un sistema de software, de un modo estándar incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema. UML implementa un lenguaje de modelado común para todos los desarrolladores por lo que se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje de programación utilizado para el desarrollo (Larman, 1999).

## 1.5. Técnicas para la captura de requisitos

La captura de requisitos es el proceso mediante el cual los interesados en un sistema de software descubren, revelan, articulan y entienden sus requisitos. En muchos casos, se requiere tiempo para llegar a especificar claramente lo que el interesado espera de la aplicación de software, por lo que se hace necesario por parte de los analistas el empleo de técnicas que permitan establecer una buena comunicación con los interesados del producto y así lograr la satisfacción del cliente. (M. GERBER, 2001)

A continuación se enuncian las técnicas utilizadas para la captura de requisitos.

### Entrevistas

Es la más tradicional de las técnicas de obtención y consiste en reuniones analista-interesado en las cuales se suceden preguntas y respuestas para extraer el dominio de la aplicación (M. GERBER, 2001). En (Pressman, 2006) se muestra un conjunto de preguntas que se pueden utilizar en el desarrollo de esta técnica, que tiene una alta participación del analista y se realiza en conjunto con otras técnicas.

### Tormenta de ideas

Es una técnica de reuniones en grupo cuyo objetivo es la generación de ideas en un ambiente libre de críticas o juicios. Puede ayudar a generar una gran variedad de vistas del problema y a formularlo de diferentes formas, sobre todo al comienzo del proceso de captura, cuando los requisitos son todavía muy difusos (Pressman, 2006).

### Análisis de Documentos

El análisis de documentos se utiliza para obtener requisitos mediante el examen de la documentación existente y la identificación de la información relevante para los requisitos. Se puede analizar una amplia variedad de documentos, que podrían ayudar a obtener requisitos relevantes. Los ejemplos de documentos que se podrían analizar incluyen, entre otros: planes de negocio, literatura de mercadeo, acuerdos, solicitudes de propuesta, flujos de procesos actuales, modelos lógicos de datos, repositorios de reglas de negocio, documentación del software de la aplicación, documentación de procesos de negocio o interfaces, casos de uso, otra documentación de requisitos, registro de problemas/incidentes, políticas, procedimientos y documentación normativa como leyes, códigos y ordenanzas (Pressman, 2006).

## 1.6. Técnicas para la validación de requisitos

Como una buena práctica que le proporciona al ingeniero de software una medida de la calidad con que obtuvieron los requisititos, se utilizarán las siguientes técnicas de validación.

**Revisión Técnica Formal**

El objetivo de la revisión formal es descubrir errores en la función, la lógica o la implementación de cualquier producto del software, verificar que satisface sus especificaciones, que se ajusta a los estándares establecidos, señalando las posibles desviaciones detectadas. Es un proceso de revisión riguroso. Su objetivo es llegar a detectar lo antes posible, los posibles defectos o desviaciones en los productos que se van generando a lo largo del desarrollo. Esta característica fuerza a que se adopte esta práctica únicamente para productos que son de especial importancia, porque de otro modo podría frenar la marcha del proyecto (López, 2003).

**Prototipos no funcionales**

El uso de prototipos para recoger requisitos o comprobar si se han entendido perfectamente es una práctica cada vez más extendida, especialmente en sistemas que suponen un elevado grado de interactividad. En este caso los prototipos a evaluar no serán más que maquetas no operativas o especificaciones formales que un grupo de expertos deberán evaluar (López, 2003).

## 1.7. Patrones de diseño

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objeto. Los patrones de diseño no son fáciles de entender, pero una vez entendido su funcionamiento, los diseños serán mucho más flexibles, modulares y reutilizables (Larman, 1999).

### 1.7.1. Patrones GRASP

Estos representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es el acrónimo para General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades) (Larman, 1999).

**Experto:** se encarga de asignar una responsabilidad al experto en información, o sea, aquella clase que cuenta con la información necesaria para cumplir la responsabilidad (Larman, 1999).

**Creador:** este patrón es el responsable de asignarle a la clase B la responsabilidad de crear una instancia de clase A. B es un creador de los objetos A (Larman, 1999).

**Alta Cohesión:** plantea que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase (Larman, 1999).

**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 (Larman, 1999).

**Controlador:** asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase (Larman, 1999).

### 1.7.2. Patrones GoF

Los patrones GoF (Gang of Four, en español Pandilla de los Cuatro, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides) se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Gracia, 2005). A continuación se describen los patrones GoF que se van a utilizar para el desarrollo de la solución.

**Decorador:** este patrón perteneciente a los estructurales, responde a la necesidad de añadir dinámicamente funcionalidad a un Objeto. Esto permite no tener que crear sucesivas clases que hereden de la primera incorporando la nueva funcionalidad, sino otras que la implementan y se asocian a la primera (Gracia, 2005).

**Comando:** este patrón perteneciente a los de comportamiento, encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer la operaciones (Gracia, 2005).

## 1.8. Tecnologías

Las tecnologías seleccionadas se escogieron de acuerdo a la actualización y uso que presentan para el desarrollo web. A continuación se describen cada una de ellas.

### 1.8.1. Ajax

Ajax por sus siglas en inglés *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML) es la técnica de desarrollo web que se usará para poder hacer consultas asíncronas al servidor sin necesidad de recargar la página. Esta surge de la combinación de tres tecnologías existentes: HTML (o XHTML) y Hojas de Estilo en Cascada (CSS) para presentar la información, Document Object Model (DOM) y JavaScript para interactuar dinámicamente con los datos, además de XML y XSLT para intercambiar y manipular datos de manera sincronizada con un servidor web (TONG, 2005).

Ventajas que ofrece:

• Mejora completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, mediante la creación de un elemento intermedio entre el usuario y el servidor (TONG, 2005).

• Puede ser utilizada en cualquier plataforma o navegador (TONG, 2005).

• Las peticiones HTTP al servidor se sustituyen por peticiones JavaScript que se realizan al elemento encargado de AJAX. Las más simples no requieren intervención del servidor, por lo que la respuesta es inmediata y en caso contrario la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX (TONG, 2005).

### 1.8.2. Marco de trabajo

Un marco de trabajo, en el desarrollo de sistemas computarizados, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Es definido como un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar ().

#### 1.8.2.1. Marco de trabajo Symfony 2

Symfony 2 es un marco de trabajo de PHP diseñado para optimizar el desarrollo de las aplicaciones web, basado en principio en el patrón Modelo Vista Controlador. Se encarga de todos los aspectos comunes de las aplicaciones web, dejando que el programador se dedique a aportar valor desarrollando las características únicas de cada proyecto.

Symfony 2 es de código abierto y utiliza PHP 5.3 en lo adelante. Se ha convertido en una comunidad ya que cada día es más amplio y ofrece más posibilidades de documentación, de intercambio de conocimientos y

de archivos dentro de la red global. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows ().

### 1.8.3. Doctrine 2

Para la capa de acceso a datos se empleará Doctrine 2. Es un sistema ORM por sus siglas en inglés Object Relational Mapper (Mapeo Objeto-Relacional) para PHP 5.2 o superior que incorpora una DBL (capa de abstracción a base de datos). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las consultas de la base de datos orientada a objeto. Esto proporciona una alternativa poderosa a diseñadores de SQL, manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Además, exporta una base de datos existente a sus clases correspondientes (Reyero, 2008).

### 1.8.4. JQuery 2

JQuery es una librería gratuita y de código abierto de JavaScript, que simplifica la creación de páginas web altamente interactivas. Funciona en todos los exploradores de internet modernos y abstrae características específicas de cada uno de estos, permitiendo al desarrollador enfocarse en el diseño y resultado final, en lugar de tratar de desarrollar funciones complejas en exploradores individuales (González, 2013).

JQuery facilita:

- La búsqueda y manipulación de contenido en una página HTML.
- Trabajar con el modelo de eventos de los exploradores modernos.
- Añadir efectos y transiciones sofisticadas que se ven en páginas modernas, como animaciones disparadas por eventos.
- Hace transparente el soporte de la aplicación para los navegadores principales.
- Provee de un mecanismo para la captura de eventos.
- Integra funcionalidades para trabajar con AJAX.

### 1.8.5. Lenguajes de programación

Los lenguajes de desarrollo que se emplean para la obtención de la solución que se propone, han sido establecidos de acuerdo al marco de trabajo definido para la confección del producto.

#### 1.8.5.1. Lenguaje de programación PHP 5.3

PHP es un lenguaje de programación interpretado, originalmente diseñado para el desarrollo de aplicaciones web dinámicas. Es gratuito, fácil de usar y aprender, portable, de código abierto y multiplataforma. Presenta interfaces para una gran cantidad de sistemas de base de datos, así como bibliotecas incorporadas para muchas tareas web habituales. 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. (SANDHU, 1993).

Algunas de las ventajas que presenta:

- 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).
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con PostgreSQL.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.

Su empleo para el desarrollo del sistema se debe al uso de Symfony 2, ya que es un marco de trabajo de PHP, además de las ventajas anteriormente enunciadas.

#### 1.8.5.2. Lenguaje de programación Java Script

JavaScript es un lenguaje de programación de alto nivel que se adapta bien a los estilos de programación orientados a objetos y funcional utilizado principalmente para crear páginas web dinámicas, o sea, páginas web que incorporan efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlo. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (SANDHU, 1993).

Su empleo para el desarrollo del sistema se debe a la librería JQuery 2 que utiliza el marco de trabajo Symfony 2, la cual se encargará de la vista.

## 1.9. Patrón arquitectónico Modelo-Vista-Controlador (MVC)

Los patrones arquitectónicos especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes. Además, se pueden ver como la descripción de un problema en particular y recurrente de diseño, que aparece en contextos de diseño arquitectónicos específicos y representa un esquema genérico demostrado con éxito para su solución (A. J. MENEZES, 1996).

El patrón arquitectónico MVC es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres capas o componentes distintos. El patrón MVC se utiliza frecuentemente en aplicaciones web, donde la vista es la página HTML (*del inglés,* ***Hypertext Markup Language***) que se visualiza en el navegador y el código que proporciona de datos dinámicos a la página, el modelo contiene una representación de los datos que maneja el sistema y su lógica de negocio. Por último, el controlador es el responsable de recibir los eventos de entrada desde la vista, enviarlos al modelo y manejar también las respuestas del modelo y transmitirlas hacia la vista (). La utilización de este patrón se debe a que está implementado dentro del marco de trabajo Symfony 2, sobre el cual será desarrollado el sistema.

## 1.10. Herramientas

A continuación se describen las herramientas a emplear para el desarrollo de la solución que se propone.

### 1.10.1. Herramienta CASE Visual Paradigm 8.0

Se empleará Visual Paradigm 8.0 como herramienta CASE (Computer Aided Software Engineering*, por sus siglas en inglés*). Utiliza UML como lenguaje de modelado, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Permite dibujar todos los tipos de diagramas de clases, generar diagramas desde código, generar código desde diagramas y generar documentación, ayudando a construir aplicaciones de calidad más rápido, mejor y en bajo costo. (Fowler).

### 1.10.2. Servidor de aplicaciones Apache 2.2

Se utilizará como servidor web Apache 2.2 pues es una tecnología gratuita de código abierto compatible con muchos Sistemas Operativos. Posee el soporte que se necesita para tener páginas dinámicas. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Tiene una alta capacidad de configuración en la creación y gestión de registros de actividad. Apache permite la creación

de ficheros de registro a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor (Potencier, y otros, 2008).

### 1.10.3. Sistema gestor de base de datos PostgreSQL 9.2

PostgreSQL es un sistema de gestión de bases de datos relacional orientada a objetos de código libre. Este agiliza la interacción entre cliente, servidor y base de datos, donde PostgreSQL es el que realiza la mayoría del trabajo referente a bases de datos cuando se le realizan peticiones. Permite la definición de tipos de datos personalizados, incluye un modelo de seguridad completo y ejecuta consultas complejas y uniones de gran tamaño. Soporta transacciones, claves ajenas con comprobaciones de integridad referencial y almacenamiento de objetos de gran tamaño (CAVSI.com, 2007).

## 1.11. Métricas para la validación del diseño

Aunque los términos medida, medición y métricas se utilizan a menudo indistintamente, existe gran confusión a la hora de referirse a ellos. Dentro del contexto de la ingeniería del software, una medida proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto. La medición es el proceso por el cual los números o símbolos son asignados a atributos o entidades en el mundo real tal como son descritos de acuerdo a reglas claramente definidas. También es definida como el acto de determinar una medida (Falgueras, 2003). Las métricas permiten descubrir y corregir problemas potenciales antes de convertirse en defectos catastróficos. Se emplean con el objetivo de llevar un control de la calidad del producto que se está desarrollando, evaluar la efectividad del proceso y mejorar la calidad del trabajo.

*Métricas propuestas por Lorenz y Kidd*

Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaño para una clase se centran en cálculos de atributos y de operaciones para una clase individual, para luego promediar los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y asuntos relacionados con el código y las métricas orientadas a valores externos examinan el acoplamiento y la reutilización (Pressman, 2006).

## 1.12. Pruebas de software

Las pruebas del software son un elemento crítico para garantizar la calidad de un software y representan una visión final de las especificaciones, del diseño y de la codificación. Una vez generado el código fuente, el software debe ser probado para descubrir y corregir el máximo de errores posibles antes de su entrega al cliente (C. GUTIÉRREZ, 2004).

### 1.12.1. Pruebas de caja blanca

La prueba de caja blanca, denominadas a veces prueba de caja de cristal es un método de diseño de casos de prueba que utiliza la estructura de control del diseño procedimental para obtener los casos de prueba.

La prueba de caja blanca que se aplicará a la solución desarrollada será la Prueba del Camino Básico, que permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, garantizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa (C. GUTIÉRREZ, 2004).

### 1.12.2. Pruebas de caja negra

Las pruebas de caja negra, también denominadas pruebas 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 (C. GUTIÉRREZ, 2004). Los métodos de caja negra que se utilizarán para asegurar la calidad de la aplicación desarrollada serán:

*Partición de equivalencia*

La partición de equivalencia es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico (C. GUTIÉRREZ, 2004).

**Conclusiones parciales del capítulo**

Con la realización del presente capítulo se concluye lo siguiente:

- Los conceptos utilizados ayudan a lograr un mejor entendimiento de la investigación científica en cuestión, conformando la fundamentación teórica que sustenta el desarrollo de la investigación.
- La realización del estado del arte acerca de los sistemas de gestión existentes, permite arribar a la conclusión de que ninguno de los sistemas estudiados satisface las necesidades planteadas por la Facultad 3, en aras de una mejor gestión de las estrategias educativas.
- Como resultado del análisis de las tecnologías, lenguajes y herramientas se escogió como metodología de desarrollo AUP, como marco de trabajo Symfony 2, PostgreSQL como gestor de base de datos, Apache 2.2 como servidor web y Visual Paradigm 8.0 como herramienta CASE.

# Capítulo 2. Modelado del negocio, requisitos, análisis y diseño

## Introducción

Este capítulo describe la solución propuesta siguiendo las disciplinas de la fase de ejecución: modelado del negocio, requisitos y análisis y diseño que propone la metodología AUP. Se muestran las características específicas del negocio mediante el modelado y descripción de los procesos del mismo. Además, se registran en el modelo conceptual los conceptos fundamentales del negocio. Se identifican los actores y la responsabilidad que tiene cada uno con el sistema; se especifican los requisitos funcionales y no funcionales, así como las técnicas de captura y validación de requisitos utilizadas. Se elabora el diseño mediante los diagramas de clases del diseño y el diagrama de paquetes, modelo físico de datos y la utilización de patrones dentro del diseño del sistema. Finalmente se valida el diseño propuesto mediante el uso de métricas.

## 2.1. Modelado del negocio

La metodología escogida propone como primera disciplina de la fase de ejecución el modelado del negocio, para una mayor comprensión de los procesos que se realizan en el mismo.

El modelado de procesos de negocio es la representación de los procesos de negocio de una empresa u organización con el objetivo de que puedan ser analizados y mejorados. Su elaboración brinda una serie de beneficios a las organizaciones permitiéndoles reutilizar procesos que sean más eficientes, ventajosos y productivos, detectar tareas que no puedan ser realizadas, así como mejorar de forma general los procesos que se realizan (Noguera, 2011).

La confección de los modelos de procesos de negocio es realizada principalmente por los analistas, labor que les brinda una serie de beneficios como: la agilización del proceso de desarrollo y de la carga de trabajo, la identificación de errores en las fases tempranas y mayor nivel de abstracción (Noguera, 2011).

Siguiendo la metodología de desarrollo seleccionada, se elaboraron los artefactos de salida: Modelo conceptual, Diagrama de procesos de negocio y Descripción de procesos de negocio. A continuación de detalla cada uno de estos artefactos y los resultados obtenidos por estos.

### 2.1.1. Modelo Conceptual

El modelo conceptual es un diagrama conformado con los conceptos que son significativos para el área que se analice, así como las relaciones existentes entre ellos. Permite identificar, organizar y realizar razonamientos sobre los componentes y comportamientos del sistema, siendo la guía para el proceso de

diseño del software, pudiéndose usar además, como referencia para evaluar un diseño particular y razonar sobre la solución realizada (Reyero, 2008).

El modelo conceptual del proceso de gestión de las Estrategias Educativas fue confeccionado con todas las clases conceptuales identificadas en este proceso, junto con las relaciones existentes entre las mismas. En la Figura 2 se muestra una versión reducida del Modelo conceptual del proceso de gestión de las Estrategias Educativas de la Facultad 3, pudiéndose observar el diagrama con los atributos asociados a cada concepto en el Anexo 1.

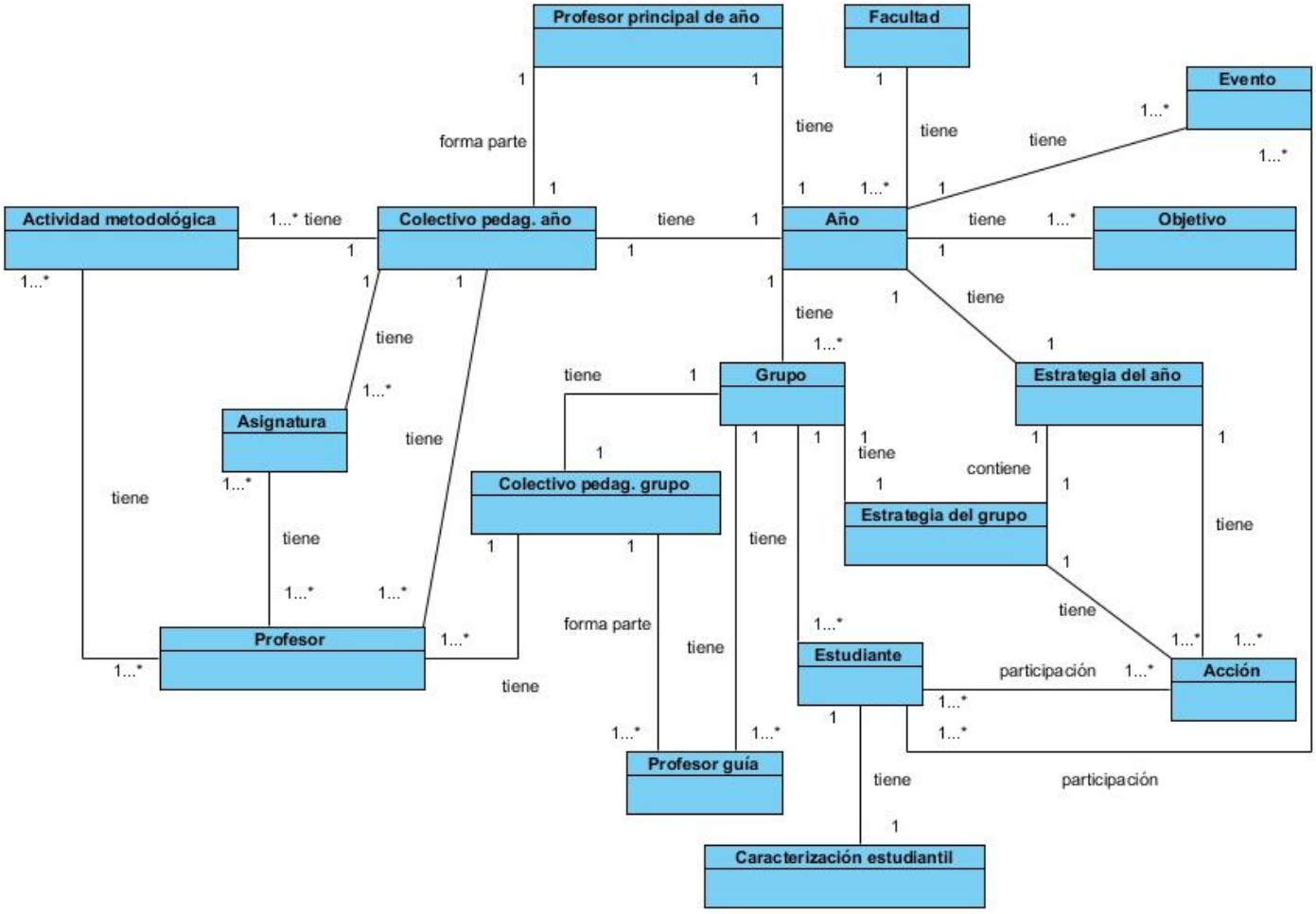



Figura 2 Modelo conceptual del proceso de gestión de las Estrategias Educativas de la Facultad 3.

