Universidad de las Ciencias Informáticas

Título:

Análisis y Diseño de Plan de Trabajo y Emulación de la FEU en la Universidad de las Ciencias Informáticas.

Trabajo de Diploma para optar por el título de Ingeniería en Ciencias Informáticas

Autor: Ariel Ramirez Alvarez

Tutores: Yoenmy González Almaguer

Co-tutor: Osay González Fuentes

2007

DECLARACIÓN DE AUTORÍA

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

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

Ariel Ramirez Alvares

_____

Yoenmy Gonzalez Almaguer

_____

AGRADECIMIENTOS

Le agradezco el apoyo a mis compañeros José Enrique e Ygraine, quienes han sido como otro par de Tutores.

Le agradezco al resto del equipo de Kainos, especialmente Jonathan González, Arian Abel Linares, Yoan Sánchez y Reldy Pérez.

Le agradezco a mi familia que me ha apoyado, a mis amigos y a los educadores de la UCI que de una forma u otra han contribuido a mi formación profesional.

DEDICATORIA

Le dedico este trabajo a mi familia que me ha apoyado siempre,

Le dedico este trabajo a la Revolución que me ha dado la posibilidad de cursar esta carrera,

Y les dedico este trabajo a todos los educadores de la UCI.

RESUMEN

La FEU es una organización estudiantil en constantes cambios y hoy camina con pasos de gigantes y se monta en el tren de las nuevas Tecnologías de las Informática y las Comunicaciones buscando solucionar sus problemas de Organización con estas herramientas.

En este trabajo de diploma se analiza el funcionamiento de la FEU, principalmente en el Plan de Trabajo y la Emulación. El Plan de Trabajo consiste en la organización de las actividades que se propone la FEU en un curso, mientras que la emulación consiste en la forma competitiva donde se promueve el desarrollo de las facultades. Actualmente estas funciones se llevan de una forma manual lo cual conlleva a algunos problemas de tipo organizativo.

Nuestro interés es realizar el Análisis y el Diseño de este módulo permitiendo posteriormente su desarrollo e inserción dentro del proyecto Kainos, el cual es el responsable de la informatización de los procesos de la FEU.

Índice

Introducción

La Federación Estudiantil Universitaria (FEU), es una organización estudiantil que representa los intereses y hace valer los derechos del estudiantado cubano. Entre sus objetivos actuales está agrupar a los universitarios para continuar la obra de la Revolución cubana; promueve el perfeccionamiento del nivel y el rigor docentes; y hace de la Extensión Universitaria la vía más importante para llevar el deporte, la cultura y el pensamiento intelectual a la sociedad.

La sociedad cubana, dentro del marco socialista utiliza la emulación como método de competencia capaz de promover el desarrollo de las fuerzas productivas. Dentro de la FEU también se utiliza la emulación como una forma competitiva de mejorar los resultados dentro de diferentes esferas como son la docencia, la cultura, y el deporte entre otros.

En esta nueva era de las Tecnologías de las Informáticas y las Comunicaciones, la FEU al ser una fuerza joven, se mantiene al día. Se ha planteado mejorar su trabajo si lo logra llevar a un entrono informático desde el cual lograrían gestionar sus funciones y los diferentes resultados estarían disponibles para todos los estudiantes a nivel nacional.

Estos esfuerzos se están llevando a cabo dentro del proyecto de la UCI, que se llamó Kainos, una palabra griega que significa unión, por ser el nombre adecuado para los objetivos finales de la FEU. Actualmente se están desarrollando varios módulos los cuales se integrarán posteriormente para brindar un producto final.

Situación Problémica.

La FEU de la Universidad de las Ciencias Informáticas, tiene dificultad con la planificación de las diferentes actividades realizadas por el centro, debido a la mayor diversidad de años docentes y el surgimiento de nuevos eventos; esto permite que no se aprovechen bien las diferentes fechas y muchos de los eventos planificados no se cumplen con la calidad necesaria. También existen problemas con el

manejo de la emulación, lo que permite que no exista el ambiente de competitividad propicio para el mejoramiento de los resultados en diferentes esferas.

El Problema.

¿Qué es necesario conocer para averiguar las causas de que el funcionamiento de los procesos de Plan de Trabajo y Emulación de la FEU sean ineficientes en el marco de la UCI?

Preguntas científicas.

¿Cómo la FEU de la Universidad de las Ciencias Informáticas planifica las diferentes actividades?

¿Cómo la FEU de la Universidad de las Ciencias Informáticas desarrolla su emulación?

Actualidad y necesidad del trabajo.

Se hace urgente mejorar el trabajo de la FEU, tanto dentro como fuera de la UCI. La función del Plan de Trabajo, que consiste en planificar todas las actividades de la organización dentro de un curso escolar, puede perderse si no se controla sistemáticamente y de una forma clara. De la misma forma, el propósito de competencia capaz de impulsar el desarrollo de la emulación, puede verse nublado por la sombra de la incertidumbre a la hora de mostrar los resultados, debido a la falta de transparencia en los mismos.

Antecedentes

Sistema de Gestión Nacional de la FEU.

Aportes prácticos esperados del trabajo.

Con este trabajo se espera desarrollar una herramienta informatizada que permita mejorar el trabajo planificado por la FEU y su emulación.

Objeto de estudio.

Plan de Trabajo de la FEU en la Universidad de las Ciencias Informáticas, y su emulación.

Objetivo general.

Analizar y diseñar una aplicación que permita el manejo del Plan de Trabajo y la Emulación de la Federación Estudiantil Universitaria en la Universidad de las Ciencias Informáticas.

Objetivos específicos.
Conocer el funcionamiento y la estructura del Plan de Trabajo y el sistema de Emulación de la FEU en la UCI.

Analizar la gestión del Plan de Trabajo.

Analizar la gestión de la Emulación.

Diseño de la gestión del Plan de Trabajo.

Diseño de la gestión del Plan de Trabajo.

Tareas de Investigación:

1. Determinar las principales tendencias históricas y de desarrollo de los sistemas de gestión de organizaciones estudiantiles en Cuba.
2. Valorar críticamente la situación actual de la FEU de la UCI mediante diagnóstico. Realizar un estudio a profundidad dentro de la UCI del funcionamiento del Plan de Trabajo y la Emulación.
3. Analizar los resultados del estudio anterior y tomar en cuenta los elementos esenciales que formarán parte del posterior análisis y diseño del sistema de gestión en cuestión.
5. Elaborar el modelo teórico.
6. Elaborar el instrumento práctico.
7. Someter el instrumento a valoración

Estructura del contenido:

Introducción: Una explicación breve acerca de los objetivos del Trabajo de Diploma.

Capitulo 1: Incluye un estado del arte del tema tratado, a nivel internacional, nacional y de la Universidad, de las tendencias, técnicas, tecnologías, metodologías y software usados en la actualidad o en las que se apoya para la solución del problema que se enfrenta, sobre los que es necesario profundizar. Debe ser un estudio crítico, valorativo, no una mera reproducción de referencias y estadísticas.

Capitulo 2: Contiene el modelo del negocio, o sea, el estudio del Plan de Trabajo y de la Emulación dentro la UCI. Se examina la estructura de estas dos funcionalidades y de los roles que existen en estos. Además se revisan los requisitos funcionales y los no funcionales que debe cumplir el sistema.

Capitulo 3: Incluye la definición del modelo de análisis del sistema con el modelo de clases. Describe los diagramas de secuencia del diseño y también muestra el diagrama de clases del diseño, y la descripción de cada una de estas clases. Por último se muestra el diseño de la Base de Datos y la descripción de cada una de las tablas relacionadas en el Modelo Entidad-Relación.

Capítulo 1: Fundamentación Teórica

En la actualidad existen varios softwares llamados de e-goverment, o gobiernos electrónicos, que permiten gestionar el desempeño de diferentes negocios u organizaciones de una forma informatizada. La FEU busca aprovechar estas tendencias para mejorar también su funcionamiento.

Sin embargo en Cuba no existen precedentes de la gestión informatizada de las organizaciones estudiantiles. Podemos encontrar como ejemplos muchos de los sitios informativos de las Universidades del país, pero en ninguno de estos se va más allá de la publicación de información estática.

Actualmente es el proyecto Kainos de la UCI el líder en esta materia en Cuba y ya se ha realizado una primera versión de un Sistema de Gestión Nacional de la FEU. Este trabajo es otra de las tesis que se encuentra en elaboración.

Por estos motivos podemos mencionar como antecedente de nuestro trabajo al un Sistema de Gestión Nacional de la FEU, donde se han sentado algunas de las pautas a seguir dentro de las herramientas a utilizar y la forma de organización dentro de cada uno de los módulos que pasarán a formar parte luego de este Sistema.

Software Libre:

La mayoría de las aplicaciones realizadas en nuestro país están regidas por una misma plataforma de ahí que usen software libre.

El término de software libre de lo contrario del software patentado se aplica a las aplicaciones informáticas que están libremente disponibles bajo un acuerdo de licencia pública de manera que cualquiera puede adaptarlos y mejorarlos. Las quejas más comunes sobre los programas de software patentados de Microsoft son sus costes relativamente elevados y que, cuando fallan, sólo Microsoft puede repararlos. Cada vez más, grandes organizaciones de toda clase optan por soluciones libres para sus necesidades

informáticas porque los productos libres son generalmente más baratos, más fiables y más fáciles de reparar cuando fallan. El mercado de las computadoras personales no se está quedando muy atrás.

“Software Libre" se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades de los usuarios del software:

1. La libertad de usar el programa, con cualquier propósito (libertad 0).
2. La libertad de estudiar cómo funciona el programa, y adaptarlo a tus necesidades (libertad 1). El acceso al código fuente es una condición previa para esto.
3. La libertad de distribuir copias, con lo que puedes ayudar a tu vecino (libertad 2).
4. La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. (libertad 3). El acceso al código fuente es un requisito previo para esto.

Un programa es software libre si los usuarios tienen todas estas libertades. Así pues, deberías tener la libertad de distribuir copias, sea con o sin modificaciones, sea gratis o cobrando una cantidad por la distribución, a cualquiera y a cualquier lugar. El ser libre de hacer esto significa (entre otras cosas) que no tienes que pedir o pagar permisos.

También deberías tener la libertad de hacer modificaciones y utilizarlas de manera privada en tu trabajo u ocio, sin ni siquiera tener que anunciar que dichas modificaciones existen. Si publicas tus cambios, no tienes por qué avisar a nadie en particular, ni de ninguna manera en particular.

La libertad para usar un programa significa la libertad para cualquier persona u organización de usarlo en cualquier tipo de sistema informático, para cualquier clase de trabajo, y sin tener obligación de comunicárselo al desarrollador o a alguna otra entidad específica.

En esta investigación optamos por un entorno mayormente de software libre analizando también que la política de nuestro país y en particular nuestra universidad, es ir migrando todos sus sistemas hacia esta tendencia.

Herramientas y Tecnologías:

Las herramientas utilizadas para la realización de este trabajo de diploma son las siguientes:

Visual Paradigm

Al escoger las herramientas para analizar y diseñar el software en cuestión nos basamos en las tendencias más actuales a nivel nacional e internacional, seleccionamos para el análisis y diseño la herramienta de modelado Visual Paradigm 5.3 Enterprise Edition pues aunque es Software privativo gratuito para modelado en UML, la universidad adquirió la licencia. Su uso se ha extendido entre los analistas de todo el mundo, dadas sus facilidades de integración con distintos entornos de desarrollo, incluye UML 2.0, brindando la posibilidad de modelar el sistema orientado a procesos en el futuro.

Subversion (Tortoise)

Para garantizar un correcto control de versiones y gestión de cambio se decidió utilizar el Subversion con el cliente Tortoise, ya que es una de las herramientas que más se ha venido utilizando en nuestra universidad ocupando el 98% de los proyectos que la utilizan actualmente, frente a un 2 % de los proyectos que usan el Visual Source Safe.

J2EE

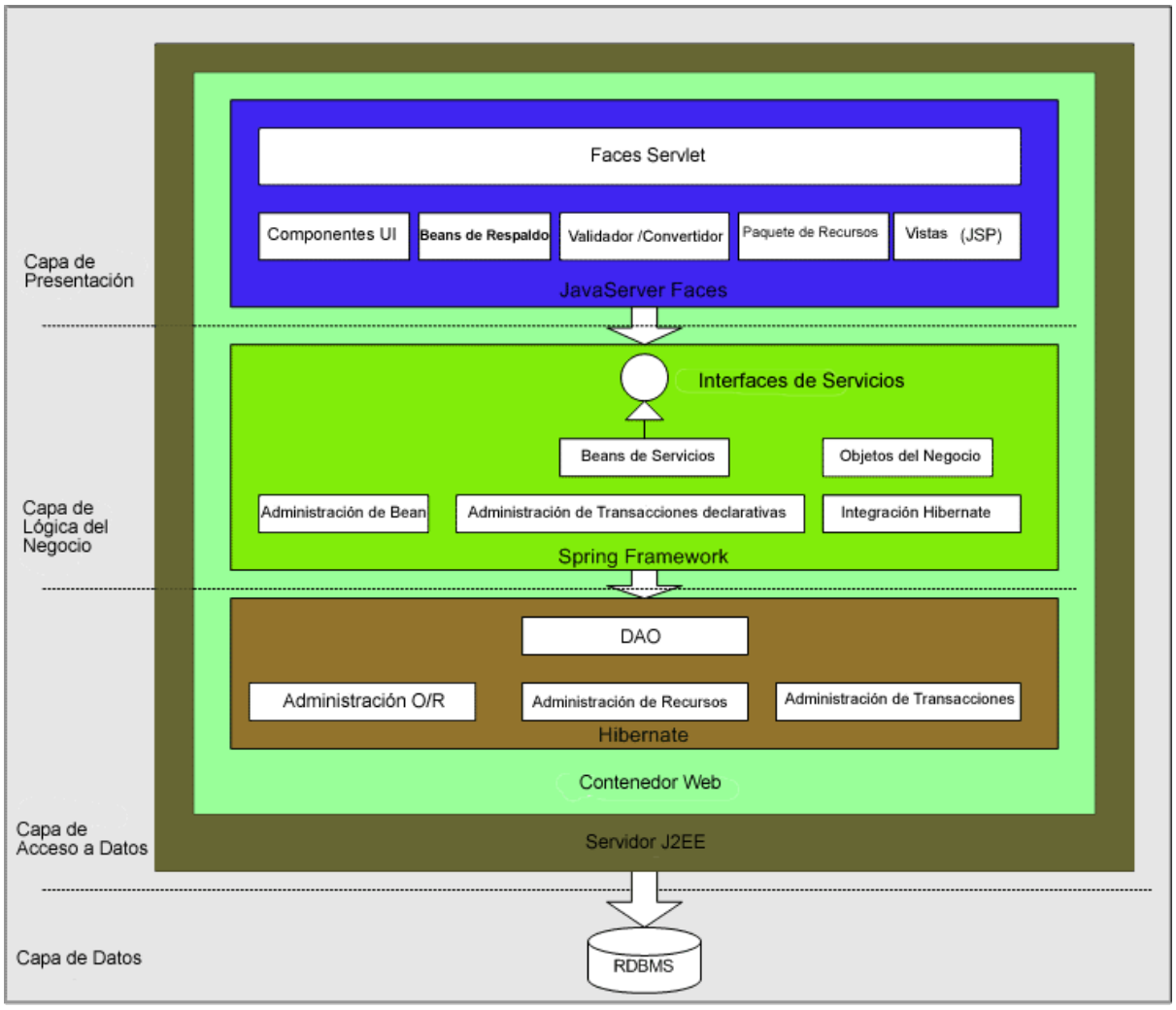
Java 2 Enterprise Edition, es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Java con arquitectura de N niveles distribuidos, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. Java EE es también considerada informalmente como un estándar. Es por esto que se decidió utilizar J2EE.

La utilización de Servidores de Aplicaciones y la ejecución de aplicaciones basadas en tecnología J2EE impactan en mayor o menor medida en distintas disciplinas dentro de una organización. Algunas de las más relevantes:

1. Modelado de Datos.
2. Análisis de Requerimientos.
3. Arquitectura y Diseño.

4. Codificación.
5. Testing.
6. Entorno y Operación.
7. Administración de Proyectos.

Arquitectura basada en Hibernate+Spring+JSF:



Diseño de la Arquitectura de Alto nivel

La arquitectura de alto nivel implica subdividir la aplicación en componentes funcionales y particionar estos componentes en capas. El diseño de la arquitectura de alto nivel es neutral a las tecnologías utilizadas.

Arquitectura Multi-Capa

Una arquitectura multicapa particiona todo el sistema en distintas unidades funcionales: cliente, presentación, lógica-de-negocio, acceso a datos, y bases de datos. Esto asegura una división clara de responsabilidades y hace que el sistema sea más mantenible y extensible. Los sistemas con tres o más capas se han probado como más escalables y flexibles que un sistema cliente-servidor, en el que no existe la capa central de lógica-de-negocios.

La capa del cliente es donde se consumen y presentan los modelos de datos. Para una aplicación Web, la capa cliente normalmente es un navegador web. Los clientes pequeños basados-en-navegador no contienen lógica de presentación; se trata en la capa de presentación.

La capa de presentación expone los servicios de la capa de lógica-de-negocio a los usuarios. Sabe cómo procesar una petición de cliente, cómo interactuar con la capa de lógica-de-negocio, y cómo seleccionar la siguiente vista a mostrar.

