

Universidad de las Ciencias Informáticas

FACULTAD 6



Título:

“Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnología de Gestión de Datos v2.0 (DRI Management System v2.0)”

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

Autor:

Darlon Antonio Santana Carvajal

Tutores:

Ing. Niurka Martínez Durán

Ing. Alexander Delgado Gutiérrez

Ciudad de la Habana, Cuba

17 de junio del 2013

“Año 55 de la Revolución”



“Debemos usar el tiempo sabiamente y darnos cuenta de que siempre es el momento para hacer las cosas bien...” Nelson Mandela

Declaración de Autoría

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Darlon Antonio Santana Carvajal

Firma del Autor

Ing. Niurka Martínez Durán

Firma del Tutor

Ing. Alexander Delgado Gutiérrez

Firma del Tutor

DATOS DE CONTACTO

Tutores:

- Ing. Niurka Martínez Durán

Especialidad de graduación: Ingeniero en Ciencias Informáticas

Años de graduado: 5

Correo Electrónico: nduran@uci.cu

- Ing. Alexander Delgado Gutiérrez

Especialidad de graduación: Ingeniero en Ciencias Informáticas

Años de graduado: 1

Correo Electrónico: adgutierrez@uci.cu

Agradecimientos

AGRADECIMIENTOS

Como bien diría un sabio "...toda la gloria del mundo cabe en un grano de maíz...", y nunca será grande el esfuerzo para estimular a esas personas que han hecho posible que hoy me encuentre aquí. Hoy solo encuentro palabras para ofrecerle mi agradecimiento a cada uno de ellos:

Primeramente quiero agradecer a mi mamá María del Carmen Carvajal Hernández y a mi papá Ernesto Alexis Santana Lorez, por su amor y cariño, por su apoyo en todas mis decisiones, por el esfuerzo y trabajo con el cual me han criado haciendo de mí lo que ahora soy.

A mis hermanos por su infinito amor y por estar siempre presente en cada una de mis acciones para apoyarme.

A mis amigos de batalla que siempre han estado a mi lado a pesar de las dificultades y los malos momentos.

A mi compañera de sentimientos comunes que hasta ahora me ha soportado y me brinda un cariño desinteresado.

A la UCI por las oportunidades ofrecidas.

A mis compañeros del impresionante grupo 6101, que bastante que hemos desandado en esta búsqueda de un camino profesional.

A esa gran familia FEU que me convirtió en una persona mejor.

A esas personas que de una forma u otra siempre estuvieron presente en este trabajo de diploma: Yuned, René, Roilan, Reynaldo y Marcos Michel.

A mis tutores y muy especialmente a la profe Niurka que me enseñó a trabajar siempre en busca de una perfección mayor, pues como bien diría ella "quien bien te quiere te hará sufrir".

A mi nueva familia adquirida en el transcurso de estos 5 años, que me enseñaron que no hace falta una vinculación de sangre para querer a una persona y entregarse por completo.

Por último agradecer a todos los que de una forma u otra han contribuido a lo largo de estos cinco años a mi preparación profesional y como una mejor persona, a mis compañeros de aula, profesores y a todos con los que he compartido de una forma u otra, todos han puesto su granito de arena. Hoy los nombres nos son importantes pues sus recuerdos están perpetuados ya en mi existencia. De corazón muchísimas gracias. A todos y cada uno de ustedes muchísimas gracias...

DEDICATORIA

A mis padres María del Carmen y Ernesto Alexis, por ser siempre esa inspiración que nunca desaparece.

A mis hermanos Ernesto y María Leonor que nunca me dejarán flaquear ante los obstáculos de la vida.

A esta Revolución Cubana, que me ha dado la oportunidad de llevar a cabo este acontecimiento y me ha cubierto de una inmensa gama de valores por la historia que la misma representa.

A toda mi familia por esperar siempre lo mejor de mí.

A mis compañeros de estudio y a toda esa otra familia que a pesar de no poseer lazos comunes sanguíneos nunca me han abandonado.

RESUMEN

Los sistemas para la gestión de información son conjuntos de elementos interrelacionados con el propósito de prestar atención al control de la información en las organizaciones. El presente Trabajo de Diploma tiene como objetivo desarrollar la segunda versión del Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnología de Gestión de Datos. Para ello se realizó un análisis teórico del tema en cuestión, teniendo en cuenta las tecnologías, herramientas adecuadas para su desarrollo y la arquitectura de implementación seguida en la primera versión. Se identificaron las funcionalidades que debe cumplir el software, se realizó el diseño a partir de los patrones seleccionados y se implementó la solución. Se obtuvo como resultado un sistema desarrollado sobre las nuevas tendencias tecnológicas, evidenciado en la utilización del framework Symfony2, renovando el empleado en la primera aplicación. Además mantiene los logros alcanzados en la primera versión e incorpora nuevas operaciones como la gestión de puestos de trabajo, permitiendo realizar una gestión más amplia sobre los recursos ubicados en los locales. En la seguridad se empleó la política de control de acceso basado en roles empleando una función resumen superior a la del primer sistema. La solución propuesta es de gran significado para el centro ya que confiere a sus directivos de una útil herramienta para la gestión y el control del personal.

PALABRAS CLAVE

Gestión, información, puesto de trabajo, recursos, sistema.

Tabla de Contenidos

TABLA DE CONTENIDOS

RESUMEN.....	I
INTRODUCCIÓN.....	1
CAPÍTULO I: FUNDAMENTOS TEÓRICOS.....	5
Introducción.....	5
1.1 Sistemas para la gestión de la información (SGI) existentes en el mundo.....	5
1.2 Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos v1.0.....	8
1.3 Herramientas, tecnologías y metodologías.....	11
1.3.1 Metodología de desarrollo de software.....	12
1.3.2 Lenguaje de modelado de sistemas.....	14
1.3.3 Lenguajes de programación.....	15
1.3.4 Framework y librería de desarrollo.....	16
1.3.5 Herramienta ORM (Object Relational Mapper - Mapeo Relacional de Objeto).....	18
1.3.6 Servidor web.....	19
1.3.7 Servidor de base de datos.....	20
1.3.8 Entorno integrado de desarrollo.....	21
Conclusiones del capítulo.....	21
CAPÍTULO II: CARACTERÍSTICAS Y DISEÑO DEL SISTEMA.....	23
Introducción.....	23
2.1 Modelo de Dominio.....	23
2.1.1 Descripción de las Clases del Dominio.....	24
2.2 Requisitos del sistema.....	25
2.2.1 Requisitos funcionales.....	25
2.2.2 Requisitos no funcionales.....	33

Tabla de Contenidos

2.3 Modelo de casos de uso	34
2.3.1 Diagrama de casos de uso del sistema	36
2.3.2 Descripción de los casos de uso del sistema	38
2.4 Modelo de diseño.....	42
2.4.1 Diagrama de Clases del Diseño	42
2.4.2 Patrón de arquitectura	43
2.4.3 Patrones de diseño utilizados.....	44
2.4.4 Diagrama de Interacción	47
2.5 Modelo de Datos	48
Conclusiones del capítulo	50
CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA.....	51
Introducción	51
3.1 Diagrama de componentes	51
3.2 Mapa de navegación del sistema	53
3.2.1 Imágenes del sistema.....	54
3.3 Validación del sistema.....	57
3.3.1 Pruebas de caja negra	57
3.3.2 Pruebas de carga	59
3.3.3 Pruebas de stress	59
3.4 Diagrama de Despliegue.....	60
Conclusiones del capítulo	61
CONCLUSIONES	62
RECOMENDACIONES	63
REFERENCIAS	64
BIBLIOGRAFÍA.....	67

Tabla de Contenidos

GLOSARIO DE TÉRMINOS	70
----------------------------	----

INTRODUCCIÓN

La información ha devenido como recurso estratégico determinante de la competitividad y sustento a los procesos de decisión, siendo el único elemento capaz de crear conocimiento y satisfacer las necesidades de las personas y de las entidades. Se ha convertido en la materia prima indispensable para el desarrollo, equilibrio y adaptabilidad de las organizaciones, sectores y países. Es vista objetivamente como la expresión material del conocimiento con fines de uso, enmarcándose en el progreso de las Tecnologías de la Información y las Comunicaciones (TIC).

Las TIC a su vez, forman un eslabón notable en el desarrollo de la sociedad, manteniendo la presencia de servicios y productos informáticos con un alto grado de calidad, que se hacen cada vez más necesarios. El uso correcto de las TIC brinda un entorno de soluciones amplias que incluyen la recuperación, el almacenamiento, el envío de datos de un sitio a otro y el procesamiento de la información para poder definir el alcance de los resultados y elaborar informes.

En Cuba el desarrollo de la informática y las comunicaciones es un proceso priorizado, donde se busca lograr un continuo avance, que posibilite una mejora constante en los sectores de la economía y la sociedad. La Universidad de las Ciencias Informáticas (UCI), no está exenta de este proceso, siendo la misma, un eje conductor para llevar a cabo procesos de cambios y transformaciones en todos los sectores del ámbito social.

La UCI, universidad de referencia nacional por sus métodos de formación docente, vincula el estudio con la producción de software. En ella existe un orden organizacional basado en el control específico de la información que se manipula. La misma está conformada por una estructura organizativa fragmentada en áreas por objetivos, una de ellas es la encargada de la producción de software, que agrupa a los centros productivos entre los que se encuentra el Centro de Tecnologías de Gestión de Datos (DATEC).

DATEC posee la misión de crear bienes y servicios informáticos relacionados con la gestión de datos, área del conocimiento que agrupa tanto a los sistemas de información, como a los denominados sistemas de inteligencia empresarial o de negocios, cuyo propósito fundamental es apoyar el proceso de toma de decisiones (1). Este centro está constituido por cuatro departamentos Bioinformática, Almacenes, PostgreSQL e Integración de Soluciones. Además posee tres grupos de trabajo Calidad, Mercadotecnia y Soporte.

El grupo de Soporte realiza una serie de procesos, encaminados a la orientación del control de la disposición de los trabajadores y estudiantes por local, y al seguimiento del estado técnico y prestaciones

de los ordenadores. En estos procesos de control de la información se materializan un número de actividades, destacándose: la gestión de información relacionada con las incidencias, las descargas y los recursos ubicados en cada local.

Partiendo del procesamiento de información de cada una de estas tareas, se desarrolló el Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos v1.0. Esta aplicación permite conocer la distribución de los estudiantes y profesores auxiliándose de la generación de reportes sobre los datos obtenidos. Hace posible también controlar el acceso, los permisos y los roles de los usuarios dinámicamente; lo que proporciona perdurabilidad ante posibles cambios en roles definidos inicialmente. A través de las trazas se obtienen los registros de la navegación de todo el personal autenticado. Este sistema confiere a los directivos de la entidad una útil herramienta para la gestión de los recursos disponibles y control del personal.

Independientemente de las prestaciones que hoy brinda se puede afirmar, que la misma presenta algunas limitantes. Entre ellas se puede encontrar, la implantación de brechas de seguridad en la creación del sistema. Así mismo se debe aludir que la versión 1.0 del Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos se limita solo al ciclo básico de la gestión de procesos. La opción de visualizar reportes, no contempla la cantidad necesaria de los mismos, teniendo en cuenta el gran volumen de información que se almacena. Además al centro se le entregó el control total de los laboratorios siendo esto una nueva característica del ambiente de desarrollo que no se encuentra contemplado en la aplicación desplegada.

Por todo lo mencionado se identifica como **problema de la investigación**: ¿Cómo satisfacer las necesidades actuales del grupo de Soporte perteneciente al centro DATEC?; teniendo como **objeto de estudio**: Sistemas de gestión de información; y enmarcando la investigación en el **campo de acción**: Sistemas para gestión de información de incidencias, descargas y recursos.

Dada la problemática planteada se propone como **objetivo general**: Desarrollar el Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos v2.0.

Para darle cumplimiento a este objetivo de desglosan los **objetivos específicos** siguientes:

- ✓ Fundamentar la selección de la metodología, herramientas y tecnologías a utilizar en el desarrollo del sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos v2.0.
- ✓ Diseñar la propuesta del sistema v2.0.

- ✓ Implementar la propuesta del sistema v2.0.
- ✓ Validar el sistema propuesto.

Para cumplir con estos objetivos se desprenden las siguientes **tareas de la investigación**:

- ✓ Análisis de las tecnologías y tendencias actuales de sistemas vinculados a servicios de soporte.
- ✓ Análisis de los procesos vinculados a la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos.
- ✓ Definición de las características del sistema a implementar.
- ✓ Rediseño del sistema a implementar.
- ✓ Rediseño de la interfaz gráfica del sistema a implementar.
- ✓ Implementación del sistema rediseñado en la capa de presentación de datos.
- ✓ Implementación del sistema rediseñado para satisfacer los requisitos funcionales.
- ✓ Realización de pruebas de caja negra al sistema implementado.
- ✓ Realización de pruebas de carga al sistema implementado.
- ✓ Realización de pruebas de stress al sistema implementado.

El presente trabajo de diploma se organizó en 3 capítulos:

Capítulo 1: Fundamentos teóricos.

Este capítulo describe todo el contenido relacionado con los conceptos y definiciones fundamentales respecto a los sistemas de gestión de información. Se realiza un análisis de sistemas existentes en el mundo y en Cuba orientados a la gestión de la información, caracterizando fundamentalmente los componentes de la versión 1.0 del Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos. Se fundamenta el uso de las distintas herramientas, tecnologías y metodología utilizadas para el desarrollo del sistema propuesto.

Capítulo 2: Características y diseño del sistema

En el capítulo se definen las características y se realiza el rediseño del sistema a desarrollar. Incluye el modelo de dominio, la definición de los requisitos funcionales y no funcionales. Contiene además los actores y casos de uso representados en el diagrama de casos de uso del sistema y la descripción textual de uno de sus casos de uso arquitectónicamente significativos. En el proceso del diseño se brinda un acercamiento a la implementación del sistema, construyendo para ello los diagramas de interacción del mismo y el modelo de clases del diseño siguiendo el estilo arquitectónico y los patrones de diseño

seleccionados, proporcionando la base para proceder a la etapa de implementación. Además se muestra el modelo de datos con la descripción de las tablas más significativas.

Capítulo 3: Implementación y pruebas del sistema

En el capítulo se describe cómo los elementos del modelo del diseño se implementan en términos de componentes y se muestra una descripción detallada de los paquetes de implementación. Se presenta el mapa de navegación del sistema desarrollado y algunas imágenes del sistema para una mejor comprensión del mismo. Se realiza la validación del sistema mediante las pruebas de software de caja negra, carga y stress, para comprobar la operatividad de las principales funcionalidades. Y finalmente se muestra el diagrama de despliegue, con una breve descripción de los nodos que lo conforman.

CAPÍTULO I: FUNDAMENTOS TEÓRICOS

Introducción

Durante las últimas décadas el mundo se ha enfrentado a continuos cambios en la esfera científico-técnica, la era de la información y el conocimiento está transformando los sistemas económicos, políticos y sociales. Los sistemas de gestión de información juegan en este proceso un papel determinante en la actualidad. Este capítulo engloba los conceptos y definiciones fundamentales respecto a los sistemas de gestión de información. Se estudian los sistemas existentes en el mundo y en Cuba orientados a la gestión de la información, caracterizando fundamentalmente los componentes de la versión 1.0 del Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos. Además se realiza un análisis que fundamenta el uso de las distintas herramientas, tecnologías y metodología utilizadas para el desarrollo del sistema propuesto.

1.1 Sistemas para la gestión de la información (SGI) existentes en el mundo

La gestión de información es el proceso que se encarga de suministrar los recursos necesarios para la toma de decisiones, así como para mejorar los procesos, productos y servicios de la organización. Josep Curto destacado investigador de la ciencia empresarial plantea que es el “... proceso que incluye operaciones como extracción, manipulación, tratamiento, depuración, conservación, acceso y/o colaboración de la información adquirida por una organización a través de diferentes fuentes y que gestiona el acceso y los derechos de los usuarios sobre la misma.... Se establece, por lo tanto, como una disciplina transversal que aparece entrelazada en todas las diferentes capas o tejidos de una organización, en todos los conceptos de management (recursos humanos, marketing, finanzas, estrategias, operaciones,...) y les proporciona soporte ” (1).

La investigación sobre los SGI mostró que existe una tendencia creciente de su utilización en la actualidad, los cuales se encuentran orientados a diversas esferas de la actividad humana entre ellas el soporte técnico. Al realizar un análisis sobre estos tipos de sistemas a nivel internacional y nacional se encontraron algunos como el ISOCLOUD, GIPFARMA y CEDRUX los cuales se describen a continuación.

ISOCLOUD

ISOCLOUD es una aplicación que permite adaptarse a las empresas de una forma rápida y a través de la web. Contiene tres servicios fundamentales: Sistema de Gestión Integrado (14 módulos), Sistema de

Capítulo I: Fundamentos Teóricos

Gestión de Calidad (10 módulos) y Sistema de Gestión Medio Ambiental (13 módulos). Algunas de sus características fundamentales son:

- ✓ Generación de informes de incidencias, objetivos y metas, y resultados de encuestas de satisfacción.
- ✓ Gestión documental y proceso de control de firmas.
- ✓ Permite el registro de consumos, residuos peligrosos y no peligrosos con el fin de llevar a cabo el control operacional en las organizaciones.
- ✓ Gestión de diferentes tipos de incidencias (no conformidades internas, no conformidades de proveedor, acciones correctivas, entre otras.), relacionadas con distintas procedencias (encuestas de satisfacción, planes de formación y auditorías).
- ✓ Definición y evaluación de aspectos ambientales, a través de una metodología que permite establecer distintas variables y valoraciones, tanto para los aspectos ambientales reales como potenciales (3).

Esta herramienta resulta una novedosa aplicación para la gestión de información, pero la misma, no sostiene la gestión de descargas y recursos de una entidad. Además de ser un proyecto que solicita de un capital financiero estable para las actualizaciones que él demanda.

GIPFARMA

GIPFARMA es una herramienta informática diseñada para conseguir una eficiente organización de todos los recursos de una oficina (particularmente en una farmacia) o alguna empresa. GIPFARMA no sólo planifica tareas, gestiona horarios, guarda y ordena documentos sino además lo más importante es que coordina los recursos disponibles para que funcionen como una unidad. Este sistema agrupa sus funcionalidades en 6 gestores totalmente comunicados entre sí:

- ✓ Gestor de Agenda

Es el visor principal de los diferentes eventos. Aporta una comunicación total entre los usuarios. Según los permisos asignados, podrá ver tareas asignadas, circulares, noticias, notas, reuniones, biblioteca, contactos, alarmas, horarios y equipamientos reservados.

- ✓ Gestor de Personas

Incluye: gestión total de horarios, plan de formación continuada y profesiograma.

- ✓ Gestor de Tareas

Capítulo I: Fundamentos Teóricos

Posee una base de datos propia. La entidad irá activando tareas al ritmo que crea conveniente. La arquitectura de las tareas y su ejecución son exclusivas de GIPFARMA y están desarrolladas para una realización eficaz e intuitiva.

- ✓ Gestor de Incidencias

Se contabilizan todas las incidencias generadas. Cada incidencia tiene un responsable y un índice de calidad crítico que sirve de ruptura para que el administrador reciba una alarma de verificación del proceso.

- ✓ Gestor de Equipamientos

Se dará de alta los diferentes equipamientos de la entidad: aire acondicionado, un ordenador, una estantería, un bolígrafo, entre otros. Puede generar mantenimientos, controles de entrada y salida.

- ✓ Gestor de Información

Genera su propia biblioteca de documentos, ya sean físicos o virtuales. Genera canales de noticias y proporciona permisos para abrir canales de noticias según nuestras preferencias (4).

Este sistema a pesar de todas las facilidades que puede brindar no se acopla a las necesidades que se solicitan en DATEC, debido a que no realiza la gestión de descargas que es una de las actividades que el grupo de Soporte exige.