Con la utilización de las técnicas de obtención de información: entrevistas y análisis de documentos, llevadas a cabo con el cliente, se llegaron a detectar los procesos de negocio: realizar estrategia educativa del año (macro-proceso) y realizar estrategia educativa del grupo (sub-proceso). En el Anexo 2 se puede consultar el diagrama del proceso del negocio realizar estrategia educativa del año. A continuación se muestra el

diagrama del procesos de negocio realizar estrategia del grupo. Para consultar el diagrama de procesos de negocio realizar estrategia del año ver el Anexo 2.

### 2.1.2. Diagrama de procesos de negocio

Un proceso de negocio es un conjunto de actividades relacionadas que definen cómo se crea un producto o servicio final y tiene como objetivo fundamental satisfacer al cliente (Noguera, 2011). El diagrama de procesos de negocio va a permitir modelar el flujo de trabajo del proceso de gestión de las Estrategias Educativas de la Facultad 3. En la Figura 3 se muestra el diagrama de procesos de negocio realizar estrategia del grupo.

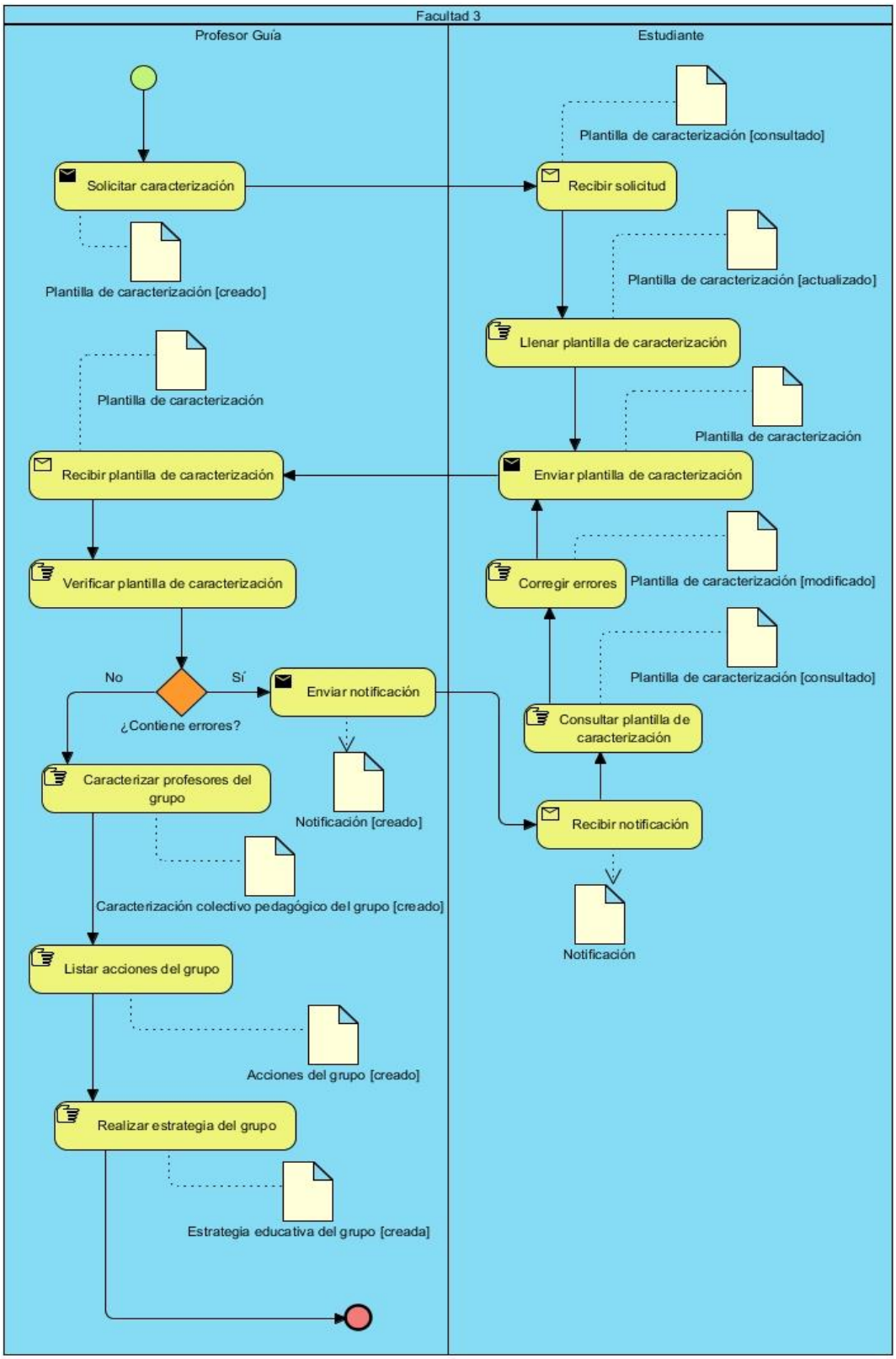



Figura 3. Diagrama del procesos de negocio realizar estrategia educativa del grupo.

A continuación se realiza la descripción del proceso para una mayor comprensión del diagrama mostrado.

### 2.1.3. Descripción del proceso de negocio realizar estrategia educativa del grupo.

A continuación se muestran en la Tabla 2 los actores involucrados en el negocio.

Tabla 2. Actores del negocio

| **Actores** | **Descripción** |
|---|---|
| Estudiante | Persona encargada de registrar sus datos personales, docentes y extensionistas. |
| Profesor Guía | Persona encargada de pedir y analizar la caracterización estudiantil, detectar posibles problemas del grupo y realizar un listado de los profesores del grupo para luego confeccionar la estrategia del año. |

En la Tabla 3 se muestra la descripción del proceso realizar estrategia educativa del grupo.

Tabla 3. Descripción del proceso realizar estrategia educativa del grupo.

| | |
|---|---|
| **Objetivo** | Caracterizar el proceso realizar estrategia del grupo |
| **Evento(s) que lo genera(n)** | Envío de las acciones del año por parte del profesor principal a los profesores guías. |
| **Pre condiciones** | El profesor guía cuenta con una lista de las acciones del año. |
| **Marco legal** | N/A |
| **Reglas de negocio** | Para la gestión de la estrategia del grupo se deben gestionar las acciones y las caracterizaciones de los estudiantes del grupo.<br>Las acciones del grupo solo podrán ser gestionadas por el profesor guía.<br>Los datos de cada estudiante solo podrán ser modificados por el mismo estudiante. |
| **Responsable** | Profesor Guía |
| **Clientes internos** | Profesor Guía |
| **Clientes externos** | Profesor Principal<br>Vicedecana de Formación |

| | |
|---|---|
| | Decana<br>Asesora Metodológica |
| **Entradas** | Plantilla de Caracterización (PC) [Creado]<br>Listado de Profesores (LProf.) [Creado]<br>Listado de Acciones (LA) [Creado] |

**Flujo de eventos**

**Flujo básico**

1. **Solicitar Caracterización:** el profesor guía envía solicitud de llenar la plantilla de caracterización al estudiante.
2. **Recibir solicitud:** el estudiante recibe el pedido de llenar la plantilla de caracterización con sus datos.
3. **Llenar plantilla de caracterización:** el estudiante llena la plantilla con sus datos.
4. **Enviar plantilla de caracterización:** el estudiante envía la plantilla de caracterización con sus datos.
5. **Recibir plantilla de caracterización:** el profesor guía recibe la plantilla de caracterización.
6. **Verificar plantilla de caracterización:** el profesor guía verifica que no existan errores en los datos de las plantilla de caracterización.
7. **Caracterizar profesores del grupo:** el profesor guía elabora un listado de los profesores del grupo con los datos de cada uno.
8. **Listar acciones del grupo:** el profesor guía elabora un listado con las acciones que se proponen en el aula.
9. **Realizar estrategia del grupo:** el profesor guía elabora el documento estrategia educativa del grupo que contiene las acciones del grupo y las caracterizaciones de los estudiantes y del colectivo pedagógico.

**Pos-condiciones**

1. El profesor guía cuenta con la plantilla de caracterización de cada estudiante y los listados de profesores y acciones.

**Salidas**

1. Estrategia del Grupo (EG)

**Flujos paralelos**

**N/A**

**Flujos alternos**

**6.a El estudiante envía la planilla de caracterización con errores.**

1. El estudiante envía la planilla con datos incorrectos.
2. El profesor guía recibe la planilla.
3. El profesor guía revisa la planilla enviada, detectando los errores.
4. El profesor guía envía una notificación al estudiante solicitando la corrección de los datos.
5. El estudiante recibe la notificación y consulta la plantilla de caracterización.
6. El estudiante corrige los errores.
7. Volver al paso 4 del flujo básico.

**Asuntos pendientes**

N/A

Una vez identificados y descritos los procesos del negocio que se desea informatizar, la metodología seleccionada propone realizar la disciplina de Requisitos, destinada a desarrollar el modelo del sistema que se va a construir.

## 2.2. Requisitos

La gestión de requisitos es el proceso encargado de la identificación, asignación y seguimiento de los requisitos para la creación de un proyecto, incluyendo la interfaz, verificación, modificación y control a todo lo largo del ciclo de vida del proceso. Es el conjunto de actividades que lleva el aseguramiento de las especificaciones, por ejemplo, los requisitos que son reunidos para la satisfacción del cliente. Es el proceso que inicia con la concepción de un proyecto y continúa hasta el resultado final del producto (2012).

Como primera tarea de esta disciplina se realiza la identificación de requisitos, a través de técnicas de levantamiento de requisitos. Seguidamente, se especifican los requisitos mediante el artefacto de salida Descripción de requisitos. Finalmente, como una buena práctica que le proporciona al ingeniero de software una medida de la calidad con que realizó las tareas de la disciplina, son validados mediante el uso de técnicas de validación.

A continuación se describen las técnicas de levantamiento de requisitos utilizadas y los resultados obtenidos en ellas.

### 2.2.1. Técnicas de levantamiento de requisitos.

Para el levantamiento de los requisitos fueron aplicadas las siguientes técnicas:

**Entrevista**

A través de entrevistas no estructuradas realizadas a la asesora docente metodológica y a la profesora principal del cuarto año académico de la Facultad 3, se realizaron preguntas relacionadas con los procesos de Estrategias Educativas a nivel de año y grupo en función de obtener las necesidades a informatizar. Esto proporcionó como resultado una descripción detallada del proceso de gestión de las Estrategias Educativas, permitiendo que fueran identificados algunos de los requisitos por los que se guiaría el desarrollo de las funcionalidades.

**Análisis de documentos**

Se llevó a cabo el estudio y análisis de las estrategias educativas de todos los años académicos de la facultad en el curso 2014-2015 (Peña, 2014) (Reyes, 2014) (Lorenzo, 2014) (Torres, 2014), obteniéndose

algunos de los requisitos mediante el examen de esta documentación e identificándose información relevante para los mismos.

**Tormenta de ideas**

En cada encuentro realizado con la asesora docente metodológica fueron debatidos los requisitos identificados, dando lugar a varias ideas acerca de los mismos, lo cual arrojó como resultado que se obtuviera una visión más amplia de lo que se quería implementar, favoreciéndose de esta forma el avance de futuras etapas en el desarrollo del sistema.

### 2.2.2. Requisitos Funcionales (RF)

Los requisitos funcionales indican características y restricciones sobre la funcionalidad del software, además, son la condición necesaria de un atributo para que cumpla una función determinada (Koch, 2002). Luego de haber aplicado las técnicas de levantamiento de requisitos antes mencionadas, se identificaron los siguientes requisitos funcionales:

**Gestionar Estudiantes**

RF 1: Adicionar estudiante.

RF 2: Modificar estudiante.

RF 3: Eliminar estudiante.

RF 4: Mostrar estudiante.

**Gestionar Profesores**

RF 5: Adicionar profesor.

RF 6: Buscar profesor.

RF 7: Modificar profesor.

RF 8: Eliminar profesor.

RF 9: Mostrar profesor.

**Gestionar Asignaturas**

RF 10: Listar asignaturas.

RF 11: Adicionar asignatura.

RF 12: Modificar asignatura.

RF 13: Eliminar asignatura.

**Gestionar Objetivos del Año**

RF 14: Listar objetivos.

RF 15: Adicionar objetivo.

RF 16: Modificar objetivo.

RF 17: Eliminar objetivo.

**Gestionar Acciones de año**

RF 18: Listar acciones de año.

RF 19: Adicionar acción de año.

RF 20: Modificar acción de año.

RF 21: Eliminar acción de año.

RF 22: Mostrar acción de año.

RF 23: Buscar acción de año.

**Gestionar Actividades Metodológicas**

RF 24: Listar actividades metodológicas.

RF 25: Adicionar actividad metodológica.

RF 26: Modificar actividad metodológica.

RF 27: Eliminar actividad metodológica.

RF 28: Mostrar actividad metodológica.

RF 29: Buscar actividad metodológica.

**Gestionar Grupos**

RF 30: Listar grupos.

RF 31: Adicionar grupo.

RF 32: Modificar grupo.

RF 33: Eliminar grupo.

RF 34: Mostrar grupo.

**Gestionar participación de estudiantes en eventos**

RF 35: Listar participaciones.

RF 36: Adicionar participación al estudiante.

RF 37: Modificar participación del estudiante.

RF 38: Eliminar participación del estudiante.

**Gestionar Eventos**

RF 39: Listar eventos.

RF 40: Adicionar evento.

RF 41: Modificar evento.

RF 42: Eliminar evento.

RF 43: Mostrar evento.

**Gestionar Familia**

RF 44: Listar familia.

RF 45: Adicionar familiar.

RF 46: Modificar familiar.

RF 47: Eliminar familiar.

**Gestionar Investigaciones**

RF 48: Listar investigaciones.

RF 49: Adicionar investigación.

RF 50: Modificar investigación.

RF 51: Eliminar investigación.

**Gestionar Acciones de grupo**

RF 52: Listar acciones de grupo.

RF 53: Adicionar acción de grupo.

RF 54: Modificar acción de grupo.

RF 55: Eliminar acción de grupo.

RF 56: Mostrar acción de grupo.

RF 57: Buscar acción de grupo.

**Gestionar Usuarios**

RF 58: Listar usuarios.

RF 59: Adicionar usuario.

RF 60: Modificar usuario.

RF 61: Buscar usuario.

**Gestionar Roles**

RF 62: Listar roles.

RF 63: Modificar rol.

RF 64: Buscar rol.

**RF 65: Planificar Evento.**

**RF 66: Generar Estrategia Educativa del año.**

**RF 67: Generar Estrategia Educativa del grupo.**

**RF 68: Mostrar acciones realizadas por año en un rango de fechas.**

**RF 69: Mostrar estudiantes con (n) cantidad de participaciones en eventos por año.**

**RF 70: Mostrar acciones realizadas por grupo en un rango de fechas.**

**RF 71: Mostrar estudiantes con (n) cantidad de participaciones en eventos por grupo.**

**RF 72: Enviar notificación a los involucrados en las acciones del año.**

**RF 73: Enviar notificación a los involucrados en las acciones del grupo.**

**RF 74: Enviar notificación a los involucrados en las actividades metodológicas.**

**RF 75: Actualizar estado de las acciones del año.**

**RF 76: Actualizar estado de las acciones del grupo.**

**RF 77: Actualizar estado de las actividades metodológicas.**

**RF 78: Mostrar Claustro pedagógico del año.**

**RF 79: Autenticar Usuario.**

**RF 80: Exportar a Word Estrategia Educativa del año.**

**RF 81: Exportar a Word Estrategia Educativa del grupo.**

### 2.2.3. Descripción de requisitos

Para consultar algunos de los artefactos de Descripción de requisitos ver el Anexo 1. A continuación se muestra la descripción textual del requisito funcional Modificar asignatura, donde se especifica con más detalle el funcionamiento del mismo.

Tabla 4. Especificación del requisito Modificar asignatura.

| | |
|---|---|
| **Precondiciones** | La asignatura ha sido seleccionada. |
| **Flujo de eventos** | |
| **Flujo básico** | |
| 1 | Se modifican los siguientes datos en los campos del formulario:<br>-Nombre<br>-Semestre |
| 2 | El sistema valida (ver validación 1) los datos seleccionados. |
| 3 | Si los datos son correctos el sistema modifica los datos de la asignatura. |
| 4 | El sistema confirma la modificación realizada. |
| 5 | Concluye el requisito. |
| **Pos-condiciones** | |
| 1 | Se modificó la asignatura |
| **Flujos alternativos** | |
| **Flujo alternativo 2.a Información incompleta** | |
| 1 | El sistema señala los datos vacíos y permite corregirlos. |

| | |
|---|---|
| – | El usuario corrige los datos. |
| – | Volver al paso 2 del flujo básico. |

| **Pos-condiciones** | |
|---|---|
| 1 | N/A |

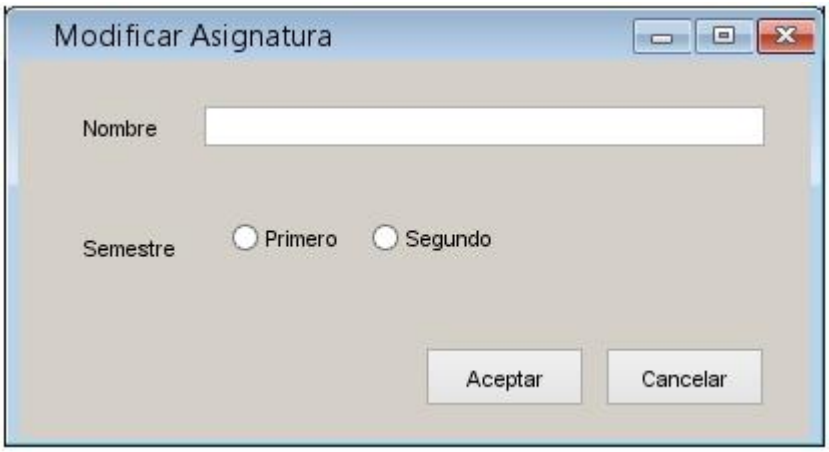
| **Flujo alternativo *.a El usuario cancela la acción** | |
|---|---|
| 1 | Concluye el requisito. |

| **Pos-condiciones** | |
|---|---|
| 1 | N/A |

| **Validaciones** | |
|---|---|
| 1 | Se validan los datos según lo establecido en el artefacto Modelo conceptual SGEE-MC-001. |

| **Conceptos** | **Asignatura** | **Visibles en la interfaz:** |
|---|---|---|
| | | Nombre |
| | | Optativa |

**Prototipo elemental de interfaz gráfica del Modificar asignatura**




A continuación se describen los requisitos no funcionales identificados.

### 2.2.4. Requisitos No Funcionales (RNF)

Son propiedades o cualidades que el producto debe tener. Los RNF deben establecer restricciones en el producto que está siendo desarrollado, en el proceso de desarrollo y en restricciones específicas que el producto pueda tener (Koch, 2002).

De acuerdo a los atributos de calidad que propone la metodología seleccionada se muestran los RNF de la propuesta de solución.

**Apariencia o interfaz externa.**

- El sistema debe que ofrecer una interfaz amigable, fácil de operar.
- Debe ser interactivo con el usuario.
- La comunicación entre el servidor Web y las máquinas clientes será mediante HTTP y entre el servidor y el directorio activo mediante LDAP.

**Usabilidad**

- El idioma de todas las interfaces de la aplicación será el español.
- Los campos de texto tendrán un tamaño estándar de acuerdo con el espacio que posea el área de la página y en la medida que se llene esa área primaria se agregará la barra de desplazamiento vertical.
- El sistema expondrá el menú general desde cualquiera de sus páginas.
- El sistema mostrará las opciones desactivadas siempre que no se hayan cumplido las condiciones previas para su activación, evitando así errores del usuario en la gestión.

**Disponibilidad**

- El sistema estará disponible las 24 horas del día, para ello debe instalarse en un servidor central.

**Mantenibilidad**

- El sistema debe estar en capacidad de permitir en el futuro su fácil mantenimiento con respecto a los posibles errores que se puedan presentar durante la operación del sistema.

**Portabilidad**

- El sistema debe ser multiplataforma.

**Seguridad**

- El sistema manejará la seguridad de acceso y administración de usuarios mediante el otorgamiento de privilegios, roles y asignación de perfiles.

- Se concederá acceso al sistema a partir de un nombre de usuario y una contraseña a través del servicio LDAP que brinda la universidad.
- El sistema concederá acceso a cada usuario autenticado sólo a las funciones que le estén permitidas de acuerdo a la configuración del sistema y a los permisos establecidos al rol que posea.

**Hardware**

Cliente:

- Requerimientos mínimos: procesador Pentium IV a 800 MHz, 512 mb de memoria RAM.
- Las computadoras clientes deben estar conectadas en red.

Servidor:

- Requerimientos mínimos: procesador Dual Core a 3.00 GHz, 2gb de memoria RAM y una capacidad de 40gb de disco duro.
- El servidor debe tener al menos una tarjeta de red para establecer la conexión.