La capa de la lógica-de-negocio contiene los objetos y servicios de negocio de la aplicación. Recibe peticiones de la capa de presentación, procesa la lógica de negocio basada en las peticiones, y media en los accesos a los recursos de la capa de datos. Los componentes de la capa de lógica-de-negocio se benefician de la mayoría de lo servicios a nivel de sistema como el control de seguridad, de transacciones y de recursos.

La capa de acceso a datos es el puente entre la capa de lógica-de-negocio y la capa de datos. Encapsula la lógica para interactuar con la capa de datos.

Los datos de la aplicación persisten en la capa de datos. Contiene bases de datos relacionales, bases de datos orientadas a objetos, y sistemas antiguos.

La Capa de Presentación y JavaServer Faces

JSF es un marco de trabajo de componentes de interfaz de usuario del lado del servidor para aplicaciones Web basadas en Java. JSF contiene un API para representar componentes UI y manejar sus estados, manejar sus eventos, la validación del lado del servidor, y la conversión de datos, definir la navegación entre páginas, soportar internacionalización y accesibilidad; y proporcionar extensibilidad para todas estas características. También contiene dos librerías de etiquetas JSP (JavaServer Pages) personalizadas para expresar componentes UI dentro de una página JSP y para conectar componentes a objetos del lado del servidor.

La capa de presentación recoge la entrada del usuario, presenta los datos, controla la navegación por las páginas y delega la entrada del usuario a la capa de la lógica-de-negocio. La capa de presentación también puede validar la entrada del usuario y mantener el estado de sesión de la aplicación.

La Capa de Lógica-de-Negocio y el Marco de Trabajo Spring

Los objetos y servicios de negocio existen en la capa de lógica-de-negocio. Un objeto de negocio no sólo contiene datos, también la lógica asociada con ese objeto específico.

Los servicios de negocio interactúan con objetos de negocio y proporcionan una lógica de negocio de más alto nivel. Se debería definir una capa de interfaz de negocio formal, que contenga los interfaces de servicio que el cliente utilizará directamente. Con la ayuda del marco de trabajo Spring, implementará la capa de lógica-de-negocio de la aplicación JCatalog.

Manejo de Beans con contexto de aplicación:

Spring puede organizar de forma efectiva nuestros objetos de la capa central y manejar las conexiones por nosotros. Spring puede eliminar la proliferación de solitarios y facilita unas buenas prácticas de programación orientada a objetos, por ejemplo utilizando interfaces.

Integración con Hibernate:

Spring no nos fuerza a utilizar su potente característica de abstracción JDBC. Se integra bien con marcos de trabajo de mapeo O/R, especialmente con Hibernate. Spring ofrece un manejo seguro y eficiente de sesiones Hibernate, maneja la configuración de la SessionFactorie de Hibernate y las fuentes de datos JDBC en el contexto de la aplicación, y hace que la aplicación sea más fácil de testear.

La Capa de Integración e Hibernate

Hibernate es un marco de trabajo de mapeo Open Source que evita la necesidad de utilizar el API JDBC. Hibernate soporta la mayoría de los sistemas de bases de datos SQL. El *Hibernate Query Language*, diseñado como una extensión mínima, orientada a objetos, de SQL, proporciona un puente elegante entre los mundos objeto y relacional. Hibernate ofrece facilidades para recuperación y actualización de datos, control de transacciones, repositorios de conexiones a bases de datos, consultas programáticas y declarativas, y un control de relaciones de entidades declarativas.

RUP:

Utilizamos la metodología RUP por las tendencias actuales que existen, que también hablan mucho de la utilización de XP o Extreme Programming que proponen un ciclo de desarrollo de software de manera ágil y novedosa, contando siempre con un cliente interno que tendrá un contacto directo con el equipo de desarrollo, razón por la cual no es apropiado para nuestro sistema precisamente por las características de nuestro cliente, RUP es la que mejor se adapta a nuestro caso de estudio y a las condiciones del equipo de desarrollo, teniendo en cuenta y analizando que el organigrama de equipo del mismo es descentralizado democrático y la comunicación es abierta. RUP es uno de los procesos más generales y abarcadores de los existentes actualmente, ya que en realidad está pensado englobar todo el desarrollo del mismo, además de ser una metodología probada y segura. RUP se basa en casos de uso para

describir lo que se espera del software y está orientado a la arquitectura del sistema, documentándose lo mejor posible, basándose en UML (Unified Modeling Language) como herramienta principal.

Otras Propuestas:

Además de utilizar las herramientas y tecnologías antes mencionadas, en esta investigación proponemos las siguientes para el trabajo en las fases restantes:

Apache Tomcat:

Como servidor de aplicaciones la propuesta es Apache Tomcat, debido a que constituye uno de los más completos contenedores de Servlet gratuito, que programadores de Servlet de Java o Páginas de JavaServer (JSP) utilizan con frecuencia para probar su código. Tomcat persigue además la total compatibilidad con las versiones de Servlet y las especificaciones de API JSP que admite. Sin embargo, se trata de algo más que un servidor de pruebas, dado que muchas empresas lo emplean en la actualidad en entornos de producción debido a su contrastada estabilidad.

¿Por qué necesitamos usar Tomcat para ejecutar JAVA en Apache?

El funcionamiento principal de Apache desde su creación fue la de aceptar y responder requests de páginas de Internet, y como fue mencionado en servidores de páginas y JAVA Application Server. Estos requests correspondían a documentos estáticos (puro HTML), es por esto que cuando se necesita ejecutar algún tipo de contenido dinámico (programas como JAVA) es necesario coordinar los esfuerzos de Apache con otro ambiente. En el caso de JAVA es precisamente Tomcat. Quién ofrece facilidades para ejecutar los dos componentes más utilizados en ambientes JAVAS: JSP, (Java Server Pages) y servlets.

IDE Eclipse:

Como Entorno Integrado de Desarrollo (IDE) proponemos la utilización del Eclipse, el cual no es más que un software creado originalmente por IBM y actualmente, la Fundación Eclipse, una entidad sin ánimo de lucro, fundada por IBM se encarga de llevar a cabo el desarrollo de este proyecto. La definición que da la Fundación Eclipse acerca de su software es: "una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular".

Eclipse es un entorno integrado de desarrollo (IDE, Integrated Development Environment) multiplataforma englobado en el movimiento de software libre para crear aplicaciones clientes de cualquier tipo.

Beneficios.

- Es Open-Source.
- Soporta la construcción de una variedad de herramientas para el desarrollo de aplicaciones.
- Soporta el desarrollo de aplicaciones basadas en GUI y non-GUI.
- Soporta herramientas que manipulan diferentes tipos de archivos como por ejemplo Java, C, C++, EJB, HTML, GIF, etc.
- Corre en una gran cantidad de sistemas operativos incluyendo Windows y Linux.
- Provee a los desarrolladores, herramientas (ej.- PDE) que facilitan la creación de pluggins.
- Mediante JDT facilita la creación de aplicaciones programadas en Java.

Desventajas.

Si bien Eclipse es multiplataforma, los pluggins no tienen por qué serlo. Existen pluggins que sólo corren en una plataforma, o que aún no han sido desarrollados para más de una. Al ser una herramienta open-Source, se desarrollan pluggins que no tienen todas las funcionalidades que tienen en otras herramientas comerciales, como ser IBM Websphere.

PostgreSQL:

Proponemos el uso de PostgreSQL, por ser un motor de base de datos, servidor de base de datos relacional libre, liberado bajo una licencia de software libre.

Algunas de sus características fundamentales son:

- Alta concurrencia
- Amplia variedad de tipos nativos
- Los usuarios pueden crear sus propios tipos de datos.
- Claves ajenas también denominadas Llaves ajenas o Llaves Foráneas (*foreign keys*).
- Disparadores (*triggers*).
- Vistas.
- Integridad transaccional.
- Herencia de tablas.
- Tipos de datos y operaciones geométricas.
- Funciones

Inconvenientes:

- Consume recursos y carga el sistema.
- Límite del tamaño de cada fila de las tablas a 8k.
- Es de 2 a 3 veces más lenta que MySQL.
- Menos funciones en PHP.

Java:

Proponemos que la implementación sea en el lenguaje JAVA dadas sus características y potencialidades. Java está totalmente basado en clases y objetos. Todo en Java (aparte de algunos tipos primitivos) es un objeto. Contrariamente a lenguajes híbridos como C++ o el popular Visual Basic, en Java no se permite programar fuera de un objeto o clase. No hay módulos ni funciones globales.

Una característica todavía más distintiva de Java es su capacidad multiplataforma. Lenguajes como C o COBOL se han implementado en múltiples plataformas, pero siempre han necesitado recopilaciones o adaptaciones al pasar de una a otra. En cambio, Java desde el principio ha sido pensado para adaptarse a varios entornos. Esto lo consigue no sólo a nivel de código fuente, sino también a nivel de código compilado. El programa que escribimos se puede compilar en Windows o en Linux, y funciona. Pero, además, hasta el programa compilado puede ejecutarse sin más preparación, en distintas máquinas. Eso

lo consigue porque Java se compila y ejecuta, no en un procesador o entorno en particular, sino en lo que se llama una "virtual machine", una máquina virtual. Nuestro programa Java podrá ejecutarse en cualquier sistema operativo que tenga una máquina virtual Java compatible.

Java no es solamente un lenguaje, es una tecnología. Al estar basado en clases y objetos, viene acompañado de un conjunto de éstos, que nos sirven como base para la programación de aplicaciones, de texto, gráficas o que se ejecuten en una página Web como "applets".

Java maneja prácticamente todas las bases de datos relacionales, desde Microsoft SQL Server, Oracle, Sybase, Informix, como Access, MySQL y PostgreSQL.

Conclusión: Capítulo 1

La FEU de la UCI, como vanguardia en los temas relacionados con las TICs, está explorando nuevas posibilidades dentro de su funcionamiento, como lo es la gestión automatizada de su trabajo. Este paso atrevido demuestra las potencialidades que esconde esta organización estudiantil capaz de mantenerse a tono con su época.

Los sistemas de gestión, cada uno diseñado alrededor de su organismo, pretenden agilizar y mejorar su funcionamiento a través de la informatización. Nuestro proyecto, Kainos, pretende estar a la altura de la necesidad planteada y de ofrecer un producto capaz de cumplir con las expectativas que de ella se demandan.

Capítulo 2: características del sistema

Problema y situación problémica:

La FEU tiene entre sus tareas el diseño de un plan de Trabajo que le permite mantener un control respecto a las actividades que se van a desarrollar a lo largo de curso. También tiene la responsabilidad de crear un plan de emulación para promover la competencia y el mejor funcionamiento de las facultades, en los diferentes aspectos que se emulan.

Para el desarrollo de estas dos actividades existe un responsable de Plan de Trabajo y Emulación, el cual es el encargado de realizar estas dos tareas.

Actualmente esto equivale a que los dirigentes de la FEU, principalmente el Vicepresidente, exige la realización de estas tareas, además de controlar su cumplimiento. Luego el responsable realiza tanto el Plan de Trabajo como el sistema de Emulación. Una vez creados estos, debe mantener el Plan de Trabajo actualizado con las actividades que se agregan, eliminan o cambian de fecha. Igual ocurre con el Sistema de emulación, del que debe estar al tanto de cuales son los resultados de cada uno de los elementos en los que las facultades compiten.

Todas estas actividades que lleva a cabo el responsable del Plan de Trabajo y Emulación se realizan de una forma no automatizada, trayendo como consecuencia que las actividades no se divulguen a tiempo, y que la emulación no sea clara y transparente.

Modelo de negocio.

Actores del Negocio

Actores del negocio	Justificación
Dirigentes de la FEU	**Gestión del Plan de Trabajo.** Son los interesados de tener el control de las actividades de la organización.
	Gestión de la Emulación. Son los que solicitan la confección de una emulación para competir en diferentes esferas.
	Gestionar Resultados de Emulación. Así como solicitan la confección de la emulación, exigen que los resultados obtenidos en las emulaciones se mantenganactualizados.
	Mostrar Calendario. Son los interesados en revisar el calendario, cuando necesiten.
	Mostrar Emulación. Son los interesados en revisar el estado de la emulación inter-facultades.

Trabajadores del Negocio.

Trabajadores del negocio	Justificación
Responsable de Plan de Trabajo y Emulación	**Gestión del Plan de Trabajo.** Es el encargado de llevar el control de las actividades y llevar el calendario lo más actualizado posible.
	Gestión de la Emulación. Es el encargado de confeccionar anualmente el sistema a través del cual las facultades emularán.
	Gestionar Resultados de la Emulación. También se responsabiliza de tener los resultados de la Emulación en las diferentes esferas y actualizar los resultados.
	Mostrar Calendario. Cuando un Dirigente se muestra interesado por conocer el calendario de actividades, debe buscarlo para mostrarlo.
	Mostrar Emulación. De la misma forma cuando un dirigente se muestra interesado por saber cual es el estado de la emulación, debe tener los resultados actualizados para mostrarlos.

Diagrama de Casos de Uso del Negocio

El Diagrama de Casos de Uso muestra la relación entre los actores y los casos de uso y permite un mejor entendimiento del mismo, llevándolo a un modelo comprensible para todos. Básicamente nuestro diagrama de casos de uso del negocio esta compuesto por cinco casos de uso:

- Gestión del Plan de Trabajo.
- Gestión de la Emulación.
- Gestión de Resultados de la emulación.
- Mostrar Calendario.
- Mostrar Emulación.

Ver diagrama de casos de uso del negocio en el Anexo.

Diagramas de Actividades.

Los diagramas de actividades describen el flujo de la información en cada uno de los Casos de Usos del negocio. Describen como interactúa el Actor del Negocio con el Trabajador del negocio, y cuales son las entidades que se generan o utilizan.

Ver diagramas de Actividades en el Anexo.

Modelo de Objetos

Como parte del proceso del modelado del negocio se desarrolla un modelo de objetos del negocio compuesto por trabajadores, entidades del negocio y unidades de trabajo.

Una entidad del negocio representa algo que los trabajadores toman, manipulan, modifican, utilizan. Una unidad de trabajo es un conjunto de entidades de trabajo.

En el caso de nuestro negocio en particular podemos ver que las entidades del negocio están todas relacionadas con un solo actor: el Responsable de Plan de Trabajo y Emulación, quién actualmente realiza todas las tareas del negocio.

Ver Modelo de Objetos del Negocio en el Anexo.

Especificación de los requisitos de software.

1) Se debe autenticar en el sistema de la Universidad donde cada persona tiene un usuario único.
2) Debe ser accesible desde el portal del sitio Kainos, donde existen otras funcionalidades relacionada con el trabajo de la FEU.

Requerimientos Funcionales:

1. Confeccionar Calendario.
2. Confeccionar un Sistema de Emulación por puntos.
3. Notificación automatizada para actividades.
4. Mostrar calendario.
5. Mostrar el resultado de la emulación de las Facultades.
6. Adicionar actividades en el calendario.
7. Modificar actividades en el calendario.
8. Eliminar actividades en el calendario.
9. Adicionar puntos a emular.
10. Modificar puntos a emular.
11. Eliminar puntos a emular.
12. Manejar los resultados de la emulación por cada punto.

Requerimientos no funcionales. Apariencia o interfaz externa.

- Usabilidad e Interfaz:

El sistema deberá tener una interfaz sencilla de maniobrar. Deberá también tener un aspecto amigable al usuario.

- Rendimiento. Soporte. Portabilidad

La aplicación funcionará a través de un portal web, para lo que se necesita un servidor para la aplicación y conectividad a la red interna de la universidad. Para ello deberá ser accesible tanto desde entornos de software libre (como son las diferentes versiones de Linux) y Mozilla Firefox, como desde

ambientes Microsoft Windows y su Internet Explorer. Al ser accesible desde la red, los servicios de la aplicación estarán disponibles para todos los usuarios.

- Seguridad

La seguridad estará basada en el modelo de usuario-contraseña siguiendo la filosofía de niveles de accesos. Además, el lenguaje Java brindará mayor seguridad a través de diferentes Beans.

- Ayuda y documentación en línea.

El portal contará con un sistema de ayuda para facilitar el uso de la aplicación y sus funciones.

Definición de los casos de uso del negocio.

