**Universidad de las Ciencias Informáticas**

**Facultad 4**


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

# *"Sistema para controlar los presupuestos de gastos administrativos de la Facultad 4 y el Centro FORTES".*

***Autores:*** *Alexis Borges Mora*

*Victor Abel Reyes Quesada*

***Tutor:*** *Ing. Yasirys Terry González*

"Ciudad de La Habana, junio 2016".

"Año 58 de la Revolución"

# *Declaración de autoría*

Declaramos ser autores de este trabajo de diploma y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo es su beneficio.

Para que así conste firmo la presente a los ____ días del mes _____ del año 2016.

_____
Firma del autor
Victor Abel Reyes Quesada

_____
Firma del autor
Alexis Borges Mora

_____
Firma del tutor
Ing. Yasirys Terry González

*"En la tierra hacen falta personas que trabajen más y critiquen menos, que construyan más y destruyan menos, que prometan menos y resuelvan más, que esperen recibir menos y dar más, que digan mejor ahora que mañana"*

***Ernesto Che Guevara***

# Dedicatoria

*De Victor Abel Reyes Quesada :*

*A mis padres por ser el pilar fundamental en todo lo que soy, por el apoyo incondicional en toda mi educación, tanto académica, como en la vida.*
*Todo este fruto ha sido posible gracias a sus esfuerzos.*

*De Alexis Borges Mora:*

*A mi mamá y mi papá. Mami estés donde estés, este resultado es por tí y para tí, significas todo en mi vida.*

## *Agradecimientos*

*De Victor Abel Reyes Quesada:*

*La gratitud en silencio no sirve de nada y por ello agradezco desde lo más profundo de mi corazón a mis padres, Magdalena y Alberto, quiero que sepan que con su amor y dedicación me ayudaron a ser una mejor persona, no me alcanzaría la vida para agradecerle todo el esfuerzo y sacrificio que han hecho por mí. Gracias por haber confiado siempre en mí.*

*A mis hermanos Anita y José Alberto gracias por quererme, ayudarme siempre y además por ser los mejores ejemplos de mis mayores aspiraciones.*

*Gracias a mis primas Zonia y Claudia, a mis primos Junior y Alejandro, a mi tía Iris, y a mi familia en general por estar ahí para mí.*

*Una mención especial para Ana y Naranjo por su apoyo incondicional desde que me conocieron. Gracias por tanto y todo.*

*Agradezco a los profesores Feliberto, Maritza y Graciela, porque llevaron la profesión un poquito más allá del aula.*

*Los amigos también ocupan un lugar importante y por eso no puedo obviar a los que están en la universidad y los que ya no están, como Malidia, Yadian, Daniela, Gisselle, Arlety, Eduardo, Nael, Ronald, Manuel,Yisel y muchos más.*

*Quiero agradecerle a Ailen por ser una amiga que siempre ha estado presente en estos años de la universidad, dándome aliento para que terminara mi anhelado sueño, la considero además de una amiga una hermana, gracias por haberme dedicado parte de su tiempo en ayudarme y comprenderme.*

*A Andrés por su afecto, su lealtad incondicional, por tantos buenos momentos compartidos y estar cuando siempre te necesito.*

*A Jorge Luis por apoyarme y ser tan leal conmigo, gracias por escucharme y darme tus sabios consejos.*

*A Addiel por ser mi gran amigo, eres como un hermano y cuenta conmigo para lo que necesites.*

*A Yunierkis por ser un amigo de verdad, desde hace tiempo estamos juntos, eres para mí como un hermano.*

*A Neobel por ser un excelente amigo, por soportar mis defectos, tolerar mis humores y sobre todas las cosas por entenderme.*

*A Alejandro, que en poco tiempo que lo conocí se ganó mi afecto y cariño, y a pesar que no está a mi lado siempre se preocupa por mí.*

*A Leonar por compartir conmigo mis penas, alegrías, triunfos, derrotas, te quiero mucho amigo y deseo todo lo mejor para ti.*

*A Yayi por ser una amiga tan sincera y que siempre estás disponible para darme su apoyo. Me siento feliz de compartir esta amistad que espero sea para siempre.*

*Especialmente a Yileni y a Kenier por ayudarme durante tanto tiempo, no importándole la hora ni el lugar para transmitirme y brindarme sus conocimientos.*

*A mi compañero de tesis Alexis por ser un gran amigo, por ayudarme todos estos meses y por verse preocupado por mí cuando lo necesité.*

*A mi tutora Yasirys por la infinita paciencia que ha tenido con nosotros, por su exigencia, conocimientos transmitidos y dedicación incondicional, por llevarme de la mano en este camino hacia mi graduación, y la dedicación a enseñarme la luz al final del túnel.*

*A todos y cada una de las personas que influyeron en mi formación como profesional, muchas gracias.*

***De Alexis Borges Mora:***

*Agradecerles a todas las personas que han influido positivamente en mi formación como ingeniero me tomaría un día entero, quizás dos, sin embargo, existen personas sin las cuales este logro habría sido imposible.*

*A mi papá, gracias por apoyarme cuando quise continuar mis estudios, por preocuparte por mí, gracias por ser mi amigo.*

*A los profesores que ayudaron a formarme, los buenos y los no tan buenos.*

*A mis amigos, los que hoy se encuentran aquí y los que no, ellos saben quiénes son, gracias por compartir tantas historias y por hacer de estos últimos años de carrera algo más divertido.*

*A Jessica, a Padilla, a Alberto y a Danay (sin importar el orden) que dejaron hace mucho tiempo de ser mis amigos, para convertirse en otros miembros muy importantes de mi familia. Con Uds viví los mejores años de estudiante, gracias por estar siempre.*

*A mi novia, gracias por compartir momentos de la vida tan lindos conmigo, por soportarme y por lavarme la ropa cuando estaba estudiando, gracias por todo, eres muy importante para mí.*

*A mi tío Manolo gracias por ayudarme y preocuparte por mí.*

*A Oniel, sin ti esta tesis no hubiese comenzado con el pie derecho, gracias por ayudarme a dar mis primeros pasos en la programación.*

*A Yasirys, que sin quererlo se convirtió en otro miembro de mi círculo de buenos amigos, gracias por tus consejos y por ser tan paciente con nosotros.*

*A mi compañero de tesis y amigo Victor, gracias por tus locuras y por obligarme a superarme, fue interesante hacer esto contigo.*

*A mi mamá, gracias por formarme y hacer de mí el hombre que soy hoy, en estos momentos deberías estar aquí conmigo compartiendo este logro, porque el sacrificio fue mío, pero la idea de hacerme ingeniero fue toda tuya, gracias por estar ahí cuando me hizo falta, gracias por ser mi amiga, mi novia, mi consejera, mi guía, gracias por ser la mejor madre del mundo y de siempre. Donde quiera que te encuentres ahora mismo: Gracias.*

## Resumen

El control presupuestario es una herramienta imprescindible para la gestión de la empresa y anticiparse a los problemas que se pueden producir. Actualmente el proceso de control de los Presupuestos de Gastos Administrativos de la Facultad 4 y el Centro de Tecnologías para la Formación se realiza de forma manual, lo cual genera un gran cúmulo de información, la que se encuentra en diferentes soportes provocando el difícil acceso a la misma, además está expuesta al acceso y modificación por personas no autorizadas; esta situación provoca la disminución de la seguridad de la información que se genera. Para dar solución a los problemas identificados se plantea como objetivo de la investigación desarrollar un sistema informático que permita aumentar la seguridad de la información que se genera en el proceso de control de los Presupuestos de Gastos Administrativos de la Facultad 4 y Centro FORTES de la Universidad de las Ciencias Informáticas. El desarrollo del sistema para controlar los Presupuestos de Gastos Administrativos fue guiado por la metodología ágil Programación Extrema. Para el desarrollo del mismo se utilizaron Groovy y Java como lenguajes de programación, Grails como marco de desarrollo web, NetBeans (8.1) como entorno de desarrollo, Apache Tomcat (7.0) como servidor de aplicaciones, Visual Paradigm (8.0) como herramienta CASE y PostgreSQL (9.4) como servidor de base de datos.

**Palabras clave:** seguridad de la información, gastos administrativos, presupuesto.

# Índice

# Índice de Tablas

# Índice de Figuras

# INTRODUCCIÓN

La planificación está definida como un plan general y metódicamente organizado para obtener un objetivo determinado, tal como el desarrollo armónico de una ciudad, el desarrollo económico, la investigación científica y el funcionamiento de una industria (1). La planificación no solo es útil para una nueva empresa, sino también para empresas existentes que quieren cambiar de estrategia para penetrar nuevos mercados o simplemente hacerse más eficientes y rentables en el mercado que están.

En la planificación se deben tener en cuenta: el mercado, la competencia, la logística, la política de precios, la organización, entre otros elementos (2).Todos los trabajadores de la empresa, a su vez, deben ser conocedores de los objetivos que se quieren alcanzar con la planificación hecha para que el plan se siga o sea cambiado si es necesario, para un mejor rendimiento y en un final lleve a la empresa al cumplimiento de las metas trazadas.

Parte importante en la planificación es el presupuesto con el que se cuente, los recursos financieros que tenga asociados y los controles que se realicen a este; con el objetivo de garantizar que estos recursos están siendo bien utilizados y además están ayudando en la implementación del plan, ya que mediante el control presupuestario se puede realizar un análisis detallado de los gastos reales en los que se ha incurrido (3) (4) (5).

El control presupuestario es una herramienta imprescindible para controlar la gestión de la empresa y anticiparse a los problemas que se pueden producir (6), permite medir y examinar los resultados obtenidos en el período, para evaluarlos y decidir las medidas correctivas que sean necesarias. Toda empresa con un objetivo bien definido, un plan acorde para lograrlo y un sistema de control de presupuesto aumenta la posibilidad de alcanzar el éxito (3).

El presupuesto es una herramienta esencial para planificar cómo se determinan y gestionan los recursos necesarios para lograr los mejores resultados posibles (7) (6). La presupuestación requiere el uso de pronósticos, además de conocer el origen de los costos de las actividades y la capacidad para integrar las actividades de la organización (7).

Los presupuestos se clasifican de distinta manera, según el grado de flexibilidad de las asignaciones, como rígidos o flexibles; según el período de tiempo que abarcan, como corto plazo o largo plazo;

según el sector en que serán aplicados, como público y privado y según el campo de aplicación dentro de la empresa, como operativos o financieros (7).

El Presupuesto de Gastos Administrativos (PGA) es un tipo de presupuesto flexible que engloba las secciones de administración general, dirección general y gastos generales comunes. Se consideran todos los gastos por naturaleza que con carácter principal o accesorio realicen los departamentos descritos. Son imputables gastos tales como salarios, material de oficina, transportación, alimentación y alojamiento del personal, gastos de contratación (7).

Para la planificación del PGA son necesarios datos y experiencia documentada, donde se puedan obtener bases para plantear la necesidad de un giro en la política económica y de mercado de una empresa. Estos datos se deben recopilar de la manera más eficiente posible para así no hacer del medio un lastre para el fin. A su vez, se necesita pensar en la eficiencia y optimización de los métodos a utilizar para lograr el objetivo propuesto. Si se puede hacer más con menos al final la ganancia es mayor y la empresa crece y se fortalece; de aquí la importancia de la toma de decisiones acertadas.

La Universidad de las Ciencias Informáticas (UCI), institución presupuestada perteneciente al Ministerio de Educación Superior (MES), fue creada no solo con el objetivo de ser una entidad educativa, sino también productiva, que enfoca su aspiración a la informatización del país; es considerada una de las principales empresas productoras de software en Cuba.

La UCI está conformada por facultades y centros productivos vinculados a las actividades docentes, productivas e investigativas. En la Facultad 4 de dicha universidad se desarrollan como procesos sustantivos: la formación de los estudiantes, superación de los profesionales, capacitación del personal no docente, extensión universitaria y la producción de software; así como otros procesos administrativos que apoyan los antes mencionados. El desarrollo de estos procesos incide directamente en la ejecución de los presupuestos de gastos de la Facultad 4 y el Centro de Tecnologías para la Formación (FORTES).

Anualmente, desde la Dirección de Planificación y Estadísticas de la Universidad, se le asigna a la Facultad 4 un presupuesto en Pesos Cubanos Convertibles (CUC) y un presupuesto en Pesos Cubanos (CUP); al igual que al Centro FORTES. Mensualmente se debe realizar un control del comportamiento de la ejecución de estos presupuestos, haciendo análisis con respecto al plan. En este control se hace necesario manejar, desde el Vicedecanato Administrativo (VDA), datos como la matrícula de estudiantes por año, la cantidad de alumnos ayudantes y los gastos asociados a

transportación, alimentación y alojamiento de trabajadores por concepto de viáticos. El manejo de estos datos y las comparaciones de ejecuciones con respecto al plan se hacen en formato duro y con la utilización de herramientas ofimáticas, lo que provoca alta exposición a ocurrencia de errores humanos y dificultad y demora en el proceso, que requiere el control de datos sensibles y numerosos cálculos.

También se deben realizar análisis comparativos con ejecuciones de períodos anteriores, por lo que se hace necesario guardar la información asociada a la ejecución del presupuesto de al menos cinco años anteriores. Esta información digital se encuentra dispersa en diferentes archivos, atentando contra la adecuada toma de decisiones con respecto a la ejecución del presupuesto.

La información asociada al presupuesto (digital y en formato duro) se archiva en el local del VDA y los intercambios de información entre los involucrados en el proceso de control de presupuestos de gastos (Decano, Director del Centro FORTES y Vicedecano Administrativo), se realizan fundamentalmente vía correo electrónico y en despachos personales. Esta situación dificulta el acceso a la información por el personal autorizado, y a su vez, no brinda a esa información la seguridad requerida, propiciando posibles pérdidas o modificaciones indebidas.

Por lo anteriormente descrito, se plantea el siguiente **problema a resolver:** ¿cómo aumentar la seguridad de la información que se genera en el proceso de control de los PGA en la Facultad 4 y el Centro FORTES de la UCI? El **objeto de estudio** está enmarcado en el proceso de control de los PGA. Se determina para la investigación como **campo de acción**, la informatización del proceso de control de los PGA en la Facultad 4 y el Centro FORTES. Para dar solución al problema de investigación planteado se define como **objetivo general:** desarrollar un sistema informático que permita aumentar la seguridad de la información que se genera en el proceso de control de los PGA de la Facultad 4 y Centro FORTES de la UCI. El objetivo general se desglosa en los siguientes **objetivos específicos:**

1. Fundamentar los conceptos y características relacionados con el control de los PGA de la Facultad 4 y del Centro FORTES, mediante el estudio bibliográfico.
2. Diseñar el sistema de control de los PGA de la Facultad 4 y el Centro FORTES.
3. Implementar el sistema de control de los PGA de la Facultad 4 y el Centro FORTES.
4. Validar el sistema implementado a través de pruebas de software.

Para dar cumplimiento a los objetivos específicos se sustenta la siguiente **hipótesis:** el desarrollo de un sistema informático permitirá el control de los PGA de la Facultad 4 y Centro FORTES de la UCI aumentando la seguridad de la información que se genera en este proceso.

Lugo del análisis sobre el concepto de seguridad de la información tomado de la ISO/IEC 27001 (8) se asume en la presente investigación como seguridad de la información: al "conjunto de medidas preventivas y reactivas que permiten resguardar y proteger la información buscando mantener la confidencialidad, la disponibilidad e integridad de la misma".

Para lograr el cumplimiento de los objetivos se proponen las siguientes **tareas de investigación:**

- Caracterización del proceso de control de los PGA.
- Realización de entrevistas a los trabajadores de la Facultad 4 que están involucrados en el proceso de control de los PGA.
- Análisis de las características y funcionalidades de soluciones similares a la que se desea desarrollar.
- Selección de las herramientas y tecnologías a utilizar en el desarrollo del sistema para controlar los PGA de la Facultad 4 y el Centro FORTES.
- Diseño de las interfaces de usuario del sistema para controlarlos PGA de la Facultad 4 y el Centro FORTES.
- Diseño de la base de datos para el sistema a desarrollar.
- Modelado de los artefactos definidos en la metodología seleccionada.
- Implementación del sistema para controlar los PGA de la Facultad 4 y el Centro FORTES.

Los **métodos científicos** que se utilizaron en el transcurso del proceso de la investigación a nivel **teórico** fueron los métodos: **histórico–lógico** y **analítico–sintético**. Con el primero se profundizó en los antecedentes y tendencias actuales relacionadas con el objeto de estudio y el segundo permitió analizar los procesos y documentos del área del Vicedecanato Administrativo, la síntesis de conceptos, teorías, técnicas y herramientas; además del procesamiento de información y arribo a conclusiones.

A nivel **empírico** se utilizó el método **entrevista**, en el cual se seleccionó la entrevista abierta para conocer los aspectos esenciales referentes al proceso de control de los PGA del VDA de la Facultad 4 y el Centro FORTES.

El documento está estructurado en 3 capítulos, como se muestra a continuación:

**Capítulo 1. Fundamentación teórica:** se realiza un análisis de los conceptos fundamentales relacionados con el objeto de estudio. Se lleva a cabo un estudio de soluciones similares existentes. Además, se realiza el análisis de las distintas herramientas, tecnologías y metodologías a utilizar para el desarrollo de la aplicación**.**

**Capítulo 2. Exploración y Planificación:** se realiza una descripción de la propuesta de solución y sus características principales. Se exponen las funcionalidades que va a proporcionar el sistema y los artefactos generados correspondientes a la metodología utilizada.