**Software**

Cliente:

- Sistema operativo con interfaz gráfica y soporte para red.
- Las interfaces deben ser compatibles con Mozilla Firefox 3.0 o superior.

Servidor:

- Servidor web Apache 2.2 o superior.
- Gestor de base de datos PostgreSQL 9.1 o superior.

Una vez identificados y descriptos los requisitos se realiza la validación de los mismos.

### 2.2.5. Validación de requisitos

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 (Koch, 2002).

Para la validación de los requisitos de la solución propuesta fueron aplicadas las siguientes técnicas:

**Revisión técnica formal**

Fue desarrollada a partir de las revisiones entre el equipo de trabajo y el cliente. A través de diferentes encuentros y entregas de artefactos para su revisión, se examinaron las especificaciones y se realizó una búsqueda de errores en el contenido, malas interpretaciones, información incompleta, inconsistencias y que los requisitos no fueran contradictorios, imposibles o inalcanzables, dándosele solución a todas las incongruencias encontradas. Con esta técnica se pudo validar que los requisitos no fueran ambiguos, ni tuviesen omisiones o errores y además que cada uno cumplía con las necesidades del cliente.

**Prototipado de interfaces de usuario**

A partir de las especificaciones de requisitos fueron conformados prototipos de interfaz de usuario. Con esto se validó que los requerimientos estaban en concordancia con las necesidades plasmadas por el cliente. El empleo de esta técnica ofreció como resultados que el cliente tuviera una idea de la estructura de la interfaz de usuario y se favoreciera la comunicación con el mismo, teniendo una visión inicial de las funcionalidades a través de las cuales se gestionaría el proceso de gestión de las Estrategias Educativas. En la Figura 4 se muestra el prototipo para la gestión de las asignaturas.

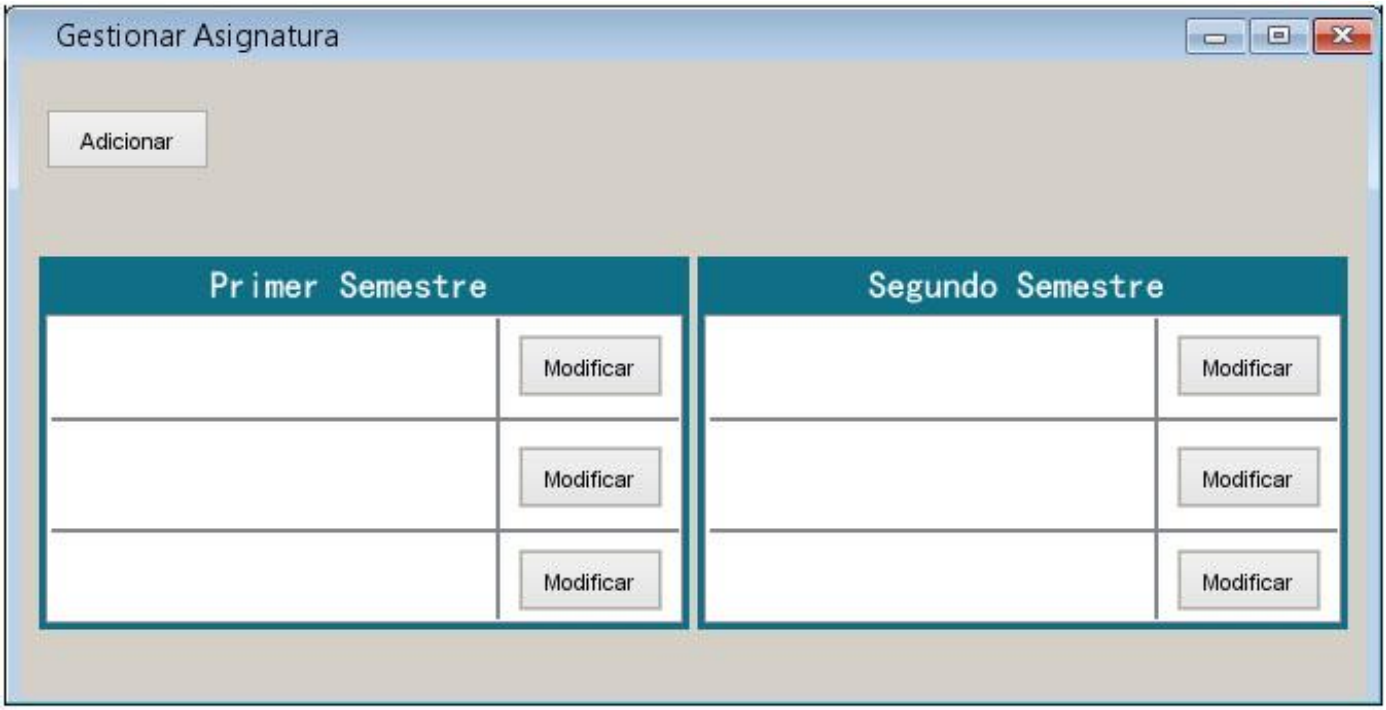



Figura 4. Prototipo del requisito Gestionar asignatura.

La utilización de estas técnicas permitió la validación de los requisitos funcionales y no funcionales antes definidos, terminándose así las actividades de la disciplina Requisitos. Luego de concluida esta disciplina la metodología propone la realización del Análisis y diseño de la solución, la cual se describe a continuación.

### 2.3. Análisis y diseño

El análisis y diseño es la disciplina donde rige la creatividad, en la cual los requisitos del cliente, las necesidades de negocio y las consideraciones técnicas se unen en la formulación de un producto o sistema. Crea una representación o modelo del software, proporciona detalles acerca de las estructuras de datos, las arquitecturas, las interfaces y los componentes del software que son necesarios para implementar el sistema (Pressman, 2006).

De acuerdo a lo que propone la metodología se realiza la descripción del uso del patrón arquitectónico definido, el diagrama de paquetes y de clases del diseño, la utilización de los patrones de diseño y finalmente como una buena práctica se valida el diseño propuesto mediante el uso de métricas.

#### 2.3.1. Patrón Arquitectónico

El patrón arquitectónico propuesto para la solución, es el MVC que utiliza el marco de trabajo escogido Symfony 2. Este marco de trabajo toma lo mejor de la arquitectura MVC y la realiza de modo que el desarrollo de aplicaciones sea rápido y sencillo. En el controlador se encuentran las acciones, las cuales son el núcleo de la aplicación, pues contienen toda la lógica de la aplicación. Estas acciones utilizan el modelo y precisan las variables para la vista. Al realizarse una petición web en una aplicación Symfony 2, la URL define una acción y los parámetros de la petición (ZANINOTTO, 2009).

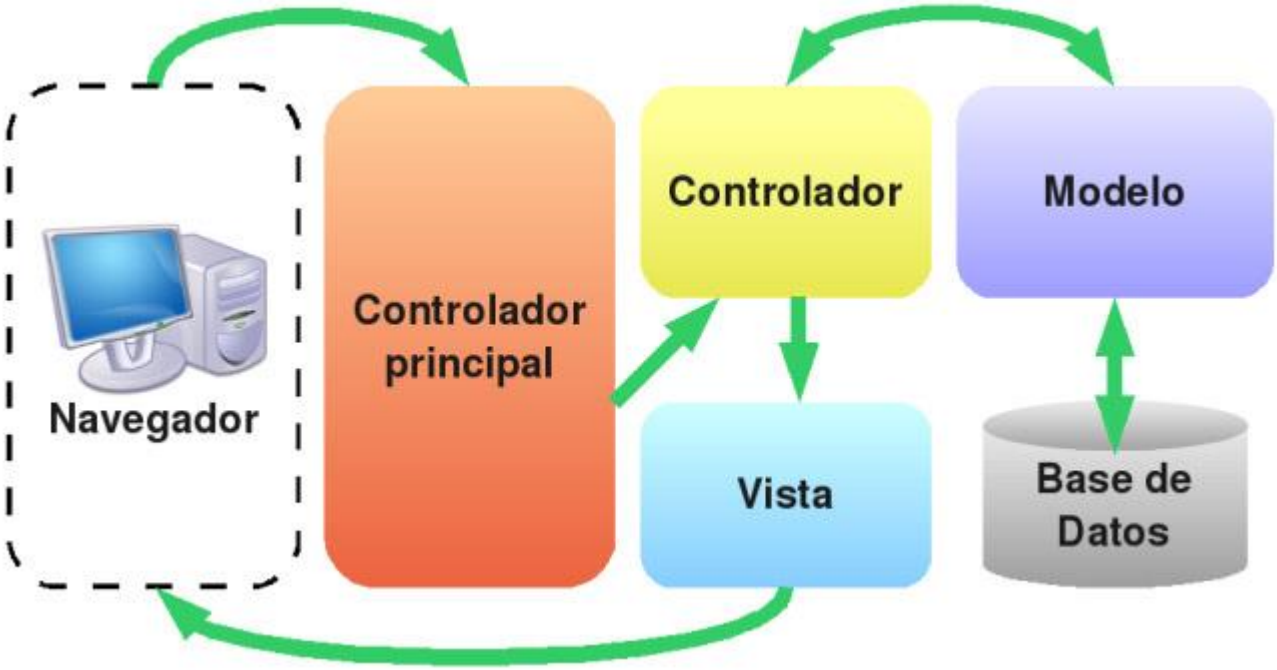



Figura 5. Representación del patrón MVC en Symfony 2

La vista es la encargada de originar las páginas que son mostradas como resultado de las acciones, donde se encuentra el layout, que es común para todas las páginas de la aplicación. La vista en Symfony 2 está conformada por varias partes, preparadas cada una de ellas especialmente para ser fácilmente transformables por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones.

En el Modelo se encuentran las clases que son generadas de forma automática según la estructura de la BD. En Symfony 2, el acceso y la modificación de los datos que se almacenan en la base de datos, se realiza mediante objetos. Doctrine es el motor generador que se encarga de esta generación automática para construir sus clases, creando la estructura y generando el código de las mismas.

Teniendo como base la arquitectura definida se realiza el diseño del sistema mediante los diagramas de clases del diseño y el diagrama de paquetes, los cuáles se muestran a continuación.

### 2.3.2. Diagrama de Paquetes

Un paquete es un mecanismo utilizado para agrupar y organizar los elementos modelados con UML, facilitando de esta forma el manejo de los modelos de un sistema complejo. Los paquetes pueden estar anidados unos dentro de otros y unos paquetes pueden depender de otros. Se pueden utilizar para plantear la arquitectura del sistema a nivel macro (UML Diagramas de Paquetes (UML ilustrado), 2009 Septiembre). En la Figura 6 se muestra el diagrama de paquetes de la solución propuesta, el cual está compuesto por

tres paquetes fundamentales SRC, Web y Vendor. En SRC se encuentra el código generado por el desarrollador, en Web los datos mostrados al usuario y en Vendor los componentes generados por terceras partes.

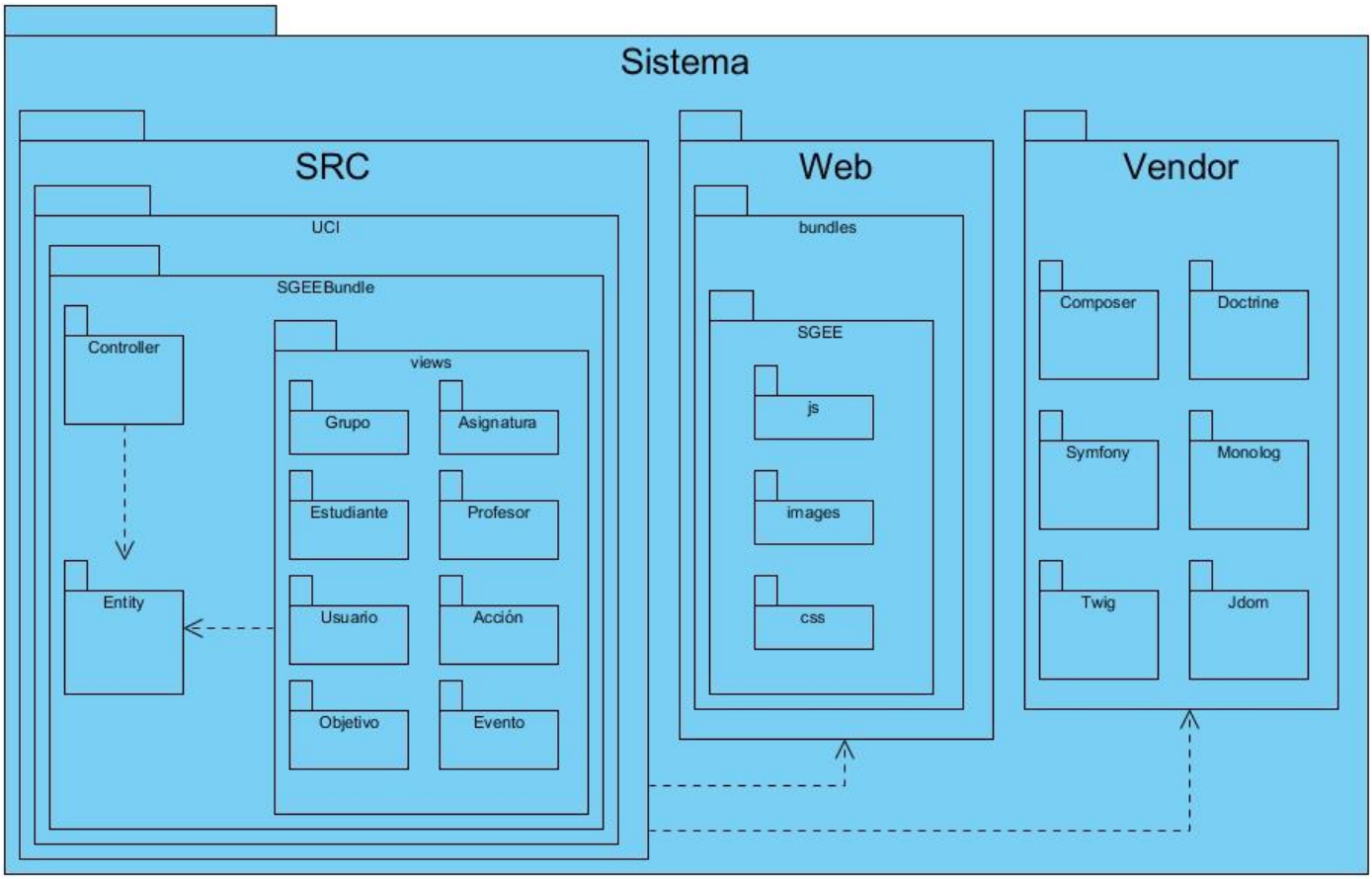



Figura 6. Diagrama de paquetes de la solución propuesta.

### 2.3.3. Diagrama de Clases del Diseño

El diagrama de Clases del Diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación, contiene información como asociaciones, atributos, métodos y dependencias (Pressman, 2006).

Para consultar algunos de los diagramas de clases del diseño ver Anexo 4. En la figura 7 se muestra el diagrama de clases del diseño de la solución propuesta, perteneciente al escenario Gestionar asignatura. En este se puede observar cómo funciona el patrón MVC agrupando las clases por capas, la vista, el controlador y el modelo.

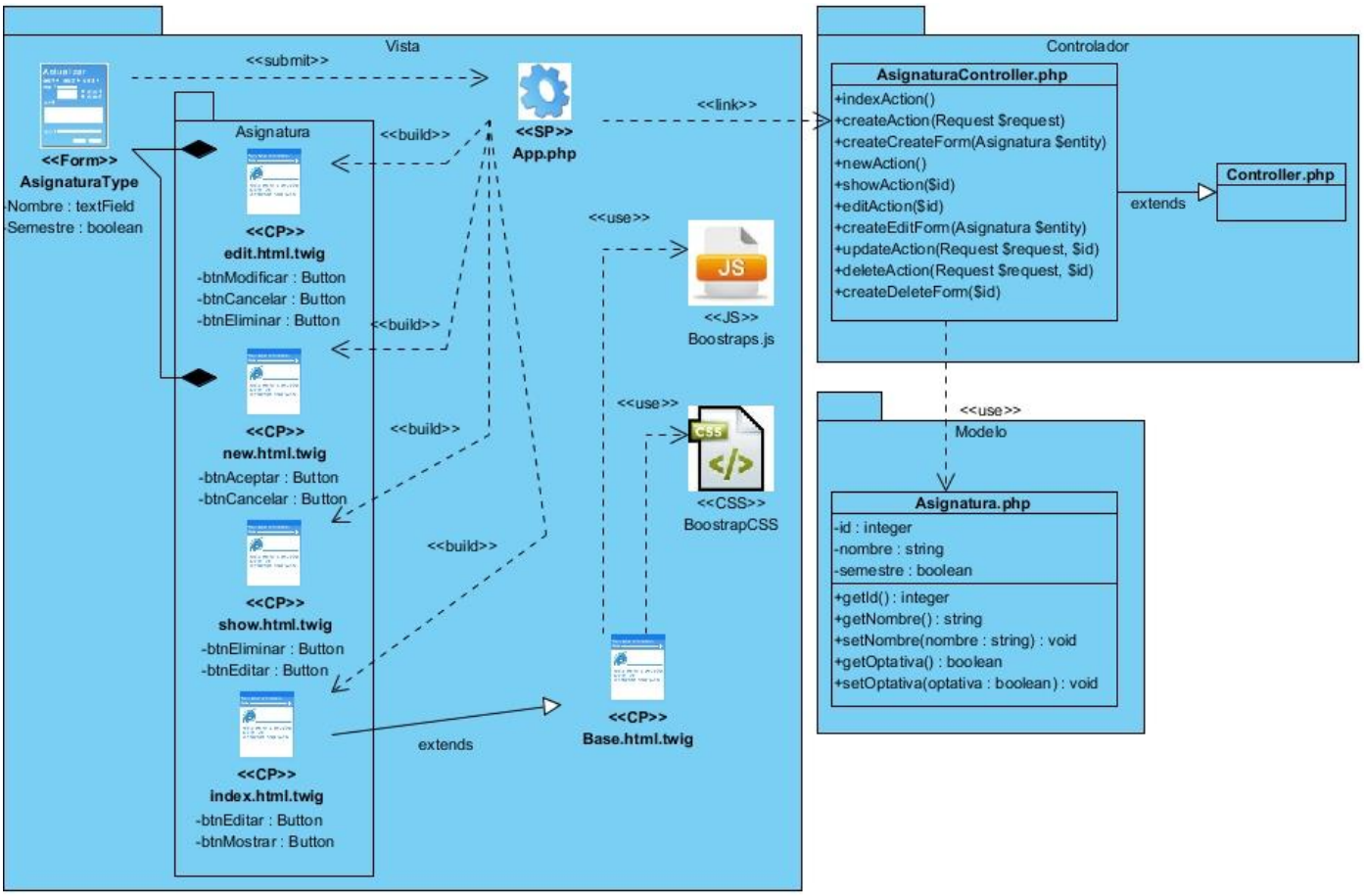



Figura 7 Diagrama de clases del diseño de la solución propuesta.

Para una mejor comprensión del diagrama antes expuesto a continuación se realiza la descripción de las clases del diseño del escenario Gestionar asignatura.

### 2.3.3.1 Descripción de las clases del diseño

A continuación se presenta la descripción de la clase AsignaturaController.php con el objetivo de brindar un mayor entendimiento de su funcionamiento.

Tabla 5. Descripción de la clase del diseño

| **Nombre:** AsignaturaController.php | |
|---|---|
| **Tipos de clase: Controlador** | |
| **Para cada responsabilidad:** | |
| **Nombre** | indexAction() |

| | |
|---|---|
| **Descripción** | Se crea un objeto con todos los valores de la tabla asignatura y lo envía a la página index.html.twig para que esta pueda cargar todos los datos del objeto. |
| **Nombre** | createAction(Request $request) |
| **Descripción** | Crea un objeto del tipo Asignatura con los valores recogidos en el formulario, luego son validados y si son correctos guarda este nuevo objeto en la tabla asignatura, por último llama al método showAction($id) con el parámetro id del objeto creado. |
| **Nombre** | createCreateForm(Asignatura $entity) |
| **Descripción** | A partir de un objeto del tipo Asignatura se manda a crear un formulario con los valores del objeto y retorna el formulario. |
| **Nombre** | newAction() |
| **Descripción** | Se crea un objeto del tipo Asignatura y manda a crear un formulario con el objeto por parámetro. |
| **Nombre** | showAction($id) |
| **Descripción** | Se crea un objeto del tipo Asignatura con los datos de la tabla asignatura donde el id sea igual al parámetro entrado para luego ser mostrado en la página show.html.twig. |
| **Nombre** | editAction($id) |
| **Descripción** | Se crea un objeto del tipo Asignatura con los datos de la tabla asignatura donde el id sea igual al parámetro y se crea el formulario con los valores del objeto. |
| **Nombre** | createEditForm(Asignatura $entity) |
| **Descripción** | A partir de un objeto entrado por parámetro este manda a generar el formulario para luego ser mostrado. |
| **Nombre** | updateAction(Request $request, $id) |
| **Descripción** | Se crea un objeto del tipo Asignatura con los datos de la tabla asignatura donde el id sea igual al parámetro y se valida que sean correctos. De ser así se actualizan los datos en la tabla. |
| **Nombre** | deleteAction(Request $request, $id) |
| **Descripción** | Se hace una petición al formulario con el id entrado por parámetro y según la respuesta de este, se crea un objeto del tipo Asignatura con los datos de la tabla asignatura donde el id sea igual al parámetro entrado y manda a eliminar de la tabla asignatura el objeto creado. |

| | |
|---|---|
| **Nombre** | createDeleteForm($id) |
| **Descripción** | Se crea un formulario con la acción de eliminar con el parámetro entrado. |

Para la realización del diseño se utilizaron patrones los cuales se describen a continuación.

