Universidad de las Ciencias Informáticas

**Facultad 3**

# Sistema para la Sincronización de Calendarios


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

**Autores:** Yaicel Torres Garces
Yunier José Sosa Vázquez

**Tutores:** Ing. Leonardo Darell Antúnez Naranjo
Lic. Raynel Batista Tellez

**Co-Tutor:** Ing. Alieski Veliz Vega

**La Habana, Cuba**

Junio de 2015

## Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____.

Yaicel Torres Garces

Firma del autor

Yunier José Sosa Vázquez

Firma del autor

Ing. Leonardo Darell Antúnez Naranjo

Firma del tutor

Lic. Raynel Batista Tellez

Firma del tutor

Ing. Alieski Veliz Vega

Firma del co-tutor

**Dedicatoria**

*Dedico el presente trabajo de diploma a mis padres, a mi familia por todo su amor, sus sacrificios, su cariño y su confianza y a mi amigo Joel quien no se encuentra hoy entre nosotros pero su recuerdo sigue presente y me motivó.*

*A Fidel y a la Revolución.*

*A todos mis compañeros y amigos.*

*A la Universidad de las Ciencias Informáticas.*

Yaicel Torres Garces

*Dedico este trabajo de diploma a mamá Elena, mi abuelo Julio, mi padrastro Ever y a toda mi familia en general por ser fuente de inspiración para la realización de este trabajo.*

*A mis compañeros y amigos.*

*A Firefoxmanía y Mozilla.*

*A la Revolución.*

Yunier José Sosa Vázquez

# Agradecimientos

**De Yaicel:**

*A mi mamá Eloina Garcés y mi hermano Yordanis Torres que han sido todo para mí y han estado siempre ahí gracias por toda la confianza, apoyo y sacrificio dado. A Yosmara mi cuñada que junto a mi hermano ha sido mi otra madre y me ha cuidado cuando ella no estaba. A mi padre Félix Torres por todo su apoyo y confianza depositada en mí. A mis amigos Mario y Joel que son hermanos de los que no son de sangre gracias por no rendirse conmigo y apoyarme hasta el final. A mis vecinos, que siempre me apoyaron.*

*A la gran familia que he creado desde que empecé aquí. En especial a Raynel por ser más que un amigo, ha sido un padre para mí, gracias por todo el apoyo, la confianza y por creer en mí, a Marian quien me ha acompañado durante estos años y me ha dado todo su apoyo incondicional, amor y confianza, gracias por soportarme y estar en los momentos más difíciles, a Aniel quien ha sido un gran amigo en el cual se puede confiar, gracias por todo tu apoyo y confianza, a Alieski por su amistad, confianza, apoyo y café.*

*A Yoanys y Dannel gracias por todo su apoyo, amistad y confianza. A Pupo por ser un gran amigo mío y de en esta gran familia gracias por el apoyo. A Leo por su amistad, apoyo y dedicación.*

*Le agradezco a mis compañeros que han compartido conmigo tantos momentos durante estos años, en especial: Roli, Bauta, Negrín, Ale, Yotsan, los gimas, José A, Yunior, Oda, Ara, Alicia, Osvaldo, Burgo y a mi compañero de tesis y amigo Sosa.*

*A todo el claustro de profesores que de una forma u otra han influido en mi preparación como profesional y que me han enseñado a transitar en el camino por la búsqueda del conocimiento.*

*Al gran equipo de SO3 donde comencé a forjarme gracias por todo el apoyo, conocimiento y amistad brindada.*

# Agradecimientos

**De Yunier:**

*A mi mamá Elena por ser madre y padre a la vez, por ser la razón de mi existir y el pilar fundamental de mi vida, por confiar siempre en mí, por su apoyo, dedicación, confianza, por su amor incondicional, por guiarme siempre por el buen camino y por ser la fuente de inspiración para alcanzar las metas y superar los obstáculos.*

*A mi abuelo Julio por ser un padre para mí y enseñarme muchas de las cosas hoy sé, por su cariño y su ayuda incondicionalidad en todo momento, por su confianza y ánimos para seguir adelante. Siempre te recordaré en lo más profundo de mi corazón.*

*A mi padrastro Ever por su ayuda en estos últimos doce años, por su confianza, el respeto y su sacrificio.*

*A mi tía Nery y mis primos Yuniesky y Julio Antonio por apoyarme siempre.*

*A Adelaida por su apoyo incondicional, por sus consejos y darme ánimos para seguir adelante.*

*A Hortensia, Isabel, Oro, Oneida y Deisy por su afecto en todo momento y cuidarme como un hijo.*

*A mi familia, a cada uno de ellos que forman una parte importante en mi vida.*

*A Carlos, Burgos, Tony y Ernesto por ser los mejores amigos, por sus consejos, su ayuda, por aguantarme, por estar siempre cuando los necesité.*

*A Erick por enseñarme muchas cosas y adentrarme en el mundo de Mozilla, por ser como un hermano y ayudarme incondicionalmente en todo.*

*A mis colegas de Firefoxmanía: mi tocayo, Pochy, Manuel, Ody, Nilmar, Abraham, Roberto y los amigos de Mozilla Hispano, en especial a Guillermo, Rubén, Franc y Fernando. Junto a ellos he aprendido muchas cosas buenas y disfrutado de excelentes momentos.*

*A mis tutores Raynel, Leo y Alieski por su guía, apoyo y sus enseñanzas.*

*A mi compañero de tesis Yaicel por ayudarme y realizar conmigo este genial trabajo de diploma.*

*A Miriam Nicado por su confianza y creer siempre en mí.*

*A Philipp Kewisch y Gareth Aye por su colaboración en el tema tratado en esta investigación.*

*A Alain, Anel, Raúl, Ricardito y Oscar, compañeros del viejo apartamento y que juntos compartimos buenos momentos.*

*A todas las personas del grupo que me hicieron sentir en familia y me hicieron su presidente y amigo. En especial a Aniel, Negrín, Marielys, Yaicel, Odalys, Araí, Bauta, Rolando, los jimaguas, Alicia, Danaray, Alejandro, Ernesto y Addel.*

*A Vilma, Yanet y Giselle por su cariño.*

*A Macías, Yuri, Bárbaro, Dayana, Chicho y Anita.*

*A Yunet, Yarennys, Aylín y Arnolys por apoyarme y motivarme siempre.*

*A toda la gente del barrio por darme aliento para continuar.*

*A todas las personas que de una forma u otra han aportado un granito de arena en mi formación como ingeniero.*