**Capítulo 3. Implementación y prueba:** se describen las tareas a realizar para llevar a cabo la implementación de las principales funcionalidades definidas. Se muestra el análisis realizado sobre los tipos de pruebas que se pueden aplicar para comprobar el correcto el funcionamiento de la propuesta de solución, detallando el tipo de prueba seleccionada para detectar las no conformidades y obtener como resultado un software que cumpla con las expectativas del cliente.

# CAPÍTULO 1

# FUNDAMENTACIÓN TEÓRICA

## 1.1 Introducción

En la actualidad los sistemas informáticos se han convertido en herramientas para aumentar la productividad personal, académica, profesional y laboral; entre ellos se encuentran los sistemas de planificación de recursos empresariales que permiten la planificación y control de los PGA. En el presente capítulo se realiza un análisis detallado de las herramientas y tecnologías más utilizadas para el desarrollo de aplicaciones web, así como un estudio de la información relacionada con el proceso de control del PGA de la Facultad 4 y el Centro FORTES.

## 1.2 Sistemas similares desarrollados en Cuba

Nuestro país ha identificado desde muy temprano la conveniencia y necesidad de introducir en la práctica social, las tecnologías de la información y las comunicaciones para satisfacer las necesidades de todas las esferas de la sociedad en su búsqueda por alcanzar una mayor eficiencia y eficacia en todos los procesos. El sector de la economía no es ajeno a esta situación, ya que en nuestro país se han desarrollado varios sistemas que buscan informatizar la actividad económica, contable y financiera de las empresas. A continuación, se describen algunos sistemas que se utilizan en el proceso de control del presupuesto.

### 1.2.1 eTES

Es un sistema de planificación de recursos empresariales (ERP por sus siglas en inglés) desarrollado en la empresa DATYS y compuesto por varios módulos interconectados. Es multiplataforma e integra la información de todas las áreas de la empresa como Compras, Finanzas, Recursos Humanos y Contabilidad (9) (10), el cual permite (9) (11):

- Realizar filtros a la información deseada.
- Crear roles y usuarios.
- Crear y visualizar reportes de gastos.
- Crear versiones al presupuesto.

- Generar trazas detalladas de las operaciones que se realizan en el sistema.

### 1.2.2 Versat Sarasola

Es una herramienta que permite la planificación económica, el control y el análisis de gestión, orientado a la gestión empresarial y presupuestaria. Se estructura en un grupo de subsistemas en los cuales se procesan y contabilizan los documentos primarios, donde se anotan los movimientos, los recursos materiales, laborales y financieros que se utilizan en una entidad, además es compatible con plataformas Windows (12). Entre las funcionalidades que ofrece están (12) (13) (14):

- Permitir generar reportes y visualizar los presupuestos.
- Permitir el registro contable de los gastos y demás hechos económicos de la empresa.
- Permitir realizar operaciones con varias monedas.
- Permitir definir las cuentas y elementos de gastos a planificar.

### 1.2.3 Sistema para la planificación y control del presupuesto en el Centro GEYSED

Es un sistema informático que permite planificar y controlar el presupuesto del centro GEYSED perteneciente a la Universidad de las Ciencias Informáticas, al mismo tiempo permite generar reportes asociados a esas tareas a través de gráficos. Funciona en computadoras con memoria RAM[1] de 512 Mb o más. Además el sistema permite (15):

- Crear gráficos para hacer análisis.
- Generar reporte del control de los gastos reales del centro.
- Exportar reporte a PDF.
- Insertar partidas.
- Autenticar usuario.

En el estudio realizado a los sistemas desarrollados en el país, se apreció en sentido general, que no se ajustan a los requisitos del cliente. El sistema eTES posee funcionalidades como las descritas anteriormente, que podrán ajustarse a los requisitos del cliente. Sin embargo, este es un software privativo lo que trae consigo gastos por concepto de uso. El sistema Versat Sarasola es uno de los softwares de gestión empresarial más conocido por las empresas cubanas, solo es compatible con plataformas *Windows*, por lo que su uso iría en contra de la política de migración a software libre que lleva a cabo la universidad y trae consigo gastos en licencias de uso. Posee una interfaz poco

[1] Memoria principal de la computadora, donde residen programas y datos, sobre la que se pueden efectuar operaciones de lectura y escritura.

amigable para el usuario y es una aplicación de escritorio, lo que implica tener que darle mantenimiento al software en cada una de las estaciones de trabajo donde esté instalado. En el caso del “Sistema para la planificación y control del presupuesto en el Centro GEYSED”, este garantiza la planificación del presupuesto, pero no ofrece funcionalidades para controlar algunos elementos de la gestión financiera de la Facultad, tales como la matrícula, los alumnos ayudantes y el préstamo estudiantil, como tampoco permite la gestión de vales y viáticos; además es una aplicación de escritorio, características que lo hace inapropiado para el cliente. A pesar de ello, estos sistemas ofrecen características que servirán de guía y apoyo al sistema que se desea implementar, tales como:

- Generar reporte del presupuesto.
- Exportar reporte a PDF.
- Crear gráfico para realizar análisis.
- Realizar operaciones con varias monedas.

## 1.3 Sistemas similares externos

### 1.3.1 Openbravo ERP ComunityEdition

Es una aplicación de código abierto gestión empresarial del tipo ERP destinada para pequeñas y medianas empresas. Es un sistema completamente web que ha sido desarrollado siguiendo el Modelo-Vista-Controlador[2] (MVC), lo que facilita el desacoplamiento de las áreas de desarrollo, permitiendo el crecimiento sostenible de la aplicación y una mayor facilidad en el mantenimiento del código (16). A continuación, se explican algunas características de dicho sistema (17):

- Programar notificaciones para avisar al usuario cuando ocurre una determinada condición.
- Gestionar proyectos, gestionar el presupuesto, los costos y las compras, de manera integrada con el resto de la aplicación.
- Generar informes de presupuestos, para dar seguimiento a las acciones sobre estos.
- Realizar búsquedas personalizadas de los informes mediante filtros, garantizando obtener la información deseada.
- Gestionar roles, con el propósito de controlar qué pantallas son accesibles desde el menú y son visibles para los distintos usuarios.

[2] Es un patrón de arquitectura para aplicaciones software.

### 1.3.2 Odoo (OpenERP)

Odoo es uno de los software de gestión empresarial más populares del mundo. Es un completo sistema ERP de código abierto que cubre las necesidades de las áreas de: Contabilidad y Finanzas, Ventas, RRHH, Compras, Proyectos, Almacenes (SGA), CRM y Fabricación. Es una solución modular, lo que permite la utilización del módulo deseado e integrarlo posteriormente con otros, donde todos comparten la misma base de datos (18). Además, es un sistema multiplataforma, que posee las siguientes funcionalidades (18):

- Gestionar planes contables generales, analíticos y auxiliares.
- Gestionar cierres parciales.
- Permitir operaciones con varias divisas.
- Permitir la definición de diferentes grupos, roles y usuarios.
- Crear informes del estado presupuestario y el estado de las notificaciones.

### 1.3.3 SAP *Business One*

Es una solución completa de gestión de negocios privativa y es compatible solo con plataformas *Windows*. Diseñada específicamente para las *pequeñas y medianas empresas*, ayuda a aumentar la rentabilidad y el control, y automatiza los procesos de negocios. Esta solución incluye las funciones básicas necesarias para dirigir un negocio en crecimiento. Además, cuenta con once áreas de funcionalidad que ayudan a ampliar sus capacidades mucho más allá de su función administrativa primaria y que permiten optimizar todos sus procesos de negocios, algunas de estas son: ventas, finanzas, compras, reportes, etc. Ofrece las siguientes funcionalidades (19):

- Gestionar presupuestos.
- Generar informes gráficos.
- Definir cifras presupuestarias en cualquier divisa.
- Brindar un centro de costos presupuestarios.
- Generar reportes financieros.
- Visualizar reportes comparativos de las cifras proyectadas contra las reales.

En el estudio realizado a estos sistemas, se tuvo en cuenta indicadores tales como, las plataformas compatibles, funcionalidades y licencias de uso. Se apreció que, desde el punto de vista de solución, algunos como es el caso de SAP *Business One*, a pesar de contar con funcionalidades que podrían satisfacer al cliente, tales como crear presupuestos, crear reportes, visualizar gráficas, no se escoge como solución, además que es un sistema privativo y solo puede ejecutarse en sistemas operativos

*Windows*, lo que implicaría grandes gastos por conceptos de licencias de uso y en mantenimientos al sistema. En el caso de Odoo, es un sistema con características y funcionalidades acordes con las necesidades del cliente, el cual no se escoge como solución, debido a que el módulo de finanzas no permite: hacer cálculos internos asociados a partidas específicas del prepuesto, resaltar partidas que reflejan cambios con respecto al mes anterior, hacer versiones del presupuesto; por último en el caso de Openbravo, al igual que Odoo, brinda funcionalidades que se ajustan a los requerimientos del cliente, pero carece de otras que son críticas para el correcto control de los PGA de la Facultad 4 y el Centro FORTES, además ambos sistemas brindan soporte a la gestión empresarial de varios países en el que no se incluye a Cuba.

## 1.4 Metodología de desarrollo de software

El desarrollo de software es una tarea difícil de controlar y como resultado a este problema ha surgido una alternativa, las metodologías. Estas imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. El impacto de elegir la mejor metodología para un equipo en un determinado proyecto es muy importante para el éxito del producto, por lo que su selección debe ser analizada cuidadosamente.

Una metodología de aplicaciones informáticas es un conjunto de métodos que permiten sistematizar actividades. Estas indican cómo hacer las cosas a través de procedimientos bien escritos (20).

Las metodologías de desarrollo se pueden enmarcar en dos grandes grupos, las llamadas metodologías tradicionales y las metodologías ágiles. La diferencia fundamental entre ellas está en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, las herramientas y notaciones que se usarán.

Las metodologías tradicionales están guiadas por una fuerte planificación durante todo el proceso de desarrollo, donde se realiza una intensa etapa de análisis y diseño antes de la construcción del sistema (21). Se centran especialmente en el control del proceso mediante una rigurosa definición de roles, actividades, artefactos, herramientas, notaciones para el modelado y documentación detallada.

Sin embargo, las ágiles constituyen una solución a medida para entornos cambiantes, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto. Se basan en un ciclo de vida de desarrollo del software iterativo e incremental, corporativo (cliente y desarrolladores trabajan juntos constantemente con una cercana comunicación), sencillo y adaptable (permite realizar cambios de último momento), se apoyan en la adaptabilidad de cualquier cambio como medio para aumentar las posibilidades de éxito de un proyecto (21).

A partir de lo planteado anteriormente se decide trabajar con un enfoque ágil ya que las características de este se adecúan al entorno del trabajo que se desea crear, en el mismo existirá una estrecha relación entre los desarrolladores y el cliente, ya que se incluye a este último como otro miembro del equipo de desarrollo. La selección de este enfoque se basa además en que el mismo responde a las necesidades del cliente y permitirá ajustarse con facilidad a los cambios que pueda solicitar. Por último, se está en presencia de un equipo de trabajo pequeño (2 personas) y el cliente necesita el producto lo antes posible.

### 1.4.1 Análisis de metodologías ágiles

Entre las metodologías ágiles se encuentran: *Extreme Programing (XP), Scrum, Dynamic Systems Development Method (DSDM), Crystal Methodologies, Feature-Driven Development (FDD), Adaptive Software Development (ASD) y AUP*. Con el objetivo de hacer una correcta selección que satisfaga las particularidades del proyecto, se realizó un análisis de algunas de las metodologías de desarrollo ágiles más aceptadas internacionalmente.

#### 1.4.1.1 Programación Extrema (XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentarlos cambios (22).

XP es adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. El desarrollo de software con XP tiene un conjunto de características que lo distinguen del resto de las metodologías y tiene como objetivos establecer las mejores prácticas de Ingeniería de software en los desarrollos de proyectos, mejorar la productividad y garantizar la calidad del software que se está desarrollando, haciendo que este supere las expectativas del cliente. XP define las siguientes características (23):

- El cliente o el usuario se convierte en miembro del equipo.
- Se obtiene retroalimentación de usuarios y clientes desde el primer día.
- El software es liberado en entregas frecuentes tan pronto como sea posible.
- Los cambios se implementan rápidamente tal y como fueron sugeridos.
- Las metas en características, tiempos y costos son reajustadas permanentemente en función del avance real del proyecto.

- Comienza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- No introduce funcionalidades antes que sean necesarias.

Ventajas de la metodología XP (24):

- Apropiado para entornos inestables.
- La capacidad de respuesta ante un cambio, significa reducir su costo.
- Planificación transparente para los clientes.
- Permite definir en cada iteración cuales son los objetivos de la siguiente.
- Posibilita la retroalimentación de los usuarios.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

Desventajas de la metodología XP (24):

- Es recomendable emplearlo solo en proyectos a corto plazo.
- Altas comisiones en caso de fallar.

A continuación, se mencionan las prácticas propuestas por XP, estas son la concentración de los principios y valores que fundamentan las metodologías ágiles:

- Entregas pequeñas: se realizan pequeñas entregas del programa cada dos o tres semanas.
- Metáforas: desarrollada por los programadores al inicio del proyecto, define una historia de cómo funciona el sistema completo.
- Diseño simple: proporciona un sistema que cubra las necesidades inmediatas del cliente, ni más ni menos.
- Refactorización (*Refactoring*): permite a los equipos de programadores XP mejorar el diseño del sistema a través de todo el proceso de desarrollo.
- Programación en parejas: requiere que todos los programadores XP escriban su código en parejas, compartiendo una sola máquina.
- Propiedad colectiva del código: consiste en que nadie es el propietario de nada, todos son el propietario de todo.
- Integración continua: reúne el código y reconstruye el sistema varias veces al día, reduciendo los problemas de integración comunes en proyectos largos estilo cascada.
- 40 horas por semana: minimiza las horas extras y mantiene a los programadores frescos y se generará código de mayor calidad.

- Cliente en el sitio: da poder al cliente para determinar los requerimientos, definir la funcionalidad, señalar las prioridades y responder las preguntas de los programadores.
- El juego de planificación: se escriben las necesidades, definiendo las actividades que realizará el sistema. Se crean las Historias del usuario.

#### 1.4.1.2 Scrum

La metodología ágil *Scrum* se puede usar para gestionar y controlar desarrollos complejos de software y productos usando prácticas iterativas e incrementales. Es un soporte de proceso que incluye un conjunto de prácticas y roles predefinidos (25). Dentro de esta metodología existen varios roles como los que se muestran a continuación:

- *Scrum Master*: mantiene los procesos y trabajos junto con el jefe de proyecto.
- *ProductOwner*: representa a las personas implicadas en el negocio.
- *Team*: incluye a los desarrolladores.

*Scrum* permite la creación y organización de grupos de trabajos con el objetivo de que todos los miembros del equipo se encuentren en un mismo local. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos: el desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días, donde el resultado de cada sprint es un incremento ejecutable que se muestra al cliente; las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. Promueve la comunicación verbal y las disciplinas implicadas en el proyecto. Una de las mayores ventajas de *Scrum* es que es muy fácil de entender y requiere poco esfuerzo para comenzar a usarse.

**Ventajas:**

- Es fácil de aprender.
- Requiere muy poco esfuerzo para comenzarse a utilizar.
- Permite abarcar proyectos donde los requisitos de negocio están incompletos.
- Permite el desarrollo, testeo y correcciones rápidos.
- Mediante las reuniones diarias se ven claramente los avances y problemas.
- Como toda metodología ágil, obtiene mucho *feedback* del cliente.
- Facilita la entrega de productos de calidad a tiempo.

**Desventajas:**

- Si no se define una fecha de fin, los *stakeholders* siempre pedirán nuevas funcionalidades.
- Si una tarea no está bien definida puede incrementar costos y tiempos.
- Si el equipo no se compromete hay mucha probabilidad de fracasar.
- Solo funciona bien en equipos pequeños y ágiles.
- Se requieren miembros del equipo experimentados.
- Solo funciona cuando el *Scrum Manager* confía en su equipo.
- Que un miembro abandone el equipo durante el desarrollo puede conllevar grandes problemas.

#### 1.4.1.3 *Dynamic Systems Development Method* (DSDM)

Define el marco para desarrollar un proceso de producción de software. Surge con el objetivo de crear una metodología RAD (Desarrollo Rápido de Aplicaciones) unificada. Sus principales características son: proceso iterativo e incremental, el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases. El software para los desarrolladores es una solución y no un sistema y en lugar de priorizar el desarrollo se prefiere enfatizar en la entrega. DSDM puede complementar metodologías de XP, RUP o combinaciones de todas ellas. Es relativamente antiguo y constituye una metodología madura. Principios de DSDM (26):

- Involucrar al cliente es la clave para llevar un proyecto eficiente y efectivo.
- El equipo del proyecto debe tener el poder para tomar decisiones que son importantes para el progreso del proyecto.
- Se centra en la entrega frecuente de productos, asumiendo que entregar algo temprano es siempre mejor que entregar todo al final.
- El principal criterio de aceptación de entregables en DSDM reside en entregar un sistema que satisface las actuales necesidades de negocio.
- El desarrollo es iterativo e incremental.
- Todos los cambios durante el desarrollo son reversibles.
- Las pruebas son realizadas durante todo el ciclo vital del proyecto.
- La comunicación y cooperación entre todas las partes interesadas en el proyecto es un prerrequisito importante para llevar un proyecto efectivo y eficiente.