### 2.3.4. Patrones de diseño

Los patrones de diseño expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software. El diseño de la propuesta de solución se realizó aplicando los patrones GRASP y GoF. Para la implementación de Symfony 2 se utilizan estos patrones, situándolos en las capas de Vista, Modelo y Control que plantea el patrón arquitectónico MVC (ZANINOTTO, 2009).

**GRASP**

A continuación se describe el uso de los patrones GRASP en el diseño de la solución.

**Experto:** es uno de los patrones que más se utiliza cuando se trabaja con Symfony 2, con la inclusión de Doctrine para mapear la base de datos. Symfony 2 utiliza este ORM para implementar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad y contienen la información necesaria de la tabla que representan.

**Creador:** en las clases controladoras del SGEEBundle se encuentran las funcionalidades con el sufijo “Action”, la cuales contiene las acciones y son las encargadas de ejecutarlas. Dentro de estas existen varias funcionalidades en las cuales se crean varios objetos o instancias de las clases que representan cada una de las tablas existentes en la base de datos.

**Controlador:** todas las peticiones web son manipuladas por un controlador frontal, ejemplo de esto son las clases “app.php” y “app_dev.php”, que es el punto de entrada único de toda la aplicación en un entorno determinado. Este patrón se evidencia en las clases controladoras del sistema. En Symfony2 hay una capa específicamente para los controladores, que son el núcleo del mismo, aquí cada clase en esta capa tiene su responsabilidad y es única, hay controladores que se encargan de la seguridad del sistema trabajando con ficheros “yml” de configuración. Esto se aplica a la clase “app.php” que es el controlador frontal.

**Alta cohesión:** se emplea en las clases controladoras ya que fueron asignadas responsabilidades a las clases de forma tal que la cohesión siguiera siendo alta, o sea, cada clase se encargará de realizar solamente las funciones que estén en correspondencia con la responsabilidad que posea. Lo cual permite que el software sea flexible a cambios sustanciales con efecto mínimo.

**Bajo acoplamiento:** este patrón se evidencia dentro del marco de trabajo Symfony2 en las clases que implementan la lógica del negocio y de acceso a datos que se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja. Como existe poca dependencia entre esas clases se permite una mayor reutilización.

**GOF**

Con el uso del marco de trabajo Symfony 2 se aplican los siguientes patrones GoF:

**Comando:** este patrón se observa en las clases “app.php” y “app_dev.php”, son las encargadas de establecer el módulo y la acción que se va a usar según la petición del usuario. Esto se aplica en la clase “RoutingManipulator.php”, que está desactivada por defecto y procede según las necesidades del administrador del sistema donde se aplique el marco de trabajo, la cual se puede activar o desactivar. En este método se comprueba la URL con el objetivo de precisar los parámetros de la misma y de esta forma saber el action que debe responder a la petición.

**Decorador:** este patrón se observa en la clase “base.html.twig”, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo nombrado “layout.php.twig” 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 el otro punto de vista, el layout decora la plantilla. Este procedimiento es una implementación del patrón Decorator.

Para una mayor comprensión de como están representados los datos en el sistema se realiza el modelo de datos que se muestra a continuación.

### 2.3.5. Modelo de datos

El modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Permite describir las estructuras de datos de la base de datos (el tipo de los datos que incluye la base de datos y la forma en que se relacionan), las restricciones de integridad (las condiciones que los datos deben cumplir para reflejar correctamente la realidad deseada) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos de la base de datos (CAVSI.com, 2007).

A continuación se presenta en la Figura 8 el modelo de datos del sistema el cual cuenta con 19 tablas, mostrándose las relaciones existentes entre ellas.

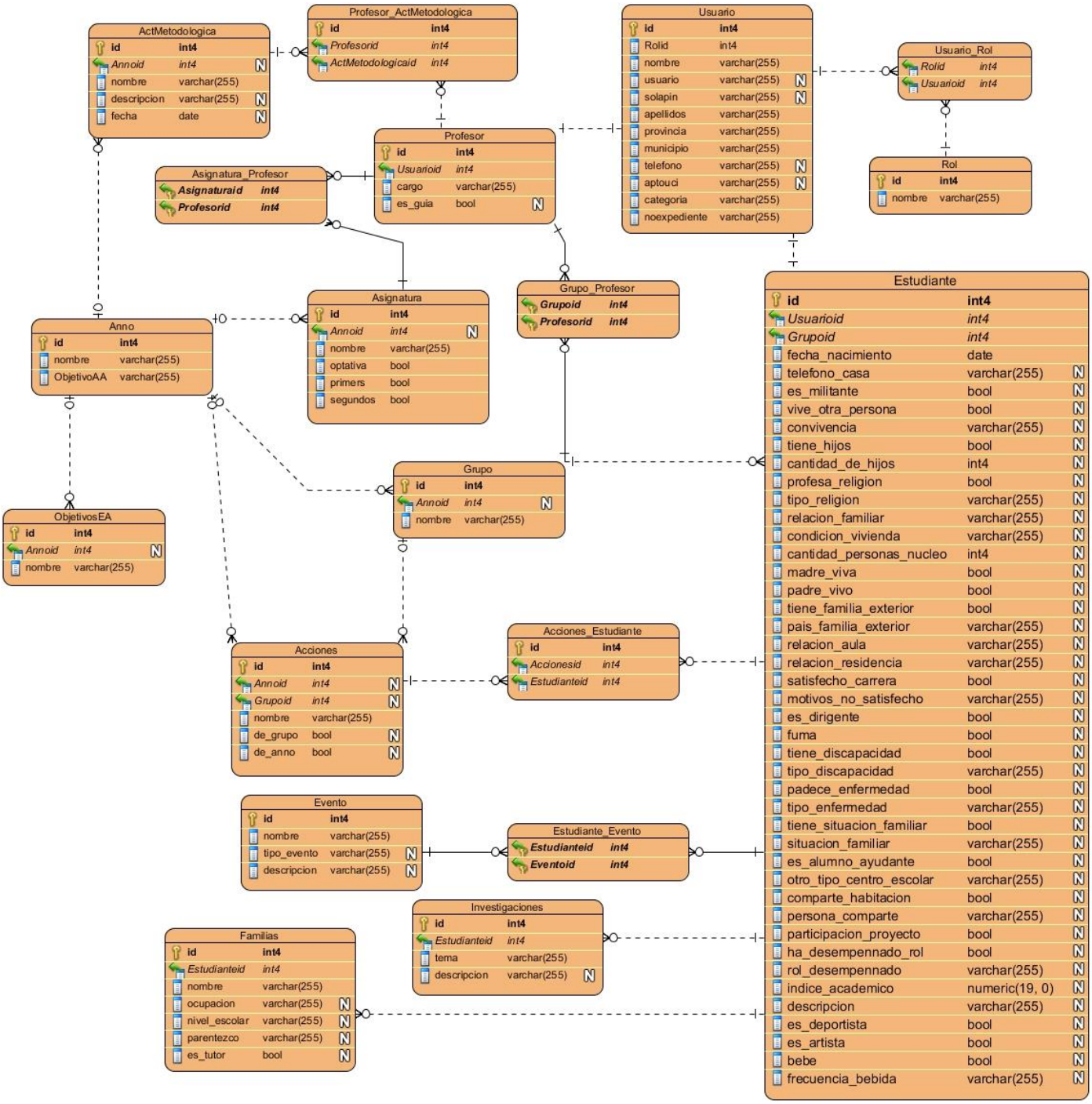



Figura 8. Modelo de datos del sistema.

Como una buena práctica que le proporciona al ingeniero de software una medida de la calidad con que realizó el diseño propuesto se realiza la tarea de validación del mismo. A continuación se describen las métricas a utilizar así como los resultados obtenidos.

### 2.3.6. Métricas para la validación del diseño

Para la validación del diseño se realizó un estudio de las métricas existentes, escogiéndose para la misma las métricas Tamaño operacional de clase y Relaciones entre clases debido al conjunto de atributos de calidad de diseño que miden.

**Tamaño operacional de clase (TOC)**

Se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado.

Tabla 6. Tamaño operacional de clase (TOC).

| Atributo que afecta | Modo en que lo afecta |
|---|---|
| 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 en el grado de reutilización de la clase. |

Tabla 7. Rango de valores para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC.

| Atributos | Categoría | Criterio |
|---|---|---|
| Responsabilidad | Baja | < =Prom. (PO) |
| | Media | Entre Prom. Y 2* Prom |
| | Alta | > 2* Prom |
| | Baja | < =Prom |

| | | |
|---|---|---|
| Complejidad de implementación | Media | Entre Prom. y 2* Prom |
| | Alta | > 2* Prom |
| Reutilización | Baja | > 2*Prom |
| | Media | Entre Prom. y 2* Prom |
| | Alta | <= Prom |

Tabla 8. Tamaño de las clases del sistema.

| No | Clases | No. Atributos | No. Operaciones |
|---|---|---|---|
| 1 | AsignaturaController | | 10 |
| 2 | ProfesorController | | 10 |
| 3 | EstudianteController | | 10 |
| 4 | GrupoController | | 10 |
| 5 | AccionesController | | 10 |
| 6 | ObjetivosController | | 10 |
| 7 | ActMetodologicasController | | 10 |
| 8 | FamiliarController | | 10 |
| 9 | EventosController | | 10 |
| 10 | ParticipacionEAController | | 8 |
| 11 | InvestigacionController | | 10 |
| 12 | UsuarioController | | 10 |
| 13 | RolesController | | 10 |
| 14 | EventosController | | 10 |
| 15 | Asignatura | 6 | 11 |
| 16 | Grupo | 3 | 5 |

| | | | |
|---|---|---|---|
| 17 | Estudiante | 41 | 81 |
| 18 | Profesor | 3 | 5 |
| 19 | Objetivos | 3 | 5 |
| 20 | Anno | 3 | 5 |
| 21 | Acciones | 6 | 11 |
| 22 | Usuario | 12 | 23 |
| 23 | ActMetodologicas | 4 | 7 |
| 24 | Familias | 7 | 13 |
| 25 | Investigaciones | 4 | 7 |
| 26 | Eventos | 4 | 7 |
| 27 | Roles | 3 | 5 |
| **Total** | | **99** | **323** |

En la figura 9 se puede apreciar la incidencia de los resultados para el atributo Responsabilidad según la evaluación de la métrica TOC.

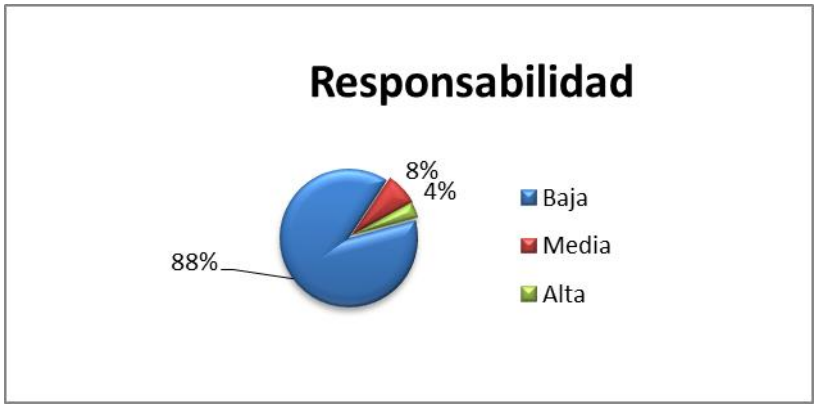



Figura 9. Resultado del atributo Responsabilidad.

En la figura 10 se puede apreciar la incidencia de los resultados para el atributo Complejidad de implementación según la evaluación de la métrica TOC.

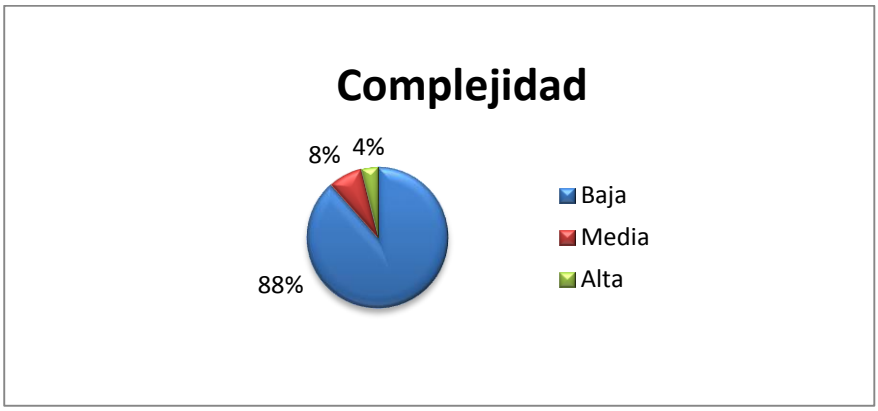



Figura 10. Resultado del atributo Complejidad de implementación.

En la figura 11 se puede apreciar la incidencia de los resultados para el atributo Reutilización según la evaluación de la métrica TOC.


Figura 11. Resultado del atributo Reutilización.

Como resultado de la aplicación de la métrica TOC se puede apreciar de manera general que el 88% de las clases del diseño tienen baja responsabilidad. Esto significa que cuentan con menor grado de complejidad de implementación, lo que favorece notablemente la reutilización de las clases. Mientras que el 12% restante, presenta una responsabilidad y complejidad algorítmica alta, disminuyendo la posibilidad de ser reutilizables.

**Relaciones entre clases (RC)**

Dado por el número de relaciones de uso de una clase. Está determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Cantidad de pruebas y Reutilización, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

Tabla 9. Atributos de calidad evaluados por la métrica RC

| Atributo de calidad | Modo en que lo afecta |
|---|---|
| Acoplamiento | Un aumento del RC implica un aumento del Acoplamiento de la clase. |
| Complejidad de mantenimiento | Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase. |
| Reutilización | Un aumento del RC implica una disminución en el grado de reutilización de la clase. |
| Cantidad de pruebas | Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase. |

Tabla 10. Criterios de evaluación para la métrica RC

| Atributo | Categoría | Criterio |
|---|---|---|
| Acoplamiento | Ninguno | 0 |
| | Bajo | 1 |
| | Medio | 2 |
| | Alto | > 2 |
| | Baja | < =Prom. |

| | | |
|---|---|---|
| Complejidad de mantenimiento | Media | Entre Prom y 2* Prom. |
| | Alta | > 2* Prom. |
| Reutilización | Baja | > 2*Prom. |
| | Media | Entre Prom y 2* Prom |
| | Alta | <= Prom. |
| Cantidad de pruebas | Baja | < =Prom. |
| | Media | Entre Prom y 2* Prom. |
| | Alta | > 2* Prom. |

**Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC).**

Representación en porciento de los resultados obtenidos en el instrumento agrupados en los intervalos definidos, ver Figura 12:

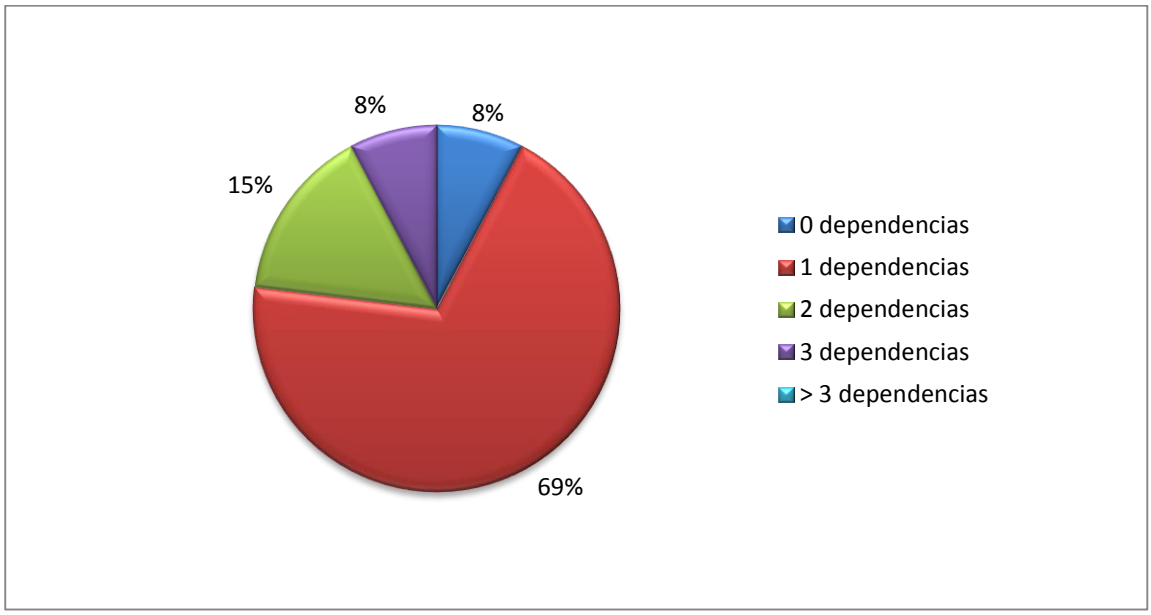



Figura 12. Porciento de los resultados obtenidos

En la figura 13 se puede apreciar la incidencia de los resultados para el atributo Acoplamiento según la evaluación de la métrica TOC.

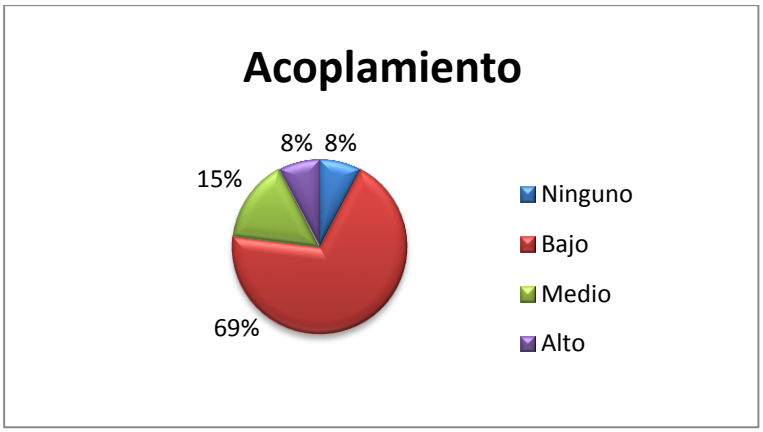



Figura 13. Resultado del atributo Acoplamiento.

En la figura 14 se puede apreciar la incidencia de los resultados para el atributo Complejidad de mantenimiento según la evaluación de la métrica TOC.

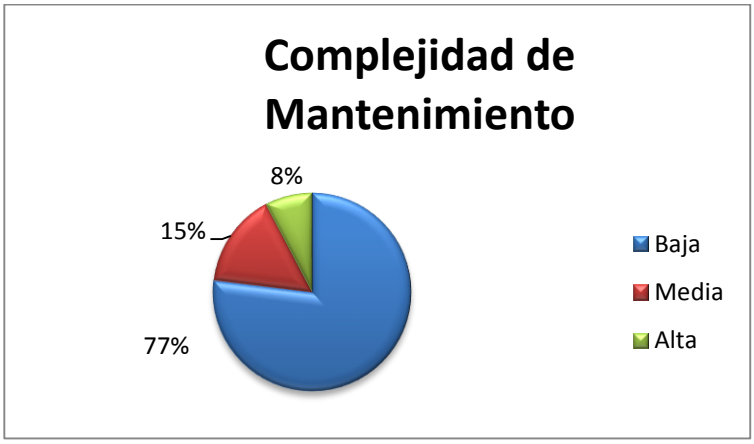



Figura 14. Resultado del atributo Complejidad de mantenimiento.

En la figura 15 se puede apreciar la incidencia de los resultados para el atributo Cantidad de pruebas según la evaluación de la métrica TOC.

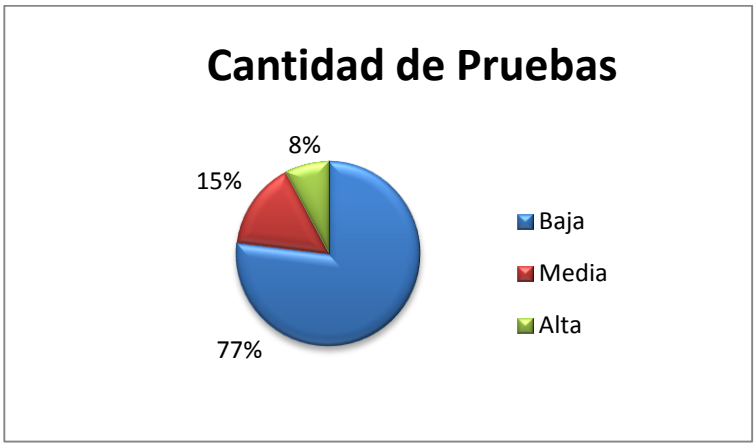



Figura 15. Resultado del atributo atributo Cantidad de pruebas.

