Universidad de las Ciencias Informáticas

Facultad 6

"Sistema colaborativo de planificación y control de eventos curriculares"

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

Autor(es): Alain Tejas Escobar
Osiel Suárez Guerra

Tutor(es): Ing. Glennis Tamayo Morales
Lic. Yoandry Pacheco Aguila

La Habana, 2015

"Año 57 de la Revolución"

“Al mundo no le importará tu autoestima. El mundo esperará que logres algo, independientemente de que te sientas bien o no contigo mismo.”

Bill Gates

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

Alain Tejas Escobar

_____

Firma del Autor

Osiel Suárez Guerra

_____

Firma del Autor

Ing. Glennis Tamayo Morales

_____

Firma de la Tutora

Lic. Yoandry Pacheco Aguila

_____

Firma del Tutor

Datos de Contacto

Tutora:

Ing. Glennis Tamayo Morales

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo Electrónico: gtamayo@uci.cu

Tutor:

Lic. Yoandry Pacheco Aguila

Universidad de las Ciencias Informáticas, La Habana, Cuba

Correo Electrónico: andypa@uci.cu

Agradecimientos

Agradecimientos de Alain

Primero que todo a lo que considero más hermoso en la vida, mis padres Yoel y Elizabeth, que supieron, a pesar de la distancia, estar presentes en todo momento, apoyándome de todas las formas posibles y sirviéndome de ejemplo a seguir en la vida.

Gracias mami por darme tanto, y ser tan linda conmigo. Papi, tu sabes que siempre has sido mi mejor amigo, y que si vuelvo a nacer los quisiera tener a mi lado por siempre.

Agradecerle, a mis abuelos, Magda, Rolando, Alicia y Roberto, que sinceramente no sé cómo expresar todo lo que han hecho por mí desde el mismo momento en que nací.

A la persona más especial que he conocido en mi vida, Sonia, por sus cariños, monerías y malcriadeces, que me fueron enamorando desde el primer momento en que la conocí. Siempre estuvo presente para los lindos y difíciles instantes que hemos vivido en estos cinco años de universidad.

A todos mis tíos y primos, Roly, Frank, Blady, Robertico, Alex, Frank Yoel, y demás, que siempre estuvieron pendiente a mis resultados para darme aliento.

A la familia de mi novia, a sus padres y hermanos, con los que he compartido momentos muy lindos y han sabido ayudarme de forma incondicional.

A mi compañero de tesis, Osiel, que aunque tiene un carácter serio, supo reír y compartir conmigo en uno de los momentos más difíciles de la carrera, el proceso de tesis.

A los Tutores, que de no ser por su ayuda hubiera sido un camino muy difícil, mis más sinceros respetos y admiración por su labor profesional y entrega en todos los momentos que los necesité.

A mis compañeros de grupo, en especial Carlos, Frank, Yoan y Arian, que nunca los olvidaré.

A todas las personas que no mencioné, pero con las que compartí estos cinco años en la UCI y de las que me voy a llevar un grato recuerdo.

¡Muchas Gracias a todos!

Agradecimientos

Agradecimientos de Osiel

Quiero agradecer primeramente a mis abuelos Leida y Onel por apoyarme en todos los momentos y por estar siempre presente en cada instante de mi vida, al igual que a mi abuela Modesta, que aunque no está ya en este mundo siempre estuvo luchando conmigo desde que nací, a mis padres Moraima y Onel por haber confiado en mí y brindarme todo su apoyo. Gracias a su esfuerzo se hizo realidad este sueño.

A mis tíos Omar, Jorge, Eduardo y Cecilio por sus consejos, confianza, apoyo y constante preocupación.

A mi novia Mayi que en estos últimos 5 meses ha estado al tanto de mí, apoyándome y dándome todo su amor incondicionalmente.

A mis tutores Yoandry Pacheco y Glennis que nos guiaron, nos representaron y nos ayudaron para este momento tan importante.

Al profesor Castaño por guiarme y aconsejarme en momentos difíciles de la carrera.

De manera general a todas aquellas personas que de una forma u otra formaron o forman parte de mi vida.

"Gracias"

Dedicatoria

Dedicatoria de Alain

Le dedico este título a mis padres y abuelos, que realmente fueron la fuerza y el impulso que llevó a graduarme.

A mi bisabuela Lela, que aunque ya no se encuentra entre nosotros, siempre me quiso ver graduado y fue como una madre para mí.

A esa persona especial que desde que la conocí no ha dejado de ayudarme.

A todos los que formaron parte de la gran familia en la UCI, y me ayudaron en todo momento.

Dedicatoria de Osiel

Le dedico este trabajo a toda mi familia, especialmente a mis abuelos y a mis padres sin los cuales no hubiera llegado hasta aquí, les estaré eternamente agradecido por todo lo que han hecho por mí.

Resumen

La Universidad de las Ciencias Informáticas cuenta con una gran cantidad de profesores y especialistas, los cuales bajo las orientaciones del reglamento docente, tienen que realizar una gran variedad de actividades reflejadas en su plan de trabajo, que no solo están encaminadas a la superación y desempeño laboral, sino también a su superación científica y profesional. Teniendo en cuenta esto, sus evaluaciones dependen en gran medida del porciento de participación en cada una de estas actividades planteadas a nivel de facultad o universidad, surgiendo así la necesidad de llevar el control y evolución de las mismas, a través de diferentes aplicaciones. A pesar de ser aplicaciones que permiten la planificación de una u otra forma, no compensan de forma satisfactoria las deficiencias que existen actualmente en la universidad, concretándose la idea de desarrollar una aplicación web que permita la planificación y control de eventos curriculares en el entorno de la Universidad de las Ciencias Informáticas, donde se obtuvo un sistema capaz de encargarse de la planificación de estos eventos, organizar las actividades cronológicamente, la colaboración en la elaboración de planes de trabajo y a la vez su centralización y resguardo, junto a las consultas y análisis que se le pueden hacer a los reportes que genera dicha aplicación. Además, se plantearon un conjunto de pruebas para garantizar el correcto funcionamiento de las funcionalidades propuestas, obteniéndose como resultado un sistema que cumple con las necesidades de los docentes y especialistas en los procesos de planificación.

Palabras Claves:

Actividades, aplicación web, eventos curriculares, plan de trabajo.

Abstract

The University of Computer Sciences has a lot of professors and specialists, which under the guidance of teachers regulation, have to perform a variety of activities reflected in the work plan, which are not only aimed at the improvement and performance labor, but also its scientific and professional development. Given this, their evaluations rely heavily on the percentage of participation in each of these activities posed a college or university level, thus resulting in the need for control and development thereof, through different applications. Despite being applications that enable the planning of one form or another, they do not compensate satisfactorily the deficiencies that currently exist in the university, specifying the idea of developing a web application that allows planning and control curricular events in the environment University of Information Sciences, where a system capable of handling the planning of these events, organizing activities chronologically, collaboration in preparing work plans and also centralization and protection, with consultation and analysis was obtained they will be made to the reports generated by the application. In addition a set of tests were raised to ensure the proper functioning of the functionalities proposals, resulting in a system that meets the needs of teachers and specialists in planning processes.

Keywords:

Activities, web application, curricular events, workplan.

Índice

Índice de Figuras

Índice de Tablas

Introducción

La Universidad, como institución de Educación Superior tiene la función de preparar el personal competente en las diferentes materias de la vida, otorgando grados académicos y títulos profesionales que validan su preparación. En el caso de los estudiantes, les permite ampliar sus experiencias, potenciando su profesionalismo, como una importante vía para el desarrollo económico y social de cada país.

En Cuba, la Universidad juega un papel significativo en el modelo económico actual, siendo esta la principal fuente de recursos humanos cualificados con la preparación general integral necesaria para enfrentarse a la sociedad y a sus retos, con el más alto nivel y esfuerzo que caracteriza a un graduado universitario, acorde a su tiempo. Un ejemplo de ello es la Universidad de las Ciencias Informáticas (UCI), creada en el 2002 a partir de las ideas de su principal promotor, el Comandante en Jefe de la Revolución Fidel Castro Ruz, con el fin de formar jóvenes calificados con un alto nivel profesional, comprometidos con los principios y que deben mantener las conquistas de la Revolución, para, de esta forma, contribuir de modo significativo con el desarrollo económico y social del país en un contexto mundial de desarrollo acelerado de las tecnologías de la informática y las comunicaciones.

Esta universidad al igual que otras empresas e instituciones del país establece, con el fin de lograr sus objetivos, la elaboración periódica de los planes de trabajos de forma escalonada, en cada área y a diferentes niveles operativos, desde lo general hasta lo particular; tratando de mantener las líneas de trabajo en función de los objetivos de la empresa o institución. Estos planes generan un volumen creciente de información a medida que se desciende de nivel operacional, e incluso existen eventos que deben replicarse en diferentes niveles operativos.

La planificación es un proceso de concertación que por su carácter dinámico, evoluciona y se adecua a un contexto social, espacial y temporal; es el proceso mediante el cual se obtiene una visión del futuro, donde es posible determinar y lograr los objetivos, a través de la elección de un curso de acción. Una adecuada planificación contribuye al desarrollo profesional y empresarial, reduce riesgos e incrementa el aprovechamiento del tiempo y los recursos.(Escobedo, 2014)

Un plan de trabajo es considerado un instrumento de planificación, donde se sistematiza información de modo que pueda tenerse una visión del trabajo a realizar, indicando objetivos, metas, actividades, responsables y cronograma de eventos del currículo laboral, profesional, estudiantil o del área de trabajo.

Introducción

Es una herramienta que permite ordenar y sistematizar información relevante para realizar un trabajo. Este suele ser válido para un determinado período de tiempo, de manera que las acciones que propone deben desarrollarse en un cierto plazo y los objetivos tienen que ser cumplidos antes de una fecha límite.(Escobedo, 2014)

Por su parte la planificación se realiza de manera permanente, para cumplir los objetivos propuestos en las áreas de docencia, investigación, producción, extensión universitaria, entre otras. También se planifican las actividades a realizar por cada trabajador en cada una de las áreas anteriormente mencionadas. Este proceso no se ve materializado hasta que no es realizado el plan de trabajo, el cual hace uso de las actividades a desarrollar por los trabajadores para cumplir con las metas propuestas durante un período de trabajo.

La habilidad para planificar se considera un aspecto fundamental dentro de las competencias que definen el rol profesional de los profesores y especialistas, los cuales reciben un entrenamiento específico en tareas de confección y programación del currículo para ganar en eficiencia y calidad de los mismos.

La planificación en las áreas docentes y productivas, como áreas fundamentales dentro de la Universidad de las Ciencias Informáticas, son procesos secuenciales a través de los cuales se establecen una serie de pasos que conducen la enseñanza a una meta final.

Las principales dificultades que enfrenta la universidad en los procesos de planificación, están dadas por la falta de estandarización, necesidad de emplear mucho tiempo de la jornada laboral de los trabajadores docentes y especialistas en la elaboración de su plan de trabajo individual, la descentralización e integración manual y tardía de los planes de trabajo, lo cual dificulta o imposibilita el análisis, organización y control de las actividades docentes.

Las dificultades que aún persisten se contradicen con las exigencias actuales de la planificación y organización del trabajo, develando la siguiente interrogante como **problema de investigación**, ¿Cómo gestionar la integración de los planes de eventos curriculares que se realizan en la Universidad de las Ciencias Informática, para evitar retrasos en sus diseños y reducir las incompatibilidades entre los mismos?

Se define como **objeto de estudio:** El proceso de diseño y control de los planes de eventos curriculares que se realizan periódicamente dentro de la Universidad de las Ciencias Informáticas.

Introducción

Se define como **campo de acción:** Las herramientas informáticas para el diseño y control de los planes de eventos curriculares.

Para dar solución al problema planteado se define el siguiente **objetivo general:** Desarrollar un sistema colaborativo de planificación y control de eventos curriculares para evitar retrasos en sus diseños y reducir las incompatibilidades entre los mismos.

Teniendo como **objetivos específicos:**

1. Realizar un análisis de las herramientas necesarias para el desarrollo de un sistema de planificación y control de eventos curriculares en el entorno de la Universidad de las Ciencias Informáticas.
2. Realizar el análisis y diseño del sistema colaborativo de planificación y control de eventos curriculares.
3. Realizar el proceso de implementación y prueba del sistema colaborativo de planificación y control de eventos curriculares.

Para dar cumplimiento a los objetivos se definen las siguientes **tareas de investigación:**

1. Caracterización de los organizadores personales y herramientas de planificación para definir las funcionalidades del sistema.
2. Selección de los lenguajes y las herramientas para el desarrollo del sistema.
3. Identificación y descripción de las funcionalidades básicas del sistema para guiar el proceso de implementación.
4. Diseño de la solución a partir de las funcionalidades identificadas.
5. Implementación de la solución a partir de las funcionalidades identificadas.
6. Diseño de los casos de prueba para la realización de las pruebas a la solución.
7. Realización de pruebas de software para validar que la solución desarrollada cumpla con los requerimientos identificados.
8. Confección del Manual de Usuario para facilitar el uso del sistema.

Preguntas científicas:

- ✓ ¿Cómo llevar a cabo el proceso de caracterización de los organizadores personales y herramientas de planificación?

- ✓ ¿Cuáles son las características de los estándares para el registro y planificación de eventos curriculares?
- ✓ ¿Cómo determinar un apropiado lenguaje y herramientas para el desarrollo del sistema?
- ✓ ¿Cómo elaborar las funcionalidades del sistema, basado en los requisitos funcionales identificados?
- ✓ ¿Cómo realizar las pruebas a la solución implementada para validar su correcto funcionamiento?

Resultados esperados

- ✓ Un sistema de planificación de eventos curriculares para el personal de la universidad.
- ✓ Un sistema organizador de actividades cronológicas.
- ✓ Un sistema de colaboración en la elaboración de planes de trabajos.
- ✓ Un sistema de centralización y resguardo de los planes de trabajos.
- ✓ Un sistema de consulta y análisis curricular.

Para llevar a cabo las tareas propuestas y arribar satisfactoriamente al resultado final de la investigación se utilizan los siguientes métodos teóricos y empíricos de la investigación.

Métodos teóricos

Analítico-Sintético*:* Permite el análisis y síntesis de la bibliografía encontrada, de la cual se extraen los elementos más importantes que sirvan para la implementación del sistema en cuestión, y tener un mayor conocimiento del proceso de eventos curriculares en la Universidad.

Modelación: Posibilita realizar la abstracción del proceso de planificación y control de eventos curriculares, además de otros modelos para una mayor comprensión de los desarrolladores.

Histórico lógico: Se utiliza para resumir la evolución que han tenido los procesos de elaboración y control de los eventos curriculares.

Métodos empíricos

Entrevista: Realizada a profesores y trabajadores para determinar sus principales afectaciones con el proceso actual de planificación de tareas, y hacer uso de esta información para el levantamiento de requisitos funcionales. (Ver Anexo 1)

El presente documento se encuentra estructurado en tres capítulos:

Capítulo 1: Fundamentación Teórica

En este capítulo se hace referencia a los conceptos fundamentales relacionados con el proceso de planificación. Se investiga sobre algunas aplicaciones existentes en la universidad, relacionadas con este proceso, además de hacer un estudio sobre las principales herramientas de desarrollo, metodologías y tecnologías a utilizar en la confección del sistema colaborativo de planificación y control de eventos curriculares.

Capítulo 2: Análisis y diseño de la propuesta solución

Se hace una descripción detallada del sistema a desarrollar, facilitando así un mayor entendimiento por parte de los desarrolladores. Como algunos de los elementos que se abordan en este capítulo están: descripción del modelo de dominio, especificación de los requisitos funcionales y no funcionales, representación de los actores y casos de uso a través del diagrama de casos de uso del sistema; posibilitando a su vez identificar la arquitectura y los patrones de diseño a utilizar.

Capítulo 3: Implementación y prueba

En el presente capítulo, se realiza todo lo relacionado con los flujos de trabajo de implementación y prueba, donde se muestran los diagramas de despliegue y de componentes para dar una visión de los principales elementos físicos y de implementación necesarios para el sistema, al igual que las relaciones entre ellos. Se muestran fragmentos relevantes del código y las vistas principales de la aplicación; así como ejemplos de las pruebas funcionales aplicadas, que permite examinar las funcionalidades de la aplicación y garantizar la calidad del software.

CAPÍTULO 1: Fundamentación Teórica

Introducción

En el contenido del siguiente capítulo se abordan de forma general los conceptos fundamentales relacionados con el proceso de planificación que se realiza periódicamente dentro de la UCI, al igual que el resultado del estudio e investigación de las metodologías, herramientas, lenguajes de programación y tecnologías utilizadas en el proceso de desarrollo de software, enfocadas a la planificación de los docentes y especialistas. Así, a lo largo de un minucioso estudio del estado del arte, se pueden obtener los elementos necesarios y adecuados para desarrollar un sistema colaborativo de planificación y control de eventos curriculares.

1.1. Aplicaciones de planificación existentes en la Universidad de las Ciencias Informáticas.

En la Universidad de las Ciencias Informáticas existe heterogeneidad en los procesos de planificación así como en sistemas informáticos desarrollados para este fin, en los que se destacan:

GESPRO: Es el sistema de gestión de proyectos, considerado un planificador desarrollado en el lenguaje *Ruby*, sobre el marco de trabajo *Ruby on Rails*, mediante una personalización del Redmine, siendo este un sistema de propósito general en cuanto a los procesos de planificación en la universidad. Gestiona información sobre las principales áreas de cada centro de desarrollo sobre las tareas, los proyectos y los recursos humanos. El GESPRO, por cuestiones de rendimiento, en relación a la cantidad de usuarios conectados en un mismo instante y el tiempo de respuesta del sistema, está dirigido única y exclusivamente a la planificación y control de los procesos de producción de software y servicios informáticos.

DataFEU: Es un sistema de gestión desarrollado en el lenguaje PHP, que permite el registro y control de las actividades realizadas por los miembros de la organización estudiantil. Cuenta con los módulos Mi perfil, Mis AA, Mis dirigentes, Actividades, Mi página y Libro del Graduado, los cuales permiten visualizar información en el sistema referente al usuario, ya sea de su perfil, de sus alumnos ayudantes, de sus dirigentes y de las actividades que ya se han planificado a nivel de facultad o de universidad.

