UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
Facultad 9

TÍTULO: MaGIStral. Sistema Integral de Gestión de Recursos para el proyecto GENESIG.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTORES: Henry González Arias
Dayron Hernández Muñoz
TUTOR: Ing. Yoenis Pantoja Zaldívar

Ciudad de la Habana, 2010
Año del 51 Aniversario del Triunfo de la Revolución

DEDICATORIA

Henry González Arias

Quiero dedicar mi tesis a todas esas personas maravillosas que me quieren mucho y que sin ellas no hubiera sido posible disfrutar este sueño grandioso hecho realidad, en especial:

A mi mamá y a mi papá por haber estado ahí para mí desde que nací, compartiendo las alegrías y los momentos de vicisitudes, guiándome a ser una mejor persona antes que todo, brindándome todo su amor y sabiduría.

A mi hermano por ser mi amigo y por haberme ayudado desde que éramos chicos, y al que quiero muchísimo.
A mi hermanita que desde que vino al mundo fue siempre una alegría para mí.

A mi abuelita de oro Olga, que siempre ha estado pendiente de mis problemas brindándome todo lo que tiene para que esté bien, además de su amor, comprensión y sus valiosos consejos que tanto me han ayudado en la vida.
A mi otra abuelita de oro Teresa, que ya no está con nosotros físicamente, pero que sigue viva en mis pensamientos y que me dio mucho amor cuando era pequeño.
A mi tío Nené por ser como un padre para mí en los buenos y malos momentos.
A mi tía Yanet, que ha cuidado de mí cuando lo he necesitado.
A mi tío Nolberto por ser un ejemplo a seguir para mí.

A mi novia Darianis por compartir conmigo los tragos buenos y amargos de la vida desde que la conocí.

A Dayron por ser además de mi compañero de tesis, como un hermano para mí en todos los años de la universidad.
A Mandy, mi amigo entrañable desde primer año.

Dayron Hernández Muñoz

A mi abuela, a ella le dedico mi vida y con ello todos mis logros, se que ella me está observando desde donde pueda estar y hoy se siente orgullosa de mí, es mi inspiración, es mi vida, es lo más grande que tuve y lo que más necesito.

A mi madre, por sobre todas las cosas, va dedicado mi esfuerzo y todos mis años de estudio, por estar siempre a mi lado, presente en cada decisión y celebra mis triunfos como si fueran suyos.

A mi padre, porque es mi ejemplo a seguir, porque es la meta y siempre estoy y estaré orgulloso de él Y sé que él también lo está de mí; por su confianza, por su apoyo y porque ha estado ahí donde lo he necesitado

A mi hermano Dariel, por ser uno de los tesoros más grande que me ha dado la vida.

A mi familia, porque siempre han estado al tanto de mis estudios, mi futuro y mi vida.

A mis compañeros de estudio, y en especial a quienes la palabra amigo, le quedaría corta, Dayana, Liudmila, Mandy, Lilibet, Yake, el Mello, Nuris, Maceo, que han sido siempre incondicionales conmigo.

A Henry, que más que un amigo, es mi hermano, mi compañero de tesis, que sabe comprenderme y poner nuestra amistad por encima de todo.

A Darlin, a Lili, a Mónica, a Saili, Arián, Henry a todos, que no son pocos, los que de una forma u otra, también han sido parte de este logro.

AGRADECIMIENTOS

Henry González Arias

Un gran proyecto es imposible de realizar sin la colaboración de personas que brindan su apoyo emocional y profesional, y es por eso que esta tesis no habría sido posible sin la ayuda de:

Toda mi familia que me brindó su apoyo en todo momento.

Yoenis Pantoja Zaldívar, tutor del presente proyecto y que siempre estuvo a nuestra disposición.

Los compañeros del tribunal de tesis, lo cuales hicieron de este trabajo de diploma un mejor proyecto científico y profesional.

Rolando Toledo Fernández, amigo que ha respondido mis frecuentes preguntas sobre programación y otras cuestiones tecnológicas.

Todos mis amigos, en especial a Dayron y Mandy, los que me apoyaron en el desarrollo de la tesis.

Mis compañeros de aula de todos los cursos de la carrera, los cuales aportaron valiosas cualidades morales.

La Revolución Cubana, la cual me ha permitido llegar hasta esta altura y seguir superándome para obtener un futuro mejor.

Dayron Hernández Muñoz

A mis padres y mi hermano, por no dudar de mí en ningún momento, brindarme toda la fuerza que necesité para poder llegar hasta el final, apoyarme a pesar de todas las dificultades que se presentaron a lo largo de estos 5 años y del desarrollo de la tesis.

A mis abuelos, mis tíos y primos porque siempre han estado al tanto de mis estudios, mi futuro y mi vida.

A Lili por ser mi compañera todo este tiempo, por su cariño y comprensión, y sobre toda los cosas por tener fe en mí y aconsejarme en los momentos más críticos.

A mi tutor Yoenis por su disposición en todo momento y toda la ayuda que nos dio.

A las personas del tribunal por su paciencia y contribución para que hoy el presente trabajo de diploma cuente con la calidad requerida para su publicación.

A mis amistades que de una forma u otra me apoyaron y a mis compañeros de aula en estos años de carrera, a ustedes muchas gracias.

A todos muchas gracias

Dayron

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 9 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 2010.

_____	_____
AUTOR	AUTOR
Dayron Hernández Muñoz	**Henry González Arias**

_____

TUTOR

Ing. Yoenis Pantoja Zaldívar

RESUMEN

El proyecto GeneSIG tiene la misión de elaborar la Plataforma Soberana para el Desarrollo de Sistemas de Información Geográfica (GeneSIG) que se concibe como un producto soberano que brinde soporte al desarrollo de aplicaciones de Sistemas de Información Geográfica (SIG) en entornos Web con tecnologías libres. En este marco el Jefe de Proyecto tiene entre sus responsabilidades el control, organización y planificación de los recursos que lo componen. En este entorno se encuentran un grupo de deficiencias que dificultan el desarrollo de las actividades mencionadas; pues la información que se maneja: control del personal, control de recursos materiales con los que dispone, asignación de tareas, entre otras, se realiza de forma manual lo que hacen que la comunicación entre los miembros del Proyecto, no se efectúe de manera clara y con la rapidez que se requiere.

Para solucionar esta deficiencia es necesario realizar una aplicación que sea capaz de gestionar todos los recursos asociados al proyecto, permitiendo controlar, planificar y organizar el funcionamiento del mismo.

Con el presente trabajo de diploma se pretende realizar el análisis, diseño e implementación de una aplicación Web con la cual se pueda disponer de un sistema rápido, fiable, con alto grado de uniformidad y consistencia, permitiendo hacer un seguimiento a los recursos humanos y materiales a utilizar en el proyecto.

PALABRAS CLAVES

Herramientas de Gestión de Recursos, Recursos Humanos, Recursos Materiales

ÍNDICE

TABLAS Y FIGURAS

INTRODUCCIÓN

Una mala planeación y/o ejecución de un proyecto causa pérdidas relacionadas principalmente con el factor tiempo y dinero, razón por la cual este debe planearse y ejecutarse teniendo en cuenta que los proyectos se desarrollan para obtener una mejora significativa en la empresa cumpliendo con las expectativas de calidad, costo y tiempo. La Gestión de Proyectos está enfocada en optimizar las tareas de planificación y monitoreo de las actividades, recursos materiales y humanos que intervienen dentro de un proyecto.

Cuba se ha insertado de forma limitada en el mercado mundial en lo que respecta a herramientas de gestión de recursos a nivel empresarial o de manera específica, en proyectos de software. Actualmente en el país la producción de software es llevada a cabo por algunas universidades y empresas, la cuales generalmente han desarrollado productos para beneficio propio y muy pocos para uso general, tal es el caso de la empresa DESOFT.
La Universidad de las Ciencias Informáticas (UCI) a pesar de contar con pocos años de experiencia es una de las mayores instituciones productoras de software del país. Bajo la premisa de estudiar y producir, los estudiantes que se forman en ella, desde segundo año ya están vinculados a proyectos productivos que de una forma u otra contribuyen a la economía y sirven de base en la formación de profesionales con grandes conocimientos en áreas de la Informática. Sin embargo, la informatización de procesos es aún un área incipiente en el marco universitario a pesar de que se ha trabajado en este sentido no sólo de manera interna sino también a partir de algunos acuerdos con empresas extranjeras.

La UCI cuenta con 337 registros de tesis, disponibles en la biblioteca, que están vinculados con el desarrollo de herramientas de gestión de recursos en proyectos o con la investigación de esta rama, ejemplo de éstas son:

- Sistema de Gestión de Información de los Recursos Humanos de la Facultad 2.
- Análisis y diseño de un sistema automatizado para el control de los recursos humanos en los polos productivos de la facultad 9.

- Sistema para la Gestión de Recursos Informáticos en la Universidad de las Ciencias Informáticas. Facultad 4.

El Proyecto GeneSIG, perteneciente al Polo Productivo Geoinformática de la Facultad 9 surge como necesidad de contar con una línea de trabajo enfocada al desarrollo de aplicaciones de información geográfica (SIG) con el manejo de tecnologías libres e independientes. Su composición está representada por estudiantes, profesores y especialistas que laboran armoniosamente bajo los conceptos de modelos de desarrollo continuos a lo largo de todo el curso y niveles de formación e investigación asociados a éstos. Pero la integración de roles, tareas y responsabilidades en todo el ambiente productivo hace que la gestión de todos estos recursos se convierta en un proceso complejo dentro del proyecto y obligue a tomar variantes de control que se basen en la utilización de herramientas informáticas. Actualmente toda la gestión de sus recursos, tanto materiales como humanos, así como la planificación de éstos, dígase tiempo de máquina y generación de planes de trabajo se lleva a cabo de forma tradicional en pequeños informes digitales en formato Word y Excel, que no están a disposición de todos los integrantes del proyecto y que traen consigo falta de coordinación y comunicación entre los miembros, así como dificultades a la hora de enfrentar procesos de toma de decisiones o balances de rendimiento o estimación.

Partiendo de la situación expuesta con anterioridad se ha definido como **Problema Científico:** La dificultad para la gestión y control de los recursos humanos, recursos materiales y actividades del proyecto GeneSIG.

Como **Objeto de Estudio** se presenta: Los procesos de gestión de recursos y administración y se define como **Campo de Acción:** La automatización de los procesos de gestión de recursos y administración en el proyecto GeneSIG.

El **Objetivo General** que se propone es**:** Desarrollar un sistema integral personalizado para la gestión de los recursos del proyecto GeneSIG.
Dada la necesidad de solución de los problemas tratados anteriormente se plantea la siguiente

Idea a Defender: Contar con un sistema integral personalizado que gestione los recursos del proyecto GeneSIG, permitirá una mejor administración de sus recursos.

Para dar respuesta al objetivo general planteado se han formulado las siguientes **Tareas de la investigación:**

1. Caracterizar las herramientas para gestionar recursos humanos y tecnológicos para poder realizar el estudio del arte.
2. Seleccionar las herramientas tecnológicas para el desarrollo del software.
3. Documentar análisis y diseño del sistema.
4. Implementar el sistema.
5. Realizar las pruebas del sistema.
6. Analizar la factibilidad de la solución obtenida.
7. Desplegar software para su uso en el proyecto.

Durante el desarrollo de la investigación se ponen de manifiesto diferentes métodos tanto teóricos como empíricos que en ocasiones son aplicados de manera intuitiva. A continuación se identifican algunos de ellos ejemplificando las etapas en las cuales se manifiestan con mayor peso.

Entre los métodos teóricos se evidencia el Histórico-Lógico en el proceso de identificación de posibles soluciones existentes del problema como son caracterización de herramientas de modelación e implementación, metodologías de desarrollo, tecnologías Web o de escritorio, entre otros; el método Analítico-Sintético en la manera de analizar por separado toda la documentación a partir de las partes que componen el objeto de estudio en cuestión y luego sintetizar la información obtenida para comprender las relaciones de las partes que se auto complementan en la práctica; por último la modelación en el desarrollo de las diferentes tablas y diagramas que se construyen como resultado de los diferentes procesos de la Ingeniería de Software.

Como resultado de frecuentes reuniones con algunos de los integrantes del proyecto se pudo constatar de una manera práctica cómo funciona el proyecto GeneSIG, dígase manejo de los recursos humanos y materiales, codificación, tiempo de máquina, reuniones internas y externas. Para poder conocer dichos elementos se hizo necesario entrevistar al jefe de proyecto y al equipo de desarrolladores, quienes aportaron valiosos detalles para una buena caracterización del funcionamiento de GeneSIG. Además se realiza la observación con el fin de

obtener información individual o colectiva referente al tema que se desarrolla, en un ambiente de trabajo real como lo es el proyecto GeneSIG.

Estructura del contenido

En el **Capítulo 1** se aborda todo lo relacionado con la Fundamentación Teórica que sustenta la investigación, son expuestos los principales conceptos que contribuyen al mejor entendimiento del problema en cuestión, se especifican detalladamente todos los argumentos que esclarecen el objeto de estudio y se explican otras soluciones existentes.

En el **Capítulo 2** se brinda una caracterización sobre las herramientas de desarrollo actuales que puedan dar una óptima solución para el desarrollo del software. Estas están agrupadas en los lenguajes de programación, sistemas gestores de bases de datos, metodologías de desarrollo de software y herramienta de modelado utilizadas.

En el **Capítulo 3** se presenta la solución propuesta a partir de la descripción del modelo de negocio y la identificación de requisitos funcionales y no funcionales, que finalmente se agrupan en casos de uso del sistema para conformar el Modelo de Casos de Uso del Sistema.

Por último en el **Capítulo 4** se expone la construcción de la solución propuesta. Se abordan los diagramas de Clases del Análisis y Diseño, Diseño de la Base de Datos además del Modelo de Despliegue y el Modelo de Implementación.
Para finalizar se presentan las conclusiones, las recomendaciones, las referencias bibliográficas y la bibliografía, el glosario de términos y el conjunto de anexos para un mejor entendimiento de lo expuesto a lo largo del trabajo.

CAPÍTULO 1

Fundamentación Teórica.

1.1 Introducción

En el presente capítulo se ofrece una síntesis del estado del arte de la gestión de recursos para el desarrollo de la aplicación. Se exponen conceptos asociados al tema con el fin de esclarecer ambigüedades que puedan surgir alrededor del problema. Además se brinda una descripción exhaustiva del objeto de estudio y la organización que rodea a la situación problemática.

1.2 Conceptos asociados al dominio del problema

A continuación se exponen conceptos que se consideran determinantes para el correcto entendimiento del problema que se trata y de la solución que se propone.