CEDRUX

Es un sistema de gestión empresarial que se desarrolló en la UCI. Cuenta con un subsistema para el control de los recursos de cómputo que pretende servir de apoyo a la toma de decisiones en las entidades. El subsistema es capaz de gestionar recursos de forma dinámica, tributa mediante operaciones al sistema contable de la empresa. Incluye además temas como la agrupación de recursos y la generación de documentos (5).

Este sistema permite la gestión de los recursos, pero los demás procesos que realiza no se ajustan a las necesidades de la entidad. El sistema CEDRUX tampoco posee ninguna funcionalidad para la gestión de incidencias y descargas.

Estos SGI cumplen con las características necesarias para los cuales fueron creados pero ninguno se ajusta a las especialidades que se requieren para darle solución a la problemática de este trabajo de diploma. Además algunas de estas soluciones han sido implementadas sobre herramientas propietarias, deviniendo en gastos elevados al país por el pago de sus licencias.

1.2 Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos v1.0

Es un sistema implementado para el grupo de Soporte del centro DATEC, que busca automatizar el control de la gestión de los recursos y locales de esta organización, y permite elaborar diferentes tipos de reportes que ayudan a la toma de decisiones de los directivos de la entidad. Esta aplicación posibilita la gestión de la información de los departamentos, proyectos, locales y recursos, así como las solicitudes de descargas e incidencias (6). Además brinda la facilidad que cada usuario reporte averías directamente sobre el sistema y pueda de este modo darle un seguimiento en cuanto a su restablecimiento. Ofrece un seguimiento de trazas, permitiendo conocer las actividades realizadas sobre el sistema por los usuarios que hayan interactuado con él. El sistema está estructurado por diferentes áreas de trabajo las cuales se describen a continuación:

- ✓ Gestión de departamento. En esta capa de presentación se registran todos los departamentos que existen en el centro. De cada uno de ellos se tiene el nombre y una pequeña descripción. En esta sección se realizan las operaciones de añadir, modificar y eliminar según los permisos que tenga el usuario que se haya autenticado.

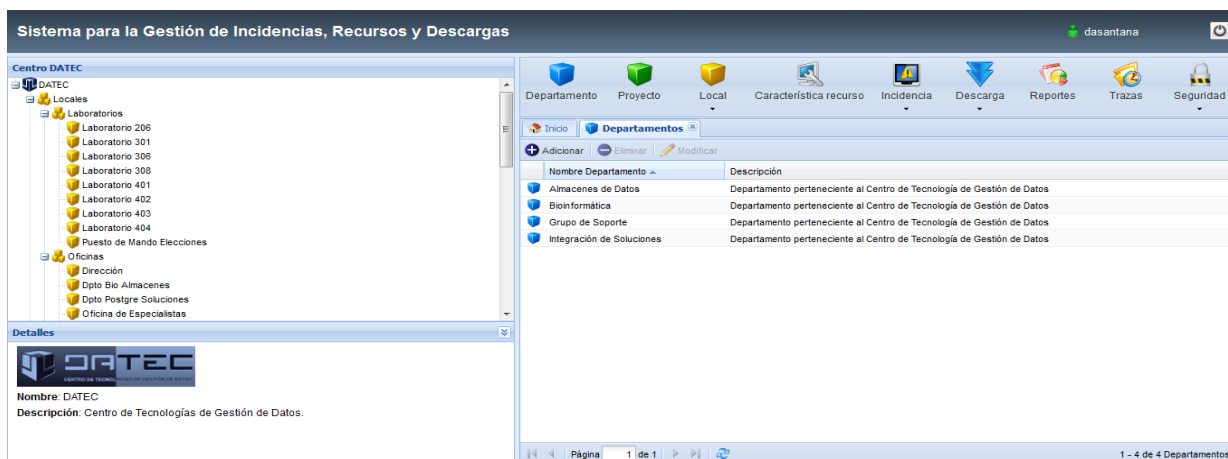


Figura 1: Gestionar departamento.

- ✓ Gestión de proyecto. Es donde se lleva el control de los proyectos registrados en el sistema como existentes. De ellos se recogen el nombre, el departamento al cual pertenecen y una breve descripción de cada uno. Brinda las opciones de añadir, modificar y eliminar según los permisos que tenga el usuario que se haya autenticado.

Capítulo I: Fundamentos Teóricos

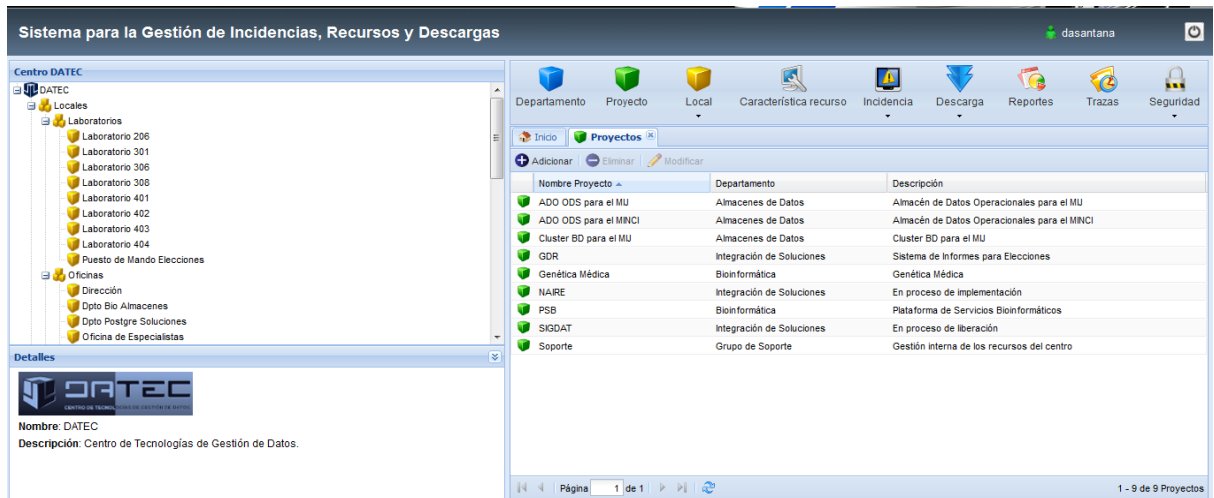


Figura 2: Gestionar proyecto.

- ✓ Gestión de local. Este campo se encuentra dividido en dos secciones:
 - ❖ Gestión de categoría de local. Encargada de administrar cada una de las categorías que se tienen en el centro enmarcadas en este producto informático.
 - ❖ Gestión de local. Encargada de administrar los locales que se encuentran concebidos en la aplicación. De cada uno de ellos se recoge el nombre, su categoría y una breve descripción.

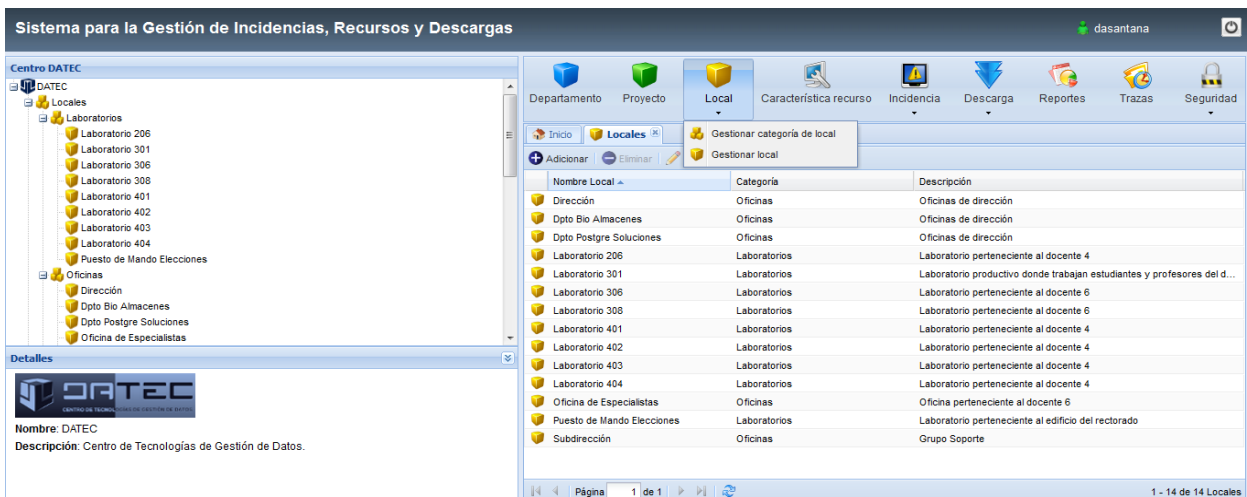


Figura 3: Gestionar local.

Capítulo I: Fundamentos Teóricos

- ✓ Gestión de característica de recurso. Es la encargada de almacenar y administrar toda la información relacionada con los recursos que existen.
- ✓ Gestión de Incidencia. Se encuentra desglosado por dos sectores:
 - ❖ Reportar Incidencia. Es donde se notifica la incidencia detectada.
 - ❖ Administrar incidencia. Es donde se manejan todas las incidencias reportadas, posibilitando su control.
- ✓ Gestión de descargas. Se encuentra fragmentada en dos campos:
 - ❖ Solicitar descarga. Lugar en el cual se manifiesta la petición de una o más descargas.
 - ❖ Administrar descarga. Sector en el cual se lleva toda la información relacionada con cada una de las descargas solicitadas.
- ✓ Generación de reportes. Es donde se visualizan los reportes que están contemplados en el sistema. Ellos pueden ser adquiridos por el usuario en varios formatos (Adobe Acrobat Documento, Documento de Microsoft Word, Texto OpenDocument, Hoja de cálculo de Microsoft Excel 97-2003, Hoja de cálculo OpenDocument, Presentación de Microsoft PowerPoint).
- ✓ Gestión de seguridad. Este módulo es el encargado de gestionar todos los procesos encaminados a la protección del sistema. Se encuentra dividido en cuatro campos:
 - ❖ Cambiar contraseña. Permite que los usuarios locales puedan cambiar su contraseña
 - ❖ Gestionar categoría de usuario. Tiene la finalidad de administrar las categorías de los usuarios.
 - ❖ Gestionar rol. Es el sector encargado de adicionar, modificar y eliminar los roles definidos en el sistema, delimitando en cada uno de ellos los permisos que pueden poseer sobre la aplicación.
 - ❖ Gestionar usuario. Es donde se adicionan, eliminan, modifican y hasta se cambian las contraseñas de los usuarios registrado en el software. De cada uno de ellos se registra su nombre, apellidos, usuario, rol y categoría.
- ✓ Gestión de recurso. Es la sección encargada de manejar el control de los recursos que se encuentran ubicados en los locales. De cada recurso se almacena su posición, una breve descripción y las características específicas que poseen. Se brindan las opciones de añadir, eliminar y modificar según los permisos que tenga el usuario que se haya autenticado.

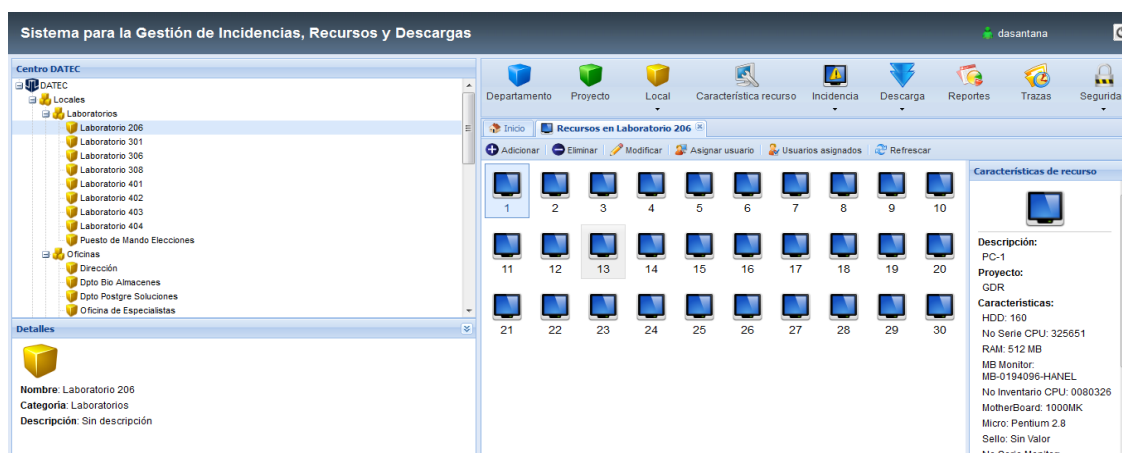


Figura 4: Gestionar recurso.

Esta versión cumple con las normas del ciclo básico para la cual fue desarrollada, pero la misma presenta ciertas limitaciones que hoy la imposibilitan de brindar un funcionamiento adecuado que se ajuste a las necesidades que sostiene DATEC. Aunque resulta una solución útil para los directivos del centro, este producto no maneja en su política de seguridad el concepto de carga bajo demanda. De conjunto con esto, se debe indicar que resulta insuficiente la cantidad de reportes que hoy genera, requiriendo que se extienda esta proporción para lograr una gestión de la información más amplia. A su vez necesita que se le incorporen nuevas características estructurales que no posee; teniendo en cuenta, que la gestión de recursos, aspecto fundamental en el ambiente de desarrollo, no posee las especificidades que se solicitan en la institución. Teniendo en cuenta que el recurso es visto como una computadora en su composición, y no es capaz de ver los accesorios que ella contiene; imposibilitando la gestión de los mismos. Y es específicamente en esta área donde se requiere que se amplíe el margen de información y además se incorpore una gestión encaminada a recursos adicionales, que son componentes no computacionales que forman parte de la ambientación de los locales.

1.3 Herramientas, tecnologías y metodologías

A continuación se presentan las descripciones de las herramientas, metodologías y tecnologías utilizadas para dar solución al problema. Para definir la selección se tomaron en cuenta las necesidades existentes, tendencias actuales, el entorno donde se desplegará el sistema y la arquitectura implementada en la

Capítulo I: Fundamentos Teóricos

versión 1.0 del Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos.

1.3.1 Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas, herramientas y soporte documental para el desarrollo de un producto (software). Se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo e incremental.). Son para estructurar, planear y controlar el proceso de desarrollo del software. Constituyen una guía que define las tareas y actividades que se deben realizar para obtener un software de buena calidad (7).

Metodologías ágiles

Son orientadas a la interacción con el cliente y el desarrollo incremental del software. Mostrando versiones parcialmente funcionales de la aplicación al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Se enmarca más en la capacidad de respuesta ante un cambio realizado que en el seguimiento estricto de un plan. Consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. Entre ellas se destacan: XP (eXtreme Programming), Scrum, ASD (Adaptive Software Development) y OpenUP.

Independientemente de estas también existen las que tienen un enfoque tradicional, que son conocidas como metodologías robustas.

OpenUP

OpenUp es una variante ágil del Proceso Unificado que aplica métodos iterativo e incremental dentro de un ciclo de vida estructurado. Abraza una filosofía pragmática y ágil que se centra en la naturaleza de colaboración de desarrollo de software.

Es un proceso de desarrollo iterativo del software que es mínimo, completo, y extensible. El proceso es mínimo en que solamente el contenido fundamental es incluido; es completo en que puede ser manifestado como todo el proceso para construir un sistema; extensible en que puede ser utilizado como fundamento sobre el cual el contenido de proceso se pueda agregar o adaptar según lo necesitado. OpenUP como metodología de desarrollo es conducida por el principio de colaboración para alinear intereses y para compartir su comprensión. Se presenta la metodología OpenUp por las siguientes características:

Capítulo I: Fundamentos Teóricos

- ✓ Es un proceso de desarrollo del software. Es completo en el sentido que puede ser manifestado como todo el proceso para construir un sistema.
- ✓ Es extensible ya que en el proceso se pueda agregar o adaptar según lo vayan requiriendo los sistemas. OpenUP es un proceso ágil.
- ✓ Es ligero y proporciona una comprensión detallada del proyecto, beneficiando a clientes y desarrolladores sobre productos a entregar y su formalidad.
- ✓ Se centra en una arquitectura temprana para reducir al mínimo los riesgos y organizar el desarrollo.
- ✓ OpenUp es la metodología utilizada por desarrolladores de alto nivel en casi todo el mundo por sus altas cualidades administrativas (8).

Fases del ciclo de vida de OpenUp:

- 1. Concepción:** Primera de las 4 fases en el proyecto del ciclo de vida, acerca del entendimiento del propósito y objetivos y obteniendo suficiente información para confirmar que el proyecto debe hacer. El objetivo de esta fase es capturar las necesidades de los stakeholder (individuo, grupo u organización) en los objetivos del ciclo de vida para el proyecto.
- 2. Elaboración:** Es la segunda de las 4 fases del ciclo de vida del OpenUP donde se trata los riesgos significativos para la arquitectura. El propósito de esta fase es establecer la base la elaboración de la arquitectura del sistema.
- 3. Construcción:** Esta fase está enfocada al diseño, implementación y prueba de las funcionalidades para desarrollar un sistema completo. El propósito de esta fase es completar el desarrollo del sistema basado en la Arquitectura definida.
- 4. Transición:** Es la última fase, cuyo propósito es asegurar que el sistema es entregado a los usuarios, y evalúa la funcionalidad y performance del último entregable de la fase de construcción.

Haciendo un análisis de lo anteriormente expuesto, se decide seleccionar a OpenUP para enfrentar el desarrollo del software propuesto. Teniendo en cuenta además, que es una metodología ágil, orientadas al código y que se ajusta a los continuos cambios del proceso de desarrollo.

1.3.2 Lenguaje de modelado de sistemas

Es un lenguaje artificial diseñado para expresar modelos, los que habitualmente se muestran en forma de diagramas por comodidad, que suelen incorporar notaciones gráficas. Igual que los naturales, los de modelado poseen un léxico y una sintaxis (9).

UML

El Lenguaje de Modelado Unificado (UML), es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (10).

UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten.

Otro objetivo de este modelado visual es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos).

UML es además un método formal de modelado. Esto aporta las siguientes ventajas:

- ✓ Mayor rigor en la especificación.
- ✓ Permite realizar una verificación y validación del modelo realizado.
- ✓ Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel, de la estructura de un proyecto.

Analizando estas características para el modelado del sistema se concluyó utilizar UML, uno de los lenguajes de modelado de sistemas más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group), en español Grupo de Gestión de Objetos.

Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas

pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores (11).

Visual Paradigm 6.4

Visual Paradigm es una herramienta CASE que permite construir el diagramado UML, como son los flujos de eventos del sistema, las clases, todo lo que es documentación tanto de desarrollo como procesos de negocio. La versión Visual Paradigm UML 6.4 Community Edition, alternativa libre y gratuita de este software, permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos (12).

Proporciona características tales como la ingeniería inversa y la generación de código y de informes. Permite invertir el código fuente de los programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Otra de las ventajas que ofrece es la navegación intuitiva entre el modelo visual y el código, además de permitir la sincronización entre el código fuente y el modelo en tiempo real o bajo demanda.

Teniendo en cuenta estas características se decide utilizar como herramienta CASE el Visual Paradigm, debido a que también posee disponibilidad en múltiples plataformas (Windows, Linux) y posibilita el uso de un lenguaje estándar común a todo el equipo de desarrollo facilitando la comunicación.

1.3.3 Lenguajes de programación

Un lenguaje de programación consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente. Es diseñado para describir el grupo de acciones consecutivas que un equipo debe ejecutar, lo que permite crear programas informáticos posibilitando al desarrollador comunicarse con los dispositivos hardware y software existentes (13).

PHP

PHP, acrónimo de (Hypertext Pre Processor) constituye un lenguaje de programación interpretado, de propósito general y ampliamente difundido en el mundo. Diseñado originalmente para la creación de páginas web dinámicas, principalmente en interpretación del lado del servidor aunque puede ser incrustado dentro de código HTML.