Calendario de Zimbra: Es un módulo integrado al cliente de correo web Zimbra, que permite que cada usuario planifique y organice sus actividades, en las cuales puede incluir la participación de otros usuarios que intervienen en el desarrollo de la misma, además de notificar las actividades pendientes.

Sistema de Gestión y Control de las Tareas: Es un sistema de gestión sencillo desarrollado en .NET que usa como gestor de bases de datos PostgreSQL, donde los profesores del Departamento de Programación y Sistemas Digitales de la Facultad 6 registran sus actividades en las áreas de formación, producción, investigación, postgrado y extensión universitaria, sirviéndoles como un sistema de planificación. Cuenta con varios módulos, entre ellos se encuentra el de Balance del claustro, en este se pueden realizar análisis estadísticos de los profesores más destacados, tareas cumplidas, incumplidas, entre otros datos útiles, además de generar un gráfico en función de los mismos. Fue creado con el objetivo de facilitar el proceso de evaluación del desempeño laboral de cada profesor en el departamento.

Horario docente: Este sistema está diseñado única y exclusivamente para la planificación de las actividades docentes de cada brigada en cada una de las asignaturas. Está desarrollado con la tecnología .NET y desplegado sobre el servidor de Microsoft Internet Information Services. Contiene funcionalidades que permiten filtrar la búsqueda por facultad, semana, año, brigada, actividad, profesor y local, además de una opción avanzada, que permite otros criterios de búsqueda.

Herramientas de Office: Actualmente esta es la herramienta más utilizada en el proceso de planificación dentro de la universidad, ya sea usando Microsoft Word o Microsoft Excel, donde se generan documentos en diferentes formatos, que se distribuyen por correo electrónico entre los diferentes niveles operacionales.

Aún cuando estos sistemas de gestión de la planificación, facilitan la gestión de la información relacionada con el proceso de organización de las tareas en las diferentes áreas, esta se ve afectada en alguno de los casos por su ejecución manual y descentralizada, ya que no todos los departamentos la realizan de la misma forma ni usan los mismos sistemas. En algunos casos existen retrasos en los documentos que se generan y envían a las diferentes áreas. Además los sistemas mencionados presentan deficiencias en cuanto a rendimiento, soberanía tecnológica, homogeneidad en la estructura de la información o son de propósitos muy específicos de un área de trabajo.

De lo anterior se puede concluir que es necesario crear un sistema colaborativo de planificación y control de eventos curriculares, que permita a cada usuario desde su área consultar los planes de trabajo públicos, registrar e importar nuevas actividades en su plan de trabajo organizándolas por categorías en

función del área en que se desempeña, en relación directa con las áreas de resultados claves definidas en la universidad, con el objetivo de lograr tener un mayor control de sus actividades y en función de eso emplear mejor el tiempo, aumentando significativamente la eficiencia y participación en cada una de sus tareas.

1.2. Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos de software, en la que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.(Universidad de Murcia, 2011)

1.2.1. *Metodología Ágil*

La metodología ágil combina una filosofía y un conjunto de directrices de desarrollo, busca la satisfacción del cliente y la entrega temprana de software incremental, equipos de proyectos pequeños y con una alta motivación, utilizando métodos informales, con un mínimo de productos de trabajo de la ingeniería del software y una simplicidad general del desarrollo. (Pressman, 2005)

Las directrices de desarrollo resaltan la entrega sobre el análisis y el diseño (aunque estas actividades no se descartan), y la comunicación activa y continua entre los desarrolladores y los clientes.

Estos principios son las características que diferencian un proceso ágil de uno tradicional. Los dos primeros son generales y resumen gran parte del espíritu ágil:

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor. Un proceso es ágil si a las pocas semanas de empezar ya entrega software que funcione aunque sea rudimentario. El cliente decide si pone en marcha dicho software con la funcionalidad que ahora le proporciona o simplemente lo revisa e informa de posibles cambios a realizar.

II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva. Este principio es una actitud que deben adoptar los miembros del equipo de desarrollo. Los cambios en los requisitos deben verse como algo positivo. Les va a permitir aprender más, a la vez que logran una mayor satisfacción del cliente. Este principio implica

además que la estructura del software debe ser flexible para poder incorporar los cambios sin demasiado coste añadido. El paradigma orientado a objetos puede ayudar a conseguir esta flexibilidad.

De forma general este tipo de metodología tiene como principales características que se encarga de valorar al individuo y las iteraciones del equipo más que a las herramientas o los procesos utilizados; le es mucho más importante crear un producto de software que funcione, donde el cliente esté en todo momento colaborando en el proyecto y se genere solo la documentación necesaria.

Desde el surgimiento de estas metodologías de desarrollo ágil, han surgido diferentes variantes como es el caso de SCRUM, *OpenUP* (Open Unified Process), Crystal Methodologies, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Feature-Driven Development (FDD), Lean Development (LD) y Programación Extrema (EXTREME PROGRAMMING, XP).

Open UP

La metodología Open UP, en inglés *Open Unified Process*, es una metodología de Proceso Unificado que aplica enfoques iterativos e incrementales dentro de un ciclo de vida estructurado, utiliza una filosofía ágil que se enfoca en la naturaleza de colaboración para el desarrollo de software, basada en RUP (Rational Unified Process), que contiene el conjunto mínimo de prácticas que ayudan a un equipo de desarrollo de software a realizar un producto de alta calidad, de una forma eficiente. (Barrios, 2014) .

Esta metodología fue propuesta por el grupo de empresas conformado por: IBM Corp, Telelogic AB, Armstrong Process Group Inc., Number Six, Software Inc. y Xansa; quienes la donaron a la Fundación Eclipse en el año 2007, que la ha publicado bajo licencia libre.

Es considerada un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es incluido. No abarca todos los elementos que se manejan en un proyecto pero tiene los componentes básicos que pueden servir de base a procesos específicos y la mayoría de los elementos de esta metodología están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances.

El ciclo de vida de un proyecto, según la metodología Open UP, permite que los integrantes del equipo de desarrollo aporten con micro-incrementos, que pueden ser el resultado del trabajo de unas pocas horas o unos pocos días. El progreso se puede visualizar diariamente, ya que la aplicación va evolucionando en función de este micro-incremento. El objetivo de Open UP es ayudar al equipo de desarrollo, a lo largo de

todo el ciclo de vida de las iteraciones, para que sea capaz de añadir valor de negocio a los clientes, de una forma predecible, con la entrega de un software operativo y funcional al final de cada iteración.

Cuenta con cuatro fases, las cuales se muestran en la figura 1: inicio, elaboración, construcción y transición.

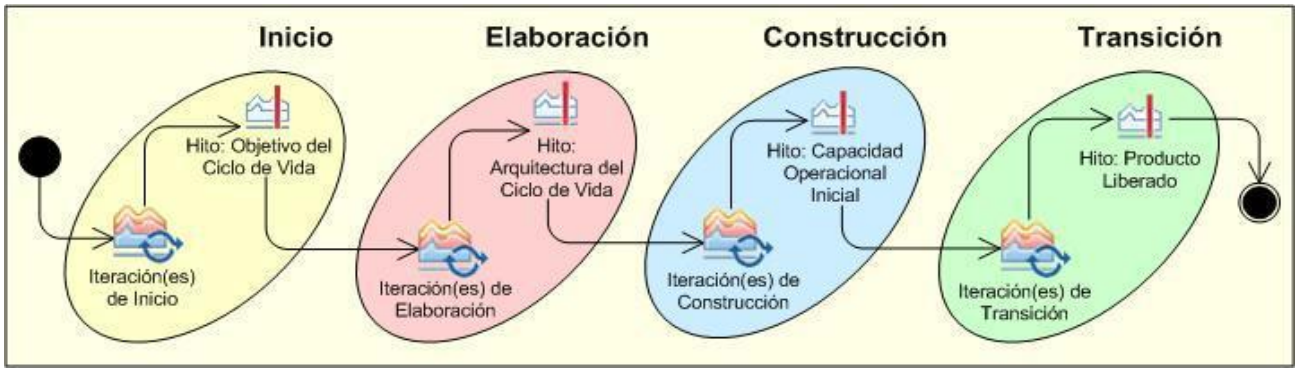


Fig. 1 Ciclo de vida de Open UP

- ✓ Fase de Inicio: Se esbozan las arquitecturas candidatas, se identifican los riesgos, se prepara un plan de proyecto y estimación de costes.
- ✓ Fase de Elaboración: Se realizan tareas de análisis del dominio y definición de la arquitectura del sistema, se elabora el plan de proyecto, se establecen todos o algunos requisitos y la arquitectura estable del sistema. Al finalizar esta fase se debe tener una definición clara y precisa de los casos de uso y todos sus actores.
- ✓ Fase de Construcción: Es la fase más larga del proyecto, el sistema es construido sobre la base de lo especificado en la fase de elaboración y el resultado de cada iteración es una versión ejecutable del software.
- ✓ Fase de Transición: En esta fase el sistema es desplegado para los usuarios finales y la retroalimentación recibida permite incorporar refinamientos al sistema en las sucesivas iteraciones. Esta iteración también cubre el entrenamiento de los usuarios para la utilización del sistema.

Considerando los elementos y características de Open Up, como es el caso de poder disminuir las posibilidades de riesgo y descubrir errores tempranos a través de ciclos iterativos, ayudando significativamente al equipo de desarrollo, a lo largo de todo el ciclo de vida de las iteraciones, con la

entrega de un software operativo y funcional al final de cada iteración, se define esta como la metodología de desarrollo de software a utilizar.

1.3. Herramientas y tecnologías actuales

El proceso de desarrollo de aplicaciones de software normalmente involucra varias etapas. Los programas de software usan muchos lenguajes y tecnologías diferentes, con las herramientas típicamente creadas para tecnologías específicas. El desarrollo de software puede ser una actividad compleja y larga, por lo que las herramientas disponibles pueden reducir el estrés y aumentar el desempeño tanto de desarrolladores como de las aplicaciones resultantes. Existen herramientas para cada etapa en el proceso de desarrollo de software.

1.3.1. *Servidor web*

Un servidor web es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir a lo que es llamado hipertextos, páginas web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.(cristalab, 2008)

NodeJS 0.10.24

NodeJS es una plataforma robusta y escalable, especialmente diseñada para realizar operaciones de entrada / salida (Input / Output o simplemente I/O en inglés) en redes informáticas por medio de distintos protocolos, apegada a la filosofía UNIX[1].(Gámez, 2012) Está basado en eventos, *streams* y construido encima del motor de JavaScript V8, que es con el que funciona el JavaScript de Google Chrome. Este es programado del lado del servidor, lo que indica que los procesos para el desarrollo de software en "Node" se realizan de una manera muy diferente que los de JavaScript del lado del cliente.

Otras de las características es su programación asíncrona, la cual es capaz de crear diferentes hilos, con diferentes procesos que llevarán un tiempo en ejecutarse, de modo que se hagan todos a la vez. Además, se podrá especificar código (*callbacks*) que se ejecute al final de cada uno de esos procesos largos, permitiendo un aumento significativo en la escalabilidad de los sistemas. De igual manera cuenta con la programación orientada a eventos, utilizando un modelo basado en eventos, donde el servidor acepta la petición y la envía a la fila, atiende la siguiente petición y las continúa enviando a la fila, cuando la primera

[1] Unix (registrado oficialmente como UNIX®) es un Sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969, por un grupo de empleados de los laboratorios Bell de AT&T.

petición o las subsecuentes terminan la respuesta, se envía a la fila y al llegar su turno devuelve la respuesta. La diferencia está en el modelo basado en eventos, que siempre se mantiene ocupado sin esperar a que las otras conexiones terminen las operaciones que bloquean ciertos recursos, con la particularidad que los eventos en esta plataforma son orientados a cosas que suceden del lado del servidor y no del lado del cliente como los que se conocen anteriormente en JavaScript.

Existen varios ejemplos de sitios y empresas que ya están usando NodeJS donde los casos de éxito son realmente representativos. Quizás el más evidente sea el de LinkedIn, la plataforma de contacto entre profesionales a modo de red social. Al pasar a NodeJS, LindkedIn ha reducido sensiblemente el número de servidores que tenían en funcionamiento para dar servicio a sus usuarios, específicamente de 30 servidores a 3, donde se evidencia un aumento de la eficiencia y ahorro de recursos de forma significativa.(Gámez, 2012)

NodeJS tiene un *footprint*[2] de memoria bajo, es decir, los procesos de NodeJs ocupan niveles de memoria sensiblemente menores que los de otros lenguajes, por lo que los requisitos de servidor para atender al mismo número de usuarios son menores. Por aproximar algo, se podría llegar a tener 1.000 usuarios conectados a la vez y el proceso de NodeJS ocuparía solamente 5 MB de memoria. Al final, todo esto se traduce en que empresas grandes pueden tener un ahorro importante en costes de infraestructura. Otros ejemplos, además de LinkedIn son eBay, Microsoft, empresas dedicadas a *hosting* como Nodester o Nodejitsu, redes sociales como Geekli.st, y muchos más.(Gámez, 2012)

Luego de unas pruebas realizadas con la herramienta jMeter, se obtuvieron los siguientes resultados con respecto a los lenguajes de Node, Node-Cluster, Java y PHP. Se estableció la comparación según la cantidad de hilos concurrentes, ciclos y elementos de muestra, obteniéndose los siguientes elementos:

- ✓ Tuvo como resultado con 8 hilos concurrentes, 125 ciclos para un total de 1000 muestras, un tiempo de respuesta promedio de 237 milisegundos con PHP, 64 milisegundos con Node, 58 milisegundos con Node-Cluster y 58 milisegundos con Java.
- ✓ Tuvo como resultado con 10 hilos concurrentes, 50 ciclos para un total de 1000 muestras, un tiempo de respuesta promedio de 685 milisegundos con PHP, 131 milisegundos con Node, 92 milisegundos con Node-Cluster, 115 milisegundos con Java.

[2] El *footprint* de memoria de un programa ejecutable indica sus requerimientos de memoria en tiempo de ejecución, mientras que el programa se ejecuta. Esto incluye todo tipo de regiones de memoria activos como el segmento de código que contiene.

- ✓ Tuvo como resultado con 20 hilos concurrentes, 50 ciclos para un total de 1000 muestras, un tiempo de respuesta promedio de 709 milisegundos con PHP, 113 milisegundos con Node, 91 milisegundos con Node-Cluster, 118 milisegundos con java.

Para un mayor entendimiento se pueden observar en el Anexo 2, las gráficas que demuestran dichos resultados.

Se puede apreciar que los tiempos de respuesta con NodeJS son menores a los demás lenguajes, siendo considerado esto un elemento determinante en la elección de este lenguaje como intérprete y servidor web utilizado en el desarrollo de la aplicación.

1.3.2. *Lenguaje de modelado*

Lenguaje de Modelado Unificado (UML por sus siglas en inglés) es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos (OO). Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software.(Larman, 2002)

UML es un lenguaje estándar con el que es posible modelar todos los componentes del proceso de desarrollo de aplicaciones. Este lenguaje permite enseñar al cliente una posible aproximación de lo que será el producto final, dándole un primer acercamiento al problema que permite visualizar cuál sería el resultado.

Herramienta CASE

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés, *Computer Aided Software Engineering*), son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. Estas facilitan tareas como el diseño del proyecto, el cálculo de costos, la implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras.

Existe un variado número de herramientas CASE, cada una con sus propias particularidades, que van desde la complejidad, variedad de opciones, la dedicación a diferentes tipos de proyectos hasta los costos; algunas de estas son *Rational Rose, Umbrello, Argo UML y Visual Paradigm for UML*.

Visual Paradigm 8.0

Visual Paradigm es una herramienta CASE que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. En la etapa de análisis y diseño de la aplicación deseada es indispensable su uso, ya que esta tiene una serie de características que la hacen extremadamente útil. Como principales características que presenta esta potente herramienta se encuentran:

- ✓ Disponibilidad de uso en múltiples plataformas, ya sea Windows o Linux.
- ✓ Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- ✓ Licencia gratuita y comercial.
- ✓ Generación de bases de datos a través de la transformación de diagramas de Entidad-Relación en tablas de base de datos.
- ✓ Soporta aplicaciones web.
- ✓ Soporte de UML versión 2.1.
- ✓ Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- ✓ Comparte y genera los diagramas y diseños en formatos como PDF20, HTML, JPG y Microsoft Word.
- ✓ Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.

1.3.3. *Lenguaje de Programación*

Un lenguaje de programación no es más que un sistema estructurado y diseñado principalmente para que las máquinas y computadoras se entiendan entre sí y con los seres humanos. Contiene un conjunto de acciones consecutivas que el ordenador debe ejecutar.(Lenguajes De Programación, 2010)

Los lenguajes de programación pueden clasificarse según el paradigma que usan en: procedimentales, orientados a objetos, funcionales, híbridos y lógicos.

Estos facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar.

JavaScript

Es el lenguaje interpretado orientado a objetos desarrollado por Netscape que se utiliza en millones de páginas web y aplicaciones de servidor en todo el mundo. JavaScript de Netscape es un gran conjunto del lenguaje de scripts del estándar ECMA-262 de (ECMAScript[3]) que presenta sólo leves diferencias respecto a la norma publicada.(Alvarez, 2010)

JavaScript es un lenguaje de programación dinámico que soporta construcción de objetos basado en prototipos. La sintaxis básica es similar a Java y C++ con la intención de reducir el número de nuevos conceptos necesarios para aprenderlo. Las estructuras, tales como sentencias if, y bucles for y while, y bloques switch y try... catch funcionan de la misma manera que en estos lenguajes o prácticamente igual.

Las capacidades dinámicas de JavaScript incluyen construcción de objetos en tiempo de ejecución, listas variables de parámetros, variables que pueden contener funciones, creación de scripts dinámicos (mediante eval[4]), introspección de objetos (mediante for...in), y recuperación de código fuente (los programas de JavaScript pueden recuperar el código fuente original del cuerpo de las funciones. También está compuesto por los objetos intrínsecos Number, String, Boolean, Date, RegExp y Math.(Alvarez, 2010)

Características de JavaScript

Compilador: No es un lenguaje que necesite que sus programas se compilen, sino que éstos se interpretan por parte del navegador u otro intérprete como NodeJS

Orientado a objetos: Está formado casi en su totalidad por objetos, los cuales encapsulan los datos y el comportamiento.