*A todos los amigos que hice y se me olvidó mencionar.*

*A todos ellos muchas gracias.*

# Resumen

La Universidad de las Ciencias Informáticas es una organización muy activa debido a su compromiso con la sociedad cubana. Su dinámica genera constantemente actividades que involucran disímiles personas. La mayoría de dichas actividades es gestionada por diversas soluciones informáticas, de modo que una misma persona debe consultar varios sistemas para gestionar las mismas. La incomunicación entre los sistemas para la planificación operativa provoca en algunos casos sobrecargas y conflictos laborales que podrían alterar el clima laboral y afectar la motivación de las personas. Se dice que generalmente tan solo el 20% de nuestro tiempo contribuye al 80% de los resultados. El resto, suelen ser imprevistos, urgencias, interrupciones, en definitiva, desorden. Esta problemática limita la precisión al planificar tiempo y recursos disponibles afectando el rendimiento de los usuarios. Para dar respuesta a este problema se traza como objetivo desarrollar una herramienta para la sincronización de calendarios entre usuarios y aplicaciones del dominio uci.cu que permita elevar la precisión al planificar tiempo y recursos disponibles, contribuyendo al rendimiento de los usuarios. El proceso de desarrollo fue guiado por la metodología XP y se recurrió al lenguaje de programación JavaScript con AngularJS (cliente) y Node.js (servidor), MongoDB como sistema gestor de base de datos y el marco de trabajo MEAN.IO. Como resultado de esta investigación se obtuvo KLNDA, una aplicación web capaz de sincronizar calendarios de múltiples sistemas existentes, gestionar el conflicto entre actividades, mantener informado a las personas a través de notificaciones y propiciar el intercambio de información mediante servicios web.

**Palabras clave:** estándar, evento, planificación, servicio web, sincronización

# Tabla de contenidos

## CAPÍTULO 2: DESARROLLO DE LA SOLUCIÓN



## CAPÍTULO 3: VALIDACIÓN DE LA PROPUESTA

# Introducción

El desarrollo avanzado de las nuevas Tecnologías de la Información y las Comunicaciones (TIC), ha provocado profundos cambios en el contexto social y laboral en el que se desenvuelve el hombre moderno. Como resultado directo de este proceso ha emergido el concepto de Sociedad de la Información, el cual ha irrumpido como un nuevo fenómeno a escala mundial donde la información y el conocimiento ocupan un lugar privilegiado en la sociedad (Antúnez Naranjo 2012).

La alta disponibilidad y accesibilidad de la información en la sociedad actual ha abierto el espectro a un cúmulo de aplicaciones orientadas a diferentes sectores como por ejemplo: la medicina, la administración pública y empresarial, transporte público y la educación. Estos sistemas responden a las necesidades específicas del dominio o negocio para el cual fueron creados, motivo por el cual, el grado de descentralización y dispersión de los datos almacenados, representa un tema en consideración, especialmente en una sociedad que consume y crea enormes volúmenes de información.

La penetración de las tecnologías en casi todas las áreas de la sociedad obliga a utilizar varias soluciones informáticas en función de sus intereses y objetivos, para lo cual se debe dedicar un tiempo finito y asimilar la información contenida. Generalmente, estas aplicaciones podrían estar integradas, y en ese caso, dichas personas estarían ahorrando tiempo útil.

El tiempo es la magnitud física que permite ordenar la secuencia de los sucesos, estableciendo un pasado, un presente y un futuro (Real Academia Española 2012). Se dice que generalmente tan solo el 20% de nuestro tiempo contribuye al 80% de los resultados. El resto suelen ser imprevistos, urgencias, interrupciones, correcciones, en definitiva, desorden. Por esa razón, se debe controlar bien el tiempo para aprovecharlo al máximo (Alter-Nos Comunicaciones 2014).

Cuba no está exenta a esas oportunidades y apuesta por la informatización de la sociedad al fundar en 2002 la Universidad de las Ciencias Informáticas (UCI), con la misión de formar profesionales comprometidos con su patria y altamente calificados en la rama de la Informática. Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación. Servir de soporte a la industria cubana de la informática (UCI 2015).

La UCI juega un papel fundamental en la informatización del país, de ahí que su *slogan* sea: "*Conectados al futuro, conectados a la Revolución*". Uno de los desafíos de la UCI sigue siendo la integración entre sus aplicaciones. Desde la fundación de la alta casa de estudios, esta se ha caracterizado por la gran cantidad de recursos humanos que en ella se desenvuelven, llegando a tener

en determinados períodos más de 14 mil personas vinculadas a los procesos claves y de soporte que en ella se desarrollan. Se puede concluir, por tanto, que una correcta planificación de actividades y eventos en este centro de estudios tan dinámico y numeroso, reviste vital importancia en el desarrollo integral de la institución.

La UCI es una organización muy activa debido a su compromiso con la sociedad cubana. Su complejidad eleva su estructura a un orden matricial, de modo que a pesar que una misma persona tenga un responsable administrativo que lo atiende, puede recibir instrucciones y afectaciones por otras vías, canales y personas. En ocasiones cuando se han planificado actividades, surgen factores internos que afectan la planificación.

Su dinámica genera constantemente eventos[1] que involucran disímiles personas. La gestión de la mayoría de dichas actividades es realizada por distintas soluciones informáticas entre las que podemos encontrar: el Sistema de Planificación (SIPAC), el calendario del Cliente web de Zimbra, la *suite* de Gestión de Proyectos (GESPRO) y el Sistema de Planificación del horario docente UCI. De modo que una misma persona debe consultar varios sistemas para gestionar la mayoría de sus actividades, lo que implica un empleo ineficiente del tiempo disponible.

La incomunicación entre estos sistemas que gestionan actividades de las personas provoca en muchos casos sobrecargas y conflictos laborales que generan un inadecuado funcionamiento en la organización alterando el clima socio laboral y pudiendo afectar la motivación de las personas.

Teniendo en cuenta la situación planteada anteriormente, se define como **problema de investigación,** el modo en que se realiza la asignación de eventos entre usuarios y aplicaciones del dominio **uci.cu** limita la precisión al planificar tiempo y recursos disponibles afectando el rendimiento de los usuarios. Este problema conduce a que se plantee como **hipótesis** que si se desarrolla una herramienta para la sincronización de calendarios entre usuarios y aplicaciones del dominio uci.cu, permitirá elevar la precisión al planificar tiempo y recursos disponibles, contribuyendo al rendimiento de los usuarios.

Para ello se tendrá como **objeto de estudio** las aplicaciones para la planificación operativa.

Se identifica además como **campo de acción** las soluciones informáticas de planificación operativa en el dominio uci.cu y como **objetivo general** se traza desarrollar una herramienta para la sincronización

[1] Se refiere a reunión, cita, tarea, actividad, evento docente o de otra índole

de calendarios entre usuarios y aplicaciones del dominio uci.cu que permita elevar la precisión al planificar tiempo y recursos disponibles, contribuyendo al rendimiento de los usuarios.

Para dar cumplimiento al **objetivo general,** se consideran los siguientes **objetivos específicos:**

- Analizar herramientas de sincronización de calendarios para comparar sus características e identificar limitaciones, enfoques y tendencias.
- Implementar una herramienta que permita la sincronización de calendarios entre usuarios y aplicaciones del dominio uci.cu.
- Validar la herramienta propuesta aplicando pruebas unitarias, de aceptación y validación.