Capítulo I: Fundamentos Teóricos

Está especialmente diseñado para incrementar el dinamismo de las páginas web, aprovechando los recursos de las redes informáticas y los escasos requerimientos de hardware que solicita el lenguaje para que sus aplicaciones funcionen correctamente. Además, permite aplicar técnicas de programación orientada a objetos. Es un lenguaje completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. No requiere definición de tipos de variables aunque se pueden evaluar por el tipo que estén manejando en tiempo de ejecución. El código fuente escrito en este lenguaje es invisible al navegador ya que es el servidor el encargado de ejecutar el código y enviar su resultado HTML al cliente, logrando una programación segura y confiable. Posee una amplia documentación, dado que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda (14).

JavaScript

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. Es manejado como un lenguaje de programación que se maneja principalmente para crear páginas web dinámicas. Se utiliza principalmente en la programación del lado del cliente. Actualmente los navegadores modernos interpretan el código JavaScript integrado en las páginas web (15).

Por todo lo anteriormente analizado se seleccionó para la programación del lado del servidor el lenguaje de programación PHP, debido a todas las facilidades de uso que brinda para la realización de sistemas sobre la web. En la programación del lado del cliente será usado el lenguaje Java Script ya que permite la creación de páginas web de forma dinámica.

1.3.4 Framework y librería de desarrollo

Un framework, en el desarrollo de software, es una estructura de soporte definida 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 software para ayudar a desarrollar y unir los diferentes componentes de un proyecto (16).

Symfony2

Symfony2 es la versión más reciente de Symfony, framework para desarrollar aplicaciones PHP. Se anunció por primera vez a principios de 2009 y supone un cambio radical tanto en arquitectura interna como en filosofía de trabajo respecto a sus versiones anteriores (17).

Esta versión ha sido ideada para utilizar al máximo todas las nuevas características de PHP 5.3 y por eso es uno de los frameworks PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajen en los proyectos. También es el framework que más ideas incorpora del resto de frameworks, incluso de aquellos que no están programados con PHP.

Algunas de las ventajas que Symfony brinda son:

- ✓ Administración centralizada de todos los servicios.
- ✓ Atención de solicitudes a través de múltiples canales.
- ✓ Fácil de configurar y adaptable a sus necesidades.
- ✓ Estructura altamente flexible e intuitiva.
- ✓ Administración de niveles de servicio.
- ✓ Administración de la base del conocimiento.
- ✓ Seguridad a través de perfiles.
- ✓ Plataforma independiente.

Symfony2 ha cambiado la carga automática de clases con respecto a sus versiones anteriores para ser más universal, más rápida e independiente de la necesidad de vaciar la caché. Esto provocó que el dilema del rendimiento de Symfony1 fuera resuelto logrando evacuar esta situación. Así mismo esta versión facilita el trabajo con la aplicación al lograr una integración entre sus componentes, o sea que puedes desarrollar una sola aplicación donde se maneje la parte gráfica y la controladora en una misma área.

Analizando lo anteriormente planteado se decide utilizar para el desarrollo del sistema Symfony2, teniendo en cuenta las características expuestas, que sostienen una actualización tecnológica mayor en correspondencia con la implementación de la primera versión.

ExtJS 3.3.1

Es una librería de JavaScript para el desarrollo de aplicaciones web enriquecidas, haciendo un uso intensivo de las tecnologías AJAX, XHTML, DHTML y DOM. Originalmente fue creado como una

Capítulo I: Fundamentos Teóricos

extensión de Yahoo User Interface (YUI). Incluye interoperabilidad con jQuery, Prototype y Scriptaculo.US. La versión 3.0 fue liberada el 3 de junio de 2009 con grandes mejoras (18).

Mejoras

- ✓ Nuevos ejemplos y componentes (incluye un componente para gráficas).
- ✓ Administración de memoria mejorada para el navegador Internet Explorer 6.
- ✓ Interfaz de programación de aplicaciones (API5) documentada y CSS refactorizado.

Por todo lo antes expuesto se decide utilizar framework Symfony2 y la librería ExtJS 3.3.1.

1.3.5 Herramienta ORM (Object Relational Mapper - Mapeo Relacional de Objeto)

El Mapeo Relacional de Objeto es la traducción lógica de cada objeto a la lógica tradicional. El proceso convierte cada tabla de la base de datos en una clase permitiéndola utilizar como si fuera un objeto.

Doctrine 2

El Proyecto Doctrine es el hogar de un conjunto seleccionado de bibliotecas PHP centrada principalmente en la prestación de servicios de persistencia y funcionalidades relacionadas. Contiene varios proyectos de excelencia, destacándose entre ellos el de la capa de abstracción de base de datos. A continuación se presentará algunos de sus proyectos (19).

Capa de abstracción de base de datos

Potente capa de abstracción de base de datos con muchas características para la introspección esquema de base de datos, gestión de esquema y la abstracción PDO.

Migraciones

Las migraciones de Doctrine ofrecen una funcionalidad adicional en la parte superior de la capa de abstracción de base de datos (DBAL) para versionar el esquema de base de datos y desplegar fácilmente cambios en él. Es una herramienta muy fácil de usar y potente.

Mapeo Relacional de Objeto (1.2)

Mapeo Relacional de Objeto (ORM) para PHP se encuentra en la parte superior de una capa de abstracción de bases de datos de gran alcance (DBAL). Una de sus principales características es la posibilidad de escribir consultas de base de datos en un dialecto propio orientado a objetos llamado SQL Query Language Doctrina (DQL), inspirado en Hiberna HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad sin necesidad de duplicación de código innecesario.

Common

El proyecto Doctrine Common es una biblioteca que proporciona la funcionalidad básica de extensiones para PHP.

Teniendo en cuenta estas características y conociendo la estrecha relación que guarda este proyecto con el framework seleccionado, la herramienta escogida para realizar este proceso es Doctrine 2.

1.3.6 Servidor web

Un servidor web es un programa que sirve datos en forma de Páginas Web, hipertextos o páginas HTML (HyperText Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos. La comunicación de estos datos entre cliente y servidor se hace por medio un protocolo, concretamente del protocolo Http. Con esto, un servidor Web se mantiene a la espera de peticiones HTTP, que son ejecutadas por un cliente HTTP; conocido como un Navegador Web.

Es la tecnología que tiene implícito programas informáticos que procesan aplicaciones realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente, generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente (21).

Apache 2.2

Apache 2.2 constituye una tecnología gratuita de código abierto. Es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos. Permite la creación de sitios web dinámicos mediante el uso de Server Side Includes (SSI) por sus siglas en inglés, de lenguajes de scripting como PHP, JavaScript, Python, entre otros. Se ejecuta en varios sistemas operativos. Posee una arquitectura modular que admite ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos. Tiene una alta configurabilidad en la creación y gestión de logs; y soporta personalizar la respuesta ante los posibles errores que se puedan generar en el servidor.

Partiendo de las características analizadas se resolvió usar para la implementación del sistema el servidor web Apache 2.2, teniendo en cuenta además, el lenguaje de programación web escogido para la programación del lado del servidor.

1.3.7 Servidor de base de datos

Servidores de Bases de Datos también conocidos como RDBMS (acrónimo en inglés de Relational DataBase Management Systems), son programas que permiten organizar datos en una o más tablas relacionadas. Son grandes proveedores de información para todo tipo de usuarios. Ellos deben ofrecer soluciones de forma fiable, rentable y de alto rendimiento. A estas tres características, se le debe añadir una más: debe proporcionar servicios de forma global y, en la medida de lo posible, independientemente de la plataforma. Ejemplos de estos servidores son: MySQL, Oracle-XE y PostgreSQL (23).

PostgreSQL 9.2

Es un sistema de gestión de base de datos relacional orientada a objetos, libre y publicado bajo la licencia BSD. Es un gestor de bases de datos de código abierto avanzado, ofreciendo control de concurrencia multi-versión, soportando casi todas las sintaxis SQL como subconsultas, transacciones y funciones definidas por el usuario, cuenta además con un amplio conjunto de enlaces con lenguajes de programación como C, C++, Java y Python . PostgreSQL está dirigido por una comunidad bien organizada de desarrolladores y organizaciones comerciales, de ahí el nivel de aceptación y de calidad del mismo.

Entre sus características se encuentran:

- ✓ Poseer una gran escalabilidad. Es capaz de ajustarse al número de Unidades Centrales de Procesamiento (Central Processing Units, CPU por sus siglas en inglés) y a la cantidad de memoria que posee el sistema de forma óptima.
- ✓ Tener la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos.
- ✓ La simplificación del proceso de administración de licencias de software, que no es necesario cuando se usa software libre.
- ✓ Poder lidiar con grandes volúmenes de datos.

PostgreSQL 9.2 cuenta con soporte nativo para JSON, que abarca los índices, las mejoras de replicación y de desempeño proporcionando un mecanismo eficaz para la creación y el almacenamiento de documentos para APIs web. Con la adición de una escalabilidad lineal de hasta 64 núcleos, las exploraciones de índices sólo y las reducciones en el consumo de energía de la CPU hace que esta versión haya mejorado significativamente la escalabilidad y la flexibilidad de desarrollo para las cargas de trabajo más exigentes (24).

Capítulo I: Fundamentos Teóricos

Teniendo en cuenta lo antes analizando, para el desarrollo del sistema se decidió utilizar el gestor de bases de datos PostgreSQL 9.2; por las particulares expuestas y además por la existencia en el centro de una línea de investigación relacionada con este gestor, que garantiza facilidades de experiencias y conocimientos que se pueden adquirir.

1.3.8 Entorno integrado de desarrollo

Para la ejecución del sistema se hizo necesario escoger un entorno integrado de desarrollo que proporcionara el uso y la integración de todas las tecnologías y lenguajes antes mencionados, por lo que se decidió utilizar el NetBeans IDE 7.0. Las mejoras realizadas a esta última versión en cuanto a la programación web pueden ser apreciadas en la siguiente descripción.

NetBeans IDE 7.0

El Entorno de Desarrollo Integrado (IDE, sigla en inglés de Integrated Development Environment) NetBeans es “una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso” (25).

La idea de seleccionar NetBeans 7.0 como IDE parte de su relación con Symfony, el framework de desarrollo escogido, ya que NetBeans en esta versión le da soporte, facilitando el trabajo del programador con el framework. De esta manera se tiene completamiento de código no solo de la sintaxis de PHP sino de las clases creadas que tengas dentro de tu proyecto y del `include_path` de NetBeans, que en este caso se le incluye la carpeta de la versión de Symfony que se esté utilizando y permite marcado de error.

Conclusiones del capítulo

En el capítulo se realizó una investigación de las soluciones existentes, tanto en el ámbito internacional como nacional en el campo de las herramientas y aplicaciones dedicadas a la gestión de soporte y de información. Se hizo además un análisis de la primera versión del Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos. Estos estudios permitieron dar comienzo al desarrollo de una aplicación capaz cubrir las necesidades actuales del grupo de Soporte. Teniendo en cuenta esto, se exhibe una síntesis de las tecnologías, herramientas y metodología a utilizar durante la elaboración del software fundamentando su selección. Se decidió usar

Capítulo I: Fundamentos Teóricos

como lenguaje de programación del lado del cliente JavaScript auxiliándose de la librería de ExtJS 3.3.1 y el lenguaje de programación PHP del lado del servidor. Se estableció utilizar como framework de desarrollo Symfony2.0, como IDE el NetBeans 7.0, como gestor de base de datos PostgreSQL 9.2, como metodología de desarrollo OpenUP y como herramienta CASE Visual Paradigm 6.4. Este ambiente de desarrollo seleccionado apoya la implementación del sistema propuesto, brindando agilidad y variedad de funciones a los desarrolladores.

Capítulo II: Características y Diseño del Sistema

CAPÍTULO II: CARACTERÍSTICAS Y DISEÑO DEL SISTEMA

Introducción

En este capítulo se describen las características y se realiza el rediseño del sistema a desarrollar. Para ello primeramente, se procede al levantamiento de requisitos funcionales y no funcionales a partir del modelo conceptual construido, basándose en los objetos relevantes del problema, además se identifican los casos de uso que contiene el sistema y se describe uno de sus casos de uso arquitectónicamente significativos, elemento guía en el proceso de desarrollo de software. Luego de haber comprendido estos artefactos se procede al rediseño de la solución propuesta. En esta etapa se propone un acercamiento a la implementación del sistema, evidenciado en la construcción de los diagramas de interacción del mismo y el modelo de clases del diseño siguiendo el estilo arquitectónico y los patrones de diseño seleccionados, proporcionando la base para proceder a la etapa de implementación. Finalmente se muestra el modelo de datos con la descripción de las tablas más significativas.

2.1 Modelo de Dominio

El modelo de dominio es una representación gráfica del ambiente real de los objetos del proyecto. Son utilizados para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. Resulta ser un artefacto de ayuda en la comprensión de los conceptos que utilizan los usuarios, con los que trabajan y con los que deberá trabajar el software (26). A continuación se muestra el modelo de dominio (ver figura 4) del Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnología de Gestión de Datos v2.0

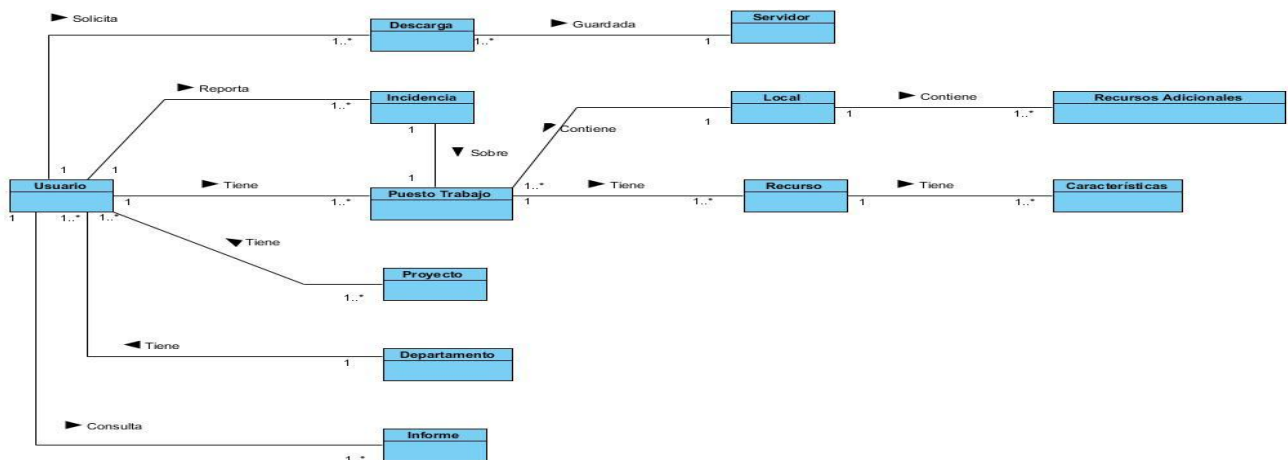


Figura 5: Modelo de dominio.

Capítulo II: Características y Diseño del Sistema

En este diagrama se refleja el funcionamiento de los procesos de las solicitudes de descarga y los reportes de incidencias sobre el los recursos de los puestos de trabajo. A través de la relaciones entre las clases se define el flujo de estos procesos. Los números en las relaciones referencian la cantidad de instancias de una clase a otra siguiendo la dirección de las relaciones.

2.1.1 Descripción de las Clases del Dominio

Usuario: Es todo el personal del centro que interactúa con los recursos existentes.

Descarga: Son las solicitudes de descargas notificadas por cualquier personal perteneciente al centro, que aluden a una información o aplicación específica que no se encuentre en la universidad.

Incidencia: Es un reporte realizado por un usuario sobre cualquier situación fuera de lo normal en un recurso determinado. Permite informar el estado técnico de un recurso en específico.

Puesto Trabajo: Se refiere al conjunto de recursos para desarrollar actividad informática que se le asigna a una persona dentro de la organización.

Proyecto: Es una estructura organizacional que agrupa a personal capacitado para la realización de tareas.

Departamento: Es la estructura con mayor nivel organizacional, dentro de la cual se encuentran los proyectos del centro.

Informe: Es una información detallada obtenida sobre las entidades que intervienen en el dominio. Relaciona casi siempre datos de más de una clase.

Servidor: Es un ordenador usado como servidor donde se almacenan las descargas realizadas.

Local: Hace referencia a los diferentes locales existentes en el centro. Estos están divididos en diferentes categorías.

Recurso: Se refiere a los recursos tangibles que están ubicados en los puestos de trabajo de los locales.

Recursos Adicionales: Son componentes adicionales de ambientación que pueden existir en un local determinado.

Características: Son las cualidades que puede tener un recurso. Puede ser información de hardware, software, accesorios informáticos y ambientación.

Capítulo II: Características y Diseño del Sistema

2.2 Requisitos del sistema

Los requisitos de un sistema establecen con detalles las funciones, servicios y restricciones operativas del software. Proporcionan una descripción de lo que debe hacer la aplicación y que necesita para un correcto funcionamiento en el entorno donde se lleve a cabo su despliegue (27).

2.2.1 Requisitos funcionales

Definen el comportamiento interno de un software, representan capacidades o condiciones que el sistema debe poseer (28). Muestran las funcionalidades que deben satisfacerse para cumplir con las especificaciones de software. Para el desarrollo del Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos v2.0 se definieron los siguientes requisitos funcionales, especificando la manera en que este deben reaccionar ante determinadas entradas y cómo debe comportarse el sistema en situaciones particulares:

RF 1 Adicionar usuario.

Descripción: Se registra en el sistema un nuevo usuario.

Entrada: Se registran los detalles del nuevo usuario (usuario, modo de autenticación ldap o local, nombre y apellidos, rol, categoría, contraseña).

Salida: Se introduce un nuevo usuario en la base de datos y se muestra un mensaje de confirmación.

RF 2 Modificar usuario.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad usuario.

Entrada: Cambio en los campos que desean modificar del usuario.

Salida: Se obtiene el usuario modificado y se muestra un mensaje de confirmación.

RF 3 Eliminar usuario.

Descripción: Se elimina del sistema el usuario seleccionado previamente.

Entrada: Se selecciona de la lista de usuarios el que se desea eliminar.

Salida: Se actualiza la lista de usuarios y se muestra un mensaje de confirmación.

RF 4 Autenticar usuario.

Descripción: Permitirá a los usuarios realizar el proceso de autenticación en la aplicación ofreciendo a la misma los datos correspondientes (usuario y contraseña).

Entrada: Contraseña y usuario de la persona que desea entrar al sistema.

Salida: Se actualiza en la Base de Datos el campo logged del usuario.

Capítulo II: Características y Diseño del Sistema

RF 5 Adicionar categoría de usuario.

Descripción: Se registra en el sistema una nueva categoría de usuario.

Entrada: Se registran los detalles de la nueva categoría de usuario (nombre y descripción).

Salida: Se introduce en la base de datos una nueva categoría de usuario y se muestra un mensaje de confirmación.

RF 6 Modificar categoría de usuario.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad categoría de usuario.

Entrada: El cambio en los campos que desean modificar de la categoría de usuario.

Salida: Categoría de usuario modificada y se muestra un mensaje de confirmación.

RF 7 Eliminar categoría de usuario.

Descripción: Se elimina del sistema la categoría de usuario seleccionada previamente.

Entrada: Se selecciona de la lista de categorías de usuarios la que se desea eliminar.

Salida: Se actualiza la lista de categorías de usuarios y se muestra un mensaje de confirmación.

RF 8 Adicionar departamento.

Descripción: Se registra en el sistema un nuevo departamento.

Entrada: Detalles del nuevo departamento que se desea a adicionar (nombre, descripción).

Salida: Se introduce en la base de datos un nuevo departamento y se muestra un mensaje de confirmación.