1.2.1 Sistema de Información Geográfica.

(SIG o *GIS*, en su acrónimo inglés) es una integración organizada de hardware, software y datos geográficos diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión. También puede definirse como un modelo de una parte de la realidad referido a un sistema de coordenadas terrestre y construido para satisfacer unas necesidades concretas de información. En el sentido más estricto, es cualquier sistema de información capaz de integrar, almacenar, editar, analizar, compartir y mostrar la información geográficamente referenciada. En un sentido más genérico, los SIG son herramientas que permiten a los usuarios crear consultas interactivas, analizar la información espacial, editar datos, mapas y presentar los resultados de todas estas operaciones. (Bolstad, 2005)

1.2.2 Plataforma.

La plataforma define un estándar alrededor del cual un sistema puede ser desarrollado. Una vez que la plataforma ha sido definida, los desarrolladores de software pueden producir el

software apropiado y los gerentes pueden comprar el hardware apropiado para su uso. El término a menudo es usado como un sinónimo de sistema operativo. El término plataforma Cruzada se refiere a la capacidad de software o hardware para correr de modo idéntico sobre plataformas diferentes. La informática de plataforma cruzada se hace cada vez más importante a medida que las redes de área local se perfeccionan en la integración de las máquinas de diferentes plataformas. Por ejemplo, el ambiente de programación de plataforma cruzada permite a un programador desarrollar programas para muchas plataformas simultáneamente. (Guerra, 2009)

1.2.3 Recursos Humanos (RRHH).

En la administración de empresas, se denomina recursos humanos al trabajo que aporta el conjunto de los empleados o colaboradores de esa organización. Pero lo más frecuente es llamar así a la función que se ocupa de seleccionar, contratar, formar, emplear y retener a los colaboradores de la organización. Estas tareas las puede desempeñar una persona o departamento en concreto (los profesionales en Recursos Humanos) junto a los directivos de la organización. (Mejía, 2004)

1.3 Objeto de Estudio

1.3.1 Procesos de Gestión de Recursos de Administración.

La Gestión de Recursos ha existido desde tiempos muy antiguos, fuertemente relacionada con proyectos de ingeniería de construcción de obras civiles, ejemplos de ellos han sido los proyectos de Ingeniería Hidráulica en Mesopotamia, donde entraban en juego la logística o la creación de equipos de trabajo, con sus categorías profesionales definidas, o la cultura ingenieril desarrollada por el Imperio Romano, donde aparece el control de costos y tiempos y la aplicación de soluciones normalizadas), y en campañas militares, donde también entran en juego muchos elementos de gestión como: identificación de objetivos, gestión de recursos humanos, logística, identificación de riesgos y financiación entre otros.

Pero es a partir de la Segunda Guerra Mundial, donde las potencias en conflicto tuvieron que buscar la forma de manejar una logística muy compleja, que en ese momento era uno de los elementos más importantes para ganar la guerra, fue ahí cuando el avance de estas técnicas

desde el punto de vista profesional transformaron la Gestión de Recursos en una disciplina de investigación.

Desde principios de la década del 90 la gestión de proyectos ha venido instaurándose en todo tipo de empresas, aumentando su impacto a medida que ha ido creciendo la complejidad de los proyectos y soluciones, además por constituir una guía para organizaciones que desarrollen proyectos de cualquier índole.

Según una comparación efectuada entre las principales tendencias la gestión de proyectos se considera la aplicación de conocimientos, habilidades, técnicas y herramientas dentro de un proyecto para definir los requerimientos del mismo. Integra los procesos de inicialización, planeación, ejecución, monitoreo y control y cierre. Ha surgido para ayudar a organizar las acciones y procesos dentro del desarrollo de un proyecto para alcanzar con el mayor éxito posible los objetivos propuestos por el mismo. También contribuye a gestionar los recursos y tener un dominio sobre cada área que influya en el proceso de desarrollo. Las regiones con mayor representación y motivación para crear y adoptar estrategias y metodologías acerca de la gestión de proyectos se concentran en Europa y Norteamérica, aunque en África, Asia y Oceanía se han concretado algunas acciones como creación de asociaciones, organizaciones y guías para la gestión de proyectos. (Mulet, 2008)

Los procesos de gestión de recursos de un proyecto determinado están dados por la gestión de los recursos materiales y humanos que este pueda tener. Para esto se usan herramientas y software de administración necesarios que conducen a una mejor planificación.

Un software de administración de proyectos es un software que tiene dentro de sus responsabilidades la asignación de recursos, programación, colaboración, comunicación y sistemas de documentación, utilizados para ayudar a organizar un proyecto complejo en diferentes tareas y en un tiempo determinado. (Restrepo, 2000)

Técnicas del software de administración de proyectos

I. Programación

Una de las tareas más comunes en la administración es programar y hacer seguimiento a una serie de acontecimientos; la complejidad de esta tarea puede variar considerablemente, dependiendo de las necesidades de la organización de que se trate, de los usuarios y de cómo se utiliza la herramienta. Algunos desafíos comunes incluyen:

- Acontecimientos que dependen el uno del otro de diversas maneras.
- Recursos humanos disponibles para trabajar en las diversas tareas.
- Incertidumbres en las estimaciones de la duración de cada tarea.
- Ordenación de las tareas para satisfacer los plazos.
- Interferencia entre múltiples proyectos, para satisfacer distintos requerimientos simultáneos. (Macedonio, 2005)

II. Cálculo de la Ruta Crítica

En muchos proyectos complejos, habrá una trayectoria crítica o serie de acontecimientos que dependan uno del otro y que sus duraciones determinen directamente la longitud del proyecto entero. Algunos usos del software (por ejemplo, soluciones de la matriz de la estructura de dependencia) pueden destacar estas tareas, que son las que concentrarán el esfuerzo de seguimiento y optimización. (Díaz, 2008)

III. Abastecimiento de la información

El software de planeamiento de proyectos necesita proporcionar mucha información a diversas personas, para justificar el tiempo que se lleva usándolo. Los requisitos típicos podrían incluir:

- Listas de tareas para la gente, y la programación de la asignación de los recursos.
- Información descriptiva acerca de cuánto tiempo tomarán las tareas para terminarse.
- Detección temprana de riesgos del proyecto.
- Información sobre carga de trabajo, por la planeación de días festivos.
- Información histórica sobre cómo han progresado los proyectos, y en particular, cómo se relaciona el desempeño planeado con el actual. (Suhanic, 2003)

1.3.2 Descripción actual del dominio del problema

El objetivo del proyecto GeneSIG consiste en la elaboración de la plataforma GeneSIG que se concibe como un producto soberano que sirva como soporte al desarrollo de aplicaciones de Sistemas de Información Geográfica en entornos Web con tecnologías libres, capaces de realizar la representación geoespacial de la información asociada a negocios específicos, además de permitir realizar análisis sobre dicha información. Es un producto de gran auge en la actualidad con grandes expectativas en el mercado internacional.

1.3.2.1 Personal de dirección

El equipo de proyecto está compuesto por especialistas funcionales, profesores y estudiantes donde estos últimos conforman actualmente el 69% del personal del proyecto; están divididos entre segundo y quinto año (12 de segundo, 4 de tercero, 15 de cuarto año y 19 de quinto). Entre los profesionales se cuenta con 9 profesores, 7 técnicos de la FAR y 6 especialistas. Para conformar una fuerza laboral de 72 personas, como se muestra en la Figura 1.
La estructura vigente es la que se presenta a continuación:

- Un líder de proyecto que cuenta con un equipo de dirección compuesto por 2 líderes técnicos, el Analista Principal, el Arquitecto Principal de Software y el Jefe de Desarrollo.
- Un equipo de especialistas de GEOCUBA y las Fuerzas Armadas.
- Un equipo de implementadores homogeneizados con profesionales y estudiantes de todos los años.
- Un equipo de analistas que incluye analistas de riesgos y probadores.
- Un equipo de arquitectos de software y arquitectos de información.
- Un equipo de calidad compuesto por profesores del Polo productivo.
- Un diseñador Web.
- Un administrador de Bases de Datos.

La dirección a nivel superior está compuesta por el Decano y el Vicedecano de Producción e Investigaciones de la Facultad 9, el Jefe del Centro UCID y la Jefe del Polo productivo Geoinformática.

Figura 1 Estructura Organizativa del proyecto GeneSIG.

Los recursos materiales se encuentran el el laboratorio 807 de UCIFAR, donde se cuenta con 24 PC (23 PC Assus de 3.0 GHz de velocidad, con 160 GB de capacidad de disco duro y 1512 MB de RAM y 1 PC Dell de 2.0GHz, 160 GB de disco duro y 2048 MB de RAM) en las que se divide el tiempo de trabajo de los estudiantes en 3 sesiones (mañana, tarde y noche) para dar cumplimiento a las tareas orientadas por parte de los profesores y líderes del proyecto.

1.3.3 Situación Problemática

En la actualidad toda la toda la gestión de los recursos del proyecto, tanto materiales como humanos, así como la planificación de estos se lleva a cabo de forma manual en papel o en pequeños informes digitales (dígase documentos Word y tablas Excel) que no están al acceso de todos los integrantes del proyecto, trayendo consigo que no haya una coordinación estable entre los miembros de la dirección de GeneSIG, así como poca claridad a la hora de enlazar los recursos humanos y materiales con que cuenta la plataforma, trayendo consigo la poca rapidez y disponibilidad de los mismos

1.4 Análisis de otras soluciones existentes

Las herramientas de gestión de proyectos pueden funcionar como aplicaciones desktop en el ordenador de cada usuario, brindando una amplia y amigable capacidad gráfica. Pero la tendencia actual al uso de aplicaciones Web deja al margen de las preferencias a los programas de escritorio ya que de esta manera se amplían las opciones de control de acceso y mantenimiento de las mismas. (Caldera, 2004)

1.4.1 Software de administración de proyectos basado en Aplicaciones Web.

La administración de proyectos se puede llevar a cabo mediante una aplicación Web accediendo a través de una Intranet o de una red externa usando un navegador Web.

A continuación se muestran las ventajas y desventajas de aplicaciones Web.

- Se puede acceder desde cualquier tipo de computadora sin la instalación de software.
- Facilidad del control de acceso.
- Naturalmente multiusuario.
- Solamente una instalación/versión de software para mantener.
- Originalmente es más lento para responder que las aplicaciones de escritorio.
- Capacidad gráfica más limitada que las aplicaciones de escritorio. (Suhanic)

En la actualidad existen en el mundo varias aplicaciones que se dedican a la gestión de recursos dentro de un proyecto cabe señalar entre los más populares ***EGroupWare, Track y Project Insight.***

EGroupWare es una aplicación para el trabajo en grupo en una red corporativa o intranet, la cual está integrada por diferentes aplicaciones tales como calendario, correo electrónico, libreta de direcciones, administración de proyectos, entre otras, las cuales serán descritas más adelante.

Al estar basada en servidor Web el usuario no queda limitado a trabajar sólo en una empresa o universidad, sino que puede acceder a datos personales desde cualquier punto del mundo, sólo se necesita tener una computadora con una conexión a Internet por medio de un navegador Web.

Los datos están centralizados en una base de datos y gestionados a través de permisos de acceso asignados a cada usuario o grupo en particular. Esto permite, por ejemplo, tener centralizados en un servidor los calendarios y tareas de varios usuarios y poderlos compartir siempre en función de los permisos de acceso que se hayan establecido.

Beneficios:

• Fomentar el trabajo en equipo.

• Mejora la comunicación entre departamentos y/o sucursales.

• Acceso y divulgación de información, normas, problemas y soluciones.

• Control de los contactos de la organización.

• Automatización de tareas y proceso de la transmisión de información. (Puebla, 2001)

Esta herramienta presenta inconvenientes pues fue creada para la venta y el marketing, y ese no es el propósito de la organización que requiere el producto, solamente permite la gestión de la planificación y no del control de otros recursos como materiales y humanos.

En el caso de la herramienta ***Track*** es un sistema para la gestión de proyectos, el cual comparte información entre los usuarios que interactúan con el sistema, todo este proceso de compartir información recibe el nombre de WIKI (Gestión de conocimientos).Este sistema es un software libre programado en PYTHON.

Track da cumplimiento a una serie de funcionalidades tales como:

- Gestionar tareas a partir de hitos de la ingeniería de software.
- Autenticación por LDAP (Servidor de Nombres de Dominio) que permite un acceso de lectura rápida.
- Subversión (Gestión de Configuración), la cual organiza el repositorio de control de versiones.

Esta herramienta tiene algunas desventajas como por ejemplo: que cada proyecto es independiente y no comparten la base de datos entre ellos, el sistema de permisos no es actualmente muy flexible, no hay integrados motores de búsqueda avanzada, por lo que no da cumplimiento a nuestro objetivo. De dicha herramienta no se utilizará ninguna funcionalidad. (Capote, y otros, 2008)

Project Insight es un Software de administración de proyectos de tecnología Web y Ajax que permite a la dirección, gestión de proyectos y miembros del equipo para colaborar eficazmente en los proyectos. (West, 2007)

Project Insight comenzó en 1997 como un equipo de desarrolladores de aplicaciones Web dispersos geográficamente que trabajaban conjuntamente en proyectos de software. El equipo desarrollo el sistema inicial para ser capaz de trabajar juntos de manera más efectiva desde ubicaciones remotas, mientras a la vez mejoraban en la comunicación de los objetivos de los

proyectos. *Project Insight* apareció disponible de manera comercial en 2002. (Wire, September 17, 2002)

Pero la deficiencia de este software radica en que es propietario, para trabajar con este se necesita pagar la licencia, razón por la cual tampoco se puede usar para dar solución a la situación problemática.

1.5 Conclusiones

Por la índole de los conceptos que se relacionan en este capítulo, que desde el punto de vista teórico permiten un mejor entendimiento de lo esbozado en la situación problemática o el marco del problema en sentido general, es posible constatar que la investigación realizada cuenta con un grado de rigor medio, acorde al tipo de sistema que se desea desarrollar.

La descripción de la situación problemática, así como del problema a resolver que da pie a la realización del presente trabajo, es un ejercicio objetivo en cuanto a que se sustenta sobre principios sólidos obtenidos a partir de la búsqueda directa de los detalles de los mismos en el ambiente donde tienen lugar.

También se identifican como posibles soluciones los sistemas de administración de proyectos, *EGroupWare, Track y Project Insight*, los cuales se descartan cuando se realizan las caracterizaciones de cada uno, determinando que no satisfacen los requisitos que el proyecto GeneSIG requiere.

CAPÍTULO 2

Tecnologías a utilizar.

2.1 Introducción

En el presente capítulo se brinda una caracterización sobre las tendencias y desarrollo de las tecnologías actuales que se utilizan para el desarrollo de MaGIStral. Éstas están agrupadas en los Lenguajes de Programación, Sistema Gestor de Base de Datos, Metodología de Desarrollo de Software y Herramienta *CASE (Computer Aided Software Engineering)*

2.2 Arquitectura Cliente-Servidor.

Esta arquitectura se divide en dos partes claramente diferenciadas, la primera es la parte del servidor y la segunda la de un conjunto de clientes. Normalmente el servidor es una máquina bastante potente que actúa de depósito de datos y funciona como un sistema gestor de base de datos (SGBD). Por otro lado los clientes suelen ser estaciones de trabajo que solicitan varios servicios al servidor. Ambas partes deben estar conectadas entre sí mediante una red.

Este tipo de arquitectura es la más utilizada en la actualidad, debido a que es la más avanzada y la que mejor ha evolucionado en estos últimos años.

Se puede decir que esta arquitectura necesita tres tipos de software para su correcto funcionamiento:

- Software de gestión de datos: Este software se encarga de la manipulación y gestión de los datos almacenados y requeridos por las diferentes aplicaciones. Normalmente este software se aloja en el servidor.
- Software de desarrollo: este tipo de software se aloja en los clientes y sólo en aquellos que se dedique al desarrollo de aplicaciones.
- Software de interacción con los usuarios: También reside en los clientes y es la aplicación gráfica de usuario para la manipulación de datos, siempre claro a nivel usuario (consultas principalmente). (Alvarez, 2007)

Además de estos, existen más aplicaciones software para el correcto funcionamiento de esta arquitectura pero ya están condicionados por el tipo de sistema operativo instalado, el tipo de red en la que se encuentra, etc.

2.3 Lenguaje Web del lado cliente: JavaScript.

Los lenguajes de lado cliente son aquellos que pueden ser directamente "digeridos" por el navegador. Cada uno de estos tipos tiene por supuesto sus ventajas y sus inconvenientes. Así, por ejemplo, un lenguaje de lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio sin necesidad de pagar más ya que, por regla general, los servidores que aceptan páginas con scripts de lado servidor son en su mayoría de pago o sus prestaciones son muy limitadas. Entre los lenguajes del lado del cliente cabe señalar que no sólo se encuentra HTML además están JavaScript, Applets De Java, Visual Basic Script, CSS entre otros.