<table>
<tr><td>Nombre del CU</td><td colspan="2">Gestionar Emulación</td></tr>
<tr><td>Actor</td><td colspan="2">Dirigente</td></tr>
<tr><td>Trabajador</td><td colspan="2">Responsable Plan de Trabajo y Emulación</td></tr>
<tr><td>Breve Descripción</td><td colspan="2">A través de este proceso se gestionan todos los puntos de emulación y el plan de emulación. Las funciones que comprende son: Crear Plan de Emulación, Adicionar Punto de Emulación, Actualizar Punto de Emulación y Eliminar Punto de Emulación.</td></tr>
<tr><td>Precondiciones</td><td colspan="2"></td></tr>
<tr><td rowspan="6">Flujo Normal de los eventos</td><td>Acción del Actor</td><td>Respuesta del Negocio</td></tr>
<tr><td>1. Necesita un Plan de Emulación actualizado.</td><td></td></tr>
<tr><td></td><td>1.1 Comprueba si existe un Plan de Emulación.</td></tr>
<tr><td></td><td>1.2 En caso de que exista un Plan pregunta que acción a tomar: Actualizar un punto, Adicionar un punto de o ninguna otra acción.</td></tr>
<tr><td></td><td>1.3 En el caso de Actualizar punto, muestra todos los puntos de Emulación</td></tr>
<tr><td>2. Escoge que punto Modificar.</td><td></td></tr>
</table>

		2.1 Pregunta que acción a tomar: Cambiar o Eliminar.
		2.2 En el caso de que se vaya a cambiar pide los nuevos datos a insertar.
	3. Introduce los nuevos datos	
		3.1 Actualiza el punto
		3.2 Actualiza el plan de emulación.
		3.3 Pregunta si va a continuar.
		3.4 En caso negativo se termina el proceso.
Flujos Alternos	**Acción del Actor**	**Respuesta del Negocio**
	Acción 1.1	
		1.4 Si no existe un Plan de Emulación, el Responsable Crea uno nuevo y pasa a la acción 1.2.
	Acción 1.2	
		1.5 En el caso de Adicionar punto, Pide los datos del nuevo punto.
	4. Introduce los datos del punto a confeccionar.	
		4.1 Crea el nuevo punto de emulación y pasa a la acción 3.2.
	Acción 1.2	
		1.6 En el caso de ninguna otra pasa directamente hasta la acción 3.4
	Acción 2.1	
		2.3 En el caso de eliminar, elimina el punto de emulación y pasa a la acción 3.2.
	Acción 3.3	
		3.5 En el caso de desear continuar regresa a la acción 1.2

Nombre del CU	**Gestionar Plan de Trabajo**	
Actor	Dirigente	
Trabajador	Responsable Plan de Trabajo y Emulación	
Breve Descripción	A través de este proceso se gestionan todos las actividades y su calendario. Las funciones que comprende son: Crear Calendario, Adicionar Actividad, Actualizar Actividad y Eliminar Actividad.	
Precondiciones		
Flujo Normal de los eventos	**Acción del Actor**	**Respuesta del Negocio**
	1. Necesita un Calendario actualizado.	
		1.1 Comprueba si ya existe un Calendario.
		1.2 En caso de que exista un Calendario pregunta que acción a tomar: Actualizar una Actividad, Adicionar una Actividad de o ninguna otra acción.
		1.3 En el caso de Actualizar Actividad, muestra todos los Actividades del calendario.
	2. Escoge que Actividad Modificar.	
		2.1 Pregunta que acción a tomar: Cambiar o Eliminar.
		2.2 En el caso de que se vaya a cambiar pide los nuevos datos a insertar.
	3. Introduce los nuevos datos	
		3.1 Actualiza la Actividad
		3.2 Actualiza el calendario.
		3.3 Pregunta si va a continuar.

		3.4 En caso negativo se termina el proceso.
Flujos Alternos	**Acción del Actor**	**Respuesta del Negocio**
	Acción 1.1	
		1.4 Si no existe un Calendario, el Responsable Crea uno nuevo y pasa a la acción 1.2.
	Acción 1.2	
		1.5 En el caso de Adicionar Actividad, Pide los datos de la nueva Actividad.
	4. Introduce los datos de la Actividad a confeccionar.	
		4.1 Crea la nueva Actividad y pasa a la acción 3.2.
	Acción 1.2	
		1.6 En el caso de ninguna otra pasa directamente hasta la acción 3.4
	Acción 2.1	
		2.3 En el caso de eliminar, elimina la Actividad y pasa a la acción 3.2.
	Acción 3.3	
		3.5 En el caso de desear continuar regresa a la acción 1.2

Nombre del CU	**Gestionar Resultados de Emulación**
Actor	Dirigente
Trabajador	Responsable Plan de Trabajo y Emulación
Breve Descripción	Cuando se celebra alguna de las actividades en las que se está emulando, se recogen los resultados, los cuales se actualizan.

Precondiciones		
Flujo Normal de los eventos	**Acción del Actor**	**Respuesta del Negocio**
	1. Va a actualizar los resultados de la Emulación	
		1.1 Pregunta cual de los puntos de Emulación va a actualizar.
	2. Escoge el punto a actualizar e introduce los nuevos datos.	
		2.1 Actualiza los valores del punto seleccionado.
		2.2 Pregunta si va a continuar.
		2.3 En caso negativo se termina el proceso.
Flujos Alternos	**Acción del Actor**	**Respuesta del Negocio**
	Acción 2.2	
		2.4 En el caso que se desee continuar, regresa a la acción 1.1

Nombre del CU	**Mostrar Calendario**	
Actor	Dirigente	
Trabajador	Responsable Plan de Trabajo y Emulación	
Breve Descripción	Cuando se solicita conocer las actividades previstas, el Responsable del Plan de Trabajo y Emulación busca el Calendario que ha confeccionado y lo muestra.	
Precondiciones	Que ya exista un calendario hecho.	
Flujo Normal de los eventos	**Acción del Actor**	**Respuesta del Negocio**
	1. Pide el calendario de actividades.	
		1.1 Busca el calendario que ha creado.

		1.2 Muestra el calendario confeccionado.
Flujos Alternos	**Acción del Actor**	**Respuesta del Negocio**

Nombre del CU	**Mostrar Emulación**	
Actor	Dirigente	
Trabajador	Responsable Plan de Trabajo y Emulación	
Breve Descripción	Cuando se solicita conocer el estado de la Emulación, el Responsable del Plan de Trabajo y Emulación busca los resultados de Emulación que ha ido controlando y los muestra.	
Precondiciones	Que ya exista un Plan de Emulación.	
Flujo Normal de los eventos	**Acción del Actor**	**Respuesta del Negocio**
	1. Pide los resultados de la emulación.	
		1.1 Busca los resultados de Los puntos emulados.
		1.2 Muestra los resultados.
Flujos Alternos	**Acción del Actor**	**Respuesta del Negocio**

Objeto de automatización.

Después de analizar el funcionamiento actual del Plan de Trabajo y Emulación, determinamos que para un mejor funcionamiento es necesario automatizar las siguientes funciones:

- Gestión del Plan de Trabajo.

Comprende la confección y actualización de un calendario formado con las actividades de la FEU.

- Gestión de la Emulación.

Comprende la confección y actualización de un plan de Emulación formado por los diferentes puntos que son objetivos de la emulación.

- Gestionar Resultados de la Emulación.

Es la actualización, luego de cada evento competitivo entre facultades, de los resultados de la emulación.

- Notificación de Actividades.

Al acercarse la fecha de una actividad del calendario, se envía una notificación automatizada por correo a los interesados.

- Mostrar Calendario

Cada vez que un dirigente se interese por saber cuales son las actividades del calendario puede verlo.

- Mostrar Emulación.

Cada vez que un estudiante desee saber cual es el resultado de la emulación de las facultades puede verlo.

Información que se maneja.

Como hemos podido ver los principales documentos que se manejan actualmente son el Calendario de Actividades y el Plan de Emulación.

Propuesta de sistema.

Para dar solución al problema presentado, proponemos una automatización de las funciones correspondientes al Responsable del Plan de Trabajo y Emulación.

Partimos de la concepción de que la función del Responsable del Plan de Trabajo y Emulación, se pueden dividir en dos Trabajadores del negocio diferentes: El responsable de Plan de Trabajo y el Responsable de Emulación. Ambos responsables son Dirigentes de la FEU, que a su vez son Estudiantes universitarios.

En la Gestión del Plan de Trabajo el Responsable correspondiente solo tendrá que agregar, eliminar o cambiar las actividades en el calendario a través de un sistema informatizado.

De igual forma la Gestión de la Emulación tendrá las funciones de agregar, eliminar o cambiar los puntos de la emulación. Los resultados de la emulación serán actualizados también a través del sistema, otorgando sólo los valores predefinidos por su Responsable.

También se podrán consultar a través de la red los resultados de Emulación y en caso de ser dirigente, se podrán consultar también, el calendario con las actividades previstas. Además para mantener a los dirigentes actualizados con las actividades en que deben participar, el sistema les enviará una notificación a tiempo de forma automática.

De esta forma, una vez las funciones controladas por un sistema informatizado, se podrá tener una organización clara de las actividades y la Emulación podrá ser transparente para todos los estudiantes.

Ver Diagrama de Casos de Uso del Sistema en el Anexo 1.

Descripción de Casos de Uso del Sistema.

Caso de Uso:	CU-1 Gestionar Emulación
Actor:	Responsable de Emulación.
Resumen:	Permite la gestión global de la Emulación agregando modificando y eliminando los diferentes puntos de Emulación.
Precondiciones:	
Referencias	RF-2, RF-9, RF-10, RF-11
Prioridad	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor solicita gestionar el Plan de emulación.	**1.1** Comprueba si existe un Plan de Emulación.
	1.2 En caso de que exista un Plan, pregunta que acción a tomar: Actualizar un punto, Adicionar un punto de o ninguna otra acción.
	1.3 En el caso de Actualizar punto, muestra todos los puntos de Emulación
2. Escoge que punto Modificar.	**2.1** Pregunta que acción a tomar: Cambiar o Eliminar.
	2.2 En el caso de que se vaya a eliminar pide confirmación de eliminar.
3. Confirma si va a eliminar o no.	
	3.1 En caso positivo elimina el punto de emulación.
	3.2 Actualiza el plan de emulación.
	3.3 Pregunta si va a continuar.

	3.4 En caso negativo se termina el proceso.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 1.1	**1.4** Si no existe un Plan de Emulación, el Responsable Crea uno nuevo y pasa a la acción 1.2.
Acción 1.2	**1.5** En el caso de Adicionar punto, Pide los datos del nuevo punto.
4. Introduce los datos del punto a confeccionar.	**4.1** Crea el nuevo punto de emulación y pasa a la acción 3.2.
Acción 1.2	**1.6** En el caso de ninguna otra pasa directamente hasta la acción 3.4
Acción 2.1	**2.3** En el caso de cambiar pide los nuevos datos a insertar.
5. Introduce los nuevos datos	**5.1** Actualiza el punto y pasa a la acción 3.2.
Acción 3.1	**3.5** Si el decide no eliminar el punto, el sistema retorna a la acción 3.3.
Acción 3.3	**3.6** Si el actor desea continuar la gestión de la emulación, el sistema regresa a la acción 1.2
Post- condiciones	El Plan de Emulación debe haber sido actualizado.

Caso de Uso:	CU-2 Gestionar Plan de Trabajo
Actor:	Responsable de Plan de Trabajo.
Resumen:	Permite la gestión global del Plan de Trabajo agregando modificando y eliminando las diferentes Actividades del Calendario.
Precondiciones:	
Referencias	RF-1, RF-6, RF-7, RF-8
Prioridad	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor solicita gestionar el Plan de trabajo.	**1.1** Comprueba si existe un Calendario de Actividades..
	1.2 En caso de que exista un Calendario, pregunta que acción a tomar: Actualizar una actividad, Adicionar una actividad de o ninguna otra acción.
	1.3 En el caso de Actualizar actividad, muestra todas las actividades del Calendario.
2. Escoge que actividad Modificar.	**2.1** Pregunta que acción a tomar: Cambiar o Eliminar.
	2.2 En el caso de que se vaya a eliminar pide confirmación de eliminar.
3. Confirma si va a eliminar o no.	
	3.1 En caso positivo elimina la actividad del Calendario.
	3.2 Actualiza el Calendario.
	3.3 Pregunta si va a continuar.
	3.4 En caso negativo se termina el proceso.
Prototipo de Interfaz	
Flujos Alternos	

Acción del Actor	Respuesta del Sistema
Acción 1.1	**1.4** Si no existe un Calendario, el Responsable crea uno nuevo y pasa a la acción 1.2.
Acción 1.2	**1.5** En el caso de Adicionar Actividad, Pide los datos de la nueva actividad.
4. Introduce los datos de la actividad a confeccionar.	**4.1** Crea la nueva actividad y pasa a la acción 3.2.
Acción 1.2	**1.6** En el caso de ninguna otra pasa directamente hasta la acción 3.4
Acción 2.1	**2.3** En el caso de cambiar pide los nuevos datos a insertar.
5. Introduce los nuevos datos	**5.1** Actualiza la actividad y pasa a la acción 3.2.
Acción 3.1	**3.5** Si el decide no eliminar la actividad, el sistema retorna a la acción 3.3.
Acción 3.3	**3.6** Si el actor desea continuar la gestión de la plan de trabajo, el sistema regresa a la acción 1.2
Post- condiciones	El Plan de Trabajo debe haber sido actualizado.

Caso de Uso:	CU-3 Gestionar Resultados de Emulación
Actor:	Responsable de Emulación.
Resumen:	Permite actualizar los resultados de la emulación.
Precondiciones:	Ya existe un Plan de Emulación con Puntos a emular.
Referencias	RF-12

Prioridad	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Va a actualizar los resultados de la Emulación	**1.1** Pregunta cual de los puntos de Emulación va a actualizar.
2. Escoge el punto a actualizar e introduce los nuevos datos.	**2.1** Actualiza los valores del punto seleccionado.
	2.2 Pregunta si va a continuar.
	2.3 En caso negativo se termina el proceso.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 2.2	**2.4** En el caso que se desee continuar, regresa a la acción 1.1
Post- condiciones	Los resultados de la emulación deben haber sido actualizados.

Caso de Uso:	CU-4 Mostrar Calendario
Actor:	Dirigente.
Resumen:	Muestra el Calendario con las Actividades programadas.
Precondiciones:	Ya existe un Calendario con Actividades.
Referencias	RF-4
Prioridad	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**

1. Pide el calendario de actividades.	**1.1** Busca el calendario que ha sido creado.
	1.2 Muestra en una página el calendario confeccionado.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Post- condiciones	El Calendario de Actividades sea mostrado.

Caso de Uso:	CU-5 Mostrar Resultados.
Actor:	Estudiante.
Resumen:	Muestra el los resultados de las diferentes actividades que han sido emuladas.
Precondiciones:	Ya existe un Plan de Emulación con Puntos a emular.
Referencias	RF-5
Prioridad	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Pide los resultados de la emulación.	**1.1** Busca los resultados de Los puntos emulados.
	1.2 Muestra los resultados.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Post- condiciones	Los resultados de la emulación sean mostrados.

Caso de Uso:	CU-6 Notificar Actividad.
Actor:	Dirigente.
Resumen:	Cuando una actividad planificad en el calendario está próxima a su fecha de cumplimiento se envía una notificación automática a los involucrados.
Precondiciones:	Ya existe un Calendario con Actividades. El sistema tenga acceso a algún tipo de correo electrónico.
Referencias	RF-5
Prioridad	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
	1. Revisa sistemáticamente las fechas de las actividades.
	1.1 Si existe una actividad próxima a celebrarse, se confecciona una notificación a partir de los datos de la actividad.
	1.2 Envía la notificación por correo electrónico.
2. Recibe el correo electrónico.	**2.1** Termina el proceso.

Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Acción 1.1	Si no existe ninguna actividad próxima a celebrarse se pasa a la acción 2.1.

Post- condiciones	La notificación se envíe satisfactoriamente.

Conclusiones: Capítulo 2

Como hemos podido ver a lo largo de este capítulo la función del Plan de Trabajo consiste en la gestión y control de las actividades que se organizan dentro de un Calendario. Por otra parte la emulación se gestiona a partir de un conjunto de puntos que conforman el plan de emulación.

Los casos de uso del negocio describen como se realizan todos los procesos de gestión por un solo trabajador, el Responsable de Plan de Trabajo y Emulación. A partir de estos, nuestra propuesta de Sistema busca organizar y facilitar el trabajo de este trabajador, al cual convertiremos por conveniencia en dos trabajadores diferentes para separar los procesos del Plan de Trabajo y los de la Emulación.

CAPÍTULO 3: ANÁLISIS Y DISEÑO DEL SISTEMA

Dentro del Proceso Unificado del Desarrollo de Software, el Análisis y Diseño es uno de los Flujos de Trabajo fundamentales, por estar más cercano a la arquitectura final del sistema y el acceso a los datos.

El análisis ofrece un refinamiento mayor de los requisitos dándoles una estructura de mayor comprensión, además de servir como una primera aproximación al diseño. El diseño por su parte es el responsable de dar una forma más consistente al sistema, utilizando el lenguaje de programación final, por lo que el resultado del modelo del negocio es un paso previo al flujo de trabajo de implementación.

En este capítulo realizaremos, pues, el análisis y el diseño del sistema que proponemos, dejándolo listo para otros trabajos posteriores que puedan realizar su implementación.

Análisis.

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describe cómo implementar el sistema. El análisis consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. A pesar de que el modelo del análisis hay un refinamiento de los requisitos, no se toman en cuenta el lenguaje de programación a usar en la construcción, la plataforma en la que se ejecutará la aplicación, los componentes prefabricados o reusables de otras aplicaciones, entre otras características que afectan al sistema, ya que el objetivo del análisis es comprender perfectamente los requisitos del software y no precisar cómo se implementará la solución.

En el análisis se desarrollan los diagramas de clases del análisis. El objetivo de estos es ofrecer un descripción visual de cómo es la interacción entre las diferentes capas que conforman al sistema, y el actor del sistema.

Las clases son abstracciones que representan un objeto a partir de atributos y métodos. En el Diagrama de Clases del Análisis se utilizan principalmente tres tipos de clases:

- Las clases Interfaz: permiten al actor interactuar con el sistema, pueden ser una pantalla, un reporte, formulario, etc.
- Las clases de Control: representan coordinación, secuencia, transacciones y control de otros objetos y se usan con frecuencia para encapsular el control de un caso de uso en concreto. También realizan cálculos complejos dentro de la lógica del negocio.
- Las clases Entidad: representan información o datos persistentes; suelen mostrar una estructura de datos lógica y contribuyen a comprender de qué información depende el sistema.