**Ventajas** (27):

- La calidad del producto es mejorada a través de la participación de los usuarios a lo largo del ciclo de vida del proyecto y la naturaleza iterativa del desarrollo.
- DSDM asegura desarrollos rápidos.
- Reduce los costos de proyectos a través de las ventajas mencionadas anteriormente.
- Permite realizar cambios de forma fácil.
- Permite la reutilización de aplicación a través de los módulos existentes.

**Desventajas** (27):

- Se necesita una alta participación de los usuarios en el desarrollo, para evitar que los desarrolladores asuman criterios que no son ciertos.
- No es una metodología de desarrollo común. El proceso es un tanto difícil de comprender.

#### 1.4.1.4 Desarrollo basado en funciones (FDD)

Enfoque ágil para el desarrollo de sistemas. Se enfoca en iteraciones cortas que entregan funcionalidades tangibles. Dicho enfoque no hace énfasis en la obtención de los requerimientos sino en cómo se realizan las fases de diseño y construcción. Sin embargo, fue diseñado para trabajar con otras actividades de desarrollo de software y no requiere la utilización de ningún modelo de proceso específico. Hace énfasis en aspectos de calidad durante todo el proceso e incluye un monitoreo permanente del avance del proyecto. Al contrario de otras metodologías, FDD afirma ser conveniente para el desarrollo de sistemas críticos. Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software (28).

**Características de FDD** (28):

- Se preocupa por la calidad, por lo que incluye un monitoreo constante del proyecto.
- Ayuda a contrarrestar situaciones como el exceso en el presupuesto, fallas en el programa o el hecho de entregar menos de lo deseado.
- Propone tener etapas de cierre cada dos semanas. Se obtienen resultados periódicos y tangibles.
- Se basa en un proceso iterativo con iteraciones cortas que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorear.
- Define claramente entregas tangibles y formas de evaluación del progreso del proyecto.
- No hace énfasis en la obtención de los requerimientos sino en cómo se realizan las fases de diseño y construcción.

**Ventajas de FDD** (28):

- Cada componente del producto final ha sido probado y satisface los requerimientos.
- Rápida respuesta a cambios de requisitos a lo largo del desarrollo.
- Entrega continua y en plazos cortos de software funcional.
- Trabajo conjunto entre el cliente y el equipo de desarrollo.
- Importancia de la simplicidad, al eliminar el trabajo innecesario.
- Atención continua a la excelencia técnica y al buen diseño.
- Mejora continua de los procesos y el equipo de desarrollo.

**Desventajas de FDD** (28):

- Problemas derivados de la comunicación oral. Este tipo de comunicación resulta difícil de preservar cuando pasa el tiempo y está sujeta a muchas ambigüedades.
- Fuerte dependencia de las personas. Como se evita en lo posible la documentación y los diseños convencionales, los proyectos ágiles dependen críticamente de las personas.

#### 1.4.1.5 AUP

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (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. El AUP aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas (*test driven development* -TDD en inglés).
- Modelado ágil.
- Gestión de Cambios ágil.
- Refactorización de Base de Datos para mejorar la productividad.

Al igual que en RUP en AUP se establecen cuatro fases que transcurren de manera consecutiva:

- Inicio: el objetivo de esta fase es obtener una comprensión común cliente - equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
- Elaboración: el objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- Construcción: durante la fase de construcción el sistema es desarrollado y probado en el ambiente de desarrollo.

- Transición: el sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación, aceptación y finalmente se despliega en los sistemas de producción.

AUP-UCI constituye la metodología de desarrollo de software propuesta para ser aplicable en todos los centros de desarrollo de software de la UCI. Su objetivo principal es unificar el proceso de desarrollo de software. La variación establecida para la UCI cuenta con tres fases (inicio, ejecución, cierre). Dentro de cada fase se encuentran ocho disciplinas (modelado del negocio, requisitos, análisis y diseño, implementación, pruebas internas, pruebas de liberación, pruebas de aceptación, gestión de la configuración, planeación de proyecto, monitoreo y control de proyecto) (29).

Después de un análisis de las características, ventajas y desventajas de las metodologías antes expuestas y de las características del proyecto que se desea realizar, se decidió adoptar XP para el desarrollo de la solución, partiendo de que el sistema a desarrollar no es de gran envergadura y se cuenta con poco tiempo para la implementación del mismo. Dicha metodología propicia un buen clima de trabajo, se basa en retro-alimentación continua entre el cliente y el equipo de desarrollo, presentando así una comunicación fluida entre todos los participantes y se ajusta a los cambios que pudieran ocurrir en el transcurso del proyecto.

## 1.5 Lenguaje de modelado

### 1.5.1 UML 2.0

El Lenguaje Unificado de Modelado (UML) se ha convertido en el lenguaje aceptado universalmente para los planos del diseño software (30), es usado para especificar, visualizar, construir y documentar artefactos de un sistema de software. Incluye aspectos esenciales tales como procesos de negocio, funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos. UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas.

UML no es una metodología de desarrollo, por lo que no dice cómo pasar del análisis al diseño y de este al código. No es una serie de pasos que llevan a producir código a partir de unas especificaciones y al no ser ninguno de los aspectos anteriores es independiente del ciclo de desarrollo a seguir, puede encajar en un tradicional ciclo en cascada, o en un evolutivo ciclo en espiral, o incluso, en los métodos ágiles de desarrollo (31).

### 1.5.2 Herramienta CASE

CASE es un acrónimo para *Computer-Aided Software Engineering*. Esencialmente, un CASE es una herramienta que ayuda al ingeniero de software a desarrollar y mantener software, también es definido como “Herramientas individuales para ayudar al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software (o mantenimiento)” (32).

#### 1.5.2.1 Visual Paradigm v8.0

El Visual Paradigm para UML es una herramienta CASE que utiliza como lenguaje de modelado al UML y soporta todas las etapas del ciclo de vida completo del desarrollo de software. Esta herramienta permite el desarrollo de aplicaciones informáticas, permite aumentar la calidad del software a través de la mejora de la productividad en el desarrollo y mantenimiento del software (33). Está diseñada para una amplia gama de usuarios interesados en construir sistemas de software fiables con el uso del paradigma orientado a objetos, incluyendo actividades como ingeniería de software, análisis de sistemas y análisis de negocios.

## 1.6 Marco de trabajo (*Framework*)

*Framework* define en términos generales, 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. Un *framework*, simplifica el desarrollo de las aplicaciones mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes, ya que encapsula operaciones complejas en instrucciones sencillas (34).

Este concepto se emplea en muchos ámbitos del desarrollo de sistemas de software y hace referencia a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un *framework* se puede considerar como una aplicación genérica incompleta y configurable a la que es posible añadirle las últimas piezas para construir una aplicación concreta.

Para el desarrollo de la aplicación se escoge como marco de trabajo a Grails, debido a que es de interés del cliente que esta aplicación se integre con una herramienta desarrollada con esta tecnología. Posteriormente se detallan los *frameworks* y los lenguajes de programación empleados para el desarrollo de la aplicación.

### 1.6.1 Grails

Grails es una plataforma de código abierto, para el desarrollo web, que ofrece nuevos niveles de productividad de los desarrolladores mediante la aplicación de principios como la “convención sobre configuración”, “no te repitas” y “mejoremos la rueda, no la reinventemos”. Grails ayuda a los equipos de desarrollo que abrazan las metodologías ágiles a entregar aplicaciones de calidad en menor cantidad de tiempo, y a centrarse en lo realmente importante: la creación de aplicaciones de alta calidad y fáciles de usar. Grails, naturalmente, complementa el desarrollo de aplicaciones Java, ya que se basa en Spring y sobre la base de Groovy, el principal idioma dinámico para la plataforma Java (35).

Este *framework* funciona bajo el MVC (Modelo Vista Controlador) y utiliza una cuarta capa llamada servicio para separar la inteligencia de negocio logrando así un mejor entendimiento a la hora de analizar la aplicación. Grails también ofrece la posibilidad de crear plantillas no solo CSS, sino que también incluyen código HTML mediante las plantillas. Además, cuenta con un lenguaje llamado *Groovy Server Pages* (GSP) para poder programar directamente sobre la aplicación web en código HTML (36).

**Características** (37)**:**

- Ofrece un *framework* web de alta productividad para la plataforma Java.
- Ofrece un *framework* consistente que reduce la confusión y que es fácil de aprender.
- Ofrece documentación para las partes del *framework* relevantes para sus usuarios.
- Proporciona lo que los usuarios necesitan en áreas que a menudo son complejas e inconsistentes.
- Bibliotecas de etiquetas dinámicas para crear fácilmente componentes web.

### 1.6.2 Bootstrap

*Bootstrap* fue desarrollado como marco de trabajo para fomentar la consistencia a través de herramientas internas, además facilita la construcción de aplicaciones que utilicen el patrón **ResponsiveDesign**[3] mediante tecnologías CSS3, HTML5 y JavaScript (38) y (39). Este marco de

[3] Es una serie de prácticas aplicadas al diseño web que le permiten al usuario acceder a un sitio web desde diferentes medios, partiendo de la base que todo diseño web debe estar centrado en la experiencia del usuario al momento de acceder a un sitio web.

trabajo permite varias formas de poner un texto en énfasis, otorgar estilos a las abreviaturas y los formatos de negrita y cursiva, alinear textos, etc.

Algunas de sus características son:

- Permite definir diferentes *layouts*.
- Amplio conjunto de componentes para el desarrollo.
- Posibilidad de compilar el *framework* con diferentes valores.
- Soporte HTML5 y CSS3.

La selección de *Bootstrap* como *framework* se basa en una mejor personalización de la vista de la propuesta solución.

## 1.7 Lenguajes de Programación

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo por máquinas como las computadoras (40). Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. En el desarrollo de la aplicación se utilizaron varios lenguajes de programación ya que Groovy admite sentencias de Java.

### 1.7.1 Groovy

Groovy es un lenguaje de programación Orientada a Objeto que se basa en Java. Tiene características similares a Python, Ruby, Perl y Smalltalk. Es simple de entender y dinámico lo cual hace que sea muy cómodo para los desarrolladores. Además, es un lenguaje muy compacto lo cual permite escribir menos código para una misma sintaxis que en Java y de esta manera se convierte en un salto natural para los programadores que desarrollan en Java.

Desde Groovy se puede acceder directamente a todas las Interfaces de Programación de Aplicaciones (API, del inglés *Application Programming Interface*) existentes en Java. El bytecode (clase Java) generado en el proceso de compilación es totalmente compatible con el generado por el lenguaje Java para la Java Virtual Machine (JVM), por tanto, puede usarse directamente en cualquier aplicación Java. Todo lo anterior unido a que la mayor parte del código escrito en Java es totalmente válido en Groovy, hacen que este lenguaje sea de muy fácil adopción para programadores Java; la curva de aprendizaje

se reduce mucho en comparación con otros lenguajes que generan *bytecode* para la JVM, tales como Jython o JRuby. Groovy puede usarse también de manera dinámica como un lenguaje de scripting (41).

### 1.7.2 Java

Java es un lenguaje de programación de propósito general, concurrente, Orientado a Objeto y basado en clases, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

*Java Enterprise Editions* es un conjunto de componentes que forman una plataforma de programación para el desarrollo y ejecución de aplicaciones desarrolladas en lenguaje de programación Java con arquitectura de N capas distribuidas, que se apoya ampliamente en componentes de software modulares ejecutándose en un servidor de aplicaciones.

La sintaxis del lenguaje de programación Java deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a *bytecode* que puede ejecutarse en cualquier máquina virtual Java sin importar la arquitectura de la computadora subyacente (42).

**Características** (42)**:**

- La orientación a objeto, se refiere a un método de programación y al diseño del lenguaje.
- La independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware.
- En Java el problema fugaz de memoria se evita en gran medida gracias a la recolección de basura. El programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java (*Java runtime*) es el responsable de gestionar el ciclo de vida de los objetos.

Entornos de funcionamiento:

- En dispositivos móviles y sistemas empotrados.
- En el navegador web.
- En aplicaciones de escritorio.
- En sistema servidor.

### 1.7.3 JavaScript

JavaScript es un lenguaje de programación de secuencia de comandos (*scripts*) Orientado a Objeto. Estos *scripts* consisten en funciones que son llamados desde el propio HTML cuando algún evento sucede. JavaScript es actualmente utilizado en internet, junto con las páginas web (HTML o XHTML), está directamente incluido en ellas y mejora una página HTML, añadiendo interacción del usuario, animación, ayudas a la navegación, entre otros. Los programas escritos en JavaScript se pueden probar en cualquier navegador sin necesidad de procesos intermedios (43) y (44). La selección de JavaScript como lenguaje de *script* del lado del cliente se basa en utilizar la librería JQuery, que es empleada por el *Framework Bootstrap*.

### 1.7.4 HTML

HTML es el lenguaje de marcas de hipertexto, básico de la web para organizar su contenido, compartir información, crear documentos y aplicaciones que puedan ser utilizadas en cualquier lugar. Es un estándar a cargo de la W3C (*World Wide Web Consortium*), organización dedicada a la estandarización de la mayoría de las tecnologías ligadas a la web (45).

La última versión, HTML5, que provee básicamente tres características: estructura, estilo y funcionalidad, pretende corregir los problemas con los que los desarrolladores web se encuentran, así como rediseñar el código, actualizándolo a las nuevas necesidades que demanda la web. HTML5 propone estándares para cada aspecto de la web y también un propósito claro para cada una de las tecnologías involucradas (46) y (47).

### 1.7.5 CSS

La hoja de estilo en cascada es un lenguaje, que trabaja junto con HTML para proveer estilos visuales a los elementos del documento (48). Este lenguaje es un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML (47). La versión 3 de CSS está dividida en varios documentos separados, llamados “módulos”, cada módulo incorpora nuevas funcionalidades a las definidas en la versión anterior para conservar la compatibilidad. Algunas de estas funcionalidades son:

- Incluye la posibilidad de aplicar sombra a los elementos.
- Nuevas alternativas para dibujar bordes.
- Capacidad de rotación de elementos.
- Opciones de transformación de elementos.

CSS3 fue seleccionado como lenguaje para escribir el estilo visual de la propuesta solución porque es un estándar determinado por la W3C, además permite la separación de la estructura del documento de su presentación.

Para la necesaria evolución de las web HTML, CSS y JavaScript se convirtieron en la más perfecta combinación, donde HTML provee los elementos estructurales, CSS se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista y JavaScript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales (45) y (47).

## 1.8 Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado (IDE, del inglés *Integrated Development Environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes (49).

### 1.8.1 NetBeans

*NetBeans* IDE es una herramienta escrita en Java, porque puede ser utilizado desde cualquier sistema operativo que tenga la máquina virtual de Java. Es un IDE multilenguaje completo y modular que tiene soporte para Java SE, Java EE, Java ME y con vinculación de gran cantidad de módulos (*plugins*). Cuenta con depurador, perfilador, herramientas para refactorizaciones, completamiento de código, editores y herramientas para los lenguajes XML, HTML, CSS, PHP, JavaScript, etc (51).

## 1.9 Sistema Gestor de Bases de Datos (PostgreSQL)

PostgreSQL es un sistema gestor de bases de datos objeto-relacional de carácter libre muy utilizado últimamente por su fácil comprensión. PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos mediante un sistema denominado Acceso Concurrente Multiversión (MVCC, del inglés *Multiversion Concurrency Control*) (50).

**Principales características** (50):

- Alta concurrencia: Cada usuario obtiene una visión consistente de lo último a lo que se le hizo *commit*. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.
- Amplia variedad de tipos nativos: PostgreSQL provee nativamente soporte para números de precisión arbitraria, texto de largo ilimitado, figuras geométricas (con una variedad de funciones

asociadas) y direcciones IP (IPv4 e IPv6). Adicionalmente los usuarios pueden crear sus propios tipos de datos.

**Ventajas** (50)**:**

- Fácil de administrar.
- Su sintaxis SQL es estándar y fácil de aprender.
- Multiplataforma.

Se estará trabajando con PostgreSQL 9.4 para la confección de la base de datos, pues los autores ya han interactuado con dicha herramienta anteriormente, además permite la interacción con el software de forma simultánea, es fácil de manejar y administrar.

## Conclusiones parciales

Con la realización de este capítulo se dejaron fundamentadas las definiciones relacionadas con la investigación que ayudaron a un mejor entendimiento de la situación planteada. Se definieron las herramientas fundamentales seleccionando XP como metodología para guiar el proceso de desarrollo. Se decidió utilizar Groovy y Java como lenguajes de programación, Grails como marco de desarrollo web, NetBeans 8.1 como entorno de desarrollo, Apache Tomcat 7.0 como servidor de aplicaciones, Visual Paradigm 8.0 como herramienta CASE y PostgreSQL 9.4 como servidor de base de datos.