Propósito: Generalmente para escribir programas que se ejecuten en páginas web.

[3] **ECMAScript** es una especificación de lenguaje de programación publicada por ECMA Internacional. Define un lenguaje de tipo dinámico, ligeramente inspirados en Java y otros al estilo de C.
[4] **Eval** en varios lenguajes de programación, es una función que evalúa el contenido pasado por parámetro como si fuera una expresión.

Chequeo de tipo: Permite asignar a una variable la información que se desee, independientemente del tipo de ésta, además de poder cambiar el tipo de información de una variable en cualquier momento.(Pérez, 2009)

Teniendo en cuenta las características que presenta JavaScript junto a las pruebas realizadas a otros lenguajes, se obtiene que sus cualidades de flexibilidad, compatibilidad, dinamismo y simplicidad en el desarrollo web, lo convierten en un factor determinante para ser el lenguaje seleccionado para el desarrollo del sistema.

1.3.4. *Marco de trabajo*

Qooxdoo 4.1

Es un marco de trabajo de JavaScript que permite crear aplicaciones para una amplia gama de sistemas informáticos. Con su modelo de programación orientado a objetos puede crear aplicaciones de internet enriquecidas (RIA[5]), aplicaciones para dispositivos móviles, ligeras aplicaciones web tradicionales, o incluso hacer aplicaciones que se ejecuten fuera del explorador.

Qooxdoo aprovecha las tecnologías web más modernas, como HTML5 y CSS3. Es de código abierto, contando con plataformas totalmente basadas en clases que tratan de aprovechar las características orientadas a objetos de JavaScript. Se basa completamente en los espacios de nombres y no se extienden de los tipos nativos de JavaScript para permitir una fácil integración con otras bibliotecas y códigos de usuarios existentes.(Qooxdoo, 2014)

Características

Una aplicación típica de qooxdoo se crea mediante el aprovechamiento de las herramientas de desarrollo integrado y el modelo de programación del lado del cliente basada en orientación a objetos de JavaScript.

Tiempos de ejecución

Qooxdoo soporta una amplia gama de entornos de JavaScript:

- ✓ Navegadores como Internet Explorer, Firefox, Opera, Safari y Chrome.
- ✓ Navegadores móviles (iOS[6], Android[7]).

[5] Las rich Internet application, o **RIA** (en español "aplicaciones de Internet enriquecidas"), son aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales.
[6] **iOS** es un sistema operativo móvil de la multinacional Apple Inc. Originalmente desarrollado para el iPhone (iPhone OS), después se ha usado en dispositivos como el iPod touch y el iPad.

- ✓ No requiere de plugins como ActiveX, Java, Flash ni Silverlight.
- ✓ No presenta críticas modificaciones de los objetos nativos de JavaScript para permitir una fácil integración con otras bibliotecas y código personalizado.

Orientación a Objetos

- ✓ Qooxdoo está basado totalmente en clases.
- ✓ Además de las clases regulares, ofrece clases abstractas y estáticas.
- ✓ Tiene constructores y destructores.
- ✓ Herencia simple, polimorfismo completo.

Programación

- ✓ Puro JavaScript.
- ✓ No requiere conocimientos de HTML.
- ✓ No se requieren conocimientos de CSS.
- ✓ No se requieren conocimientos necesarios de DOM.
- ✓ Soporte completo para programación basada en eventos.
- ✓ Desarrollo de aplicaciones qooxdoo totalmente compatibles con todas las plataformas, como Windows, Linux, todos los sistemas Unix, Mac OS X.
- ✓ Muchas aplicaciones de muestra y ejemplos.
- ✓ Diseñado para un alto rendimiento.
- ✓ Gestión de historial del navegador, es decir, botón atrás / adelante, marcadores.

Referencia de la API[8]

- ✓ Comentarios del código fuente.
- ✓ Completa referencia de la API.
- ✓ La funcionalidad de búsqueda.

---

[7] **Android** es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tablets; y también para relojes inteligentes, televisores y automóviles.
[8] La interfaz de programación de aplicaciones, abreviada como API, es el conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Guaraiba 0.1

Es un marco de trabajo desarrollado en JavaScript y qooxdoo que permite la creación de aplicaciones web para NodeJS bajo el patrón arquitectónico Modelo Vista Controlador. Está basado en el marco de trabajo "Ruby on Rails" y aprovecha las bondades de qooxdoo para el desarrollo, compilación y documentación de aplicaciones bajo el paradigma de programación orientada a objetos.

Características

- ✓ Una aplicación en Guaraiba se crea a partir de una aplicación servidora en qooxdoo que hereda de la clase guaraiba.Application. De igual modo ofrece clases bases para la implementación de los controladores, las clases del modelo, el controlador de rutas y la configuración de la aplicación.
- ✓ Las clases bases del modelo de datos facilitan la interacción con bases de datos sobre los gestores MySQL, MariaDB, PostgreSQL, SQLite3 y Oracle.
- ✓ Ofrece múltiples servicios de autenticación tales como local, ldap, soap, facebook, google entre otros de la familia de los módulos passport de NodeJS.
- ✓ Contiene una capa de abstracción para el desarrollo de servicios REST[9] completos.
- ✓ Permite la implementación de las vistas en los formatos ejs, handlebars, jade, jasper report, mustache y swig.
- ✓ Permite de forma nativa y en correspondencia con las vistas implementadas, generar las respuestas a las peticiones en los formatos txt, json, js, xml, html, xhtml, pdf, docx, rtf, pptx, xlsx, xls, csv, odt, ods, odp, swf y jpeg.
- ✓ Permite configurar y ejecutar la aplicación en forma de cluster en relación con el número de procesadores disponibles en el servidor.

1.3.5. *Entorno de Desarrollo Integrado*

NetBeans 7.4

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para una amplia variedad de lenguajes que tiene soportado. Existe además un número importante de módulos para extender el

[9] **REST** por sus siglas en inglés (*Representational State Transfer*) es un estilo de arquitectura software para sistemas hipermedia distribuidos como la *World Wide Web.*

NetBeans IDE. Es un producto libre y gratuito sin restricciones de uso. Está diseñado bajo código abierto, por lo que queda disponible para su reutilización. (NetBeans, 2012)

Este se ejecuta sobre una máquina virtual de Java, que permite su ejecución en Windows, Linux, Mac OS X y Solaris. Contiene todos los módulos necesarios para el desarrollo de aplicaciones Java. Permite la ingeniería inversa, así como la integración con diferentes marcos de trabajo. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Se ha convertido en un IDE apto para la mayoría de los lenguajes de programación de código abierto modernos, por lo que se determina que este sea el entorno de desarrollo a utilizar en el proceso de implementación del sistema.

1.3.6. *Motor de Reportes*

JasperReports 5.6

JasperReports es una biblioteca de clases Java de código abierto que proporciona una API para facilitar la capacidad de generar informes a partir de cualquier tipo de aplicación Java. Está licenciado bajo la Licencia Pública GNU Lesser (LGPL). Esta licencia fue elegida para JasperReports, ya que, a diferencia de la GPL, permite a JasperReports ser utilizado en aplicaciones tanto de código abierto como en las privativas. De esta forma, si se desean hacer modificaciones al código fuente del JasperReports, tendrá que ser publicado bajo la LGPL.(Heffelfinger, 2006)

Además de los datos textuales, JasperReports es capaz de generar informes profesionales, incluyendo imágenes, tablas y gráficos. Algunas de sus principales características incluyen:

- ✓ Diseño del informe flexible.
- ✓ Capaz de presentar los datos textual o gráficamente.
- ✓ Permite a los desarrolladores que proporcionen datos de múltiples maneras.
- ✓ Se puede aceptar datos de múltiples fuentes de datos.
- ✓ Puede generar marcas de agua.
- ✓ Puede generar informes integrados.
- ✓ Es capaz de exportar informes a una variedad de formatos.

Informe de diseño flexible: permite separar los datos en las secciones del informe. Además de permitir definir las secciones, JasperReports permite la creación de diseños elaborados dinámicamente, basados

en el contenido del informe. Por ejemplo, dependiendo del valor de un campo, los datos se pueden ocultar o mostrar en el mismo, o pueden ser agrupados en secciones lógicas.

Múltiples formas de presentar datos: ofrece la posibilidad de mostrar los datos del informe textualmente o gráficamente, permite utilizar las expresiones del mismo para generar informes que muestren datos dinámicos.

Los informes generados con JasperReports se pueden exportar en varios formatos, como pdf, xls, rtf, html, xml, csv y texto sin formato. También hay una biblioteca de terceros para exportar informes JasperReports al formato OpenDocument (ODF). El formato OpenDocument es una especificación de formato de archivo basado en XML estándar para aplicaciones de oficina elaboradas por la Organización para el Avance de Estándares de Información Estructurada (OASIS). OpenOffice.org versión 2.0 utiliza ODF como formato predeterminado.

1.3.7. *Sistema Gestor de Base de Datos (SGBD)*

Un Sistema Gestor o Manejador de Bases de Datos (SGBD) es un conjunto de programas que permiten a los usuarios crear y mantener una base de datos (BD). Por lo tanto, el SGBD es un software de propósito general que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones.

PostgreSQL 9.3

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[10] y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará al resto y el sistema continuará funcionando.(Martínez, 2010)

Sus características técnicas lo convierten en uno de los gestores de bases de datos más potentes y robustos del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez a el sistema.

A continuación se muestran algunas de las características más importantes y soportadas por PostgreSQL:

[10] BSD es una licencia de software libre permisiva como la licencia de OpenSSL o la MIT License, que permite el uso del código fuente en software no libre.

- ✓ Es un gestor de base de datos 100% *ACID* [11].
- ✓ Replicación asincrónica/sincrónica / *Streaming replication - Hot Standby*.
- ✓ *Presenta dos fases de confirmación (Two-phase commit*).
- ✓ Almacenamiento y copia continua de todas las transacciones (PITR - *point in time recovery*).
- ✓ Copias de seguridad en caliente (*Online/hot backups*).
- ✓ Estándar de codificación internacional (Unicode).
- ✓ Juegos de caracteres internacionales.
- ✓ Regionalización por columna.
- ✓ Control de concurrencia mediante versiones múltiples (*Multiversion concurrency control* o MVCC).
- ✓ Múltiples métodos de autenticación.
- ✓ Acceso cifrado vía SSL.
- ✓ Actualización in-situ integrada (pg_upgrade).
- ✓ Control de acceso SE-postgres.
- ✓ Completa documentación.
- ✓ Licencia BSD.
- ✓ Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX,Mac OS X, Solaris, Tru64) y Windows 32/64bit.

Es considerado el sistema gestor de bases de datos a usar en la aplicación ya que se caracteriza, entre otros elementos, por su robustez, estabilidad, potencia y capacidad de trabajo con las bases de datos.

1.3.8. *Herramienta de Administración de Base de Datos*

El software de administración de bases de datos es la herramienta principal de software del enfoque de la administración de base de datos, dado que controla la creación, el mantenimiento y el uso de la base de datos de una organización y de sus usuarios finales. (O'Brien, 2006)

[11] **ACID** por sus siglas en inglés (*Atomicity, Consistency, Isolation y Durability*), en español, Atomicidad, Consistencia, Aislamiento y Durabilidad, son un conjunto de propiedades necesarias para que un conjunto de instrucciones, sean consideradas como una transacción en un sistema de gestión de bases de datos.

pgAdmin 1.16.1

Es una aplicación de diseño y manejo de bases de datos para su uso con PostgreSQL.(pgAdmin3, 2009) La aplicación se puede utilizar para manejar PostgreSQL 7.3 o superiores y funciona sobre casi todas las plataformas. Este software fue diseñado para responder a las necesidades de todos los usuarios, desde la escritura de simples consultas SQL a la elaboración de bases de datos complejas. La interfaz gráfica es compatible con todas las características de PostgreSQL y facilita la administración. La aplicación también incluye un editor de la sintaxis SQL, un editor de código del lado del servidor, un agente para la programación de tareas «SQL/batch/shell», soporte para el motor de replicación Slony-I y mucho más. La conexión del servidor se puede realizar mediante TCP/IP o Unix Domain Sockets (en plataformas *nix), y puede ser cifrado mediante SSL por seguridad. Esta incluye como algunas de sus funcionalidades:

- ✓ Interfaz administrativa gráfica.
- ✓ Herramienta de consulta SQL (con un EXPLAIN gráfico).
- ✓ Editor de código procedural.
- ✓ Agente de planificación SQL/shell/batch.
- ✓ Administración de Slony-I.

La interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración. Está disponible en más de una docena de lenguajes y para varios sistemas operativos, incluyendo Microsoft Windows, Linux, FreeBSD, Mac OSX y Solaris.

Conclusiones del capítulo

Las herramientas estudiadas que existen en la UCI carecen de algunas funcionalidades necesarias en el proceso de planificación, de ahí, se decide desarrollar un sistema colaborativo de planificación y control de eventos curriculares. Se seleccionó como metodología de desarrollo de software Open UP, la cual guiará todo el proceso de desarrollo del sistema. Se seleccionó JavaScript como lenguaje de programación, Visual Paradigm 8.0 como herramienta CASE y UML como lenguaje de modelado. El entorno de desarrollo NetBeans 7.4, utilizando los marcos de trabajo Qooxdoo 4.1 y Guaraiba 0.1, como intérprete y servidor web NodeJS 0.10.24 además hacer uso de PostgreSQL 9.3 como sistema gestor de bases de datos.

Capítulo 2: Análisis y diseño de la propuesta solución

Introducción

En este capítulo se exponen una serie de características que ayudan en gran medida a los desarrolladores a comprender de una mejor forma la aplicación a desarrollar. Se representará el modelo del dominio, se especificarán los requisitos funcionales y no funcionales necesarios, dando paso a la confección del diagrama de casos de uso, el cual muestra la relación entre los actores que intervienen con el sistema y sus respectivas descripciones, además de definir la arquitectura y los patrones de diseño a utilizar.

2.1. Modelo de Dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno en el que trabaja el sistema.(Jacobson, 2000) Muchos de estos objetos del dominio o clases pueden obtenerse de una especificación de requisitos o mediante la entrevista con los expertos del dominio. El modelo del dominio se describe mediante diagramas de UML, especialmente mediante diagramas de clases. Estos muestran a los clientes, usuarios, revisores y a otros desarrolladores las clases del dominio y cómo se relacionan entre sí mediante asociaciones.

Este puede ser tomado como el punto de partida para el diseño del sistema, ya que cuando se realiza la programación orientada a objetos, el funcionamiento interno del software va a imitar en alguna medida a la realidad. En la figura 2 se representa el diagrama del modelo de dominio realizado.

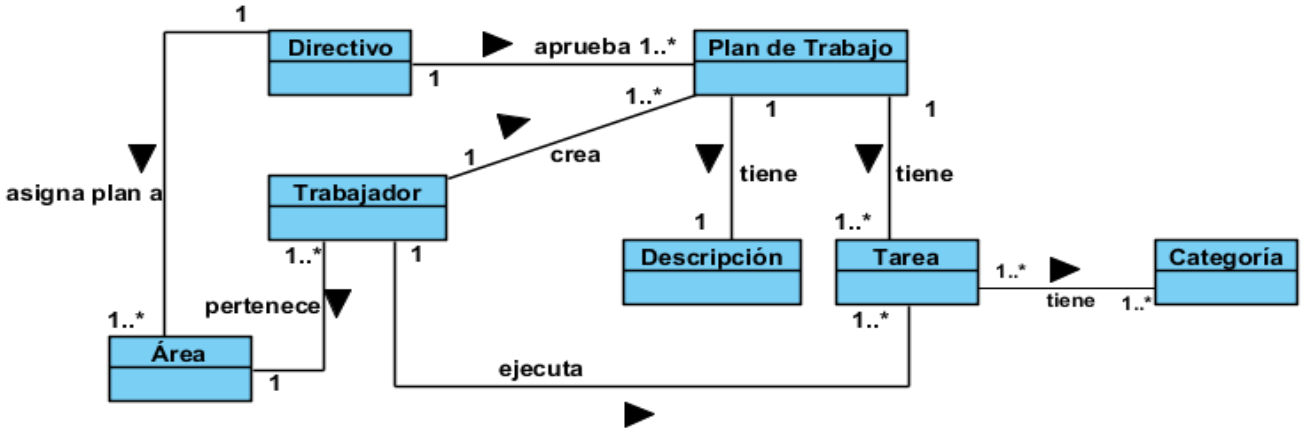


Fig. 2 Diagrama del Modelo de Dominio

Entidades y conceptos principales

Tabla 1. Entidades y conceptos principales

Concepto	Descripción
Trabajador	Persona que trabaja en la Universidad.
Directivo	Persona responsable de revisar y asignar un plan de trabajo a un área determinada.
Plan de Trabajo	Documento que se confecciona para mantener el control y la planificación de una organización. Contiene las tareas de un área o persona.
Área	Se refiere a las diferentes áreas en las que está dividida el árbol jerárquico de la Universidad.
Tarea	Actividades que componen a los planes de trabajo, planificadas por el usuario.
Descripción	Conjunto de ideas que caracterizan al plan de trabajo, brindando una mejor concepción del mismo.
Categoría	Definen el tipo de tarea por las categorías de formación, producción, investigación, postgrado y de extensión universitaria.

2.2. Requisitos funcionales del sistema

Los requisitos funcionales de un software definen qué es lo que el sistema debe hacer. Funcionalidades requeridas y restricciones que debe presentar el sistema que son aprobados en mutuo acuerdo con el usuario final. (Pressman, 2002)

A continuación se exponen los requisitos funcionales de la aplicación que se propone.

RF1: Autenticar Usuario.

Descripción: El sistema debe permitir introducir un usuario y una contraseña, de esta forma es validada la autenticidad de los datos haciendo uso del LDAP[12] a través del servicio web SOAP UCI, permitiendo el acceso a la aplicación en caso de ser válidos.