Definición del modelo de análisis. Modelo de clases de análisis.
Ver Diagrama de Clases de Análisis en el Anexo 2.

Diseño.

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva cómo cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades.

El modelo de diseño intenta preservar la estructura definida por el modelo de análisis. Es un modelo de objetos que describe la realización física de los casos de uso, además de tener en cuenta los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia, tecnologías de interfaz de usuario, tecnologías de gestión de transacciones, entre otras. Otra de las funciones del diseño es que crea un punto de partida para las actividades de implementación.

Ver Diagrama de Secuencia en el Anexo 2.

DIAGRAMA CLASES DEL DISEÑO

Ver Diagrama de Clases del Diseño en el Anexo 2.

Descripción de las clases.

Nombre: Actividad	
Tipo de clase: Entidad	
Atributo	**Tipo**
Fecha	date
Responsible	string[]
Citados	string[]
Nombre	string
Descripción	string
Id_Act	int
Para cada responsabilidad:	
Nombre:	**Descripción:**

Nombre: Calendario	
Tipo de clase: Entidad	
Atributo	**Tipo**
Actividades	int[]
Id_Cal	int
Para cada responsabilidad:	
Nombre:	**Descripción:**

Nombre: Notificacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Destinatario	string[]
Actividad	Actividad
Para cada responsabilidad:	
Nombre:	**Descripción:**

Nombre: Punto_Emulacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Valor_Max	int
Resultados	int
Nombre	string
Id_Pto	int
Para cada responsabilidad:	
Nombre:	**Descripción:**
Nombre: Plan_Emulacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Puntos	int[]
Id_Plan	int
Para cada responsabilidad:	
Nombre:	**Descripción:**

Nombre: ActividadDaoImpl	
Tipo de clase: Control	
Atributo	**Tipo**
Actividad	Actividad
Para cada responsabilidad:	
Nombre:	**Descripción:**
Crear_Actividad	Crea una Actividad.
Obtener_Responsables	Obtiene una lista con los responsables de una actividad.
Obtener_Citados	Obtiene una lista con los citados de una actividad.
Obtener_Fecha	Obtiene la fecha de una actividad
Cambiar_Fecha	Cambia la fecha de una actividad.

Cambiar_Responsables	Cambia la lista de los responsables de una actividad.
Cambiar_Citados	Cambia la lista de los citados de una actividad.
Cambiar_Nombre	Cambia el nombre de una actividad.
Obtener_Nombre	Obtiene el nombre de una actividad.
Obtener_Descripcion	Obtiene la descripción de una actividad.
Cambiar_Descripcion	Cambia la descripción de una actividad.

Nombre: CalendarioDaoImpl	
Tipo de clase: Control	
Atributo	**Tipo**
Calendario	Calendario
Para cada responsabilidad:	
Nombre:	**Descripción:**
Obtener_Actividad	Obtiene una actividad del calendario.
Obtener_Actividades	Obtener el listado de actividades del calendario.
Buscar_Act	Busca una actividad en el calendario a partir de su nombre.
Crear_Calendario	Crea el Calendario.
Cambiar_Actividad	Cambia una actividad del calendario.
Agregar_Actividad	Agrega una actividad al calendario.
Eliminar_Act	Elimina una actividad del calendario.

Nombre: NotificacionDaoImpl	
Tipo de clase: Control	
Atributo	**Tipo**
Notificacion	Notificacion
Para cada responsabilidad:	
Nombre:	**Descripción:**
Crear_Notificacion	Crea una notificación.

Obtener_Destinatarios	Obtiene los destinatarios de la actividad que forma la Notificación para la creación de la notificación.

Nombre: Punto_EmulacionDaoImpl	
Tipo de clase: Control	
Atributo	**Tipo**
Punto_Emulacion	Punto_Emulacion
Para cada responsabilidad:	
Nombre:	**Descripción:**
Crear_Pto	Crea un punto de emulación.
Obtener_Valor	Obtiene el valor máximo de un punto de emulación.
Obtener_Nombre	Obtiene el nombre de un punto de emulación.
Obtener_Resultados	Obtiene la lista de resultados de un punto de emulación.
Cambiar_Valor	Cambia el valor máximo de un punto de emulación.
Cambiar_Nombre	Cambia el nombre de un punto de emulación.
Cambiar_Resultados	Cambia el listado de resultados de un punto de emulación.

Nombre: Plan_EmulacionDaoImpl	
Tipo de clase: Control	
Atributo	**Tipo**
Plan_Emulacion	Plan_Emulacion
Para cada responsabilidad:	
Nombre:	**Descripción:**
Crear_Plan	Crea el Plan de Emulación.
Obtener_Resultados	Obtiene el listado de resultados de un punto del Plan de Emulación.
Obtener_Pto	Obtiene un Punto de Emulación del plan de Emulación.
Adicionar_Pto	Adiciona un Punto de Emulación en el Plan de Emulación.

Cambiar_Pto	Cambia un Punto de Emulación en el Plan de Emulación.
Eliminar_Pto	Elimina un Punto de Emulación en el Plan de Emulación.
Buscar_Pto	Busca un Punto de Emulación en el Plan de Emulación por su nombre.

Nombre: ServiciosPlanDeTrabajoImpl	
Tipo de clase: Control	
Atributo	**Tipo**
ActividadDao	ActividadDao
CalendarioDao	CalendarioDao
NotificacionDao	NotificacionDao
Para cada responsabilidad:	
Nombre:	**Descripción:**
Adicionar_Actividad	Adiciona una nueva actividad en el Calendario.
Crear_Actividad	Crea una nueva actividad.
Crear_Calendario	Crea un Calendario
Crear_Vista	Crea una vista con las Actividades en el Calendario.
Crear_Notificacion	Crea una Notificación de una Actividad próxima.
Cambiar_Actividad	Cambia una Actividad del Calendario.
Buscar_Actividad	Busca una Actividad en el Calendario.
Obtener_Calendario	Obtiene el Calendario.
Obtener_Actividades	Obtiene el lista de Actividades en el Calendario.
Chequera_Actividades	Chequea las Actividades para comprobar si hay alguna próxima a celebrarse.
Enviar_Notificaciones	Envía una Notificación.
Eliminar_Actividad	Elimina una Actividad del Calendario.

Nombre: ServiciosEmulacionImpl	
Tipo de clase: Control	
Atributo	**Tipo**
Plan_EmulacionDao	Plan_EmulacionDao
Punto_EmulacionDao	Punto_EmulacionDao
Para cada responsabilidad:	
Nombre:	**Descripción:**
Crear_Pto	Crea un Punto de Emulación.
Crear_Plan	Crea un Plan de Emulación.
Obtener_Plan_Emulacion	Obtiene el Plan de Emulación.
Obtener_Punto	Obtiene un Punto de Emulación.
Crear_Vista	Crea una vista con los resultados de la Emulación.
Adicionar_Pto	Adiciona un Punto al Plan de Emulación
Buscar_Pto	Busca un Punto en el Plan de Emulación
Cambiar_Pto	Cambia un Punto del Plan de Emulación
Eliminar_Pto	Elimina un Punto del Plan de Emulación
Actualizar_Resultados	Actualiza los Resultados de un Punto de Emulación.
Obtener_Resultados	Obtiene los Resultados de un Punto para generar la Vista.

Nombre: NotificacionBean	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
EnviarNotificacion	Envía la Notificación creada por el sistema.

Nombre: ListaActividadesBean	
Tipo de clase: Interfaz	
Atributo	**Tipo**

ListaActividades	List
Para cada responsabilidad:	
Nombre:	**Descripción:**
getListaActividades	Obtiene el listado de Actividades del Calendario
setListaActividades	Cambia el Listado de Actividades del Calendario
MostrarCalendario	Muestra el Calendario de Actividades

Nombre: ActividadBean	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
AdicionarActividad	Adiciona una Actividad al calendario
EliminarActividad	Elimina una Actividad del Calendario
ModificarActividad	Cambia una Actividad del Claendario
getActividad	Obtiene una Actividad del Calendario

Nombre: ResultadosBean	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
ModificarResultados	Modifica los resultados de un Punto de Emulación.

Nombre: Punto_EmulacionBean	
Tipo de clase: Interfaz	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**

ModificarPunto	Cambia un Punto de Emulación.
AdicionarPunto	Adiciona un Punto de Emulación
EliminarPunto	Elimina un Punto de Emulación
getPunto	Obtiene una Punto de Emulación.

Nombre: Plan_EmulacionBean	
Tipo de clase: Interfaz	
Atributo	**Tipo**
ListaPuntos	List
Para cada responsabilidad:	
Nombre:	**Descripción:**
getListaPuntos	Obtiene la lista de Puntos del Plan de Emulación.
setListaPuntos	Cambia la lista de Puntos del Plan de Emulación.
MostrarResultados	Muestra los Resultados de la Emulación.

Diseño de la BD

Ver Diagrama Entidad Relación de la BD en el anexo 3.

Descripción de las tablas.

Nombre: Calendario		
Descripción: La función principal de la tabla Calendario es su relación con la Tabla Actividad.		
Atributo	**Tipo**	**Descripción**
Id_Cal	Interger(10)	Llave de la tabla.

Nombre: Actividad		
Descripción: La tabla Actividad almacena todos los datos de una actividad como nombre, descripción y destinatarios.		
Atributo	**Tipo**	**Descripción**
Id_Act	Interger(10)	Llave de la tabla.
CalendarioId_Cal	Interger(10)	Llave foránea
Fecha	Date	Fecha a celebrarse la actividad.
Nombre	Varchar(255)	Nombre de la Actividad
Descripcion	Varchar(255)	Breve descripción de la Actividad

Nombre: Destinatarios		
Descripción: La tabla destinatarios comprende tanto los Responsables como los Citados de una Actividad.		
Atributo	**Tipo**	**Descripción**
Id_Dest	Interger(10)	Llave de la tabla.
ActividadId_Act	Interger(10)	Llave foránea.
TipoDestinatario	bit	Diferencia entre Responsables (1) y citados (0)
Nombre	Varchar(255)	Nombre del destinatario.
Email	Varchar(255)	Dirección de correo electrónico del destinatario.

Nombre: Notificación		
Descripción: Las Notificaciones se generan a Partir de cada una de las actividades que llegan a una fecha tope.		
Atributo	**Tipo**	**Descripción**
ID_Not	Interger(10)	Llave de la tabla.
ActividadId_Act	Interger(10)	Llave foránea

Nombre: Plan_Emulacion		
Descripción: : La función principal de la tabla Plan_Emulacion es su relación con la Tabla Punto_Emulación.		
Atributo	**Tipo**	**Descripción**
Id_Plan	Interger(10)	Llave de la tabla.

Nombre: Punto_Emulacion		
Descripción: (2)		
Atributo	**Tipo**	**Descripción**
ID_Pto	Interger(10)	Llave de la tabla.
Plan_EmulacionId_Plan	Interger(10)	Llave foránea
Nombre	Varchar(255)	Nombre del Punto a Emular
Valor_Max	Interger(3)	Valor máximo posible en un punto de Emulación

Nombre: Resultados		
Descripción: Actualmente al existir 10 facultades en la UCI cada punto de Emulación Guardia un grupo de 10 resultados diferentes cada uno correspondiente a una de estas facultades.		
Atributo	**Tipo**	**Descripción**
ID_Res	Interger(10)	Llave de la tabla.
Punto_EmulacionID_Pto	Interger(10)	Llave foránea
Resultado	Interger(3)	Resultado obtenido en determinado punto de la Emulación.

Estimación de Costo-Beneficio:

La Estimación por Puntos de Caso de Uso es un método de estimación de esfuerzo de un proyecto de desarrollo de software a partir de los casos de uso.

Puntos de Casos de Uso

El primer paso es el cálculo de puntos de casos de uso sin ajustar. Lo cual se hace a partir de la siguiente ecuación:

UUCP = UAW + UUCW donde:

UUCP: Puntos de casos de uso sin ajustar.

UAW: Factor de peso del los actores sin ajustar.

UUCW: Factor de peso de los casos de uso sin ajustar.

Cálculo de UAW

Actor del Sistema:	Tipo de Actor	Factor de Peso
Estudiante	Complejo	3
Dirigente	Complejo	3
Responsable de Plan de Trabajo	Complejo	3
Responsable de Emulación	Complejo	3

Cantidad de actores de tipo complejo: 4

UAW= Sumatoria de la multiplicación de la cantidad de actores de un tipo específico con su factor de peso.

UAW = 4 x 3

UAW = 12

Cálculo de UUCW

Éste se calcula por los casos de uso del sistema y su complejidad. La complejidad esta dada por la cantidad de secuencias atómicas que tenga el caso de uso.

Casos de Uso	**Tipo de Caso de Uso**	**Factor de Peso**
Mostrar Emulación	Simple	5
Mostrar Calendario	Simple	5
Notificar Actividades	Simple	5
Gestionar Resultados Emulación	Simple	5
Gestionar Emulación	Medio	10
Gestionar Plan de Trabajo	Medio	10

Cantidad de casos de uso simples: 4

Cantidad de casos de uso medios: 2

UUCW = Sumatoria de la multiplicación de la cantidad de Casos de Uso de un tipo específico con su factor de peso.

UUCW = ((4 x 5) + (2 x 10))

UUCW = 40

Al ser **UAW =** 12 y **UUCW =** 40

UUCP = UAW + UUCW sustituyendo tenemos:

UUCP = 12 + 40

UUCP = 52

Después de haber calculado los puntos de Casos de Uso sin ajustar se calculan los puntos de Casos de Uso ajustados con la siguiente ecuación:

UCP = UUCP x TCF x EF donde:

UCP: Puntos de Casos de Uso ajustados.

UUCP: Puntos de Casos de Uso sin ajustar.

TCF: Factor de complejidad técnica.

EF: Factor de ambiente.

Cálculo del Factor de Complejidad Técnica (TCF).

Se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante

Factor	Descripción	Peso	Valor asignado
T1	Sistema distribuido	2	0
T2	Objetivos de performance o tiempo de respuesta	1	3
T3	Eficiencia del usuario final	1	3
T4	Procesamiento interno complejo	1	2
T5	El código debe ser reutilizable	1	4
T6	Facilidad de instalación	0.5	0
T7	Facilidad de uso	0.5	5
T8	Portabilidad	2	2
T9	Facilidad de cambio	1	4
T10	Concurrencia	1	2
T11	Incluye objetivos especiales de seguridad	1	1
T12	Provee acceso directo a terceras partes	1	0
T13	Se requieren facilidades especiales de entrenamiento	1	0

El Factor de Complejidad Técnica se calcula mediante la siguiente ecuación:

TCF = 0.6 + 0.01 x Σ (Peso i x Valor asignado i)

TCF = 0.6 +0.01 x Σ ((2 x 0) + (1 x 3) + (1 x 3) + (1 x 2) + (1 x 4) + (0.5 x 0) + (0.5 x5) + (2 x2) + (1 x 4) + (1 x 2) + (1 x 1) + (1 x 0) + (1 x 0))

TCF = 0.6 + (0.01 x 23.5)

TCF = 0.6+0.235

TCF = 0.835

Cálculo del Factor Ambiente (EF).

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos actores son los que se contemplan en el cálculo del Factor de ambiente.

Factor	Descripción	Peso	Valor asignado
E1	Familiaridad con el modelo de proyecto utilizado	1.5	5
E2	Experiencia en la aplicación	0.5	3
E3	Experiencia en trabajo orientado a objetos	1	5
E4	Capacidad del analista líder	0.5	4
E5	Motivación	1	4
E6	Estabilidad de los requerimientos	2	3
E7	Personal a tiempo completo	-1	2
E8	Dificultad del lenguaje de programación	-1	1

El Factor de ambiente se calcula mediante la siguiente ecuación:

EF = 1.4 - 0.03 x Σ (Peso i x Valor asignado i)

EF = 1.4 - 0.03 x Σ ((1.5 x5) + (0.5 x 3) + (1 x 5) + (0.5 x 4) + (1 x 4) + (2 x 3) + (-1 x 2) + (-1 x 1))

EF = 1.4 - (0.03 x 23)

EF = 0.71

Al ser **UUCP = 58**, **TCF = 0.715** y **EF = 0.71**

UCP = UUCP x TCF x EF sustituyendo tenemos:

UCP = 52 x 0.835 x 0.71

UCP = 30,8282

El esfuerzo en horas-hombre viene dado por:

E = UCP x CF donde:

E: Esfuerzo

UCP: Puntos de Casos de Uso ajustados (calculado anteriormente).

CF: Factor de conversión (para este tipo de proyecto 20 horas-hombre/Punto de Casos de Uso).

Al tener **UCP =** 30,8282 y **CF** = 20 horas-hombre/Punto de Casos de Uso

E = UCP x CF sustituyendo tenemos:

E = 30,8282 x 20 horas-hombre/Punto de Casos de Uso

E = 616,564 horas-hombre

Actividad	Porcentaje	Esfuerzo
Análisis	15	264,241714
Diseño	25	440,402857
Implementación	35	616,564
Prueba	15	264,241714
Sobrecarga	10	176,161143
Total	100%	1761,61143

El proyecto requerirá de 616,564 horas-hombre a desarrollar por una persona en un tiempo de 270 días; tomando que se trabajará en el proyecto como promedio 8 horas diarias. Por lo cual se puede decir que se terminará el proyecto en aproximadamente 9 meses ajustándose a los requerimientos del cliente.