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas. Una página Web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java, que si es orientado a objeto (OO). Legalmente, JavaScript es una marca registrada de la empresa Sun Microsystems. (Pérez, 2009)

2.3.1 Librería de JavaScript: ExtJS

La programación con la librería ExtJS, que extiende la librería YUI e integra AJAX, Prototype y Scriptaculous. Con el tiempo se convirtió en un *framework* independiente y a principio de 2007 se creó una compañía para comercializar y dar soporte al ExtJS.

ExtJS es neutral al lenguaje que se use en el servidor. Siempre que el resultado se envíe a la página en el formato adecuado, ExtJS no se preocupará de lo que pase en el servidor. Hay

docenas de componentes a escoger en ExtJS, incluyendo composiciones automáticas de páginas, pestañas, menús, barras de herramientas, diálogos, vistas en árbol. Proporciona un selector de nodos DOM extremadamente poderoso llamado DomQuery (puede usarse como una librería independiente, pero en el contexto de ExtJS se usará para seleccionar elementos para poder interactuar con ellos a través de la interfaz Element, contiene mucho de los métodos y propiedades de DOM que se necesitará proporcionando una interfaz conveniente, unificada y multinavegador).

Para trabajar con las librerías de ExtJS es necesario que exista un adaptador, donde esta clase sobre la que luego se define las funciones de las librería (el elemento Ext. como tal).

En su última versión Ext. 3.0 es muy extenso y recordando todas las funciones, propiedades o configuraciones disponible es casi imposible. La documentación del API está muy completa. Se emplean IDEs de desarrollo de JavaScript, el Aptana Studio es una herramienta potente y disponible para el apoyo directo en las aplicaciones desarrolladas en ExtJS. (Leyva, 2008)

2.4 Lenguaje Web del lado servidor: PHP.

PHP es el lenguaje de programación más utilizado en la realización de páginas Web avanzadas. El principal objetivo de PHP5 ha sido mejorar los mecanismos de Programación Orientada a Objeto para solucionar las carencias de las anteriores versiones. Una de las principales ventajas que presenta PHP, es su soporte para una gran cantidad de bases de datos además de poder ser utilizado en los distintos sistemas operativos ya sea en Linux, muchas variantes Unix, Windows, Mac OS X, entre otros, además de soportar la mayoría de los servidores de hoy en día, incluyendo Apache, Xampp y Microsoft Internet Information Server. (Potencier, y otros, 2008)

2.4.1 Symfony

Un *framework* simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un *framework* proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Existen algunos *frameworks* que se consideraron como opciones viables para el desarrollo de la aplicación en cuestión ya que se plantea la necesidad del uso de PHP5 como lenguaje del

lado del servidor. De modo que, aunque CodeIgniter, Cake PHP y Symfony comparten propiedades como el soporte para el lenguaje mencionado, la implementación del patrón modelo vista controlador y múltiples ventajas para validación, el primero no cuenta con opciones para autenticación haciendo de la seguridad un problema de mayor complejidad, ni con soporte para múltiples bases de datos disminuyendo el grado de escalabilidad de un sistema de este tipo. La decisión entre los dos candidatos potenciales que quedaron después del análisis anterior, estuvo determinada por el nivel de experiencia de los desarrolladores en cada uno de los *framework*.

Por tanto se propone finalmente el uso de Symfony que es además un completo *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones Web Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación Web

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios Web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix y Linux) como en plataformas Windows. (Potencier, y otros, 2008)

2.5 El Lenguaje Unificado de Modelado (UML) como soporte de la modelación de la solución propuesta.

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas,

dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. La estructura estática define los tipos de objetos importantes para un sistema y para su implementación, así como las relaciones entre los objetos. El comportamiento dinámico define la historia de los objetos en el tiempo y la comunicación entre objetos para cumplir sus objetivos.

El modelar un sistema desde varios puntos de vista, separados pero relacionados, permite entenderlo para diferentes propósitos. UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo, en un entorno de desarrollo complejo. Contiene construcciones para representar decisiones de implementación y para elementos de tiempo de ejecución en componentes.

UML no es un lenguaje de programación. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguajes de programación, así como construir modelos por ingeniería inversa a partir de programas existentes. (Guerra, Mayo 2009)

2.6 Metodología de Desarrollo

2.6.1 El Proceso Desarrollo de Software (RUP).

RUP Proceso Unificado Racional es una metodología cuyo fin es entregar un producto de software. Se estructura todos los procesos y se mide la eficiencia de la organización. Es un proceso de desarrollo de software el cual utiliza el lenguaje unificado de modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El RUP es un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Describe como aplicar enfoques para el desarrollo del software, llevando a cabo unos pasos para su realización. Se centra en la

producción y mantenimiento de modelos del sistema además de servir como guía de cómo usar UML. RUP implementa el desarrollo iterativo del software: que permite comprender los requerimientos que hacen crecer el sistema, sigue un modelo que busca las tareas más riesgosas, reduciendo así los riesgos del proyecto. La administración de requerimientos: que describe como se obtienen, organizan, documentan los requerimientos, capta y comunica los requerimientos de la organización y documenta las decisiones. El uso de arquitecturas basadas en componentes: se basa en diseñar una arquitectura que sea flexible, fácil de modificar, comprensible y que se fundamenta en la reutilización de sus componentes. El Modelado visual del software: el cual modela visualmente la organización, permite analizar la consistencia entre los componentes, el diseño y su implementación, verifica la calidad del software y el control de cambios. El RUP está basado en 5 principios:

Adaptar el proceso: El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico, aunque se debe tener en cuenta el alcance del proyecto.

Balancear prioridades: Debe encontrarse un balance que satisfaga los deseos de todos.

Demostrar valor iterativamente: Los proyectos se entregan en etapas iteradas. En cada iteración se analiza la opinión, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

Elevar el nivel de abstracción: Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes de cuarta generación (SQL, lenguajes de consulta), o esquemas (*frameworks*). Esto previene a los ingenieros de software ir directamente de los requisitos a la codificación de software a la medida del cliente. Un nivel alto de abstracción también permite discusiones sobre diversos niveles arquitectónicos. Éstos se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con UML.

Enfocarse en la calidad: El control de calidad debe en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminar cada una de ellos, estos a la vez se dividen en fases y donde se debe tomar una decisión importante. El ciclo de vida organiza las tareas en fases e iteraciones. RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades.

Inicio: se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos.

Elaboración: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.

Construcción: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.

Transición: se Instala el producto en el cliente y se entrena a los usuarios. Surgen nuevos requisitos a ser analizados.

RUP en cada una de sus fases pertenecientes a la estructura estática realiza una serie de artefactos que sirven para comprender mejor tanto el análisis como el diseño del sistema. Brinda una forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo). Pretende implementar las mejores prácticas en Ingeniería de Software, posee un desarrollo iterativo, administración de requisitos. Usa una arquitectura basada en componentes que posibilita el control de cambios. También ofrece un modelado visual del software así como la verificación de la calidad del mismo.

El RUP se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. La metodología RUP es más apropiada para proyectos grandes (Aunque también pequeños), dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En proyectos pequeños, es posible que no se puedan cubrir los costos de dedicación del equipo de profesionales necesarios. (Guerra, Mayo 2009)

2.7 Herramientas CASE

2.7.1 Visual Paradigm.

Con la propuesta de utilización de UML como lenguaje de modelado se crea la disyuntiva en la decisión del uso de Rational Rose o Visual Paradigm como herramienta *CASE (Computer Aided Software Engineering)*, ya que ambas soportan dicho lenguaje, pero es más optimo Visual Paradigm porque que es multiplataforma lo que permite que el desarrollo de la aplicación sea mucho más flexible.

El Visual Paradigm es un producto distinguido que facilita la organización de los diagramas, su misión es diseñar, integrar y desplegar las aplicaciones de la empresa y sus bases de datos subyacentes. La herramienta ayuda al equipo de desarrollo de software a agilizar el modelado

del software, aumentando al máximo y acelerando el trabajo en equipo y las contribuciones individuales**.** La herramienta CASE Visual Paradigm posibilita:

1. Generación de código (PHP).
2. Entorno de creación de diagramas para UML.
3. Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
4. Análisis de textos.
5. Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
6. Capacidades de ingeniería directa (versión profesional) e inversa.
7. Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
8. Disponibilidad de múltiples versiones, para cada necesidad.
9. Disponibilidad en múltiples plataformas. (Hernández, 2005)

2.8 Sistema Gestor de Bases de Datos

2.8.1 PostgreSQL

Se eligió este gestor de base datos por las facilidades que brinda, es un servidor de base de datos relacional libre (lo cual es muy importante para la universidad y el país, que en estos momentos está tratando de migrar para software libre con el objetivo liberarse de Microsoft). Tiene la característica de que se puede modificar de acuerdo a las necesidades y si es usado en plataformas *nix (Unix y Linux) es inmune a los virus. Además se puede adquirir mediante un costo bajo o nulo, los requerimientos para su instalación son mínimos y presenta gran estabilidad y confiabilidad.

En cuanto a los diferentes gestores de base de datos que fueron analizados además del antes mencionado, MySQL y SQL Server, tienen algunas desventajas en comparación con PostgreSQL, como por ejemplo ambos gestores son propietarios y en el caso de MySQL, no tiene tantas capacidades como otros gestores profesionales, no tiene integridad referencial, es lento con grandes bases de datos y no es tan robusto, ni soporta transacciones, ni muchos usuarios. En el caso de SQL Server el costo es muy elevado y las licencias son privadas a diferencia de PostgreSQL. (Oktaba, Hanna y otros. 2004)

2.9 Conclusiones.

Queda reflejado que la propuesta de solución consiste en una aplicación Web, donde el lenguaje de programación a usar para implementar la lógica del negocio del sistema y manejar las conexiones a la base de datos es PHP con el uso del *Framework Symfony* del lado del servidor y JavaScript con ExtJS en la interfaz del cliente. Para el almacenamiento de los datos que procesa la aplicación se escoge PostgreSQL así como el uso de RUP como metodología de desarrollo en su forma ágil por ser MaGIStral una aplicación relativamente pequeña que necesita ser desplegada en el menor tiempo posible.

CAPÍTULO 3

Presentación de la solución propuesta.

3.1 Introducción

En el presente capítulo se muestra formalmente el Modelo Dominio a emplear en la investigación, se describe el entorno donde se desarrolla el sistema, el diagrama de clases del dominio y se presenta el glosario con todos los términos utilizados para posibilitar el entendimiento del mismo. Finalmente las especificaciones de los requerimientos funcionales y no funcionales del sistema, que a partir de los cuales se construye el modelo del sistema con su diagrama de casos de uso y la descripción de los mismos.

3.2 Entorno donde trabajará el sistema

El Modelo de Dominio se utiliza para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. Similares a los mapas mentales utilizados en el aprendizaje, el modelo de dominio es utilizado por el analista como un medio para comprender el sector industrial o de negocios al cual el sistema va a servir. Para la elaboración del mismo se requiere de la participación de Expertos de Negocio. Quienes los dividen en tres pasos:

- Identificar las Clases Conceptuales.
- Dibujarlas en un Diagrama de Clases.
- Añadir Relaciones y Atributos.

Debido a que en el vigente trabajo no están bien definidos los procesos del negocio ya que no se define la fronteras de los mismo y tampoco se puede establecer diferencia al establecer los actores y trabajadores del negocio ya que para este caso es la misma persona, se procede a realizar una modelación del dominio y se explica cada uno de los conceptos que forman parte del mismo.

3.2.1 Eventos principales del entorno

El Jefe de Proyecto es el encargado de dirigir el personal presente en el proyecto (Estudiantes y Profesores) a quienes le asigna las actividades y las tareas a desarrollar, con las tareas y las actividades del proyecto se genera el Plan de Trabajo que no es más que un cronograma de todos los eventos del proyecto. Los profesores y estudiantes se encargan de dar cumplimiento a dichos eventos, para esto disponen de los recursos materiales presentes en el proyecto.

3.2.2 Modelo Conceptual del Dominio

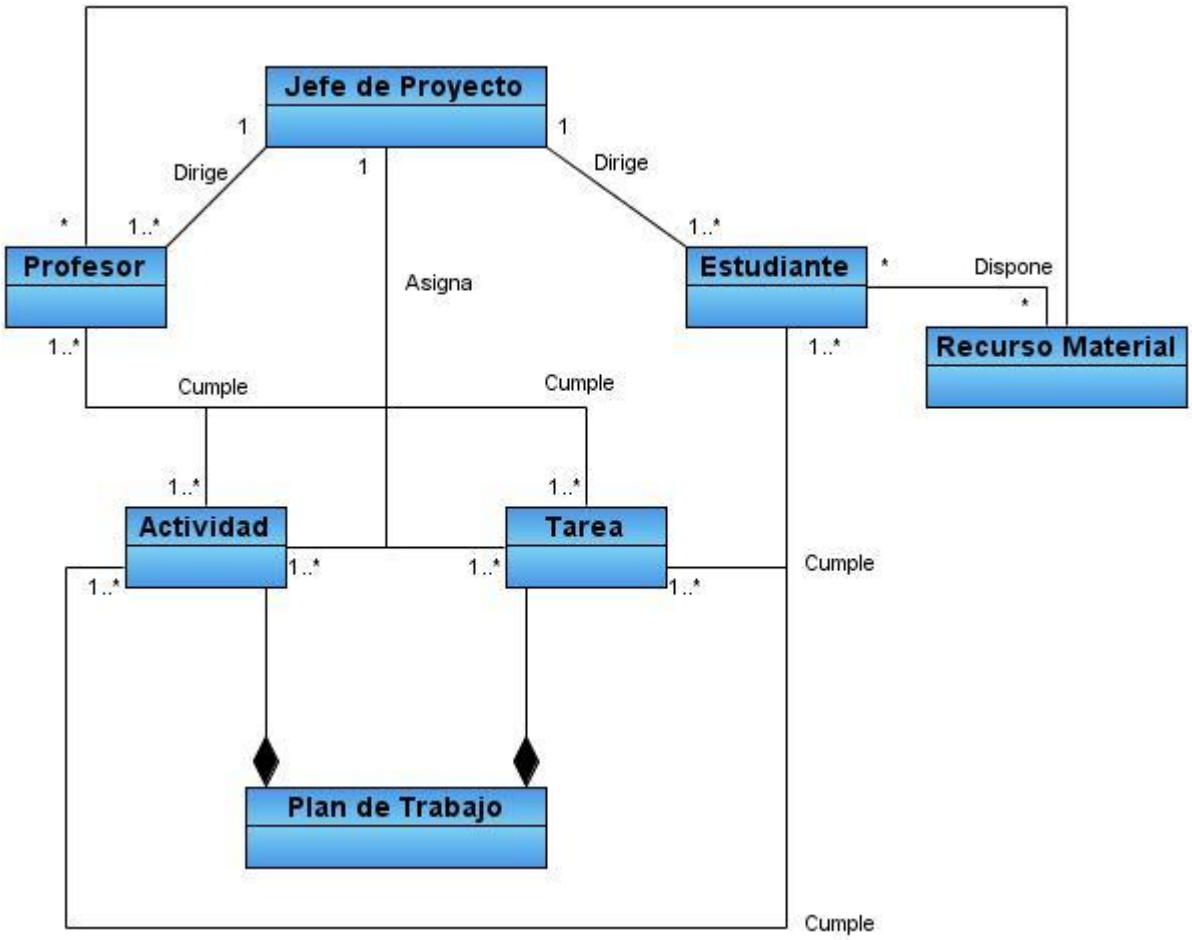

Figura 2 Modelo Conceptual del Dominio.