Entrada: Usuario y contraseña.

Salida: Valida el usuario y la contraseña con el servidor LDAP, haciendo uso del servicio web SOA UCI.

RF2: Listar planes de trabajo.

Descripción: El sistema debe permitir mostrar los planes de trabajo de un usuario determinado.

Entrada: Usuario.

Salida: Listado de planes de trabajo del usuario.

RF3: Adicionar un Plan de Trabajo.

Descripción: El sistema debe permitir crear un plan de trabajo nuevo con los datos definidos por el usuario.

Entrada: Título, Objetivo, Publicidad.

Salida: Plan de trabajo adicionado.

RF4: Modificar un Plan de Trabajo.

Descripción: El sistema debe permitir, dado un plan de trabajo, modificar sus datos.

Entrada: Título, Objetivo, Publicidad.

Salida: Plan de trabajo modificado.

RF5: Eliminar Plan de Trabajo.

Descripción: El sistema debe permitir eliminar un plan de trabajo seleccionado por el usuario.

Entrada: Listado de planes de trabajo.

Salida: Plan de trabajo eliminado de la lista.

[12] **LDAP** son las siglas de ***L****ightweight* ***D****irectory* ***A****ccess* ***P****rotocol* (en español *Protocolo Ligero/Simplificado de Acceso a Directorios*) que hacen referencia a un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

RF6: Visualizar plan de trabajo

Descripción: El sistema debe permitir mostrar las actividades de un plan de trabajo.

Entrada: Plan de trabajo.

Salida: Actividades del Plan de trabajo.

RF7: Importar tarea.

Descripción: El sistema debe permitir importar una tarea de un plan de trabajo público o del que sea considerado observador.

Entrada: Tarea del plan de trabajo público.

Salida: Tarea agregada al plan de trabajo personal.

RF8: Listar Tareas.

Descripción: El sistema debe permitir dado un plan de trabajo, listar todas las tareas que lo componen.

Entrada: Plan de trabajo.

Salida: Listado de tareas del plan de trabajo.

RF9: Adicionar tarea al plan de trabajo.

Descripción: El sistema debe permitir agregar una tarea a un plan de trabajo.

Entrada: Título, localización, fecha de inicio, fecha de fin, notificaciones, categoría, tipo de frecuencia, frecuencia, repetir hasta, día de la semana.

Salida: Tarea agregada.

RF10: Modificar tarea en el plan de trabajo.

Descripción: El sistema debe permitir modificar cualquier tarea del plan de trabajo.

Entrada: Título, localización, fecha de inicio, fecha de fin, notificaciones, categoría, tipo de frecuencia, frecuencia, repetir hasta, día de la semana.

Salida: Tarea modificada.

RF11: Eliminar tareas en el plan de trabajo.

Descripción: El sistema debe permitir eliminar cualquier tarea del plan de trabajo.

Entrada: Tarea a eliminar.

Salida: Tarea seleccionada eliminada.

RF12: Visualizar fecha en el calendario.

Descripción: El sistema debe permitir visualizar la fecha en un formato corto al estar minimizada la ventana del calendario, y en un formato largo al estar maximizada.

Entrada: Tarea seleccionada.

Salida: Ventana con el formato de fecha.

RF13: Mostrar cantidad de tareas en un día.

Descripción: El sistema permite visualizar la cantidad de tareas planificadas en un día determinado al estar minimizada la ventana, y al estar maximizada dicha ventana permite visualizar en una tabla las tareas de ese día.

Entrada: Día seleccionado.

Salida: Tareas.

RF14: Buscar tarea en un plan de trabajo.

Descripción: El sistema debe permitir buscar una tarea en un plan de trabajo.

Entrada: Tarea a buscar.

Salida: Resultado de la búsqueda.

RF15: Añadir Observador.

Descripción: El sistema debe permitir seleccionar los usuarios que considere observadores de su plan de trabajo.

Entrada: Usuarios a seleccionar.

Salida: Usuarios observadores del plan de trabajo.

RF16: Listar Observadores.

Descripción: El sistema debe permitir mostrar los usuarios que están considerados observadores de su plan de trabajo.

Entrada: Plan de trabajo.

Salida: Lista de observadores de un plan de trabajo.

RF17: Eliminar Observadores.

Descripción: El sistema debe permitir eliminar los usuarios que ya están considerados observadores de su plan de trabajo.

Entrada: Lista de observadores que se quieren eliminar.
Salida: Actualizar el estado de los observadores, después de haberse eliminado los deseados.

RF18: Generar Reporte

Descripción: El sistema debe permitir exportar en extensiones pdf, docx, xlsx, odt, ods, los reportes deseados por el usuario, donde se puede seleccionar un rango de fecha, una categoría y una plantilla predefinida, que puede ser un reporte del plan de trabajo por áreas o una evaluación del desempeño laboral.

Entrada: Fecha de inicio, fecha de fin, categoría, plantilla, formato.

Salida: Reportes en formato pdf, docx, xlsx, odt, ods.

2.3. Requisitos no Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener, características que lo hacen atractivo, usable, rápido o confiable. Normalmente están vinculados a requerimientos funcionales, es decir una vez se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con la toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. (Pressman, 2002)

Facilidad de uso:

RNF1: El sistema debe ser utilizado por cualquier usuario de la universidad con las siguientes características:

- ✓ Conocimientos básicos relativos al uso de una computadora.
- ✓ Conocimientos básicos del sistema operativo en el que se trabaje.

RNF2: El sistema debe contar con un Manual de Usuario, que será de ayuda en el entendimiento de las funcionalidades de la aplicación.

Portabilidad:

RNF3: El sistema debe ser multiplataforma (Windows 7 hasta Windows 8.1 y en GNU/Linux [13] Debian 8.0, Ubuntu 14.04 y Nova 4.0).

Disponibilidad:

RNF4: El sistema debe estar disponible para su uso las 24 horas del día a los usuarios autorizados, garantizando el acceso a la información en cualquier momento.

Seguridad:

RNF5: El sistema debe contar con acceso restringido a usuarios del dominio, estos serán validados dada la autenticidad de los datos haciendo uso del LDAP a través del servicio web SOAP UCI.

RNF6: La conexión entre la pc cliente y el servidor deberá ser mediante el protocolo seguro de transferencia de hipertexto HTTPS (por sus siglas en inglés, *Hypertext Transfer Protocol Secure*).

Interfaz:

RNF7: El sistema debe tener un diseño de interfaz estructurada de forma sencilla, confiable y nítida para no desviar la atención del usuario, con un color azul oscuro en las partes superiores e inferiores de la aplicación, y con un fondo blanco de forma general en la interfaz. Teniendo en cuenta las funcionalidades requeridas para la que será desarrollada, tendrá un área de navegación sencilla de interactuar, con la mayoría de las funcionalidades al alcance de un clic en la pantalla principal. Los iconos serán pequeños con unas dimensiones de 16 x 16 y otros de 24 x 24 píxeles, con colores verde claro, para una mejor visibilidad y uso del sistema.

Requerimientos de Software:

[13] GNU/Linux es uno de los términos empleados para referirse a la combinación del núcleo o kernel libre similar a Unix denominado Linux con el sistema GNU.

RNF8: El sistema sebe ser desplegado en versiones de Windows, desde Windows 7 hasta Windows 8.1, y con respecto a GNU/Linux Debian 8.0, Ubuntu 14.04 y Nova 4.0. Se recomienda como navegador Mozilla Firefox 37 y Google Chrome en su versión 40. Del lado del servidor NodeJS 0.10.24, y como gestor de base de datos PostgreSQL 9.3.

Requerimientos de Hardware:

RNF9: En la pc del cliente se requiere de una máquina con 512Mb de RAM como mínimo, el servidor de aplicaciones y de bases de datos deberá tener 2 GB de RAM y 50 GB de capacidad de almacenamiento en disco como mínimo, todas las máquinas implicadas en la funcionalidad de la aplicación deben estar conectadas a la red (poseer una tarjeta de red).

2.4. Patrones de Casos de Uso

Son comportamientos que deben existir en el sistema, ayudan a describir qué es lo que el sistema debe hacer, es decir, describen el uso del sistema y cómo este interactúa con los usuarios.(Larman, 2002) Estos patrones son utilizados generalmente como plantillas que describen cómo deberían ser estructurados y organizados los casos de uso. De esta forma, los patrones de casos de uso capturan mejores prácticas para modelar casos de uso, obteniéndose una manera ágil de resolver problemas que se presentan en el modelado de sistemas.

Los patrones de casos de uso utilizados en el desarrollo del sistema, fueron:

CRUD[14] Completo:

El patrón CRUD Completo consiste en un caso de uso para administrar la información (CRUD Información), permite modelar las diferentes operaciones para administrar una entidad de información, tales como crear, leer, cambiar y eliminar o dar de baja.

En la siguiente figura se representa este patrón, evidenciado a través de los casos de uso: *Gestionar Plan de Trabajo* y *Gestionar Tarea*.

[14] (CRUD) Su nombre es un acrónimo de las palabras en inglés Create, Read, Update, Delete.

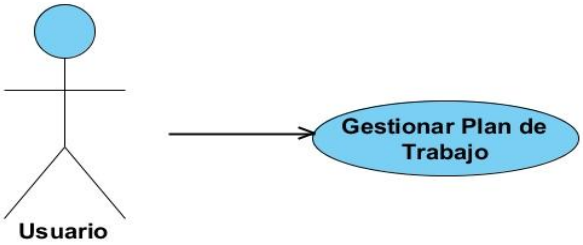


Fig. 3 Evidencia del patrón CRUD completo

CRUD Parcial:

Algunas de las alternativas del caso de uso, puede ser modelada como caso de uso independiente. Este patrón es preferible cuando uno de los flujos alternativos del caso de uso es más significativo, muy largo o mucho más complejo que el patrón completo.

En la siguiente figura se representa este patrón, evidenciado a través del caso de uso: *Administrar Observador*.

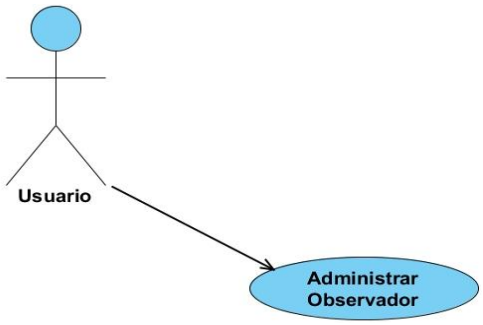


Fig. 4 Evidencia del patrón CRUD parcial

2.5. Modelo de caso de uso del sistema

El modelo de caso de uso ayuda al cliente, a los usuarios y a los desarrolladores a llegar a un acuerdo sobre los requisitos, es decir, sobre las condiciones y posibilidades que debe cumplir el sistema. El modelo de caso de uso sirve como acuerdo entre clientes y desarrolladores, y proporciona la entrada fundamental para el análisis, el diseño y las pruebas. (Jacobson, 2000)

A continuación se muestra el diagrama de casos de uso del sistema, en la figura 5.

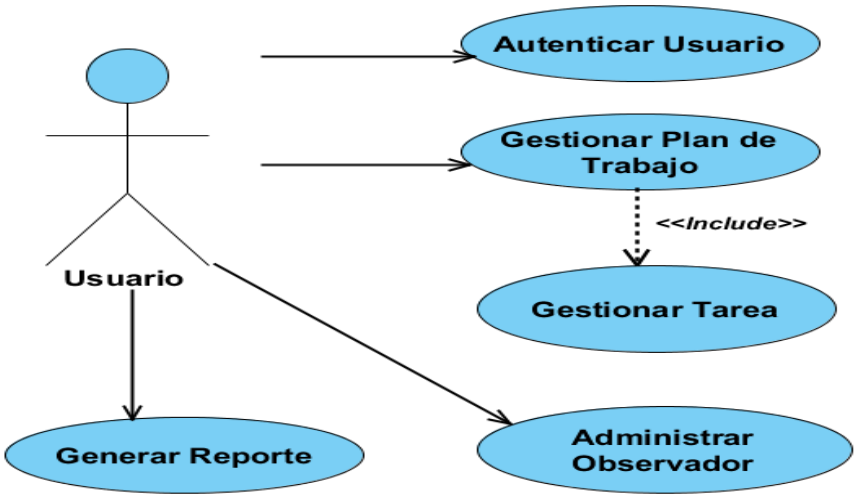


Fig. 5 Diagrama de Casos de Uso del Sistema

Autenticar Usuario: Permite a los usuarios del sistema autenticarse en el mismo.

Gestionar Plan de Trabajo: Permite las operaciones de Listar, Adicionar, Modificar, Eliminar y Visualizar los planes de trabajo en el sistema.

Gestionar Tarea: Permite las operaciones de Listar, Importar, Adicionar, Modificar, Eliminar y Buscar las tareas en el sistema.

Administrar Observador: Permite Adicionar y Eliminar a un observador en el sistema.

Generar Reporte: Permite generar los reportes del sistema.

2.6. Matriz de Trazabilidad:

La Matriz de Trazabilidad de Requisitos ayuda a realizar seguimiento a los requisitos a lo largo del ciclo de vida del proyecto para asegurar su cumplimiento de manera satisfactoria. Permite a través de una tabla cruzada, ubicar los requisitos funcionales del sistema, y agruparlos por casos de uso. En la tabla 2 se muestra la matriz de trazabilidad utilizada para representar los requisitos funcionales identificados contra los casos de uso del sistema a desarrollar.

Tabla 2. Matriz de Trazabilidad

RF/CU	Autenticar Usuario	Gestionar Plan de Trabajo	Gestionar Tarea	Administrar Observador	Generar Reporte
RF1	**X**				
RF2		**X**			

RF3		**X**			
RF4		**X**			
RF5		**X**			
RF6		**X**			
RF7			**X**		
RF8			**X**		
RF9			**X**		
RF10			**X**		
RF11			**X**		
RF12			**X**		
RF13			**X**		
RF14			**X**		
RF15				**X**	
RF16				**X**	
RF17				**X**	
RF18					**X**

2.7. Descripciones textuales de los Casos de Uso del Sistema

La descripción textual de un caso de uso, consiste en una descripción simple y consistente en cómo interactúan los actores y casos de uso con el sistema. Se centra en los comportamientos externos del sistema, ignorando el funcionamiento interno y utiliza los lenguajes y metodologías usadas por el cliente. En la tabla 3, se muestra un fragmento de la descripción textual del CU Gestionar Plan de Trabajo.

Para ver las restantes descripciones de los casos de uso, ver artefacto "*0114_Especificación_de_casos_de_uso*".

CU 2. Gestionar Plan de Trabajo

Tabla 3. Fragmento de la descripción del CU Gestionar Plan de Trabajo

Objetivo	Gestionar los planes de trabajo.
Actores	Usuario(Inicia)

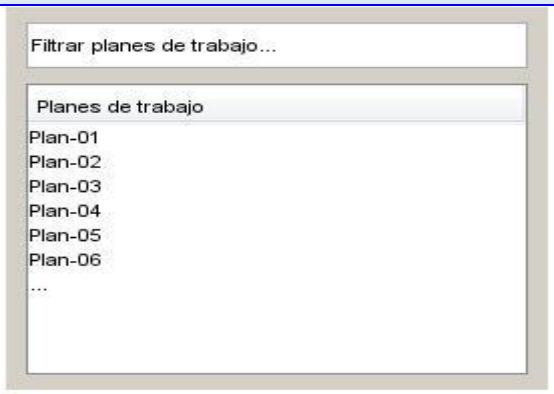
Resumen	El caso de uso se inicia cuando el usuario selecciona una de las siguientes opciones: **Adicionar:** Cuando el usuario desea crear un nuevo plan de trabajo, este introduce los datos necesario para su construcción y el sistema registra los mismos, finalizando así el CU. **Modificar:** El usuario selecciona el plan de trabajo a modificar, el sistema muestra los datos correspondientes, el usuario los modifica, finalizando así el CU. **Eliminar:** El usuario selecciona el plan de trabajo e indica eliminar, el sistema lo elimina, finalizando así el CU. **Listar:** El sistema muestra los planes de trabajo existentes. **Importar:** El usuario selecciona el plan de trabajo que desea importar, el sistema agrega el plan de trabajo al usuario, finalizando así el CU. **Visualizar:** El usuario selecciona un plan de trabajo de la lista, el sistema muestra el resultado en el calendario**,** finalizando así el CU.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Usuario autenticado en el sistema.
Postcondiciones	En dependencia de la acción del usuario: - Se crea un nuevo plan de trabajo. - Se listan los planes de trabajo. - Se modifica un plan de trabajo existente. - Se elimina un plan de trabajo existente. - Se visualizan los planes de trabajo.

Flujo de eventos

Flujo básico <Gestionar Plan de Trabajo>

	Actor	Sistema
1.	Selecciona gestionar un plan de trabajo	
2.		Permite: - Listar planes de trabajo, ver Sección 1: “Listar planes de trabajo”. - Adicionar plan de trabajo, ver Sección 2: “Adicionar un Plan de Trabajo”. - Modificar un plan de trabajo, ver Sección 3: “Modificar un Plan de Trabajo”. - Eliminar un plan de trabajo, ver Sección 4: “Eliminar

		Plan de Trabajo". - Visualizar el plan de trabajo de un día, ver Sección 5: "Visualizar plan de trabajo por día". - Visualizar el plan de trabajo de una semana, ver Sección 6: "Visualizar plan de trabajo por semana". - Visualizar el plan de trabajo de un mes, ver Sección 7: "Visualizar plan de trabajo por mes".
3.		Termina el caso de uso.
Sección 1: Listar planes de trabajo		
Flujo básico "Listar planes de trabajo"		
	Actor	**Sistema**
1.		Muestra los planes de trabajo del usuario en una lista.
Flujos alternos		
Nº Evento <Flujo alterno 1: En caso de no existir planes de trabajo creados>		
	Actor	**Sistema**
1.		Muestra una lista vacía.
Prototipo de Interfaz		



2.8. Modelo del Diseño

El Modelo de Diseño se utiliza para documentar el diseño de un sistema. Es un modelo de objeto que describe la realización de los casos de uso y sirve como una abstracción del Modelo de Implementación y del código fuente. Se utiliza como entrada esencial para las actividades en los flujos de trabajo

Implementación y Prueba. Es un artefacto integral que abarca todas las clases del diseño y sus relaciones, e incluye los diagramas de clases y de interacción del diseño.