Para dar cumplimiento al **objetivo general** propuesto y solucionar la situación problemática planteada se plantean un grupo de **tareas investigativas**:

- Caracterización de los métodos de sincronización de calendarios.
- Caracterización de tecnologías y lenguajes a emplear para desarrollar la herramienta.
- Diseño y validación de los prototipos.
- Realización del análisis y diseño de la herramienta.
- Implementación de los requisitos que permitan el diseño de las interfaces de usuario.
- Validación de la herramienta propuesta a partir de pruebas unitarias, de aceptación y validación.

El presente trabajo de diploma está estructurado en 3 capítulos:

**Capítulo 1: Fundamentación teórico-conceptual.**

En este capítulo se expone la fundamentación teórica del trabajo, incluyéndose en el mismo el estudio de otras soluciones informáticas existentes en la actualidad y de las metodologías, modelos de desarrollo de *software* y herramientas a utilizar.

**Capítulo 2: Desarrollo de la solución.**

En este capítulo se describe la implementación de la solución al problema planteado con anterioridad, exponiendo los prototipos de interfaces, los artefactos generados en las fases de planificación, diseño y codificación.

**Capítulo 3: Validación de la solución propuesta.**

En este capítulo se plasman los resultados de los casos de prueba, obteniendo una valoración integral de la solución propuesta que permita determinar los puntos débiles del sistema.

# CAPÍTULO 1:

# FUNDAMENTACIÓN TEÓRICO-CONCEPTUAL

## 1.1. Introducción.

El desarrollo vertiginoso de las TIC ha provisto al hombre de numerosas herramientas que mejoran considerablemente las condiciones de vida de la sociedad. Muchos han sido los avances en disímiles áreas como la telefonía celular, la creación de *hardware*[2] y *software*[3] cada vez más profesionales y menos costosos, la televisión digital (DTV)[4] e Internet. Esta evolución hacia una sociedad cada vez más necesitada de información, ha sido el combustible ideal de esta revolución del conocimiento que se ha vuelto, en los albores del siglo XXI, parte intrínseca de todos los sectores de la sociedad. Por estas razones, el uso de tecnologías adecuadas diseñadas con especial atención a los aspectos éticos, culturales, sociales y económicos de la comunidad a la que se dirigen, cobra vital importancia en un mundo que se mueve entre bits y señales satelitales.

## 1.2. Gestión de la planificación.

Desde el inicio de la humanidad el hombre se ha propuesto hacer un uso racional y eficiente de los recursos y especialmente del tiempo. Existen distintos mecanismos de planificación entre los que toman un gran predominio los planes de intervalos cortos, tiempo durante el cual se desarrolla habitual o regularmente una acción o se realiza una actividad, también conocido como horario.

La planificación es una parte indispensable de la vida y de la sociedad que nos rodea. Se puede decir que funciona como un reflejo incondicional, desde el mismo momento en que una persona se levanta pensando en las faenas que debe cumplir durante ese día y la mejor manera de lograrla, ubicándose mejor en el tiempo y espacio necesario para ello. Las ideas concebidas con el fin de lograr una meta, alcanzar un propósito, de forma consciente o no, se traducen sencillamente en “planificación”.

Uno de los exponentes más prominentes en materia de planificación, y considerado como el padre de la planificación en los tiempos actuales, James A. Finch, plantea que “la planificación no es más que el

[2] Conjunto de los componentes que integran la parte material de una computadora.
[3] Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.
[4] Tecnología de transmisión avanzada que ha transformado la experiencia de ver televisión. La DTV ha permitido a las transmisoras ofrecer televisión con mejor calidad de imagen y sonido. También ofrece múltiples opciones de programación, lo que se denomina multidifusión, y funciones interactivas.

proceso de establecer metas y definir medios para alcanzar las mismas" (Finch, Freeman y Gilbert 1996).

Por otra parte, el Máster en Administración y Dirección de Empresas Andrés M. Núñez Parada propone que la planificación "consiste en determinar las metas u objetivos a cumplir seleccionando las misiones, objetivos y acciones para alcanzarlos" (Núñez Parada 2010). Partiendo de la esencia de las ideas anteriormente expuestas, este trabajo se acoge al concepto de que la planificación es una de las funciones del ciclo administrativo que permite establecer un nivel de estabilidad en las entidades y proporciona condiciones favorables ante los cambios avizorados o no en un futuro.

## 1.3. Soluciones Informáticas para la Planificación.

Numerosos han sido los esfuerzos realizados por el hombre para controlar el tiempo, aumentando su importancia en la agricultura debido a la necesidad de predecir las estaciones climáticas a fin de controlar los períodos de siembra, cosecha, etc. Los primeros calendarios conocidos aparecieron en las culturas babilónica y egipcia.

La palabra Calendario proviene del latín *calendarium*, nombre que recibían los libros de contabilidad romanos. Por su parte, *Calendarium* deriva de *Kalendae* o sea calendas, la cual para los romanos simbolizaba el primer día de cada mes (Anders 2014). Los autores de esta investigación, atendiendo a los términos tratados anteriormente, decidieron bautizar al sistema para sincronizar calendarios con el nombre de KLNDA.

Fue gracias a Julio César, quien encargó al sabio Sosígenes la normalización del calendario, dando lugar al conocido como calendario juliano, base del actualmente más extendido, de doce meses y 365 días de duración, aumentando a 366 días cada cuatro años. La divergencia de paso y significación de la noción temporal - ya sea para cada época, ya sea para cada sociedad - es consecuencia de la organización social del tiempo, por lo que cada período histórico ha sido marcado por diferentes formas pensar y artefactos para su dominio (Ceballos 2003).

Con el progreso y expansión de la informática y las telecomunicaciones, se han desarrollado soluciones informáticas destinadas a proveer herramientas que propicien un alto control del tiempo a personas y empresas. Beneficiando a todo aquel que las emplee de forma eficaz pues contribuye a elevar el rendimiento debido a que cada uno sabe qué hacer en cada momento.

Para un correcto intercambio de información entre calendarios se han creado estándares que garantizan el acoplamiento y una estructura similar de los elementos. Entre los estándares de

calendarios definidos por la IETF[5] (en español Grupo de Trabajo de Ingeniería de Internet) podemos encontrar iCalendar[6] creado por Apple y más conocido como iCal, y el microformato hCalendar[7]. La información de los calendarios puede ser compartida y editada utilizando un servidor WebDAV[8] o su extensión CalDAV[9] a través del consumo de servicios web.

A continuación se describen algunas de las soluciones informáticas de interés para la investigación, ya sea por su notoriedad en el mercado o por sus características que las relacionan con el contexto del problema enunciado.