Beneficios:

El presente trabajo, como parte de las aplicaciones que se desarrollan con el objetivo de ser usadas dentro de la Universidad de las Ciencias Informáticas, no reporta un beneficio monetario directo. Sin embargo; desde el punto de vista de los beneficios intangibles que reporta se puede decir, que con un sistema para la gestión del plan de trabajo y la emulación de la FEU en la UCI, se mejorará el manejo de la información relacionada con estas actividades.

Conclusiones: Capítulo 3

A lo largo de este capítulo hemos mostrado el desarrollo de la propuesta del sistema en los diferentes diagramas. Hemos visto cómo el sistema toma forma bajo el Flujo de Trabajo de Análisis y Diseño. Además se ha analizado la viabilidad del desarrollo de este módulo del proyecto.

CONCLUSIONES

Este trabajo demostró la necesidad del análisis y diseño de un sistema de Web que automatice el Plan de Trabajo y la Emulación de la FEU en la UCI. Entre los aspectos que se tuvieron en cuenta se encuentra:

- El estudio sobre las tendencias actuales que rodean el problema; y a través de estos conceptos y definiciones planteadas se fundamenta los principios de este trabajo y análisis completo de las tecnologías que serán utilizadas a lo largo del desarrollo del sistema propuesto.
- Se desarrolló la propuesta de solución y se pudo obtener un listado con las funciones que debe tener el sistema y se describieron paso a paso todas las acciones de los actores del sistema con los casos de uso con los que interactúan.
- Se relaciona los requisitos funcionales y los no funcionales, llevándolo a una expresión de la programación.
- Se realizó la estimación del costo del proyecto software, donde pudimos ver la duración y costo del proyecto.

Como resultado final se obtiene un modelo de análisis y diseño de un módulo del Proyecto Kainos, que es destinado para la construcción del portal Web de la FEU nacional.

RECOMENDACIONES

Recomendamos que ya una vez realizado el Análisis y el Diseño de nuestro módulo en Kainos, que se continúe el trabajo desarrollado hasta ahora con las fases siguientes, como son la construcción y transición, y poder brindar un producto final a la FEU de la UCI.

REFERENCIA BIBLIOGRÁFICA

1. BOOCH, G. “The Unified Modeling Language User Guide“. Addison-Wesley 1999. 482 p.
2. SANCHEZ, M. A. M. “Metodologías De Desarrollo De Software”. 2004, Disponible en: www.informatizate.net/articulos/pdfs/.
3. SHEN, D. Y. “Integración de JSF, Spring e Hibernate para crear una Aplicación Web del Mundo Real”. 2006, Disponible en: http://programacion.com/tutorial/jap_jsfwork/.
4. STALLMAN, R. M. “El movimiento del Software Libre y el sistema operativo GNU/Linux” En 2007.

BIBLIOGRAFÍA

1. GARCIA, R. M. M. Diseño de Bases de Datos. 1999.
2. GONZÁLEZ, H. S. Manual Hibernate. 2003, Disponible en: http://www.javahispano.org/articles.article.action?id=82.
3. GRADY BOOCH, I. J., JAMES RUMBAUGH. El proceso unificado del desarrollo de software 1999, Disponible en: http://www.monografias.com/trabajos22/desarrollo-software/desarrollo-software.shtml.
4. LARMAN, C. La Captura de Requerimientos. Venezuela: Universidad Simón Bolívar, Departamento de Computación y Tecnología de la Información, [Consultado el: 05/04 de 2007]. Disponible en: http://www.ldc.usb.ve/~teruel/ci4712/clases/clase2.html.
5. UML y Patrones. Introducción al análisis y diseño orientado a objetos. Segunda ed. 2002. 499 p.
6. MOLPECERES, A. Procesos de desarrollo: RUP, XP y FDD. 2002, Disponible en: http://www.javahispano.org/articles.article.action?id=76.
7. PULIDO, A. S. Introducción al UML. 2001, Disponible en: http://www.yoprogramo.com/articulo4.php.
8. SMALDONE, J. Software Libre versus Software Propietario [Consultado el: 08/12 de 2006]. Disponible en: http://www.smaldone.com.ar/opinion/slvssp.shtml.
9. VILAS, A. F. Diagramas de Casos de Uso [Consultado el: 24/02 de 2007]. Disponible en: http://www.javahispano.org/articles.article.action?id=82.
10. YOEMNY GONZALEZ, A. Y. P. Sistema para la Informatización de los Servicios Médicos de la UCI. Computación. Universidad de la Habana, 2004.

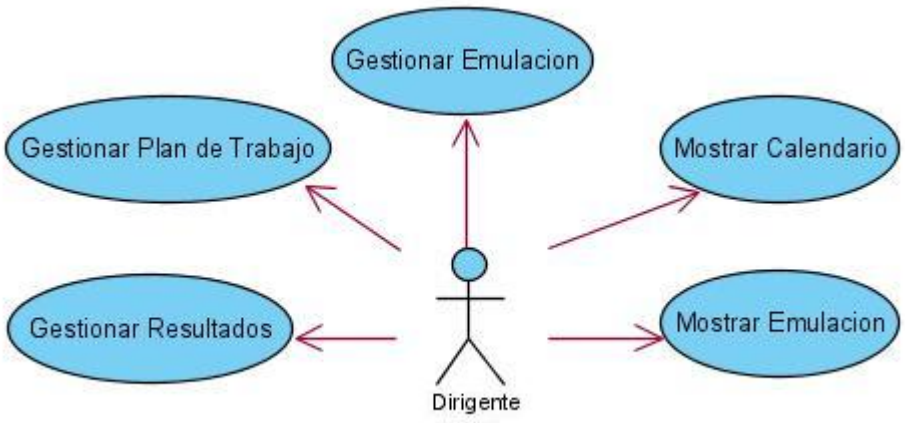
ANEXOS

Anexo 1

Modelo del negocio:

1.1 Diagrama de casos de uso del negocio:

Diagrama de casos de uso del negocio



1.2 Diagramas de Actividades:

Diagrama de actividad: Mostrar Emulación

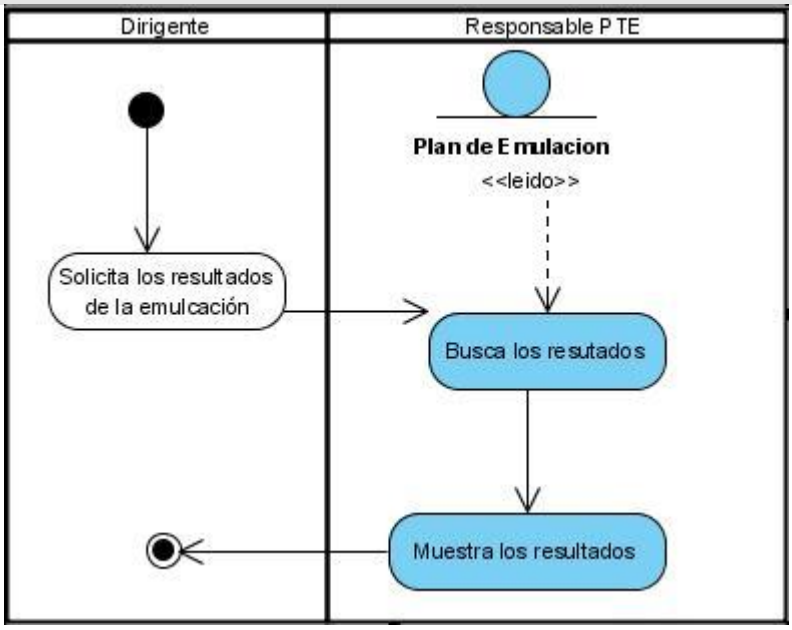


Diagrama de actividad: Gestionar Emulación

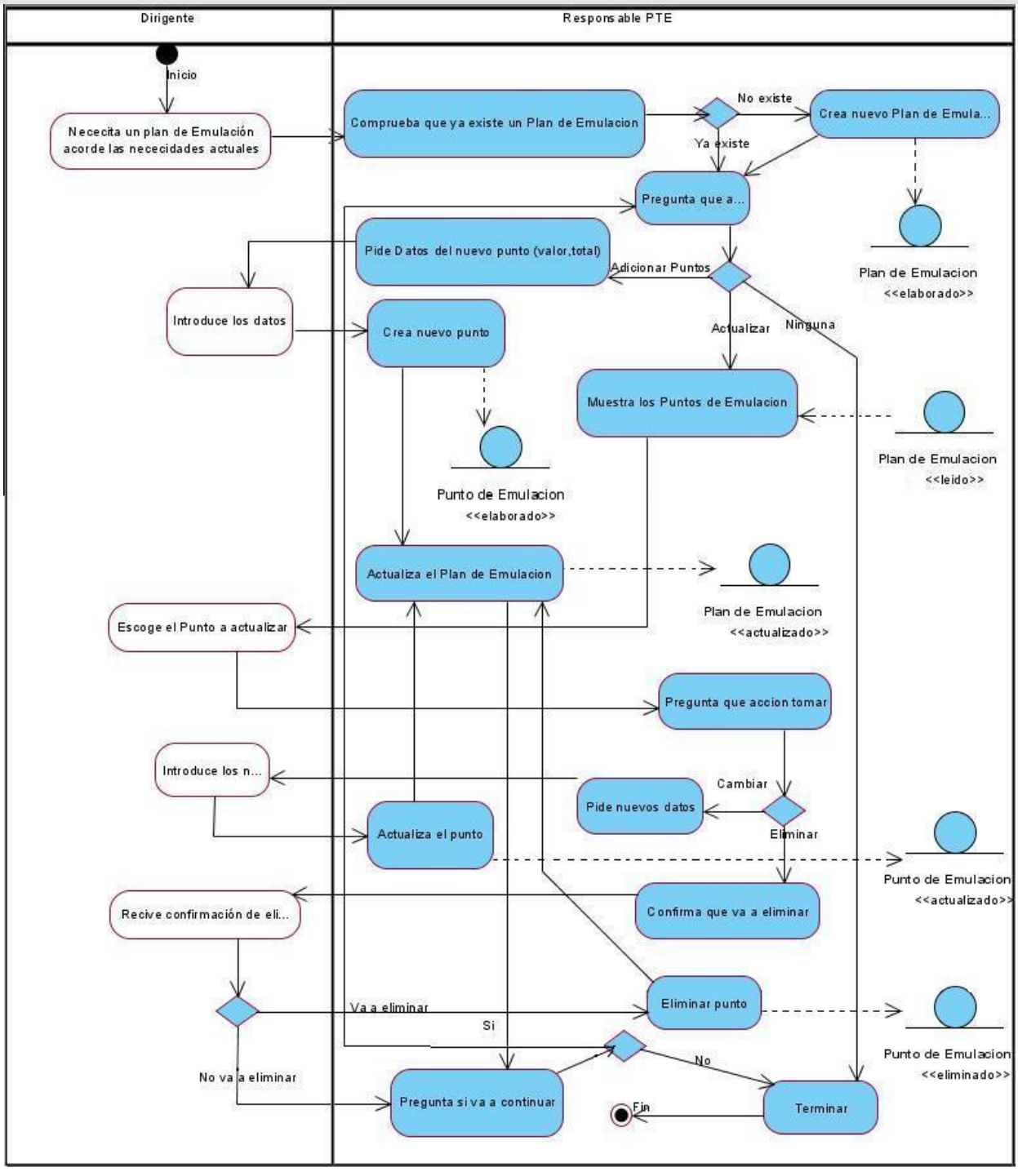


Diagrama de actividad: Gestionar Resultados de la Emulación

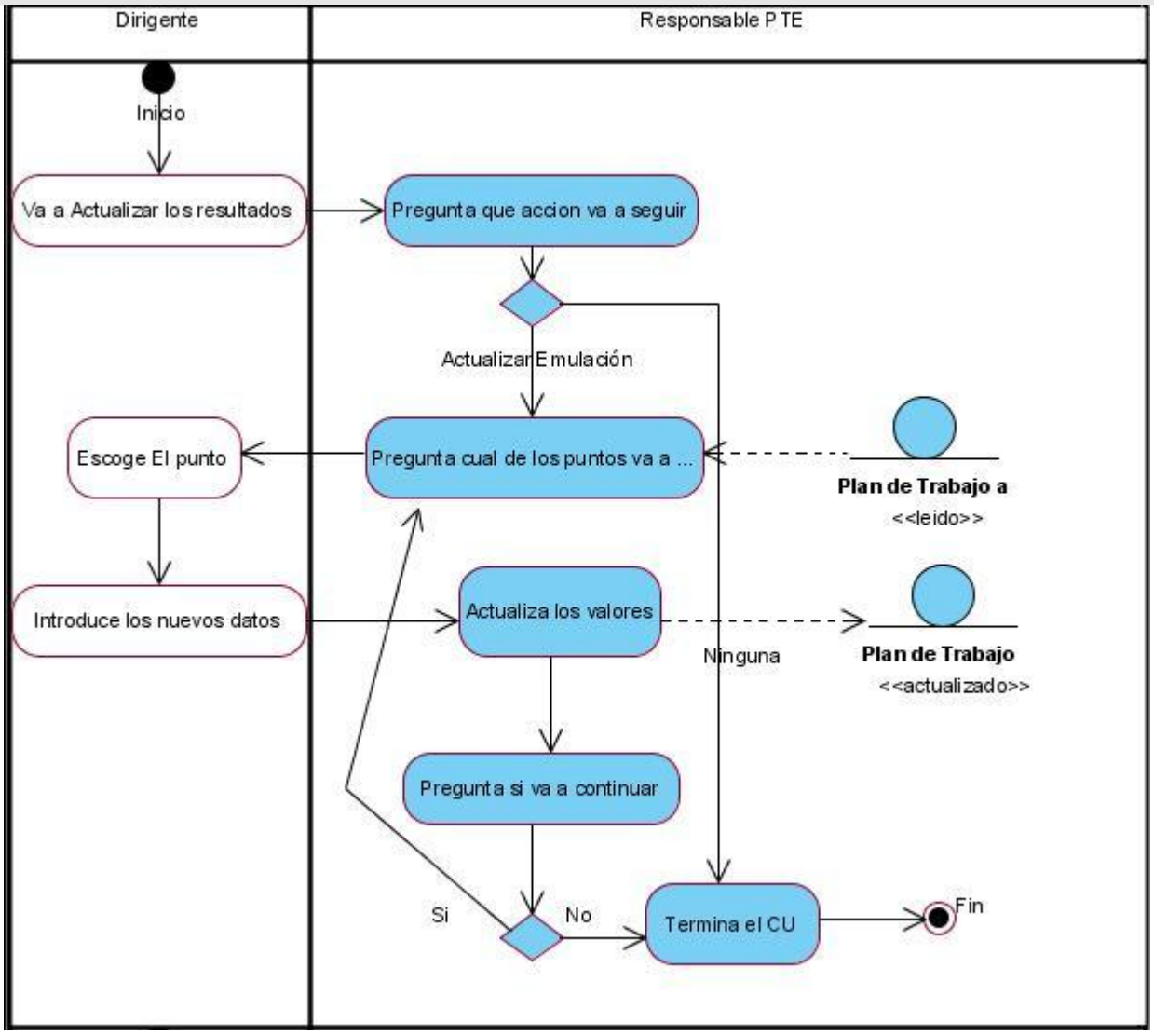
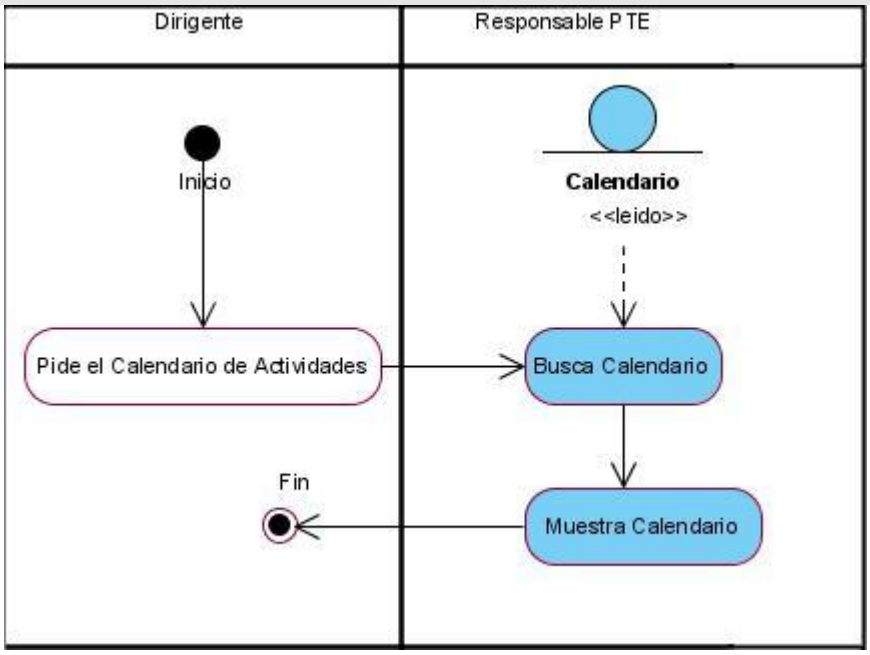