RF 9 Modificar departamento.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad departamento.

Entrada: El cambio en los campos que desean modificar del departamento.

Salida: Departamento modificado y se muestra un mensaje de confirmación.

RF 10 Eliminar departamento.

Descripción: Se elimina del sistema el departamento seleccionado previamente.

Entrada: Se selecciona de la lista de departamentos el que se desea eliminar.

Salida: Se actualiza la lista de departamentos y se muestra un mensaje de confirmación.

RF 11 Adicionar proyecto.

Descripción: Se registra en el sistema un nuevo proyecto.

Entrada: Se registran los detalles del nuevo proyecto (nombre, departamento, descripción).

Capítulo II: Características y Diseño del Sistema

Salida: Se introduce en la base de datos un nuevo proyecto y se muestra un mensaje de confirmación.

RF 12 Modificar proyecto.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad proyecto.

Entrada: El cambio en los campos que desean modificar del proyecto.

Salida: Proyecto modificado y se muestra un mensaje de confirmación.

RF 13 Eliminar proyecto.

Descripción: Se elimina del sistema el proyecto seleccionado previamente.

Entrada: Se selecciona de la lista de proyectos el que se desea eliminar.

Salida: Se actualiza la lista de proyectos y se muestra un mensaje de confirmación.

RF 14 Adicionar categoría de local.

Descripción: Se registra en el sistema una nueva categoría de local.

Entrada: Se registran los detalles de la nueva categoría de local (nombre, descripción).

Salida: Se introduce en la base de datos una nueva categoría de local y se muestra un mensaje de confirmación.

RF 15 Modificar categoría de local.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad categoría de local.

Entrada: El cambio en los campos que desean modificar de la categoría de local.

Salida: Categoría de local modificada y se muestra un mensaje de confirmación.

RF 16 Eliminar categoría de local.

Descripción: Se elimina del sistema la categoría de local seleccionada previamente.

Entrada: Se selecciona de la lista de categorías de local la que se desea eliminar.

Salida: Se actualiza la lista de categorías de local y se muestra un mensaje de confirmación.

RF 17 Adicionar local.

Descripción: Se registra en el sistema un nuevo local.

Entrada: Se registran los detalles del nuevo local (nombre, categoría, descripción).

Salida: Se introduce en la base de datos un nuevo local y se muestra un mensaje de confirmación.

RF 18 Modificar local.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad local.

Entrada: El cambio en los campos que desean modificar del local.

Salida: Local modificado y se muestra un mensaje de confirmación.

Capítulo II: Características y Diseño del Sistema

RF 19 Eliminar local.

Descripción: Se elimina del sistema el local seleccionada previamente.

Entrada: Se selecciona de la lista de locales el que se desea eliminar.

Salida: Se actualiza la lista de locales y se muestra un mensaje de confirmación.

RF 20 Adicionar característica de recurso.

Descripción: Se registra en el sistema una nueva característica de recurso.

Entrada: Se registran los detalles de la nueva característica de recurso (nombre, descripción).

Salida: Se introduce en la base de datos una nueva característica de recurso y se muestra un mensaje de confirmación.

RF 21 Modificar característica de recurso.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad característica de recurso.

Entrada: El cambio en los campos que desean modificar de la característica de recurso.

Salida: Característica de recurso modificada y se muestra un mensaje de confirmación.

RF 22 Eliminar característica de recurso.

Descripción: Se elimina del sistema la característica de recurso seleccionada previamente.

Entrada: Se selecciona de la lista de características de recurso la que se desea eliminar.

Salida: Se actualiza la lista de características de recurso y se muestra un mensaje de confirmación.

RF 23 Adicionar Puesto de trabajo.

Descripción: Se registra en el sistema un nuevo puesto de trabajo correspondiente a un local.

Entrada: Se registran los detalles del nuevo puesto de trabajo (posición, descripción).

Salida: Se introduce un nuevo puesto de trabajo en la base de datos y se muestra un mensaje de confirmación.

RF 24 Modificar Puesto de trabajo.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad puesto de trabajo.

Entrada: El cambio en los campos que desean modificar del puesto de trabajo.

Salida: Puesto de trabajo modificado y se muestra un mensaje de confirmación.

RF 25 Eliminar Puesto de trabajo.

Descripción: Se elimina del sistema el puesto de trabajo seleccionada previamente.

Entrada: Se selecciona de la lista de puestos de trabajo el que se desea eliminar.

Capítulo II: Características y Diseño del Sistema

Salida: Se actualiza la lista de puestos de trabajo y se muestra un mensaje de confirmación.

RF 26 Adicionar recurso.

Descripción: Se registra en el sistema un nuevo recurso correspondiente a un puesto de trabajo.

Entrada: Se registran los detalles del nuevo recurso (categoría, características).

Salida: Se introduce un nuevo recurso en la base de datos y se muestra un mensaje de confirmación.

RF 27 Modificar recurso.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad recurso.

Entrada: El cambio en los campos que desean modificar de recurso.

Salida: Recurso modificado y se muestra un mensaje de confirmación.

RF 28 Eliminar recurso.

Descripción: Se elimina del sistema el recurso seleccionado previamente.

Entrada: Se selecciona de la lista de recursos el que se desea eliminar.

Salida: Se actualiza la lista de recursos y se muestra un mensaje de confirmación.

RF 29 Adicionar usuario a puesto de trabajo.

Descripción: Permitirá al puesto de trabajo seleccionado previamente asignar uno o varios usuarios de los existentes en el sistema a los cuales el usuario conectado tiene permiso para listarlo.

Entrada: Usuario que se selecciona para asignarle al puesto de trabajo.

Salida: Se introduce en la base de datos nuevos usuarios para el puesto de trabajo seleccionado y se muestra un mensaje de confirmación.

RF 30 Modificar usuario del puesto de trabajo.

Descripción: Permitirá modificar el usuario que se tiene asignado al puesto de trabajo seleccionado previamente.

Entrada: Usuario que se selecciona para asignarle al puesto de trabajo.

Salida: Puesto de trabajo previamente seleccionado con usuario asignado modificado.

RF 31 Eliminar usuario del puesto de trabajo.

Descripción: Permitirá invalidar del puesto de trabajo seleccionado previamente a uno o varios usuarios que le estén asignados.

Entrada: Puesto de trabajo y la selección de los usuarios que se le desautorizarán.

Salida: El puesto de trabajo sin los usuarios que tenía asignados.

RF 32 Adicionar incidencia.

Capítulo II: Características y Diseño del Sistema

Descripción: Se registra en el sistema una nueva incidencia.

Entrada: Se registran los detalles de la nueva incidencia (tipo de incidencia, puesto de trabajo, descripción).

Salida: Se registra una nueva incidencia en la base de datos y se muestra un mensaje de confirmación.

RF 33 Modificar incidencia.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad incidencia.

Entrada: El cambio en los campos que desean modificar de la incidencia.

Salida: Incidencia modificada y se muestra un mensaje de confirmación.

RF 34 Eliminar incidencia.

Descripción: Se elimina del sistema la incidencia seleccionada previamente.

Entrada: Se selecciona de la lista de incidencias la que se desea eliminar.

Salida: Se actualiza la lista de incidencias y se muestra un mensaje de confirmación.

RF 35 Modificar estado de incidencia.

Descripción: Permitirá modificar el campo estado que componen a la entidad incidencia.

Entrada: El cambio en lo campo que desea modificar de la incidencia.

Salida: Campo de estado de incidencia modificado y se muestra un mensaje de confirmación.

RF 36 Adicionar solicitud de descarga.

Descripción: Se registra en el sistema una nueva solicitud de descarga.

Entrada: Se registran los detalles de la nueva solicitud de descarga (url, descripción).

Salida: Se registra en la base de datos una nueva solicitud de descarga y se muestra un mensaje de confirmación.

RF 37 Modificar solicitud de descarga.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad solicitud de descarga.

Entrada: El cambio en los campos que desean modificar de la solicitud de descarga.

Salida: Solicitud de descarga modificada y se muestra un mensaje de confirmación.

RF 38 Eliminar solicitud de descarga.

Descripción: Se elimina del sistema la solicitud de descarga seleccionada previamente.

Entrada: Se selecciona de la lista de solicitudes de descargas la que se desea eliminar.

Salida: Se actualiza la lista de solicitudes de descargas y se muestra un mensaje de confirmación.

Capítulo II: Características y Diseño del Sistema

RF 39 Modificar estado de solicitud de descarga.

Descripción: Permitirá modificar el campo estado que componen a la entidad solicitud de descarga.

Entrada: El cambio en lo campo que desea modificar de la solicitud de descarga.

Salida: Campo de estado de solicitud de descarga modificado y se muestra un mensaje de confirmación.

RF 40 Visualizar reportes.

Descripción: Permitirá mostrar los diferentes reportes que se pueden generar a los cuales el usuario conectado tiene permiso.

Entrada: Reporte que se desea generar.

Salida: Reporte generado.

RF 41 Visualizar trazas.

Descripción: Permitirá mostrar las diferentes trazas que se han sucedido en el sistema.

Entrada: Seleccionar la opción de trazas.

Salida: Lista de trazas.

RF 42 Eliminar trazas.

Descripción: Se elimina del sistema la traza seleccionada previamente.

Entrada: Se selecciona de la lista de trazas la que se desea eliminar.

Salida: Se actualiza la lista de trazas y se muestra un mensaje de confirmación.

RF 43 Adicionar categoría de Recurso.

Descripción: Se registra en el sistema una nueva categoría de recurso.

Entrada: Se registran los detalles de la nueva categoría de recurso (nombre, descripción).

Salida: Se registra en la base de datos una nueva categoría de recurso y se muestra un mensaje de confirmación.

RF 44 Modificar categoría de Recurso.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad categoría de recurso.

Entrada: El cambio en los campos que desean modificar de la categoría de recurso.

Salida: Categoría de recurso modificada y se muestra un mensaje de confirmación.

RF 45 Eliminar categoría de Recurso.

Descripción: Se elimina del sistema la categoría de recurso seleccionada previamente.

Entrada: Se selecciona de la lista de categorías de recurso la que se desea eliminar.

Capítulo II: Características y Diseño del Sistema

Salida: Se actualiza la lista de categorías de recurso y se muestra un mensaje de confirmación.

RF 46 Adicionar proyecto a usuario.

Descripción: Permitirá al usuario seleccionado previamente asignar uno o varios proyectos de los existentes en el sistema a los cuales el usuario conectado tiene permiso para listarlo.

Entrada: Proyecto que se selecciona para asignarle al usuario.

Salida: se introduce en la base de datos nuevos proyectos para el usuario seleccionado y se muestra un mensaje de confirmación.

RF 47 Modificar proyecto del usuario.

Descripción: Permitirá modificar el proyecto que se tiene asignado al usuario seleccionado previamente.

Entrada: Proyecto que se selecciona para asignarle al usuario.

Salida: Usuario previamente seleccionado con proyecto asignado modificado.

RF 48 Eliminar proyecto del usuario.

Descripción: Permitirá invalidar del usuario seleccionado previamente a uno o varios proyectos que le estén asignados.

Entrada: Usuario y la selección de los proyectos que se le desautorizarán.

Salida: El usuario sin los proyectos que tenía asignados.

RF 49 Adicionar Recursos adicionales.

Descripción: Se registra en el sistema un nuevo recurso adicional correspondiente a un local.

Entrada: Se registran los detalles del nuevo recurso adicional (nombre, descripción).

Salida: Se registra en la base de datos un nuevo recurso adicional y se muestra un mensaje de confirmación.

RF 50 Modificar Recursos adicionales.

Descripción: Permitirá modificar cualquiera de los campos que componen a la entidad recurso adicional.

Entrada: El cambio en los campos que desean modificar del recurso adicional.

Salida: Recurso adicional modificado y se muestra un mensaje de confirmación.

RF 51 Eliminar Recursos adicionales.

Descripción: Se elimina del sistema el recurso adicional seleccionado previamente.

Entrada: Se selecciona de la lista de recursos adicionales el que se desea eliminar.

Salida: Se actualiza la lista de recursos adicionales y se muestra un mensaje de confirmación.

Capítulo II: Características y Diseño del Sistema

2.2.2 Requisitos no funcionales

Indican cómo debe ser el sistema, especifican criterios que pueden usarse para juzgar las operaciones que una aplicación realiza. Son propiedades o cualidades que el producto debe tener. Estas propiedades se ven como las características que hacen al software atractivo, usable, rápido o confiable (29). Los requisitos no funcionales del sistema son:

RNF1 Eficiencia

El sistema responderá de manera inmediata siempre que se cumplan los requisitos de hardware necesarios:

Para 100 peticiones concurrentes el sistema responderá en menos de 10 segundos.

Para 250 peticiones concurrentes el sistema responderá en menos de 25 segundos.

RNF2 Interfaz

El sistema deberá presentar una interfaz agradable a la vista del usuario. Se incorporarán sólo las imágenes necesarias para no sobrecargar el diseño.

RNF3 Seguridad

La información solo podrá ser vista por los usuarios con el nivel de acceso requerido para ello; permitiendo que las funcionalidades del sistema se muestren de acuerdo al nivel del usuario que esté activo.

RNF4 Hardware

Para el servidor de aplicaciones:

- ✓ Ordenador Pentium IV o superior, con 1,7 GHz de velocidad de microprocesador.
- ✓ Disco Duro de al menos 5Gb de capacidad para la instalación del sistema.
- ✓ Memoria RAM de al menos 1 GB.

Para el ordenador cliente:

- ✓ Ordenador Pentium III o superior, con 1.7 GHz de velocidad de microprocesador.
- ✓ Memoria RAM de al menos 256 MB.

RNF5 Software

Para el servidor de aplicaciones:

- ✓ Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 10.10 o superior, Debian 4 GNU/LINUX o superior.
- ✓ Paquetes: apache2, php5, libapache2-mod-php5, php5-cli, php5-mysql, php5-pgsql, php5-sqlite, php5-sybase, php5-xsl y php5-gd.

Capítulo II: Características y Diseño del Sistema

- ✓ PostgreSQL versión 9.0 o superior.
- ✓ PgAdmin III o algún administrador para PostgreSQL.
- ✓ PostgreSQL debe estar correctamente configurado para aceptar conexiones vía TCP/IP utilizando el método de autenticación por sha512.
- ✓ Usuarios con permisos de administrador en el Sistema Operativo y con privilegios para instalar la base de datos.

Para el ordenador cliente:

- ✓ Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 10.10 o superior, Debian 4 GNU/LINUX o superior, Microsoft Window XP o superior.
- ✓ Navegador web: Mozilla Firefox.

RNF6 Portabilidad

El sistema deberá ser desarrollado de forma tal que sea multiplataforma.

2.3 Modelo de casos de uso

El modelo de casos de uso permite describir los requisitos funcionales del sistema, dando lugar a un acuerdo entre el cliente y los desarrolladores de la aplicación. Proporciona una explicación clara y consistente de lo que debería hacer el software, de modo que el modelo se use a lo largo del proceso de desarrollo. Posibilita que se obtenga una base para realizar verificaciones del producto informático, siendo la entrada fundamental para el análisis, el diseño y las pruebas (30).

Caso de Uso

Los casos de uso son descripciones funcionales del sistema que permiten definir los límites del software y las relaciones entre la aplicación y el entorno; describen cómo los actores pueden usar un sistema. Especifican una secuencia de acciones que debe devolver algún resultado de valor a un actor (30).

Actores del sistema

Un actor es una entidad externa al sistema representada por un ser humano, una máquina o un software que interactúa con el sistema. Representa un tipo particular de usuario del negocio más que un usuario físico, debido a que varios usuarios físicos pueden realizar el mismo papel en relación al negocio, o sea, ser instancias de un mismo actor (30). En la tabla 1 se muestran los actores que tendrá el sistema.

Actor	Descripción
Administrador	Es el único usuario que tendrá los permisos para

Capítulo II: Características y Diseño del Sistema

	realizar todas las funcionalidades del sistema.
Usuario	Son todos los usuarios del sistema. Sus funcionalidades serán gestionar incidencia de puesto de trabajo y gestionar solicitud de descarga personal.
Técnico	Tiene permiso para gestionar los puestos de trabajo de cada uno de los locales que existan en el sistema y obtendrá además los mismo permisos que el actor Usuario.
JefeDepartamento	Tendrá las mismas facilidades de opciones que el actor Usuario. Además se le permitirá gestionar los proyectos, los usuarios, los usuarios de los puestos de trabajo y podrá de visualizar los reportes que estén disponibles en el sistema.
LDAP	El Protocolo Ligero de Acceso a Directorios (Lightweight Directory Access Protocol, LDAP por sus siglas en inglés) es un conjunto de protocolos de acceso a directorios de información. De él se utilizarán las prestaciones para la autenticación, regidas por los elementos usuario y contraseña, que representarán a todos los usuarios del dominio UCI. Todos los usuarios de la universidad.

Tabla 1: Actores del sistema.

La tabla 1 muestra los 5 actores del sistema que son: Administrador, Usuario, Técnico, JefeDepartamento y el conjunto de protocolos de acceso a directorios de información, usuarios del dominio UCI (LDAP). En ella se describe lo que representa cada cual y las actividades que le corresponden a cada uno, según los permisos que se le tienen predefinidos en la lógica del sistema.

Patrones de Casos de Uso

Son comportamientos que deben existir en la aplicación, ayudan a describir qué es lo que debe hacer el software, es decir, describen el uso del sistema y cómo este interactúa con los usuarios. Estos patrones son utilizados generalmente como plantillas que detallan como debería ser estructurados y organizados

Capítulo II: Características y Diseño del Sistema

los casos de uso. Son patrones que capturan mejores prácticas para modelar casos de uso (30). Estos tienen varias clasificaciones, a continuación se muestran algunas de ellas.

- ✓ **CRUD:** patrón definido en los casos de uso donde se quiere realizar altas, bajas, cambios y consultas a alguna entidad del sistema. Su nombre es un acrónimo de las palabras en inglés Create, Read, Update, Delete que traducidas al español son crear, leer, actualizar y eliminar.
- ✓ **Concordancia:** extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de Casos de Uso y es expresado por separado. Tiene dos variantes: Reuso y Adición.
- ✓ **Extensión Concreta o Inclusión:** este patrón está dividido en concreta extensión o concreta inclusión. Extensión consiste en dos casos de uso y una relación extendida entre ellos. Se aplica cuando un flujo puede extender el flujo de otro caso de uso así como ser realizado en sí mismo. Inclusión se define cuando existe una relación del caso de uso base al caso de uso de inclusión.
- ✓ **Múltiples Actores:** se define cuando varios actores interactúan con un mismo caso de uso, y ellos influyen sobre él de una forma común o diferente.

2.3.1 Diagrama de casos de uso del sistema

El diagrama de casos de uso del sistema, es el encargado de recoger el comportamiento de un sistema desde el punto de vista de los usuarios. Se utiliza para describir las especificidades de un software y documentar su comportamiento, mostrando la relación que existe entre el usuario final y las funcionalidades (30). La figura 6 muestra el diagrama de casos de uso del sistema a desarrollar.