En la figura 16 se puede apreciar la incidencia de los resultados para el atributo Reutilización según la evaluación de la métrica TOC.

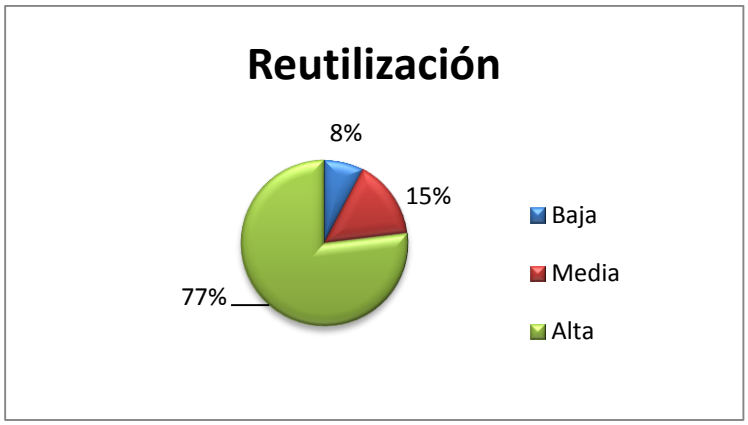



Figura 16. Resultado del atributo Reutilización.

Los resultados obtenidos muestran que el 69% de las clases presentan bajo acoplamiento. Esto propicia que el 77% de las clases no tengan gran complejidad de mantenimiento, lo que permite un alto grado de reutilización y que se requieran pocas pruebas para probarlas. Sin embargo, el 23% restante presenta un acoplamiento alto, lo que provoca que la complejidad de mantenimiento y la cantidad de pruebas unitarias también sea mayor, por lo que su reutilización es menor en comparación con el resto de las clases.

**Conclusiones parciales del capítulo**

- La elaboración del modelo conceptual permitió un mayor entendimiento del dominio del problema, los principales conceptos y sus relaciones.
- Mediante la disciplina de Requisitos se obtuvieron los requisitos funcionales y no funcionales del sistema, validándose satisfactoriamente por el cliente.
- A través de los artefactos generados en la disciplina de Análisis y diseño se logró un mejor entendimiento de cómo se realizará la implementación del sistema.
- Con la utilización de las métricas TOC y RC se obtuvo como resultado que el diseño es sencillo y posee una calidad aceptable.

# Capítulo 3. Implementación y pruebas

## Introducción

El presente capítulo se realizan las disciplinas: Implementación, Pruebas internas, de liberación y aceptación propuestas por la metodología seleccionada. Como parte de la implementación se muestra el modelo de despliegue realizado para el sistema y se especifican los estándares de codificación para un mayor entendimiento del código. Se describen las pruebas de caja negra y caja blanca realizadas al software y los resultados obtenidos con el fin de validar que el sistema desarrollado posea la mayor calidad posible. Como parte de las pruebas se genera el artefacto de Diseño de casos de pruebas. Finalmente se realiza la validación de las variables de la investigación.

A continuación muestra el Modelo de despliegue.

## 3.1. Modelo de despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Muestra la configuración de los elementos de hardware (nodos) y cómo los elementos y artefactos del software se trazan entre estos. Permite modelar las relaciones físicas de los distintos elementos que componen un sistema y el reparto de los componentes sobre dichos nodos (2000).

Para el modelo de despliegue realizado se definió una arquitectura cliente-servidor de la siguiente manera:

**PC Cliente**: es una computadora en la cual la aplicación se ejecutará a través de un navegador, en este caso se debe usar el Mozilla Firefox.

**Servidor Web**: servidor donde radica la lógica de negocio de la aplicación. Servidor web Apache 2.2, utilizando el lenguaje de programación del lado del servidor PHP 5.

**Servidor de Autenticación**: servidor a través del cual se realiza la autenticación de los usuarios del sistema pertenecientes al dominio uci.cu, en aras de lograr que no accedan al sistema usuarios no autorizados.

**Servidor de base de datos**: servidor de datos PostgreSQL 9.2, donde se encuentra la base de datos que utiliza el sistema, este puede estar instalado en la misma computadora donde se encuentra el servidor web.

**Impresora**: utilizada para imprimir los reportes del sistema en caso de ser necesario.

En la figura 17 se muestra el modelo de despliegue elaborado para el sistema.

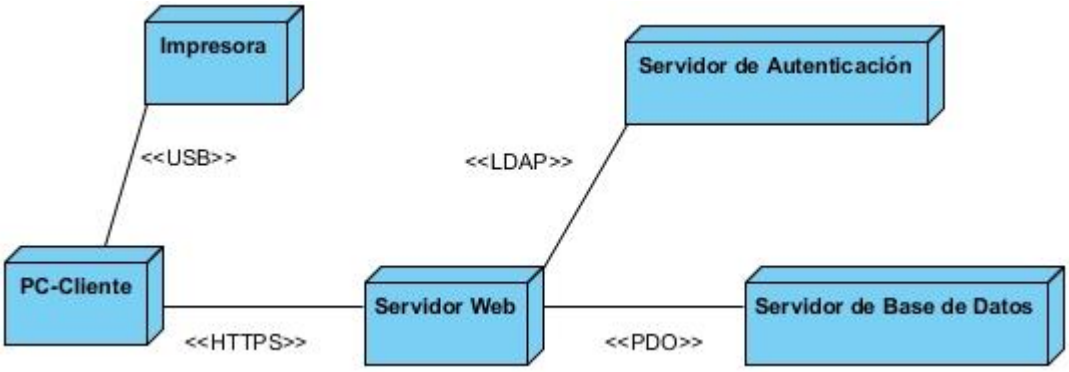



Figura 17. Modelo de despliegue del sistema.

### 3.2. Estándares de codificación

Los estándares de codificación son una guía que define la estructura y composición sobre la cual se debe regir la codificación de una solución informática. Su objetivo fundamental es lograr un correcto, común y comprensible formato del código que compone un sistema. Estos estándares deben servir de apoyo a quien interactúa con dicho código para identificar de forma sencilla cuál es el objetivo y las funcionalidades que brinda cada una de las clases, funciones y demás componentes de software. Además, debe servir de guía para posteriores implementaciones y/o modificaciones del sistema (Garlan, 1996).

En general, un estándar de codificación son reglas que se siguen para la escritura del código fuente. De tal manera que a otros programadores se les facilite entender el código (como identificar las variables, las funciones o métodos) (Garlan, 1996).

Se describen a continuación los estándares de codificación empleados durante la implementación de la solución propuesta:

**PascalCasing**

El estándar PascalCasing establece que los nombres de clases están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula (Benjamín, 2008). La nomenclatura de las clases se realizó sobre la base de este estándar, usando palabras compuestas sugerentes acordes al propósito de la misma.

Nomenclatura de clases según su tipo

EstudianteController, Estudiante, Asignatura, GrupoType.

**CamelCasing**

El estándar CamelCasing es parecido al PascalCasing con la particularidad de que la letra inicial del identificador no comienza con mayúscula (Benjamín, 2008). Esta notación se utilizó para el nombre de funciones.

Nomenclatura de las funciones

El identificativo a emplear para las funciones o métodos se escribe con la primera palabra en minúscula utilizando la notación CamelCasing y nombres que deduzcan su propósito. Ejemplo: obtenerAccionesEntreFechas. Los denominadores de las acciones de las clases controladoras tienen la peculiaridad de ir seguidos por la palabra Action. Ejemplo: estrategiaAction.

**Underscore Separated**

Esta notación indica palabras separadas con guión bajo (Infocorp, 2011) y se utilizó para nombrar las variables.

Nomenclatura de las variables

Las variables se nombran convenientemente de acuerdo con la notación Underscore Separated y en minúscula siempre. Ejemplo: $fecha_nacimiento, $telefono_uci.

### 3.3. Pruebas de software

Uno de los instrumentos para determinar el estatus de la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos (Pressman, 2006).

De acuerdo con la disciplina Pruebas internas de la metodología utilizada, se describe la prueba de caja blanca realizada al sistema, mostrándose los resultados obtenidos en la misma.

#### 3.3.1. Prueba de Caja Blanca

Las pruebas de caja blanca, también conocidas como pruebas de caja transparente o pruebas estructurales, se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. (Pressman, 2006).

De las técnicas de caja blanca se tuvo en cuenta la prueba del camino básico, la cual se especifica a continuación.

#### 3.3.1.1. Técnica del camino básico

Este método (propuesto por McCabe) permite obtener una medida de la complejidad de un diseño procedimental y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución, diseñando casos de prueba que garanticen que cada camino se ejecute al menos una vez. Primeramente se hace necesario el cálculo de la complejidad ciclomática del algoritmo que vaya a ser analizado, para lo cual se deben enumerar las sentencias de código de este y a partir de ahí elaborar el grafo de flujo de esta funcionalidad (Pressman, 2006).

El grafo de flujo está compuesto de (Pressman, 2006):

**Nodos**: son los círculos representados en el grafo de flujo, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

**Aristas**: son constituidas por las flechas del grafo y representan el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.

**Regiones**: son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

Esta prueba se le realizó a todos los métodos de la aplicación, obteniéndose resultados satisfactorios. A continuación se muestra un ejemplo de la ejecución de la prueba en el método nEventosAction(). Su uso se debe a la complejidad que presenta con respecto a los demás en cuanto a ciclos y condicionales.

En la figura 18 se encuentran las sentencias enumeradas del método nEventosAction().

```
public function nEventosAction(Request $request) {
    $em = $this->getDoctrine()->getManager();
    $n_eventos = $request->request->get('n_eventos');
    $id_anno = $request->request->get('id_anno');
    $entities = $em->getRepository('SGEEBundle:Estudiante')->findByAnno($id_anno);
    $mostrar = array();
    foreach ($entities as $valor) {
        $connection = $em->getConnection();
        $statement = $connection->prepare('SELECT * FROM participacionee WHERE
        estudiante_id =:est_id AND realizado = TRUE');
        $statement->bindValue(':est_id', $valor->getId());
        $statement->execute();
        $result = $statement->fetchAll();
        $cant = count($result);
        if ($cant == $n_eventos) {
            array_push($mostrar, $valor);
        }
    }
    return array('entities' => $mostrar, 'id_anno' => $id_anno);
}
```

Figura 18. Código fuente de la funcionalidad nEventosAction().

En la figura 19 se muestra el grafo de flujo asociado a la funcionalidad nEventosAction().

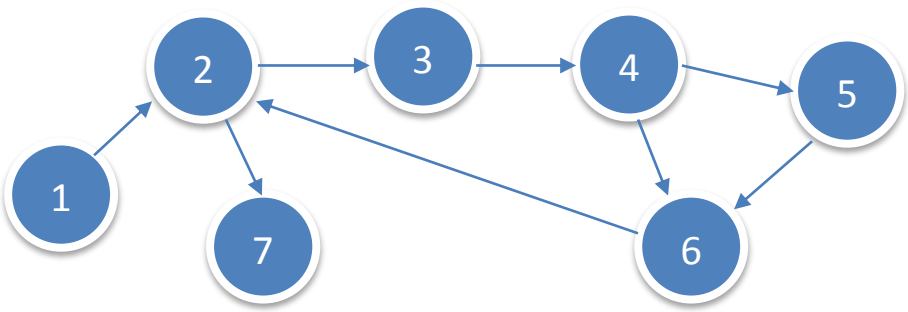



Figura 19. Grafo de flujo asociado a la funcionalidad **nEventosAction()**.

**Cálculo de la complejidad ciclomática a partir de un segmento de código**

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y ofrece un límite superior para el número de pruebas que se deben realizar para asegurar que sea ejecutada cada sentencia al menos una vez (Pressman, 2006).

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación (Pressman, 2006), las cuales deben dar el mismo resultado para asegurar que el cálculo de la complejidad sea el correcto. Las fórmulas para realizar dicho cálculo son:

1. **V (G) = (A - N) + 2**

Donde **"A"** es la cantidad de aristas y **"N"** la cantidad de nodos.

V (G) = (8 - 7) + 2

V (G) = 3

2. **V (G) = P + 1**

Siendo **"P"** la cantidad de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 2 + 1

V (G) = 3

3. **V (G) = R**

Donde **"R"** representa la cantidad de regiones en el grafo.

V (G) = 3

El cálculo efectuado mediante las fórmulas ha resultado ser el mismo, por lo que se puede determinar que la complejidad ciclomática del código analizado es 3, lo que significa que existen tres posibles caminos por donde el flujo puede circular. Según (Pressman, 2006) este valor representa el límite superior del número total de casos de pruebas que se le pueden aplicar al código.

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

**Camino básico # 1:** 1-2-7

**Camino básico # 2:** 1-2-3-4-5-6-2-7

**Camino básico # 3:** 1-2-3-4-6-2-7

Para cada uno de los caminos obtenidos se realiza un caso de prueba. Es necesario seleccionar los datos de manera tal que se establezcan apropiadamente las condiciones de los nodos predicado a medida que se prueba cada ruta. Cada caso de prueba se ejecuta y compara con los resultados esperados. Una vez completados todos los casos, la persona que aplica la prueba puede estar segura de que todas las instrucciones del programa se han ejecutado por lo menos una vez (Pressman, 2006). Los casos de prueba realizados son los siguientes:

***Caso de prueba para el camino básico # 3***

**Camino:** 1-2-3-4-6-2-7

**Descripción**: el dato que se obtiene por el método POST y los que se calculan a partir de este, cumplen con los siguientes requisitos:

$id_anno no es nulo, $n_eventos contiene la cantidad de eventos que se desea analizar, $entities contiene todos los Estudiantes del año que contiene la variable $id_anno. La variable $valor va a tomar un objeto estudiante por cada iteración de la variable $entities. Por cada ciclo la variable $result va a obtener todas las participaciones en eventos de los estudiantes. $cant contiene la cantidad de objetos que tiene $result en este caso un valor diferente a $n_eventos.

**Entrada:** $id_anno = uno de estos valores (1, 2, 3, 4, 5), $n_eventos = un número del 0 al 20.

**Resultados esperados:** se espera que no se guarden datos en el arreglo $mostrar y que este sea retornado al igual que la variable $id_anno para esta ruta.

**Resultados obtenidos:**

Satisfactorio. $mostrar = array(), $id_anno = uno de estos valores (1, 2, 3, 4, 5).

***Caso de prueba para el camino básico # 2***

**Camino:** 1-2-3-4-5-6-2-7

**Descripción**: el dato que se obtiene por el método POST y los que se calculan a partir de este, cumplen con los siguientes requisitos:

$id_anno no es nulo, $n_eventos contiene la cantidad de eventos que se desea analizar, $entities contiene todos los Estudiantes del año que contiene la variable $id_anno. La variable $valor va a tomar un objeto estudiante por cada iteración de la variable $entities. Por cada ciclo la variable $result va a obtener todas

las participaciones en eventos de los estudiantes. $cant contiene la cantidad de objetos que tiene $result en este caso un valor igual a $n_eventos. Por último se le adiciona la variable $valor al arreglo $mostrar.

**Entrada:** $id_anno = uno de estos valores (1, 2, 3, 4, 5), $n_eventos = un número del 0 al 20.

**Resultados esperados:** se espera que se adicione el objeto de la variable $valor en el arreglo $mostrar y que este último sea retornado al igual que la variable $id_anno para esta ruta.

**Resultados obtenidos:**

Satisfactorio. $mostrar = array(¨$valor,…¨) , $id_anno = uno de estos valores (1, 2, 3, 4, 5).

***Caso de prueba para el camino básico # 1***

**Camino:** 1-2-7

**Descripción**: esta ruta no se puede probar por separado ya que en el flujo normal del programa no puede obtenerse la combinación de los datos requeridos para recorrer la ruta, por lo cual se ejercita como parte de otra prueba del camino (Pressman, 2006). El dato que se obtiene por el método POST y los que se calculan a partir de este cumple con los siguientes requisitos:

$id_anno no es nulo, $mostrar va a obtener valores según se ejecuten las demás rutas.

**Entrada:** $id_anno = uno de estos valores (1, 2, 3, 4, 5), $n_eventos = un número del 0 al 20.

**Resultados esperados:** retornar los valores de las variables $id_anno y $mostrar.

**Resultados obtenidos:**

Satisfactorio. $mostrar = array(¨…¨), $id_anno = uno de estos valores (1, 2, 3, 4, 5).

Con la aplicación de los casos de prueba expuestos anteriormente se comprobó que el flujo de trabajo de las funcionalidades es correcto, se aseguró que cada sentencia es ejecutada al menos una vez, cumpliéndose así las condiciones de la prueba.

La realización de esta prueba evidenció que todos los métodos de la aplicación funcionan correctamente. Una vez probados los métodos de forma independiente, corresponde comprobar las funcionalidades del sistema mediante la disciplina Pruebas de liberación que propone la metodología seleccionada. A continuación se describe la técnica de caja negra a utilizar y los resultados obtenidos en la aplicación de la misma.

### 3.3.2. Prueba de Caja Negra

Las pruebas que se realizan al sistema para la validación de sus funcionalidades son pruebas de caja negra. Estas pruebas se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del sistema (Pressman, 2006). Para este método de prueba se hará uso de la técnica Partición equivalente, que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba (Pressman, 2006). Para cada uno de los requisitos funcionales fueron definidos casos de prueba. A continuación se muestra el artefacto Caso de prueba para el requisito Mostrar acciones realizadas por año en un rango de fechas.

**Descripción general**

Descripción_de_requisito(Mostrar acciones realizadas por año en un rango de fechas).doc

**Condiciones de ejecución**

Se debe identificar y autenticar ante el sistema, además, debe tener los permisos para ejecutar esta acción.

Se debe seleccionar la opción Reportes del Año que se desea verificar

Se debe seleccionar la opción del submenú Acciones Realizadas

**Mostrar acciones realizadas por año en un rango de fechas**

| Escenario | Descripción | Fecha1 | Fecha2 | Respuesta del sistema | Flujo central |
|---|---|---|---|---|---|
| EC 1.1 Mostrar las acciones introduciendo datos válidos | El sistema permite mostrar las acciones insertando las fechas de menor a mayor. | V<br><br>01/05/2015 | V<br><br>05/05/2015 | Mostrar todas las acciones en los diferentes rangos de fechas | Insertar cada fecha de menor a mayor en los campos de texto y presionar el botón **Buscar** |
| EC 1.2 Mostrar las acciones introduciendo datos obligatorios vacíos | El sistema no permite mostrar las acciones dejando campos obligatorios vacíos. | I<br><br>Vacío | I<br><br>Vacío | Se muestra un mensaje informando del error "Los datos de las fechas son obligatorios" | Dejar los campos de las fechas vacíos y presionar el botón **Buscar** |
| EC 1.3 Mostrar las acciones introduciendo datos incorrectos | El sistema no permite mostrar las acciones insertando las fechas de mayor a menor. | I<br><br>05/05/2015 | I<br><br>01/05/2015 | Se muestra un mensaje informando del error "Las fechas deben estar de menor a mayor" | Insertar cada fecha de mayor a menor en los campos de texto y presionar el botón **Buscar** |

Figura 20. Artefacto Caso de prueba para el requisito Mostrar acciones realizados por año en un rango de fechas.

**Descripción de las variables.**

| No | Nombre de campo | Clasificación | Valor Nulo | Descripción |
|---|---|---|---|---|
| 1 | fecha1 | Campo de texto | No | N/A |
| 2 | fecha2 | Campo de texto | No | N/A |

Figura 21. Artefacto Descripción de las variables para el caso de prueba del requisito Mostrar acciones realizados por año en un rango de fechas.

De acuerdo a los casos de prueba definidos y el listado de requisitos, la aplicación fue probada por el grupo de calidad del Centro de Informatización de Entidades (CEIGE). Los resultados obtenidos en cada una de las iteraciones se muestran en la Figura 22.

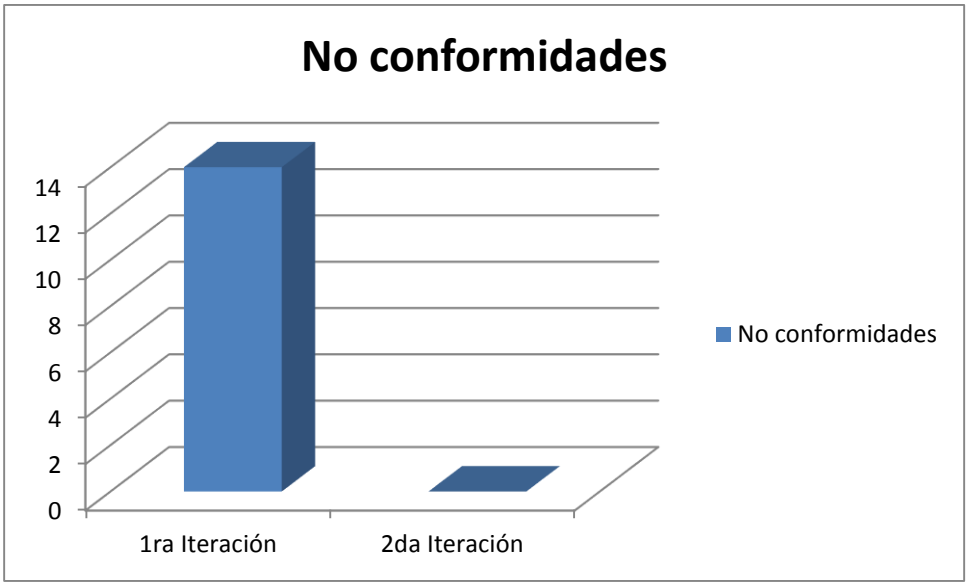



Figura 22. Resultados de las pruebas de liberación por el grupo de calidad del centro CEIGE.