2.8.1. Descripción de la arquitectura y el diseño

La arquitectura de software de un programa o sistema de cómputo es la estructura o las estructuras del sistema, que incluyen los componentes del *software*, las propiedades visibles externamente de los componentes y las relaciones entre ellos (Pressman, 2005). Es una representación que permite que un ingeniero de software analice la efectividad del diseño para cumplir con los requisitos establecidos, considere opciones arquitectónicas en una etapa en que aún resulta relativamente fácil hacer cambios al diseño y reduzca los riesgos asociados con la construcción del software.

Arquitectura

El modelo arquitectónico Cliente-Servidor es un modelo de sistema en el que dicho sistema se organiza como un conjunto de servicios y servidores asociados, más unos clientes que acceden y usan los servicios (Sommerville, 2005). Consiste en un cliente que envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio).

Fig. 6 Arquitectura Cliente-Servidor

Cliente: Permite al usuario formular los requisitos y pasarlos al servidor. Maneja todas las funciones relacionadas con la manipulación y despliegue de datos; por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario, además de acceder a los servicios distribuidos en cualquier parte de una red. Algunas de sus funciones son administrar la interfaz de usuario, interactuar con el usuario, procesar la lógica de la aplicación, hacer validaciones locales, generar requisitos de bases de datos, recibir resultados del servidor y formatear resultados.

Servidor: Encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. Las funciones del servidor son aceptar los requisitos de bases de datos que hacen los clientes, procesar esos requisitos, formatear datos para trasmitirlos a los clientes, procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

2.8.2. Patrones utilizados en la solución

Un patrón es una descripción de un problema y la solución, a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos. (Larman, 2002)

Muchos patrones proporcionan guías sobre el modo en el que deberían asignarse las responsabilidades a los objetos, dada una categoría específica del problema. En este epígrafe se señalan los patrones de caso de uso, arquitectónico y de diseño utilizados.

Patrones arquitectónicos

Los patrones arquitectónicos definen la estructura general del software, indican las relaciones entre los subsistemas y los componentes del software, además de definir las reglas para especificar las relaciones entre los elementos (clases, paquetes, componentes, subsistemas) de la arquitectura. (Pressman, 2005), proporcionando un conjunto de sub-sistemas predefinidos, especificando sus responsabilidades, reglas, directrices que determinan la organización, comunicación, interacción y relaciones entre ellos. Estos heredan muchas de las terminologías y conceptos de patrones de diseño, pero se centran en proporcionar modelos y métodos re-utilizables específicamente para la arquitectura general de los sistemas de información. En otras palabras quiere decir que a diferencia de los patrones de diseño estas son plantillas incompletas y no se pueden aplicar directamente al código con modificaciones meramente contextuales. Los patrones arquitectónicos a su vez se salen del código puro de la aplicación y suben e incluyen software, hardware, redes, inclusos las personas.

Modelo Vista Controlador

El patrón Modelo Vista Controlador (MVC) es un patrón de arquitectura de software encargado de separar la lógica de negocio de la interfaz del usuario y es el más utilizado en aplicaciones web, ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema, de forma simple y sencilla, a la vez que permite no mezclar lenguajes de programación en el mismo código.(Bahit, 2011)

Este separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes:

Modelo: El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Maneja la visualización de la información.

Controlador: Interpreta las acciones del mouse y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado. Es el intermediario entre la vista y el modelo. Es quien controla las interacciones del usuario solicitando los datos al modelo y entregándolos a la vista para que ésta, lo presente al usuario, de forma humanamente legible.

Entre las ventajas que lo caracterizan están las siguientes:

Soporte de vistas múltiples: Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente.

Adaptación al cambio: Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDA[15]. Con el uso de este patrón se persigue mejorar la reusabilidad y que las modificaciones en las vistas impacten en menor medida en la lógica de datos.

Patrones de Diseño

Un patrón de diseño describe una estructura de diseño que resuelve un problema particular dentro de un contexto específico y en medio de fuerzas que pueden tener un impacto en la manera en que se aplica y utiliza el patrón. Su finalidad es proporcionar una descripción que le permita al diseñador determinar si es aplicable al trabajo actual y si es reutilizable, brindando así ahorro en el tiempo del diseño. (Pressman, 2005) Son soluciones simples, elegantes, probadas y documentadas, a problemas específicos y comunes del diseño orientado a objetos.

[15] Un **PDA** (Personal Digital Assistant o Ayudante personal digital) es un dispositivo de pequeño tamaño que combina un ordenador, teléfono/fax, Internet y conexiones de red.

Entre los distintos patrones de diseño existentes, se utilizan en el desarrollo de la solución los que pertenecen a los GRASP(General Responsibility Asignment Software Patterns, en español Patrones Generales de Software para Asignar Responsabilidades), los cuales describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones.(Larman, 2002)

Dentro de los patrones GRASP son utilizados los siguientes:

- ✓ **Experto**

Problema: ¿Cuál es un principio general para asignar responsabilidades a los objetos?

Solución: Asignar una responsabilidad al experto en información (la clase que tiene la información necesaria para realizar la responsabilidad).

Este patrón es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo.

La clase polymita.layouts.MainBox cuenta con la información necesaria, para la visualización y control de todos los elementos referentes a un plan de trabajo, que es la responsabilidad que le corresponde en la implementación. Este es evidenciado a continuación en la figura 7.

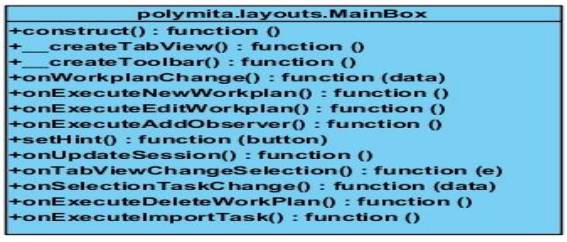


Fig. 7 Evidencia del patrón Experto en el diagrama de clases del diseño CU Gestionar Plan de Trabajo

- ✓ **Creador**

Problema: ¿Quién debería ser el responsable de la creación de una nueva instancia de alguna clase?

Solución: Asignarle a una clase la responsabilidad de crear una instancia de otra clase.

El uso de este patrón se evidencia a través de la figura 8, en la clase polymita.layouts.MainBox, la cual crea las instancias necesarias para conectarse con las clases del paquete modelo y el paquete controlador.

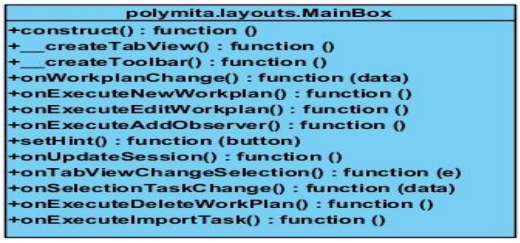


Fig. 8 Evidencia del patrón Creador en el diagrama de clases del diseño CU Gestionar Plan de Trabajo

- ✓ **Bajo Acoplamiento**

Problema: ¿Cómo soportar bajas dependencias, bajo impacto del cambio e incremento de la reutilización?

Solución: Asignar responsabilidades de forma tal que la inclusión de éstas no incremente el acoplamiento, lo que significa asignar una responsabilidad para mantener pocas dependencias entre las clases, potenciando así la reutilización de las mismas.

Se puede evidenciar el uso de este patrón entre las clases polymita.request.AbstractResource y polymita.mixins.MProperties puesto que MProperties tiene el número mínimo de dependencias con otras clases.

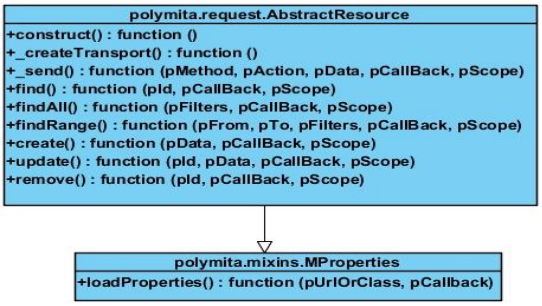


Fig. 9 Evidencia del patrón Bajo Acoplamiento en el diagrama de clases del diseño CU Gestionar Plan de Trabajo

- ✓ **Alta Cohesión**

Problema: ¿Cómo mantener la complejidad dentro de los límites manejables?

Solución: Asignar una responsabilidad de modo que la cohesión siga siendo alta, y no se sobrecarguen las funcionalidades de la clase.

Este patrón se evidencia en la relación que existe entre las clases polymita.request.AbstractResource y polymita.mixins.MProperties, ya que AbstractResource tiene funcionalidades que obtiene de MProperties y a su vez define otras que son necesarias en su implementación.

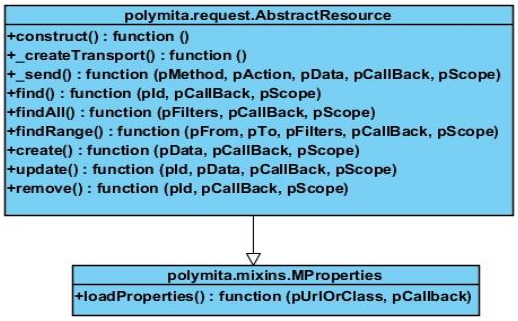


Fig. 10 Evidencia del patrón Alta Cohesión en el diagrama de clases del diseño CU Gestionar Plan de Trabajo

- ✓ **Controlador**

Problema: ¿Quién debe ser el responsable de gestionar un evento de entrada al sistema?

Solución: Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase.

El uso de este patrón se evidencia en la clase polymita.controllers.WorkPlan, al ser esta la clase encargada de obtener y manejar los datos referentes a los planes de trabajo del lado del servidor.

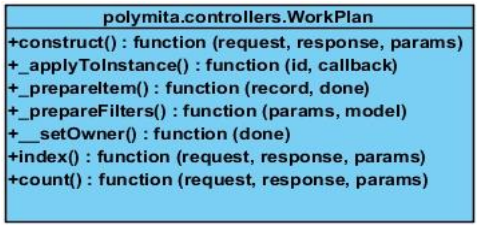


Fig. 11 Evidencia del patrón Controlador en el diagrama de clases del diseño CU Gestionar Plan de Trabajo

- ✓ **Polimorfismo**

Problema: ¿Cómo manejar las alternativas basadas en el tipo? ¿De qué manera crear componentes de software conectables?

Solución: Cuando por el tipo varían las alternativas o comportamientos afines, las responsabilidades del comportamiento se asignarán mediante operaciones polimórficas- a los tipos en que el comportamiento presenta variantes.

Al llevar a cabo una responsabilidad que dependa del tipo, se tiene que hacer uso del polimorfismo, cuando las alternativas o comportamientos relacionados varían según el tipo (clase), se asigna la responsabilidad para el comportamiento, utilizando operaciones polimórficas según los tipos para los que varía el comportamiento. Asigna el mismo nombre a servicios en diferentes objetos.

Este patrón es apreciado a través de polymita.request.AbstractResource, la cual incluye a las clases polymita.request.WorkPlan y polymita.request.Session encargadas de cumplir funciones diferentes, como es el caso de la autenticación de usuarios y el envío de datos a través de las clases controladoras.

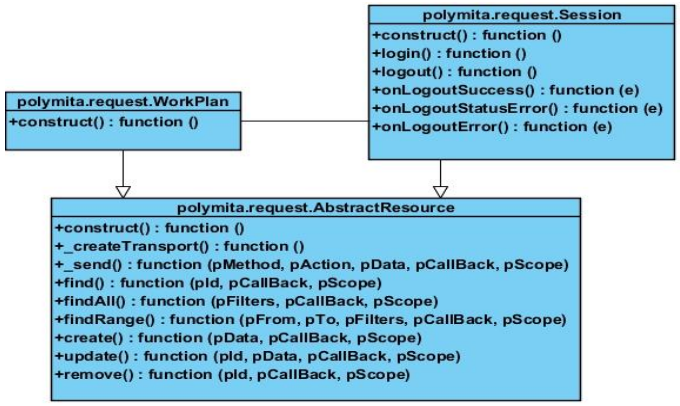


Fig. 12 Evidencia del patrón Polimorfismo en el diagrama de clases del diseño CU Gestionar Plan de Trabajo

Patrones GoF

- ✓ **Singleton (Instancia única):** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Este patrón se evidencia en la clase polymita.layouts.MainBox ya que en ella se crea una única instancia para cada clase controladora en la vista.

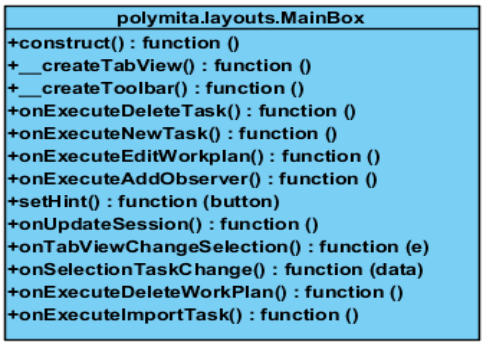


Fig. 13 Evidencia del patrón singleton (Instancia única) en el diagrama de clases del diseño CU Gestionar Plan de Trabajo

2.8.3. Diagramas de Clases del Diseño

Un diagrama de clases del diseño representa las especificaciones de las clases e interfaces en un software o aplicación. Compuesto generalmente por clases, asociaciones, atributos, interfaces con sus operadores y constantes, métodos, información acerca del tipo de atributo, navegabilidad y dependencias. (Larman, 2002)

Descripción de las clases relevantes del diseño

El diagrama representado en la figura 14, muestra la estructura de las clases del diseño, compuesta por una aplicación del lado del cliente y otra del lado del servidor, evidenciándose el uso de la arquitectura cliente-servidor. A continuación se representan los paquetes Polymita-GUI y Polymita-Server junto las relaciones con los subsistemas Qooxdoo y Guaraiba, que componen al diagrama de clases del diseño Gestionar Plan de Trabajo.

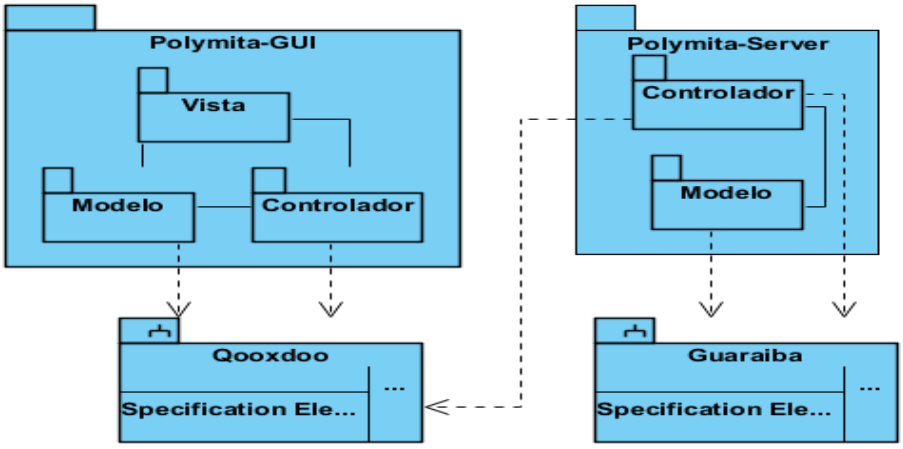


Fig. 14 Paquetes Cliente-Servidor del Diagrama de diseño CU Gestionar Plan de Trabajo.