Capítulo II: Características y Diseño del Sistema

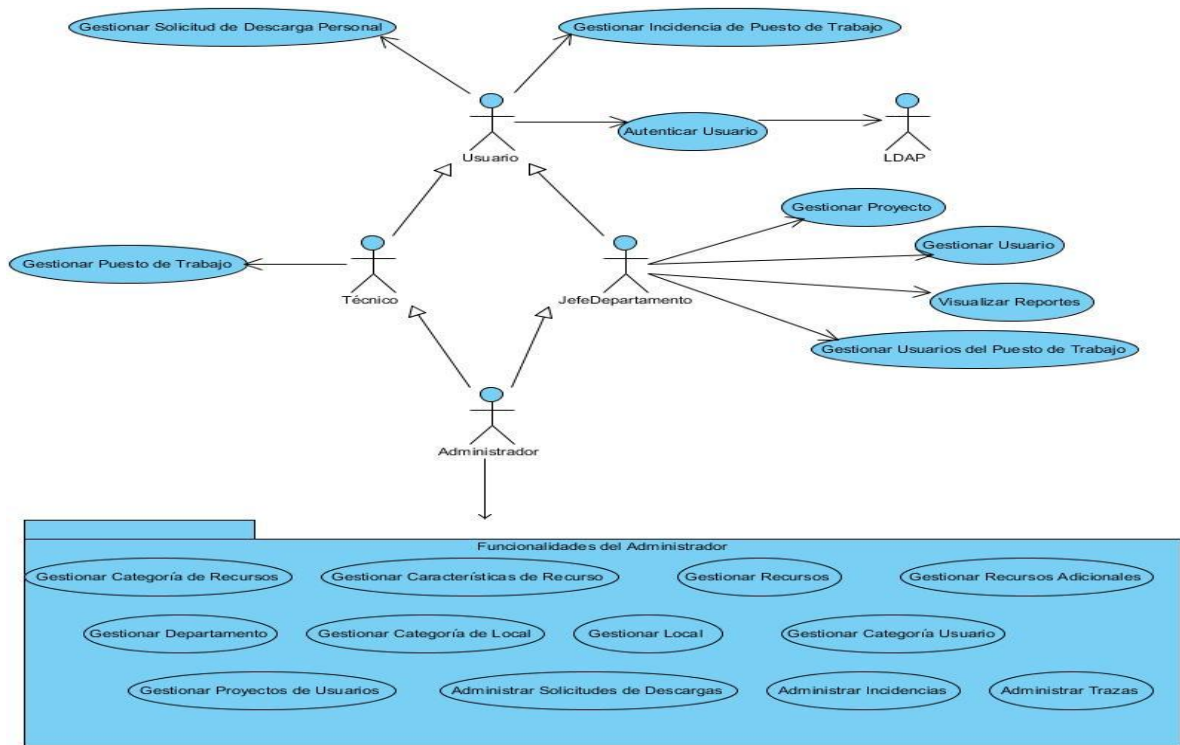


Figura 6: Diagrama de casos de uso del sistema.

La figura 6 muestra los 20 casos de uso del sistema, de ellos 4 son arquitectónicamente significativos (Gestionar Usuario, Gestionar Puesto de Trabajo, Gestionar Recurso y Gestionar Usuarios del Puesto de Trabajo). El actor Usuario podrá realizar las funcionalidades Gestionar Solicitud de Descarga Personal y Gestionar Incidencias de Puesto de Trabajo, siendo estas operaciones comunes para todos los demás actores del sistema. El Técnico tendrá la funcionalidad de Gestionar Puesto de Trabajo. El JefeDepartamento contemplará las responsabilidades de Gestionar Proyectos, Visualizar Reportes, Gestionar Usuario y Gestionar Usuarios del Puesto de Trabajo. El actor Administrador realizará todas las demás funcionalidades siendo el único capaz de llevarlas a cabo.

Para el sistema que se desea implementar se definieron 51 requisitos funcionales agrupados en 20 casos de uso teniendo en cuenta los patrones: CRUD en sus dos vertientes Total y Parcial vigente en el Gestionar Incidencia de Puesto de Trabajo y Administrar Incidencia respectivamente; y Múltiples actores en sus variantes rol común y roles diferentes, presente en la vinculación de todos los actores con el

Capítulo II: Características y Diseño del Sistema

Usuario para realizar las mismas funciones de él y en la relación que se da entre el Autenticar Usuario, el Usuario y LDAP.

2.3.2 Descripción de los casos de uso del sistema

Mediante la descripción de los casos de uso del sistema se detalla la secuencia de eventos que los actores llevan a cabo para completar un determinado proceso a través del sistema. La siguiente tabla describe los procesos adicionar, modificar y eliminar del caso de uso Gestionar puesto de trabajo, comenzando en el momento donde el usuario selecciona el puesto de trabajo al que le realizará una de las acciones anteriores.

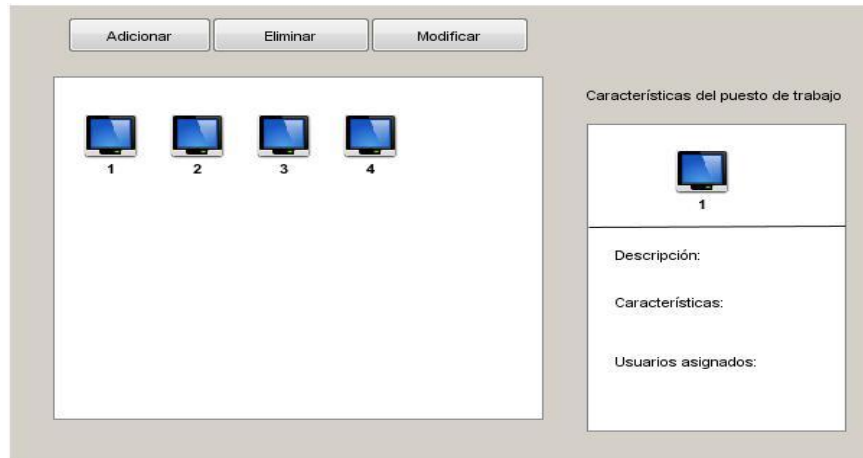
Descripción del caso de uso gestionar puesto de trabajo

Caso de Uso:	Gestionar puesto de trabajo
Actores:	Usuario
Resumen:	El caso de uso comienza al seleccionar un local y termina al realizar una de las siguientes operaciones sobre uno de los puestos de trabajo del local: Adicionar, Modificar o Eliminar.
Precondiciones:	El actor debe estar autenticado en el sistema y debe tener permisos para realizar estas operaciones. Debe existir al menos un local y un puesto de trabajo dentro del mismo. Debe estar activo el local donde se está gestionando el puesto de trabajo. Debe estar seleccionado el puesto de trabajo al que se le realizará la acción de “Modificar” o “Eliminar”.
Referencias:	RF46, RF47, RF48.
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1. El usuario selecciona el local sobre el cual desea gestionar los puestos de trabajo.	2. El sistema muestra una interfaz donde aparecen las opciones “Adicionar”, “Modificar” y “Eliminar” y una interfaz con los puestos de trabajo existentes en el local seleccionado.

Capítulo II: Características y Diseño del Sistema

<p>3. El usuario selecciona una opción.</p>	<p>4. Si selecciona:</p> <ul style="list-style-type: none"> ✓ “Adicionar”, ver Sección “Adicionar puesto de trabajo”. ✓ “Modificar”, ver Sección “Modificar puesto de trabajo”. ✓ “Eliminar”, ver Sección “Eliminar puesto de trabajo”.
---	--

Prototipo de Interfaz

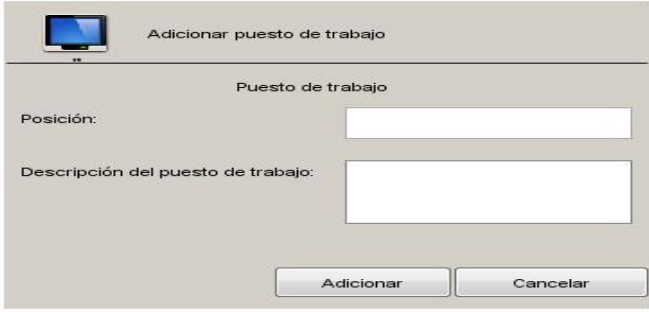


Sección “Adicionar puesto de trabajo”

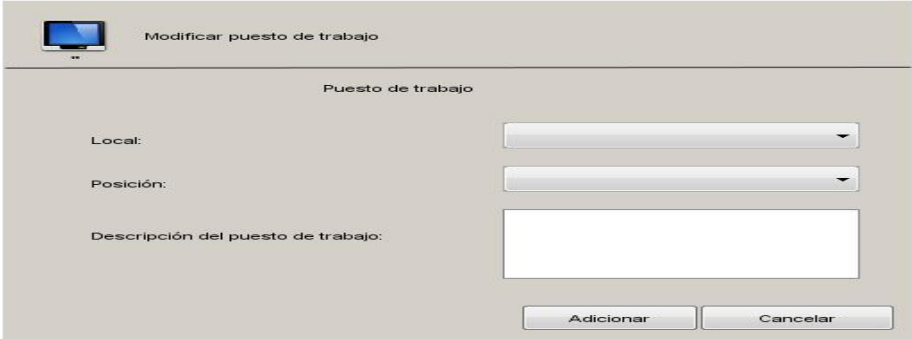
Flujos Básico

Acción del Actor	Respuesta del Sistema
	1. El sistema muestra una interfaz para introducir los datos del puesto de trabajo.
2. El usuario introduce los datos necesarios, y selecciona el botón “Adicionar”.	3. El sistema verifica que no existan campos vacíos. En caso contrario ver Flujo alterno 3. 3.1 El sistema verifica que los datos introducidos sean válidos. En caso contrario ver Flujo alterno 3.1

Capítulo II: Características y Diseño del Sistema

	4. El sistema adiciona el puesto de trabajo y muestra un mensaje de confirmación.
5. El usuario selecciona el botón "Aceptar".	6. Ver paso 2 del Flujo Normal de Eventos.
Prototipo de Interfaz	
	
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
	3 El sistema muestra un mensaje informando que existen campos vacíos.
4. El usuario selecciona el botón "Aceptar".	5. Ver paso 1 del Flujo Básico de Eventos.
	3.1 El sistema muestra un mensaje informando que existe algún campo inválido.
4. El usuario selecciona el botón "Aceptar".	5. Ver paso 1 del Flujo Básico de Eventos.
Sección "Modificar puesto de trabajo"	
Flujos Básico	
Acción del Actor	Respuesta del Sistema
	1. El sistema muestra una interfaz para modificar los datos del puesto de trabajo seleccionado.
2. El usuario modifica los datos necesarios, y selecciona el botón "Adicionar".	3. El sistema verifica que no existan campos vacíos. En caso contrario ver Flujo

Capítulo II: Características y Diseño del Sistema

	<p>alterno 3.</p> <p>3.1 El sistema verifica que los datos introducidos sean válidos. En caso contrario ver Flujo alterno 3.1</p>
	<p>4. El sistema modifica el puesto de trabajo y muestra un mensaje de confirmación.</p>
<p>5. El usuario selecciona el botón "Aceptar".</p>	<p>6. Ver paso 2 del Flujo Normal de Eventos.</p>
<p>Prototipo de Interfaz</p>	
	
<p>Flujos Alternos</p>	
<p>Acción del Actor</p>	<p>Respuesta del Sistema</p>
	<p>3 El sistema muestra un mensaje informando que existen campos vacíos.</p>
<p>4. El usuario selecciona el botón "Aceptar".</p>	<p>5. Ver paso 1 del Flujo Básico de Eventos.</p>
	<p>3.1 El sistema muestra un mensaje informando que existe algún campo inválido.</p>
<p>4. El usuario selecciona el botón "Aceptar".</p>	<p>5. Ver paso 1 del Flujo Básico de Eventos.</p>
<p>Sección "Eliminar puesto de trabajo"</p>	
<p>Flujos Básico</p>	
<p>Acción del Actor</p>	<p>Respuesta del Sistema</p>
	<p>1. El sistema muestra un mensaje para confirmar la acción de eliminar.</p>

Capítulo II: Características y Diseño del Sistema

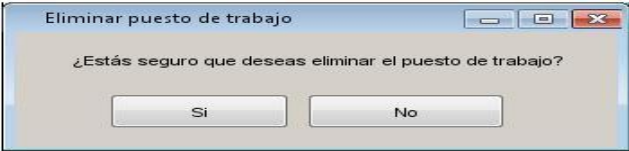
2. El usuario selecciona el botón "Sí".	3. El sistema elimina el puesto de trabajo y muestra un mensaje de notificación.
4. El usuario selecciona el botón "Aceptar".	5. Ver paso 2 del Flujo Normal de Eventos.
Prototipo de Interfaz	
	
Poscondiciones:	Queda adicionado, modificado o eliminado un puesto de trabajo previamente seleccionado por el usuario.

Tabla 2: Descripción del caso de uso Gestionar puesto de trabajo.

2.4 Modelo de diseño

El modelo de diseño tiene como entrada esencial el modelo de análisis. Es donde se modela el sistema y se encuentra su forma para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. El resultado de este proceso son especificaciones detalladas de todos los objetos, incluyendo sus operaciones y atributos (31).

2.4.1 Diagrama de Clases del Diseño

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema, y además se obtiene como resultado del refinamiento del modelo conceptual (31).

Diagrama de clases del diseño Gestionar Puesto de Trabajo

Capítulo II: Características y Diseño del Sistema

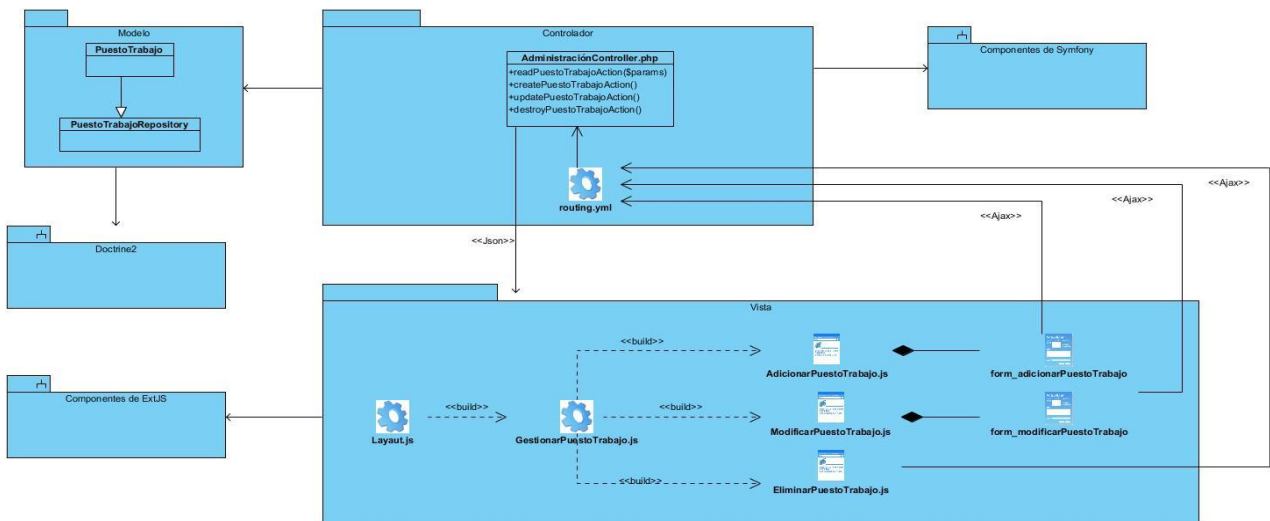


Figura 7: Diagrama de clases del diseño Gestionar Puesto de trabajo.

La figura 7 muestra la descripción del proceso del caso de uso Gestionar Puesto de Trabajo. En la estructura Vista se encuentra el layout.js que construye la interfaz principal del sistema. Este según las acciones del usuario muestra una de las paginas clientes (adicionar, modificar o eliminar) y muestra en el caso del adicionar y el modificar un formulario para introducir datos, el eliminar no tiene ningún formulario asociado. La información introducida en la vista es enviada a la clase routing.yml que se encuentra en el paquete Controlador y esta ejecuta los métodos asociados a la información recibida. Los métodos del controlador envían y obtienen los datos del componente Modelo donde se encuentran las clases mapeadas de la base de datos del sistema.

2.4.2 Patrón de arquitectura

El sistema será desarrollado sobre Symfony implicando que se manipule como patrón arquitectónico el mismo que este utiliza, el patrón Modelo-Vista-Controlador (MVC).

Modelo-Vista-Controlador (MVC):

El principio más importante de la arquitectura MVC es la separación del código del sistema en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica en el controlador.

Modelo: es la representación de la información que maneja el sistema. El modelo en sí son los datos puros que puestos en contexto del sistema proveen de información al usuario y al sistema mismo.

Capítulo II: Características y Diseño del Sistema

Vista: es la representación del modelo en forma gráfica, disponible para la interacción con el usuario. En el caso de una aplicación web, la “Vista” es una página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

Controlador: es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el modelo en caso de ser necesario (32).

En sistema las clases JS generarán las interfaces formando parte de la capa Vista. Las clases Controller.php y el routing.yml formarán la capa Controladora, debido a que son estas las encargadas de responder las peticiones de los usuarios. Por último la capa Modelo la integrarán las clases entity.php y entityRepository.php, teniendo en cuenta que ellas representan los datos del dominio de la aplicación.

2.4.3 Patrones de diseño utilizados

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Proporciona un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan (33).

Patrones GRASP:

Los Patrones Generales de Software para Asignación de Responsabilidades (General Responsibility Assignment Software Patterns, GRASP por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, siendo esta, una de las tareas fundamentales en el diseño de un software (34). Los patrones GRASP utilizados para el diseño del sistema están descritos a continuación.

Controlador: El patrón controlador sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, es utilizado para separar la lógica del negocio de la capa de presentación. En la aplicación se utiliza este patrón, debido a que en su arquitectura de definió una clase controladora para cada módulo; por ejemplo el AdministracionController tiene la responsabilidad de controlar todo el flujo de eventos asociado al gestionar puesto de trabajo.

Controlador Frontal: Se aplica cuando se tiene la creación de un único punto de acceso para todas las peticiones que se realizan en el software. Symfony2 manifiesta este patrón, pues mantiene una estructura organizada, desde los archivos yml hasta las clases php. En el sistema se utiliza este patrón, siguiendo la

Capítulo II: Características y Diseño del Sistema

misma arquitectura del framework, debido a que todas las peticiones de la capa de presentación pasan por el routing.yml y él tiene la responsabilidad de controlar todo el flujo de eventos asociado a estas, y enviarlas a sus respectivas clases Controller.php. Por ejemplo se desea adicionar un puesto de trabajo, esta funcionalidad parte de la interfaz correspondiente y llega al routing, aquí se recibe la petición y es transferida a la administraciónController encargada de ejecutar la petición.

Creador: Identifica quién debe ser el responsable de crear o instanciar nuevos objetos de clases. En el sistema se hace uso de este patrón cuando se asigna la responsabilidad, a la clase AdministracionController.php, de crear instancias de objetos de las clases Usuario, Local, Recurso, entre otras. El código mostrado en la figura 8 evidencia el uso de este patrón en el sistema.

```
Ext.ns('Ext.der');
Ext.der.PanelDerecho = Ext.extend(Ext.Panel,{
    constructor:function(){
        this.panelDerSup = new Ext.derSup.PanelDerSup();
        this.panelDerInf = new Ext.derInf.PanelDerInf();
        Ext.der.PanelDerecho.superclass.constructor.call(this,{
            id:'panelDerecho',
            style:'padding:5px 5px 5px 5px',
            width: 800,
            minWidth: 800,
            maxWidth: 800,
            split: false,
            region:"east",
            layout:"border",
            border:false,
            hideBorders : true,
            items:[this.panelDerSup, this.panelDerInf]
        });
    }
});
```

Figura 8: Código que evidencia el uso del patrón creador.

Experto: Propone asignar a las clases las responsabilidades de acuerdo a la información que contengan. En el sistema se hace uso de este patrón a través de la librería Doctrine2 la cual utiliza Symfony para realizar la capa de abstracción al modelo de datos, encapsulando toda la lógica de los datos y generando las clases con funcionalidades comunes de las entidades. Cada clase creada por Doctrine a partir de una entidad es experta en manejar su información. Por cada tabla se generan 2 clases: Departamento y DepartamentoRepository. Las clases a las que el framework añade el sufijo “Repository” trabajan directamente con la base de datos y por lo tanto son las encargadas de la abstracción. En ellas se encuentran los atributos necesarios para este proceso, de ahí la necesidad de que implementen la responsabilidad de efectuar las operaciones con la base de datos. El código mostrado en la figura 9 evidencia el uso de este patrón en el sistema.