En la primera iteración fueron identificadas 12 no conformidades de aplicación y 2 de documentación. Dentro de las no conformidades referentes a la aplicación se encontraron 10 de interfaz y 2 de funcionalidad siendo estas últimas las más significativas. En cuanto a las de documentación, las 2 encontradas fueron de redacción. Todas estas no conformidades fueron erradicadas una vez detectadas, lo que permitió que la aplicación pasara a una segunda iteración en la que no se detectaron no conformidades, obteniéndose resultados satisfactorios, lo que permitió validar el correcto funcionamiento de la aplicación realizada. Como salida de esta prueba se emitió el Acta de liberación por parte del grupo de calidad del CEIGE.

### 3.4. Pruebas de aceptación

Hace una comparación entre el comportamiento del sistema con los requisitos del cliente. El cliente se encarga de realizar, o especificar, las tareas típicas para verificar que se cumplen sus requisitos o que el grupo de trabajo los ha identificado para el mercado adonde se va a dirigir el software. En estas pruebas se puede incluir o no a los desarrolladores del sistema.

Para estas pruebas el sistema fue revisado por la asesora docente metodológica y las profesoras principales de los años académicos de la Facultad 3. Como resultado de las mismas se emitió el Acta de aceptación por parte de los clientes.

### 3.5. Validación de las variables de la investigación

Para desarrollar la investigación fueron identificados problemas en el control y seguimiento de la información en la gestión de las estrategias educativas en la facultad 3. Estas dificultades fueron dadas por:

- Las estrategias educativas se realizan mediante un formato establecido por la universidad y son guardadas por el profesor guía de la brigada, no existiendo un lugar centralizado donde tanto la brigada como el resto del colectivo pedagógico puedan consultar los datos referentes a las mismas.
- La notificación de las actividades es realizada por el profesor guía y en algunos casos por los responsables de la actividad, lo que conlleva a que muchas veces la actividad no se realice o se haga fuera de la fecha planificada debido a que no se convocó a la misma en tiempo.
- La información que generan las estrategias educativas para cada nivel (brigada, año, facultad), son gestionadas manualmente por los responsables (profesores principales del año y profesores guías), dificultándose la creación de reportes que permitan dar una vista global o parcial del proceso en toda la facultad, afectándose además, el control y seguimiento del proceso.

En la Tabla 11 se describen cómo fueron erradicados estos problemas con la implementación de la solución propuesta.

| Variable | Proceso actual en la facultad | Proceso con la utilización del sistema implementado |
|---|---|---|
| Control y seguimiento del proceso | La estrategia educativa se encuentra en formato duro y digital según el modelo definido por la universidad. Es guardada en el vicedecanato de formación y por los responsables a cada nivel, no existiendo | El sistema recoge todos los datos necesarios para la conformación de las estrategias educativas, desde la caracterización estudiantil y del colectivo pedagógico, hasta las acciones |

| | | |
|---|---|---|
| | un lugar donde estudiantes y profesores puedan consultar la misma en cualquier momento. | planificadas. Esta información queda almacenada en base de datos, por lo que puede ser consultada en cualquier momento.<br><br>El sistema, además, genera como reporte la plantilla establecida por la universidad para las estrategias educativas con toda la información necesaria. |
| | La notificación de las actividades se realiza por parte de los responsables de las mismas, lo que conlleva en ocasiones a que se incumpla con la fecha establecida. | El sistema gestiona la notificación a los involucrados en las actividades, enviando la misma con dos días de antelación. Además, controla los estados por lo que transitan las actividades: en espera, realizada e incumplida. |
| | La información del cumplimiento de las acciones de la estrategia educativa es llevada de forma manual por parte de los responsables, dificultándose la realización de los reportes para el seguimiento del proceso. | En el sistema, además de guardarse los datos inicialmente recogidos en las estrategias, se gestiona la actualización de las acciones en el tiempo, almacenándose la participación en actividades y eventos.<br><br>Además, se cuenta con un conjunto de reportes que apoyan el seguimiento, control y toma de decisiones sobre el proceso. |

Mediante la comparación realizada se puede apreciar que la solución propuesta resuelve los problemas planteados en la situación problémica, cumpliéndose con el objetivo general de la solución.

**Conclusiones parciales del capítulo**

- Los estándares de codificación utilizados permitieron realizar una implementación legible y entendible, para su futuro mantenimiento.
- Se realizó el diagrama de despliegue, el cual permitió comprender como funcionará el sistema una vez desplegado.
- Los resultados obtenidos en las pruebas de caja blanca demostraron que no existe código innecesario dentro del sistema implementado. Los resultados obtenidos en las pruebas de caja negra validaron que las funcionalidades desarrolladas son correctas.
- Se validaron las variables que forman parte del problema de la investigación, demostrándose que con el sistema se contribuye a elevar el control y seguimiento del proceso de gestión de las estrategias educativas de la Facultad 3.

## Conclusiones generales

Finalizado el desarrollo de la presente investigación se concluye lo siguiente:

- La realización del marco teórico arrojó como resultado la necesidad de implementar un sistema que dé solución a la problemática planteada, definiéndose para ello las tecnologías adecuadas.
- La metodología escogida sirvió de guía en el proceso de desarrollo donde se obtuvo una aplicación informática con funcionalidades que dan respuesta a los requisitos definidos, permitiendo realizar la gestión de las estrategias educativas en la Facultad 3.
- Los resultados obtenidos en las pruebas internas, de liberación y de aceptación mostraron que las funcionalidades desarrolladas son correctas, validándose el funcionamiento de la aplicación.
- Se validaron las variables que forman parte del problema de la investigación, demostrándose que con el sistema se contribuye a elevar el control y seguimiento del proceso de gestión de las estrategias educativas de la Facultad 3.

## Recomendaciones

Analizando los resultados obtenidos en la investigación y la experiencia adquirida durante el desarrollo de la misma, se recomienda:

- Desplegar el sistema en la Facultad 3 y extender su uso a toda la comunidad universitaria.
- Incorporar al sistema el proceso de la evaluación integrada de los estudiantes.

## Bibliografía

**A. J. MENEZES, P. OORSCHOT. 1996.** Handbook of Applied Cryptography. s.l. : Disponible en: http://www.cacr.math.uwaterloo.ca/hac/, 1996. ISBN 0-8493-8523-7.

**Ambysoft Inc. 2006.** [En línea] Ambysoft Inc., 13 de Mayo de 2006. [Citado el: 2 de Junio de 2015.] http://www.cc.una.ac.cr/AUP/.

ARGENTINA, C. C. N. CCN -CERT Argentina. s.l. : Disponible en: https://www.ccn-cert.cni.es/.

**Barbán, Andrés. 2014.** *Sistema para la informatización del proceso de Gestión de la caracterización estudiantil en la Facultad 3.* La Habana, Cuba : Universidad de las Ciencias Informáticas , 2014.

**Barroso. 2009.** *Propuesta de pautas para el diseño de un Sistema de Gestión de Información en la empresa ECIMETAL. Tesis de Licenciatura.* Habana : Departamento de Bibliotecología y Ciencia de la Información, Universidad de La Habana, 2009.

**Benítez, Lic. Idarmis González. 2000.** Scielo. *Scielo.* [En línea] 6 de Enero de 2000. [Citado el: 26 de Enero de 2014.] http://scielo.sld.cu/scielo.php?pid=S0864-21412000000100008&script=sci_arttext.

**Benjamín. 2008.** codigolinea. [En línea] 25 de Mayo de 2008. [Citado el: 02 de Junio de 2015.]

**C. GUTIÉRREZ, E. FERNÁNDEZ-MEDINA. 2004.** A Survey of Web Services Security. Springer Berlin / Heidelberg : s.n., 2004. Vol. 3043/2004, Disponible en: http://www.springerlink.com/. ISBN 978-3-540-22054-1.

**CAVSI.com. 2007.** ¿Qué es un Sistema Gestor de Bases de Datos o SGBD? [En línea] 2007. [Citado el: 6 de marzo de 2012.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

**Datasae Ltda. 2001.** SIGA - WEB. [En línea] Datasae Ltda, 2001. [Citado el: 05 de 01 de 2015.] http://www.siga.com.co/sitio/.

**2012.** ECURED. [En línea] 16 de Abril de 2012. www.ecured.cu.

**Falgueras, Benet Campderrich. 2003.** *Ingenieria del Software.* 2003.

**Fowler, Martin.** *Patterns of Enterprise Application Architecture.* ISBN: 0-32112-742-0.

**Gacitúa Bustos, Ricardo A. 2003.** *Métodos de desarrollo de software: El desafío pendiente de la estandarización.* Chillán, Chile : Theoria, 2003.

**Garlan, D y Shaw, M. 1996.** *Software Architecture: Perspectives on an Emerging Discipline.* 1996.

**González, Oscar. 2013.** codehero. [En línea] 28 de 10 de 2013. [Citado el: 15 de 05 de 2015.] http://codehero.co/jquery-desde-cero-introduccion/.

**Gracia, Joaquin. 2005.** IngenieroSoftware. [En línea] 27 de Mayo de 2005. [Citado el: 16 de Mayo de 2015.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

**IEEE. 1990.** IEEE Standard Glossary of Software Engineering Terminology. [En línea] 1990. [Citado el: 5 de marzo de 2012.] http://standards.ieee.org/findstds/standard/610.12-1990.html.

**INDER. CINID WEB. 2014.** Sitio Oficial del Instituto Nacional de Deportes, Educación Física y la Recreación. [En línea] 2014. http://www.inder.cu/FichaAtleta/documentacion/index.html.

**Infocorp. 2011.** *Estándar de nomenclatura para Base de Datos.* s.l. : Grupo Apoyo Técnico, Infocorp, 2011.

**Innova., Grupo Soluciones.** [En línea] Rational Unified Process. [Citado el: 3 de Marzo de 2012.] http://www.rational.com.ar/herramientas/rup.html.

**Javier Llácer Muñoz, Carlos Martínez Burgos y Sergio Catalá Gil.** *Informe de evaluación de ERP.*

**Koch, Nora, María José Escalona. 2002.** Departamento de Lenguajes y Sistemas Informáticos. [En línea] Diciembre de 2002. [Citado el: 05 de Mayo de 2015.] https://www.lsi.us.es/docs/informes/LSI-2002-4.pdf.

**Larman, Craig. 1999.** *UML y Patrones.* 1999. ISBN 970-1 7-0261-1.

**López, M. J. 2003.** *Criptografía y Seguridad en Computadores.* s.l. : 3ra Edición, 2003.

**Lorenzo, Yaniesis. 2014.** *Estrategia para el trabajo educativo de 4to año facultad 3.* Habana : Universidad de las Ciencias Informáticas, 2014.

**M. GERBER, R. V. SOLMS. 2001.** Formalizing information security requirements. Disponible en http://www.emeraldinsight.com/10.1108/09685220110366768 : s.n., 2001. Vol. 9. ISBN 0968-5227.

**MachángaraSoft.** RUTATEC. [En línea] http://www.rutatec.com/pagina/index.php/productosservicios/sgescolar.

**Martínez, Ing. Jenni Manso. 2010.** *Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE.* La Habana : Universidad de las Ciencias Informáticas : s.n., 2010.

**Matraspa, Verdecia D I &. 2008.** *Procedimiento para el modelado de los procesos del Negocio y la Captura de Requisitos del Software a la medida desarrollados en la UCI.* 2008.

**Ministerio de Educación Superior. 2014.** *Perfeccionamiento del sistema de gestión del proceso de formación integral de los estudiantes universitarios en el eslabón de base.* s.l. : Universitaria Felix Varela, 2014.

**Noguera, García Manuel, Kawtar Benghazi, José Luis Garrido Bullejos. 2011.** UGR | Universidad de Granada. [En línea] 2011. [Citado el: 8 de Mayo de 2015.] http://www.ugr.es/~mnoguera/collaborative_systems-business_processes_10-11.pdf.

**Pantoja, Ernesto Bascón. 2004.** *El patrón de diseño modelo -vista - controlador (MVC) y su implementación el Java Swing.* 2004.

**Peña, Marieta. 2014.** *Estrategia para el trabajo educativo de 5to año facultad 3.* Habana : Universidad de las Ciencias Informáticas, 2014.

**Potencier, Fabien y Zaninotto, François. 2008.** *Symfony la guía definitiva.* 2008.

**Pressman, Roger. 2006.** *Ingeniería de Software: Un enfoque práctico.* México DF : s.n., 2006. Vol. 6ta Edición.

Pruebas de Software. *Pruebas de Software.* [En línea] [Citado el: 1 de Mayo de 2014.] http://www.pruebasdesoftware.com/laspruebasdesoftware.htm.

*Revista Pedagogía Universitaria.* **Silva, Dr. C. Pedro Horruitiner. 2007.** 4, Habana : s.n., 2007, Vol. XII.

**Reyero, Eusebio. 2008.** Metodologías de diseño | Will Web For Food. [En línea] 2008. [Citado el: 3 de marzo de 2012.] http://wwff.thespacer.net/blog/metodologias-de-diseno/.

**Reyes, Yadira. 2014.** *Estrategia para el trabajo educativo de 3ro año facultad 3.* Habana : Universidad de las Ciencias Informáticas, 2014.

**Rodríguez, Maisel Luis Estrada. 2012.** *Sistema informático para la gestión de la caracterización integral del graduado.* La Habana, Cuba : UCI, 2012.

**Sánchez, Rodríguez Tamara. 2014.** *Metodología de desarrollo para la Actividad productiva de la UCI.* La habana : UCI, 2014.

**Sánchez, Tamara Rodríguez. 2014.** *Metodología de desarrollo para la Actividad Productiva de la UCI.* La Habana : s.n., 2014.

**SANDHU, R. S. 1993.** Lattice-Based Access Control Models IEEE. 1993. ISBN 0018-9162/93/1100-0009.

Sitio web de la Ventanilla Única de Comercio Exterior de Perú. [En línea] [Citado el: 5 de febrero de 2012.] https://www.vuce.gob.pe/.

**2000.** Sparx Systems. [En línea] SOLUS S.A., 2000. http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

Symfony para todos. *Symfony para todos.* [En línea] [Citado el: 15 de Febrero de 2014.] http://symfony.cubava.cu/introduccion/symfony-2/.

**The PHP Documentation Group. 2011.** *Manual de PHP.* 2011.

**TONG, E. YUAN y J. 2005.** Attributed Based Access Control (ABAC) for Web Services. *En IEEE International Conference on Web Services (ICWS'05).* 2005.

**Torres, Dina. 2014.** *Estrategia para el trabajo educativo de 2do año facultad 3.* Habana : Universidad de las Ciencias Informáticas, 2014.

*UML Diagramas de Paquetes (UML ilustrado).* **Gutierrez, Demián. 2009 Septiembre.** Venezuela : Universidad de los Andes, 2009 Septiembre.

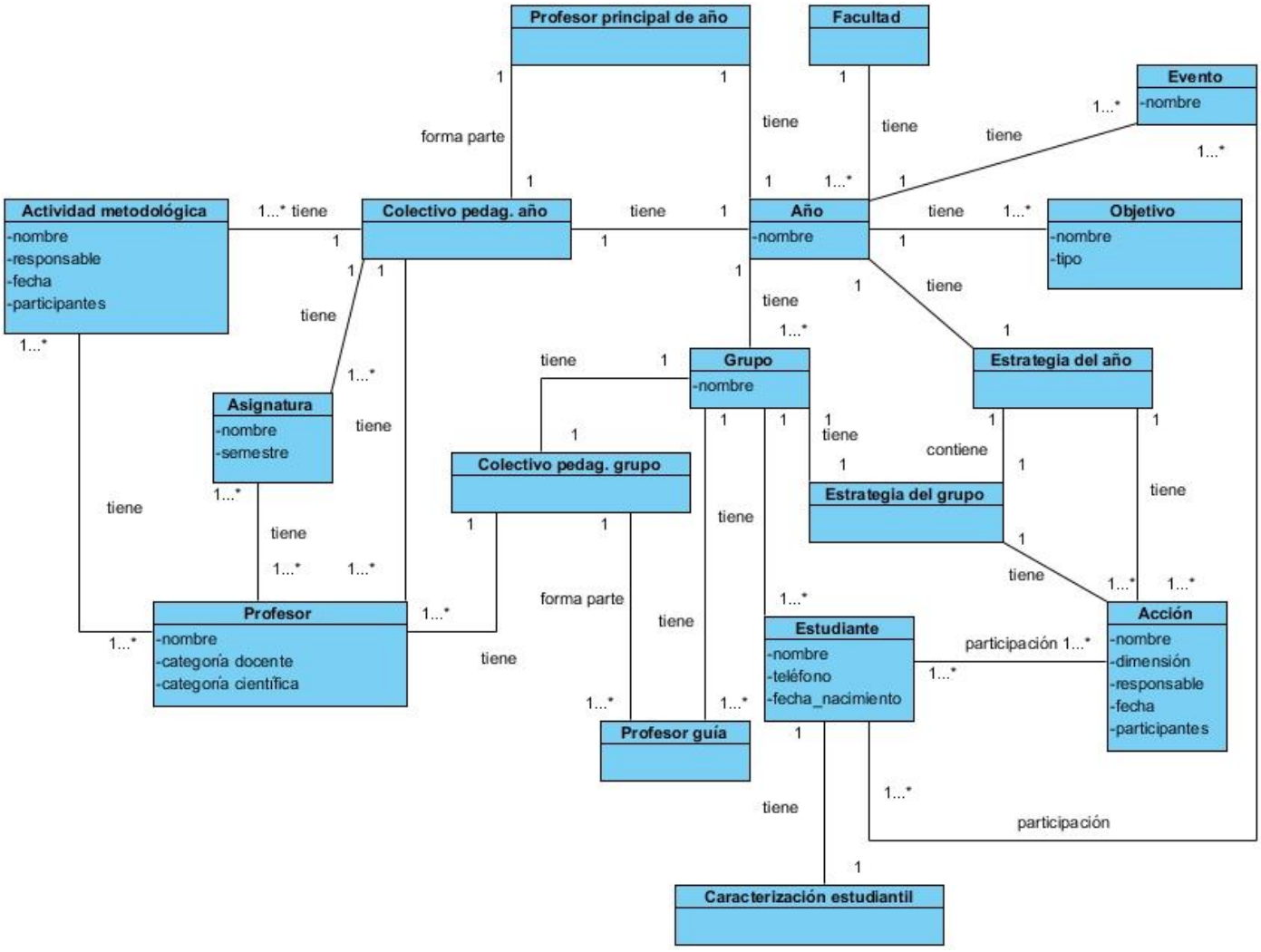
**Visual Paradigm International.** UML, BPMN and Database Tool for Software Development. [En línea] [Citado el: 3 de marzo de 2012.] http://www.visual-paradigm.com.

Yahoo. *Yahoo.* [En línea] [Citado el: 28 de Abril de 2014.] https://mx.answers.yahoo.com/question/index?qid=20090122200540AAOI1N8.