A continuación, en la figura 15, se muestra el fragmento del diagrama de clases que pertenece al paquete Polymita-GUI, el cual está compuesto a su vez por los paquetes *Modelo, Vista y Controlador.* En el paquete Vista es donde se encuentran todas las clases relacionadas con la interfaz principal. La clase polymita.Aplication está compuesta por las clases polymita.layouts.LeftBox y polymita.layouts.MainBox. La clase polymita.layouts.LeftBox se comunica a través de la clase polymita.views.tables.Workplans con la clase polymita.model.WorkPlan contenida en el paquete *Modelo,* que a su vez se relaciona con la controladora *polymita.request.WorkPlan.* MainBox visualiza los formularios a través de la clase polymita.views.form.WorkPlan, y confirma las acciones con polymita.dialogs.Confirm que junto a WorkPlan se relacionan con polymita.request.WorkPlan. Esta clase WorkPlan perteneciente al paquete controlador, incluye a la clase polymita.request.AbstractResource y se relaciona con polymita.request.Session, donde polymita.request.Session se encarga de la autenticación en el sistema y AbstractResource de llamar a los servicios REST en dependencia de la operación, y crea el transporte hacia el controlador en el paquete del lado del servidor **Polymita-Server.**

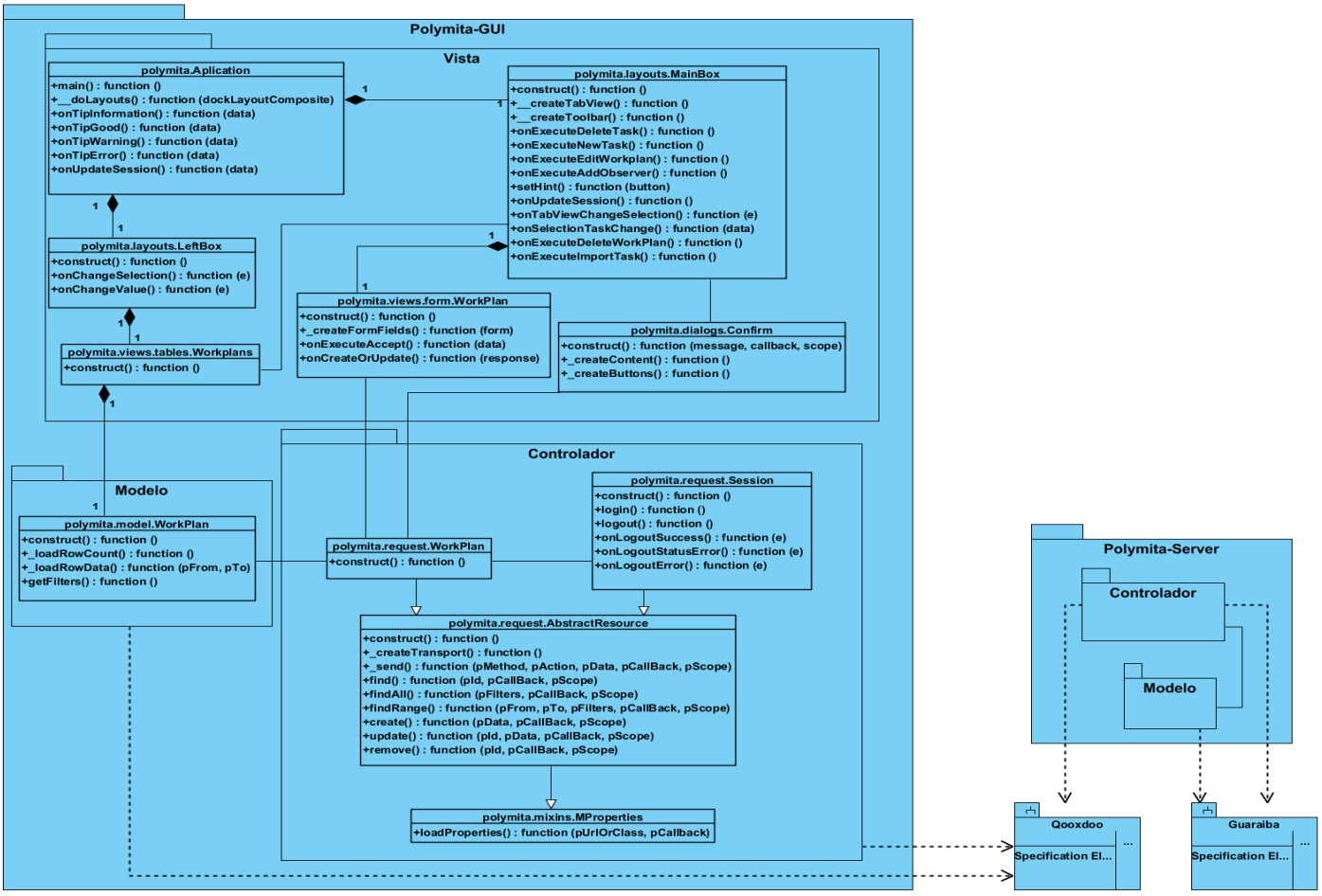


Fig. 15 Fragmento del Diagrama de Diseño del CU Gestionar Plan de Trabajo (Polymita-GUI).

De igual manera es mostrado en la figura 16 el fragmento del diagrama de clases que pertenece al paquete **Polymita-Server**, donde se encuentran las clases que pertenecen a la aplicación del lado del servidor, que cuenta con dos paquetes *Modelo* y *Controlador*. Dentro del paquete *Controlador* se encuentra la clase polymita.controllers.WorkPlan, que se relaciona con polymita.Router, polymita.Aplication y polymita.models.WorkPlan, esta última con la función de comunicarse con el paquete *Modelo* a través de la clase polymita.models.WorkPlan que obtiene los datos de la base de datos, que con la ayuda de las clases controladoras son enviados hacia la interfaz del sistema.

El resto de los diagramas de clases del diseño (Ver: Artefactos (Diagramas de Clases del Diseño)).

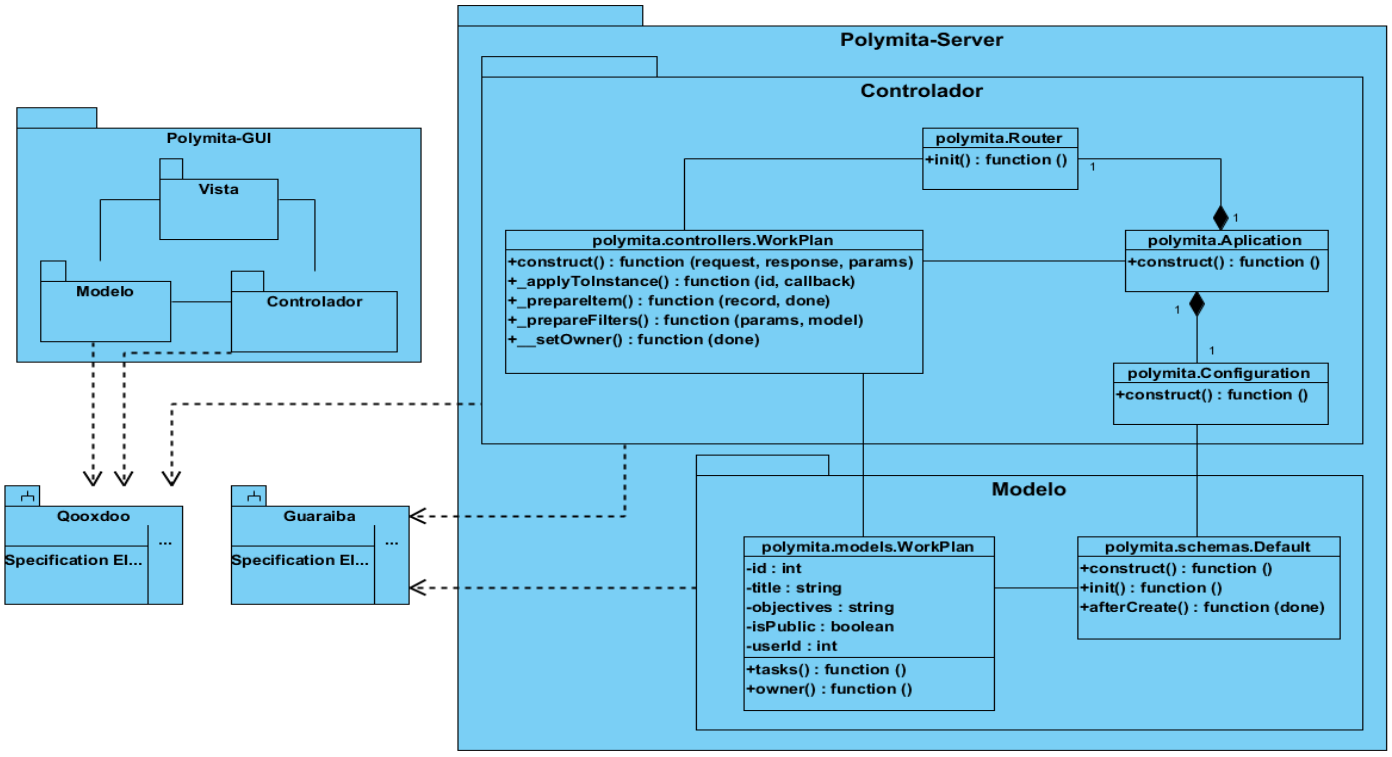


Fig. 16 Fragmento del Diagrama de Diseño del CU Gestionar Plan de Trabajo (Polymita-Server).

2.9. Diseño de la Base de Datos

El diseño de una base de datos (BD) consiste en definir la estructura de los datos que debe tener la BD para un sistema de información determinado. En el caso relacional, esta estructura será un conjunto de esquemas de relación con sus atributos, dominios de atributos, claves primarias y claves foráneas.

2.9.1. Modelo de Datos

Es considerado una herramienta útil tanto para ayudar al diseñador a reflejar en un modelo conceptual los requisitos del mundo real de interés como para comunicarse con el usuario final sobre el modelo conceptual obtenido y, de este modo, poder verificar si satisface sus requisitos.

Este es representado a través de los diagramas de entidad-relación (DER), los cuales permiten al ingeniero de software identificar objetos de datos y sus relaciones mediante una notación gráfica. En el contexto del análisis estructurado, el DER define todos los datos que se introducen, se almacenan, se transforman, y producen dentro de una aplicación. Los DER están compuestos por objetos de datos, atributos y relaciones. De forma general el modelado de datos y el diagrama entidad relación proporcionan

al analista una notación concisa para examinar datos dentro del contexto de una aplicación de procesamiento de datos. (Pressman, 2002)

En la figura 17 es representado el diseño de la base de datos del sistema, el cual está compuesto por las siguientes tablas:

workplan: encargada de almacenar los elementos de los planes de trabajo, esta se asocia con user y task_definition.

observer: existe a través de la relación mucho a mucho entre las tablas workplan y user.

user: encargada de registrar los datos de los usuarios.

task_definition: encargada de guardar la información correspondiente a una tarea, esta se relaciona con task_instance y category.

task_instance: encargada de almacenar la información referente a las repeticiones de las tareas.

category: encargada de guardar la información de las categorías.

report: encargada de registrar información de los reportes.

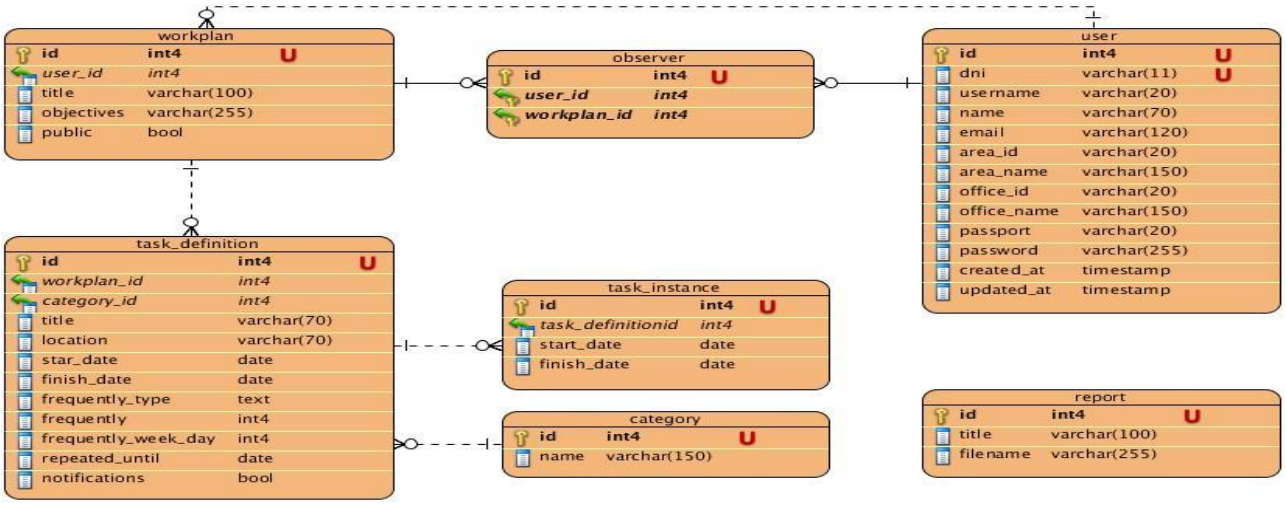


Fig. 17 Diagrama Entidad-Relación

Conclusiones del capítulo

En este capítulo es confeccionado el Modelo de Dominio, facilitando una mejor comprensión del proceso de planificación que se lleva a cabo en la universidad. Se obtiene una visión clara y objetiva de los

requisitos de software, identificándose un total de 18 requisitos funcionales y 9 no funcionales. Se describieron los elementos fundamentales de la arquitectura y el diseño, haciendo uso del patrón arquitectónico MVC, patrones de diseño GRASP y GoF, donde los GRASP identificados fueron el patrón creador, experto, bajo acoplamiento, alta cohesión, controlador, y polimorfismo; dentro de los GoF el patrón singleton (instancia única) y como patrones de casos de uso están el CRUD completo y el CRUD parcial. Se establecieron los componentes visuales, diagrama de casos de uso compuesto por cinco casos de uso, diagramas de clases del diseño, donde se diseñaron un total de 5 diagramas, y el diagrama de entidad-relación, que permitió conocer las 7 tablas que contienen toda la información del sistema.

CAPÍTULO 3: Implementación y Prueba

Introducción

En el presente capítulo se abordan los temas relacionados con los flujos de trabajo implementación y pruebas, en él se identifican los diferentes nodos del sistema y las relaciones entre ellos, mediante los diagramas de despliegue y de componentes. Además son evidenciados los estándares de codificación utilizados y los resultados obtenidos en las distintas pruebas funcionales realizadas al sistema, permitiendo así comprobar la funcionalidad y calidad de la aplicación.

3.1. Modelo de Implementación

El modelo de implementación describe cómo los elementos del modelo de diseño, las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, entre otros. También describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación o los lenguajes de programación utilizados, y cómo dependen los componentes unos de otros. (Jacobson 2000)

3.1.1. Diagrama de Despliegue

En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados. (Introducción a UML, 2010) De manera que se sitúa el software en el hardware que lo contiene. Son considerados los complementos de los diagramas de componentes que, unidos, proveen la vista de implementación del sistema.

Describen la topología del sistema, la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos, donde cada hardware es representado como un nodo.

El diagrama de despliegue representado en la figura 18, está compuesto por cuatro nodos que se relacionan a través de los protocolos HTTPS, TCP/IP y SOAP:

- **PC_cliente** (este nodo representa la computadora con que el usuario se conecta y trabaja con la aplicación).
- **Servidor NodeJS** (en este nodo es donde está desplegado o instalado el servidor de la aplicación).

- **Servidor LDAP** (este nodo representa el servidor LDAP utilizado por la UCI para obtener los datos de un usuario determinado).
- **Servidor PostgreSQL** (este nodo contiene el servidor de base datos con el que interactúa el sistema).
- **Protocolo HTTPS** (protocolo utilizado para la conexión entre las computadoras cliente y el servidor de aplicaciones).
- **Protocolo TCP/IP** (familia de protocolos utilizado para la conexión entre el servidor de aplicaciones y el servidor donde se encuentra ubicada la base de datos).
- **Protocolo SOAP:** (protocolo utilizado para establecer la conexión entre el servidor de aplicaciones y el servidor LDAP).

El proceso comienza cuando un usuario accede a la aplicación por una PC Cliente, esta se conecta a su vez con el servidor web NodeJS a través del protocolo HTTPS. Se procede a verificar si el usuario y contraseña introducidos son correctos, mediante una petición hecha al servidor LDAP utilizando el protocolo SOAP. Con estos datos personales el servidor NodeJS valida que exista el usuario en el sistema. Una vez validado el usuario, se procede a atender las peticiones del mismo, comunicándose para esto con la base de datos (BD PostgreSQL) mediante el protocolo TCP/IP de manera que los datos sean cifrados y viajen seguros por la red. En la siguiente figura se describen las relaciones entre cada nodo involucrado.

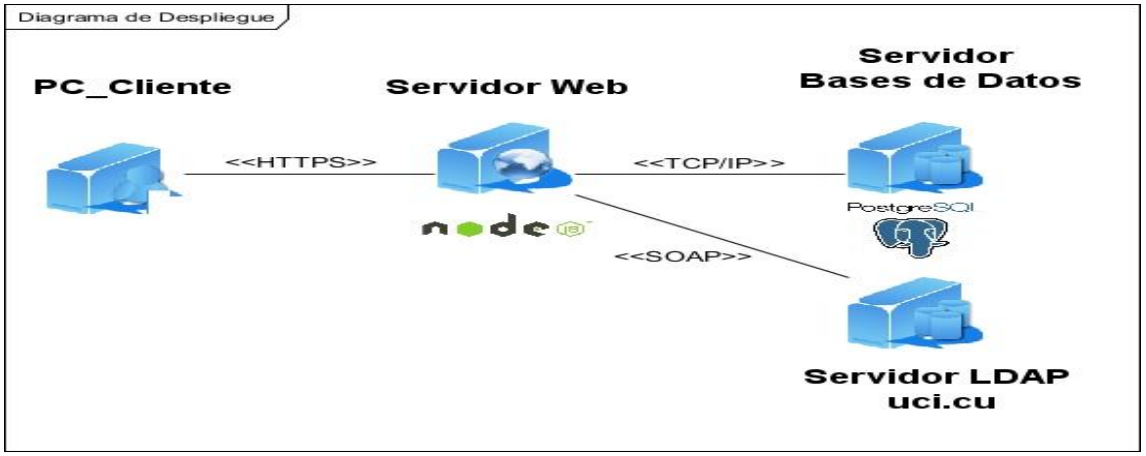


Fig. 18 Diagrama de Despliegue

3.1.2. Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos y sus relaciones en el entorno. Los componentes representan los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas, y las relaciones de dependencia se utilizan para indicar que un componente utiliza los servicios ofrecidos por otro componente. (Jacobson, 2000)

A continuación en la figura 19 se muestran los elementos que componen al diagrama de componentes asociado al caso de uso Gestionar Plan de Trabajo, con una estructura dividida en dos paquetes principales, **Polymita-GUI** y **Polymita-Server** que a su vez se comunican con los subsistemas Qooxdoo y Guaraiba, que representan los marcos de trabajo. Dentro de **Polymita-GUI** se encuentran los paquetes Modelo, Vista, Controlador que contienen a las clases del lado del cliente y dentro de **Polymita-Server** los paquetes Modelo y Controlador que contienen las clases del lado del servidor, que envían los datos de respuesta a las peticiones hechas del lado del cliente.