3.2.3 Glosario de Términos del Dominio

Jefe de Proyecto: Es quien dirige las actividades de control y planificación a desarrollar dentro del proyecto.
Profesor: Se encarga de dar cumplimiento a las tareas a desarrollar dentro del proyecto.
Estudiante: Se encarga de dar cumplimiento a las tareas a desarrollar dentro del proyecto.
Tarea: Trabajo vinculado con la producción que debe hacerse en tiempo limitado.
Recurso Material: Conjunto de medios informáticos mediante los cuales los estudiantes y profesores dan cumplimiento de las actividades y tareas asignadas.
Actividad: Evento que transcurre en el proyecto.
Plan de Trabajo: Cronograma de los eventos presentes en el proyecto.

3.3 Requerimientos Funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Los requisitos que debe de cumplir MaGIStral para la gestión de los recursos el proyecto GeneSIG son:

R1 Autenticar Usuario.
R2 Gestionar Recurso Humano.
R2.1 Adicionar Recurso Humano.
R2.2 Eliminar Recurso Humano.
R2.3 Mostrar Recurso Humano.
R2.4 Modificar Recurso Humano.
R3 Gestionar Recurso Material.
R3.1 Adicionar Recurso Material.
R3.2 Eliminar Recurso Material.
R3.3 Mostrar Recurso Material.
R3.4 Modificar Recurso Material.
R4 Generar reporte de Recursos Humanos.
R5 Generar reporte de Recursos Materiales.
R6 Exportar reporte de Recursos Humanos a Excel.
R7 Exportar reporte de Recursos Materiales a Excel.

R8 Exportar reporte de Recursos Humanos a PDF.

R9 Exportar reporte de Recursos Humanos a PDF.

R10 Gestionar Tareas.

R10.1 Adicionar Tarea.

R10.2 Eliminar Tarea.

R10.3 Mostrar Tarea.

R10.4 Modificar Tarea.

R11 Gestionar Plan de Trabajo.

R11.1 Adicionar Plan de Trabajo.

R11.2 Eliminar Plan de Trabajo.

R11.3 Mostrar Plan de Trabajo.

R11.4 Modificar Plan de Trabajo.

R12 Gestionar Actividad General del Proyecto.

R12.1 Adicionar Actividad.

R12.2 Eliminar Actividad.

R12.3 Mostrar Actividad.

R12.4 Modificar Actividad.

R13 Gestionar Ticket.

R13.1 Adicionar Ticket.

R13.2 Eliminar Ticket.

R13.3 Mostrar Ticket.

3.4 Requerimientos No Funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Son los lineamientos que demuestran que un producto sea atractivo, usable, rápido o confiable. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto y de esta forma poder decir si el producto tiene la calidad solicitada.

3.4.1 Requisitos de Software.

El sistema se desarrolla con el lenguaje PHP versión 5, corre sobre un servidor con el sistema operativo UNIX (Linux) o Windows. Se recurre a la tecnología Apache versión 2.0 o superior

para el servidor Web. El sistema incluye una base datos implementada en PostgreSQL versión 8.0 o superior.

En las PCs de los clientes debe estar instalado el navegador Mozilla Firefox, versión 1.5 o superior, Zafari u otro navegador que cumpla con los estándares de la *World Wide Web Consortium* (W3C). La comunicación de las computadoras clientes con el servidor es a través de conexiones de cables de red de par trenzado, a una velocidad constante de 100 Mbps o superior.

3.4.2 Requisitos de Hardware.

Para las PCs clientes:

- Se requiere tengan tarjeta de red.
- Al menos 64 MB de memoria RAM.
- Se requiere al menos 100 MB de disco duro.
- Procesador 512 MHz como mínimo.

Para los servidores:

- Se requiere tarjeta de red.
- El servidor requiere como mínimo 128 MB de RAM y 5GB de disco duro.
- Procesador 1 GHz como mínimo.

3.4.3 Requisitos de apariencia o interfaz externa.

La interfaz debe ser sencilla y de fácil interpretación para que puedan acceder a la misma sin problemas personas que no tienen un amplio conocimiento informático. Debe tener diferentes opciones para mostrar y confeccionar documentos, así como para visualizar reportes. Tendrá un menú dinámico que interactúe con el usuario de acuerdo a las necesidades del mismo. Debe estar concebido para simular una aplicación de escritorio. Los colores a usar deben ser refrescantes para la vista, colaborando con los requerimientos

3.4.4 Requisitos de Seguridad.

El sistema debe comunicarse mediante protocolo http, chequear si el usuario que está accediendo al sistema esta autenticado y brindarle servicio de autenticación, mostrar las operaciones de acuerdo al rol del usuario y no más, mantener la integridad de la información, es decir, que no se perderá durante su almacenamiento o transporte.

3.4.5 Requisitos de Usabilidad.

El sistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras. Además el software tendrá siempre visible la opción de Ayuda, lo que posibilitará un mejor aprovechamiento por parte de los usuarios de sus funcionalidades así como un mapa de la Web para una mejor orientación.

3.4.6 Requisitos de Soporte.

Para el servidor de aplicaciones: Se requiere que esté instalado un intérprete de ficheros PHP 5 o una versión superior y con las últimas actualizaciones del lenguaje. Para el servidor de base de datos: Se requiere que esté instalado un gestor de base de datos Postgres 8 o superior que maneje la concurrencia y transacciones. Para el cliente: Se requiere que esté instalado un navegador que interprete JavaScript y versiones HTML 3.0 o superior.

3.4.7 Portabilidad.

El sistema será multiplataforma y su puesta en marcha debe ser compatible con los sistemas operativos, Windows, Linux preferentemente este último.

3.4.8 Requisitos Legales.

El sistema debe basarse en el documento que expone las normativas para el desarrollo de software del proyecto. La mayoría de las herramientas de desarrollo son libres y del resto, las licencias están avaladas.

3.4.9 Requisitos de confiabilidad.

La información manejada por la base de datos, para proteger su confidencialidad e integridad, estará protegida de acceso no autorizado y divulgación.

3.4.10 Disponibilidad

El sistema debe estar disponible durante las 24 horas para que todos los usuarios pertenecientes al proyecto tengan acceso al mismo.

3.5 Descripción del Sistema Propuesto.

3.5.1 Descripción de los actores.

Tabla 1 Descripción de los actores del sistema.

Actores del Sistema	Justificación
Usuario	Es el actor genérico del cual todos heredan para dar cumplimiento al caso de uso Autenticarse.
Administrador	Es el encargado de otorgar los permisos a los usuarios del sistema.
Jefe Proyecto	Es quien dirige las actividades productivas a nivel de Proyecto. Es quien organiza, planifica y estructura los proyectos.
Dir UCI	Es un servicio de directorio de la universidad que provee los datos personales dado el usuario del mismo.

3.5.2 Casos de Uso del Sistema.

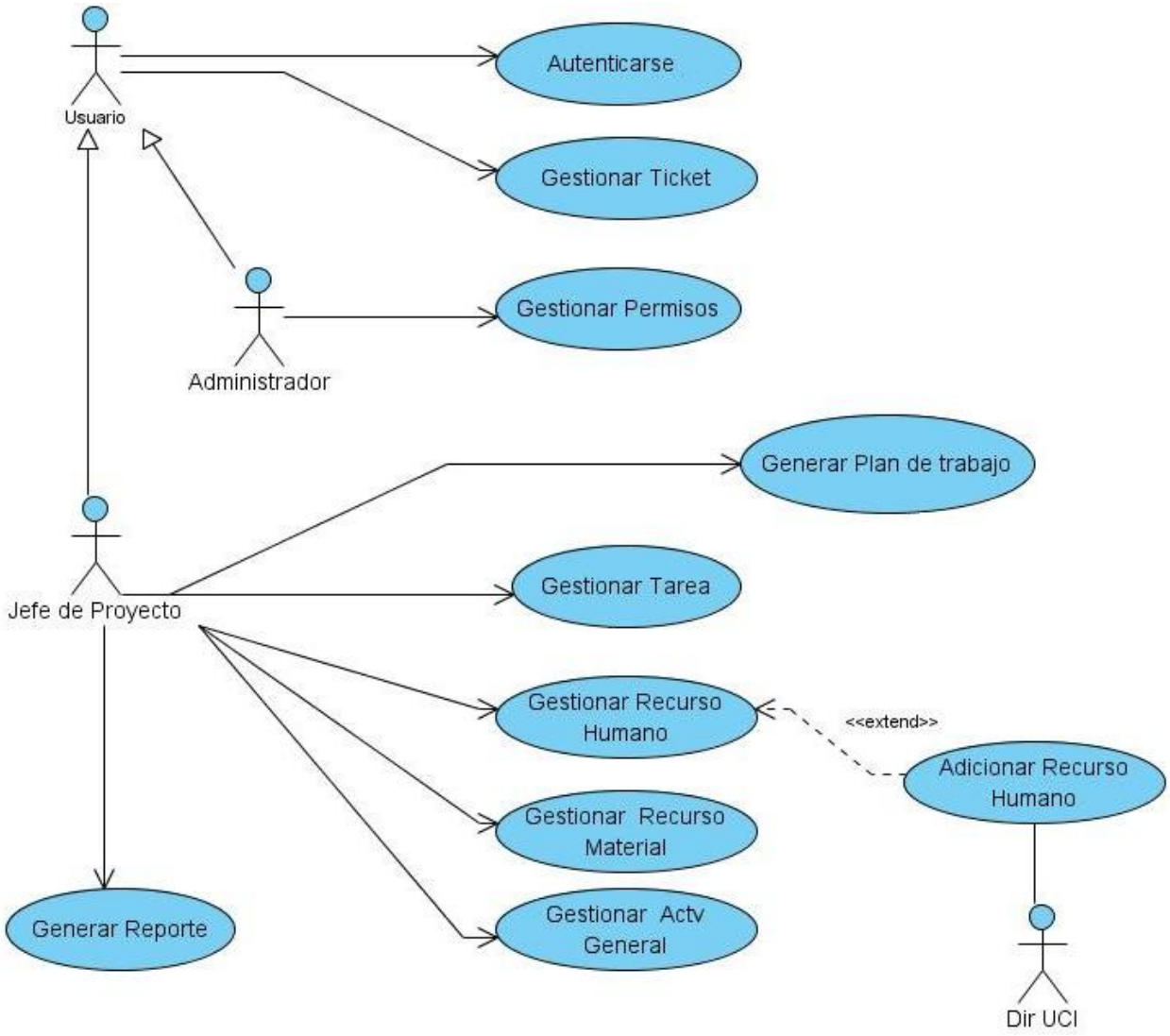

Figura 3 Diagrama de Caso de Uso del Sistema.

Tabla 2 Especificación del Caso de Uso "Autenticarse."

CU- 1	**Autenticarse**
Actor	Usuario
Descripción	El caso de uso comienza cuando el administrador, jefe de proyecto o los usuarios deseen acceder al sistema. Éste pide autenticación para el acceso, si los datos entrados son correctos el sistema le permite su acceso. En caso contrario se redirecciona

	al usuario a la misma página de autenticación y se muestra un mensaje de error
Referencia	RF-1.
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
1- El administrador, el jefe de proyecto o los usuarios desean entrar en el sistema, introducen su usuario y contraseña.	2- El sistema verifica el usuario y contraseña y permite el acceso si los datos son correctos.
Flujos alternativos	
	Línea1: en caso de que el usuario de aceptar sin haber llenado los campos previamente el sistema no prosigue hasta que éstos se llenen. Línea2: En caso de que el administrador, el jefe de proyecto o los usuarios introduzcan una contraseña no válida el sistema mostrará una interfaz con un mensaje diciendo “Vuelva a introducir sus datos”.

Tabla 3 Especificación del Caso de Uso “Gestionar Tarea”

CU- 3	**Gestionar Tarea**
Actor	Jefe de proyecto
Descripción	El caso de uso se inicia con el jefe de proyecto. El sistema muestra la interfaz donde el jefe de proyecto puede adicionar una nueva tarea para esto primero se necesita contar con el integrante responsable de la tarea por lo que este caso de uso no debe ser inicializado sin antes haber gestionado los recursos humanos, también tiene la posibilidad de eliminar o mostrar al usuario una ya

	existente o modificara. Al final se guardan los datos en el sistema, terminando así el caso de uso.
Referencia	RF-10; 10.1; 10.2; 10.3; 10.4
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
1- El jefe de proyecto desea gestionar una tarea, despliega la sesión "Tareas" en el menú principal.	2- El sistema muestra un menú con las opciones posibles para una tarea: Gestionar Tarea o Reportes.
3- El jefe de proyecto decide "Gestionar Tarea": • Adicionar tarea (ver sección "Adicionar"). • Modificar datos de la tarea (ver sección "Modificar"). • Duplicar datos de la tarea (ver sección "Duplicar"). • Eliminar datos de la tarea (ver sección "Eliminar").	
Sección Adicionar	
1. El jefe de proyecto desea adicionar una tarea nueva y selecciona la opción "Adicionar" presente en la interfaz.	2. El sistema muestra una nueva interfaz para llenar los datos de la nueva tarea.
3. El jefe de proyecto llena los campos de datos y da la opción de guardar datos haciendo click en el	4. El sistema guarda los datos y muestra la lista de tareas actualizada.

botón Adicionar.	
Sección Duplicar	
1. El jefe de proyecto selecciona la tarea que desea duplicar (que no es más que esa misma tarea asignarla a otro usuario a la vez) en el listado que se muestra en la interfaz de gestión y acciona la opción “Duplicar”.	2. El sistema muestra una interfaz con todos los integrantes del proyecto.
3. El jefe de proyecto selecciona a quien más le va a asignar la tarea y presiona “Aceptar”.	4. El sistema guarda la tarea como una tarea igual a la anterior pero con otro responsable. Y la muestra en el listado de tareas.
Sección Modificar	
1. El jefe de proyecto selecciona la tarea que desea modificar en el listado que se muestra en la interfaz de gestión y acciona la opción “Modificar”.	2. El sistema muestra una interfaz con los datos de la tarea.
3. El jefe de proyecto modifica los datos que debe y presiona “Modificar”	4. El sistema muestra la lista de tareas actualizada
Sección Eliminar	
1. El jefe de proyecto escoge de la lista que se muestra la tarea que desea eliminar y presiona el botón “Eliminar”.	2. El sistema muestra el mensaje de confirmación: “Desea eliminar la Tarea señalada”
3. El jefe de proyecto oprime “Si”	4. El sistema muestra la lista actualizada.
Flujos alternativos	

Sección Adicionar	Línea1: En el caso de que el jefe de proyecto de la opción cancelar, se retorna a la interfaz anterior. Línea2: En caso de que el jefe de proyecto de la opción de guardar los datos sin haber llenado todos los campos previamente el sistema devuelve un mensaje de error.
Sección Duplicar	Línea1: En el caso de que el jefe de proyecto de la opción cancelar, se retorna a la interfaz anterior. Línea2: En caso de que el jefe de proyecto de la opción de guardar sin haber seleccionado previamente al responsable el sistema devuelve un mensaje de error.
Sección Modificar	Línea1: En el caso de que el jefe de proyecto no señale primeramente la tarea a modificar y marque modificar, aparecerá en pantalla un mensaje el mensaje de corrección: “Debe seleccionar una tarea primero” . Línea2: En el caso de que el jefe de proyecto de la opción cancelar, no se efectuará la tarea requerida y se retornara a la interfaz anterior.
Sección Eliminar	Línea1: En el caso de que el jefe de proyecto no señale primeramente la tarea a eliminar y simplemente marque eliminar, aparecerá en pantalla el mensaje de corrección: “Debe seleccionar una tarea primero” Línea2: En el caso de que el jefe de proyecto

	de la opción cancelar, se retorna a la interfaz anterior.
Interfaz	

Tabla 4 Especificación del Caso de Uso "Gestionar Recurso Humano"

CU- 4	**Gestionar Recurso Humano**
Actor	Jefe de proyecto
Descripción	El caso de uso se inicia cuando el jefe de proyecto desea gestionar los datos de un integrante. El sistema muestra la interfaz de actualización del integrante, donde el jefe de proyecto puede Adicionar un nuevo integrante, escribe el usuario y el sistema genera mediante el directorio UCI los datos útiles para el proyecto y los que este sistema no genere se añaden manualmente (particularmente datos del proyecto); también tiene la posibilidad de eliminar un integrante ya existente o modificar alguno de sus datos. Al final se guardan los cambios en el sistema, terminando así el caso de uso.
Referencia	RF-2; 2.1; 2.2; 2.3; 2.4
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
1. El jefe de proyecto desea gestionar integrantes, selecciona el módulo de "Recursos Humanos" en el menú principal.	2. El sistema visualiza una interfaz con el listado de los integrantes presentes en el proyecto y las opciones relacionadas con los recursos humanos en la parte superior.
3. El jefe de proyecto decide: • Adicionar integrante (ver sección "Adicionar"). • Modificar datos del integrante	

(ver sección “Modificar”). • Eliminar datos del integrante (ver sección “Eliminar”).	
Sección Adicionar	
1- El jefe de proyecto desea adicionar un estudiante y selecciona la opción “Adicionar” presente en la interfaz.	2- El sistema muestra una nueva interfaz para llenar los datos del nuevo integrante.
3- El jefe de proyecto escribe el usuario y selecciona la opción “Generar Datos”.	4- Es sistema utiliza los servicios del directorio UCI para completar los posibles campos.
5- El jefe de proyecto completa los campos que no fueron llenados por el servicio del directorio UCI y da “Guardar”.	6- El sistema guarda el nuevo integrante en la base de datos y devuelve un listado con el nombre y los datos de los integrantes actualizados.
Sección Modificar	
1- El jefe de proyecto desea modificar un estudiante y lo selecciona en la lista además de la opción “Modificar” presente en la interfaz.	2- El sistema muestra una interfaz con los datos del integrante.
3- El jefe de proyecto modifica los datos que debe y presiona “Modificar”.	4- El sistema guarda los datos modificados y actualiza el listado de los datos de los integrantes.
Sección Eliminar	
1- El jefe de proyecto desea eliminar un estudiante y selecciona al estudiante de la lista así como la opción “Eliminar” presente en la interfaz.	2- El sistema muestra la siguiente confirmación: Desea eliminar el integrante señalado?