# CAPÍTULO 2

# EXPLORACIÓN Y PLANIFICACIÓN

## 2.1 Introducción

La metodología XP define en sus fases de exploración y planificación el alcance general del proyecto, a partir de las estimaciones del tiempo de desarrollo que realiza el programador a las “Historias de usuario” (HU) y el orden en que estas serán implementadas y entregadas respectivamente.

En el presente capítulo se describe el sistema a desarrollar, así como los roles que interactúan en el mismo. Se muestran los principios, prácticas y técnicas para la modelación del sistema propuesto entre las que se encuentran: HU, plan de iteraciones, plan de entregas, tarjetas Clases–Responsabilidad–Colaborador (CRC) y los patrones arquitectónicos.

## 2.2 Modelo conceptual

La metodología XP no requiere de un modelo conceptual, pero es de gran ayuda para el equipo de desarrollo la realización de un modelo conceptual, para una mejor comprensión de los conceptos asociados al negocio. A continuación, se describe la lógica de un sistema de negocios, donde se representan los principales conceptos asociados al control de los presupuestos de la Facultad 4 y el Centro FORTES.

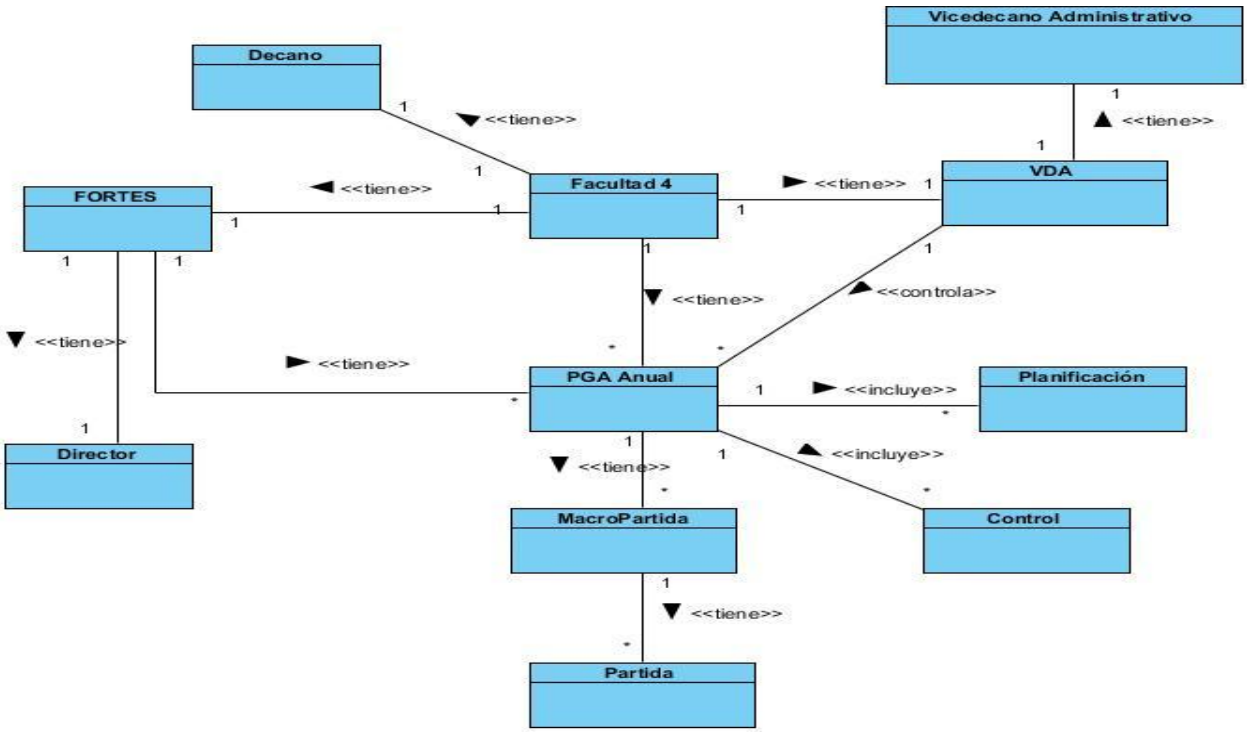


*Figura #1: Diagrama conceptual del negocio*

## 2.3 Descripción de los conceptos del modelo

- Facultad 4: Facultad perteneciente a la Universidad de las Ciencias Informáticas, en la cual se llevan a cabo los procesos formación de los estudiantes, superación de profesionales, proyectos de extensión y producción de software. Esta facultad es dirigida por el Decano y tiene asociado el Centro de Desarrollo FORTES.

- Vicedecanato Administrativo: Área de la Facultad 4 encargada de controlar la gestión financiera de dicha Facultad.

- FORTES: Centro de Tecnologías para la Formación cuya misión es desarrollar tecnologías que permitan ofrecer servicios y productos para la implementación de soluciones de formación aplicando las TIC (Tecnologías de la Información y las Comunicaciones), integrando los procesos de formación, producción e investigación.

- PGA Anual: tipo de presupuesto flexible que contempla los gastos administrativos.

- Director: persona que dirige y controla el Centro FORTES.

- Decano: persona que dirige y controla la Facultad 4.

- Vicedecano Administrativo: persona que dirige y controla los procesos pertenecientes al área del Vicedecanato Administrativo.

- Planificación: plan general metódicamente organizado y frecuentemente de gran amplitud, para obtener un objetivo determinado, la cual define los objetivos organizacionales y establece las metas que se pretenden alcanzar en un tiempo determinado (1).

- Control: es un mecanismo, que en conjunto con la planificación ayudan a fijar los objetivos económicos (tanto en el aspecto de los ingresos, gastos e inversiones) y vigilar que las acciones diseñadas para su consecución son las adecuadas para lograrlo; además sirve para detectar en qué grado se van cumpliendo las expectativas de la administración (6).

- Macropartida: conjunto de partidas relacionadas entre sí.

- Partida: se refiere a los elementos de gasto del presupuesto. Se puede utilizar como norma de comparación.

## 2.4 Listas de Funcionalidades

A continuación, se muestran las listas de funcionalidades y las características no funcionales del sistema:

*Tabla #1: Listas de funcionalidades y características no funcionales*

| No. | Lista de funcionalidades |
|---|---|
| 1. | Gestionar plan del presupuesto anual |
| El sistema debe permitir adicionar, eliminar, editar y consultar los datos de un presupuesto. De cada presupuesto se almacenará el año, fecha de notificación, vía de notificación, persona que notifica, fecha de creación y tipo de moneda. | |
| 2. | Calcular los montos para las macropartidas |
| El sistema calcula el monto por cada macropartida a partir de la sumatoria de los montos de las partidas asociadas, tanto en el plan como en las ejecuciones. | |
| 3. | Gestionar partida |
| El sistema debe permitir adicionar, editar, consultar, eliminar una partida y macropartidas del presupuesto anual; introducir los montos asociados al plan y a la ejecución real por cada partida. De cada partida se almacenará su nombre y el monto asociado tanto al plan como a la ejecución real. | |

| | |
|---|---|
| 4. | Asociar partida a macropartida |
| El sistema debe permitir asociar partida a macropartida, donde cada macropartida puede tener varias partidas. | |
| 5. | Gestionar reporte de ejecución mensual |
| El sistema debe permitir adicionar, editar, consultar y eliminar los reportes de ejecución mensual del presupuesto activo. De cada reporte se almacenará la fecha de notificación, datos de notificación, fecha de creación, cantidades ejecutadas por partidas, porciento de ejecución y observaciones. | |
| 6. | Gestionar datos de alumnos ayudantes |
| El sistema debe permitir adicionar, editar, consultar y eliminar los datos de los alumnos ayudantes. El sistema debe almacenar por cada alumno ayudante el nombre, grupo y los años de servicio. | |
| 7. | Gestionar datos de estudiante de orden 18 |
| El sistema debe permitir adicionar, editar, consultar y eliminar los datos de los estudiantes orden 18. El sistema debe almacenar por estudiante orden 18 el año académico. | |
| 8. | Gestionar datos de viáticos |
| El sistema debe permitir adicionar, editar, consultar y eliminar los datos de los viáticos del presupuesto anual. El sistema calculará por cada viático un total (sumatoria de los montos por conceptos de alojamiento, alimentación y transporte), el cual podrá ser consultado. | |
| 9. | Gestionar datos de matrícula estudiantil |
| El sistema debe permitir adicionar, editar, consultar y eliminar datos de matrícula de estudiantes por año. | |
| 10. | Calcular el gasto asociado al estipendio |
| El sistema debe permitir calcular el gasto asociado al estipendio, aquí inciden los datos asociados a la matrícula de los estudiantes, alumnos ayudantes y orden 18. | |
| 11. | Resaltar las partidas que sufran cambios |
| El sistema debe permitir resaltar las partidas que sufran cambios en los montos de ejecución, con respecto al mes anterior en el presupuesto anual. | |
| 12. | Crear resumen de análisis |
| El sistema debe permitir crear resumen de análisis, por cada uno deben almacenarse la fecha, tipo de análisis que se realiza, participantes, descripción del análisis asociado a un plan y una ejecución. | |
| 13. | Gestionar usuario |

| | |
|---|---|
| El sistema debe permitir adicionar, editar, y eliminar usuario. De cada usuario se registrará su nombre y apellidos y rol que desempeña. | |
| 14. | Asignar rol |
| El sistema debe permitir asignar rol a los usuarios del sistema. Para poder realizar esta acción es necesario estar autenticado como Vicedecano Administrativo. | |
| 15. | Gestionar registro anual de vales |
| El sistema debe permitir adicionar, editar, eliminar y consultar registro anual de vales. | |
| 16. | Gestionar versión del presupuesto |
| El sistema debe permitir adicionar, eliminar, editar y consultar versiones de presupuesto. De cada versión del presupuesto se almacenará la fecha de notificación, vía de notificación, persona que notifica, fecha de actualización, incluye resumen de diferencias con la versión anterior. | |
| 17. | Activar versión del plan del presupuesto anual |
| El sistema debe permitir activar versiones del presupuesto. Solo pueden existir 2 presupuestos (uno en CUC y uno en CUP) activos por cada área. | |
| 18. | Graficar reporte |
| El sistema debe permitir graficar cada reporte de ejecución por año y por mes, con el objetivo de realizar comparaciones con presupuestos de períodos anteriores. Además el sistema debe permitir imprimir y exportar dichos gráficos. | |
| 19. | Autenticar usuario |
| El sistema debe permitir que un usuario se autentique insertando su identificador y contraseña. Este usuario debe tener previa autorización del Vicedecano Administrativo para acceder al sistema. | |
| 20. | Generar trazas |
| El sistema debe permitir generar y guardar las trazas asociadas a las acciones de los usuarios del sistema, así como consultar las mismas. Por cada traza deben almacenarse la fecha, la acción y el usuario. | |
| 21. | Gestionar áreas |
| El sistema debe permitir adicionar, editar, eliminar y consultar las áreas asociadas al presupuesto. Por cada área debe almacenarse el nombre y el tipo de área. | |
| **Características no funcionales** | |
| 1. | **Requisito de facilidad de uso:** para poder hacer uso de la aplicación el usuario debe poseer conocimientos básicos de informática y del trabajo con alguna aplicación donde el usuario interactúa constantemente con el sistema. |

| | |
|---|---|
| 2. | **Requisitos de software:** la aplicación debe ser capaz de funcionar en cualquier sistema operativo que posea la máquina virtual de Java en su versión 7 o superior, tenga el navegador Mozilla 35 o superior; además el servidor debe tener instalado el servidor web Apache 7.0 o superior y PostgreSQL v9.4 o superior. |
| 3. | **Requisitos de hardware:** el servidor donde esté montado la aplicación debe poseer un procesador con velocidad superior a los 2.4Ghz, al menos 4Gb de memoria RAM y 750Mb de espacio libre en disco como mínimo. Las computadoras clientes deben tener al menos 1 GB de RAM y una velocidad de procesador de al menos 1.2Ghz. |
| 4. | **Requisitos de interfaz gráfica:** la interfaz gráfica de la aplicación debe ser intuitiva, amigable y fácil de aprender y utilizar por parte de los usuarios que hagan uso ella. |
| 5. | **Seguridad:** el sistema debe mostrar las funcionalidades de acuerdo al nivel de acceso del usuario activo y lo protegerá contra acciones que puedan afectar la integridad de los datos. |

## 2.5 Usuarios del sistema

Los usuarios del sistema son los roles que interactúan directamente con la aplicación, a continuación, se describen los mismos con sus responsabilidades.

*Tabla #2: Usuarios del sistema*

| Usuarios del sistema | Descripción |
|---|---|
| Vicedecano Administrativo | Tiene la responsabilidad de crear el plan del presupuesto anual, crear los reportes de ejecución mensual de todas las áreas. |
| Decano | Tiene permisos de consulta y edición de los datos entrados por el Vicedecano Administrativo, además puede consultar las trazas de todos los usuarios del sistema. |
| Director de Centro | Tiene permisos de administración y edición de los datos relacionados con los presupuestos del Centro. |
| Administrador | Posee todos los permisos. |

## 2.6 Historia de usuario

El cliente describe las características que el sistema debe poseer a través de las HU que son empleadas por la metodología XP como técnica para especificar la lista de funcionalidades y las características no funcionales (51).

Por la necesidad del cliente fueron detalladas un total de 21 HU, en este epígrafe sólo se exponen las principales, las restantes se encuentran en el Anexo 1 del documento.

*Tabla #3: Historia de usuario Gestionar plan de presupuesto anual*

| **Historia de usuario** | |
|---|---|
| **Número:** 1 | **Nombre del requisito:** Gestionar plan del presupuesto anual |
| **Programador:** Alexis Borges Mora | **Iteración asignada:** 1 |
| **Prioridad:** Alta | **Tiempo estimado:** 1 semana |
| **Riesgo en desarrollo:** Alto | **Tiempo real:** 4 días |
| **Descripción:** el sistema debe permitir adicionar, eliminar, editar y consultar los datos de un presupuesto. De cada presupuesto se almacenará el año, fecha de notificación, vía de notificación, persona que notifica, fecha de creación y tipo de moneda. | |
| **Observaciones:** el usuario deberá estar autenticado por el rol del Vicedecano Administrativo, Decano o Director del Centro. Este último solo tendrá los permisos de modificación y eliminación en los presupuestos asociados al Centro. El sistema les indicará los campos que son obligatorios para gestionar los datos del presupuesto anual. Debe permitir asociarle las macropartidas. | |

*Tabla #4: Historia de usuario Gestionar partida*

| **Historia de usuario** | |
|---|---|
| **Número:** 3 | **Nombre del requisito:** Gestionar partida |
| **Programador:** Alexis Borges Mora | **Iteración asignada:** 1 |
| **Prioridad:** Alta | **Tiempo estimado:** 1 semana |
| **Riesgo en desarrollo:** Alto | **Tiempo real:** 6 días |
| **Descripción:** el sistema debe permitir adicionar, editar, consultar, eliminar una partida y macropartidas del presupuesto anual; introducir los montos asociados al plan y a la ejecución real por cada partida. | |
| **Observaciones:** el usuario debe estar autenticado como Vicedecano Administrativo, Decano o Director del Centro. El sistema indicará los campos necesarios para gestionar las partidas. | |

*Tabla #5: Historia de usuario Gestionar datos de viático*

| **Historia de usuario** | |
|---|---|
| **Número:** 8 | **Nombre del requisito:** Gestionar datos de viático |
| **Programador:** Victor Abel Reyes | **Iteración asignada:** 2 |
| **Prioridad:** Alta | **Tiempo estimado:** 1 semana |
| **Riesgo en desarrollo:** Alto | **Tiempo real:** 7 días |
| **Descripción:** el sistema debe permitir adicionar, editar, consultar y eliminar los datos de los viáticos del presupuesto anual. El sistema calculará por cada viático un total (sumatoria de los montos por conceptos de alojamiento, alimentación y transporte), el cual podrá ser consultado. | |
| **Observaciones**: el usuario debe estar autenticado como Vicedecano Administrativo, Decano o Director del Centro y el sistema indicará los datos necesarios para realizar estas acciones. Los datos asociados a los viáticos podrán ser consultados por los usuarios Vicedecano Administrativo, Decano y Director del Centro, una vez hayan sido creados. | |

*Tabla #6: Historia de usuario Gestionar registro anual de vales*

| **Historia de usuario** | |
|---|---|
| **Número:** 15 | **Nombre del requisito:** Gestionar registro anual de vales |
| **Programador:** Victor Abel Reyes | **Iteración asignada:** 4 |
| **Prioridad:** Alta | **Tiempo estimado**:1 semana |
| **Riesgo en desarrollo:** Alto | **Tiempo real:** 5 días |
| **Descripción:** el sistema debe permitir adicionar, editar, eliminar y consultar registro anual de vales, además de poder filtrar los vales según la partida a la que estén asociados. | |
| **Observaciones:** el usuario debe estar autenticado con el rol de Vicedecano Administrativo, Decano o Director del Centro. Los vales podrán ser consultados luego de ser insertados. | |

## 2.7 Planificación

La planificación es una fase corta de la metodología XP, en la que el cliente establece el orden en que deberían implementarse las HU y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Las estimaciones de esfuerzo asociado a la implementación de las HU la establecen los desarrolladores utilizando como medida el punto, un punto equivale a una semana ideal de programación. Generalmente requieren de uno a tres puntos (52).El resultado de esta fase es un Plan de Iteraciones y un Plan de Entregas, que son de suma importancia para el desarrollo del proyecto.