### 1.3.1. Google Calendar.

Google Calendar es una aplicación web gratuita desarrollada por Google[10] que organiza la agenda personal y en la cual se pueden compartir eventos con compañeros de trabajo, familiares y amigos. Es necesario poseer una cuenta de Google para poder utilizar este servicio.

Su interfaz, en concordancia con los demás servicios de Google, presenta un diseño amigable y sencillo, haciendo que su uso sea fácil y atrayente al usuario final. Google Calendar está disponible en una aplicación para Android[11] (adaptada a pantallas pequeñas) y mediante el consumo de servicios web, es posible sincronizar el calendario de Google con aplicaciones de escritorio (Apple iCal, Mozilla Sunbird, etc.) y dispositivos móviles que tengan instalada alguna aplicación para administrar calendarios.

Al añadir personas para participar en los eventos, estas pueden confirmar su asistencia por correo electrónico o a través de la propia aplicación. De igual forma, las notificaciones de eventos por comenzar se envían por correo electrónico o al teléfono móvil a través de un SMS[12] (en español Servicio de Mensajes Cortos) (Google Inc. 2014).

Los usuarios pueden crear varios calendarios y hacer pública su planificación con personas designadas por ellos o mediante una solicitud realizada por otra. Esto no significa que todos los eventos podrán ser visualizados, el dueño de la cuenta puede crear el evento como privado.

[5] Siglas en inglés para Internet Engineering Task Force.
[6] http://tools.ietf.org/html/rfc5545
[7] http://www.microformats.org/wiki/hcalendar
[8] https://tools.ietf.org/html/rfc4918
[9] https://tools.ietf.org/html/rfc4791
[10] Empresa de Internet fundada en 1998.
[11] Sistema Operativo para Teléfonos Inteligentes desarrollado por Google.
[12] Siglas en inglés para Short Message Service.

Permite exportar e importar calendarios en formato iCal pero no realiza la función de sincronizador de agendas con otras aplicaciones. Su filosofía es ser el centro de planificación de los usuarios. Está escrito en Java.

### 1.3.2. Yahoo! Calendar.

Yahoo! Calendar es un servicio gratuito que permite crear una agenda en la que puede añadir, modificar y acceder a los eventos y tareas. Puede acceder a su calendario desde cualquier ordenador con conexión a Internet. Es necesario ser usuario registrado de Yahoo![13] para poder acceder y utilizar este servicio.

Puede programar y mandarse un recordatorio de cada evento a su dirección de correo electrónico, mediante Yahoo! Messenger, o a un dispositivo móvil (con dirección de correo electrónico). También puede mandar una invitación por correo electrónico a terceros. Yahoo! usa la dirección de correo electrónico especificada únicamente para enviar ese recordatorio o invitación.

Si el usuario desea, los datos que se incorporen serán (Yahoo! Inc. 2015):

- Públicos, todos los usuarios de Yahoo! podrán acceder a ellos.
- Restringidos a determinados usuarios de Yahoo!, se debe seleccionar y especificar quién puede acceder a dichos datos.
- Privados, solo él puede acceder a ellos.

### 1.3.3. Cliente web de Zimbra Collaboration Suite.

El cliente web de Zimbra es un completo programa de mensajería y colaboración que ofrece un servicio de correo electrónico fiable y de alto rendimiento, con libretas de direcciones, agendas, listas de tareas y la capacidad de escribir documentos.

Con la Agenda se pueden controlar y programar citas, reuniones y eventos. Además, permite mantener varias agendas, mover eventos de una agenda a otra sin problema alguno y compartir las agendas con otras personas.

[13] Empresa de Internet fundada en 1995.

En Zimbra se pueden incorporar agendas externas provenientes de otros sistemas, como por ejemplo Yahoo! Calendar, suscripciones iCal y cuentas CalDAV.

La función Tareas permite crear listas de quehaceres pendientes y gestionar los mismos hasta finalizarlos. Se puede añadir tareas a la lista predeterminada de tareas o crear otras listas para organizar las tareas por actividades más específicas como, por ejemplo, tareas profesionales o proyectos personales (Telligent Systems Inc. 2013).

A la hora de crear citas o eventos donde participen varias personas, Zimbra muestra sugerencias de horas para realizar la actividad teniendo en cuenta el horario laboral y muestra la disponibilidad personal en una línea de tiempo de cada asistente. También da la posibilidad de indicar si es necesaria o no su presencia en el encuentro que se está planificando.

Las notificaciones se realizan en forma de alertas emergentes en la aplicación si el usuario está autenticado en el momento del recordatorio. Está escrito en Java, utiliza MySQL como sistema gestor de base de datos y corre sobre un servidor web Nginx.

### 1.3.4. Sistema para la Planificación de Actividades.

El Sistema de Planificación de Actividades (SIPAC) está basado en la Instrucción número 1 del Presidente de los Consejos de Estado y de Ministros para la planificación de los objetivos y actividades en los órganos, organismos de la Administración Central de Estado, entidades nacionales y administraciones locales del Poder Popular. Está destinado a facilitar la gestión de las actividades a todos los niveles organizacionales, permite interrelacionar objetivos de trabajo y actividades en tiempo real; garantizando el seguimiento del desarrollo y cumplimiento de los objetivos y tareas principales en las entidades. Informatizando los procesos de ejecución y control de la planeación estratégica (definición de los objetivos a largo plazo y estrategias para alcanzarlos) y operativa (puntualización de las actividades que debe efectuar cada individuo a corto plazo).

SIPAC cuenta con varios módulos encargados de generar las configuraciones necesarias que otorgan al sistema y al cliente una simulación de los procesos de organización del personal, así como los niveles de subordinación necesarios e indispensables para efectuar una planificación de actividades basadas en reglas de la compartimentación de la información, permitiendo que la información planificada sea accedida por la persona autorizada, en el momento indicado. Puede catalogarse como una solución integral para la gestión de elementos de la planeación estratégica y operativa basada en actividades, objetivos, y planes, diseñada sobre las bases de la compartimentación de la información (SIPAC 2014).

Actualmente SIPAC se encuentra desplegado en la Facultad 3 de la UCI, en el Consejo de la Administración Pública de Artemisa, en la secretaría del Consejo de Ministros y en varios ministerios de Cuba. Está desarrollado sobre PHP, se ejecuta sobre un servidor Apache y utiliza PostgreSQL como sistema gestor de base de datos.

Las actividades en SIPAC pueden ser exportadas mediantes archivos en formato .xls o en una base de datos SQLite[14] pero no ofrece servicios web para que otras aplicaciones inserten u obtengan información almacenada en él.

### 1.3.5. Sistema de Planificación y publicación de horarios docentes UCI.

El Sistema de Planificación y publicación de horarios docentes UCI (Horr) es una aplicación que permite gestionar el horario docente de los estudiantes y profesores de la universidad. Está compuesto por una aplicación de escritorio (HorrPlanner) orientada a los planificadores y una aplicación web (HorrPublisher) orientada a los usuarios interesados en consultar la planificación.