3- El jefe de proyecto oprime "Si".	4- El sistema elimina el integrante seleccionado y actualiza el listado.
Flujos alternativos	
Sección Adicionar	Línea1: En el caso de que el jefe de proyecto de la opción cancelar, se retornará a la interfaz anterior. Línea2: En caso de que el jefe de proyecto de la opción de guardar los datos sin haber llenado todos los campos previamente el sistema devuelve el mensaje: "No se pudieron guardar los datos." Línea3: en caso de que el jefe de proyecto desee añadir otro integrante selecciona la opción "Limpiar" que vacía el formulario y prosigue a llenar nuevamente el mismo. Línea4: en caso de que el jefe de proyecto introduzca datos incorrectos en el campo de "usuario" el sistema muestra el mensaje "No se pudieron guardar los datos".
Sección Modificar	Línea1: En el caso de que el jefe de proyecto no señale primeramente el integrante a modificar y marque modificar, aparecerá en pantalla el mensaje de corrección: "señale un integrante primero" . Línea2: En el caso de que el jefe de proyecto de la opción cancelar, se retorna a la interfaz anterior.
Sección Eliminar	Línea1: En el caso de que el jefe de proyecto no señale primeramente el integrante a eliminar y simplemente marque eliminar, aparece en pantalla el mensaje de corrección: "señale un integrante primero".

	Línea2: En el caso de que el jefe de proyecto de la opción cancelar, se retorna a la interfaz anterior.
Interfaz	

Tabla 5 Especificación del Caso de Uso "Generar Plan de Trabajo"

CU- 7	**Generar Plan de Trabajo**
Actor	Jefe de proyecto
Descripción	El caso de uso se inicializa una vez que el jefe de proyecto tenga gestionado los recursos humanos y la asignación de tareas a los mismos. El sistema muestra una interfaz con el listado de todos los integrantes del proyecto. El jefe de proyecto selecciona uno y de este verifica el estado de las tareas y le asigna una evaluación, las tareas conforman el plan de trabajo.
Referencia	RF-11; 11.1; 11.2; 11.3; 11.4
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del actor	**Respuesta del Sistema**
1. El jefe de proyecto selecciona en el módulo "Recursos Humanos", "Gestionar".	2. El sistema muestra una interfaz con el listado de los recursos humanos disponibles.
3. El jefe de proyecto selecciona el recurso humano al cual desea verificar y presiona "Plan de trabajo".	4. El sistema muestra una interfaz con los datos del integrante y una tabla con sus tareas pendientes y cumplidas, así como su evaluación y los detalles.
5. El jefe de proyecto tiene la opción para un posterior almacenamiento de exportar el	

informe a formato PDF o formato Excel.	
Flujos alternativos	
1. El Jefe de Proyecto presiona “Plan de trabajo” sin haber seleccionado previamente el recurso humano.	2. El sistema muestra el mensaje de corrección: “Seleccione un integrante primero.”
Interfaz	

3.6 Conclusiones.

Durante el desarrollo del capítulo 3 se da cumplimiento al objetivo fundamental del modelado del dominio en cuestión a partir de la modelación del diagrama de clases del dominio y la descripción de las principales actividades o rasgos de los eventos que se lograron identificar y finalmente la especificación de los requerimientos funcionales y no funcionales de la aplicación.

CAPÍTULO 4

Construcción de la solución propuesta.

4.1 Introducción

En el presente capítulo se documenta el proceso de construcción de la solución propuesta. En él se describe detalladamente la solución en términos de diagramas de clases del diseño partiendo de los requerimientos funcionales y no funcionales; se enuncian los aspectos presentes en el diseño gráfico de la aplicación, así como el diseño del modelo de la base de dato mediante el diagrama Entidad-Relación y el modelo de despliegue. Al finalizar se aborda el modelo de implementación con los diagramas de componentes y las pruebas del sistema propuesto.

4.2 Diagramas de Clases del Diseño de los Casos de Usos Críticos

Los Diagramas de Clases del Diseño describen gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación y la realización de los casos de usos, centrándose en los requisitos funcionales y no funcionales. Normalmente están compuestos por clases, interfaces, relaciones, métodos, navegabilidad y dependencias. Estos diagramas representan una abstracción de la implementación del sistema, y es de ese modo utilizado como una entrada fundamental de las actividades de implementación.

La aplicación MaGIStral está fuertemente relacionada a la estructura del framework de php Symfony, el cual hace uso del patrón de arquitectura Modelo-Vista-Controlador (MVC), por esta razón el diseño queda construido de la siguiente manera:

El Modelo: Representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.

En las aplicaciones donde se hace uso del *framework* Symfony, el acceso y la modificación de los datos almacenados en la base de datos se realiza mediante objetos; de esta forma nunca se accede de forma explícita a la base de datos. Este comportamiento permite un alto nivel de

abstracción y permite una fácil portabilidad. Symfony genera automáticamente todas las clases del modelo según el diseño de los datos implementados sobre algunos de los gestores que son compatibles con él, además genera por cada tabla de la Base de Datos (BD) tres clases, la cuales toman su nombre del título de la tabla correspondiente comenzando con mayúsculas cada palabra que la conforma y sin guiones bajos ('_'), como muestran las siguientes plantillas: *NombreTabla*, *NombreTabla*Table y Base*NombreTabla*. Estas clases se encargan en conjunto de realizar todo el acceso de los datos y la lógica de la aplicación.

*NombreTabla y NombreTabla*Table: Se encargan en conjunto de toda la lógica del negocio.
Base*NombreTabla*: Contiene un conjunto de métodos de acceso y modificación que gestionan el acceso a los datos.
La representación del modelo para cada una de las clases de la BD queda reflejada como se muestra en la figura 5.

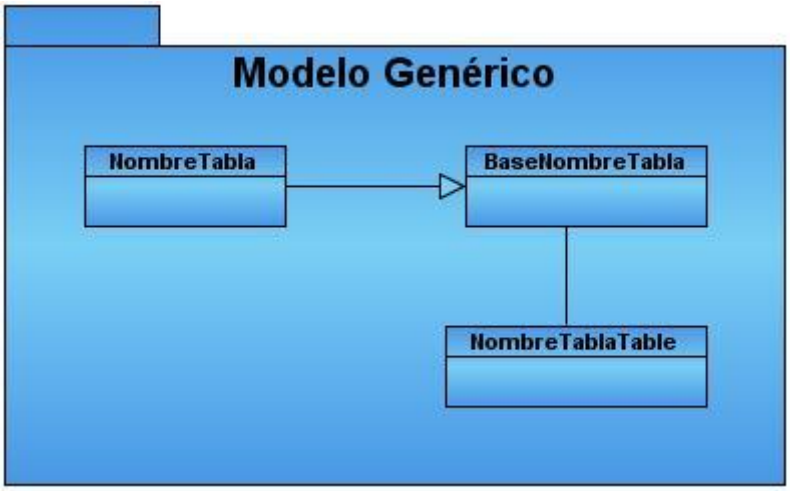

Figura 4 Estructura de las clases modelos generadas por Symfony usando Doctrine.

Para disminuir la complejidad de los diagramas de diseño se decidió representar solamente en el modelo las clases Base*NombreTabla* pues son realmente las que proporcionan información acerca de la representación y organización de los datos en la BD pues a través de ellas se realizan las relaciones entre las tablas. Para ello por cada caso de uso se especifican las tablas que intervienen en este.

El controlador: Se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Las Acciones: Se encargan de obtener los resultados del modelo y definen variables para la vista. Constituyen métodos con el nombre execute*NombreAction* que a su vez hereda de la clase sfActions propia del framework Symfony y se encuentran agrupadas por los módulos de la aplicación, es importante precisar que el prefijo *execute* de las acciones es obligatorio.

El Controlador Frontal: es el único punto de entrada de la aplicación, representa un archivo ".php", carga la configuración y determina las acciones a ejecutarse. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL escrita por el usuario. El archivo ".php" es nombrado en esta investigación index.php.

La Vista: Transforma el modelo en una página Web que le permite al usuario interactuar con ella. Contiene tres partes fundamentales, el *layout* (es la plantilla global), el complemento de las acciones (llamado plantillas) y las páginas con sus formularios.

El layout: Contiene el código HTML que es común en todas las páginas, para no tener que repetirlo en todas las páginas. El contenido de la plantilla se integra en el layout, o si se mira desde otro punto de vista, el *layout* decora la plantilla.

El complemento de las acciones o plantillas: Son las encargadas de con los resultados de la acción, insertarse en el *body* del *layout* y generar finalmente la página Web resultado de la petición de un usuario. Esta plantilla tiene el mismo nombre que el sufijo de la acción correspondiente y termina con la palabra *Success* (Por ejemplo para una acción *executeIndex* existirá una plantilla *indexSuccess)*. En este caso se utiliza una sola *Success* para todos los módulos que *IndexSuccess.php.*

Paginas Clientes y Formularios: son las páginas que se generan finalmente y con las que el usuario interactúa.

Se utiliza ExtJS que es una librería JavaScript que incorpora la tecnología AJAX para construir aplicaciones complejas con mayor eficiencia. Al utilizar dicha librería se logra un balance entre Cliente y Servidor puesto que se distribuye la carga de procesamiento.

ExtJS no viene incluido por defecto en Symfony por lo que hay que realizar una integración entre los dos *framework* mediante modificaciones de archivos de configuración para garantizar el correcto funcionamiento de la aplicación. Para esto se copia la librería de ExtJS en la raíz del directorio Web del proyecto así como los ficheros JavaScript (JS), las imágenes y los archivos CSS que contienen las configuraciones visuales del sistema en las carpetas js, images y css de dicho directorio respectivamente; luego para que Symfony pueda tener acceso a estos ficheros es necesario dirigirse al archivo view.yml, ubicado en el directorio apps/*nombredelaaplicación*/config/ dentro de la estructura del framework de php en el cual se especifican los nombres de los JS y CSS de ExtJS y del sistema que se utilizarán en las plantillas; por último se incluyen los archivos mencionados anteriormente en el layout general y en las Success de cada módulo de la aplicación según se requiera mediante los *helpers* de Symfony *include_javascripts()* e *include_stylesheets(),* los cuales usarán la configuración realizada en el archivo view.yml, también puede llevarse a cabo utilizando los *helpers* use_javascript('archivo.js') y use_stylesheet('archivo.css') si se requiere de una inclusión personalizada; ó la combinación de ambos métodos.

Para la utilización de esta librería en el diagrama se definió el uso de un paquete como muestra la figura 6.

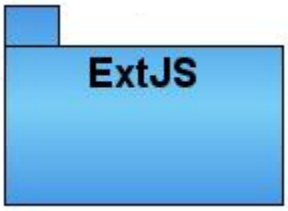

Figura 5 Paquete ExtJS.

Dentro del paquete "JS del caso de uso" están definidas los ficheros JavaScript que son utilizados por el caso de uso en cuestión.

De la figura 7 a la 11 se muestran los Diagramas de Clases del Diseño para cada uno de los casos de uso críticos de la investigación, el resto se encuentra en el Anexo 1.

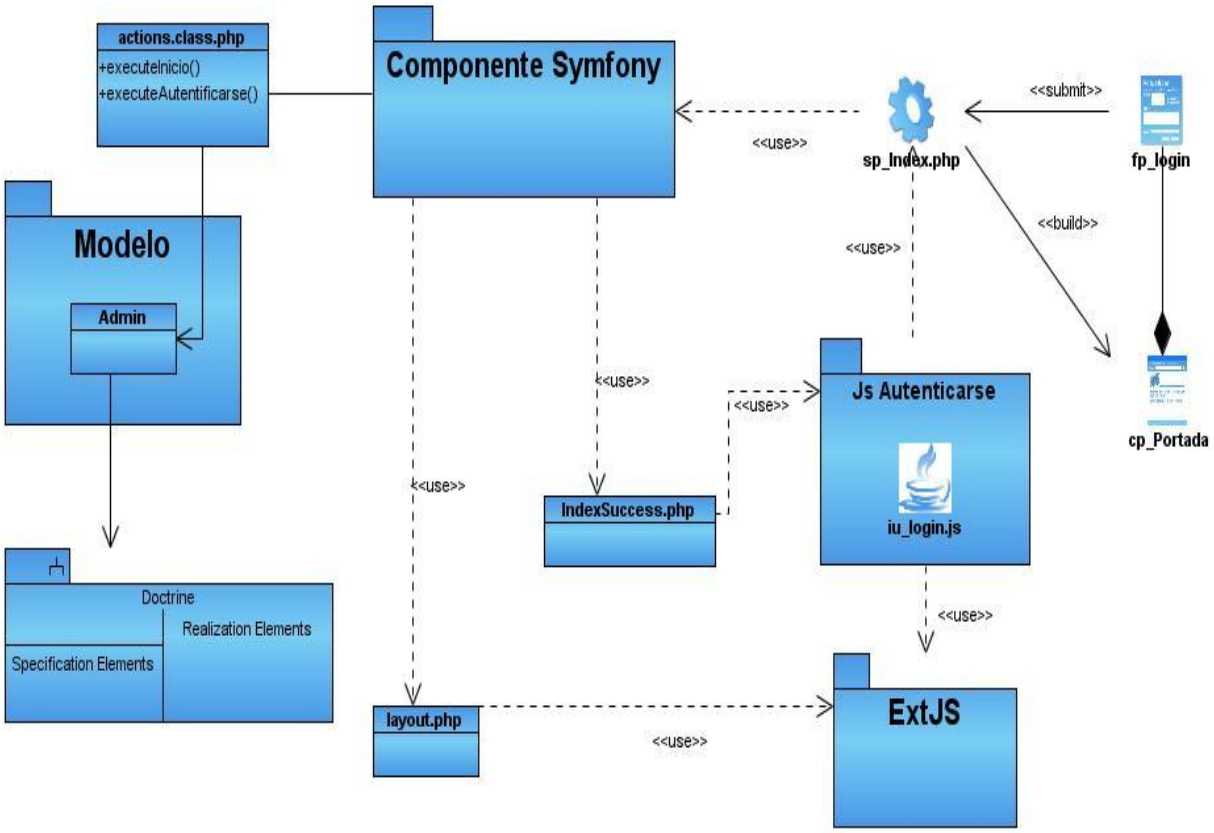

Figura 6 Diagrama de Clase de Diseño "CU Autenticarse".