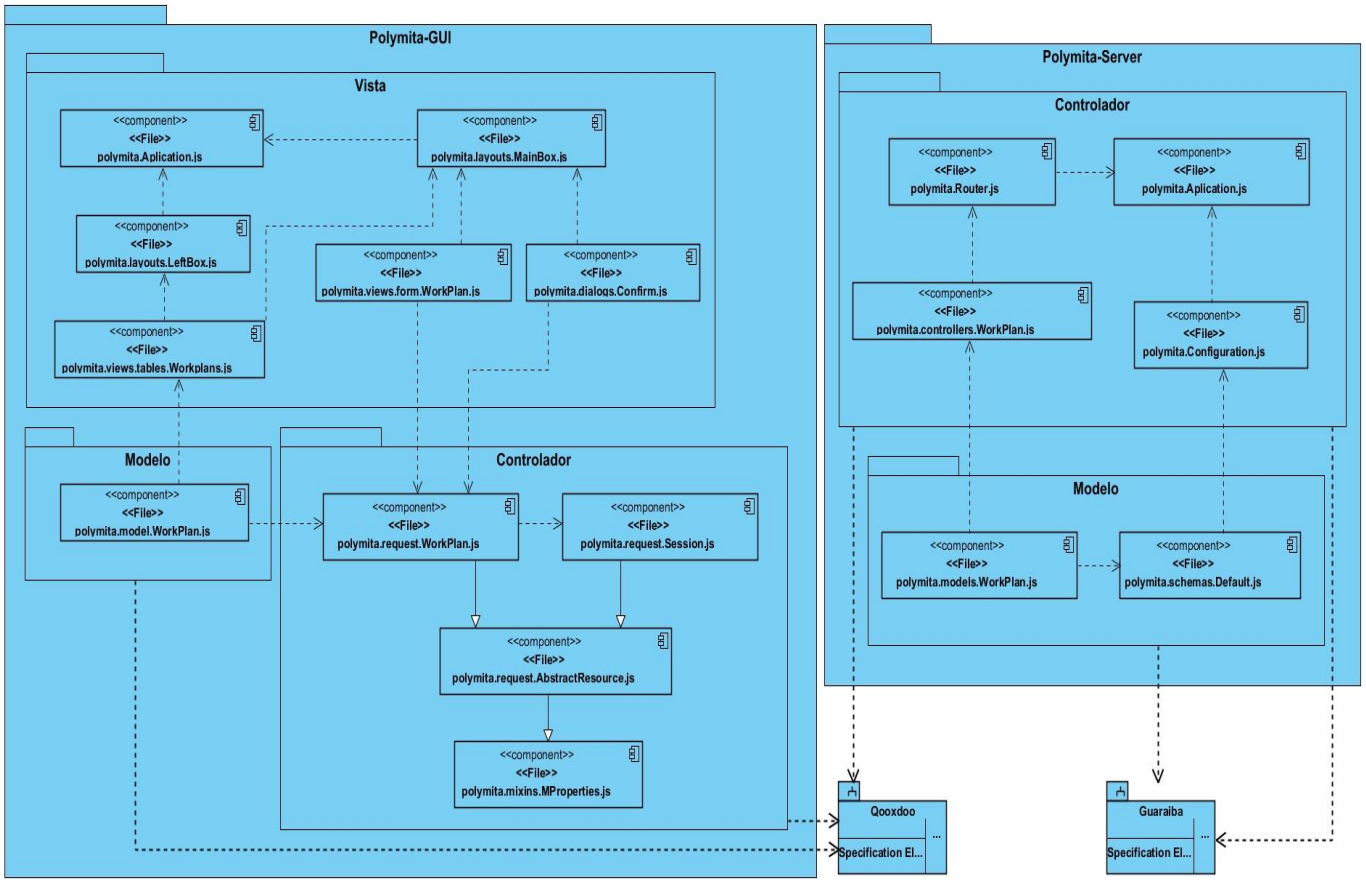


Fig. 19 Diagrama de Componentes del CU Gestionar Plan de Trabajo.

3.2. Estándares de codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Cuando el proyecto de software incorpore código fuente previo, o bien cuando realice el mantenimiento de un sistema de software creado anteriormente, el estándar de codificación debería establecer cómo operar con la base de código existente.

El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento. Dentro de los estándares de codificación existentes es seleccionado para el desarrollo de la aplicación el estándar ***CamelCase***.

CamelCase es un estilo de escritura que se aplica a frases o palabras compuestas, donde se utilizan las mayúsculas y minúsculas dentro de una misma expresión. De este estilo existen dos tipos:

UpperCamelCase: cuando la primera letra de cada una de las palabras es mayúscula.

lowerCamelCase: cuando la primera letra de cada una de las palabras es mayúscula exceptuando la primera.

Evidencias del estilo utilizado:

- Los nombres de las clases están escritos con el estándar *UpperCamelCase*.

```
qx.Class.define("polymita.dialogs.AbstractDialog", {
    type: "abstract",
    extend: qx.ui.window.Window,

    /**
     * Constructor
     *
     * @param caption {String} The caption text.
     * @param icon {String} The URL of the caption bar icon.
     */
    construct: function (caption, icon) {
        this.base(arguments, caption, icon);
```

Fig. 20 Nombre de las clases usando el estándar UpperCamelCase

- Los nombres de las funciones están escritos con el estándar *lowerCamelCase*.

```
        },

        onExecuteAccept: function (data) {
            var request = new polymita.request.Workplan();

            if (!data.id) {
                request.create(data, this.onCreateOrUpdate, this);
            } else {
                request.update(data.id, data, this.onCreateOrUpdate, this);
            }
        },

        onCreateOrUpdate: function (response) {
            if (response.successful) {
                q.messaging.emit("Application", "good", this.tr("Operation successful"));
                q.messaging.emit("Application", "update-workplans");

                this.close();
            } else {

            }
        }
    }
});
```

Fig. 21 Nombre de las funciones usando el estándar lowerCamelCase

- Los comentarios de implementación están delimitados por /*...*/ o //.

```
/**
 * Model class
 */
qx.Class.define("polymita.models.User", {
    extend: guaraiba.orm.Record,
```

Fig. 22 Comentarios en la implementación

- El código estará encerrado entre llaves ´{ }´, terminando la instrucción con la última llave ´}´ y coma ‘,’ para delimitar una función de otra.
- **Underscores** es usado en la nomenclatura de los atributos en la base de datos, para separar una palabra de otra.

task_definition		
id	int4	U
workplan_id	int4	
category_id	int4	
title	varchar(70)	
location	varchar(70)	
star_date	date	
finish_date	date	
frequently_type	text	
frequently	int4	
frequently_week_day	int4	
repeated_until	date	
notifications	bool	

Fig. 23 Nombre de los atributos usando el estándar Underscores

3.3. Pantallas principales de la aplicación.

Al acceder al sistema se muestra la interfaz principal con el formulario de autenticación, donde el usuario debe registrarse con sus credenciales del dominio, representado en la figura 24. Una vez registrado se tiene acceso a todas las funcionalidades que brinda el sistema, evidenciado en la figura 25, donde en la parte superior izquierda se visualiza el nombre de la aplicación y en la parte superior derecha se muestra el nombre del usuario autenticado seguido del botón cerrar sesión, en la columna izquierda se muestra un calendario donde se puede seleccionar los días de la semana, del mes y del año, debajo se encuentra una tabla con el listado de planes de trabajo del usuario registrado, la cual muestra el tipo de plan y el nombre del mismo, además un campo de búsqueda el cual permite filtrar los planes de trabajo existentes por título. En el panel superior se encuentran las funcionalidades relacionadas con los planes de trabajo, actividades, observadores, y los reportes, además de contar con un icono de ayuda en la parte derecha del panel, que da acceso al manual de usuario del sistema. También cuenta con un segundo nivel en la parte superior, donde están las diferentes vistas que sirven para mostrar las actividades del plan de trabajo, ya sea por día, semana, mes, listado de actividades e información del plan de trabajo.

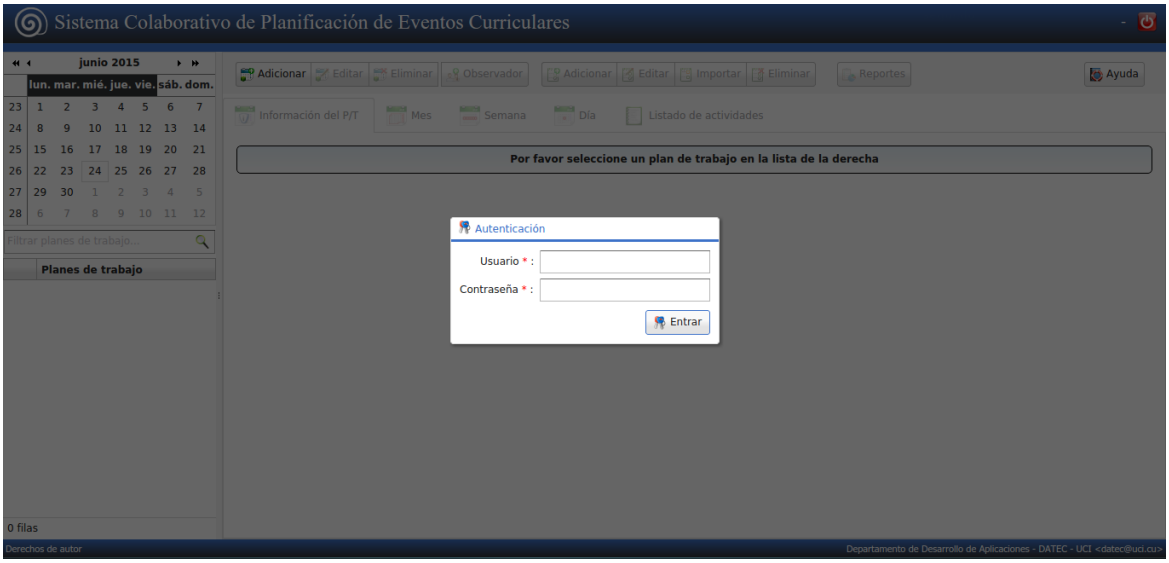


Fig. 24 Formulario de autenticación

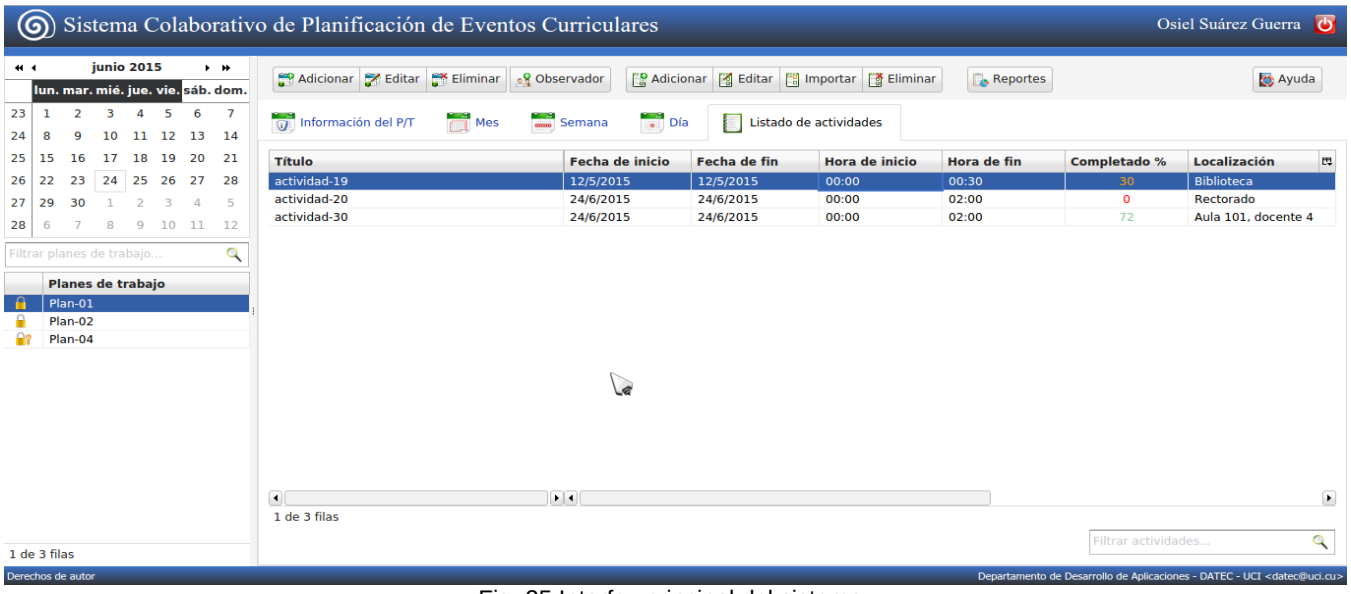


Fig. 25 Interfaz principal del sistema

3.4. Pruebas de Software

Una vez generado el código fuente, el software debe ser probado para descubrir (y corregir) el máximo de errores posibles antes de su entrega al cliente. El objetivo es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores. Es aquí donde se aplican las técnicas de prueba del software, facilitando una guía sistemática que comprueben la lógica interna de los componentes de software, y verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, comportamiento y desempeño.

“La prueba de software es el conjunto de actividades que se planean con anticipación y se realizan de manera sistemática. Por tanto es necesario definir un conjunto de pasos en que se puedan incluir técnicas y métodos específicos del diseño de casos de prueba. Las pruebas son el último bastión para la evaluación de la calidad y de manera más pragmática, el descubrimiento de errores” (Pressman, 2005)

3.4.1. *Niveles de Prueba*

En el momento de evaluar dinámicamente un sistema de software se debe comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el sistema en su conjunto. Los niveles de prueba son diferentes ángulos de verificar y validar en determinados momentos el ciclo de vida del software. Son aplicados durante todo el ciclo de desarrollo del software para diferentes objetivos y en distintos escenarios.

Serán utilizadas las pruebas a nivel de Desarrollador y de Aceptación, la cuales estarán diseñadas e implementadas por el equipo de desarrollo. De esta forma se obtendrán resultados que validen el correcto funcionamiento de los requisitos identificados.

- ✓ **Pruebas de Desarrollador**

Es la prueba diseñada e implementada por el equipo de desarrollo. Tradicionalmente estas pruebas han sido consideradas solo para las pruebas de unidad, aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración.

- ✓ **Pruebas de Aceptación**

Son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales. Estas son realizadas en dos etapas, una llamada *alpha*, en la cual los usuarios prueban el sistema usando datos de prueba, y la segunda llamada *beta* en la cual los usuarios empiezan a utilizar el sistema con los datos reales, pero siendo aún monitoreados cuidadosamente previendo que se presenten errores.

Es muy recomendable que las pruebas de aceptación se realicen en el entorno en que se va a usar el sistema, incluyendo el personal que lo utilizará. En caso de un producto de interés general, se realizan pruebas con varios usuarios que reportarán sus valoraciones sobre el producto.

3.4.2. *Tipos de Prueba*

Existen diferentes tipos de pruebas que se pueden aplicar para verificar el correcto funcionamiento de los requisitos identificados durante el proceso de análisis. Los tipos de prueba a utilizar serán las funcionales y de aceptación, a través de estas se validará el correcto funcionamiento e implementación de los requisitos funcionales del sistema de planificación y control de eventos curriculares.

- ✓ **Pruebas Funcionales**

Mediante las pruebas funcionales se valida el cumplimiento de las aplicaciones desarrolladas con respecto a los requisitos identificados, para de esta forma reducir los defectos. Este se basa fundamentalmente en el análisis de los datos de entrada y salida, esto generalmente se define en los casos de prueba definidos antes del inicio de las pruebas.

3.4.3. *Métodos y Técnica de Prueba*

Los métodos de pruebas no son más que una guía de procedimientos para realizar una prueba de software, tienen como objetivo proporcionar una vía más fácil para buscar errores garantizando la obtención de un producto de mayor calidad. Tradicionalmente se dividen en prueba de caja blanca y de prueba de caja negra, esta última se basa en las pruebas funcionales que pretenden demostrar que el software es operativo, que las entradas se aceptan y las salidas se producen de forma correcta. Para validar el correcto funcionamiento del sistema es utilizado el método de caja negra.

- ✓ **Caja negra**

Las pruebas de caja negra son las que se aplican a la interfaz del software. Una prueba de este tipo examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software. Se concentran en los requisitos funcionales del software, permitiendo derivar conjuntos de condiciones de entrada que ejercitaran por completo todos los requisitos funcionales de un programa. (Pressman, 2005) Estas pruebas tratan de encontrar errores en las categorías de funciones incorrectas o faltantes, de interfaz, en la estructura de datos o en el acceso a bases de datos externas, de comportamiento o desempeño y de inicialización y término.

Técnica de Prueba

- ✓ **Partición Equivalente:**

Este método de prueba de caja negra divide el dominio de entrada de un programa en clases de datos, a partir de las cuales deriva los casos de prueba. Este se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de pruebas que deben desarrollarse. El diseño de casos de prueba para partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. (Pressman, 2005)

3.5. Diseño de Caso de prueba

Esta es la fase de prueba donde se comprueban las funcionalidades que han sido implementadas, para de esta forma cumplir con las necesidades del cliente, haciendo uso de los casos de pruebas partiendo de la técnica antes señalada.

Los casos de pruebas se diseñaron partiendo de la descripción de los casos de usos. A continuación se muestra la tabla 4, con la descripción de las variables que se encuentran asociadas al caso de uso *Gestionar Plan de Trabajo.*

El resto de los casos de prueba se pueden ver en: Artefactos (Diseño de casos de prueba).

Tabla 4. Descripción de las variables para el caso de prueba Adicionar Plan de Trabajo.

No.	Nombre de campo	Clasificación	Valor nulo	Descripción
1.	Título	Campo de texto	No	Campo alfanumérico con una combinación de mayúsculas y minúsculas.
2.	Objetivo	Campo de texto	No	Campo alfanumérico con una combinación de mayúsculas y minúsculas.
3.	Público	Campo de selección	N/A	N/A

Tabla 5. Secciones del CU Gestionar Plan de Trabajo.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC 1. Adicionar Plan de Trabajo.	EC 1.1 Adicionar Plan de Trabajo correctamente.	Este es el escenario ideal donde todos los datos de las variables son correctos y consecuentemente es exitosa la ejecución de la prueba, donde se adiciona un plan de trabajo.
	EC 1.2 Adicionar Plan de Trabajo con campos vacíos.	Este es el escenario inválido donde los campos de las variables Título u Objetivo están vacíos y consecuentemente es errónea la ejecución de la prueba.
SC 2. Modificar Plan de Trabajo.	EC 2.1 Modificar Plan de Trabajo correctamente.	Este es el escenario ideal donde todos los datos de las variables son correctos y consecuentemente es exitosa la ejecución de la prueba, donde se modifica un plan de trabajo.
	EC 2.2 Modificar Plan de Trabajo con campos vacíos.	Este es el escenario inválido donde los campos de las variables Título u Objetivo están vacíos y consecuentemente es errónea la ejecución de la prueba.