### 2.7.1 Plan de Iteraciones

En esta fase se incluyen varias iteraciones del sistema antes de ser entregado. Cada iteración producirá un conjunto de casos de pruebas funcionales para cada una de las HU agendadas para la iteración. Las iteraciones son empleadas para medir la evolución del proyecto; una iteración terminada sin errores es una medida de avance (53). A continuación, se muestra el Plan de iteraciones de la propuesta solución.

*Tabla #7: Planificación de las iteraciones*

| Iteraciones | Orden de las HU a implementar | Cantidad de tiempo de trabajo |
|---|---|---|
| Iteración 1 | HU1. Gestionar plan del presupuesto anual<br>HU2. Calcular los montos para las macropartidas<br>HU3. Gestionar partida<br>HU4. Asociar partida a macropartida | 3 semanas |
| Iteración 2 | HU5. Gestionar reporte de ejecución mensual<br>HU6. Gestionar datos de Alumnos ayudantes<br>HU7. Gestionar datos de estudiante de orden 18<br>HU8. Gestionar datos de viático | 3 semanas |
| Iteración 3 | HU9. Gestionar datos de matrícula<br>HU10. Calcular el gasto asociado al estipendio<br>HU11 Resaltar las partidas que sufran cambios<br>HU12. Crear resumen de análisis | 3 semanas |
| Iteración 4 | HU13. Gestionar usuario<br>HU14. Asignar rol<br>HU15. Gestionar registro anual de vales<br>HU16. Gestionar versión del presupuesto | 3 semanas |
| Iteración 5 | HU17. Activar versión del plan del presupuesto anual<br>HU18. Graficar reporte<br>HU19. Autenticar usuario<br>HU20. Generar trazas<br>HU21. Gestionar áreas | 3 semanas |

### 2.7.2 Plan de entregas

El Plan de entregas establece cuales HU serán agrupadas para conformar una entrega y el orden de las mismas. Este constituye un documento oficial por el cual los clientes le exigen a los desarrolladores la entrega de las distintas versiones del producto (53). A partir de una reunión con el cliente, teniendo en cuenta las necesidades trazadas por el mismo y las evaluaciones realizadas por los desarrolladores, el equipo de desarrollo definió el siguiente Plan de Entregas.

*Tabla #8: Plan de entregas*

| **Entregable** | **1ra entrega (1ra semana de noviembre)** | **2da entrega (4ta semana de noviembre)** | **3ra entrega (2da semana de enero)** | **4ta entrega (1ra semana de febrero)** | **5ta entrega (4ta semana de febrero)** |
|---|---|---|---|---|---|
| Sistema para controlar los PGA de la Facultad 4 y el Centro FORTES | Versión 0.1 | Versión 0.2 | Versión 0.3 | Versión 0.4 | Versión 0.5 |

*Tabla #9: Funcionalidades disponibles para la entrega del producto*

| **Historia de usuario** | **1ra entrega** | **2da entrega** | **3ra entrega** | **4ta entrega** | **5ta entrega** |
|---|---|---|---|---|---|
| HU1.Gestionar plan del presupuesto anual | X | | | | |
| HU2.Calcular los montos para las macropartidas | X | | | | |
| HU3.Gestionar partida | X | | | | |
| HU4.Asociar partida a macropartida | X | | | | |
| HU5.Gestionar reporte de ejecución mensual | | X | | | |
| HU6.Gestionar datos de Alumnos ayudantes | | X | | | |
| HU7.Gestionar datos de estudiante de orden 18 | | X | | | |
| HU8.Gestionar datos de viático | | X | | | |
| HU9.Gestionar datos de matrícula estudiantil | | | X | | |
| HU10.Calcular el gasto asociado al | | | X | | |

| | | | | | |
|---|---|---|---|---|---|
| estipendio | | | | | |
| HU11.Resaltar las partidas que sufran cambios | | | X | | |
| HU12.Crear resumen de análisis | | | X | | |
| HU13.Gestionar usuario | | | | X | |
| HU14. Asignar rol | | | | X | |
| HU15.Gestionar registro anual de vales | | | | X | |
| HU16.Gestionar versión del presupuesto | | | | X | |
| HU17.Activar versión del plan del presupuesto anual | | | | | X |
| HU18.Graficar reporte | | | | | X |
| HU19.Autenticar usuario | | | | | X |
| HU20.Generar trazas | | | | | X |
| HU21.Gestionar áreas | | | | | X |

## 2.8 Tarjetas Clase – Responsabilidad-Colaborador CRC

Las tarjetas CRC son una técnica para la representación de sistemas Orientado a Objetos, que sirven para diseñar el mismo en conjunto entre todo el equipo. Su objetivo es hacer un inventario de las clases que se necesitarán para implementar el sistema y la forma en que van a interactuar (54). A continuación, se muestran las tarjetas relacionadas con las HU descritas anteriormente, el resto se encuentra en los anexos de la investigación.

*Tabla #10: Tarjeta CRC PresupuestoController*

| **Clase: PresupuestoController** | |
|---|---|
| **Responsabilidad** | **Colaborador** |
| Se encarga de adicionar, editar, consultar y eliminar los presupuestos. | Reportemensual<br>Macropartida<br>Version |

*Tabla #11: Tarjeta CRC ValesController*

| **Clase: ValesController** |
|---|

| Responsabilidad | Colaborador |
|---|---|
| Se encarga de adicionar, editar, consultar y eliminar registro anual de vales. | Partida |

*Tabla #12: Tarjeta CRC ViaticoController*

| Clase: ViaticoController | |
|---|---|
| **Responsabilidad** | **Colaborador** |
| Se encarga de adicionar, editar, consultar y eliminar los datos de los viáticos del presupuesto anual. | |

*Tabla #13: Tarjeta CRC PartidaController*

| Clase: PartidaController | |
|---|---|
| **Responsabilidad** | **Colaborador** |
| Se encarga de adicionar, editar, consultar y eliminar partidas del presupuesto anual. | Values<br>Vales<br>Macropartida<br>Reportemensual |

## 2.9 Patrones arquitectónicos

Un patrón arquitectónico especifica un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes (55). Es necesario definir las características del patrón Modelo – Vista – Controlador debido a que es la arquitectura que sigue el *framework Grails*, que es el que guía la presente investigación.

### 2.9.1 Modelo – Vista–Controlador (MVC)

Patrón que separa los datos y la lógica de negocio de una aplicación, de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Define componentes para la representación de la información y la interacción del usuario. Se basa en las ideas de reutilización de código y la separación de conceptos, busca facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento (57).

- **Modelo**: es el conjunto de clases que representan la información del mundo real que el sistema debe procesar, sin tomar en cuenta ni la forma en que esta va a ser mostrada ni los mecanismos que hacen que los datos estén dentro del modelo.
- **Vista:** son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo.
- **Controlador:** se encarga de gestionar la aplicación de la lógica del negocio, de abrir y cerrar vistas y recibir las órdenes del usuario.
- **Servicios:** contiene los componentes encargados de implementar la lógica de negocio de la aplicación.

El equipo de Grails desaconseja la inclusión de la lógica de negocio en la capa de control, y recomienda hacerlo mediante servicios, con el objetivo de poder reutilizar mayor cantidad de código y de no colocar demasiado código en el controlador, lo cual traería consigo una fase de mantenimiento complicada debido a un código incomprensible. Por lo tanto, en la presente propuesta de solución, el Controlador contendrá los componentes que reciben las órdenes del usuario y gestionará la aplicación de la lógica del negocio.

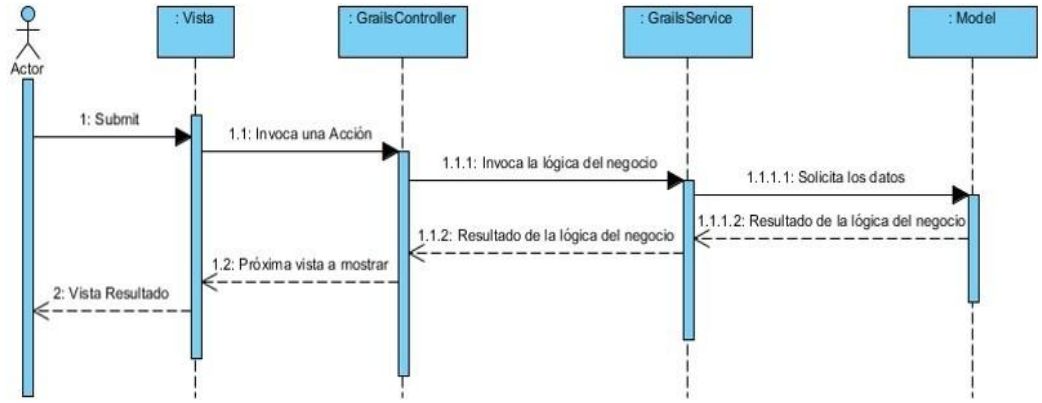


*Figura #2: Patrón Modelo-Vista-Controlador + Servicios*

En la Figura #2 se evidencia como como quedaría el flujo de los datos del MVC con la capa de servicios. El usuario pulsa un *link* o un botón *submit* en el formulario, la solicitud llega a la acción correspondiente en el controlador. Este invoca el servicio encargado de implementar la lógica del negocio, el servicio solicita los datos necesarios de la clase del dominio, la cual se la envía al servicio, en base al resultado de la invocación del servicio, el controlador decide cual es la próxima vista a mostrar y le pide a la capa de presentación que se la muestre al usuario.

A continuación, se muestra la representación del patrón MVC de la propuesta de solución:

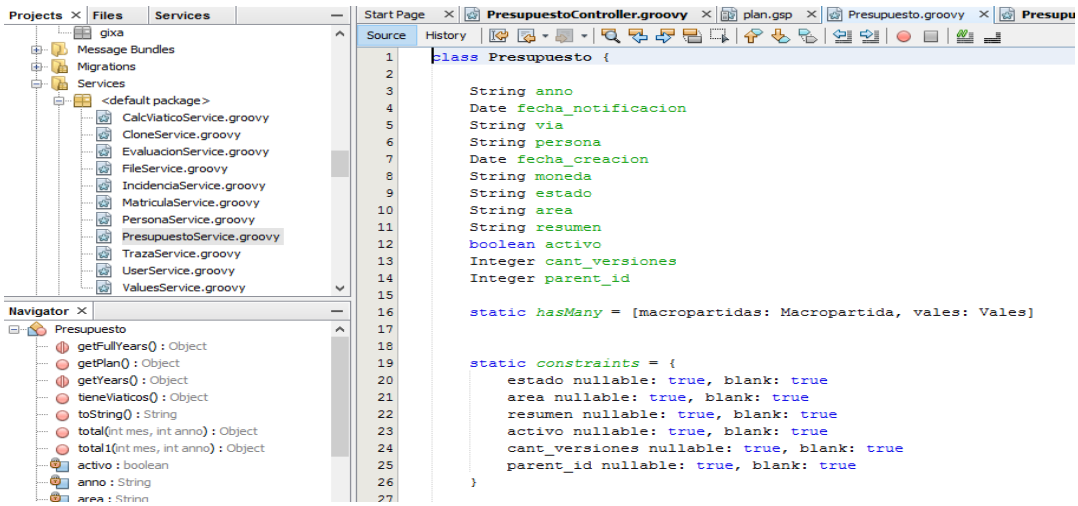


*Figura #3: Representación de la capa Modelo del patrón arquitectónico MVC*

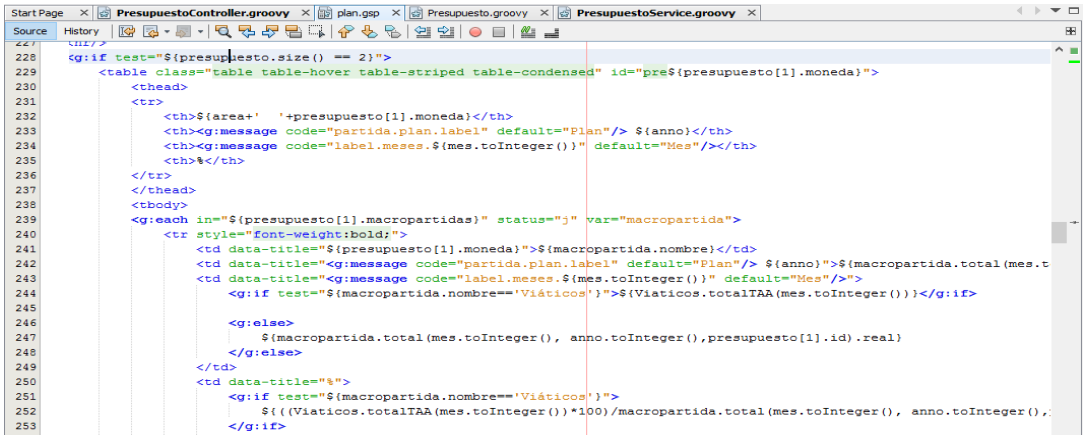


*Figura #4: Representación de la capa Vista del patrón arquitectónico MVC*

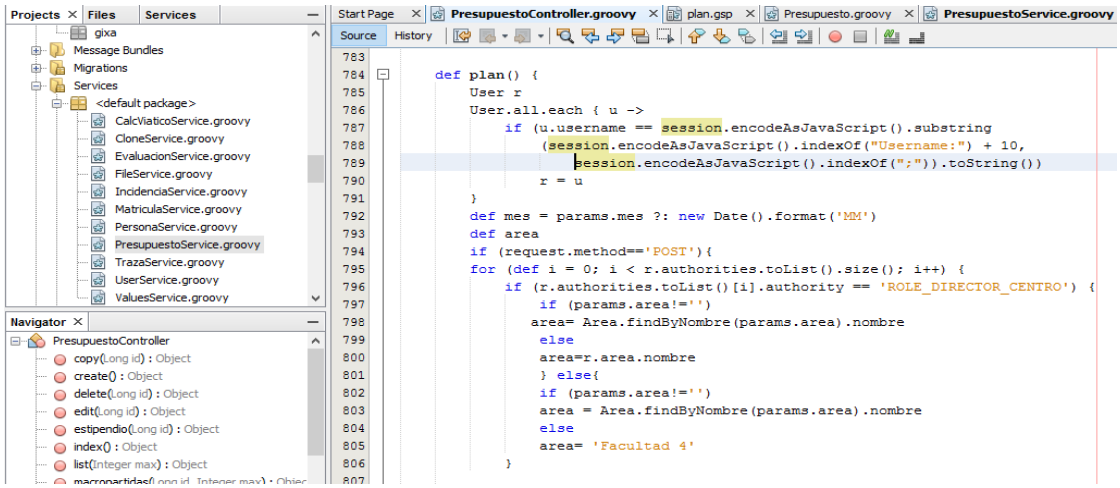


*Figura #5: Representación de la capa Controlador del patrón arquitectónico MVC*

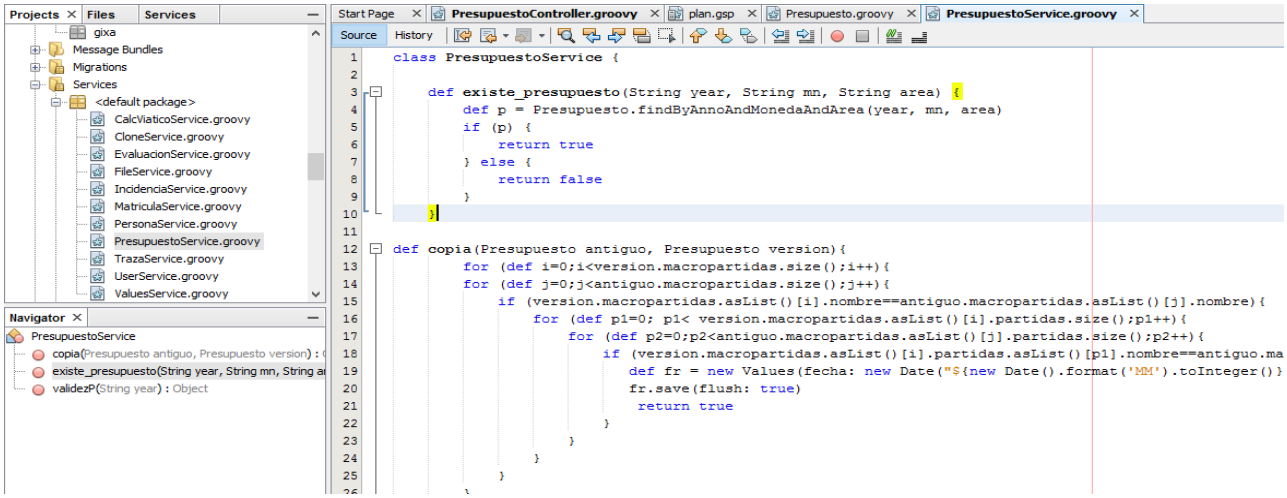


*Figura #6: Representación de la capa Servicio del patrón arquitectónico MVC*

## 2.10 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (56).

- **Inversión de Control (IoC):** la Inversión de Control (IoC, del inglés *Inversion of Control*) es un patrón utilizado en Grails, según el cual las dependencias de un componente no deben gestionarse desde el propio componente para que éste solo contenga la lógica necesaria para hacer su trabajo. El objetivo de esta técnica es mantener los componentes lo más sencillos posible, incluyendo únicamente el código que tenga relación con la lógica del negocio y deja fuera todo el código de instalación; así la aplicación será más fácil de mantener y comprender.