De manera general, el flujo de información según lo representado en los diagramas funciona de la siguiente forma:

El controlador frontal recibe una petición (generalmente una URL), luego de recibirla, y ayudado por los componentes de Symfony (paquete que encapsula el resto de los componentes internos del *framework*), determina el módulo y la acción que debe invocarse (a través de un proceso llamado enrutamiento), luego de invocada, la acción utiliza el modelo y almacena los resultados que serán accesibles a través de la vista. Para devolverle el resultado a las acciones, el modelo utiliza un ORM (*"object-relational mapping"*) que en el caso del *framework* seleccionado se trata de Doctrine, quien se encarga de todo el proceso de acceso y abstracción de los datos. Una vez realizado este proceso, en la Success general donde están definidos los ficheros JavaScript (JS) indispensables para el funcionamiento de la aplicación, se hace un llamado a

interfaces específicas de cada caso de uso, la relación que va a existir entre el modelo, la vista y el controlador está dada por las peticiones que se hacen al servidor desde los ficheros JS por un proceso de enrutamiento hacia el controlador frontal, identificando el módulo y la acción para dar respuesta a la petición a través del propio controlador frontal y la página cliente que se genera.

4.3 Principios de Diseño.

4.3.1 Patrones de diseño aplicados.

Desde que comienza a realizarse el análisis del ciclo de vida del proyecto MaGIStral es posible avizorar problemas que han sido identificados en otros contextos de la filosofía orientada a objetos. Estos problemas son por ejemplo, la necesidad de implementar el sistema de manera que: no exista sobrecarga de funciones en los componentes ya que esto conlleva a la disminución del rendimiento del sistema; se distribuyan las funcionalidades en las clases para que cada objeto tenga sólo las labores que es capaz de realizar garantizando así un alto nivel de reutilización de las mismas.

Cuando se llega al flujo de trabajo de diseño la mayoría de estos problemas encuentran solución a partir de la aplicación de los patrones de diseño. Estos patrones ofrecen un lenguaje estándar para reconocer, definir, y describir dichas soluciones en la creación de la aplicación.

GRASP (Patrones Generales de Software para Asignar Responsabilidades).

Creador

En Symfony la capa del controlador contiene el código que une la lógica del negocio con la presentación. Este cuenta con varios componentes entre los que se identifican las acciones que son los métodos que, utilizan el modelo y definen variables para la vista. En la aplicación en cuestión se cuenta con un fichero de *tipo actions.class.php* por cada módulo. Estas acciones se ejecutan en las clases "*frontalActions", "gestionActions" y "reporteActions*" de los módulos "frontal", "gestión" y "reporte" respectivamente. En las *actions* mencionadas se crean los objetos que representan las entidades evidenciando que son "creadoras" de dichas clases.

Experto

Es uno de los patrones que se emplean al trabajar con el Framework Symfony, se evidencia en la inclusión de Doctrine para mapear la base de datos. Este último, genera las clases para la gestión de la información de las tablas de dicha base de datos con las responsabilidades debidamente asignadas según el patrón Experto. Cada una de estas clases cuenta con un conjunto de funcionalidades que las convierte en expertas de la información de la tabla a la que representa. Además la aplicación de este patrón permite mover parte de la lógica del negocio hacia las clases modelo evitado así la sobrecarga del controlador.

Alta Cohesión

Una de las características principales del Framework Symfony es la organización del trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión posibilitando que a estas clases le sean asignadas responsabilidades estrechamente relacionadas, para que realicen un trabajo excesivo. Por ejemplo, las clases "*frontalActions*", *"gestionActions" y "reporteActions*" contienen responsabilidades estrechamente relacionadas, encargadas de controlar las acciones de sus respectivas plantillas. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo.

Controlador

Las clases *"frontalActions", "gestionActions" y "reporteActions*" son las controladoras del sistema y su función es servir de enlace entre la vista y el modelo. Precisamente es esta la evidencia de la existencia del patrón Controlador en el proyecto, aunque es común que este se implemente asignando el control a una sola clase, de este modo se asegura que no se sobrecargue el controlador. Symfony implementa además el patrón *Front Controller* (Controlador Frontal), en el cual existen varias clases controladoras que forman un flujo para atender las peticiones del usuario y de otras clases.

Bajo Acoplamiento

El patrón Bajo Acoplamiento determina que una clase no dependa de muchas otras clases, lo que posibilita que no se afecten por cambios en otros componentes. Un ejemplo de cómo Symfony implementa este patrón es que las clases *"frontalActions", "gestionActions" y "reporteActions"* heredan solamente de *sfActions* para lograr un bajo acoplamiento de clases.

Además en el modelo se pueden encontrar relaciones de herencia entre las clases como por ejemplo ActGeneral que hereda de BaseActGeneral.

Patrones GOF (Gang of Four, en inglés)

Creacionales:

Singleton (Instancia única)

El patrón Singleton garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En las *actions* el contexto (que se obtiene mediante sfContext::getInstance()) almacena una referencia a todos los objetos que forman el núcleo de Symfony y puede ser accedido desde cualquier punto de la aplicación. Luego en el controlador frontal hay una llamada a *sfContext::createInstance($configuration)->dispatch().*

Estructurales:

Decorator (Envoltorio)

El patrón *Decorator* añade funcionalidad a una clase dinámicamente. El archivo layout.php, que también se denomina plantilla global en Symfony, están incluidas todos los ficheros JavaScript que son común para todas las páginas de la aplicación, para no tener que repetirlo en cada una. El contenido de la plantilla se integra en el *layout*, o si se mira desde el otro punto de vista, el *layout* decora la plantilla. Este comportamiento es una implementación del patrón de diseño llamado.

4.3.2 Estándares de la interfaz de la aplicación.

Para el diseño de la interfaz de la aplicación se emplea la librería de JavaScript ExtJS descrito en el capítulo 2.

La navegabilidad de la aplicación está organizada de la siguiente manera: inicialmente aparece la página de autentificación que en dependencia de los datos que introduzca el usuario da lugar a la interfaz principal del sistema con los permisos y la configuración correspondientes para ese usuario. La vista principal está dividida en cuatro regiones: superior, izquierda, central y derecha; en la región superior o norte se muestra el nombre de la aplicación, en la región izquierda u oeste se muestran los módulos con las operaciones derivadas de los requisitos que el sistema debe cumplir, por ejemplo: registrar, mostrar, eliminar o modificar los datos de un

integrante, estación de trabajo, tarea u otros; la región central es el área de trabajo, donde se muestran gradualmente las interfaces de gestión de los recursos humanos y materiales, tareas, actividades generales, tickets, así como reportes y planes de trabajos. Por último, en región derecha o este se muestran las actividades generales vigentes en el proyecto en conjunto con los tickets del usuario. La figura 12 muestra lo descrito anteriormente.

Figura 7 Prototipo de Interfaz de la Aplicación.

4.4 Diseño de la Base de Datos

El diseño de la base de datos es de vital importancia ya que mediante el mismo se almacenan los datos de la aplicación. Una correcta selección de las tablas que forman parte de la base de datos, así como de la información que contiene cada una y sus relaciones, y teniendo en cuenta las prácticas adecuadas de normalización, proporcionan un rendimiento del sistema adecuado a las exigencias del cliente.

El diseño de la base de datos se obtiene de las clases del diseño y de las realizaciones de los casos de uso. El principal artefacto de esta actividad es el Modelo de Datos que describe la representación lógica y física de los datos persistentes.

4.4.1 Modelo Lógico de Datos.

Las clases persistentes con aquellas que mantienen su valor a través del tiempo. El diagrama de clases persistentes expresa los aspectos relacionados con el almacenamiento de datos del sistema. La información con la que el sistema trabaja es necesario guardarla en un medio permanente, es por eso que utilizamos las clases persistentes ya que estas proporcionan un almacenamiento físico seguro de los datos, protegiéndolos en caso de fallo del sistema, evitando la pérdida de la información. En la figura 13 se muestra el diagrama de clases persistentes para la aplicación MaGIStral.

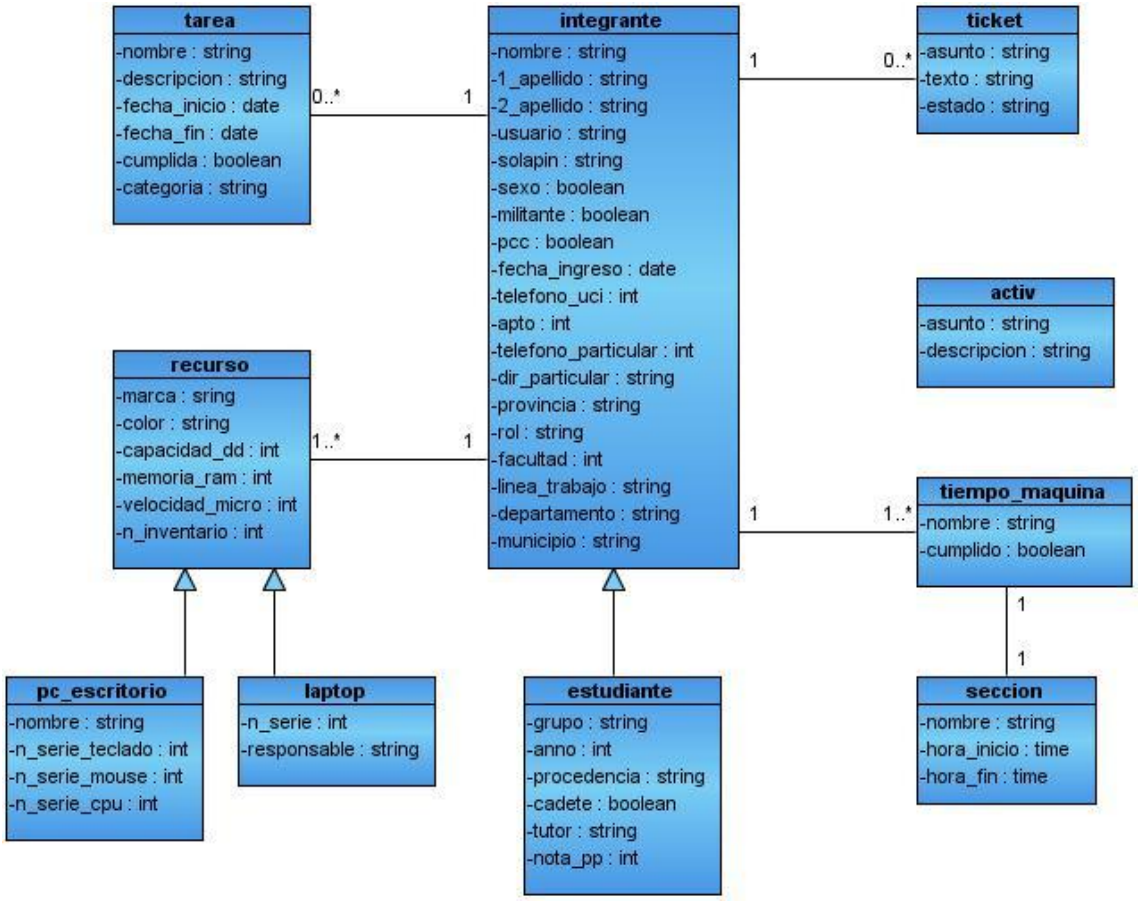

Figura 8 Diagrama de Clases Persistentes.

4.4.2 Modelo Físico de Datos.

Una vez obtenido el modelo relacional son realizadas las transformaciones necesarias con el fin de obtener un diseño de bases de datos correcto, para lograrlo debe cumplir con los siguientes parámetros:

- **Normalización hasta 3ra Forma Normal o superior.**
  - ✓ 1ra Forma Normal que plantea que los dominios no tienen elementos que a su vez sean conjuntos, Es decir que todos los atributos de la relación tengan valores atómicos, haciendo que no existan los atributos multievaluados, compuestos y sus combinaciones.
  - ✓ 2da Forma Normal donde los atributos no llaves dependen funcional y completamente de la llave primaria.
  - ✓ 3ra Forma Normal en la que se eliminan las dependencias transitivas de atributos no llaves respecto a la llave primaria.
- **Propiedades adicionales de un buen diseño.**
  - ✓ Descomposición y reunión sin pérdidas; descomposición y conservación de las dependencias:

    Al descomponer un esquema relacional se utiliza el operador de proyección, la operación inversa utiliza el operador de reunión o *join*. Para garantizar la propiedad de *join* sin pérdida de información se necesita que la operación de proyección sea reversible, esto significa que a partir de los nuevos esquemas obtenidos con la proyección, seas capaz de obtener el esquema original de lo contrario hay pérdida de información.

El modelo relacional de la base de datos del proyecto MaGIStral que se muestra en la figura 14 es correcto ya que la normalización fue hecha hasta la tercera forma normal y además cumple con las propiedades adicionales que garantizan un diseño correcto (Propiedad de descomposición y reunión sin pérdidas y Propiedad de descomposición y conservación de las dependencias).

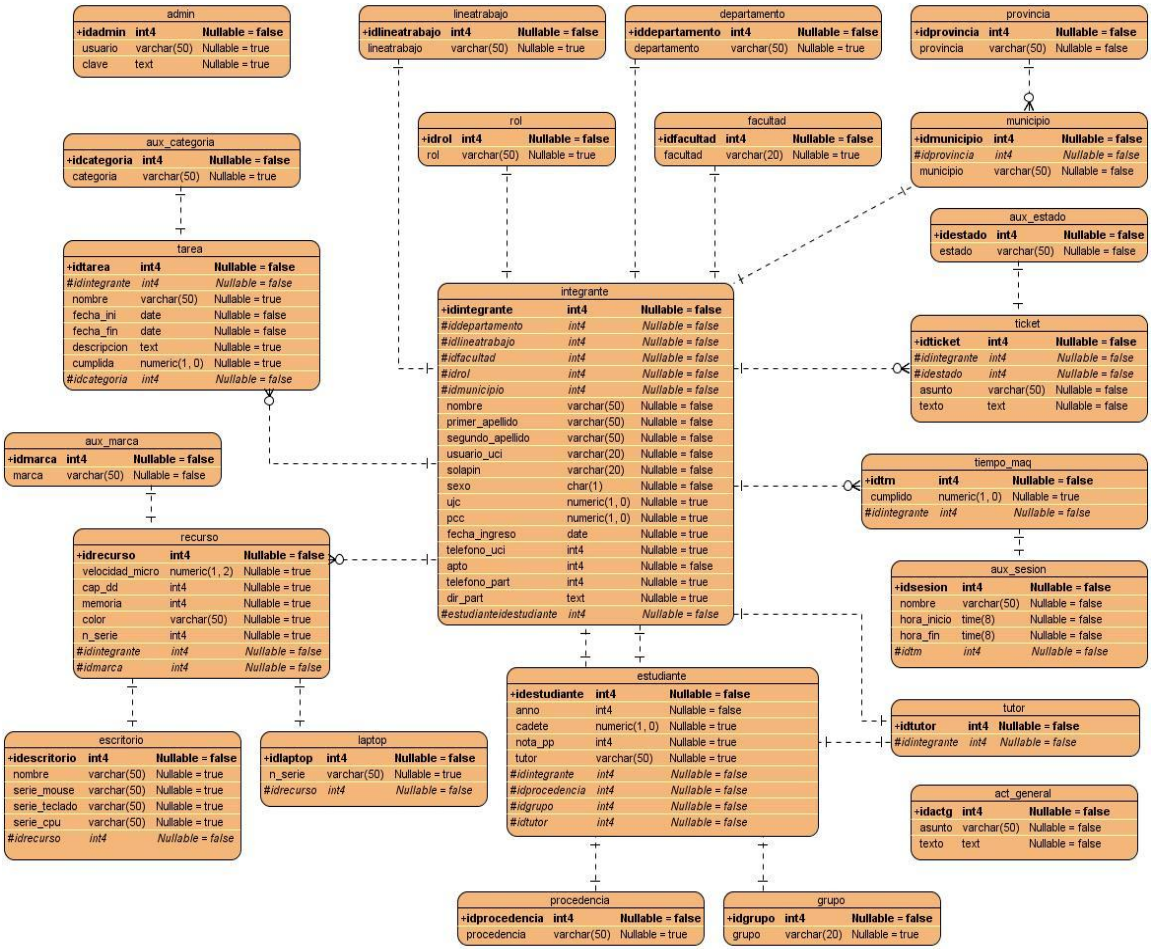

Figura 9 Diseño de la base de datos: Modelo Relacional.