SC 3. Buscar Plan de Trabajo.	EC 3.1 Buscar Plan de Trabajo correctamente.	Este es el escenario ideal donde todos los datos de las variables son correctos y consecuentemente es exitosa la ejecución de la prueba, donde se muestran un plan de trabajo.
SC 4. Listar Plan de Trabajo.	EC 4.1 Listar Planes de Trabajo correctamente.	Este es el escenario ideal donde todos los datos de las variables son correctos y consecuentemente es exitosa la ejecución de la prueba.
SC 5. Eliminar Plan de Trabajo.	EC 5.1. Eliminar Plan de Trabajo correctamente.	Este es el escenario ideal donde todos los datos de las variables son correctos y consecuentemente es exitosa la ejecución de la prueba, donde se elimina un plan de trabajo del sistema.

Se realizó una matriz de datos, donde se evaluó y probó la validez de cada uno de los valores introducidos en el sistema. Para ello se utilizó un juego de datos válidos y otros inválidos, empleando la técnica de partición equivalente.

A continuación un ejemplo de la matriz de datos del caso de uso Gestionar Plan de Trabajo, de la SC1. Adicionar Plan de Trabajo.

Las celdas de la tabla 6 contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

Tabla 6. Matriz de Datos

Escenario	**Variables**			**Respuesta del sistema**	**Flujo central**
	1	**2**	**3**		
EC 1.1 Adicionar	V	V	N/A	Una vez llenado todos los campos	1. Seleccionar el botón

Plan de Trabajo correctamente.	Plan1	Actividades del mes		correctamente y aceptado se muestra un mensaje: “Operación satisfactoria".	Adicionar plan de trabajo. 2. Introducir el título, objetivo, y escoger la publicidad. 3. Seleccionar el botón Aceptar. 4. Se muestra un mensaje "Operación satisfactoria".
EC 1.2 Adicionar Plan de Trabajo con campos vacíos.	V	I	N/A	El sistema emite un mensaje "Este cuadro es obligatorio", y sombrea de rojo el campo vacío.	1. Seleccionar el botón Adicionar plan de trabajo. 2. Introducir el título, objetivo, y escoger la publicidad. 3. Seleccionar el botón Aceptar. 4. Se introduce un Título u Objetivo, y al dejar un campo vacío el sistema emite un mensaje de alerta "Este cuadro es obligatorio", y sombrea de rojo el campo vacío.
	Plan2				

3.6. Resultado de las pruebas

Caja Negra

Las pruebas realizadas se llevaron a cabo en dos iteraciones, estas pruebas permitieron detectar un grupo de no conformidades las cuales fueron erradicadas. Fueron detectados en las iteraciones errores en la interfaz, de ortografía y de validación. En la primera iteración fueron detectadas 7 no conformidades, 6 de tipo interfaz y una de ortografía, para un total de 14 requisitos revisados; en la segunda iteración se detectaron 3 no conformidades, 2 de interfaz y una de ortografía, para un total de 18 requisitos revisados; y en la última iteración se revisaron 18 requisitos funcionales, donde no se encontraron no conformidades, cumpliendo de esta forma con los requisitos funcionales expuestos y la calidad de la solución implementada.

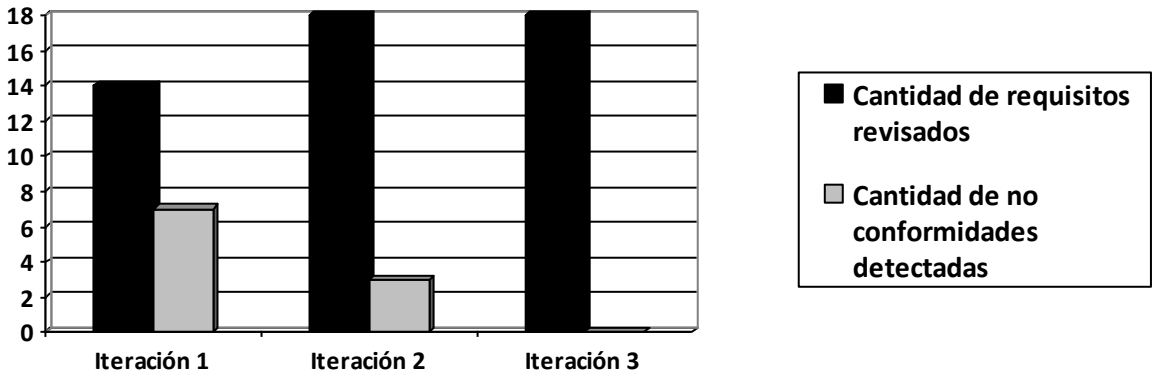


Fig. 26 Relación de no conformidades por requisitos revisados.

Prueba de Aceptación Alpha

Se realizó en conjunto con el usuario, que en este caso fue el Jefe de Departamento de Componentes en DATEC, donde emitió la satisfacción con respecto al Sistema de planificación y control de eventos curriculares desarrollado. Se confeccionó una carta de aceptación que acredita dichos elementos. (Ver Anexo 3)

Conclusiones del capítulo

En este capítulo es modelado el diagrama de despliegue, obteniéndose una visión de la distribución del sistema físicamente, quedando conformado por cuatro nodos relacionados a través de los protocolos HTTPS, TCP/IP y SOAP; de igual forma es representado el diagrama de componentes, donde se representaron los elementos que intervienen en el sistema y la relación que existen entre ellos, que permitió agilizar y organizar la implementación del sistema. Son especificados los estilos de codificación utilizados y los estándares de codificación empleados en la implementación de la solución, lográndose obtener una mayor estructura y organización en el código generado. Se efectuaron las pruebas de caja negra y de aceptación, permitiendo demostrar que la solución cumple con todas las especificaciones dadas por el cliente, en las que fueron detectadas un total de 10 no conformidades, ocho de ellas de interfaz y dos de ortografía, por lo que se hizo necesaria la realización de tres iteraciones, donde se corrigieron de forma correcta y se lograron los objetivos propuestos.

Conclusiones generales

Con la presente investigación se arribaron a las siguientes conclusiones:

- ✓ El empleo de métodos científicos facilitó conocer el estado del arte, haciéndose un análisis de soluciones similares que sirvieron de antecedentes para sentar las bases teóricas que apoyan la solución propuesta.
- ✓ Se determinó el modelo de diseño, que proporcionó una entrada al proceso de implementación, permitiendo una descripción más detallada de las clases a implementar y sus relaciones, bajo una arquitectura Modelo Vista Controlador haciendo uso de los patrones de diseño.
- ✓ La realización de pruebas sobre el sistema permitió obtener una aplicación con un correcto funcionamiento de la lógica interna del programa, o sea, que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.
- ✓ Además se pudo comprobar los valores de entrada y salida de la aplicación, analizar su comportamiento y detectar y corregir errores en su función y desempeño.
- ✓ Se obtuvieron resultados que validaron el nivel de facilidad de uso de la aplicación en el proceso de planificación para el personal de la Universidad de las Ciencias Informáticas.

Recomendaciones

- ✓ Continuar el desarrollo de la herramienta con el objetivo de incorporar en la vista de las actividades por semana, una lista de tareas por día.

Referencias Bibliográficas

ALVAREZ, M.A. 2010. Programación en JavaScript. *DesarrolloWeb.com* [en línea]. [Consulta: 17 junio 2015]. Disponible en: http://www.desarrolloweb.com/manuales/20/.

BAHIT, E. 2011. *POO y MVC en PHP* [en línea]. S.l.: Creative Commons. Disponible en: http://www.bubok.es/libros/205199/POO-y-MVC-en-PHP.

BARRIOS, C.J.H. 2014. Metodología open up ágil y tradicional. [en línea]. [Consulta: 9 abril 2015]. Disponible en: http://es.slideshare.net/carmeloh2/metodologa-open-up-39321348.

ESCOBEDO, F.J. 2014. Elementos y etapas del proceso de la organización. *Fabiola Jimenez Escobedo* [en línea]. [Consulta: 21 junio 2015]. Disponible en: https://fabyjimenez.wordpress.com/.

GÁMEZ, A.M. 2012. Introducción a NodeJS. [en línea]. [Consulta: 15 abril 2015]. Disponible en: http://www.desarrolloweb.com/articulos/intro-nodejs.html.

HEFFELFINGER, D.R., 2006. *JasperReports for Java Developers*. 2006. S.l.: Packt Publishing.

Introducción a UML. [en línea] 2010. [Consulta: 19 abril 2015]. Disponible en: http://programacion.net/articulo/introduccion_a_uml_181/7.

JACOBSON, I. 2000. *El Proceso Unificado de Desarrollo de Software* [en línea]. Madrid: Series Editors. ISBN 84-7829-036-2. Disponible en: http://sunshine.prod.uci.cu/gridfs/sunshine/books/El_Proceso_Unificado_de_Desarrollo_de_Software.pdf.

LARMAN, C. 2002. *UML y Patrones* [en línea]. 2da. S.l.: PEARSON EDUCACIÓN, S.A. ISBN 978-84-832-2927-9. Disponible en: http://gravepa.com/granaino/biblioteca/aprende/UNED%20-%20Grado%20Inform%C3%A1tica%20-/Extras%20%26%20Ediciones%20Antiguas/Una%20introduccion%20al%20analisis%20y%20diseno%20orientado%20a%20objetos%20y%20al%20proceso%20unificado.pdf.

Lenguajes De Programación Que Son Tipos Usos. [en línea] 2010. [Consulta: 26 marzo 2015]. Disponible en: http://www.areatecnologia.com/informatica/lenguajes-de-programacion.html.

MARTÍNEZ, R. 2010. Sobre PostgreSQL. [en línea]. [Consulta: 15 abril 2015]. Disponible en: http://www.postgresql.org.es/sobre_postgresql.

NetBeans. [en línea] 2012. [Consulta: 15 abril 2015]. Disponible en: https://netbeans.org/index_es.html.

O'BRIEN, MARAKAS, J.A., George M. 2006. *SISTEMAS DE INFORMACIÓN GERENCIAL* [en línea]. Séptima edición. México, D. F.: McGraw-Hill. [Consulta: 3 junio 2015]. ISBN 970-10-5630-2. Disponible en: http://www.academia.edu/8304620/SISTEMAS_DE_INFORMACI%C3%93N_GERENCIAL_SISTEMAS_DE_INFORMACI%C3%93N_GERENCIAL_S%C3%89PTIMA_EDICI%C3%93N.

PÉREZ, J.E. 2009. *Introducción a JavaScript* [en línea]. S.l.: s.n. Disponible en: http://www.librosweb.es/javascript.

pgAdmin3. [en línea] 2009. [Consulta: 15 abril 2015]. Disponible en: http://www.pgadmin.org/.

PRESSMAN, R.S. 2002. *Ingeniería del Software. Un enfoque práctico.* 5ta. Edición. S.l.: McGraw-Hill. ISBN 84-481-3214-9.

PRESSMAN, R.S. 2005. *Ingeniería del Software, un enfoque práctico* [en línea]. Sexta Edición. S.l.: Mcgraw-Hill. 9789701054734. ISBN 9701054733. Disponible en: http://www.mhhe.com/engcs/pressman/.

qooxdoo » Home. [en línea] 2014. [Consulta: 1 abril 2015]. Disponible en: http://qooxdoo.org/.

servidor web. [en línea] 2008. [Consulta: 15 abril 2015]. Disponible en: http://www.cristalab.com/tutoriales/crear-tu-propio-servidor-web-4.-apache-php-y-mysql-c51133l/.

UNIVERSIDAD DE MURCIA 2011. Ingeniería del software, metodologías de desarrollo. [en línea]. [Consulta: 9 abril 2015]. Disponible en: http://www.um.es/docencia/barzana/IAGP/IAGP2-Metodologias-de-desarrollo.html.

Anexos

Anexo 1. Entrevista

Nombre:	Glennis
Apellidos:	Tamayo Morales
Cargo:	Jefe de Departamento Desarrollo Componentes (DATEC)
Preguntas	

1. ¿Cómo se lleva a cabo el actual proceso de planificación de los planes de trabajo?
2. ¿Cuáles son las principales deficiencias que afectan dicho proceso?
3. ¿Cuáles son los motivos de los retrasos en la entrega de los planes de trabajo?
4. ¿De qué forma son entregados los planes de trabajo al directivo que los evalúa?
5. ¿Cómo se lleva a cabo el proceso de evaluación de los planes de trabajo en el departamento?
6. ¿Cómo se obtienen los elementos para la evaluación del desempeño laboral?

Anexo 2. Gráficas de los resultados a las pruebas realizadas con el JMeter a los lenguajes Node, Node Cluster, Java y PHP.

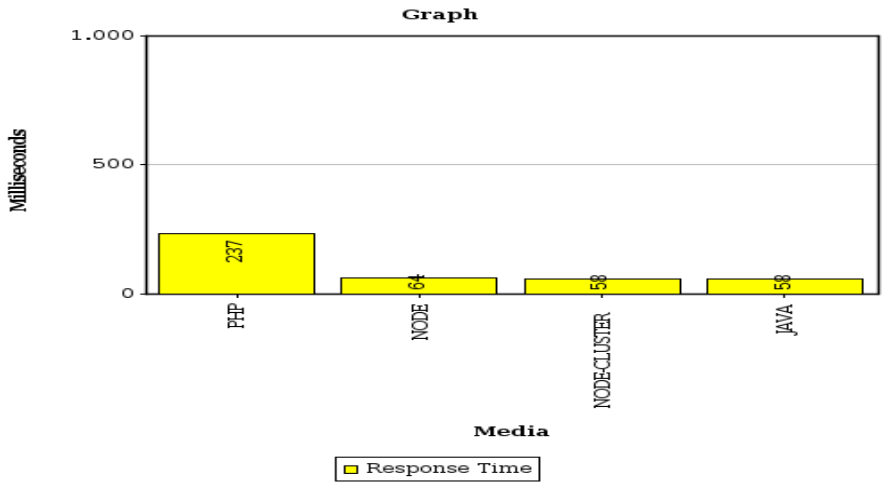


Fig. 27 Tiempo de respuesta en milisegundos con 8H-125C-1000M.

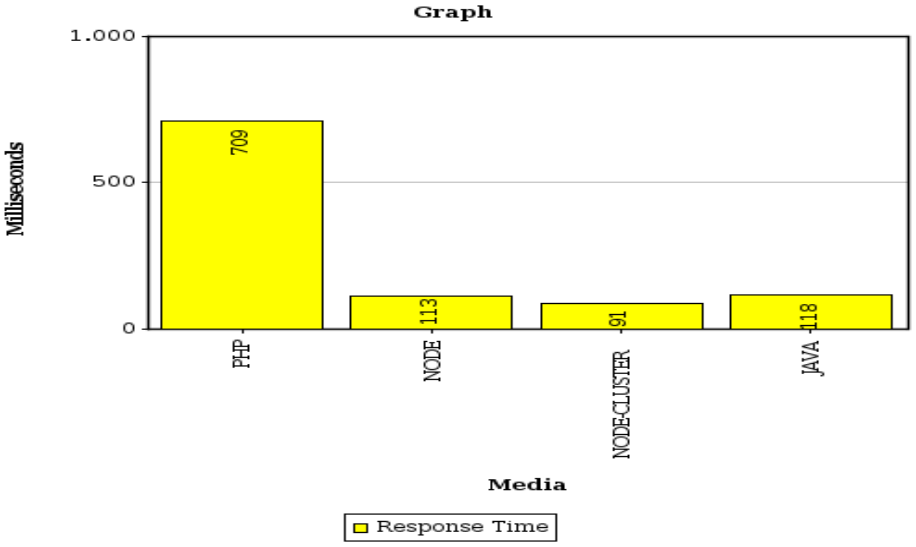


Fig. 28 Tiempo de respuesta en milisegundos con 20H-50C-1000M

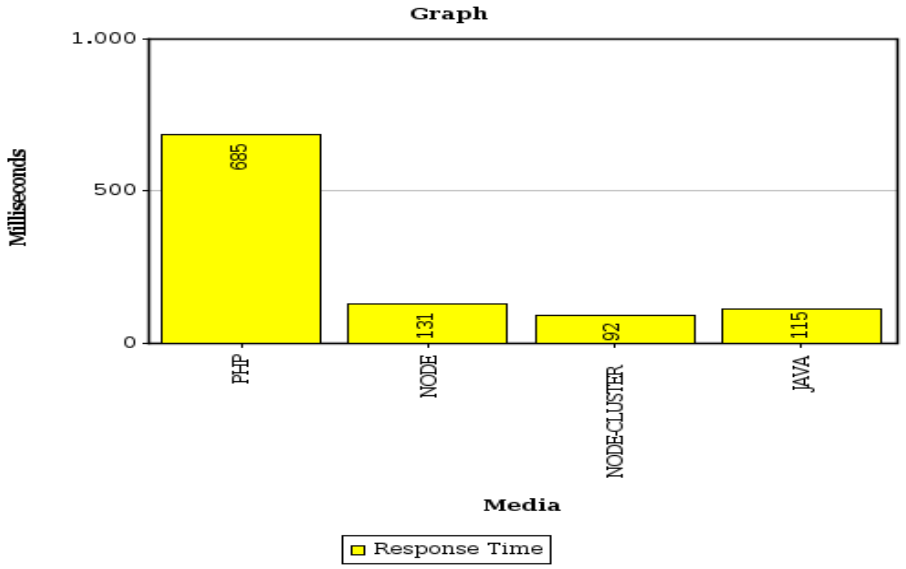


Fig. 29 Tiempo de respuesta en milisegundos con 10H-50C-1000M

Anexo 3. Carta de aceptación

UCI informáticas

CARTA DE ACEPTACIÓN

En cumplimiento con la fase de desarrollo y en función de la ejecución de la tesis Sistema colaborativo de planificación y control de eventos curriculares se hace entrega de los productos que se relacionan a continuación

- Polymita Sistema colaborativo de planificación y control de eventos curriculares(código fuente).
- Manual de usuario.

Entrega	Recibe
Nombre y apellidos: Alain Tejas Escobar, Osiel Suárez Guerra	Nombre y apellidos Glennis Tamayo Morales
Cargo: Tesistas	Cargo: Jefe de Departamento Desarrollo de Componentes
Firma:	Firma:

Fig. 30 Carta de aceptación