Capítulo II: Características y Diseño del Sistema

```
public function readDepartamentoAction($params = null) {
    try {
        $repository = $this->getDoctrine()->getRepository('GestionEundle:Departamento');
        $start = (int) (isset($params['start']) ? $params['start'] : 0);
        $limit = (int) (isset($params['limit']) ? $params['limit'] : 18);
        $sort = isset($params['sort']) ? $params['sort'] : 'nombre';
        $dir = isset($params['dir']) ? $params['dir'] : 'ASC';
        $query = isset($params['query']) ? $params['query'] : false;
        if ($query) {
            $items = $repository->findByQueryWith($query, $start, $limit, $sort, $dir);
            $totalItems = $repository->countByQuery($query);
        } else {
            $items = $repository->findAllWith($start, $limit, $sort, $dir);
            $totalItems = $repository->countAll();
        }
        return new Response(json_encode(array(
            'success' => true,
            'totalItems' => $totalItems,
            'data' => $items
        )));
    } catch (\Exception $exc) {...}
}
```

Figura 9: Código que evidencia el uso del patrón experto.

Alta Cohesión: Propone que la información almacenada una clase debe de ser coherente y debe tener (en la medida de lo posible) relación con su propósito en el sistema. Symfony2 permite la asignación de responsabilidades con alta cohesión, por ejemplo la clase *NameController.php* tiene la responsabilidad para definir las acciones sobre las plantillas y colabora con otras para realizar diferentes operaciones y crear objetos, está formada por diferentes funcionalidades que se encuentran estrechamente relacionadas, proporcionando que el software sea flexible frente a grandes cambios. Por ejemplo la clase *AdministracionController.php* tiene la responsabilidad de definir las acciones para adicionar puestos de trabajo y esta a su vez colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a los puestos de trabajo.

Bajo Acoplamiento: Propone tener las clases con la menor dependencia que se pueda entre ellas. De tal forma que en caso de producirse una modificación en alguna, se tenga la mínima repercusión posible en el resto de clases, potenciando así la reutilización (35). Las clases que implementan la lógica de negocio y de acceso a datos se encuentran en el modelo, estas clases no tienen asociaciones con las de la vista o el controlador por lo que la dependencia en este caso es baja, cumpliéndose de esta forma el patrón Bajo Acoplamiento. Se evidencia en la capa Modelo, ya que existe poca dependencia entra las clases de acceso a datos y las de abstracción de datos, lo que permite mayor reutilización. En el módulo

Capítulo II: Características y Diseño del Sistema

administración se pueden modificar las clases del modelo sin que se afecten las del controlador, y viceversa ya que estas son acciones independientes.

2.4.4 Diagrama de Interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema. La mayoría de las veces, esto implica modelar instancias concretas o prototípicas de clases, interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. Los diagramas de interacción pueden utilizarse para visualizar, especificar, construir y documentar la dinámica de una sociedad particular de objetos, o se pueden utilizar para modelar un flujo de control particular de un caso de uso. Existen dos tipos de diagramas de interacción: colaboración y secuencia (36). A continuación se pueden ver en las figuras 8, 9 y 10 los diagramas de secuencia del caso de uso Gestionar Puesto de Trabajo.

Diagrama de Secuencia Adicionar Puesto de Trabajo

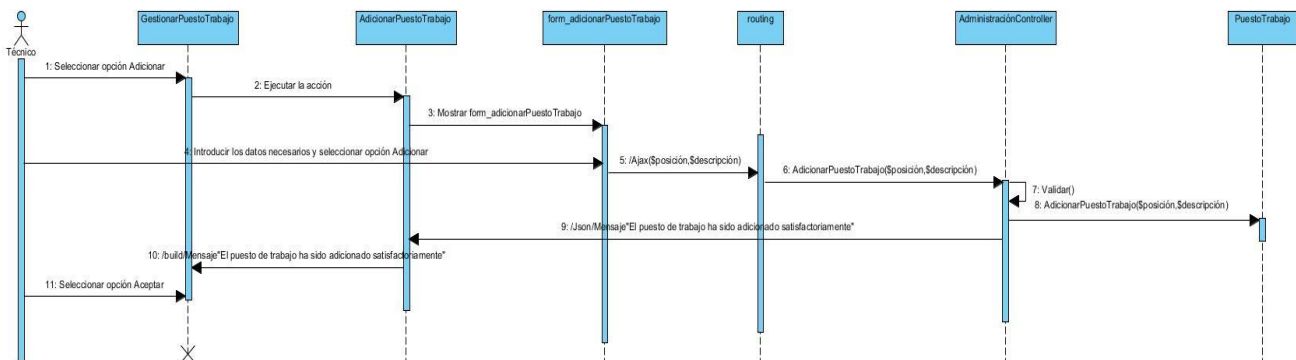
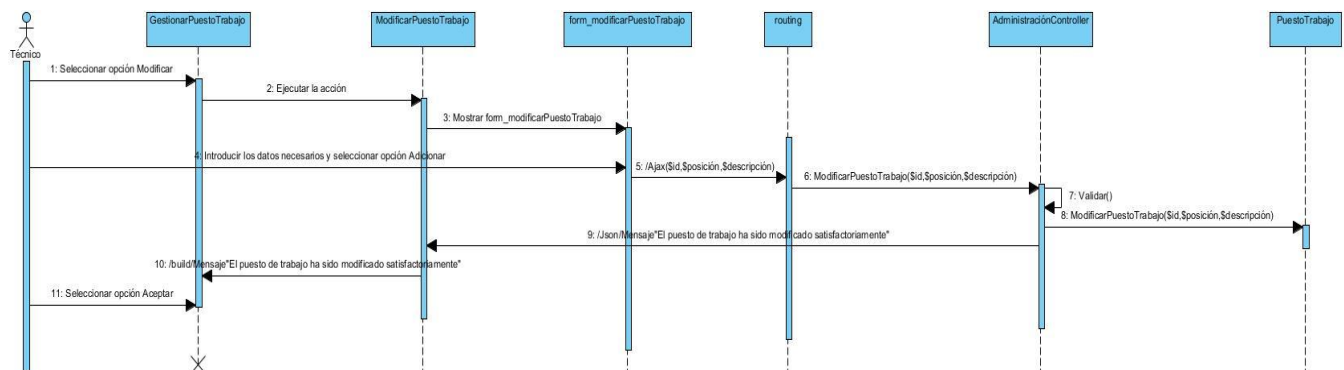


Figura 10: Diagrama de Secuencia Adicionar Puesto de Trabajo

Diagrama de Secuencia Modificar Puesto de Trabajo



Capítulo II: Características y Diseño del Sistema

Descripción de las principales tablas del modelo presentado en la figura 13.

Usuario: Tabla que contendrá los datos de todos los usuarios del sistema.

Incidencia: Almacena la información relacionada con todas las incidencias reportadas en el sistema, y permite realizar la gestión de las mismas. Cada incidencia, está relacionada a la vez con un TipoIncidencia, que contiene los tipos de incidencias, y con un EstadoIncidencia, que contiene los estados por los cuales puede transitar cada una de estas incidencias. Estas incidencias están relacionadas con Usuario y Recurso, permitiendo conocer si un usuario ha reportado una Incidencia sobre un recurso determinado.

SolicitudDescarga: Almacena la información relacionada con todas las solicitudes de descargas reportadas en el sistema, y permite realizar la gestión de las mismas. Cada SolicitudDescarga, está relacionada a la vez con un EstadoSolicitudDescarga, que contiene los estados por los cuales puede transitar cada una de estas solicitudes. Estas solicitudes están relacionadas con Usuario y Departamento, permitiendo conocer si un usuario ha reportado una solicitud de descarga y al departamento que pertenece el mismo.

Departamento: Almacena la información relacionada con todos los departamentos del sistema, y permite realizar la gestión de los mismos.

Proyecto: Almacena la información relacionada con todos los proyectos del sistema, y permite realizar la gestión de los mismos.

Local: Almacena la información relacionada con todos los locales del sistema, y permite realizar la gestión de los mismos.

Recurso: Almacena la información relacionada con todos los recursos del sistema, y permite realizar la gestión de los mismos.

Traza: Almacena la información relacionada con todas las trazas del sistema. La misma está relacionada con Usuario, permitiendo conocer el autor de cada una de las acciones que se realizan sobre el sistema.

Usuario: Almacena la información relacionada con todos los usuarios del sistema, y permite realizar la gestión de los mismos. Estos a su vez están relacionados a Rol, ya que cada usuario está restringido a los permisos del Rol al cual pertenece.

PuestoTrabajo: Almacena la información relacionada con todos los puestos de trabajo del sistema, y permite realizar la gestión de los mismos.

Capítulo II: Características y Diseño del Sistema

RecursosAdicionales: Almacena la información relacionada con todos los Recursos adicionales del sistema, y permite realizar la gestión de los mismos.

Conclusiones del capítulo

En el presente capítulo, además de hacerse una breve descripción de las características del sistema a implementar, fueron abordados algunos de los artefactos propuestos por la metodología OpenUP para el desarrollo del software. Uno de ellos es el modelo de dominio, con el cual se describe el proceso que guiará el funcionamiento de la aplicación. Con la especificación de los requisitos funcionales y no funcionales se obtiene una orientación para la creación del producto y una visión de los patrones de diseños GRASP que se seguirán para una correcta implementación del sistema. La realización de los diagramas de las clases del diseño muestra la estructura interna del producto y da una medida de los métodos que se deben desarrollar y donde deben ser ubicados. A su vez se realiza una descripción de los procesos implícitos en la aplicación a través de los diagramas de secuencias y se obtiene la estructura de la base de datos a partir de la confección del diagrama de entidad relación.

Capítulo III: Implementación y Pruebas del Sistema

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

Introducción

Una vez concluido el modelo del diseño, se dispone de los detalles suficientes para proceder a la construcción del sistema, y una vez concluido este, se procede a la verificación del cumplimiento de los requisitos funcionales mediante las pruebas de software. En el presente capítulo se describe cómo los elementos del modelo del diseño se implementan en términos de componentes y se muestra una descripción detallada de los paquetes de implementación. Se presenta el mapa de navegación del software desarrollado y algunas imágenes del mismo para brindar una mejor comprensión del sistema. Posteriormente se pasa a la validación de la aplicación mediante las pruebas de software de caja negra, carga y stress, para comprobar la operatividad de las principales funcionalidades. Y finalmente se muestra el diagrama de despliegue, con una breve descripción de los nodos que lo conforman.

3.1 Diagrama de componentes

Es uno de los diagramas del Lenguaje Unificado de Modelado. Representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Este tipo de diagrama prevalece en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema (38).

Diagrama de componentes del caso de uso Gestionar Puesto de Trabajo

Capítulo III: Implementación y Pruebas del Sistema

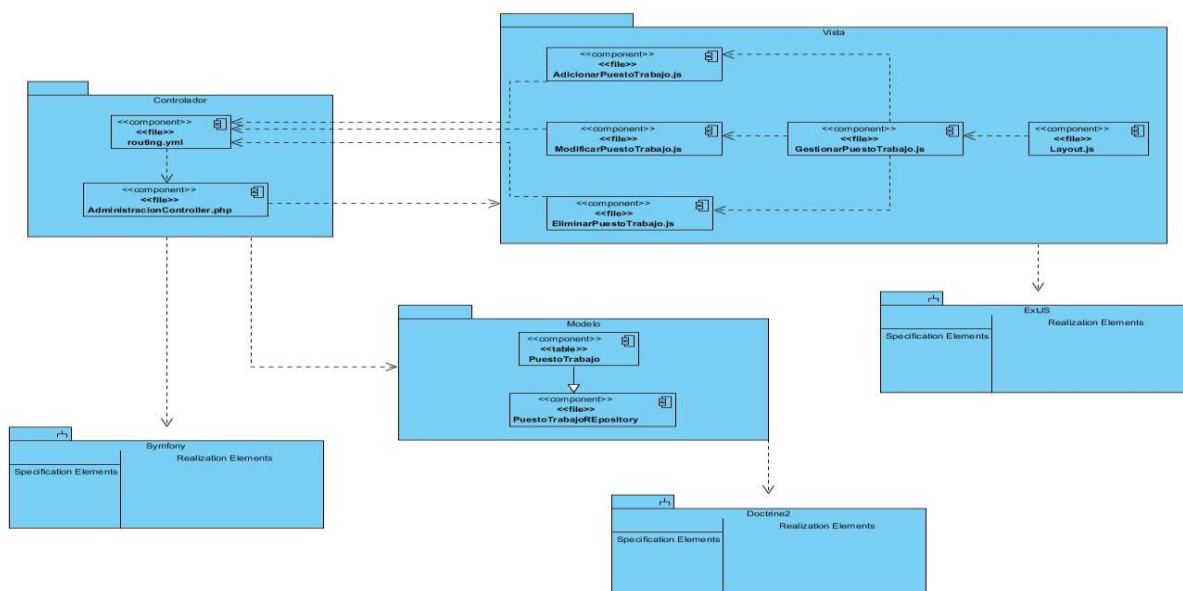


Figura 14: Diagrama de componentes del caso de uso Gestionar Puesto de Trabajo.

- ✓ **Vista:** contiene todas las interfaces de usuario asociadas al caso de uso Gestionar recurso, convertidas en componentes `<<file>>`. La Vista tiene relación de dependencia con el `<<subsystem>>` ExtJS.
- ✓ **Controlador:** contiene todas las acciones asociadas al caso de uso Gestionar puesto de trabajo, para darle respuesta a las peticiones del usuario que llegan a través de routing.yml, convertidas en componentes `<<file>>`. El Controlador tiene relación de dependencia con el `<<subsystem>>` Symfony.
- ✓ **Modelo:** contiene todas las tablas de la base de datos asociadas al caso de uso Gestionar recurso, convertidas en componentes `<<table>>`. El Modelo tiene relación de dependencia con el `<<subsystem>>` Doctrine2.
- ✓ **ExtJS:** paquete que se utiliza en la capa Vista para el diseño de las interfaces, convertido en un componente de tipo `<<subsystem>>`.
- ✓ **Symfony:** paquete que se utiliza en la capa Controlador para la implementación de las acciones, convertido en un componente de tipo `<<subsystem>>`.
- ✓ **Doctrine2:** paquete que se utiliza en la capa Modelo para convertir las tablas de la base de datos en clases, convertido en un componente de tipo `<<subsystem>>`.

Capítulo III: Implementación y Pruebas del Sistema

3.2 Mapa de navegación del sistema

Es el tratamiento comunicacional de los contenidos, en otras palabras, es la organización en la presentación de la información, expresada en un diagrama denominado mapa de navegación. La importancia de elaborar un mapa de navegación del sitio web radica en la comprensión del orden de presentación de las pantallas con los contenidos (páginas web) y la flexibilidad de moverse entre ellas (hipervínculos) (39).

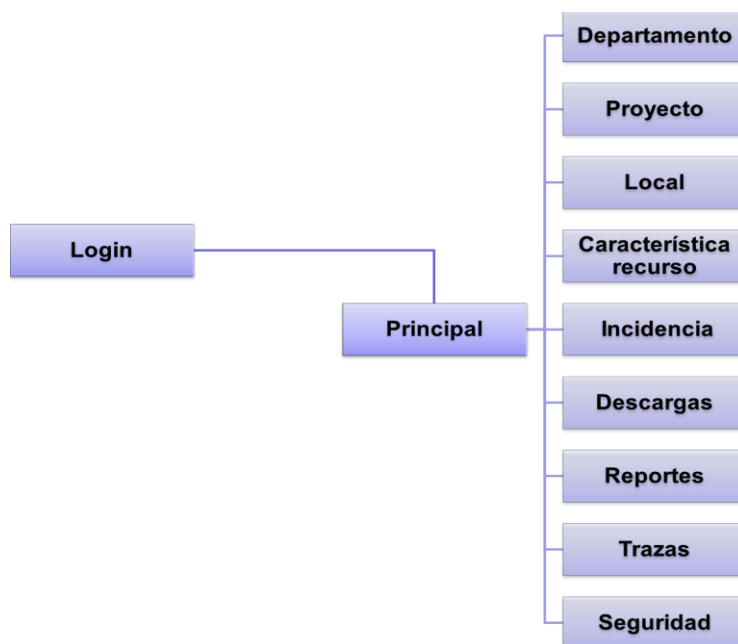


Figura 15: Mapa de navegación del sistema donde se muestra el primer nivel de opciones.

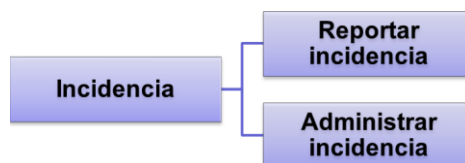


Figura 16: Fragmento del mapa de navegación a partir de la opción Incidencia.

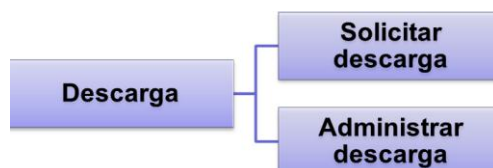


Figura 17: Fragmento del mapa de navegación a partir de la opción Descarga.

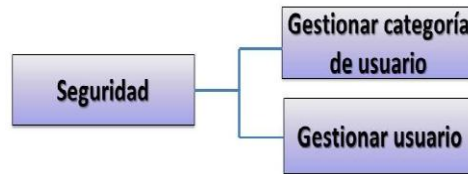


Figura 18: Fragmento del mapa de navegación a partir de la opción Seguridad.

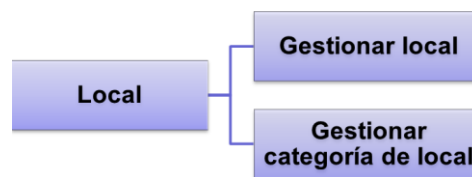


Figura 19: Fragmento del mapa de navegación a partir de la opción Local.

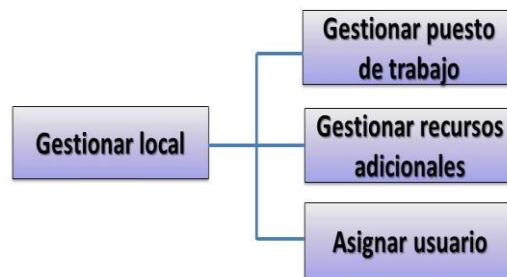


Figura 20: Fragmento del mapa de navegación a partir de la opción Gestionar local.

3.2.1 Imágenes del sistema

A continuación serán presentadas imágenes de algunas de las interfaces del sistema para una mejor comprensión de su funcionamiento.

Formulario de autenticación de usuario con los siguientes elementos:

- Etiqueta: Usuario
- Campo de entrada: Usuario
- Etiqueta: Contraseña
- Campo de entrada: Contraseña
- Botón: Entrar

Figura 21: Autenticar usuario.

Capítulo III: Implementación y Pruebas del Sistema

La figura 21 muestra la estructura gráfica utilizada para autenticar a los usuarios que deseen acceder a los servicios ofrecidos en el sistema. Contiene dos campos no opcionales para entrada de datos, correspondientes al usuario y la contraseña del actor que requiere entrar en la aplicación.