Diagrama de actividad: Mostrar Calendario



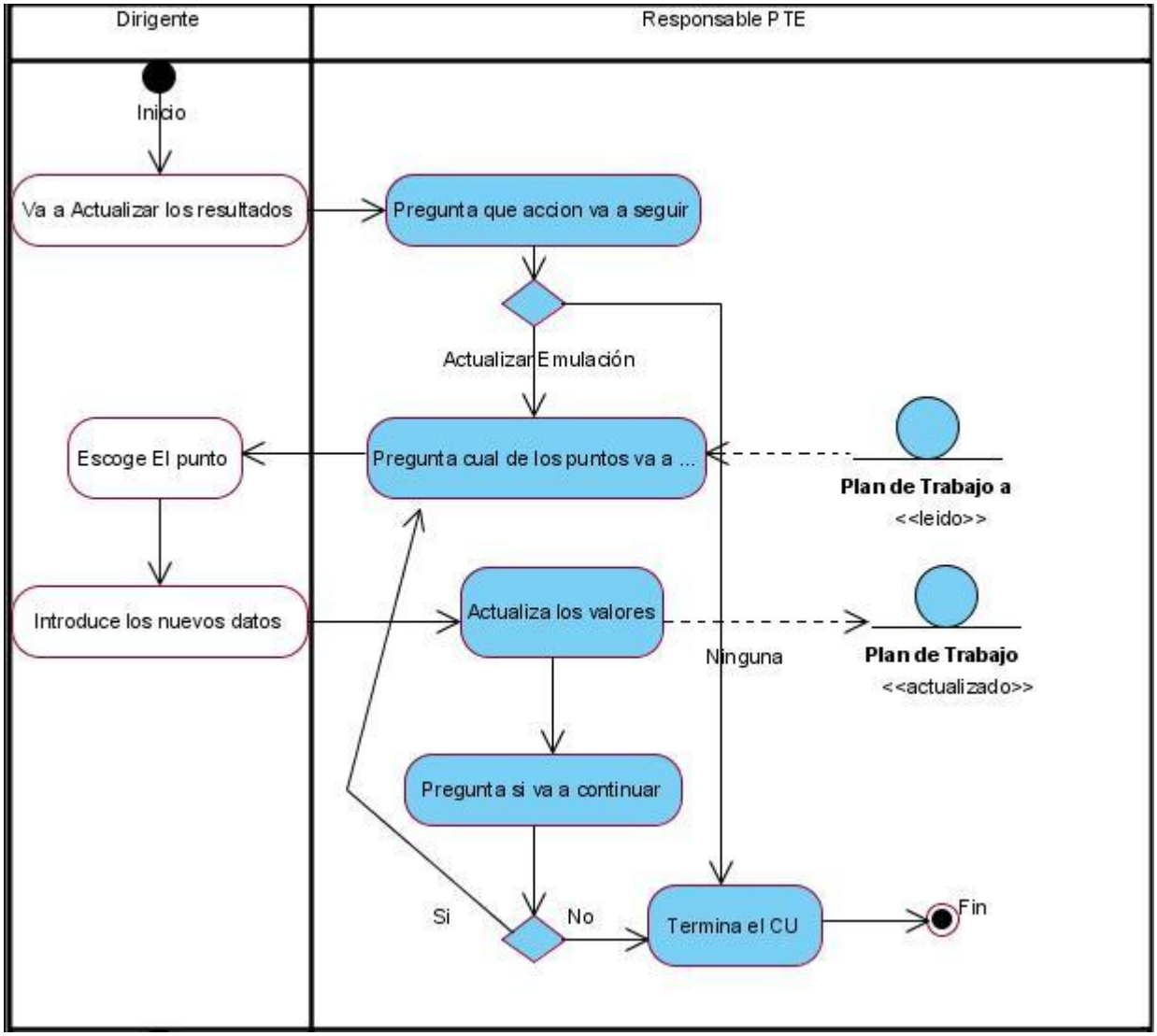
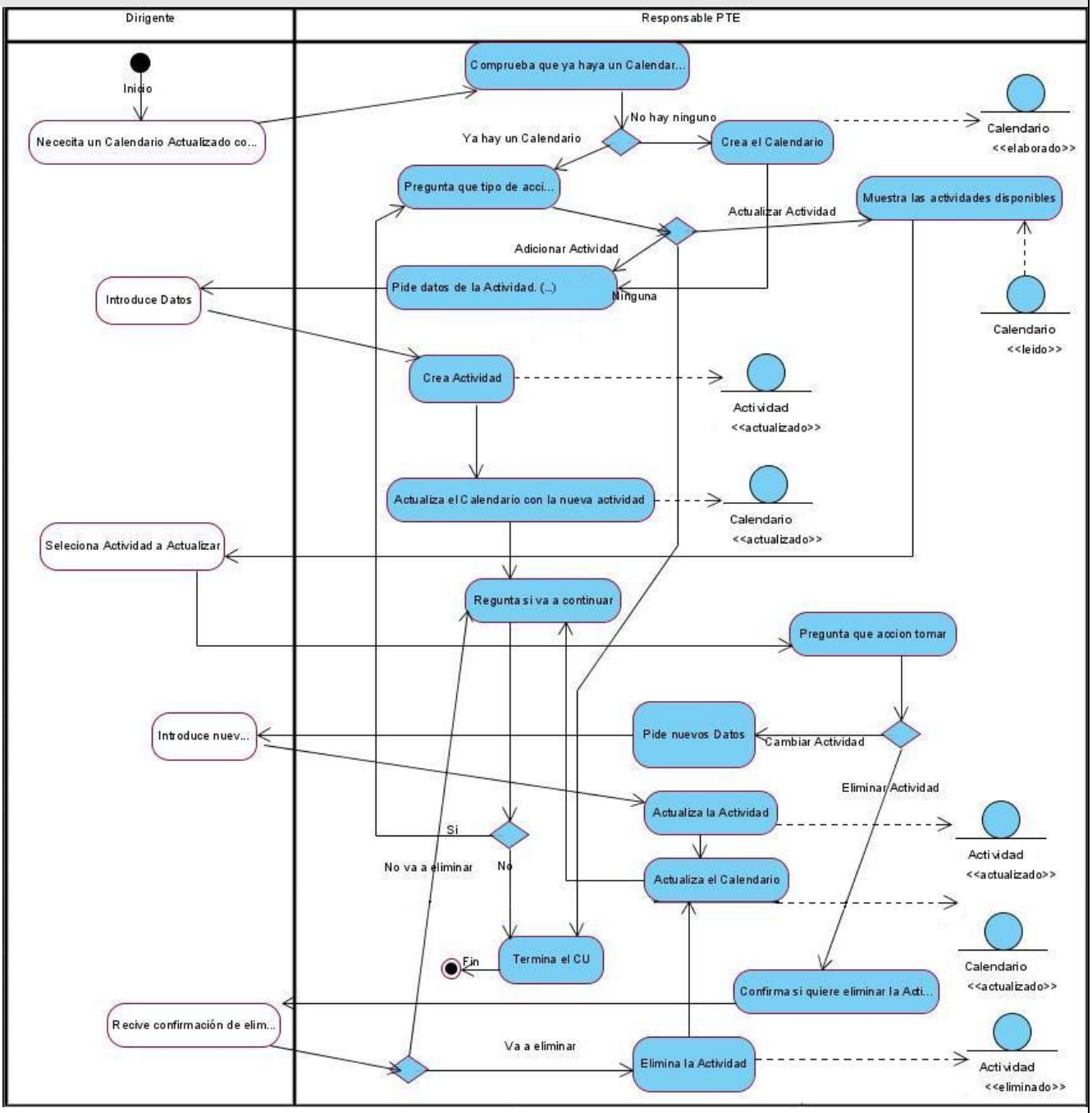
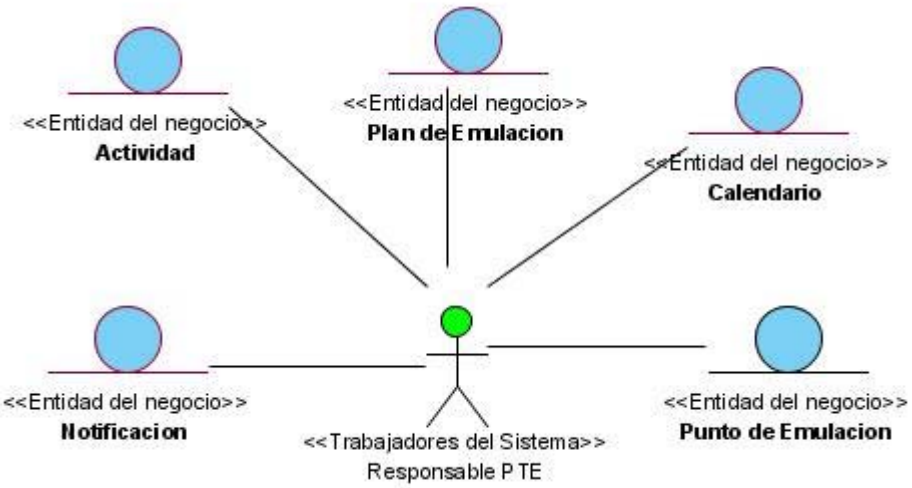

Dirigente
Responsable P TE
Inicio
Va a Actualizar los resultados
Pregunta que accion va a seguir
Actualizar Emulación
Escoge El punto
Pregunta cual de los puntos va a ...
Plan de Trabajo a
<<leido>>
Introduce los nuevos datos
Actualiza los valores
Ninguna
Plan de Trabajo
<<actualizado>>
Pregunta si va a continuar
Si
No
Termina el CU
Fin

Diagrama de actividad: Gestión del Plan de Trabajo



1.3 Modelo de Objetos:

DIAGRAMA DE CLASES DEL MODELO DE OBJETOS



1.4 Diagrama de casos de uso del negocio

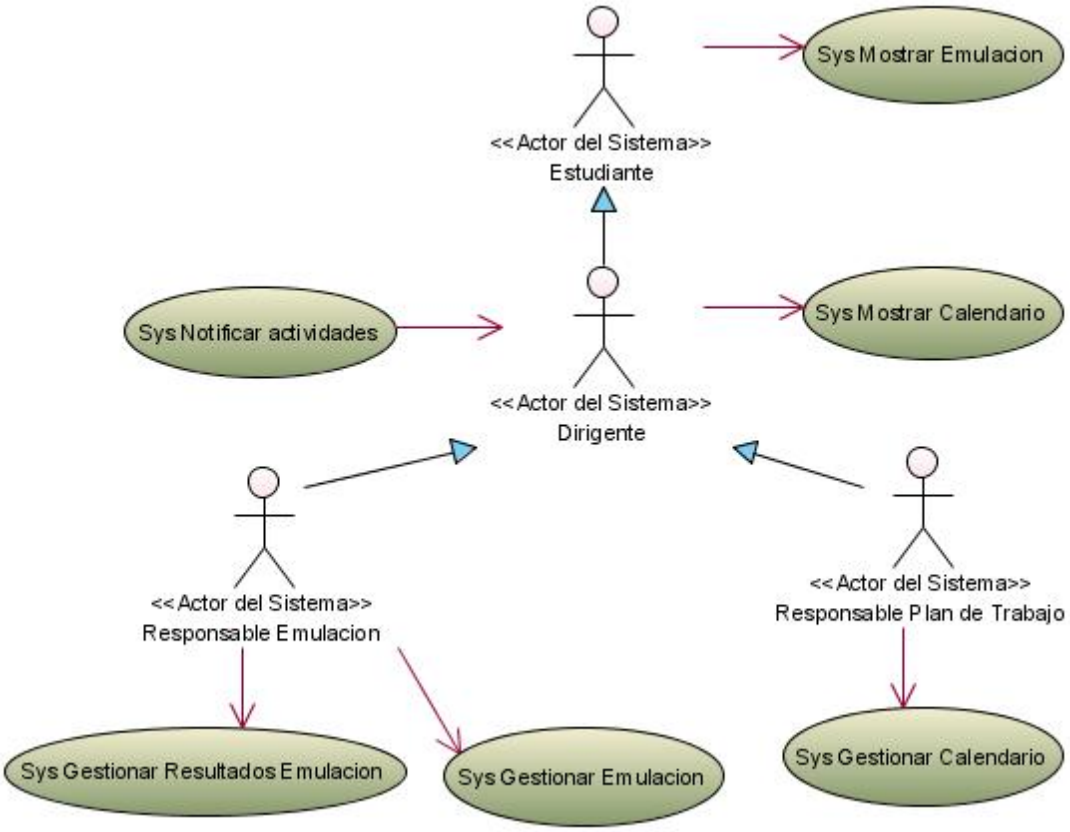


Diagrama de Casos de Uso del Sistema

Anexo 2.

Modelo del Análisis:

2.1 Diagramas de Clases de Análisis:

DIAGRAMA DE CLASES DE ANÁLISIS

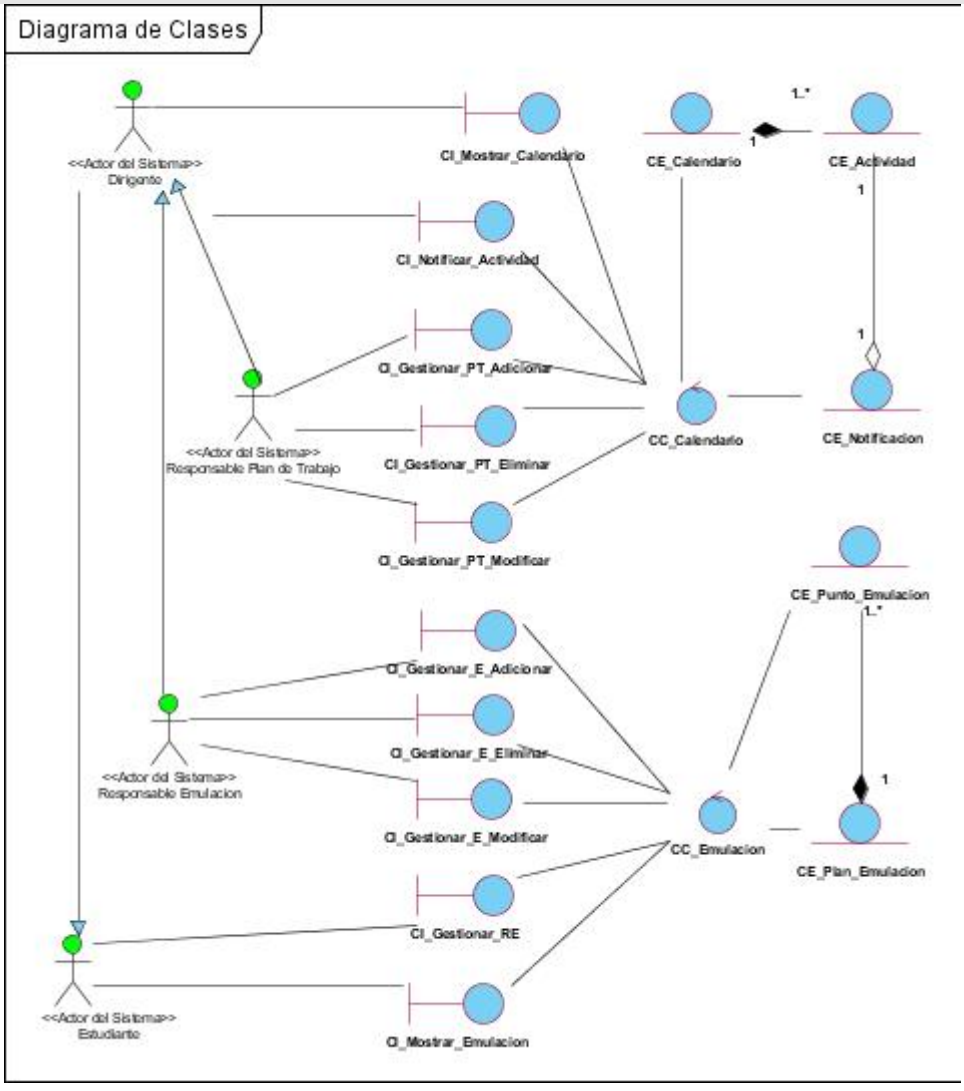


DIAGRAMA DE CLASES DE ANÁLISIS

Paquete Emulación

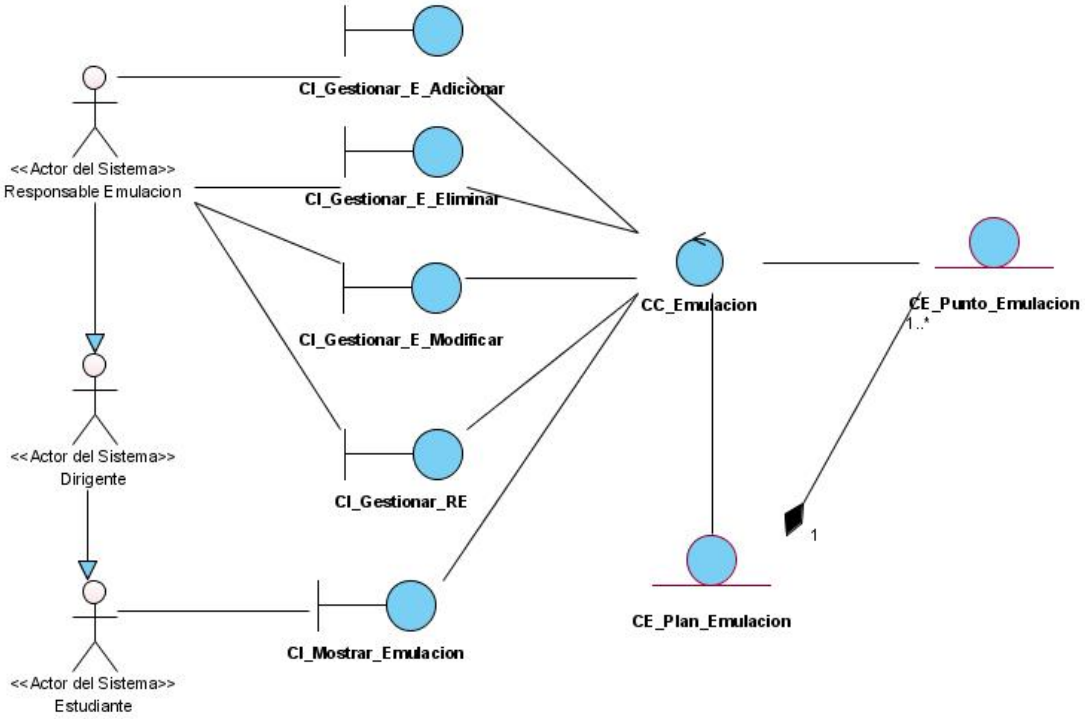
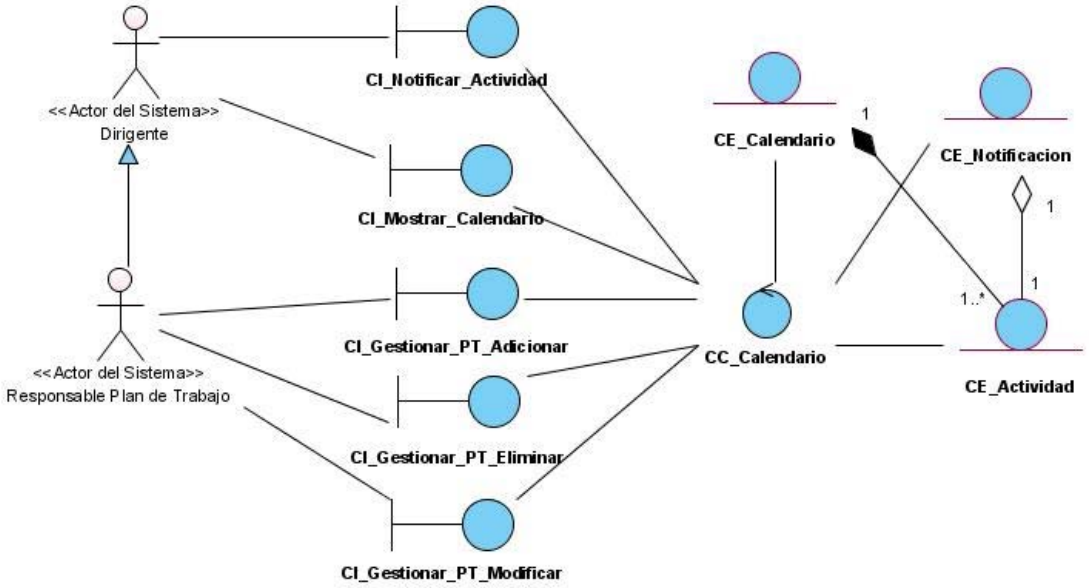


DIAGRAMA DE CLASES DE ANÁLISIS

Paquete Plan de Trabajo



Anexo 3.

Modelo del Diseño.

3.1 Diagramas Secuencia del Diseño:

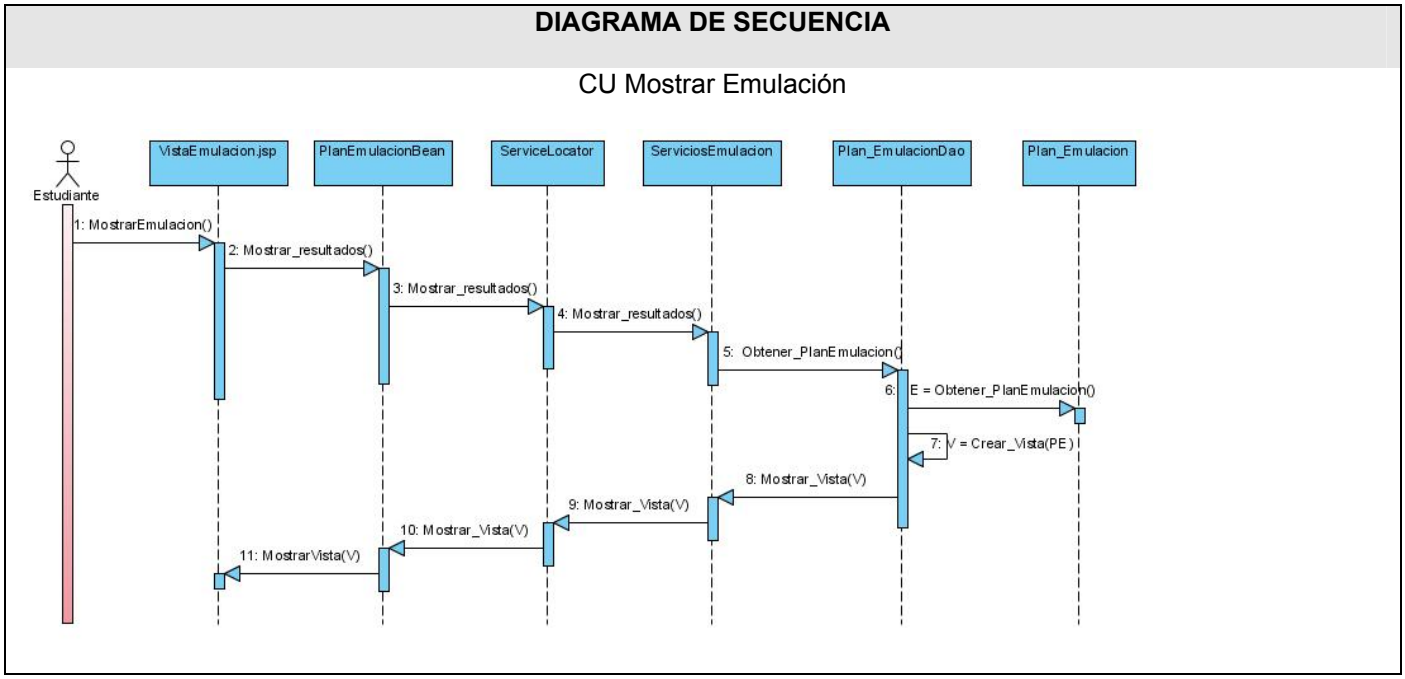


DIAGRAMA DE SECUENCIA

CU Mostrar Calendario

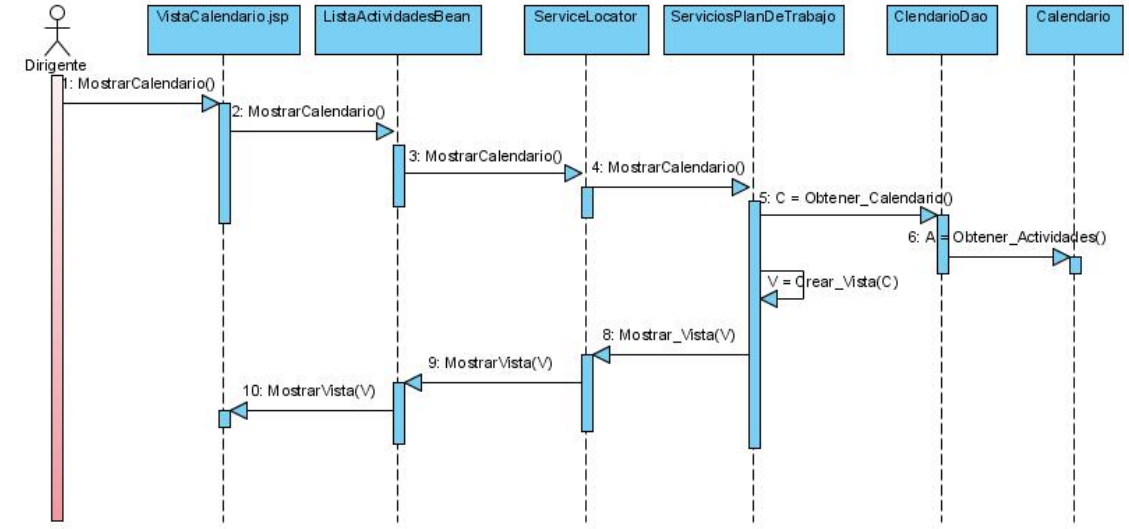


DIAGRAMA DE SECUENCIA

CU Notificar Actividades

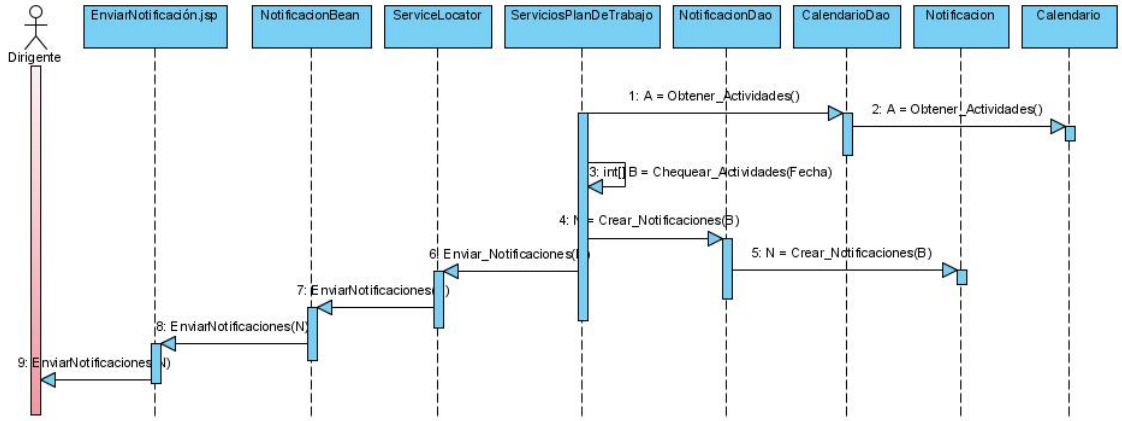


DIAGRAMA DE SECUENCIA

CU Gestionar Resultados de Emulación

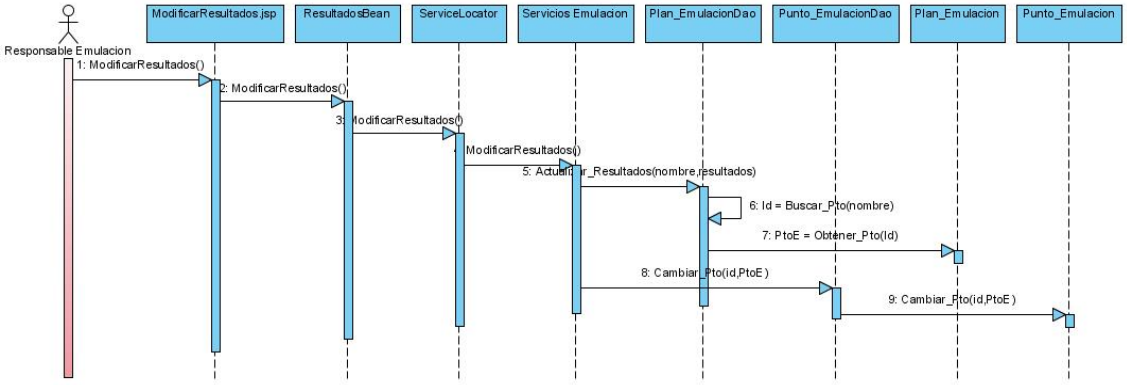


DIAGRAMA DE SECUENCIA

CU Gestionar Emulación (Adicionar)

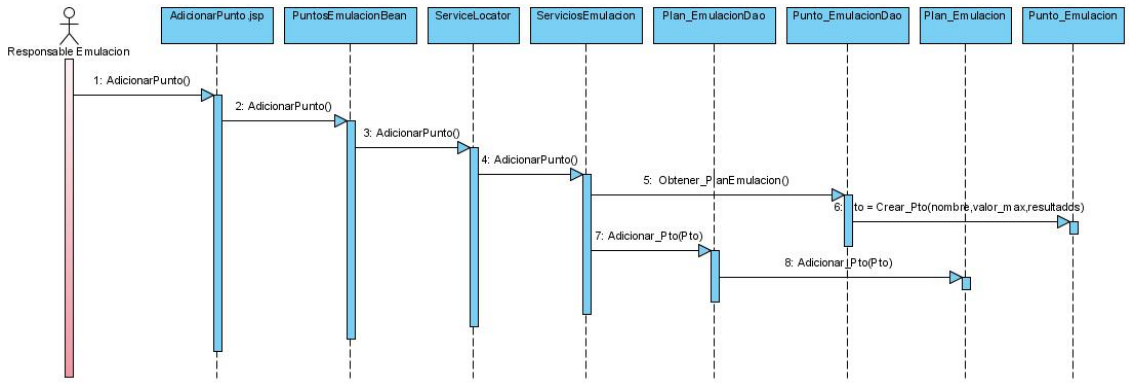


DIAGRAMA DE SECUENCIA

CU Gestionar Emulación (Eliminar)

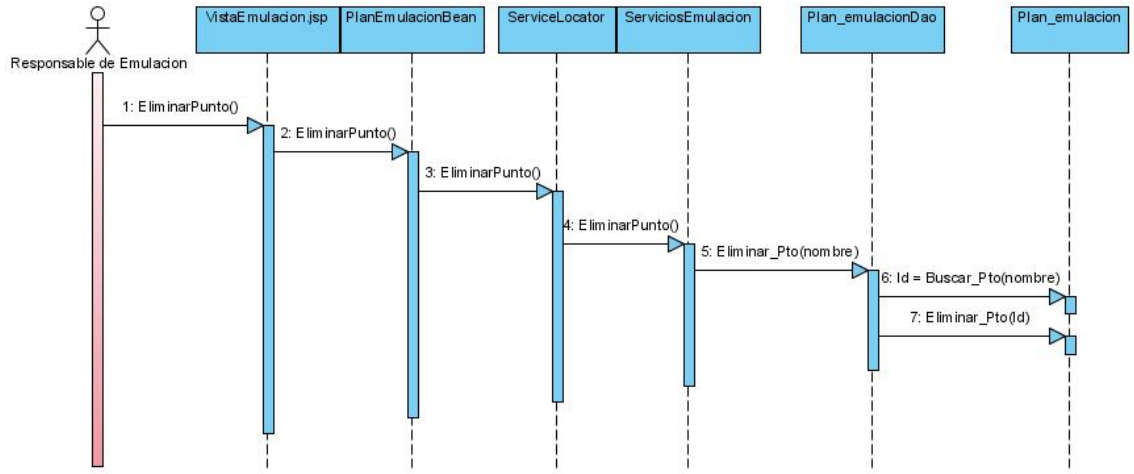


DIAGRAMA DE SECUENCIA

CU Gestionar Emulación (Modificar)

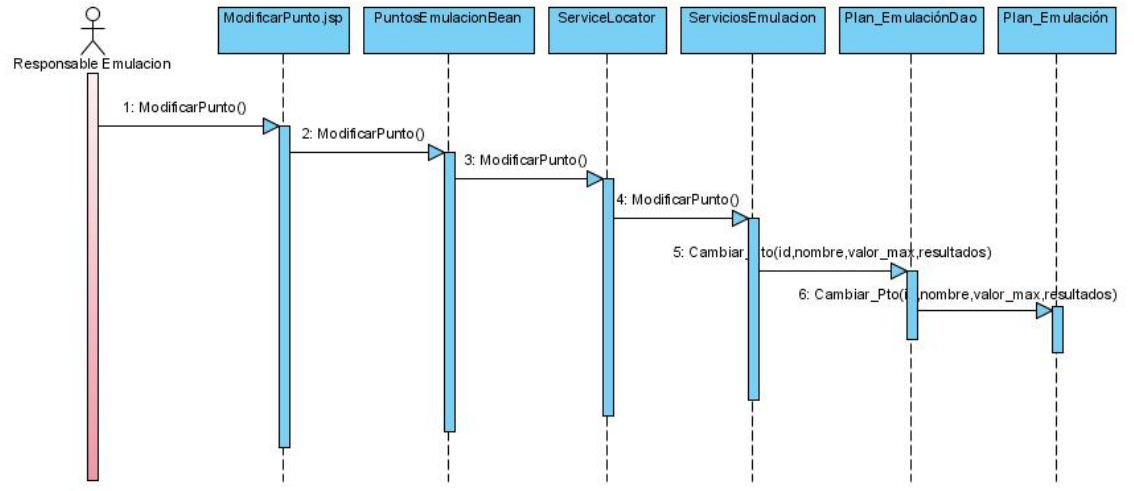


DIAGRAMA DE SECUENCIA

CU Gestionar Plan de Trabajo (Adicionar)