### 2.10.1 Patrones GRASP

Los patrones GRASP (del inglés *General Responsibility Assignment Software Patterns*) o Patrones Generales de Software para Asignación de Responsabilidades se consideran una serie de buenas prácticas de aplicación recomendable en el diseño de software (57). Los patrones básicos utilizados para aspectos fundamentales del diseño del sistema son:

- **Experto:** permite la asignación de responsabilidades al experto en información, adquiriendo de este modo un diseño con mayor cohesión y menor acoplamiento de la información (57). Es utilizado en la clase PresupuestoController, que garantiza que la creación de un presupuesto cuente con las funcionalidades que permiten realizar esta operación, mediante la información obtenida de las

clases del modelo, con quienes se encuentra relacionada. También se evidencia en las clases ValesController, ViáticoController, entre otras.

- **Creador:** indica quién es el responsable de la creación o instanciación de nuevos objetos o clases del proyecto (57). Es utilizado en las clases controladoras, donde se encuentran las acciones definidas para el sistema, por ejemplo, en la clase PresupuestoController en el método create(), cuando se crea el objeto de tipo presupuesto.
- **Controlador:** es el intermediario entre una determinada interfaz del producto y el algoritmo que implementa. Permite a los desarrolladores la reutilización del código y a la vez tener un mayor control (57). Este se evidencia en las clases que forman la capa Controlador del patrón arquitectónico MVC: PresupuestoController, ValesController, ViáticoController, MatriculaController, entre otras.

### 2.10.2 Patrones GoF

Los patrones GOF(del inglés *Gang of Four*) o Banda de los Cuatro se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro “DesignPatterns” (Patrones de Diseño) de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides (58). El patrón empleado en el diseño del proyecto fue:

**Decorator:** patrón estructural que añade funcionalidades a un objeto individual de forma dinámica y transparente, sin afectar a otros objetos. Este patrón es utilizado en la capa Vista del MVC y se emplea para incorporar modificaciones a partir de cada vista particular. Se definen en un único archivo independiente de las vistas GSP con lo que se simplifica enormemente la gestión del “look&feel” de la aplicación (59). Ejemplo de este patrón lo constituye la clase Main que es padre de todas las vistas de la aplicación. A continuación, se evidencia el patrón *Decorator en Grails*.

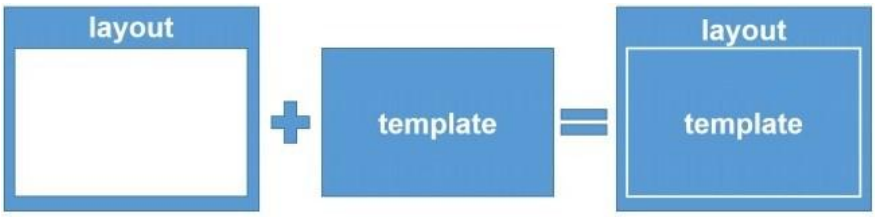


*Figura #7: Patrón Decorator en Grails*

## 2.11 Diagrama Entidad Relación (DER)

El Modelo Entidad Relación es un modelo de datos que consiste en un conjunto de objetos básicos llamados entidades y relaciones entre estos objetos, implementándose en forma gráfica a través del Diagrama Entidad Relación (60).

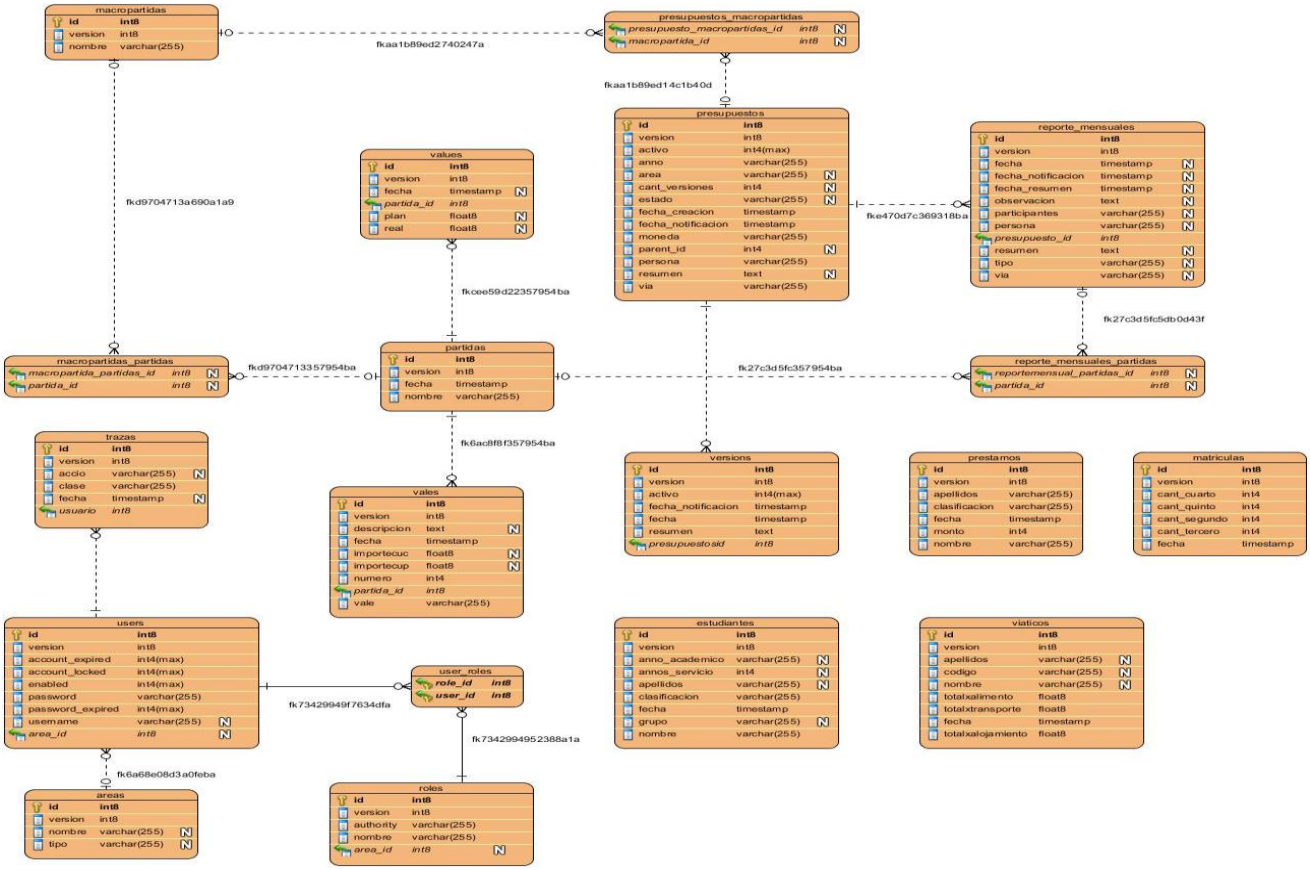


*Figura #8: Diagrama Entidad Relación*

*Tabla #14: Descripción de las tablas fundamentales que conforman la base de datos*

| Tablas | Descripción |
|---|---|
| tb_estudiantes | Esta tabla es la encargada de almacenar los datos de los estudiantes orden 18 y alumnos ayudantes. |
| tb_macropartidas | En esta tabla se guardan los datos de las macro - partidas. |
| tb_partidas | En esta tabla se recogen los datos de las partidas. |
| tb_matriculas | Esta tabla almacena por cada año la cantidad de estudiantes que matriculan. |

| | |
|---|---|
| tb_presupuestos | Se encarga de almacenar los datos de los presupuestos, como la fecha, área donde se ejecuta y tipo de moneda. |
| tb_reportes | Se utiliza para guardar los datos de cada reporte de ejecución mensual. |
| tb_roles | Almacena el nombre de los roles que crea el sistema, como Vicedecano Administrativo, Decano y Director del Centro. |
| tb_viaticos | En ella se almacenan los datos de los viáticos. |
| tb_users | Esta tabla almacena los datos de los usuarios que el Vicedecano Administrativo permita conectar al sistema. |
| tb_vales | Recoge los datos de los vales mensuales asociados a una partida. |
| tb_values | Guarda los datos de ejecución real y el plan de cada partida. |
| tb_versions | Contiene los datos de las versiones que se realicen al presupuesto. |

## 2.12 Interfaces del sistema

La siguiente figura muestra la interfaz de usuario Ejecuciones donde se muestra la ejecución del plan pudiendo consultar por año, mes o área según lo desee el usuario. Muestra además las macropartidas y las partidas asociadas, el plan, la ejecución real de cada partida, además del porciento de ejecución de cada una con respecto al plan.

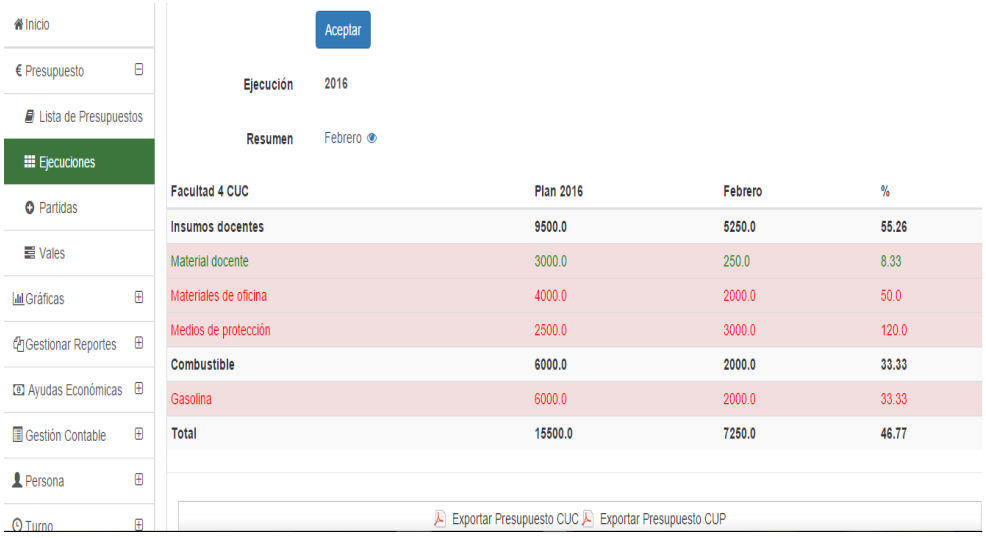


*Figura #9: Interfaz Ejecuciones del Presupuesto*

La siguiente interfaz permite gestionar los usuarios que acceden al sistema, permitiendo asignarle el rol que desempeñarán.


*Figura #10: Interfaz Gestionar usuarios*

La siguiente interfaz muestra el listado de presupuesto registrado, y las diferentes acciones que se pueden realizar con ellos.

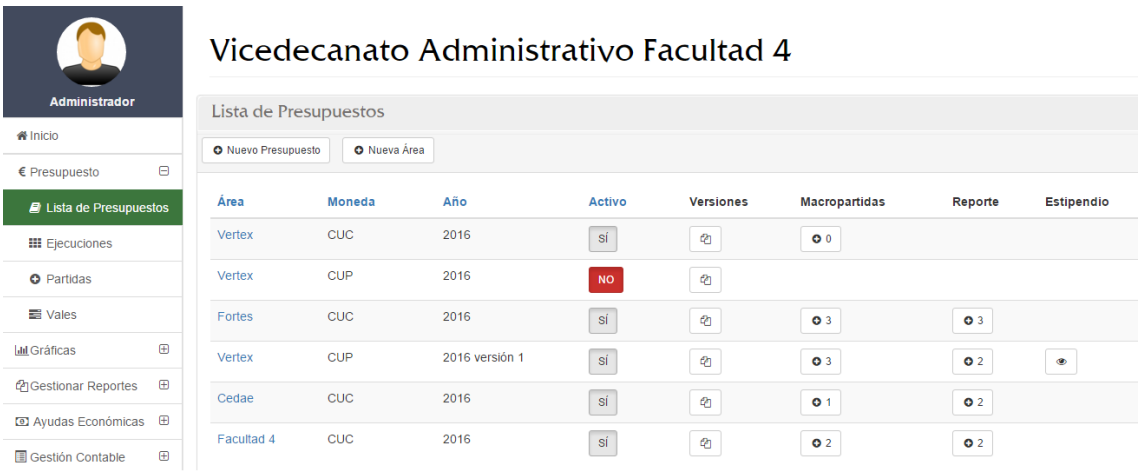


*Figura #11: Interfaz Lista de Presupuesto*

La siguiente figura muestra la interfaz principal, donde se muestra el usuario conectado y las acciones que el mismo pueda realizar en sistema.

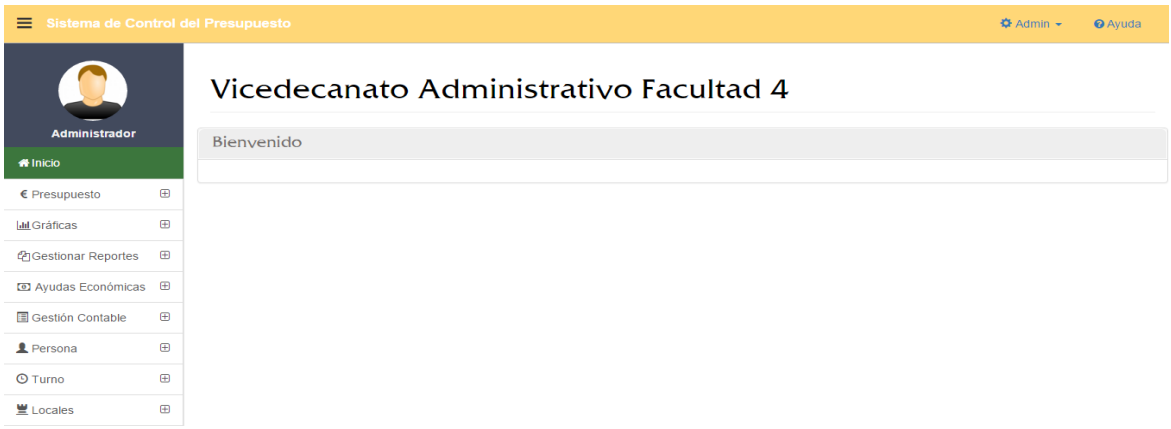


*Figura #12: Interfaz Inicio*

La siguiente interfaz muestra la gráfica de ejecución anual por partidas.

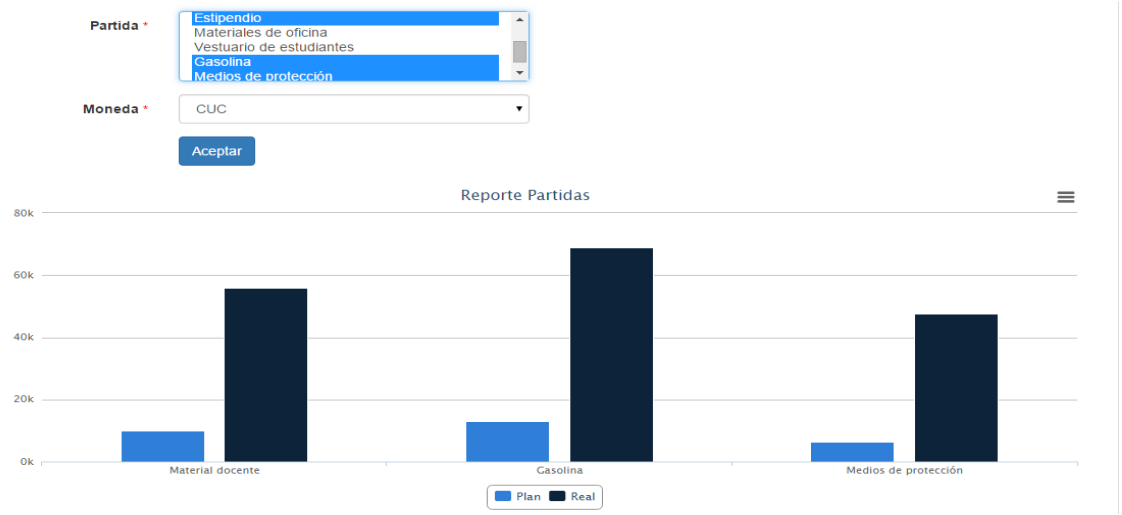


*Figura #13: Interfaz Gráfica de ejecución anual*

La figura muestra la interfaz de inicio de sesión, en la cual el usuario se autentica con su usuario y contraseña del dominio UCI.