4.5 Modelo de Implementación

En el flujo de trabajo de implementación se tiene como entrada el modelo de diseño. Los elementos de este último se describen e implementan en términos de componentes como se muestra en la figura 15 y se estructuran en forma de paquetes como se presenta en la figura 16 para lograr una mayor organización del trabajo. Luego todo esto se ordena de acuerdo a los nodos específicos determinados en el modelo de despliegue del modo que se especifica en la figura 17. Cada uno de los diagramas que componen el modelo de implementación son explicados respetando el lenguaje de modelado UML 2.0 de manera que las descripciones sean convergentes con lo que sugiere cada elemento o relación incluida en dichos diagramas.

4.5.1 Diagrama de Componente

En la figura 15 está representado el diagrama de componentes de la aplicación como parte fundamental del modelo de implementación. Con el fin de mostrar una vista de la estructura de MaGIStral se dividen los paquetes respetando la arquitectura propuesta (MVC) y las dependencias lógicas entre estos. Además se consideran los requisitos relacionados con las facilidades de desarrollo que brinda el uso del *framework* Symfony, las restricciones impuestas por el lenguaje de programación PHP5 en cuanto a la orientación a objetos y las tendencias en el desarrollo Web con el uso de Doctrine y ExtJS.

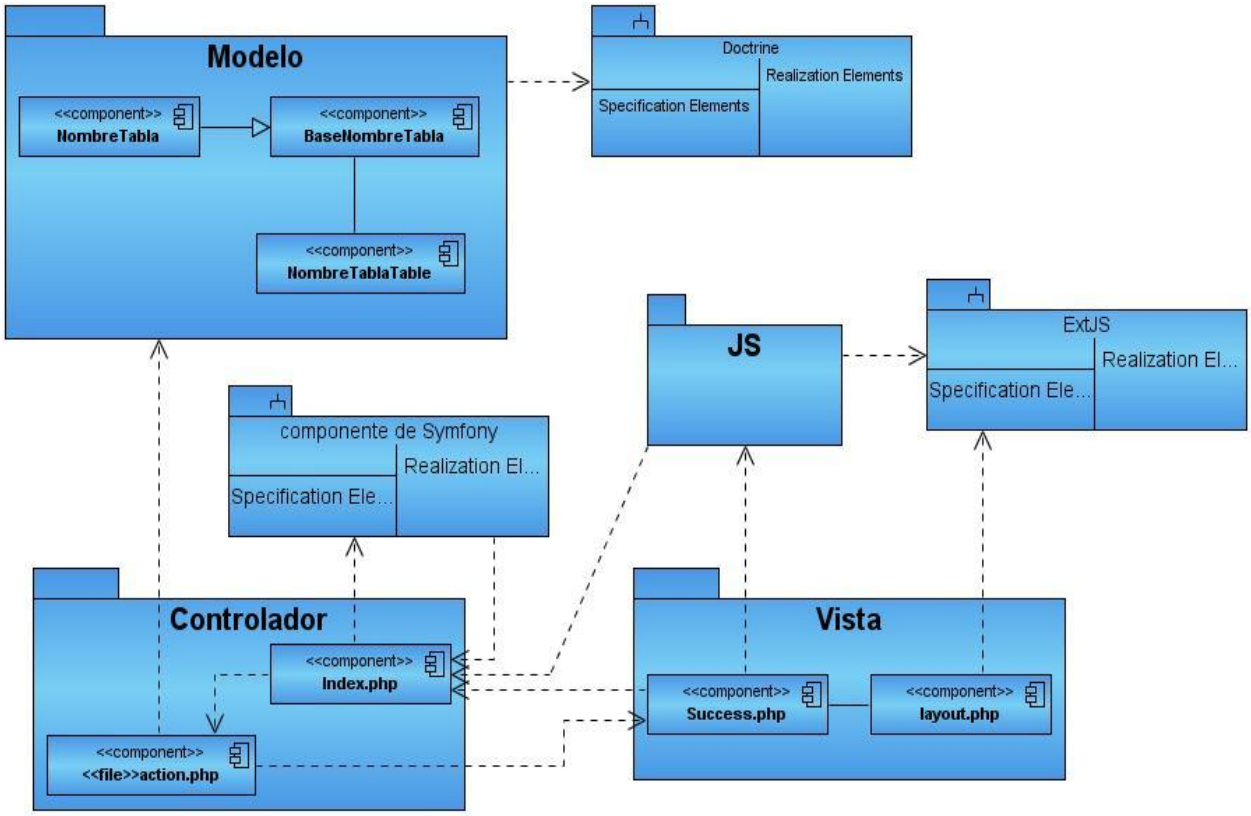

Figura 10 Diagrama de Componentes

De manera general la implementación del sistema se realizó de la siguiente manera:

Las peticiones Web realizadas por el cliente son manejadas por el controlador frontal del *framework Symfon*y, el cual constituye el único punto de entrada de la aplicación y está definido para dos entornos: el de producción en el fichero "*index.php*" y el de desarrollo en el archivo

"*frontend_dev.php*". Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL escrita por el usuario. Por ejemplo, la siguiente URL será procesada por el fichero index.php (que es el controlador frontal) y traducida en una llamada a la acción "*action*" del módulo "gestión": http://localhost/index.php/gestion/action.

En el fichero *action.class.php* del módulo gestión está definida la clase "*gestionActions*" que hereda de la clase sfActions propia del framework Symfony. Esta clase ("*gestionActions*") contiene los métodos que dan funcionalidad a cada caso de uso definido en el diseño del sistema. Estos métodos, que por regla del framework tienen como sintaxis "*executeNombreMetodo(),* implementan las funcionalidades de insertar, eliminar, modificar y mostrar, para cada caso de uso de gestión, ya sea para Recurso Humano, Recurso Material, Tarea, Actividad o Ticket, excepto en el caso de "GestionarTicket" que no incluye el método modificar. Por ejemplo, para Gestionar Tarea los métodos quedan definidos de la siguiente manera: *executeMostrarTarea(), executeInsertarTarea(), executeEliminarTarea()* y *executeModificarTarea* y lo mismo ocurre para los demás casos de usos.

Al invocar a la acción "executeInsertarTarea()" como ejemplo, el flujo de datos ocurre de la siguiente manera: primero el usuario llena los datos dentro de un formulario llamado Insertar tarea definido en el fichero tarea.js; luego el método "executeInsertarTarea()" se encarga de crear un objeto de tipo Tarea y le asigna a cada propiedad del mismo, los datos necesarios recibidos del formulario para posteriormente almacenarlo en la base de datos. Sin embargo un objeto no puede ser almacenado en una base de datos de forma directa ya que la mayoría de los sistemas de bases de datos modernos son relacionales y no son compatibles con el esquema de objetos que presenta la Programación Orientada a Objetos, paradigma sobre el cual está desarrollado Symfony, para resolver esta incompatibilidad los objetos de la clase Tarea deberán tener un comportamiento persistente, es decir que la información que encapsulan pueda ser almacenada de forma permanente, en este caso en la base de datos que utiliza el sistema.

El framework en su versión 1.4.1 incorpora el ORM (*Object Relational Mapping*) Doctrine 1.2 el cual lleva a cabo la correspondencia entre las clases del Modelo de Dominio y las tablas de la base de datos a través del patrón de diseño *Active Record*. Este patrón es implementado en la

clase *sfDoctrineRecord* la cual proporciona una interfaz a las clases derivadas con los métodos *save*() y *delete*(), operaciones que permiten insertar, actualizar y eliminar información en un registro de una tabla de la base de datos a través de las instancias del las clases del Modelo de Dominio. Es por esto que la clase de ejemplo Tarea así como las demás clases del Modelo de Dominio de Magistral debe heredar de la clase BaseTarea que a su vez hereda de *sfDoctrineRecord* para poder incorporarle a sus instancias un comportamiento persistente.

Para cada modulo de Magistral (“frontal”, “gestión” y “reporte”) hay un fichero llamado IndexSuccess.php que es en este donde referenciados los ficheros JavaScript (JS) indispensables para el funcionamiento de la aplicación. En el caso del modulo gestión son: iu_rh.js, iu_rm.js, iu_tarea.js, iu_tk.js, iu_act.js; que se muestran en la figura 16. Es en ellos donde están definidos todos los formularios y las vistas para cada caso de uso de gestión ya sea insertar, modificar o eliminar. Para esto se hace un llamado a interfaces específicas de cada caso de uso. La relación que va a existir entre el modelo, la vista y el controlador esta dado por las peticiones que se hacen al servidor desde el archivo JavaScript por un proceso de enrutamiento hacia el controlador frontal, identificando el módulo y la acción para dar respuesta a la petición a través del propio controlador frontal y la pagina cliente que se genera.

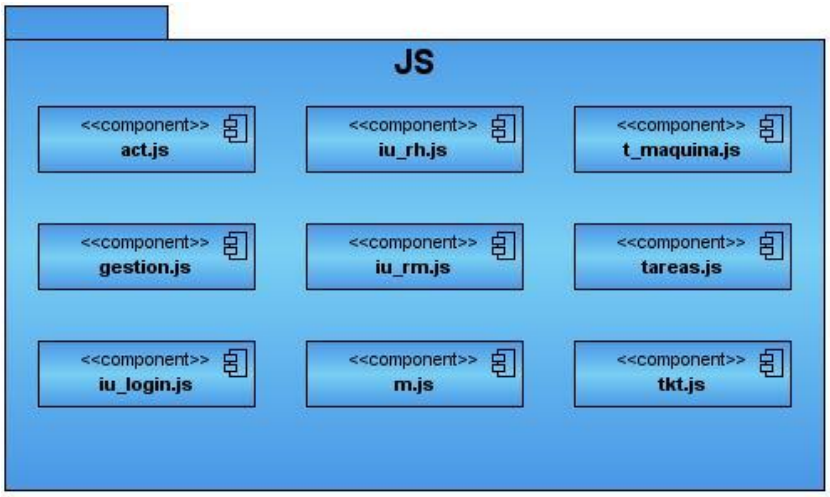

Figura 11 Componentes del paquete JS.

De esta manera ha quedado explicado el funcionamiento del módulo “gestión” que junto al módulo “frontal” (fundamentado en el epígrafe 4.6.1) y al módulo “reporte” constituyen la forma en que está dividido MaGIStral a partir de la estructura que propone Symfony para una mejor distribución de las aplicaciones. El módulo “reporte” incluye lo relacionado con la implementación de Generar Reporte y Plan de Trabajo pero de manera general el flujo de las peticiones está dirigido en el mismo sentido que el explicado para los paquetes de gestión.

4.5.2 Modelo de Despliegue

El Modelo de Despliegue constituye otro componente primordial en el modelo de implementación, al igual que el diagrama de componentes, como se enuncia en el epígrafe 4.5.1. Se utiliza en este trabajo del modo que lo define la metodología RUP ya que con él se visualiza la distribución de los componentes de software en los nodos físicos que constituyen los servidores y ordenadores de los clientes, así como los protocolos de conexión entre ellos.
El caso particular del presente trabajo sería de la siguiente manera: las PC clientes se conectan al servidor Web mediante el protocolo http, el cual para dar cumplimiento de las tareas, está conectado al servidor de la base de datos y al servidor del directorio uci mediante el protocolo TCP/IP. Esto queda reflejado en la figura 17.

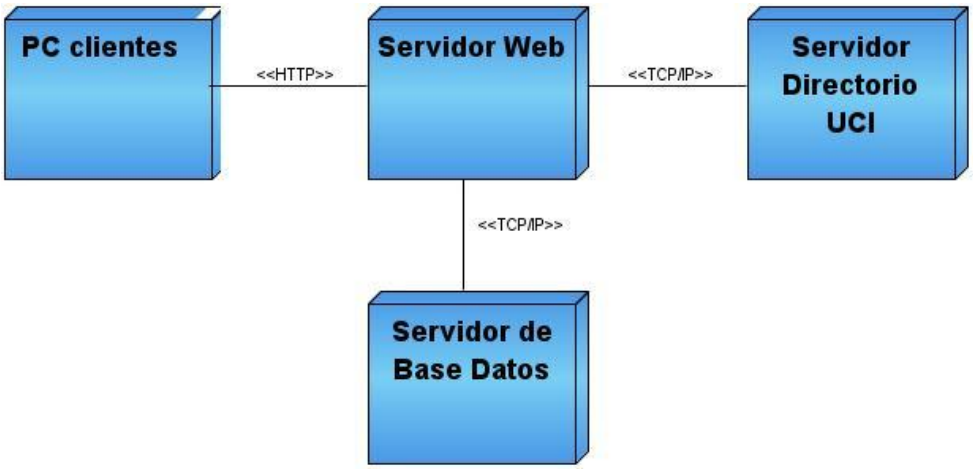

Figura 12 Diagrama de Despliegue.

4.6 Seguridad del sistema

La seguridad de MaGIStral se define como un esquema de permisos sobre algunos de los elementos de la aplicación Web por grupos de usuarios y roles determinados. Existen

diferentes módulos, botones o *links* disponibles que deben ser controlados. Con la funcionalidad de seguridad es posible definir quienes tienen acceso a qué elemento de la Web a partir del perfil de seguridad que tenga cada usuario.
Ejemplo: El módulo de administración debe ser sólo visible para administradores. Sin embargo, hay otros elementos comunes que serán vistos por todos los usuarios tales como la página principal o el módulo de gestión de tickets.

4.6.1 Estructura de usuarios

Existen dos tipos de usuarios válidos para el sistema, estos son el administrador local y el integrante del proyecto; el primero contará con una tabla en la base de datos en la que se almacenarán el usuario y una clave de acceso encriptada doblemente mediante el algoritmo md5, este usuario tendrá acceso total a cualquier funcionalidad de la aplicación, ya que está diseñado especialmente para el jefe de proyecto en caso de que falle la conexión en la red física de la infraestructura en la que esté desplegado el software y no se pueda acceder al servicio de autenticación del servidor ldap.uci.cu; mientras que los integrantes del proyecto para poder ingresar en el sistema deberán introducir los datos de su cuenta en el dominio uci para luego proceder a la autenticación mediante el servidor ldap.uci.cu, como cada miembro de la universidad tiene una cuenta en el dominio uci, entonces el usuario deberá existir previamente en la base de datos de la aplicación para poder acceder a la misma.