HorrPlanner está diseñado para ser fácil de usar y de portar aunque solamente puede emplearse en el sistema operativo Windows con el framework .Net 2.0, está escrito en ASP.NET y su versión web (HorrPublisher) corre sobre un servidor Microsoft-IIS 6.0 con X-AspNet 4.0.30319 (Silva Barrera 2011).

Entre sus principales características se encuentran:

- Creación del plan para la facultad.
- Declaración de los recursos de la facultad.
- Distribución de los recursos.
- Revisión de calidad a la planificación.

### 1.3.6. Suite de Gestión de Proyectos (GESPRO).

GESPRO es un ecosistema de *software* para la dirección integrada de proyectos (DIP) desarrollado por el Laboratorio de Investigaciones en Gestión de Proyectos de la UCI que combina el uso de una solución informática para la DIP y un sistema de formación especializada en gestión de proyectos.

[14] Gestor de base de datos relacional caracterizado por su pequeño tamaño

Esta *suite* posibilita la planificación, el control y seguimiento de los proyectos y de los recursos asociados a los mismos, en alineación con la proyección estratégica de las organizaciones. Se comercializa bajo un modelo de negocios basado en servicios entre los que se destacan: personalización de la solución informática, despliegue en diferentes escenarios, actualizaciones, soporte, consultoría y formación especializada (GESPRO 2013).

GESPRO cuenta con 33 módulos y entre ellos podemos encontrar: GESPRO Auditable, GESPRO Gestión financiera, GESPRO Indicadores de Proyectos y GESPRO Gestión de Riesgos, por solo mencionar algunos. Actualmente todos los centros productivos de la UCI gestionan y hacen seguimiento a sus proyectos apoyándose en esta *suite*.

Está escrito en Java, utiliza el marco de trabajo Ruby on Rails y emplea PostgreSQL como sistema gestor de base de datos.

### 1.3.7. Resultados del estudio de las soluciones informáticas de planificación.

Las soluciones informáticas orientadas a la gestión de la planificación que han sido descritas en esta investigación, responden a un objetivo común y casi todas presentan una misma forma de ver el negocio. Por lo que se pueden tomar varias ideas para tener en cuenta al desarrollar una aplicación.

Google Calendar es una de las soluciones para la gestión del tiempo personal más usada en el mundo por las funcionalidades que este ofrece y el prestigio que le atribuye su creador Google. Este sistema no se puede desplegar en la UCI puesto que su código no está disponible para descargar y solo está accesible bajo el dominio de Google, tampoco gestiona conflictos entre eventos. La forma de notificar a los usuarios los eventos es buena pues explota los sistemas de comunicación más comunes: correo electrónico y SMS. En el mismo caso se encuentra Yahoo! Calendar, aunque este añade notificaciones vía mensajería instantánea a través de Yahoo! Messenger.

El sistema de Planificación y publicación de horarios docentes UCI con sus versiones de escritorio y web, no es suficiente para la solución pensada pues solo se centra en el horario docente. Además, no presenta servicios web para insertar y consumir datos, lo que hace más difícil la tarea de integrarse a otras soluciones.

Por su parte, SIPAC es un planificador potente pero no brinda servicios web para insertar u obtener las actividades de un usuario en el sistema y no soporta iCal, imposibilitando la comunicación con otras soluciones similares aunque se desarrollan estas funcionalidades.

GESPRO es explotado por todos los centros productivos de la UCI y mediante él las personas vinculadas a la producción gestionan sus tareas. Por razones de tiempo y alcance, se decidió no sincronizar KLNDA con esta *suite* en la primera versión y luego integrar ambos sistemas.

El calendario presente en el cliente web de Zimbra puede ser utilizado como un centro para sincronizar y gestionar eventos de agendas externas. También sugiere una hora para llevar a cabo una actividad pues toma en cuenta las afectaciones de los implicados, característica que será incluida en la nueva solución informática. A pesar que esta se sincroniza con otros sistemas no tiene en cuenta los conflictos entre eventos existentes en un mismo instante de tiempo.

Por lo anterior expuesto se concluye que:

- Las soluciones informáticas existentes en el mercado aunque garantizan una adecuada planificación no logran llenar las necesidades específicas del problema planteado.
- Las soluciones de Google y Yahoo! aportan ideas creativas y sugerentes para incluir en una aplicación similar como por ejemplo: notificaciones mediante correos, mensajes instantáneos y SMS.
- Google y Yahoo! Calendar son gratuitas pero su código fuente no está disponible y ambas están pensadas para ser el centro, y no para consumir datos de otras soluciones.
- Todas las soluciones informáticas existentes son web.
- Dicho sistema informático debe ser una aplicación web que se encuentre disponible en todo momento, responder rápidamente a peticiones de los usuarios y brindar servicios web para insertar y exportar datos en él.
- Debido a la ausencia de una solución informática que integre los sistemas que gestionan calendarios en la UCI, se hace necesario el desarrollo de un sistema informático capaz de integrar las soluciones para la planificación en la universidad.

## 1.4. Estudio de metodologías, tecnologías y herramientas.

En la actualidad, el uso intensivo de todo tipo de soluciones informáticas en múltiples sectores de la sociedad y especialmente en los procesos productivos, ha provocado un notable crecimiento en la confiabilidad y calidad que estos deben poseer. Para desarrollar un producto que satisfaga estos indicadores se debe tener en cuenta una serie de tendencias, tecnologías, lenguajes, metodologías y

herramientas indispensables para garantizar la entrega de una solución informática competitiva y ajustada a las necesidades de los clientes.

A continuación se realiza un estudio de metodologías, tecnologías y herramientas con el propósito de determinar las más convenientes para la solución propuesta, garantizando de esta forma que la solución informática utilice las tecnologías propuestas en la arquitectura para el sistema de sincronización de calendarios.

### 1.4.1. Proceso de desarrollo de *software*.

El proceso de desarrollo de *software* "es aquel en que las necesidades del usuario son traducidas en requerimientos de *software*, estos requerimientos transformados en diseño y el diseño implementado en código, el código es probado, documentado y certificado para su uso operativo". Concretamente "es una estructura aplicada al desarrollo de un producto *software* que define quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo" (Jacobson 1998).

El proceso de desarrollo de soluciones informáticas requiere por un lado un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del *software* que comprende cuatro grandes fases: concepción, elaboración, construcción y transición (Zavala Ruiz 2008).

Para lograr la consecución exitosa de un proyecto de *software*, existen un conjunto de modelos que permiten perfilar el proceso de desarrollo, las cuales deben ser analizadas cuidadosamente. Usualmente la selección de uno de estos modelos o de la unión de varios responde a las necesidades y condiciones específicas del proyecto.