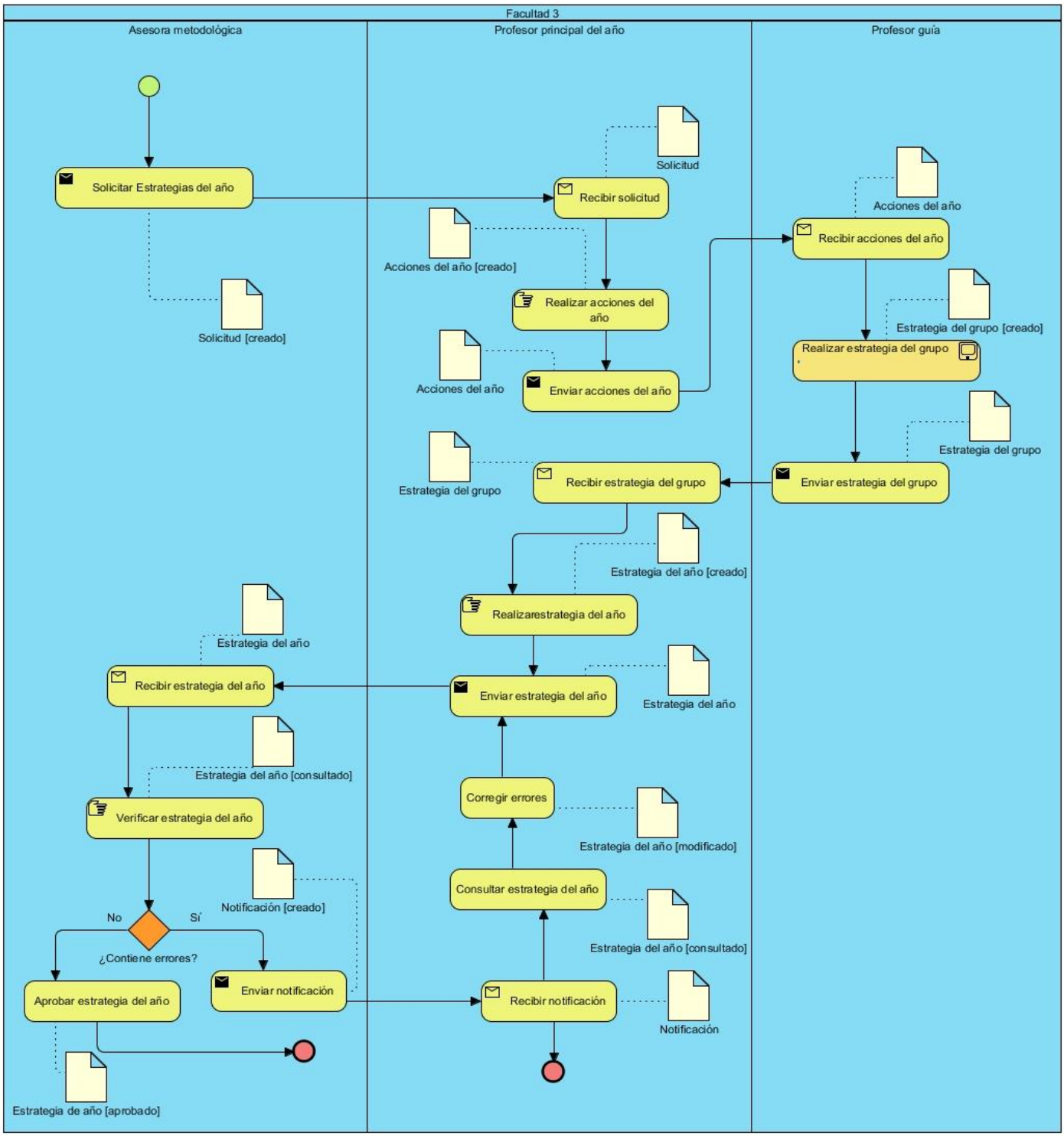
**ZANINOTTO, F. y POTENCIER, F. 2009.** librosweb.es. [En línea] 2009. [Citado el: 13 de 04 de 2015.] http://www.librosweb.es/symfony_1_1..

# Anexos

### Anexo 1 Diagrama de concepto con los atributos asociados a cada concepto.



**Anexo 2 Diagrama de procesos de negocio realizar estrategia del año.**



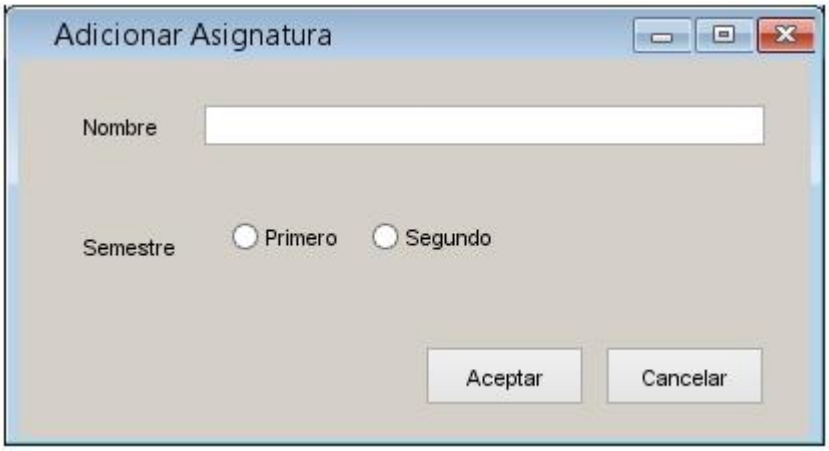
## Anexo 3 Descripciones de requisitos.

### Especificación de Requisito Adicionar asignatura

Descripción textual del requisito

| | | |
|---|---|---|
| **Precondiciones** | | Se debe encontrar en el gestionar asignaturas del año en que se encuentra |
| **Flujo de eventos** | | |
| **Flujo básico** | | |
| 1 | Se insertan los siguientes datos en los campos del formulario:<br><br>-Nombre<br><br>-Semestre | |
| 2 | El sistema valida (ver validación 1) los datos seleccionados. | |
| 3 | Si los datos son correctos el sistema inserta la asignatura con los datos ingresados. | |
| 4 | El sistema confirma la adición realizada. | |
| 5 | Concluye el requisito. | |
| **Pos-condiciones** | | |
| 1 | N/A | |
| **Flujos alternativos** | | |
| **Flujo alternativo *.a El usuario cancela la acción** | | |
| 1. | Concluye el requisito. | |
| **Pos-condiciones** | | |
| 1 | N/A | |
| **Validaciones** | | |
| 1 | Se validan los datos según lo establecido en el artefacto Modelo conceptual SGEE-MC-001. | |
| **Conceptos** | **Asignatura** | Visibles en la interfaz:<br>Nombre<br><br>Optativa<br>Utilizados internamente:<br>Id_anno |

Prototipo elemental de interfaz gráfica de usuario




**Especificación de Requisito Adicionar grupo**

Descripción textual del requisito

| | |
|---|---|
| **Precondiciones** | Se debe encontrar en el gestionar grupos del año en que se encuentra |
| **Flujo de eventos** | |
| **Flujo básico** | |
| 1 | Se insertan los siguientes datos en los campos del formulario:<br><br>-Nombre<br><br>-Guía<br><br>-Profesores |
| 2 | El sistema valida (ver validación 1) los datos seleccionados. |
| 3 | Si los datos son correctos el sistema inserta el grupo con los datos ingresados. |
| 4 | El sistema confirma la adición realizada. |
| 5 | Concluye el requisito. |
| **Pos-condiciones** | |
| 1 | N/A |
| **Flujos alternativos** | |
| **Flujo alternativo *.a El usuario cancela la acción** | |
| 1 | Concluye el requisito. |
| **Pos-condiciones** | |

| | |
|---|---|
| 1 | N/A |
| **Validaciones** | |
| 1 | Se validan los datos según lo establecido en el artefacto Modelo conceptual SGEE-MC-001. |

| | | |
|---|---|---|
| **Conceptos** | **Grupo** | Visibles en la interfaz:<br>Nombre<br><br>Guía<br>Profesores<br>Utilizados internamente:<br>Id_anno |

Prototipo elemental de interfaz gráfica de usuario


**Especificación de Requisito Modificar grupo**

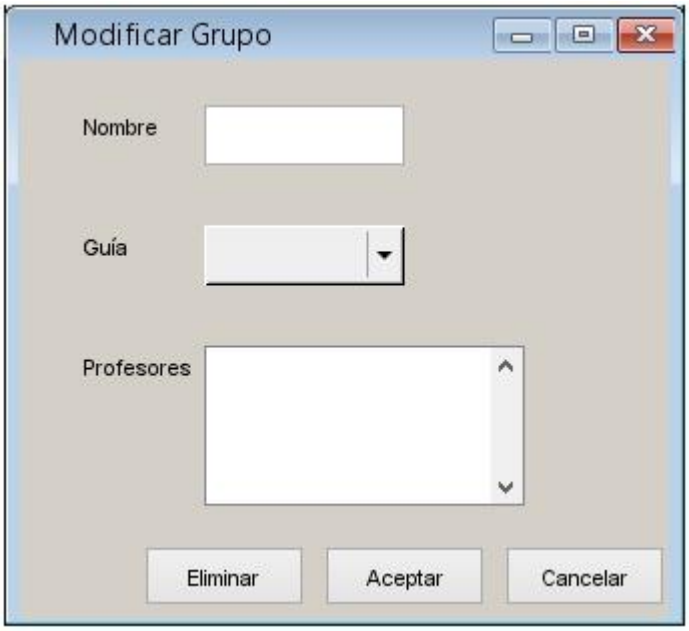
Descripción textual del requisito

| | |
|---|---|
| **Precondiciones** | El grupo ha sido seleccionado |
| **Flujo de eventos** | |
| **Flujo básico** | |
| 1 | Se modifican los siguientes datos en los campos del formulario:<br><br>- Nombre<br><br>-Guía |

| | |
|---|---|
| | -Profesores |
| 2 | El sistema valida (ver validación 1) los datos seleccionados. |
| 3 | Si los datos son correctos el sistema modifica los datos del grupo. |
| 4 | El sistema confirma la modificación realizada. |
| 5 | Concluye el requisito. |
| **Pos-condiciones** | |
| 1 | N/A |
| **Flujos alternativos** | |
| **Flujo alternativo *.a El usuario cancela la acción** | |
| 1 | Concluye el requisito. |
| **Pos-condiciones** | |
| 1 | N/A |
| **Flujo alternativo *.b El usuario elimina el grupo** | |
| 1 | Se elimina el grupo |
| 2 | Concluye el requisito. |
| **Validaciones** | |
| 1 | Se validan los datos según lo establecido en el artefacto Modelo conceptual SGEE-MC-001. |

| | | |
|---|---|---|
| **Conceptos** | **Grupo** | Visibles en la interfaz:<br>Nombre<br><br>Guía<br>Profesores<br>Utilizados internamente:<br>Id_anno |

Prototipo elemental de interfaz gráfica de usuario



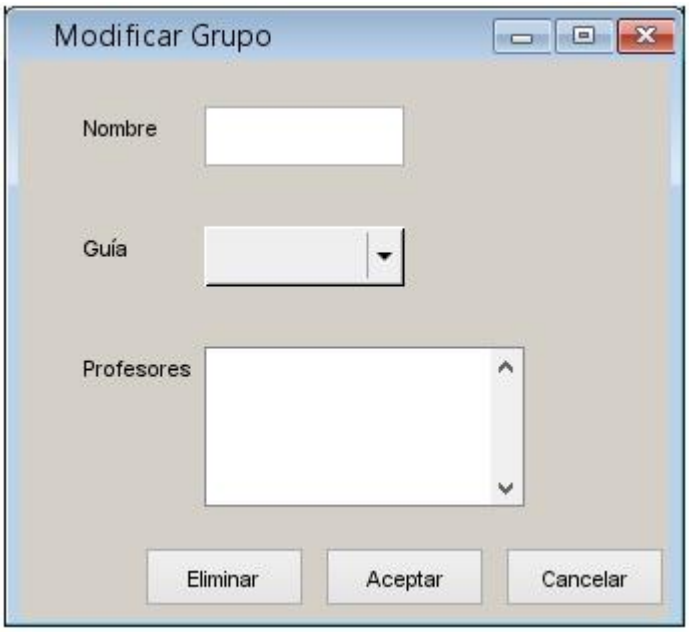

**Especificación de Requisito Eliminar grupo**

Descripción textual del requisito

| | |
|---|---|
| **Precondiciones** | Debe estar en la vista modificar grupo o mostrar grupo |
| **Flujo de eventos** | |
| **Flujo básico** | |
| 1 | Se presiona el botón eliminar |
| 2 | El sistema elimina el grupo |
| 3 | Concluye el requisito. |
| **Pos-condiciones** | |
| 1 | N/A |
| **Flujos alternativos** | |
| **Flujo alternativo** | |
| 1 | N/A |
| **Pos-condiciones** | |
| 2 | N/A |

| | | |
|---|---|---|
| **Validaciones** | | |
| 1 | N/A | |
| **Conceptos** | **Grupo** | |

Prototipo elemental de interfaz gráfica de usuario




**Especificación de Requisito Mostrar grupo**

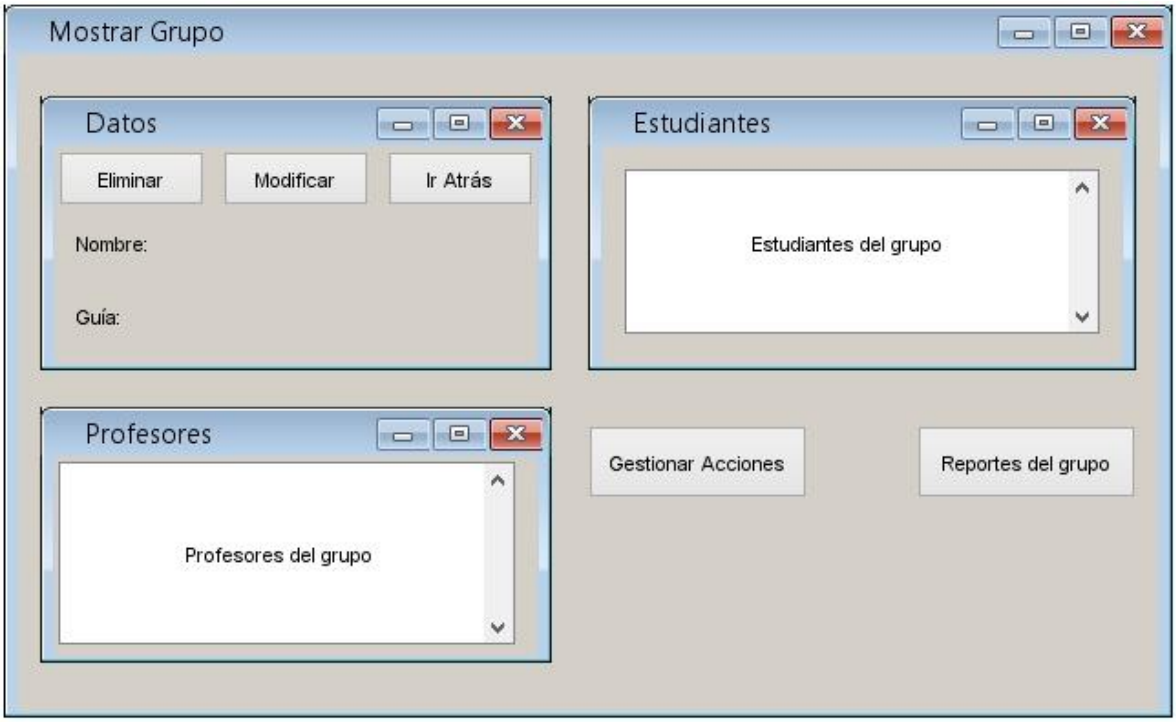
Descripción textual del requisito

| | |
|---|---|
| **Precondiciones** | El grupo ha sido seleccionado |
| **Flujo de eventos** | |
| **Flujo básico** | |
| 1 | El sistema muestra el grupo |
| 2 | Concluye el requisito. |
| **Pos-condiciones** | |
| 1 | N/A |
| **Flujos alternativos** | |
| **Flujo alternativo** | |
| 1 | N/A |
| **Pos-condiciones** | |

| | |
|---|---|
| 1 | N/A |
| **Validaciones** | |
| 1 | N/A |

| | | |
|---|---|---|
| **Conceptos** | **Grupo** | |

Prototipo elemental de interfaz gráfica de usuario




**Especificación de Requisito Adicionar profesor**

Descripción textual del requisito

| | |
|---|---|
| **Precondiciones** | Se debe encontrar en el autenticar usuario |
| **Flujo de eventos** | |
| **Flujo básico** | |
| 1 | Se insertan los siguientes datos en los campos del formulario:<br><br>-Usuario<br><br>-Contraseña |
| 2 | El sistema valida (ver validación 1) los datos seleccionados. |

| | |
|---|---|
| 3 | Si los datos son correctos el sistema inserta un usuario y al profesor con ese usuario. |
| 4 | El sistema confirma la adición realizada. |
| 5 | Concluye el requisito. |
| **Pos-condiciones** | |
| 1 | N/A |
| **Flujos alternativos** | |
| **Flujo alternativo** | |
| 1 | N/A |
| **Pos-condiciones** | |
| 1 | N/A |
| **Validaciones** | |
| 1 | Se validan los datos según lo establecido en el artefacto Modelo conceptual SGEE-MC-001. |

| | | |
|---|---|---|
| **Conceptos** | **Usuario** | Visibles en la interfaz:<br>Usuario<br><br>Contraseña |

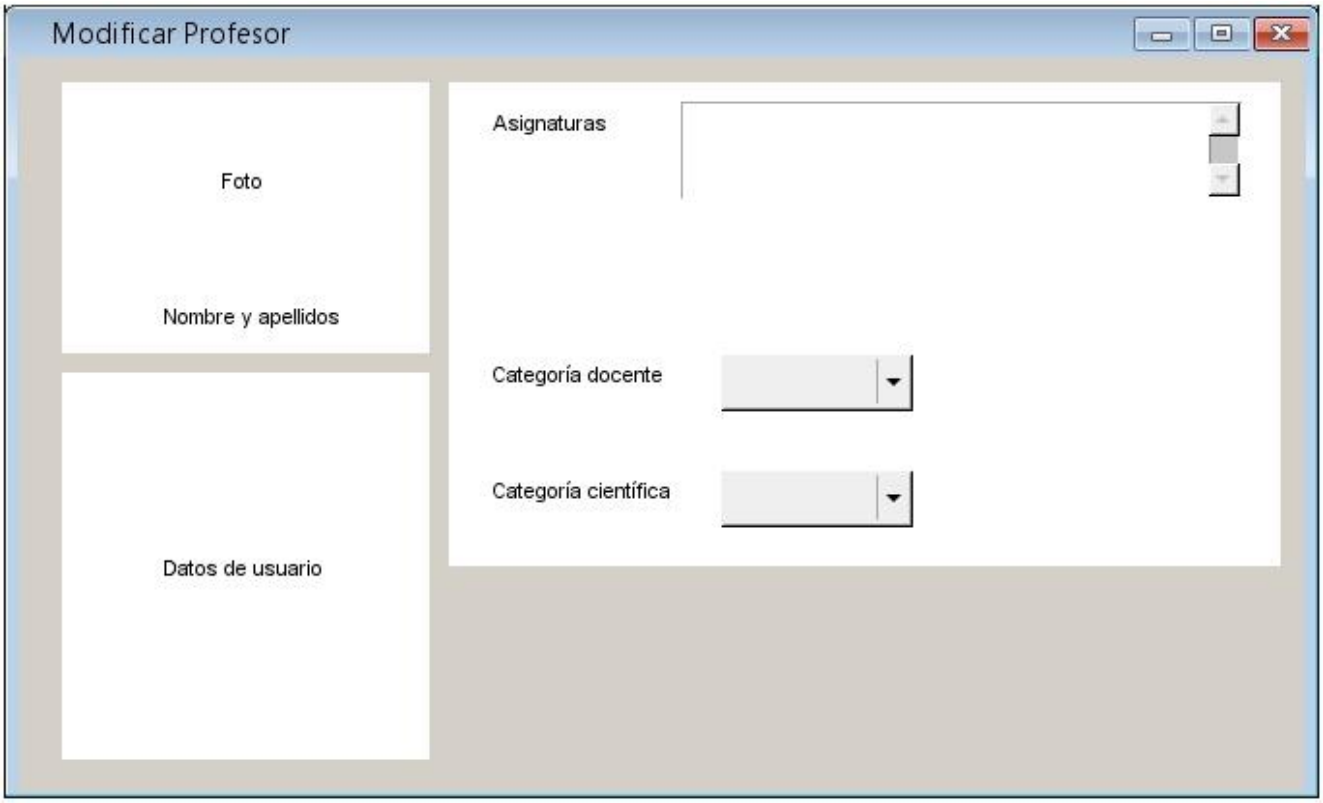
Prototipo elemental de interfaz gráfica de usuario

**Especificación de Requisito Modificar profesor**

Descripción textual del requisito

| | | |
|---|---|---|
| **Precondiciones** | El profesor ha sido seleccionado | |
| **Flujo de eventos** | | |
| **Flujo básico** | | |
| 1 | Se modifican los siguientes datos en los campos del formulario:<br><br>-Asignaturas<br><br>-Categoría docente<br><br>-Categoría científica | |
| 2 | El sistema valida (ver validación 1) los datos seleccionados. | |
| 3 | Si los datos son correctos el sistema modifica los datos del profesor. | |
| 4 | El sistema confirma la modificación realizada. | |
| 5 | Concluye el requisito. | |
| **Pos-condiciones** | | |
| 1 | N/A | |
| **Flujos alternativos** | | |
| **Flujo alternativo *.a El usuario cancela la acción** | | |
| 1 | Concluye el requisito. | |
| **Pos-condiciones** | | |
| 1 | N/A | |
| **Flujo alternativo** | | |
| 1 | N/A | |
| **Validaciones** | | |
| 1 | Se validan los datos según lo establecido en el artefacto Modelo conceptual SGEE-MC-001. | |
| **Conceptos** | **Profesor** | Visibles en la interfaz:<br>Asignaturas<br><br>Categoría docente<br><br>Categoría científica |