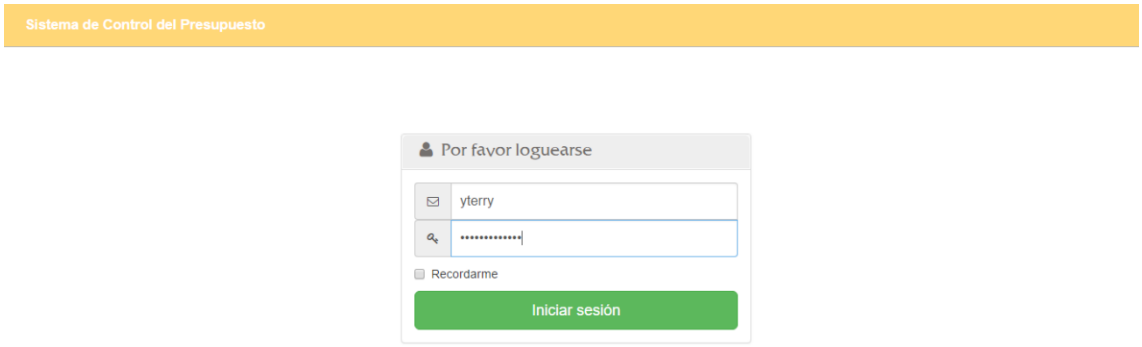


*Figura #14: Interfaz Autenticación*

## Conclusiones parciales

En este capítulo se describió la propuesta del sistema desarrollado, mediante los artefactos generados en las fases de Exploración y Planificación que propone la metodología utilizada. Se confeccionó el modelo conceptual para una mejor comprensión de la lógica del negocio y se obtuvo el diagrama de diseño de la base de datos. Se identificaron las características que el sistema debe cumplir, así como la descripción de cada una de las HU y la estimación del esfuerzo dedicado a la realización de cada una de ellas según el orden establecido de acuerdo a las necesidades del cliente. Con la obtención del plan de entregas se logró delimitar el ciclo de desarrollo del sistema. Además, se plantearon los patrones de diseño que se utilizaron en la elaboración de la solución, se definió la arquitectura de software y se elaboraron las tarjetas CRC permitiendo identificar y organizar las clases orientadas a objetos.

# CAPÍTULO 3

# IMPLEMENTACIÓN Y PRUEBA

## 3.1 Introducción

El presente capítulo se basa en las fases de implementación y prueba de la metodología seleccionada. En el mismo se especifican las tareas de ingeniería generadas a partir del desarrollo de las HU y se validan cada una de las funcionalidades mediante las pruebas de software, para garantizar el cumplimiento de los requerimientos.

## 3.2 Implementación

Durante esta fase se desarrollan cada una de las iteraciones definidas en la fase de planificación. Por cada iteración se llevan a cabo un conjunto de tareas de ingeniería, derivadas de las HU elaboradas anteriormente (61) (62) (63).

### 3.2.1 Programación en pareja

Toda la producción del código debe realizarse con trabajo en parejas de programadores. Así se incrementa la calidad del sistema desarrollado sin afectar el tiempo de entrega (64). Las principales ventajas de introducir este estilo de programación son (61):

- Inspecciones continuas de código, por lo que la tasa de errores del producto final es más baja.
- Se obtiene un código de menos tamaño, por la continua discusión de ideas de los programadores.
- Los problemas de programación se resuelven más rápido.
- Varias personas entienden las diferentes partes del sistema.
- Transparencia de conocimientos de programación entre los miembros del equipo.

Los programadores parten de la idea que quieren desarrollar, mientras un programador piensa en la táctica con la que se va a abordar el problema, el otro se encarga de las estrategias que permiten llevar dicha táctica a su máximo exponente.

### 3.2.2 Tareas de ingeniería

Las tareas de ingeniería son actividades que los programadores conocen que el sistema debe hacer. Deben ser estimables, su tiempo de implementación debe ser corto, aproximadamente entre uno y tres días, y su objetivo es resolver las historias de usuario. Una HU puede tener una o varias tareas de ingeniería, en dependencia de la funcionalidad a desarrollar. Pueden existir también tareas e ingeniería técnicas, que son aquellas que aunque no derivan directamente de una Historia de usuario, es necesario su consideración para que el sistema funcione (65).

Se implementaron un total de 21 tareas de ingeniería, en este epígrafe se muestran las principales que se relacionan con las HU descritas en el Capítulo 2, las restantes se encuentran en el Anexo 3 de la investigación.

*Tabla #15: Tarea de ingeniería Gestionar plan del presupuesto anual*

| **Tarea** | |
|---|---|
| **Número:** 1 | **Número de HU:** 1 |
| **Nombre:** Gestionar plan del presupuesto anual | |
| **Tipo de tarea:** Desarrollo | **Estimación:** 4 días |
| **Fecha inicio:** 8 de Octubre del 2015 | **Fecha fin:** 11 de Octubre del 2015 |
| **Programador responsable:** Alexis Borges Mora | |
| **Descripción:** implementar la funcionalidad “Gestionar plan del presupuesto anual” de la propuesta solución. | |

*Tabla #16: Tarea de ingeniería Gestionar partida*

| **Tarea** | |
|---|---|
| **Número:** 3 | **Número de HU:** 3 |
| **Nombre:** Gestionar partida | |
| **Tipo de tarea:** Desarrollo | **Estimación:** 6 días |
| **Fecha inicio:** 16 de Octubre del 2015 | **Fecha fin:** 21 de Octubre del 2015 |

| |
|---|
| **Programador responsable:** Alexis Borges Mora |
| **Descripción:** implementar la funcionalidad “Gestionar partida” de la propuesta solución. |

*Tabla #17: Tarea de ingeniería Gestionar datos de viático*

| **Tarea** | |
|---|---|
| **Número:** 8 | **Número de HU:** 8 |
| **Nombre:** Gestionar datos de viático | |
| **Tipo de tarea:** Desarrollo | **Estimación:** 7 días |
| **Fecha inicio:** 19 de Noviembre del 2015 | **Fecha fin:** 25 de Noviembre del 2015 |
| **Programador responsable:** Victor Abel Reyes | |
| **Descripción:** implementar la funcionalidad “Gestionar datos de viático” de la propuesta solución. | |

*Tabla #18: Tarea de ingeniería Gestionar registro anual de vales*

| **Tarea** | |
|---|---|
| **Número:** 15 | **Número de HU:** 15 |
| **Nombre:** Gestionar registro anual de vales | |
| **Tipo de tarea:** Desarrollo | **Estimación:** 5 días |
| **Fecha inicio:** 21 de Enero del 2016 | **Fecha fin:** 25 de Enero del 2016 |
| **Programador responsable:** Victor Abel Reyes | |
| **Descripción:** implementar la funcionalidad “Gestionar registro anual de vales” de la propuesta solución. | |

## 3.3 Diagrama de despliegue

La metodología XP no requiere diagramas de despliegue, pero es de gran ayuda para un mejor entendimiento de la distribución física de la propuesta de solución, por tal motivo se diseñó un diagrama de despliegue.

El diagrama de despliegue muestra la arquitectura del sistema desde el punto de vista de la distribución de los artefactos del software en los destinos del despliegue. Los artefactos representan los elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo (66). Los diagramas de despliegue simbolizan los nodos, que son objetos físicos con recursos computacionales y sus relaciones, que se etiquetan con un estereotipo que identifican el protocolo de comunicación o la red utilizada.

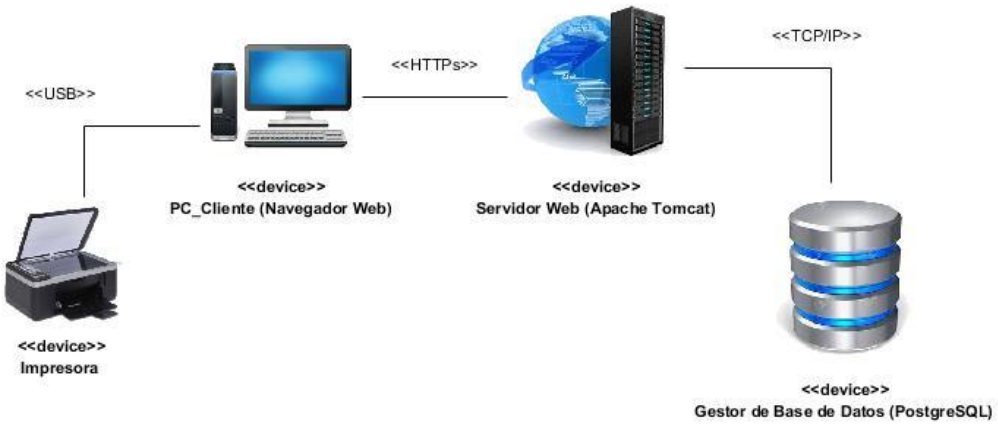


*Figura #15: Diagrama de despliegue*

## 3.4 Tratamiento de errores

En el sistema se deben analizar las diferentes situaciones que puedan ocurrir y los errores que estas puedan ocasionar. Las excepciones son identificadas en el proceso de implementación, aunque se debe definir con antelación un mecanismo efectivo para darle tratamiento a cada una de ellas.

## 3.5 Pruebas

Uno de los pilares de la metodología XP es el uso de pruebas para comprobar el funcionamiento de los códigos que se vayan implementando. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. Las pruebas se dividen en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores y pruebas de aceptación destinadas a evaluar si, al culminar una iteración, se consiguió la funcionalidad requerida diseñada por el usuario final (67).

Las pruebas del sistema son fundamentales en XP, en la que se ha desarrollado un enfoque que reduce la probabilidad de producir nuevos incrementos del sistema que introduzcan errores en el software existente (68). A continuación, se muestran los métodos, tipos y niveles de pruebas realizados al sistema, junto al ambiente en que se desarrollaron.

**Métodos de pruebas**

*Tabla #19: Métodos de pruebas*

| Pruebas de caja blanca | Pruebas de caja negra |
|---|---|
| Estas pruebas comprueban los caminos lógicos del software, verificando todos los componentes internos, para determinar si el estado real coincide con el esperado (69) (70). | Estas pruebas se desarrollan sobre la interfaz del software comprobando todas las funcionalidades sin tener en cuenta la estructura interna del sistema (69) (70). |

**Tipos de pruebas**

*Tabla #20: Tipos de pruebas*

| Pruebas unitarias | Pruebas de funcionalidad |
|---|---|
| Estas pruebas se encargan de probar cada uno de los métodos y verifica si dado unos parámetros de entrada, la salida es la esperada (69) (70). | Estas pruebas se realizan con el objetivo de verificar el cumplimiento de los requisitos funcionales (69) (70). |

- **Niveles de pruebas:** dentro de los niveles de prueba se encuentran las pruebas de aceptación, las mismas aseguran el comportamiento del sistema, son escritas por el cliente o por el usuario y especifican los aspectos a probar cuando una HU ha sido correctamente implementada, garantizando que los requerimientos han sido cumplidos y que el sistema es aceptable. Significan la satisfacción del cliente con el producto desarrollado (67) (69).

- **Ambiente de prueba:** las pruebas se ejecutaron en un ambiente compuesto por la laptop personal de uno de los desarrolladores y en una computadora personal.

*Tabla #21: Características de los ambientes de desarrollo*

| Cantidad de PC | Sistema Operativo | RAM | Procesador |
|---|---|---|---|
| 1 | **Windows** | 3 GB | Intel® Core™ i3-3110M CPU @ 2.4GHz |
| 2 | **Xubuntu** | 2 GB | Intel® Core™ i3-2130M CPU @ 3.4GHz |
| 3 | **Nova** | 1 GB | Intel® Core Duo  U2500 CPU @ 1.2GHz |

### 3.5.1 Resultados de las pruebas

Como resultado de las pruebas de aceptación se realizaron cinco iteraciones, una por cada entrega. Se detectaron un total de 21 no conformidades significativas y 19 no significativas. Al concluir la quinta iteración todas las no conformidades fueron resueltas, lo que garantiza la calidad de la propuesta de solución y la satisfacción del cliente. Los casos de pruebas se encuentran en el Anexo 4 de la investigación. A continuación se muestra una gráfica con los resultados:

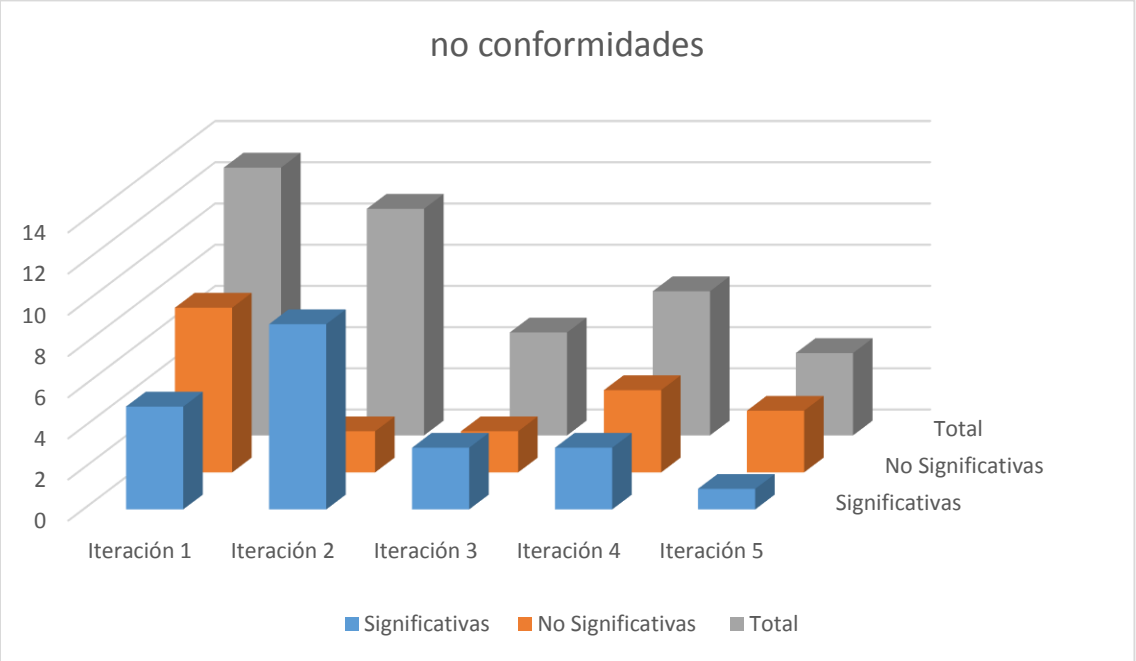


*Figura #16: Gráfica no conformidades por iteraciones*

Para aplicar las pruebas unitarias a la propuesta de solución se utilizó el marco de trabajo JUnit, el cual permite realizar este tipo de pruebas al código. A continuación, se muestran ejemplos de los resultados arrojados durante las pruebas unitarias:

- El siguiente método muestra que el campo usuario de la clase Traza, no puede ser nulo.

```
@TestMixin(GrailsUnitTestMixin)class TrazaTests {

    void test_Traza_Nullable() {

        def a = new Traza(url: 'presupuesto/traza', context: 'kjbk', accio: 'Crear', fecha: new Date(1991,10,11), clase: 'Presupuesto')

        assertFalse a.validate()
```

```
        assertEquals 'El campo usuario no puede ser nulo', 'usuario', a.errors['usuario']
    }
}
```


*Figura #17: Método para demostrar que la clase usuario no puede ser nulo*

- El siguiente método muestra que el campo usuario de la clase Traza, no puede ser en blanco.

```
@TestMixin(GrailsUnitTestMixin)class TrazaTests {
    void test_Traza_Blank() {
    def a = new Traza(url: 'presupuesto/traza',context: 'kjbk',accio: 'Crear', usuario: '',fecha: new Date(1991,10,11),clase: 'Presupuesto')
    assertFalse a.validate()
assertEquals 'El campo usuario no puede ser en blanco', 'usuario', a.errors['usuario']
    }
}
```

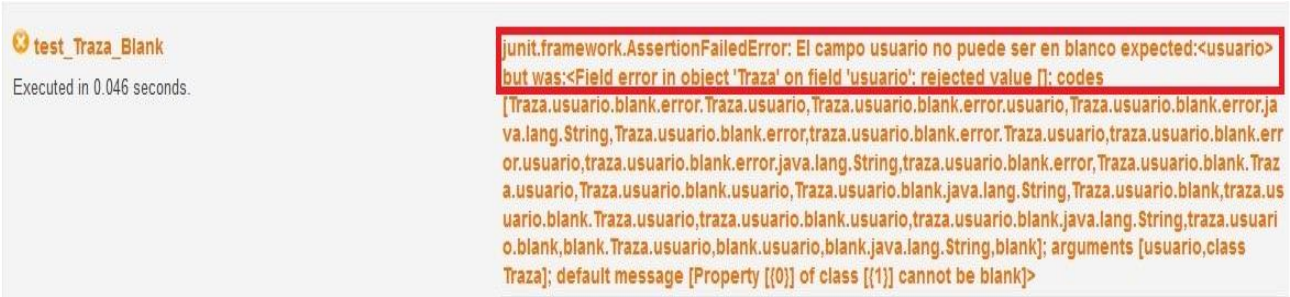


*Figura #18: Método para demostrar que la clase usuario no puede estar en blanco*

- El siguiente método muestra que se ha insertado correctamente una Traza.

```
@TestMixin(GrailsUnitTestMixin)class TrazaTests {

    void test_Traza_Crear() {

      def a = new Traza(url: 'presupuesto/traza',context: 'kjbk',accio: 'Crear',  usuario:
          'aborges',fecha: new Date(1991,10,11),clase: 'Presupuesto')

        assertTrue 'Se creó correctamente', a.validate()

    }
```


*Figura #19: Método para demostrar que se ha insertado correctamente una traza*

## Conclusiones parciales

En este capítulo se realizaron las tareas de ingeniería asociadas a cada HU. Se diseñó el diagrama de despliegue para un mejor entendimiento de la distribución física de la propuesta de solución. Además, se realizaron las pruebas unitarias y de aceptación para cada iteración del proceso de desarrollo del sistema, con el objetivo de garantizar un buen funcionamiento de la solución implementada y de esta manera cumplir con las necesidades del cliente.