Entre estos modelos o metodologías se pueden clasificar en cinco grandes grupos:

- **Modelo de cascada:** provee un enfoque muy lineal para el desarrollo de programas informáticos, ya que cada paso en esencia lleva al siguiente hasta que todo el *software* se ha completado.
- **Modelo iterativo incremental:** en muchos aspectos, el modelo incremental emula el modelo de cascada, con la diferencia que recomienda la construcción de secciones reducidas de *software* que irán ganando en tamaño para facilitar así la detección de problemas de importancia antes de que sea demasiado tarde.
- **Modelo espiral:** en cierto modo, el modelo de espiral es como una versión más grande e intensa del modelo incremental. Este modelo también es conocido por permitir el desarrollo de *software*

de alta calidad, este resultado es obtenido gracias a la característica fundamental del modelo que es la gestión de riesgos de forma periódica en el ciclo de desarrollo.

- **Modelos ágiles:** en la gestión de proyectos ágiles, existe un esqueleto básico conformado con las mejores prácticas basadas en el sentido común, pero la gestión global se basa en otros métodos. Este modelo utiliza un desarrollo iterativo como base para abogar por un punto de vista más ligero y más centrado en las personas. Los procesos ágiles utilizan retroalimentación en lugar de planificación como principal mecanismo de control.
- **Modelo de prototipos:** En el método de creación de prototipos, el *software* es desarrollado en forma similar a una cebolla. Cuando el proyecto se inicia, el equipo se centra en la construcción de un prototipo con todos o la mayoría de las características y construye todo un programa para que el cliente lo utilice. Después de que el cliente acepta el prototipo, el equipo de programación crea la siguiente capa mediante la corrección de los errores y basándose en los requerimientos fundamentales.

Teniendo en cuenta los modelos anteriormente planteados, las características de la solución que se desea desarrollar y el equipo de trabajo, los modelos ágiles, junto a los modelos de prototipos, proveen un marco idóneo para el desarrollo del producto, debido a que este tipo de modelos ofrece las siguientes ventajas (Sousa 2012).

- Cuenta con un equipo adaptado que es capaz de responder ante las necesidades cambiantes.
- La comunicación cara a cara y las entradas continuas de los representantes del cliente no dejan espacio para conjeturas.
- La documentación es nítida y precisa para ahorrar tiempo.
- El resultado final es el software de alta calidad en menos tiempo de duración posible y un cliente satisfecho.

Entre las principales metodologías ágiles se puede encontrar (Appelo 2008):

- **Scrum:** es una metodología de gestión de proyectos ágil, creado por Ken Schwaber y Sutherland Jeff. Se trata de un esqueleto que incluye un pequeño conjunto de prácticas y roles predefinidos. Una de las razones de la popularidad de Scrum es que solo consta de ciertas prácticas de sentido común que pueden aplicarse en muchas situaciones. Esto también significa que Scrum por sí mismo nunca es suficiente y que los equipos de desarrollo tienen que utilizar simultáneamente otros métodos (por lo general XP) para prácticas adicionales.
- **Extreme Programming (XP):** es una metodología de ingeniería de software ágil, creado por Kent Beck. Se trata de un conjunto de buenas prácticas de los cuales algunos son llevados a

niveles “extremos”. Al igual que con otros métodos ágiles, XP se refiere a los cambios en curso como un aspecto natural y deseable en el desarrollo de software. XP es a menudo visto como un complemento a Scrum, llenando la mayor parte de los agujeros que deja abiertos Scrum.

- **Agile Unified Process (AUP):** en español Proceso Unificado Ágil de Scott Ambler, es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Abarca siete flujos de trabajo, cuatro ingenieriles y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente.
- **Lean Software Development (LSD):** es un marco de gestión de proyectos ágiles. Originalmente promovida por Mary Poppendieck y Tom Poppendieck, Lean Software Development es una adaptación del sistema de desarrollo de productos de Toyota, y es la encarnación de la subcultura “magra” que existe dentro de la comunidad ágil. Se dice que los conceptos magros y ágiles forman una pareja perfecta.

Para el desarrollo de la solución se ha seleccionado XP, teniendo en cuenta que una de las principales ventajas de este modelo es que está centrado en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de *software*, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo (Wells 2013). Además, XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

Utilizar XP proporciona ventajas pues enfatiza el trabajo en equipo, la colaboración y las relaciones interpersonales entre sus miembros, potencia el cumplimiento del objetivo principal: lograr la satisfacción del cliente. Al implementar un entorno simple pero eficaz, permite a los equipos ser altamente productivos, auto-organizándose alrededor del problema para resolverlo lo más eficientemente posible. Por otro lado el entorno del grupo de desarrollo (equipo pequeño) es propicio para el uso de XP.

### 1.4.2. Lenguaje de modelado.

Usualmente para desarrollar una solución informática con calidad y competitiva, el uso de lenguajes de modelado es prácticamente indispensable. Esta necesidad responde a que el uso de estos lenguajes permite que el desarrollo de *software* se formalice a través de estándares unificados. Se entiende por lenguaje de modelado cualquier lenguaje artificial que puede ser utilizado para expresar la información,

el conocimiento o sistemas en una estructura que está definida por un conjunto coherente de reglas. Las reglas se utilizan para la interpretación del significado de los componentes en la estructura. Los lenguajes de modelado utilizan modelos visuales y diagramas que realizan esa representación de manera precisa, entendible y económica, lo que facilita su uso en las herramientas para la Ingeniería de *Software* Asistida por Computación (CASE[15]) convencionales (Zapata et al. 2009).

Estos lenguajes se clasifican en dos tipos:

- **Lenguajes gráficos:** utiliza una técnica de diagramas con símbolos que representan los conceptos mencionados y las líneas que conectan los símbolos representan las relaciones.
- **Lenguajes textuales:** suelen utilizar palabras claves estandarizadas acompañadas de parámetros para que las expresiones sean interpretables por un ordenador.

### 1.4.2.1. Lenguaje Unificado de Modelado (UML[16]).

El Lenguaje Unificado de Modelado (UML) es un leguaje de modelado estandarizado de propósito general en el campo de la ingeniería de *software* orientada a objetos. La norma fue creada por el Object Management Group (OMG) en 1997, y desde entonces se ha convertido en el estándar del sector para el modelado de sistemas de *software*. UML incluye un conjunto de técnicas de notación gráfica para crear modelos visuales de programación orientada a objetos (Pressman 2005).

Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de *software*, para detallar los artefactos en el sistema y para documentar y construir.

Independientemente de las características de BPMN[17], los estándares que propone el lenguaje UML ofrecen innumerables ventajas para el proyecto. Teniendo en cuenta que la solución que se desea desarrollar se desenvuelve en un dominio, resulta más complicada la representación a través de procesos de negocio, por lo que el uso de UML se hace indispensable.

[15] Siglas en inglés para Computer Aided Software Engineering.
[16] Siglas en inglés para Unified Modeling Language.
[17] Siglas en inglés para Business Process Model and Notation.