Prototipo elemental de interfaz gráfica de usuario




**Especificación de Requisito Eliminar profesor**

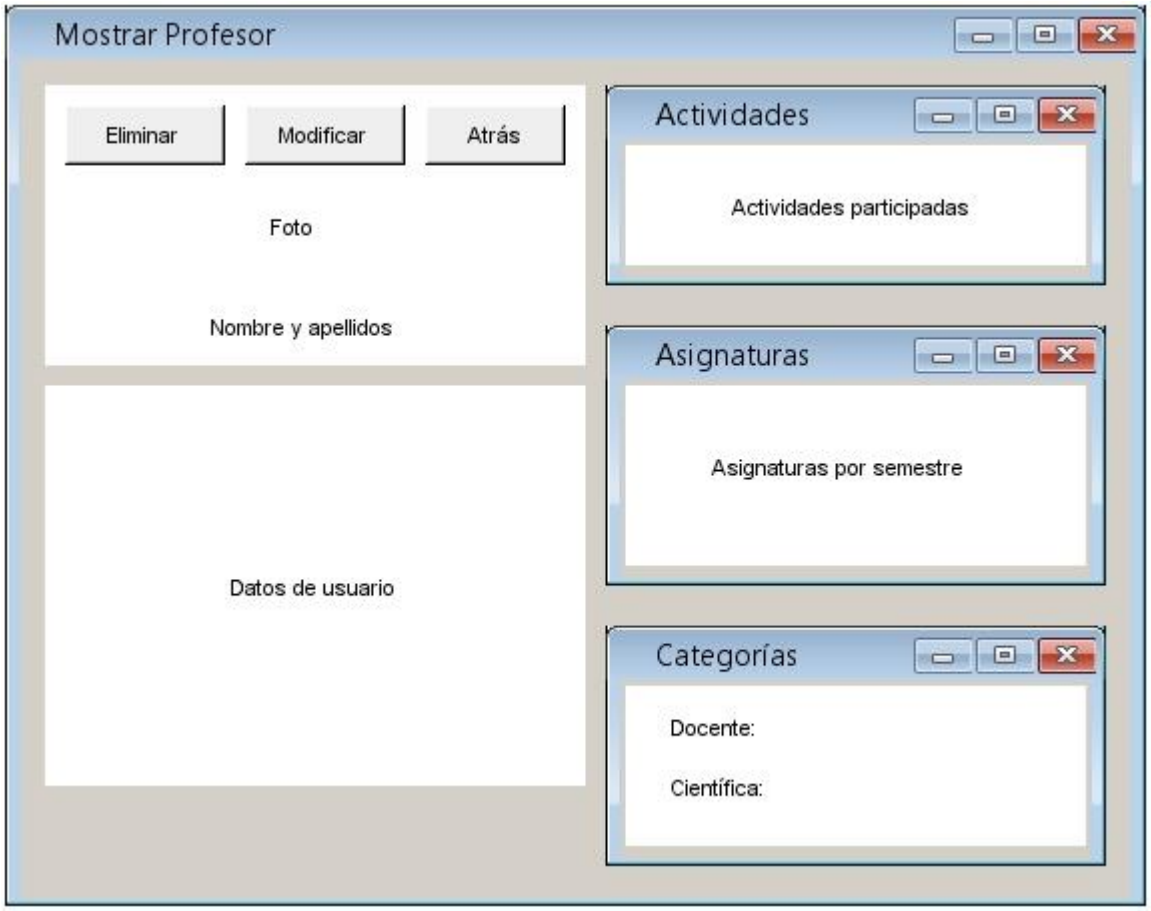
Descripción textual del requisito

| | |
|---|---|
| **Precondiciones** | Debe estar en la vista modificar profesor o mostrar profesor |
| **Flujo de eventos** | |
| **Flujo básico** | |
| 1 | Se presiona el botón eliminar |
| 2 | El sistema elimina el profesor |
| 3 | Concluye el requisito. |
| **Pos-condiciones** | |
| 1 | N/A |
| **Flujos alternativos** | |
| **Flujo alternativo** | |
| 1 | N/A |
| **Pos-condiciones** | |

| | |
|---|---|
| 1 | N/A |
| **Validaciones** | |
| 1 | N/A |

| | | |
|---|---|---|
| **Conceptos** | **Profesor** | |

Prototipo elemental de interfaz gráfica de usuario




**Especificación de Requisito Mostrar profesor**

Descripción textual del requisito

| | |
|---|---|
| **Precondiciones** | **El profesor ha sido seleccionado** |
| **Flujo de eventos** | |
| **Flujo básico** | |
| **1** | El sistema muestra el profesor |

| | |
|---|---|
| **2** | Concluye el requisito. |
| **Pos-condiciones** | |
| **1** | N/A |
| **Flujos alternativos** | |
| **Flujo alternativo** | |
| **1** | N/A |
| **Pos-condiciones** | |
| **1** | N/A |
| **Validaciones** | |
| **1** | N/A |

| | | |
|---|---|---|
| **Conceptos** | **Profesor** | |

Prototipo elemental de interfaz gráfica de usuario

**Especificación de Requisito Adicionar estudiante**

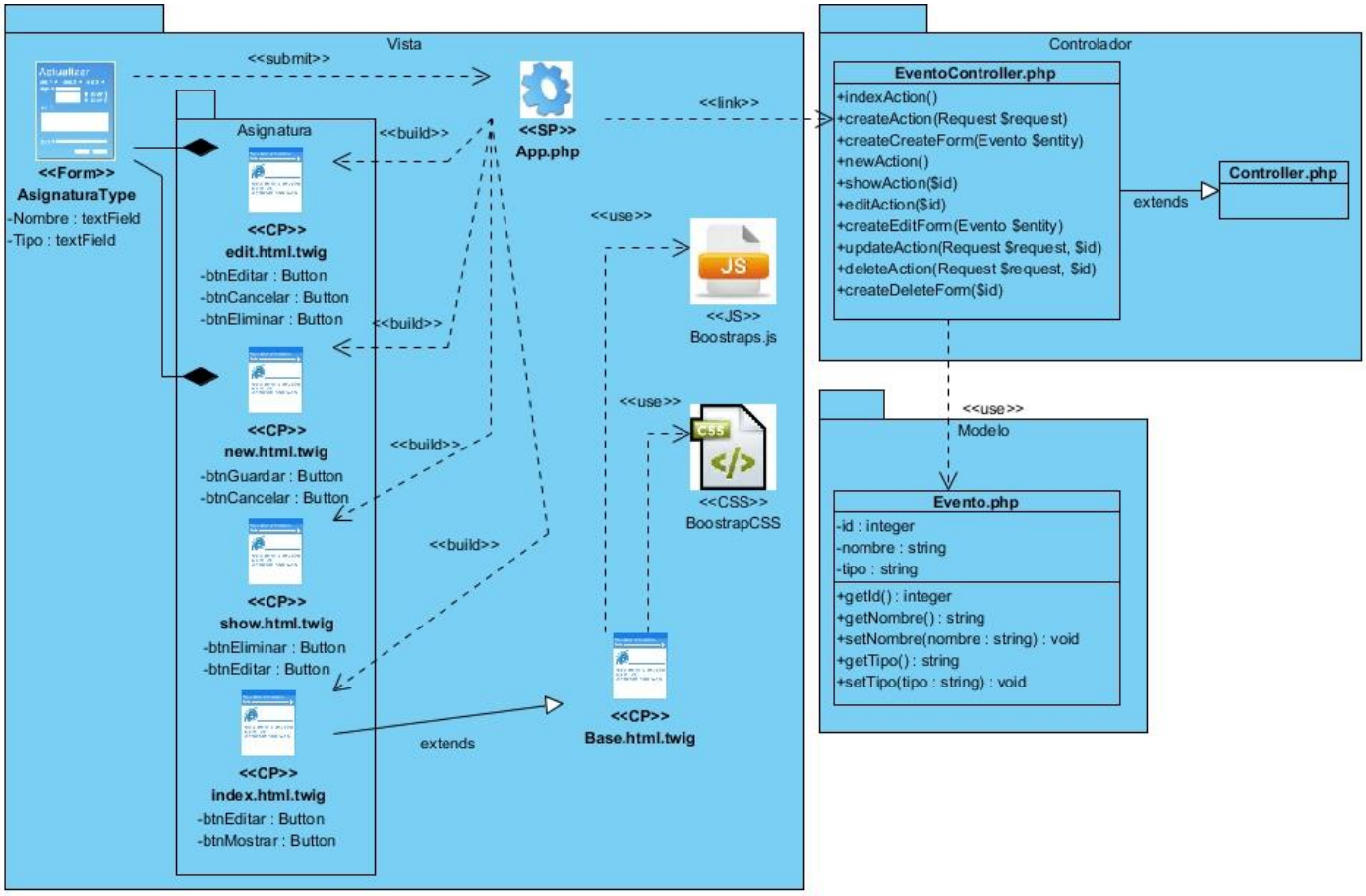
Descripción textual del requisito

| | | |
|---|---|---|
| **Precondiciones** | | Se debe encontrar en el autenticar usuario |
| **Flujo de eventos** | | |
| **Flujo básico** | | |
| 1 | Se insertan los siguientes datos en los campos del formulario:<br><br>-Usuario<br><br>-Contraseña | |
| 2 | El sistema valida (ver validación 1) los datos seleccionados. | |
| 3 | Si los datos son correctos el sistema inserta un usuario y al estudiante con ese usuario. | |
| 4 | El sistema confirma la adición realizada. | |
| 5 | Concluye el requisito. | |
| **Pos-condiciones** | | |
| 1 | N/A | |
| **Flujos alternativos** | | |
| **Flujo alternativo *.a El usuario cancela la acción** | | |
| 1 | Concluye el requisito. | |
| **Pos-condiciones** | | |
| 1 | N/A | |
| **Validaciones** | | |
| 1 | Se validan los datos según lo establecido en el artefacto Modelo conceptual SGEE-MC-001. | |
| **Conceptos** | **Usuario** | Visibles en la interfaz:<br>Usuario<br><br>Contraseña |

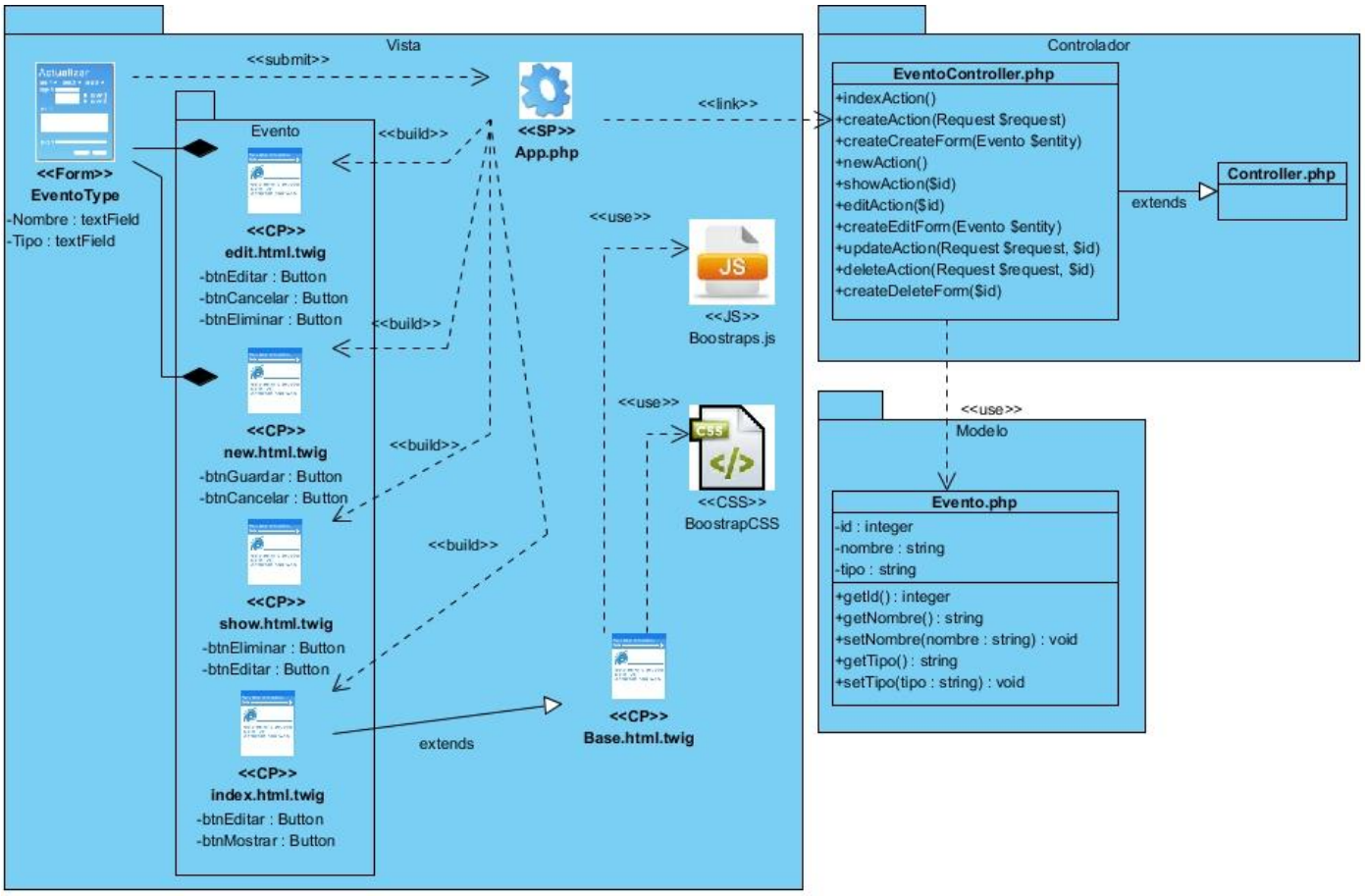
Prototipo elemental de interfaz gráfica de usuario

**Anexo 4 Diagramas de clases del diseño.**

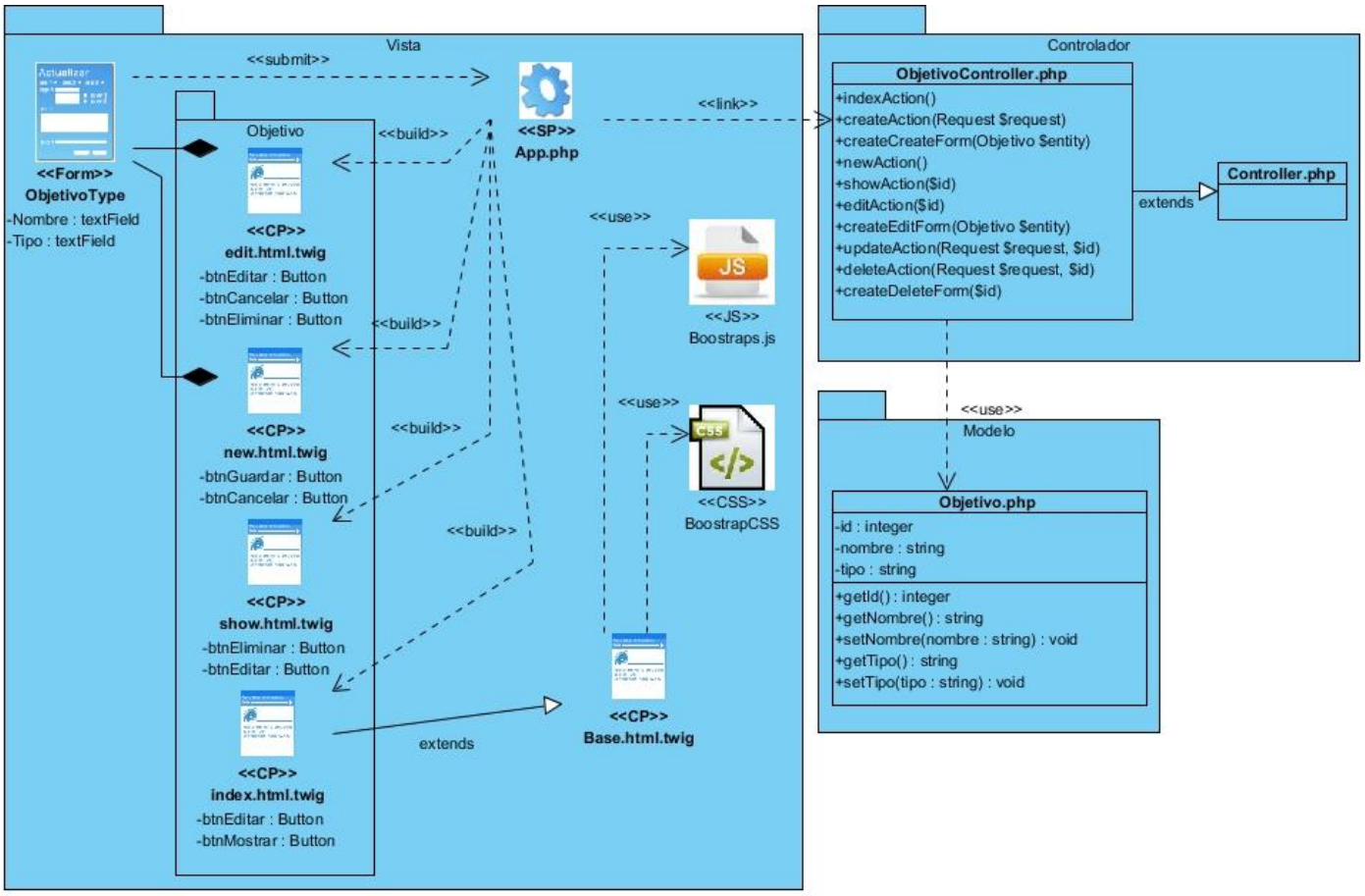
**Escenario Gestionar evento**



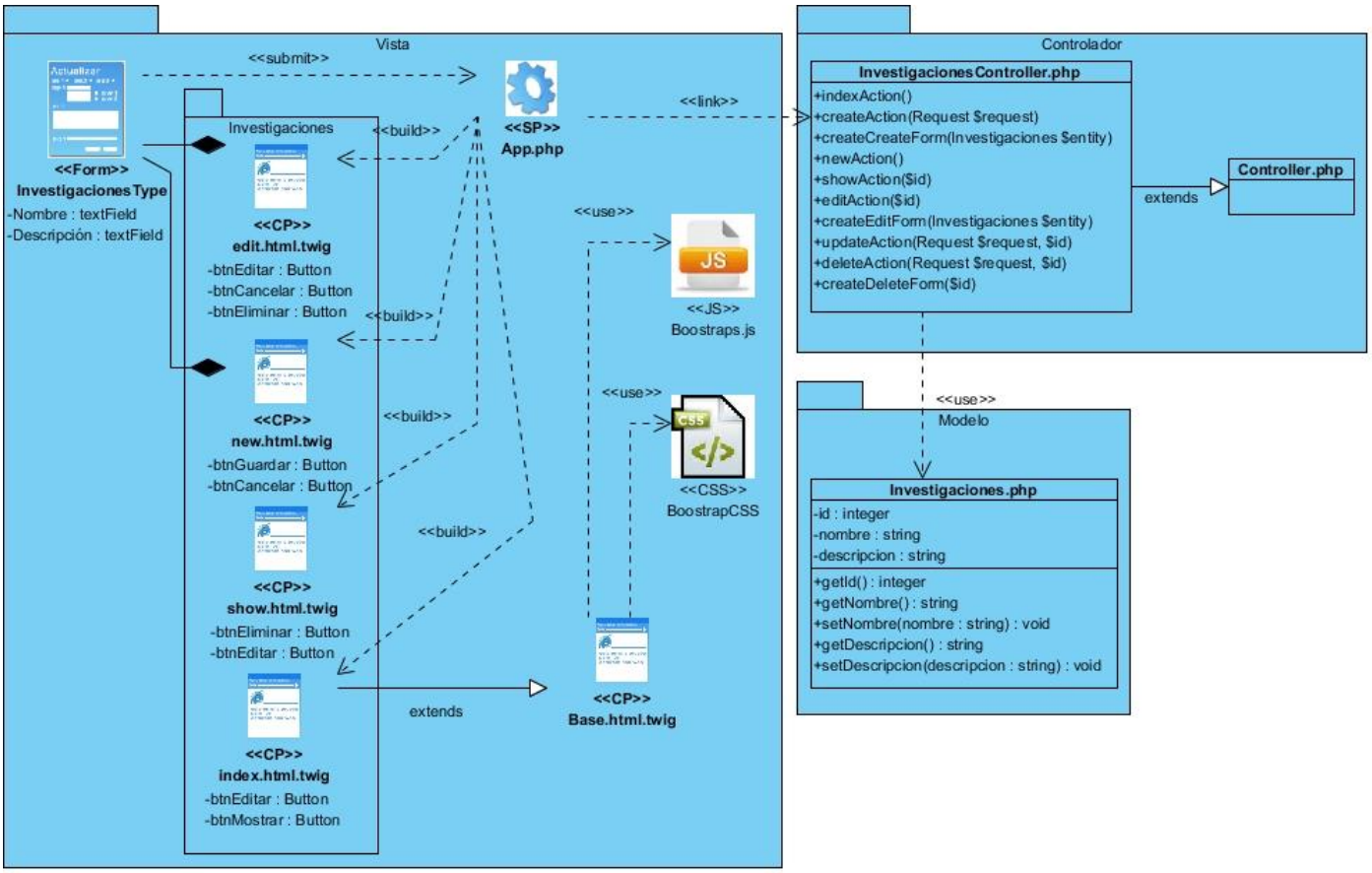
### Escenario Gestionar Evento



**Escenario Gestionar Objetivo**



### Escenario Gestionar investigaciones



**Escenario Gestionar profesor**