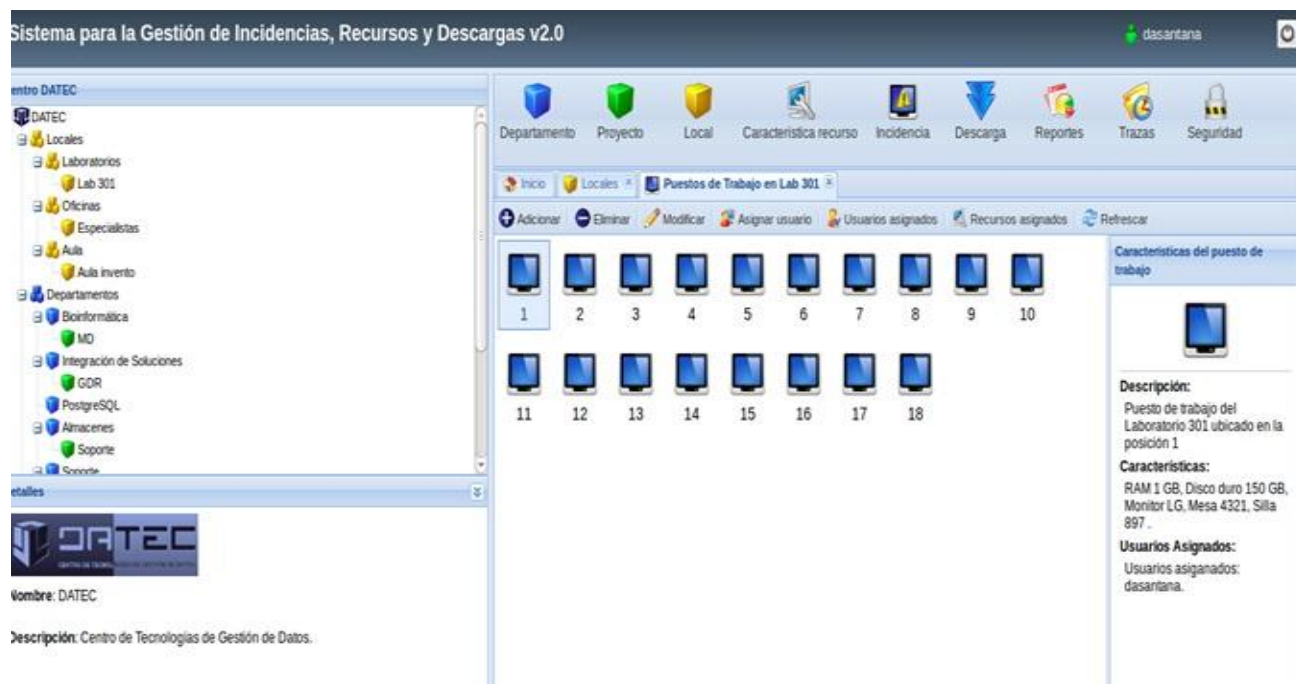


Figura 22: Imagen de la interfaz asociada al caso de uso Gestionar puesto de trabajo.

La figura 22 muestra la capa de presentación correspondiente a la funcionalidad gestionar puestos de trabajo de un local determinado. En ella se evidencian las opciones de adicionar, eliminar y modificar puestos de trabajo. De igual manera se proporcionan otras operaciones sobre ellos, tales como asignar usuarios, ver usuarios asignados y gestionar recursos.

Capítulo III: Implementación y Pruebas del Sistema

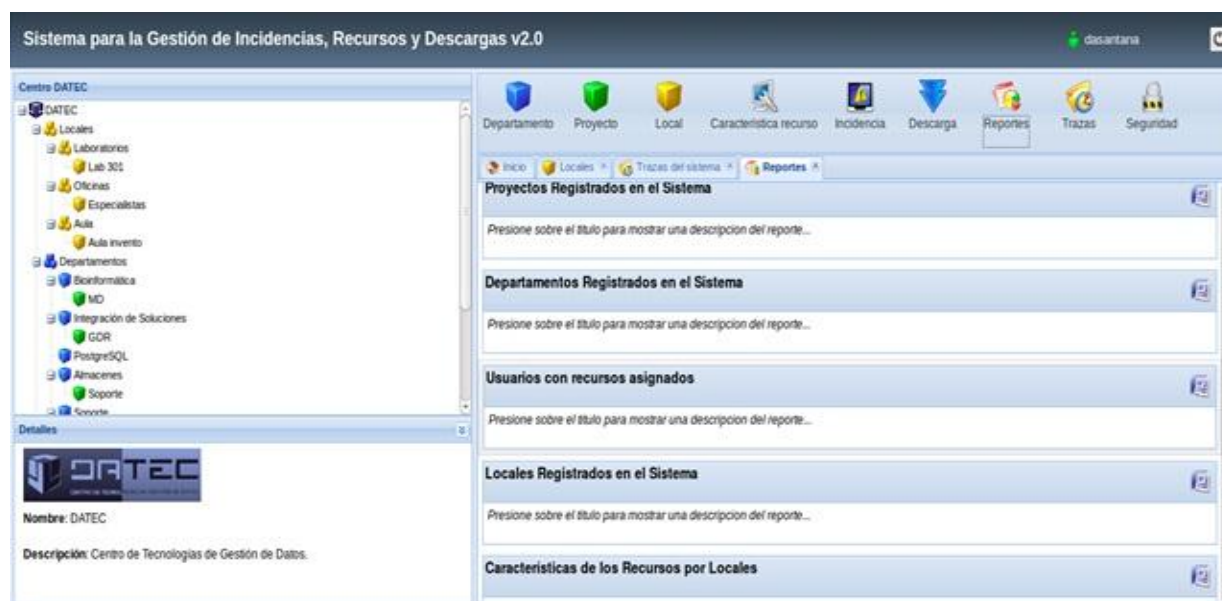


Figura 23: Imagen de la interfaz asociada al caso de uso Visualizar reporte.

La figura 23 representa la interfaz correspondiente a la visualización y generación de reportes. En la cual se ofrece toda una gama de datos, posibilitando obtener una información determinada sin necesidad de una navegación excesiva en la aplicación.

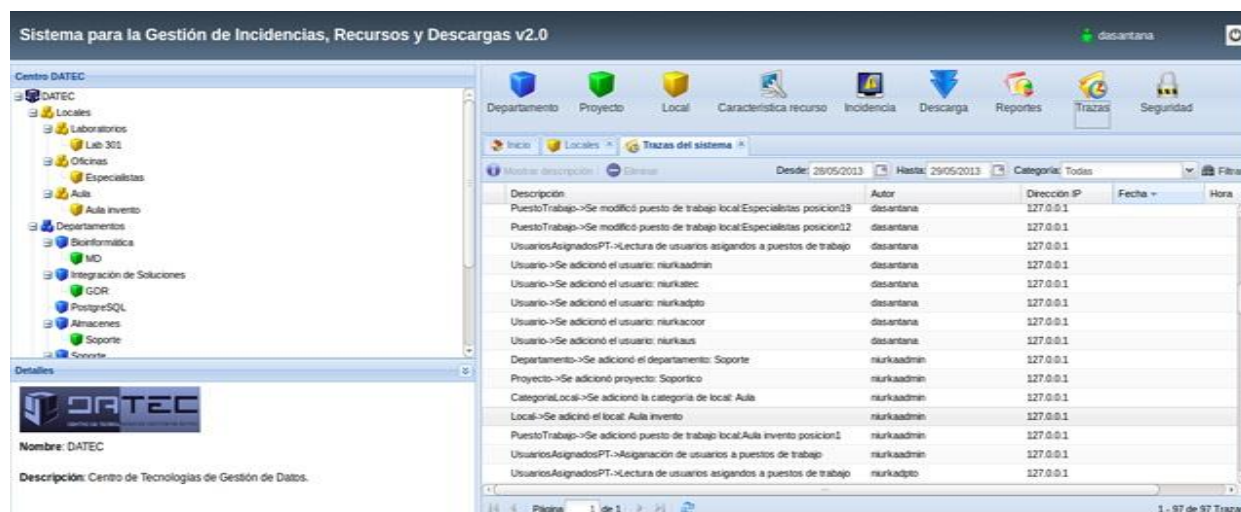


Figura 24: Imagen de la interfaz asociada al caso de uso Administrar trazas.

La figura 24 muestra la estructura gráfica en la cual se realizan las operaciones correspondientes a la administración de trazas en el sistema. En esta interfaz se dan las opciones de mostrar descripción,

Capítulo III: Implementación y Pruebas del Sistema

eliminar, filtrar por categorías y por fechas. Esta sección es utilizada para llevar el control de las operaciones que hacen los usuarios sobre la base de datos correspondiente a la aplicación.

3.3 Validación del sistema

La calidad de un producto de software es el indicador que permite determinar si los procesos de construcción de software fueron apropiados, por lo que deben utilizarse métodos y técnicas que garanticen la misma. Una forma de medir la calidad de un software es la realización de las diferentes pruebas que existen, las cuales tienen en cuenta factores como: la usabilidad, la eficiencia y la escalabilidad (40). Una vez generado el código fuente, el software debe ser probado para descubrir y corregir el máximo de errores posibles antes de su liberación. El objetivo es diseñar una serie de pruebas que tengan una alta probabilidad de encontrar errores.

Pruebas de software

Un concepto importante a tomar en consideración es el emitido por Roger S. Pressman, un norteamericano ingeniero de software, en el libro de Ingeniería del Software en su edición de 1998, que plantea lo siguiente:

“La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión de las especificaciones, del diseño y de la codificación”.

Las pruebas de software permiten verificar y revelar la calidad de un software. Son utilizadas para identificar posibles fallos durante el proceso de desarrollo. Los casos de prueba son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos específicos. Los resultados son observados y registrados, realizando una evaluación de algún aspecto o componente del sistema. Permiten encontrar y documentar los defectos que puedan afectar la calidad del software. Verifican que el software trabaje como fue diseñado. Validan y prueban cada uno de los requisitos que debe cumplir el software y determina si estos fueron implementados correctamente (40).

Para validar el sistema se realizaron pruebas a nivel de desarrollador, utilizando las del tipo funcional y aplicando el método de caja negra. De igual manera se llevaron a cabo pruebas de carga y stress.

3.3.1 Pruebas de caja negra

Las pruebas de caja negra, denominadas pruebas de comportamientos, se centran en los requisitos funcionales del software. Debido a que ellas permiten obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.

Capítulo III: Implementación y Pruebas del Sistema

La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien, se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los que identifican los métodos de caja blanca (41). La prueba de caja negra intenta identificar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación. Existen varios métodos de caja negra entre los que se pueden mencionar los métodos de prueba basados en grafo, partición equivalente y análisis de valores al límite.

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico (41). La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

- ✓ Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
- ✓ Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos no válidas.
- ✓ Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida.
- ✓ Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida.

Resultados de las pruebas de caja negra

	No correspondencia	Ortografía	Funcionalidad
Prueba exploratoria	34	12	7
1ra Iteración	26	6	4
2da Iteración	12	3	1
3ra Iteración	0	0	0

Tabla 3: Resultados de la liberación del sistema.

Capítulo III: Implementación y Pruebas del Sistema

3.3.2 Pruebas de carga

Son utilizadas para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema bajo prueba permanece constante. La variación en carga es simular la carga de trabajo promedio y con picos que ocurre dentro de tolerancias operacionales normales. El sistema fue probado con la herramienta Apache JMeter 2.3.4, la cual arrojó los resultados mostrados en la figura 25.

Muestra #	Start Time	Thread Name	Label	Tiempo de Muestra (ms)	Status	Bytes
1	16:26:48.222	Grupo de Hilos 1-1	/Sopart-2.0/Symfony/w...	377		823
2	16:26:48.429	Grupo de Hilos 1-4	/Sopart-2.0/Symfony/w...	442		823
3	16:26:48.220	Grupo de Hilos 1-2	/Sopart-2.0/Symfony/w...	514		823
4	16:26:48.580	Grupo de Hilos 1-6	/Sopart-2.0/Symfony/w...	322		823
5	16:26:48.427	Grupo de Hilos 1-3	/Sopart-2.0/Symfony/w...	546		823
6	16:26:48.778	Grupo de Hilos 1-7	/Sopart-2.0/Symfony/w...	692		823
7	16:26:49.471	Grupo de Hilos 1-7	/Sopart-2.0/Symfony/w...	73		925
8	16:26:49.545	Grupo de Hilos 1-7	/Sopart-2.0/Symfony/w...	26		819
9	16:26:49.572	Grupo de Hilos 1-7	/Sopart-2.0/Symfony/w...	25		836
10	16:26:49.598	Grupo de Hilos 1-7	/Sopart-2.0/Symfony/w...	26		577
11	16:26:49.322	Grupo de Hilos 1-4	/Sopart-2.0/Symfony/w...	424		925
12	16:26:49.321	Grupo de Hilos 1-10	/Sopart-2.0/Symfony/w...	426		823
13	16:26:49.317	Grupo de Hilos 1-9	/Sopart-2.0/Symfony/w...	431		823
14	16:26:48.487	Grupo de Hilos 1-5	/Sopart-2.0/Symfony/w...	1261		823
15	16:26:48.604	Grupo de Hilos 1-1	/Sopart-2.0/Symfony/w...	1042		925
16	16:26:48.781	Grupo de Hilos 1-8	/Sopart-2.0/Symfony/w...	1041		823
17	16:26:49.862	Grupo de Hilos 1-1	/Sopart-2.0/Symfony/w...	65		819
18	16:26:49.327	Grupo de Hilos 1-11	/Sopart-2.0/Symfony/w...	894		823
19	16:26:50.010	Grupo de Hilos 1-14	/Sopart-2.0/Symfony/w...	212		823
20	16:26:49.864	Grupo de Hilos 1-9	/Sopart-2.0/Symfony/w...	360		925
21	16:26:49.863	Grupo de Hilos 1-5	/Sopart-2.0/Symfony/w...	362		925
22	16:26:49.861	Grupo de Hilos 1-8	/Sopart-2.0/Symfony/w...	366		925
23	16:26:49.860	Grupo de Hilos 1-4	/Sopart-2.0/Symfony/w...	368		819
24	16:26:50.229	Grupo de Hilos 1-4	/Sopart-2.0/Symfony/w...	258		836
25	16:26:50.228	Grupo de Hilos 1-8	/Sopart-2.0/Symfony/w...	259		819
26	16:26:50.226	Grupo de Hilos 1-5	/Sopart-2.0/Symfony/w...	261		819
27	16:26:50.224	Grupo de Hilos 1-9	/Sopart-2.0/Symfony/w...	262		819
28	16:26:50.223	Grupo de Hilos 1-14	/Sopart-2.0/Symfony/w...	263		925
29	16:26:50.222	Grupo de Hilos 1-11	/Sopart-2.0/Symfony/w...	263		925
30	16:26:49.324	Grupo de Hilos 1-12	/Sopart-2.0/Symfony/w...	1161		823
31	16:26:50.535	Grupo de Hilos 1-8	/Sopart-2.0/Symfony/w...	159		836

No. de Muestras 100 Última Muestra 2877 Media 802

Figura 25: Resultados de la prueba de carga.

La figura 25 muestra los resultados obtenidos en la prueba de carga. Fue realizada con una simulación de 100 usuarios haciendo peticiones al sistema concurrentemente. Se pueden apreciar datos como el tiempo de respuesta en milisegundos y los errores en la columna (Status) para cada petición y la media del tiempo de respuesta que no excede los 10 segundos como se definió en el requisito no funcional de eficiencia.

3.3.3 Pruebas de stress

Es enfocada a evaluar cómo el sistema responde bajo condiciones anormales (extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible). Este tipo de prueba se realiza para determinar la solidez de un sistema en los momentos de carga extrema y ayuda a los administradores para

Capítulo III: Implementación y Pruebas del Sistema

determinar el rendimiento en caso de que la carga real supere a la carga esperada. Tras aplicar una prueba de stress con la herramienta Apache JMeter 2.3.4 simulando el uso del sistema con el doble de los usuarios respecto a la prueba de carga, el sistema arrojó los resultados mostrados en la figura 26.

Ver Resultados en Árbol

Nombre: Ver Resultados en Árbol

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo Escribir en Log Sólo Errores

Muestra #	Start Time	Thread Name	Label	Tiempo de Muestra (ms)	Status	Bytes
1	16:48:50.594	Grupo de Hilos 1-6	/Soport-2.0/Symfony/w...	2989		1896
2	16:48:50.415	Grupo de Hilos 1-4	/Soport-2.0/Symfony/w...	3810		1896
3	16:48:51.182	Grupo de Hilos 1-7	/Soport-2.0/Symfony/w...	3286		1896
4	16:48:51.164	Grupo de Hilos 1-8	/Soport-2.0/Symfony/w...	4199		1896
5	16:48:51.170	Grupo de Hilos 1-9	/Soport-2.0/Symfony/w...	4198		1896
6	16:48:51.614	Grupo de Hilos 1-10	/Soport-2.0/Symfony/w...	3760		1896
7	16:48:50.518	Grupo de Hilos 1-5	/Soport-2.0/Symfony/w...	3711		1896
8	16:48:50.316	Grupo de Hilos 1-1	/Soport-2.0/Symfony/w...	3915		1896
9	16:48:50.411	Grupo de Hilos 1-2	/Soport-2.0/Symfony/w...	4062		1896
10	16:48:50.413	Grupo de Hilos 1-3	/Soport-2.0/Symfony/w...	4059		1896
11	16:48:52.258	Grupo de Hilos 1-11	/Soport-2.0/Symfony/w...	5684		1896
12	16:48:52.261	Grupo de Hilos 1-13	/Soport-2.0/Symfony/w...	5681		1896
13	16:48:52.256	Grupo de Hilos 1-16	/Soport-2.0/Symfony/w...	37853		1896
14	16:48:52.252	Grupo de Hilos 1-12	/Soport-2.0/Symfony/w...	37851		1896
15	16:48:54.226	Grupo de Hilos 1-4	/Soport-2.0/Symfony/w...	46052		1896
16	16:48:55.115	Grupo de Hilos 1-26	/Soport-2.0/Symfony/w...	45950		1896
17	16:48:55.110	Grupo de Hilos 1-25	/Soport-2.0/Symfony/w...	45959		1896
18	16:48:55.118	Grupo de Hilos 1-7	/Soport-2.0/Symfony/w...	47755		1896
19	16:48:58.456	Grupo de Hilos 1-13	/Soport-2.0/Symfony/w...	45590		1896
20	16:48:57.953	Grupo de Hilos 1-1	/Soport-2.0/Symfony/w...	46115		1896
21	16:49:40.280	Grupo de Hilos 1-4	/Soport-2.0/Symfony/w...	4492		1896
22	16:48:58.939	Grupo de Hilos 1-28	/Soport-2.0/Symfony/w...	45878		1896
23	16:49:34.627	Grupo de Hilos 1-16	/Soport-2.0/Symfony/w...	10260		1896
24	16:49:44.070	Grupo de Hilos 1-1	/Soport-2.0/Symfony/w...	1608		1896
25	16:49:41.070	Grupo de Hilos 1-25	/Soport-2.0/Symfony/w...	4831		1896
26	16:48:58.453	Grupo de Hilos 1-9	/Soport-2.0/Symfony/w...	48150		1896
27	16:49:41.067	Grupo de Hilos 1-26	/Soport-2.0/Symfony/w...	5573		1896
28	16:49:42.874	Grupo de Hilos 1-7	/Soport-2.0/Symfony/w...	3786		1896
29	16:49:40.292	Grupo de Hilos 1-12	/Soport-2.0/Symfony/w...	6515		1896
30	16:49:44.818	Grupo de Hilos 1-28	/Soport-2.0/Symfony/w...	2598		1896
31	16:49:44.773	Grupo de Hilos 1-4	/Soport-2.0/Symfony/w...	2663		1896

No. de Muestras 250 Última Muestra 3005 Media 2451

Figura 26: Resultados de la prueba de stress.

La figura 26 muestra los resultados obtenidos en la prueba de stress. Fue realizada con una simulación de 250 usuarios haciendo peticiones al sistema al mismo tiempo. Se pueden apreciar datos como el tiempo de respuesta en milisegundos y los errores en la columna (Status) para cada petición y la media del tiempo de respuesta que no excede los 25 segundos como se definió en el requisito no funcional de eficiencia.