### 1.4.3. Herramientas CASE.

#### 1.4.3.1. 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.

Visual Paradigm 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 libre de probada utilidad para el analista que fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas informáticos de forma fiable a través de la utilización de un enfoque orientado a objetos (Visual Paradigm 2014).

### 1.4.4. Lenguajes de programación.

#### 1.4.4.1. PHP.

Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. PHP es ampliamente utilizado para fines generales aunque es especialmente adecuado para el desarrollo web y puede ser embebido en páginas HTML (The PHP Group 2014).

PHP brinda la posibilidad de usar programación procedimental o programación orientada a objetos (POO), o una mezcla de ambos. Una de las características más fuertes e importantes de PHP es su soporte para una amplia gama de bases de datos. Utilizando una de las extensiones de bases de datos específicas (por ejemplo, para MySQL[18]), o el uso de una capa de abstracción como PDO, se puede interactuar entre una página web y la base de datos.

Cuenta una amplia comunidad formada por programadores expertos y la disponibilidad de marcos de trabajo que facilitan el desarrollo de aplicaciones.

#### 1.4.4.2. Java.

Es un lenguaje de programación de propósito general, concurrente, basado en clases y orientado a objetos. Está diseñado para ser lo suficientemente simple y que lo programadores puedan ganar fluidez en el lenguaje.

[18] Sistema gestor de base de datos relacional.

El lenguaje está relacionado con C y C++, pero organizado de forma diferente y omite algunas ideas de estos. Su objetivo es ser un lenguaje de producción. Java exhibe un tipado[19] estático y fuerte, lo que permite distinguir este tipo de errores en tiempo de compilación[20]. Utiliza una máquina virtual (bytecode) para interpretar y traducir las instrucciones de alto nivel a lenguaje binario entendible para las computadoras (Gosling et al. 2015).

Java es utilizado en el desarrollo de aplicaciones de escritorio, pasando por aplicaciones web, hasta las aplicaciones para teléfonos móviles. Su sintaxis es fácil de usar y está soportado por una amplia comunidad de programadores expertos.

#### 1.4.4.3. JavaScript.

JavaScript (JS) es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de *script* para páginas web, pero también usado en muchos entornos sin navegador, tales como Node.js o Apache CouchDB[21]. Es un lenguaje *script* multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa.

El JavaScript estándar es ECMAScript. A partir de 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1. Los navegadores más antiguos soportan como mínimo ECMAScript 3. Una sexta revisión del estándar está en proceso (Mozilla y Colaboradores individuales 2015).

Entre las principales características del lenguaje se pueden encontrar:

- Es simple, no hace falta tener conocimientos avanzados de programación para poder hacer un programa en JavaScript.
- Maneja objetos dentro de la página web y sobre ese objeto se pueden definir diferentes eventos. Dichos objetos facilitan la programación de páginas interactivas.
- Es dinámico, responde a eventos en tiempo real.

[19] Término utilizado en programación para referirse al control del tipo de dato sobre las variables.
[20] Etapa en la que se traduce el código de alto nivel a bytecode.
[21] Gestor de base de datos no relacional de código abierto.

- El auge de las tecnologías web y el nacimiento de la web 2.0 a inicios del siglo XXI provocaron el incremento del uso de JavaScript en sitios web cada vez más dinámicos. Esta situación motivó al mismo tiempo el surgimiento de numerosas bibliotecas de JS que facilitan considerablemente el trabajo con los objetos y eventos del DOM[22].

JavaScript en el lado del servidor ha tomado fuerza desde el lanzamiento de Node.js en 2009, beneficiado por el alto rendimiento y rápida respuesta de Node.js en el procesamiento de peticiones, muchos de los proyectos que se inician hoy en día escogen a JS como lenguaje de programación del cliente y del servidor. Teniendo en cuenta esta tendencia y el bueno rendimiento de Node.js, JavaScript es elegido para ser empleado en el desarrollo de la solución.

### 1.4.5. Bibliotecas y marcos de trabajo para el desarrollo.

Un marco de trabajo simplifica considerablemente el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Proporciona una estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener, además de facilitar la programación de aplicaciones pues encapsula operaciones complejas en instrucciones sencillas. Por otro lado las librerías persiguen un objetivo similar, con la diferencia de que están orientadas a un uso más pasivo y usualmente no generan código fuente en la aplicación sino que el desarrollador hace uso de las funcionalidades que esta provee. A continuación se describen algunos de los principales marcos de trabajo de desarrollo para Node.js y librerías JavaScript.

#### 1.4.5.1. jQuery.

Es una biblioteca de JavaScript rápida y concisa que simplifica el uso del documento HTML, el manejo de eventos, la animación, y las interacciones Ajax para el desarrollo web rápido (The jQuery Foundation 2015).

Anunciado en 2006 por su creador, John Resig, jQuery ganó rápidamente popularidad y apoyo al convertirse en una nueva forma de utilizar JavaScript para interactuar con HTML y CSS. Los selectores de jQuery son simples, imitando selectores CSS, por lo que es una biblioteca familiar y fácil de aprender para los diseñadores y desarrolladores por igual. La librería jQuery borró la preocupación que los desarrolladores web habían sufrido al intentar crear sitios interactivos a través de una amplia gama de

[22] Siglas en inglés para Document Object Model. Es una interfaz de programación de aplicaciones (API) para documentos HTML.

navegadores por el manejo de la mayoría de los problemas de compatibilidad (Castledine y Sharkie 2010).

Aparte de ser muy fácil de usar, una de las mayores ventajas de jQuery es que maneja una gran cantidad de estándares para varios navegadores web. En la actualidad esta librería es ampliamente utilizada en disímiles aplicaciones web. Su reputación ha crecido exponencialmente al igual que la comunidad de desarrolladores que comparten sus experiencias así como *plugins* que complementan el trabajo.

### 1.4.5.2. AngularJS.

Es un marco de trabajo JavaScript de código abierto desarrollado por Google con el cual se pueden crear aplicaciones web basadas en AJAX[23]. Generalmente la complejidad en aplicaciones AJAX crece en gran escala y AngularJS trata de minimizar este problema ofreciendo un gran entorno para el desarrollo.

Una de las características más importantes de AngularJS es su capacidad para brindar una estructura a las aplicaciones web. También extiende el vocabulario de HTML para crear aplicaciones de forma declarativa, el resultado es un código limpio y de buen entendimiento.

AngularJS ofrece un gran número de características que son usadas día a día en la programación. Este soporta dos vías para la vinculación de datos, plantillas agradables, fácil interacción con REST[24], creación de componentes personalizados, enrutamiento y mucho más. Al mismo tiempo, AngularJS puede ser combinado con jQuery (Sandeep Panda 2014).

Las características ofrecidas por AngularJS en la creación de aplicaciones web lo convierten en la librería JavaScript robusta que facilita el desarrollo de productos informáticos y debido a las necesidades del proyecto es escogido como marco de trabajo en el lado del cliente en su versión 1.3.15.