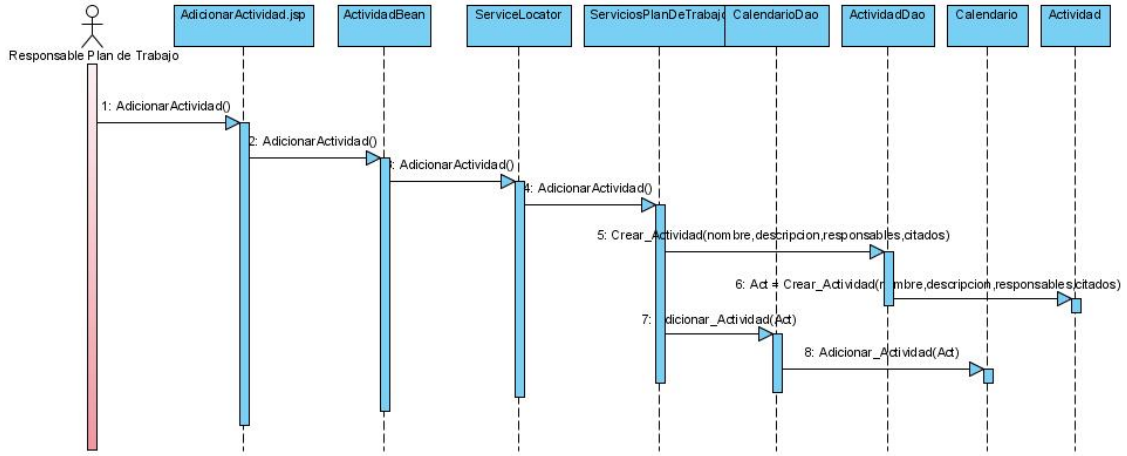


DIAGRAMA DE SECUENCIA

CU Gestionar Plan de Trabajo (Eliminar)

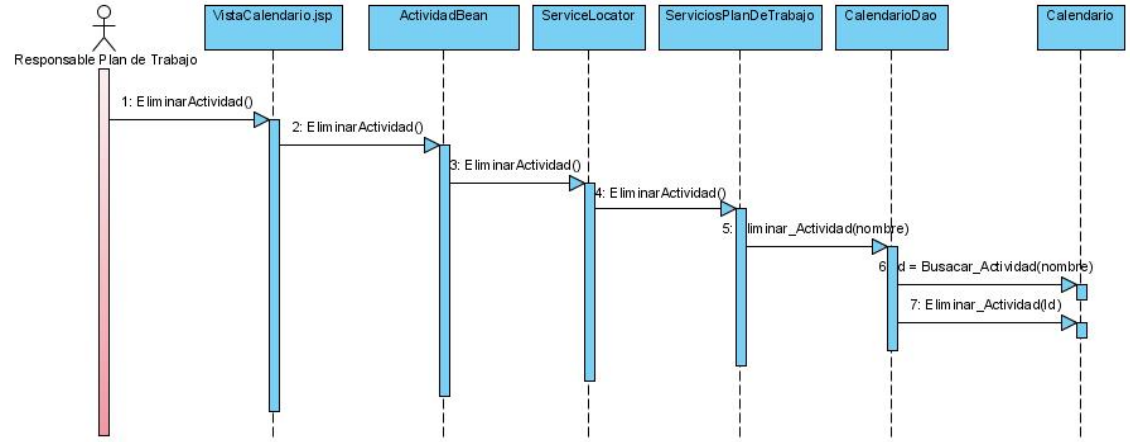
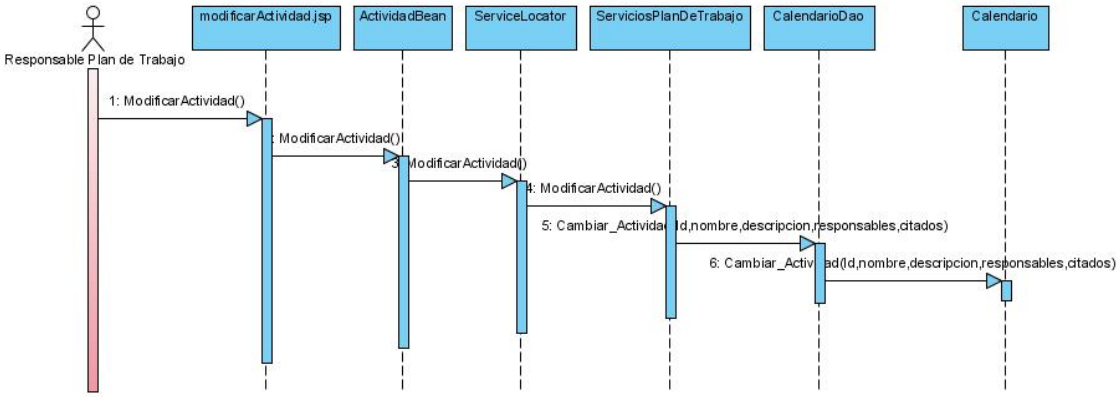
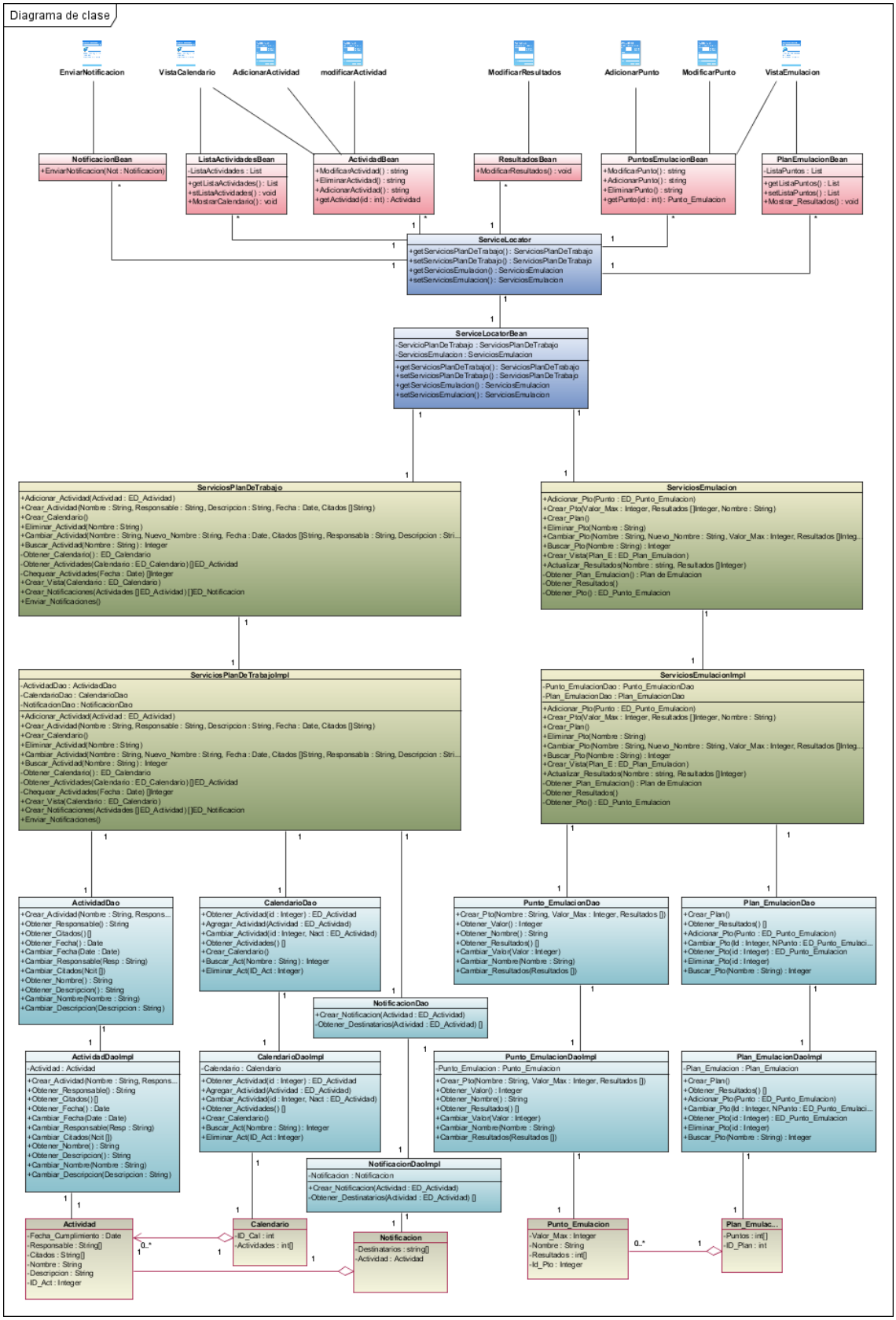


DIAGRAMA DE SECUENCIA

CU Gestionar Plan de Trabajo (Modificar)

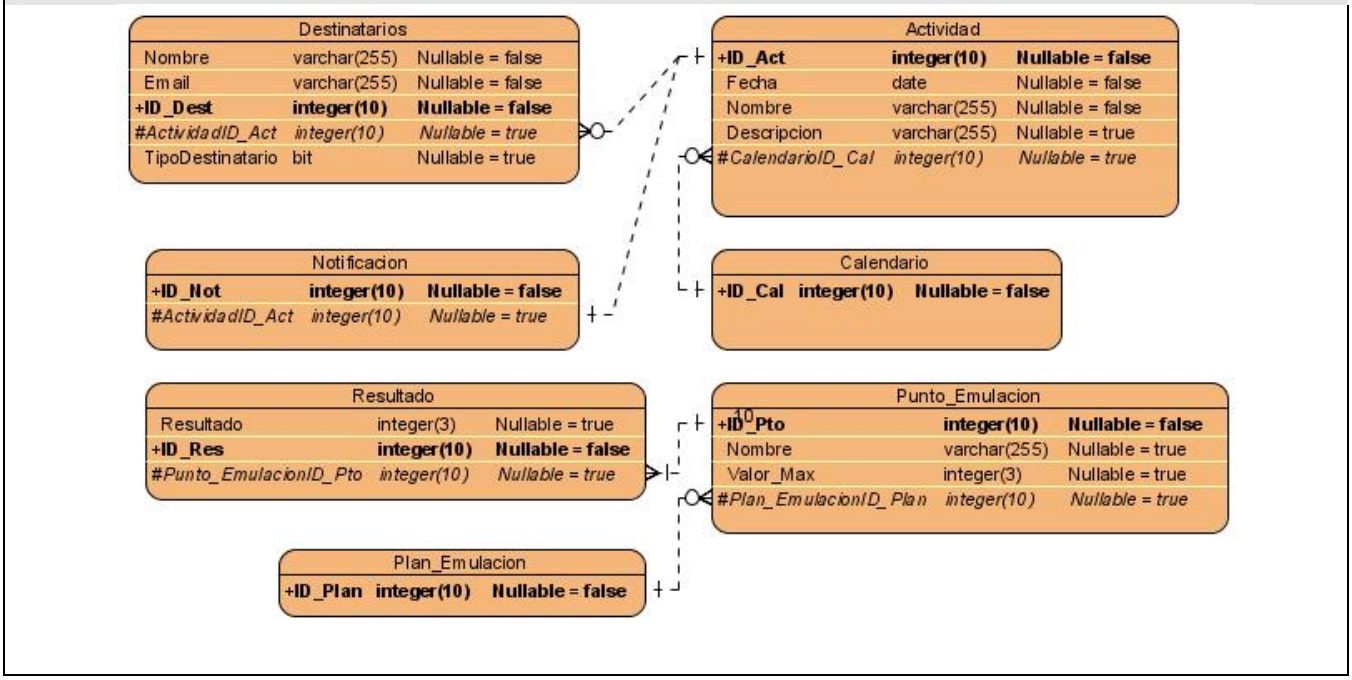


3.2 Diagrama de Clases del Diseño.



3.3 Diagrama Entidad-Relación.

DIAGRAMA ENTIDAD RELACION



GLOSARIO DE TÉRMINOS:

API: Una API (Del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones, interfaz de programación de la aplicación) es un conjunto de especificaciones de comunicación entre componentes software. Representa un método para conseguir abstracción en la programación

BD: Data Base o Base de datos.

Cliente-Servidor: Cuando se menciona este término no se piensa en otra cosa más que en bases de datos, dado que generalmente (Y de manera incorrecta) este término se usa como sinónimo de esto. Este término, en su más amplia definición, se usa para describir una aplicación en la cual dos o más procesos separados trabajan juntos para completar una tarea. El proceso cliente solicita al proceso servidor la ejecución de alguna acción en particular. Esta operación se conoce como Proceso Cooperativo, dado que dos procesos separados cooperan para completar la tarea en particular.

Componentes UI: Componentes de interfaz de usuario.

CU: Caso de uso.

DAO: Clases de acceso a datos (DAO Data Access Object)

El e-Government: , e-gobierno o gobierno electrónico consiste en el uso de las tecnologías de la información y el conocimiento en los procesos internos de gobierno y en la entrega de los productos y servicios del Estado tanto a los ciudadanos como a la industria. Muchas de las tecnologías involucradas y sus implementaciones son las mismas o similares a aquéllas correspondientes al sector privado del comercio electrónico (o e-business), mientras que otras son específicas o únicas en relación a las necesidades del gobierno.

FEU: Federación estudiantil universitaria, organización de masas que representa a todos los universitarios cubanos.

Hibernate: es el puente entre nuestra aplicación y la BD, sus funciones van desde la ejecución de sentencias SQL hasta la creación, modificación y eliminación de objetos persistentes.

IDE: Entorno integrado de desarrollo (IDE siglas en inglés).

Información*:* Agregación de datos que tiene un significado específico más allá de cada uno de éstos. Un ejemplo: 1, 9, 8 y 7 son datos; 1987 es una información. La información ha sido siempre un recurso muy valioso, revalorizado hoy más aun por el desarrollo y la expansión de las Tecnologías de la Información y de las Comunicaciones.

Interface: Conexión entre dos dispositivos de hardware, entre dos aplicaciones o entre un usuario y una aplicación que facilita el intercambio de datos.

Internet: Sistema de redes de computación ligadas entre sí, con alcance mundial, que facilita servicios de comunicación de datos como registro remoto, transferencia de archivos, correo electrónico y grupos de noticias. Internet es una forma de conectar las redes de computación existentes que amplía en gran medida el alcance de cada sistema participante.

J2EE: Java 2 Enterprise Edition, es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Java.

Java: es un lenguaje orientado a objetos y desarrollado por Sun Microsystem. Comparte similitudes con C, C++ y Objetive C. Basándose en otros lenguajes orientados al objeto, Java recoge lo mejor de todos ellos y elimina sus puntos más conflictivos. El principal objetivo de JAVA fue hacer un lenguaje que fuera capaz de ser ejecutado de una forma segura a través de Internet (aunque el código fuera escrito de forma maliciosa). Esta característica requiere la eliminación de muchas construcciones y usos de C y C++. El más importante, es que no existen punteros. Java no puede acceder arbitrariamente a direcciones de memoria. Java es un lenguaje compilado en un código llamado "codigo-byte" (byte-code). Este código es interpretado por el intérprete Java, el cual Java fue diseñado también para escribir código libre de bugs, esto se consigue en gran parte, eliminando las operaciones de localización y deslocalización de memoria del lenguaje C. Java no es un lenguaje para ser usado solo en el WWW, pero su despegue y utilización se debe al World Wide Web. Hoy día casi todos los exploradores interpretan código Java.

JSF (Java Server Faces): es un framework de desarrollo basado en el patrón MVC (Modelo Vista Controlador).

JSP o Java Server Pages: es una tecnología Java que permite a los programadores generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo. Las JSP's permiten al código Java y a algunas acciones predefinidas ser incrustadas en el contenido estático del documento web. En las JSP se escribe el texto que va a ser devuelto en la salida (normalmente, código HTML) incluyendo código java dentro de él, para poder modificar o generar contenido dinámicamente.

MER: Modelo Entidad relación.

PostgreSQL: Motor de base de datos, servidor de base de datos relacional libre, liberado bajo una licencia de software libre con grandes ventajas y funcionalidades de mucho uso en el mundo actual de las ciencias informáticas.

Proceso: Un proceso puede ser definido como un conjunto de actividades interrelacionadas entre sí que, a partir de una o varias entradas de materiales o información, dan lugar a una o varias salidas también de materiales o información con valor añadido.

Red: Una red es dos o más computadoras que están físicamente conectadas con las otras y capaces de compartir información.

R-N: Referencia al caso de uso N.

Scripts: En inglés significa "guión". De hecho, el uso es exactamente éste: el navegador lee una línea, la interpreta y la ejecuta, después pasa a la sucesiva y hace lo mismo, y así hasta el cierre del script.

SGBD: Es el software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista a la misma vez.

Sistema: Conjunto de cosas que ordenadamente relacionadas entre sí contribuyen a determinado objeto.

Sitio Web: Sistema de computación que corre un servidor Web y que se ha establecido para editar documentos en Web.

Software:(Componentes lógicos, programas, software). -- Programas o elementos lógicos que hacen funcionar un ordenador o una red, o que se ejecutan en ellos, en contraposición con los componentes físicos del ordenador o la red.

Spring: es un marco de trabajo para aplicaciones J2EE que pretende reducir el esfuerzo de desarrollo para la construcción de aplicaciones.

Subversión: es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se lo conoce también como svn por ser ese el nombre de la herramienta de línea de comandos.

Tortoise: Es el cliente más recomendable para trabajar con el Subversion Tortoise svn. Este cliente convierte el mismo Explorer en un cliente Subversion, es realmente rápido y cómodo de usar.

UML y UML 2.0: El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación. La diferencia mas notable entre UML y UML 2.0 es que el primero es orientado a objetos y el segundo es orientada a procesos, lo cual muestra su principal ventaja en que en el mundo real la mayoría de las cosas son procesos.

URL: (Uniform Resource Locator) -- (Localizador Uniforme de Recursos) Sistema unificado de identificación de recursos en la red. Las direcciones se componen de protocolo, FQDN y dirección local del documento dentro del servidor. Este tipo de direcciones permite identificar objetos WWW, Gopher, FTP, News,... Ejemplos de URL son: http://cubasi.cu o ftp://ftp.ati.es.

WEB (WWW): Red de documentos HTML intercomunicados y distribuidos entre servidores del mundo entero.