3.4 Diagrama de Despliegue

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación, donde un nodo puede contener instancias de componentes. En general, un nodo se entiende como una unidad de computación de algún tipo como es el caso de impresoras o la misma computadora (42).

Capítulo III: Implementación y Pruebas del Sistema

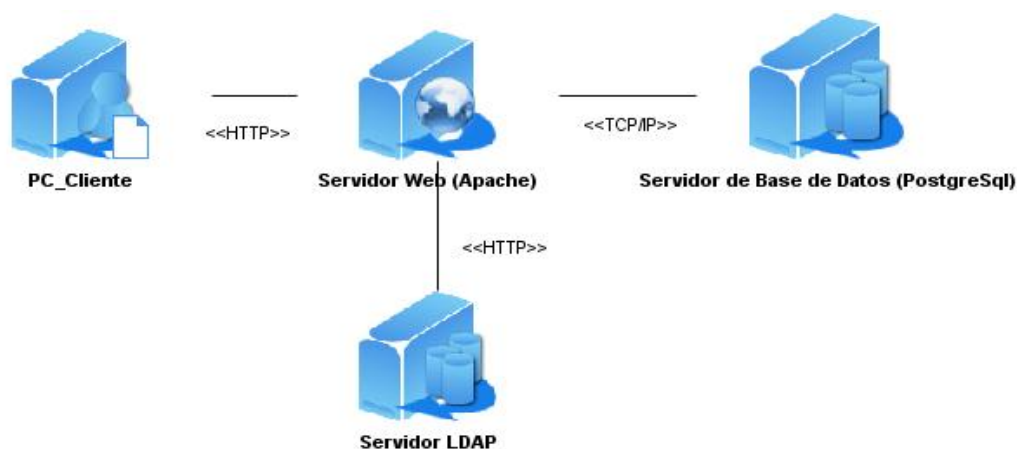


Figura 27: Modelo de despliegue.

Descripción de los nodos

Nodo PC_Cliente: Ordenador desde el cual accedería el cliente al sistema.

Nodo Servidor Web: Ordenador donde se ejecutará el sistema.

Nodo Servidor de Base de Datos: Ordenador donde se almacenará la información del sistema.

Nodo Servidor LDAP: Base de datos donde se encuentran todos los usuarios de la universidad.

HTTP: Protocolo de transferencia utilizado para conectar la computadora del cliente con el servidor donde está el sistema y este último con el servidor de los usuarios de la universidad (LDAP).

TCP/IP: Protocolo para conectar el servidor de aplicaciones con las bases de datos.

Conclusiones del capítulo

En este capítulo se obtuvieron los diagramas de componentes, y a partiendo de ellos se mostró la organización y las dependencias entre los componentes que conforman el sistema. Se realizó un mapa de navegación para una mejor comprensión de la navegabilidad dentro del mismo. Se obtuvo la implementación de la aplicación, dando solución a los requisitos funcionales identificados y se definió el diagrama de despliegue mostrando la disposición física del software. Para la validación del sistema se realizaron pruebas de caja negra, stress y carga que evaluaron y comprobaron la operatividad de las funcionalidades.

CONCLUSIONES

Una vez concluida la investigación se puede afirmar que se desarrolló la segunda versión del Sistema para gestión de información de incidencias, descargas y recursos del Centro de Tecnología de Gestión de Datos. Para ello:

- ✓ Se hizo el estudio de las aplicaciones existentes en el mundo así como el de la primera versión, que garantizó comprender la estructura de estos sistemas y sus principales características permitiendo dar inicio al diseño e implementación del sistema. Se seleccionó la metodología, herramientas y tecnologías a utilizar en el desarrollo de esta nueva aplicación posibilitando crear un software fácil de utilizar en cualquier ambiente de trabajo.
- ✓ Se definieron las características del sistema, identificando 6 requisitos no funcionales y 51 requisitos funcionales, agrupados en 20 casos de uso, reconociendo a 4 como arquitectónicamente significativos: Gestionar Usuario, Gestionar Puesto de Trabajo, Gestionar Recurso y Gestionar Usuarios del Puesto de Trabajo. La arquitectura se confeccionó siguiendo el patrón arquitectónico MVC y los patrones de diseño GRASP. Los mismos fueron aplicados en la realización de los diagramas de interacción y del modelo del diseño, dando como resultado un sistema separado en capas, altamente escalable y menos vulnerable al cambio.
- ✓ Se implementaron los requisitos funcionales definidos, obteniendo un sistema capaz de satisfacer las necesidades actuales del grupo de Soporte del Centro de Tecnología de Gestión de Datos. Brindando una aplicación que mantiene los logros de la primera versión e incorpora nuevos cambios estructurales necesarios para una gestión más amplia de información de los recursos en el centro, así como a su vez presenta una política de seguridad más robusta.
- ✓ Se validaron las funciones implementadas a través de la interfaz del software mediante las pruebas de caja negra de partición de equivalencia, las de carga y stress. Ellas permitieron encontrar un conjunto de inconformidades en la validación del producto y el tratamiento de las mismas contribuyeron a aumentar la calidad del sistema.

RECOMENDACIONES

Se recomienda para futuras versiones:

- ✓ Incorporar un módulo de auditorías para el control integral de cada uno de los procesos implementados en el sistema.
- ✓ Realizar una investigación para actualizar en el sistema desarrollado, las prestaciones tecnológicas de las computadoras en tiempo real.

Referencias Bibliográficas

REFERENCIAS

1. Curto, Josep. Information Management. *Gestión de la información, Información, Teoría*. [En línea] 28 de 11 de 2006. [Citado el: 12 de noviembre de 2012.] <http://informationmanagement.wordpress.com/category/gestion/gestion-de-la-informacion/>.
2. antiguo.itson.mx. [En línea] [Citado el: 14 de enero de 2013.] <http://antiguo.itson.mx/dii/jgaxiola/sistemas/transaccionales.html>.
3. indracompany.com. [En línea] [Citado el: 5 de enero de 2013.] www.indracompany.com/sites/default/.../isocloud_baja_2012_0.pdf.
4. optimafar.com. [En línea] [Citado el: 21 de noviembre de 2012.] http://www.optimafar.com/optimafar/gip/gip_main.jsp.
5. Blanco, Kenia Riverón Ovalle y Yaidel Rodríguez. IMPLEMENTACIÓN DEL MÓDULO DESPACHO DEL SUBSISTEMA INVENTARIO DEL SISTEMA INTEGRAL DE GESTIÓN CEDRUX.
6. Alexander Delgado Gutierrez, Arcides Acis Carballo. Sistema para la gestión de información de incidencias, descargas y recursos del Centro de Tecnologías de Gestión de Datos.
7. ecured. [En línea] [Citado el: 18 de noviembre de 2012.] http://www.ecured.cu/index.php/Metodologias_de_desarrollo_de_Software.
8. Flores, Carmina Lizeth Torres. Establecimiento de una Metodología de Desarrollo de Software. [aut. libro] Germán Harvey Alférez Salinas. 2008.
9. [En línea] [Citado el: 27 de noviembre de 2012.] <http://www.conml.org/FAQ.aspx>.
10. ecured.cu. [En línea] [Citado el: 25 de diciembre de 2012.] (<http://www.ecured.cu/index.php/UML>).
11. ecured.cu. [En línea] [Citado el: diciembre de 21 de 2012.] http://www.ecured.cu/index.php/Herramienta_CASE.
12. [En línea] [Citado el: 18 de noviembre de 2012.] (http://www.ecured.cu/index.php?title=Visual_Paradigm&oldid=1672871 5).
13. [En línea] [Citado el: 1 de noviembre de 2012.] <http://es.kioskea.net/contents/langages/langages.php3>.
14. [En línea] [Citado el: 29 de noviembre de 2012.] <ftp://ftp.prod.uci.cu/PHP/Documentacion>.
15. librosweb. [En línea] [Citado el: 23 de diciembre de 2012.] <http://www.librosweb.es/javascript/capitulo1.html>.
16. CODEBOX. [En línea] [Citado el: 11 de noviembre de 2012.] www.codebox.es/glosario.

Referencias Bibliográficas

17. Symfony. [En línea] [Citado el: 1 de diciembre de 2012.] <http://www.symfony.es/2009/03/06/asi-seran-las-novedades-de-symfony-20/>.
18. Robert., Armando. *Tesis Arquitectura de Software*.
19. [En línea] [Citado el: 5 de noviembre de 2012.] <http://www.doctrine-project.org/>.
20. Jaspersoft. [En línea] [Citado el: 5 de noviembre de 2012.] <http://www.jaspersoft.com/es/node/38063>.
21. ecured.cu. [En línea] [Citado el: 5 de noviembre de 2012] http://www.ecured.cu/index.php/Servidores_Web#.C2.BFQu.C3.A9_es_un_Servidor_Web.3F.
22. Tomcat. [En línea] <http://tomcat.apache.org/index.html>.
23. [En línea] [Citado el: 7 de noviembre de 2012.] <http://www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/sgbd.html>.
24. [En línea] [Citado el: 23 de diciembre de 2012.] <http://www.postgresql.org/about/news/1415>.
25. Parra, Eduardo. [En línea] [Citado el: 13 de noviembre de 2012.] <http://www.portalubuntu.com/2011/04/instalar-netbeans-70-en-espanol-en.html>.
26. [En línea] [Citado el: 13 de noviembre de 2012] http://www.ecured.cu/index.php/Modelo_de_dominio.
27. Sommerville, Ian. Ingeniería del Software. 7ma Edición. 2011. pág. 108. ISBN: 8478290745.
28. Sommerville, Ian. Ingeniería del Software. 7ma Edición. 2011. pág. 110. ISBN: 8478290745.
29. Sommerville, Ian. Ingeniería del Software. 7ma Edición. 2011. pág. 112. ISBN: 8478290745.
30. [En línea] [Citado el: 25 de enero del 2013] http://www.ctr.unican.es/ asignaturas/MC_OO/Doc.
31. [En línea] [Citado el: 27 de enero del 2013] http://www.ecured.cu/index.php/Flujo_de_Trabajo_Analisis_y_Diseño.
32. MSDN. [En línea] [Citado el: 25 de enero de 2013.] <http://msdn.microsoft.com/en-us/library/ff649643.aspx>.
33. Mühlrad, Daniel. Patrones de diseño. [En línea] 2008. [Citado el: 15 de febrero de 2013].
34. Larman, Craig. Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design and Iterative Development. s.l. : Prentice Hall, 2005. ISBN 0-13-148906-2.
35. PATRONES DE DISEÑO Y FRAMEWORKS. [En línea] [Citado el: 25 de febrero de 2013] <http://ingenieriasw2.blogspot.com/p/patrones-de-diseno-y-frameworks.html>.
36. [En línea] [Citado el: 25 de febrero de 2013] <http://users.dcc.uchile.cl/~psalinas/uml/interaccion.html>
37. Ivar Jacobson, Grady Booch, James Rumbaugh. El proceso unificado de desarrollo de software. Vol. 1, Consultado: 14 de febrero de 2013, Disponible en: <http://bibliodoc.uci.cu/pdf/reg00060.pdf>

Referencias Bibliográficas

38. Sparxsystems. [En línea] [Citado el: 5 de abril de 2013.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.
39. <http://portal.perueduca.edu.pe>. [En línea] [Citado el: 9 de mayo de 2013.] http://portal.perueduca.edu.pe/modulos/m_taller/mapa.htm.
40. Julio César Brito Rodríguez. Módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0.
41. Pressman, Roger S.. *Ingeniería del software. Un enfoque práctico*. Lugar de publicación desconocido. Mc Graw Hill. 1998. 614.
42. Slideshare. [En línea] [Citado el: 9 de mayo de 2013.] <http://www.slideshare.net/joshell/diagramas-uml-componentes-y-despliegue>.

BIBLIOGRAFÍA

- ✓ Bsigroup. Bsigroup. [En línea] [Citado el: 10 de Noviembre de 2012.]
- ✓ <http://www.bsigroup.com.mx/es-mx/Auditoria-y-Certificacion/Sistemas-de-Gestion/De-un-vistazo/Que-son-los-sistemas-de-gestion/>.
- ✓ Ciberaula. [En línea] [Citado el: 12 de Noviembre de 2012.]
- ✓ http://linux.ciberaula.com/articulo/linux_apache_intro/.
- ✓ Díaz Laurencio, Elennis y Nieto Cervantes, Liusmila. *LIMS DE CALIDAD DEL CENTRO DE INGENIERÍA GENÉTICA Y BIOTECNOLOGÍA: ANÁLISIS DE LA SECCIÓN DE MEJORAMIENTO DE LA CALIDAD Y DEL GRUPO DE DESARROLLO*. Ciudad de la Habana: s.n., 2007.
- ✓ Eric Freeman, Elisabeth Freeman. *Head First Design Patterns*.
- ✓ Fowler, Martin. *Pattern of Enterprises Architecture*.
- ✓ Garcilaso Jordana. Introducción OpenUP. [En línea] [Citado el: 15 de Noviembre de 2012.]
- ✓ http://www.mug.org.ar/Descargas/Jornadas/Downloads_GetFile.aspx?id=3136.
- ✓ <http://www.php.net>. [En línea] [Citado el: 4 de Diciembre de 2012.]
- ✓ <http://www.php.net/manual/es/intro-whatcando.php>.
- ✓ <http://www.amazon.com>. [En línea] [Citado el: 23 de Diciembre de 2012.]
http://www.amazon.com/Design-Patterns-Elements-Reusable-Object-Oriented/dp/0201633612/ref=la_B000AQ1ZP8_1_1?ie=UTF8&qid=1339521804&sr=1-1#reader_0201633612.
- ✓ <http://www.propelorm.org/>. <http://www.propelorm.org/>. [En línea] [Citado el: 9 de Mayo de 2013.]
- ✓ <http://www.postgreSQL.org>. <http://www.postgreSQL.org>. [En línea] [Citado el: 23 de Abril de 2013.]
- ✓ José H. Canós, Patricio Letelier y M^a Carmen Penadés. *Métodologías Ágiles en el Desarrollo de Software*. [En línea] [Citado el: 30 de Noviembre de 2012.]
- ✓ www.willydev.net/descargas/prev/TodoAgil.pdf.
- ✓ Larman, Craig. *UML y Patrones*. México: PRENTICE HALL, 1999. Consultado 22 de Enero de 2013, Disponible en: <http://bibliodoc.uci.cu/pdf/reg00061.pdf>. ISBN: 970-17-0261-1.
- ✓ Linet Lores Sánchez, Diana Monné Roque. Trabajo de Diploma: Aplicación de las pruebas de liberación al Sistema Informático de Genética Médica (Junio 2009). Junio 2009.

- ✓ Mehdi Achour, Friedhelm Betz, Antony Dovgal, Nuno Lopes, Hannes Magnusson. *php. php*. [En línea] © 1997-2010 the PHP Documentation Group, 11 de Abril de 2011. [Citado el: 16 de noviembre de 2012.] <http://www.php.net/manual/es/>.
- ✓ Pérez Rodríguez, Yudith y Coutín, Adrián. *La gestión del conocimiento: un nuevo enfoque en la gestión empresarial*. 2005, ACIMED, Vol. 13. Disponible en:
- ✓ http://bvs.sld.cu/revistas/aci/vol13_6_05/aci040605.pdf.
- ✓ PHP. [En línea] [Citado el: 18 de Noviembre de 2011.] <http://www.php.net/>.
- ✓ PostgreSQL. [En línea] [Citado el: 17 de Noviembre de 2012.]
- ✓ http://www.computerworld.com.au/article/62894/postgresql_affiliates_org_domain.
- ✓ PruebaCasoDePrueba.
<http://www.mitecnologico.com/Main/PruebaCasoDePruebaDefectoFallaErrorVerificacionValidacion>.
[En línea]
- ✓ Tipos de Pruebas.
<http://www.cetic.guerrero.gob.mx/pics/art/articles/113/file.TiposPruebasSoftware.pdf>. [En línea]
- ✓ Shea Frederick, Colin Ramsay, Steve 'Cutter' Blades. *Learning Ext JS*. [ed.] Swapna V. Verlekar. Birmingham : Packt Publishing Ltd., 2008. ISBN 978-1-847195-14-2.
- ✓ Sipec. Sipec. [En línea] [Citado el: 15 de Noviembre de 2012.]
- ✓ http://sipec.sep.gob.mx/joomla/index.php?option=com_content&task=view&id=12&Itemid=33.
- ✓ Visual Paradigm. [En línea] [Citado el: 18 de Noviembre de 2012.]
- ✓ [http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_\(M%C3%8D\)_14720_p/](http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/).
- ✓ Woodman, Lynda. *Information management in large organizations. Information management from strategies to action*. London: ASLIB, 1985, págs. 95-97.
- ✓ Curto, Josep. *Information Management. Gestión de la información, Información, Teoría*. [En línea] [Citado el: 12 de noviembre de 2012.]
<http://informationmanagement.wordpress.com/category/gestion/gestion-de-la-informacion/>.
- ✓ http://www.ctr.unican.es/asignaturas/MC_OO/Doc. [En línea] [Citado el: 25 de enero del 2013].
- ✓ http://www.ecured.cu/index.php/Flujo_de_Trabajo_Analisis_y_Diseño. [En línea] [Citado el: 27 de enero del 2013].

- ✓ MSDN. <http://msdn.microsoft.com/en-us/library/ff649643.aspx>. [En línea] [Citado el: 25 de enero de 2013].
- ✓ Mühlrad, Daniel. Patrones de diseño. [En línea] 2008. [Citado el: 15 de febrero de 2013].
- ✓ Larman, Craig. Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design and Iterative Development. s.l. : Prentice Hall, 2005. ISBN 0-13-148906-2.
- ✓ PATRONES DE DISEÑO Y FRAMEWORKS. [En línea] [Citado el: 25 de febrero de 2013] <http://ingenieriasw2.blogspot.com/p/patrones-de-diseno-y-frameworks.html>.
- ✓ <http://users.dcc.uchile.cl/~psalinas/uml/interaccion.html> [En línea] [Citado el: 25 de febrero de 2013]
- ✓ Ivar Jacobson, Grady Booch, James Rumbaugh. El proceso unificado de desarrollo de software. Vol. 1, Consultado: 14 de febrero de 2013, Disponible en: <http://bibliodoc.uci.cu/pdf/reg00060.pdf>
- ✓ Slideshare. [En línea] [Citado el: 9 de mayo de 2013.] <http://www.slideshare.net/joshell/diagramas-uml-componentes-y-despliegue>.
- ✓ <http://portal.perueduca.edu.pe>. <http://portal.perueduca.edu.pe>. [En línea] [Citado el: 9 de Mayo de 2013.] http://portal.perueduca.edu.pe/modulos/m_taller/mapa.htm.
- ✓ Julio César Brito Rodríguez. Módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0.
- ✓ Pressman, Roger S.. *Ingeniería del software. Un enfoque práctico*. Lugar de publicación desconocido. Mc Graw Hill. 1998. 614.

GLOSARIO DE TÉRMINOS

Descarga

Es una información o un software obtenido de internet a petición de un usuario perteneciente al sistema.

Framework

Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

Incidencia

Son los eventos que afectan la disponibilidad de los recursos, como las roturas en los componentes del hardware o un mal funcionamiento del software.

Metodología

Un sistema de principios y normas generales de organización y estructuración teórico práctica de actividades.

ORM

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. En la práctica esto crea una base de datos orientada a objetos virtual.

Puesto de trabajo

En el marco del presente trabajo de investigación describen los recursos tangibles que se le brindan al personal integrado al centro para que puedan realizar sus funciones como informáticos.

Reporte

Los reportes agrupan datos como cantidad de computadoras en un laboratorio, uso del CPU y la RAM y las computadoras que presentan un mal funcionamiento y usuarios responsables de las mismas entre otros datos.

TCP/IP.

Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP).

UML

Lenguaje Unificado de Modelado (Unified Modeling Language).