4.6.2 Autenticación

MaGIStral es una aplicación desarrollada haciendo uso del *framework Symfony* por lo cual está distribuida en módulos, de los cuales uno de ellos (frontal) está dedicado a la seguridad mediante el proceso de autenticación de usuarios que se describe a continuación.
El módulo “frontal” hace uso de la clase *Login* definida en el archivo lib/Login.php, que se encarga de conectar la aplicación con el directorio activo de la universidad en el proceso de autenticación del usuario, mediante el cual se controla el acceso individual al sistema a partir de la identificación de los mismos utilizando las credenciales provista por ellos. Luego de una autenticación exitosa, el usuario es asociado con varias variables de sesión de php que garantizarán su identificación en cada momento de que utilice la aplicación, estas variables son “nombre”, “categoria”, e “identificador”; la primera almacena el nombre completo del usuario, la segunda si el usuario es profesor o estudiante y la tercera el identificador del registro que le

corresponden en la tabla “integrante” de la base de datos. La variables de sesión son manejadas por la clase sfUser de Symfony, la cual amplia las funcionalidades de dichas variables, ejemplo la creación de credenciales a través de las cuales se pueden formar grupos de usuarios así como establecer permisos determinados a distintas funcionalidades.

4.6.3 Seguridad de las acciones

Debido a la heterogeneidad de los roles que desempeñan los usuarios, es necesario restringir el acceso de algunas funcionalidades que no deben ser explotadas por integrantes que no tienen necesidad de acceder a dichas funcionalidades por el rol asignado dentro del proyecto. Para esto se hace uso del sistema de seguridad que brinda Symfony mediante la configuración del archivo “*security.yml*; en caso de que se determine que solamente accedan a las acciones de un módulo los usuarios autenticados se edita el archivo “security.yml” del modulo y se escribe “is_*secure*” y en caso de que se quiere restringir alguna acción en específico se adiciona el nombre de la acción, por ejemplo “insertarIntegrante:” y luego “is_*secure*”, seguido de las credenciales que le sean otorgadas al usuario, por ejemplo “administrador”.

4.7 Prueba del sistema propuesto

Dado que MaGIStral es un software producto de un proceso ingenieril, del cual se conocen las funcionalidades específicas para las cuales fue diseñado, se pueden llevar a cabo pruebas que demuestran que cada función es completamente operativa. Este enfoque se refiere a la pruebas de caja negra que se llevan a cabo sobre la interfaz de la aplicación. Por esta razón los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Estas pruebas examinas algunos aspectos del sistema sin tener mucho en cuenta la estructura interna del software.

Objetivo

El objetivo de realizar este tipo de prueba al sistema, es detectar el incorrecto o incompleto funcionamiento de este, así como los errores de interfaces, rendimiento, errores de inicialización y terminación.

Alcance

El proceso de pruebas de caja negra se va a centrar principalmente en los requisitos funcionales del software para verificar el comportamiento de la interfaz gráfica, su interacción con el usuario y la calidad funcional.

Descripción

Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar. El método que se propone para diseñar los casos de pruebas es una variante del de particiones equivalentes, propuesto por RUP y al que se le han añadido algunas consideraciones, este método consta de 3 pasos fundamentales y usa como artefacto de entrada los casos de usos.

Casos de Prueba

Para verificar que se cumplan los requerimientos funcionales establecidos, se realizan las pruebas a los casos de uso más críticos. Estas se hacen con el objetivo de evaluar la interacción del usuario con el sistema, dependiendo de las funcionalidades a las que tenga acceso el usuario y además se verifica que cada una de estas se corresponda, con las descripciones de cada uno de los casos de uso del sistema realizadas. Los casos de pruebas se encuentran en el anexo 2.

4.8 Conclusiones

El presente capítulo fue confeccionado con el objetivo de reflejar en un grado reducido pero claro, los elementos fundamentales de la fase de construcción del software MaGIStral. En la medida que se ha desarrollado el proceso mencionado se han identificado entre estos elementos momentos claves de su ciclo de vida que se revierten en conclusiones importantes a las que arribar, como por ejemplo:

- El conjunto Lenguaje-Herramientas-Metodología escogido para el desarrollo de MaGIStral, ha sido explotado en esta fase del desarrollo de la aplicación demostrando

que el nivel de acoplamiento entre ellos proporciona facilidades de rendimiento, seguridad, modularidad que responden directamente a las especificidades de los requerimientos no funcionales

- La modelación del flujo de las peticiones en los diagramas de clases del diseño así como los diagramas de componente y despliegue del modelo de implementación proporcionan una visión sencilla y confiable de lo que puede resultar un paquete de código complicado de entender.

CONCLUSIONES

- MaGIStral como sistema integral personalizado, implementado con una integración de tecnologías novedosa, es una herramienta factible que abre las puertas a otros desarrolladores interesados en el tema para que puedan realizar herramientas similares aplicadas a otros proyectos.

- Con la realización del software MaGIStral se pone a disposición del proyecto GeneSIG una herramienta que permite automatizar los procesos de gestión de los recursos implicados en él.

- La aplicación elimina el problema que supone la utilización del Excel para el manejo de los datos de los recursos del proyecto, ya que cuenta con una base de datos segura y confiable para el almacenamiento y administración de los mismos.

- La realización de este trabajo demuestra que la metodología RUP es la guía idónea para desarrollar sistemas de gestión de recursos en ambiente Web.

- El trabajo ingenieril inherente a la aplicación ha permitido documentar de manera detallada los artefactos generados en cada flujo de trabajo propuesto por la metodología escogida, con el objetivo de reutilizar toda esta información en futuras versiones o revisiones.

- El desarrollo del sistema mediante el patrón arquitectónico modelo MVC, garantiza la flexibilidad y extensibilidad del mismo, evitando de esta forma la rápida obsolescencia.

RECOMENDACIONES

- Aplicar el sistema MaGIStral en el proyecto GeneSIG como herramienta de gestión de recursos y posteriormente extenderlo a otros proyectos de la facultad.

- Continuar añadiéndole nuevas funcionalidades al sistema y obtener mejoras en futuras versiones como envío por correo de notificación de tarea a un integrante o notificación de cumplimiento de tareas, logrando así mejorar su calidad y adecuarlo cada vez más a las necesidades exigidas por el cliente.

REFERENCIAS BIBLIOGRÁFICAS

- Danay Hernández León y Deigly Arteaga Alarcón, «Procedimiento para Líderes de Proyectos». (Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas), Julio del 2008.
- Bolstad, P. «GIS Fundamentals: A first text on Geographic Information Systems», Second Edition. White Bear Lake, 2005.
- Yadira Fernández Guerra, «Diseño de la plataforma soberana LiberGIS para el desarrollo de Sistemas de Información Geográfica en el Polo Geoinformática.». (Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas), Mayo del 2009.
- Lic. Rodolfo Caldera Mejía, «Planeación Estratégica de Recursos Humanos» Estrategika-Consultoria, S.A. Diciembre del 2004.
- Henry Cruz Mulet y Jessie Castell González, «Propuesta de metodología de Gestión de Proyectos de Software, Fase Conceptual.». (Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas), Junio del 2008.
- George Suhanic: Computer-Aided Project Management, ISBN 0-19-511591-0. 2003.
- Manual De Usuario Egroupware, [en línea] disponible en www.betera.es/beteraPortal/RecursosWeb/DOCUMENTOS/1/1_862_1.pd. Universidad Tecnológica de Puebla. 2001.
- Maybel Díaz Capote y Yurkiel Martínez Rivero, «Sistema de Control de los Recursos del Proyecto.». (Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas), Junio del 2008.
- Cynthia K. West, Ph.D. «Four Solutions to Prevent Project Failure». Enterprise Systems Journal, 3 de Julio del 2007.
- Business Wire, «Metafuse Announces Project Insight 3.1.1 - Web-Based Project Management Application Offers Improved Features For Greater Visibility and Efficiency», 17 de Septiembre del 2002.

- Sara Alvarez, «Características principales de este tipo de arquitectura de cara a base de datos». [En línea] http://www.desarrolloWeb.com/articulos/arquitectura-cliente-servidor.html, 30 de Agosto del 2007.
- Javier Eguíluz Pérez. «Introducción a JavaScript», 25 de marzo del 2009.
- Yandier Mayo Leyva, «Implementación de las validaciones en JavaScript para la AGR. Adaptación al framework ExtJS». (Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas), Junio del 2008.
- Fabien Potencier, François Zaninotto. «Symfony, la guía definitiva.». Capítulo 1. Introducción a Symfony. 13 de julio del 2008.
- Colectivo de autores «El Lenguaje Unificado de Software». Manual de Referencia (libro), 2000.
- Oktaba, Hanna y otros. Método de Evaluación de procesos para la industria de software. (libro), 2004
- Darlin Díaz Rodríguez, «Diseño del módulo de administración y configuración de mapas en la plataforma LiberGIS». (Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas), Junio del 2009.
- JACOBSON, Ivar; RUMBAUGH, James; BOOCH, Grady, “El proceso unificado de desarrollo”.2000. Addison Wesley. capítulos 8 Páginas 165-181, 185-204.
- Guillermo Restrepo González. El concepto y alcance de la gestión tecnológica. Medellín: Revista Facultad de Ingeniería, 2000.
- Alanis, Macedonio. Planeación de Recursos Empresariales. Monterrey: 2005.
- Yoenis Pantoja Zaldívar, «Plataforma Soberana para el Desarrollo de Sistemas de Información Geográfica» (Informe), Universidad de las Ciencias Informáticas. 2008.
- Aníbal de la Torre. Lenguajes del lado servidor o cliente. Lenguajes del lado servidor o cliente. 2008.
- Rolando Alfredo Hernández. Curso básico de visual paradigm. Curso de Postgrado. Ciudad de la Habana: 2005.

BIBLIOGRAFÍAS

- Carlos Andrés Morales Machuca, «Estado del Arte: Servicios Web» (Informe formato pdf.), Universidad Nacional de Colombia, camoralesm@unal.edu.co,
- Javier Eguíluz Pérez, «Introducción a JavaScript», www.librosWeb.es, 25 de marzo del 2009.
- Leticia González Folgueira, Miguel R. Luaces, «Una aplicación Web para la gestión empresarial orientada a proyectos de una PYME» (Informe formato pdf.), Laboratorio de Bases de Datos, Universidad de la Coruña, Campus de Elviña, 15071, A C.
- María José Ortín, Jesús García, «El Modelo del Negocio como base del Modelo de Requisitos», Grupo de Investigación de Ingeniería del Software, Departamento de Informática y Sistemas, Facultad de Informática. Universidad de Murcia, C.P. 30071 Campus de Espinardo, Murcia, España, 2002.
- Mehdi Achour, Friedhelm Betz, «PHP Manual», http://www.php.net/docs.php to get the actual version. Copyright ©, the PHP Documentation Group, 15 de Octubre del 2005.
- Ministerio de Administraciones Públicas, «Gestión De Proyectos» (Informe formato pdf.), España, 2007.
- Shea Frederick, Colin Ramsay, Steve 'Cutter' Blades, «Learning Ext JS», From Technologies to Solutions, First published: Copyright © 2008 Packt Publishing, Noviembre del 2008. (idioma Ingles)
- Yoenis Pantoja Zaldívar, «Plataforma Soberana para el Desarrollo de Sistemas de Información Geográfica» (Informe), Universidad de las Ciencias Informáticas. 2008.
- Jorge Ramon, «Ext JS 3.0 Cookbook», Copyright © 2009 Packt Publishing First published: October 2009 (idioma Ingles)
- Sara Álvarez, «FPDF es una clase para la generación dinámica de documentos PDF en PHP», (Informe), Publicado: 19 de Enero del 2010.
- «Doctrine ORM for PHP», Guide to Doctrine for PHP, Doctrine 1.2, *and License: Creative Commons Attribution-Share Alike 3.0 Unported License, Version: manual-1.2* published 11 de March 2010.

GLOSARIO

Ajax: Es una técnica de desarrollo Web para crear aplicaciones interactivas

Artefacto: Puede ser un documento, un modelo, o un elemento de modelo. Metodologías de desarrollo de software en: http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

Atributos multievaluados: Se dice del atributo tal que para una misma entidad puede tomar varios valores diferentes, es decir, varios valores del mismo dominio, son atributos que tienen límites (inferior y superior) en el número de valores para una entidad.

Base de datos relacional: Es una base de datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la base de datos operan sobre estas tablas.

CASE: Ingeniería de software asistida por computadora por sus siglas en inglés. Software que se utiliza para ayudar a las actividades del proceso de software o software que es utilizado para diseñar y para implementar otro software.

CSS: Lenguaje de hojas de estilo usado en páginas Web.

Doctrine: Librería para el mapeo objeto relacional. Permite realizar las consultas mediante un lenguaje propio llamado DQL. (Doctrine Query Language).

DOM: Es una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

Excel: Aplicación para manejar hojas de cálculo.

Framework: Es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Gestión: Acción y efecto de gestionar o de administrar.

Gestión de proyectos: disciplina de organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en el proyecto dentro del alcance, el tiempo, y coste definidos

HTML: Siglas de HyperTextMarkup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTTP: Protocolo de transferencia de hipertexto cuyo objetivo es permitir la transferencia de archivos (principalmente, en formato HTML).entre un navegador (el cliente) y un servidor Web.

IDEs: Un entorno de desarrollo integrado es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

JAVA: Lenguaje de programación orientado a objeto. Es un lenguaje multiplataforma que permite realizar aplicaciones para todo tipo de entornos, para Web, dispositivos móviles, aplicaciones de escritorio, servidor. Manual de java en: http://www.manual-java.com/

JavaScript: Es un lenguaje de scripting orientado a objetos, utilizado para acceder a objetos en aplicaciones.

Join: Sentencia que permite combinar registros de dos o más tablas en una base de datos relacional.

Librería YUI: El framework o librería YUI (Yahoo! User Interface) es un conjunto de utilidades y controles escritos en JavaScript que se utilizan para crear aplicaciones Web dinámicas complejas. El framework YUI en:
http://librosWeb.es/css_avanzado/capitulo5/el_framework_yui.html

Multiplataforma: Capacidad de soportar múltiples plataformas.

ORM ("object-relational-mapping"): Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional.

PDF: Es un formato de almacenamiento de documentos.

PHP: Es un lenguaje de scripting que permite la generación dinámica de contenidos en un servidor Web. Manual de php en: http://www.manualdephp.com/

Plataforma: El principio, en el cual se constituye un hardware, sobre el cual un software puede ejecutarse/desarrollarse. Define un estándar alrededor del cual un sistema puede ser desarrollado.

Portabilidad: Característica que posee un software de poder ser ejecutado en diferentes plataformas.

Prototype: Es un framework escrito en JavaScript que se orienta al desarrollo sencillo y dinámico de aplicaciones Web. Es una herramienta que implementa las técnicas AJAX.

RAM: Son las siglas de Random Access Memory, un tipo de memoria de ordenador a la que se puede acceder aleatoriamente.

Recursos Materiales: son los bienes tangibles con los que una organización cuenta para utilizar en el logro de sus objetivos.

Recursos Humanos: Trabajo que aporta el conjunto de los empleados o colaboradores de una organización determinada.

Scriptaculous: Es una biblioteca JavaScript que permite el uso de controles AJAX, drag 'n drop, y otros efectos visuales en una página Web.

Sistema gestor de base de datos (SGBD): Es un conjunto de programas que permiten crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad, su objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones.

SQL: Es un lenguaje de base de datos normalizado, utilizado por el motor de base de datos que permite trabajar con cualquier tipo de lenguaje (ASP o PHP) en combinación con cualquier tipo de base de datos (MS Access, SQL Server, MySQL...).Tutorial de SQL en: http://www.desarrolloWeb.com/manuales/9/

UML: Acrónimo de Unified Modeling Languaje, no es un lenguaje de programación, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

Visual Basic Script: Es un lenguaje de programación de scripts del lado del cliente, pero sólo compatible con Internet Explorer. Tutorial de Visual Basic Script en: http://www.desarrolloWeb.com/manuales/tutorial-visual-basic-script-manual.html

World Wide Web Consortium (W3C): Es una comunidad internacional que desarrolla estándares que aseguran el crecimiento de la Web a largo plazo.