# CONCLUSIONES

Con la elaboración del sistema para controlar los presupuestos de gastos administrativos de la Facultad 4 y el Centro FORTES se concluye:

- Los métodos científicos utilizados permitieron fundamentar los conceptos y características que sustentaron la investigación, logrando un mayor entendimiento por parte del equipo de desarrollo.
- El diseño realizado, así como el estudio sobre las tecnologías más adecuadas para el proceso de desarrollo, arrojó como resultado una aplicación que cumple con las funcionalidades definidas para dar solución al objetivo planteado.
- Se implementó un sistema para el control de los PGA de la Facultad 4 y el Centro FORTES, el cual aumentó la seguridad de la información.
- Las pruebas realizadas al sistema demostraron que las funcionalidades previstas en la propuesta de solución, garantizaron el cumplimiento de las expectativas del cliente.

# RECOMENDACIONES

A partir de la investigación realizada se sugieren las siguientes recomendaciones:

- Aplicar el software en el área del Vicedecanato Administrativo de la Facultad 4 y socializar su uso en el resto de las Facultades de la universidad.
- A las personas que brindaran soporte a la aplicación, mantener actualizado el mismo con las nuevas tecnologías y herramientas informáticas.

# BIBLIOGRAFÍA

1. **Real Academia Española. [En línea] 2001. [Citado el: 12 de 1 de 2016.] http://lema.rae.es/drae/.**

**2. Gedesco. [En línea] 3 de 9 de 2014. [Citado el: 12 de 1 de 2016.] http://www.gedesco.es/blog/para-que-sirve-un-plan-de-empresa/.**

**3. Kenilia y Villalón Madrazo, Mariela. ResearchGate. *La Planificaión y el Modelo Económico Cubano.* [En línea] 2008-2016. [Citado el: 12 de 1 de 2016.] https://www.researchgate.net/publication/267843837_LA_PLANIFICACION_Y_EL_MODELO_ECONOMICO_CUBANO.**

**4. Torres, Liudmila Rodríguez. Revistas ISMM. [En línea] [Citado el: 12 de Noviembre de 2015.] https://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=8&ved=0ahUKEwj6_ZP1h-zKAhVCeT4KHS2KBmUQFgg9MAc&url=http%3A%2F%2Frevista.ismm.edu.cu%2Findex.php%2Frevista_estudiantil%2Farticle%2Fdownload%2F628%2F314&usg=AFQjCNG_WXMMm1t1Rj5PYj6lKedKuvknZQ&.**

**5. Arias Morales, Augusto , Cruz Soto, Exequiel y Ruiz Escobar, Ana Amalia. Gestiopolis. [En línea] http://www.gestiopolis.com/planeacion-empresarial-dentro-del-proceso-administrativo/.**

**6. Muñiz, Luis. *Control Presupuestario.* Barcelona : Profit Editorial, 2009. 9788492956821.**

**7. Contabilidad. [En línea] [Citado el: 12 de 1 de 2016.] http://contabilidad.com.py/articulos_35_02-presupuesto-y-control-presupuestario-tipos-de-presupuesto.html..**

**8. AENOR. *Certificación ISO 27001 de Sistemas de gestión de la seguridad de la información.* [En línea] [Citado el: 7 de 6 de 2016.] http://www.aenor.es/aenor/certificacion/seguridad/seguridad_27001.asp.**

**9. DATYS. [En línea] DATYS, 2015. [Citado el: 13 de Enero de 2016.] http://www.datys.cu/spa/site/product/18.**

**10. Martinez, Liliana. Evaluando ERP. [En línea] 10 de Septiembre de 2013. [Citado el: 13 de Enero de 2016.] http://www.evaluandoerp.com/etes-primer-erp-cubano-registrado-en-evaluandoerp-com/.**

**11. Rodriguez, Danay Prestamo. Habana, 1 de Febrero de 2016.**

**12. Datazucar. [En línea] 2016. [Citado el: 13 de Enero de 2016.] http://www.datazucar.cu/?p=65.**

**13. Romero, Lorely Molla. Observatorio de la Economía Latinoamericana. [En línea] 2009. [Citado el: 13 de Enero de 2016.] http://www.eumed.net/cursecon/ecolat/cu/2009/lmr2.htm. 1696-8352.**

**14. Porteiro, Marisel Sosa y Cobo Morales, Pedro H. BetSime. [En línea] Casa Consultora DISAIC. [Citado el: 13 de Enero de 2016.] http://www.betsime.disaic.cu/secciones/eco_enemar_07.htm.**

**15. Salazar Laffita, Elizabeth y Guerra Padrón, Fabián. Biblioteca de la Universidad de las Ciencias Informáticas. [En línea] 18 de Enero de 2015. [Citado el: 13 de Enero de 2016.] http://repositorio_institucional.uci.cu/jspui/bitstream/ident/8558/1/TD_07084_13.pdf.**

**16. ASOSEM. [En línea] 2012. [Citado el: 15 de Enero de 2016.] http://www.asosem.org/docs/escaparate/openbravo.pdf.**

**17. OPENLIBRA. [En línea] 2011. [Citado el: 6 de Febrero de 2016.] http://www.etnassoft.com/biblioteca/manual-de-usuario-openbravo/.**

**18. OpenERP Spain. [En línea] Domatix. [Citado el: 6 de Febrero de 2016.] http://openerpspain.com/openerp/por-que-elegir-openerp/.**

**19. SYPSOFT. [En línea] 2011. http://www.sypsoft.net/files/SAP.pdf.**

**20. Minguet Melián, Jesús M y Hernández Ballesteros, Juan Francisco. *LA CALIDAD DEL SOFTWARE Y SU MEDIDA.* s.l. : Centro de estudios Ramón Areces,S.A, 2003. 84-8004-611-2.**

**21. Polo, Elena Gordillo. inventtatte. [En línea] 21 de 10 de 2014. [Citado el: 12 de 1 de 2016.] http://inventtatte.com/metodologia-tradicional-vs-agil/.**

**22. Beck, Kent. *Extreme Programming Explained.* 1999. 0201616416.**

**23. Heredia Ruiz, Javier, Álvarez Almanza, Lilián y Linares Pons, Naryana. Serie Científica Universidad de las Ciencias Informáticas. [En línea] 19 de Octubre de 2011. [Citado el: 23 de Diciembre de 2015.] http://publicaciones.uci.cu/index.php/SC/article/view/484/469.**

**24. Bolivariana, Universidad Unión. Universidad Unión Bolivariana. Ingenieria de Software. [En línea] [Citado el: 12 de 1 de 2016.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html..**

**25. Ortega, Raúl Jiménez. Introducción a SCRUM. [En línea] [Citado el: 12 de 1 de 2016.] http://osl.ugr.es/talleres/SCRUM/Presentacion%20SCRUM.html#slide1.**

**26. J.Stapleton. *Dsdm Dynamic Systems Development Method: The Method in Practice.* s.l. : Addison-Wesley, 1997.**

**27. Villamarin , Carlos Zambrano y sosa rosales, robert alexander . utmfci. [En línea] [Citado el: 12 de 1 de 2016.] https://sites.google.com/site/utmfci/home/ventajas-y-desventajas.**

**28. Metodología FDD – Feature Driven Development / Desarrollo Basado en Funciones . [En línea] 12 de Junio de 2012. [Citado el: 12 de 1 de 2016.] http://metodologiafdd.blogspot.com/.**

**29. Flores, Ervin. *Metodologías ágiles AUP.* 2014.**

**30. Larman, Craig. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Segunda Edición.* 2003.**

**31. IngenieroSoftware. Prácticas y métodos para mejorar el desarrollo de Proyectos de Software. [En línea] 2003. [Citado el: 6 de Junio de 2015.] http://www.ingenierosoftware.com/analisisydiseno/uml.php..**

**32. Terry, B y Logee, D. *Terminology for Software Engineering and Computer-aided Software Engineering.* s.l. : Software Engineering Notes, 1990.**

**33. Visual Paradigm para UML. [En línea] [Citado el: 12 de Septiembre de 2015.] http://www.software.com.ar/visual-paradigm-para-uml.html..**

**34. Smith, Glen y Ledbrook, Peter. *Grails in Action.* s.l. : Manning Publications Co, 2009.**

**35. Konig, Dierk y Glover, Andrew. *Groovy in Action.* s.l. : Manning Publications Co, 2007.**

**36. Grails. [En línea] [Citado el: 11 de Junio de 2015.] https://grails.org/2.1.1+Release+Notes.**

**37. Grails.com. [En línea] [Citado el: 11 de Junio de 2015.] http://www.grails.com/.**

**38. Spurlock, Jake. *Bootstrap.Primera edición.* 2013.**

**39. Bootstrap-Línea de código. [En línea] 2013. [Citado el: 20 de Noviembre de 2015.] http://lineadecodigo.com/categoria/bootstrap/.**

**40. Definición.de. [En línea] 2015. [Citado el: 20 de Noviembre de 2015.] http://definicion.de/lenguaje-de-programacion/.**

**41. Brito, Nacho. Manual de Grails. [En línea] 2009. [Citado el: 14 de Septiembre de 2015.] http://www.manual-de-grails.es.**

**42. Gosling, James, y otros. *The Java Language Specification.* California : Oracle America, Inc, 2013. JSR-000901.**

**43. Menéndez-Barzanallana, Rafael Asensio. Desarrollo de aplicaciones web.**

**44. Navarrete, Toni. Lenguaje de programación JavaScript. 2006.**

**45. World Wide Web Consortium. [En línea] Diciembre de 2013. [Citado el: 12 de Octubre de 2015.] http://www.w3.org/html/.**

**46. Cantón, Alejandro Castillo. Manual de HTML5 en español.**

**47. Gauchat, Juan Diego. *El gran libro de HTML5, CSS3 y Javascript. Primera Edición.* s.l. : marcombo, 2012.**

**48. Cascading Style Sheets (CSS) Snapshot 2010. [En línea] 2011. http://www.w3.org/TR/CSS/.**

**49. [En línea] http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollointegrado-ide/.**

**50. PostgreSQL. [En línea] http://postgresql-dbms.blogspot.com/p/limitaciones-puntosde-recuperacion.html.**

**51. Beck, Kent y Cyntia Andres. *Extreme Programing Expained: Embrace Change.* s.l. : Addison Wesley Professional, 2004. ISBM 0321278658.**

**52. Joskowicz, Jose. *Reglas y prácticas en Extreme Programming.* 2008.**

**53. *Metodologia XP.* Calabria, Luis y Piriz, Pablo. Uruguay : Catedra de Ingenieria de software, 2003.**

**54. Beck, Kent. A Laboratory For Teaching. [En línea] [Citado el: 18 de enero de 2016.] http://c2.com/doc/oopsla89/paper.html.**

**55. Potencier, Fabien. *Practical Symfony 1.3 & 1.4 for Doctrine.* s.l. : Sencio Sa, 2009. ISBM 9782918390169.**

**56. [En línea] [Citado el: 9 de febrero de 2016.] https://msdn.microsoft.com/es-es/library/bb972240.aspx.**

**57. *Fundamentos de Ingenieria de software.* Visconti, Marcello y Astudillo, Hernan. Universidad Técnica Federico Santa María : s.n.**

**58. GoF – Patrones de diseño (XII): Facade. [En línea] 29 de enero de 2013. [Citado el: 23 de enero de 2016.] https://infow.wordpress.com/category/patrones-de-disenogof/.**

**59. Larman, Craig. *UML y patrones: introducción al análisis y diseño orientado a objetos.* 1999.**

**60. Storti, Guillermo, Rios, Gladys y Compodonico, Gabriel. Tecnologia de la informacion y la comunicacion. *Bases de dato modelo Entidad Relacion.* [En línea] 2007. [Citado el: 26 de febrero de 2016.] http://www.belgrano.esc.edu.ar/matestudio/carpeta_de_access_introduccion.pdf.**

**61. Joskowicz, José. *Reglas y Prácttticas en Extreme Programing.* 2008.**

**62. [En línea] [Citado el: 12 de Enero de 2015.] http://xprogramming.com/xpmag/whatisxp..**

**63. Patricio, Letelier y Penadés, María del Carmen. 2013.**

**64. *Canós, José H., Penadés, María del Carmen y Letelier, Patricio. 2013.***

***65. Wallace, Doug y Ragget, Isobel. Extreme Programming for Web Projects. s.l. : Addisson-Wesley Professional, 2003. ISBN 0201794276.***

***66. UML. [En línea] http://umldiagramadespliegue.blogspot.com/.***

***67. Ordóñez, Meylin Pérez y Rodríguez Corbea, Maite. La Metodología XP aplicable al desarrollo del software educativo en Cuba. 2007.***

***68. Ian, Sommerville. Ingeniería del software. [trad.] María Isabel Galipienso Alfonso , y otros. Séptima edición.***

***69. Pressman, Roger S. Ingeniería de software. Un enfoque práctico. Sexta Edición. 2005.***

***70. —. Ingeniería de software. Un enfoque prático. Quinta Edición. 2001.***

***71. Real Academia Española. [En línea] 2015. http://lema.rae.es/drae/.***

***72. definición.de. [En línea] 2015. http://www.definicion.org/control.***

***73. DefiniciónABC. [En línea] 2015. [Citado el: 19 de Noviembre de 2015.] http://www.definicionabc.com/tecnologia/erp.php.***

***74. DefiniciónABC. [En línea] 2015. [Citado el: 19 de Noviembre de 2015.] http://www.definicionabc.com/general/planificacion.php.***

***75. M. Judd, Cristopher, Faisal Nusairat, Joseph y Singler, James. Beginning Groovy and Grails From Novice to Professional. s.l. : APRESS, 2008. 978-1-4302-1045-0.***

***76. Expansión. [En línea] Unidad Editorial Información Económica S.L., 2015. [Citado el: 24 de Noviembre de 2015.] http://www.expansion.com/diccionario-economico/control-presupuestario.html.***

***77. Universidad Unión Bolivariana. Ingeniería de Software. [En línea] [Citado el: 12 de 1 de 2016.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.***

***78. Gedesco. [En línea] 12 de Julio de 2013. http://www.gedesco.es/blog/la-planificacion-y-el-control-presupuestario-como-herramienta-de-gestion-empresarial/.***

***79. Badilla, Ricardo Montaño. Gestiopolis. [En línea] 10 de Febrero de 2010. http://www.gestiopolis.com/sistema-erp-definicion-funcionamiento-ventajas-desventajas/.***

***80. Muñiz, Luis. SisConsGes & Estrategia. [En línea] [Citado el: 12 de Noviembre de 2015.] http://www.sistemacontrolgestion.com/Portals/1/Control_Presupuestario_LMuniz.pdf.***

***81. domatix. [En línea] Domatix. [Citado el: 6 de Febrero de 2016.] http://www.domatix.com/odoo-openerp/.***

***82. [En línea] http://www.scielo.cl.***

***83. UML y Patrones.***

***84. [En línea] 2008. https://infow.wordpress.com/category/patrones-de-disenogof/.***

***85. [En línea] http://programacionextrema.tripod.com/fases.htm..***

***86. [En línea] http//:www.fing.edu.uy..***

***87. [En línea] http://c2.com/doc/oopsla89/paper.html..***

***88. [En línea] http://www.lsi.us.es/docencia..***

***89. Grails.com. [En línea] http://www.grails.com/.***

***90. Calahorro, Nacho Brito. Manual de desarrollo web con Grails.***

***91. [En línea] https://msdn.microsoft.com/es-es/library/bb972240.aspx..***

***92. Grails. [En línea] https://grails.org/2.1.1+Release+Notes.***

***93. Visconti, Marcello y Astudillo, Hernán. Fundamentos de Ingeniería de Software.***

***94. Calahorro, Nacho Brito. Manual de desarrollo web con Grails.***

***95. [En línea] http://www.belgrano.esc.edu.ar/matestudio/carpeta_de_access_introduccion.pdf..***

***96. Beck, Kend y Andres, Cyntia . Extreme Programing Explained;.***

***97. [En línea] 2015. http://insitutotec.blogspot.com/2013/03/21-tareas-de-la-ingenieria-de-requisitos.html.***

***98. Real Academia Española. [En línea] [Citado el: 7 de 6 de 2016.] http://www.rae.es/info/accesibilidad.***

***99. Diccionario de Infrmática y Tecnología. [En línea] [Citado el: 7 de 6 de 2016.] http://www.alegsa.com.ar/Dic/accesibilidad%20web.php.***