### 1.4.5.3. MEAN.IO.

Es un completo marco de trabajo para JavaScript que simplifica y acelera el desarrollo de aplicaciones web. Sus siglas provienen de la unión de MongoDB, Express, Node.js y Angular, herramientas que este incluye. Está diseñado para dar una forma rápida y organizada de iniciar el desarrollo de aplicaciones web basadas en MEAN con módulos útiles como Mongoose y Passport, ambos ya configurados.

[23] Siglas en inglés para Asynchronous JavaScript and XML
[24] Siglas en inglés para Representational State Transfer

Express según sus creadores, es un marco de trabajo para el desarrollo de aplicaciones web minimalista y flexible para Node.js.

Entre sus principales características se incluye la presencia de paquetes, los cuales se pueden instalar y configurar en dependencia de lo que el desarrollador necesita (Linnovate 2014). La versión a emplear será la 0.8.

### 1.4.6. Herramientas IDE[25].

#### 1.4.6.1. Eclipse.

Es un entorno de desarrollo integrado (IDE), de código abierto y multiplataforma. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java.

Eclipse, al mismo tiempo, es una comunidad de código abierto, cuyos proyectos se centran en la construcción de una plataforma de desarrollo abierta compuesta por marcos extensibles, herramientas de tiempos de ejecución para la construcción, implementación y administración de *software* a través del ciclo de vida de los mismos. La Fundación Eclipse es una organización sin fines de lucro que aloja los proyectos de Eclipse y ayuda a cultivar tanto, una comunidad de código abierto y un ecosistema de productos y servicios complementarios (Eclipse Foundation 2015).

#### 1.4.6.2. NetBeans.

Es un entorno de desarrollo integrado de código abierto para desarrolladores de *software*. Cuenta con todas las herramientas necesarias para crear aplicaciones profesionales de escritorio, empresariales, web y aplicaciones móviles con la plataforma Java, así como con C/ C + +, PHP, JavaScript y Groovy (Oracle Corporation 2015b).

NetBeans recibe el soporte integrado para lenguajes dinámicos, todo en una herramienta de gran alcance. Provee de un resaltado de código sintáctico y semántico, documentación emergente, formateo de código y plegado, y el marcado de los sucesos y puntos de salida.

El IDE también ofrece un completo editor de HTML, JS y CSS, con la ventaja de proveer un resaltado de sintaxis completo, completamiento de código inteligente, documentación *pop-up*, y la comprobación

[25] Siglas en inglés para Integrated Development Environment

de errores para HTML, CSS y JavaScript, incluyendo HTML 5, JavaScript 1.7. El editor reconoce código HTML en los archivos de JavaScript y viceversa.

Teniendo en cuenta las características anteriormente planteadas, se selecciona el IDE NetBeans para desarrollar la solución propuesta, por todas las facilidades que brinda para el trabajo con proyectos JS y al soporte a tecnologías de última generación como HTML5. Se utilizará su versión 8.0.

### 1.4.7. Servidores web.

#### 1.4.7.1. Apache.

Lanzado en el año 1995, es un servidor web de código abierto, robusto, flexible, rápido y eficiente, continuamente actualizado, adaptado a los nuevos protocolos y cuya implementación se realiza de forma colaborativa. El proyecto está dirigido y controlado por un grupo de voluntarios de todo el mundo conocido como el Grupo Apache. Apache 2 es una profunda revisión del servidor Apache, centrándose en la escalabilidad, seguridad y rendimiento.

Apache es multiplataforma, posee un sistema de módulos dinámico que pueden ser habilitados y deshabilitados mediante la ejecución de un comando y después de haber reiniciado el proceso servidor. Sus módulos de multiprocesamiento (MPM[26]) en el manejo de peticiones permiten que cada solicitud sea atendida por un hilo o proceso separado y utiliza sockets sincrónicos (Kabir 2003).

Según W3Techs.com[27], es el servidor web más usado con un 58.4%, aunque estos números han caído debido al surgimiento y avance de novedosas soluciones como Nginx que emplean menos recursos y aportan mayor velocidad al procesar las peticiones del cliente.

#### 1.4.7.2. Nginx.

Fue creado por Ígor Sysóiev, el principal administrador de sistemas del buscador ruso Rambler. En 2002 empezó a crear su propio servidor y en 2004 publicó la primera versión de este producto. Se trata de un servidor web HTTP de código abierto, licenciado bajo la Licencia BSD simplificada.

Nginx es multiplataforma, utiliza una arquitectura basada en eventos que hace que su consumo de recursos crezca de forma predecible, puede ser utilizado como un proxy inverso con opciones de caché y tiene soporte para SSL, FastCGI, *streaming* de video, autenticación, compresión *gzip* y balanceo de carga (Nedelcu 2010).

[26] Siglas en inglés para Multi-Processing Modules
[27] http://w3techs.com/technologies/cross/web_server/ranking

Actualmente este servidor web, según W3Techs.com[28] es el segundo más utilizado en el mundo por detrás de Apache con un 23.3% y va captando adeptos cada vez más. Empresas como la compañía de TV online Hulu utilizan Nginx por su estabilidad y configuración simple. Otros usuarios, como Facebook y WordPress.com, lo utilizan porque la arquitectura asíncrona del servidor web deja una pequeña huella de memoria y bajo consumo de recursos, haciéndolo ideal para el manejo de múltiples y cambiantes páginas web activas.

### 1.4.7.3. Node.js.

Es una plataforma construida sobre el entorno de ejecución JavaScript V8 de Chrome para desarrollar fácilmente aplicaciones web rápidas y escalables. Node.js utiliza un manejador de eventos, modelo no bloqueador de entradas y salidas que permite ligereza y eficiencia, ideal para aplicaciones en tiempo real de datos intensivos que se ejecutan a través de dispositivos distribuidos (Joyent Inc. 2014).

Para el manejo de paquetes, Node.js utiliza NPM[29], un gestor de paquetes que se encarga de solucionar las dependencias de *software* de terceros en la aplicación. Mediante simples comandos, los desarrolladores pueden descargar los módulos que necesitan, para esto, es necesario tener una conexión a Internet.

En comparaciones realizadas por el sitio web Changeblog[30] entre Apache y Node.js en cuanto a velocidad, peticiones procesadas, consumo de memoria RAM[31] y CPU[32], dan como claro ganador a Node.js. En esta prueba se observa que Node.js es hasta 3 veces más rápido que Apache y solo un 2.5 % de las peticiones no son bien procesadas. El consumo de memoria y CPU es mayor pero esto no debe ser una sorpresa considerando el alto rendimiento.

La tabla que a continuación se muestra, refleja los resultados de las pruebas realizadas por Changeblog.

[28] http://w3techs.com/technologies/cross/web_server/ranking
[29] Siglas en inglés para Node Package Manager
[30] http://zgadzaj.com/benchmarking-nodejs-basic-performance-tests-against-apache-php
[31] Siglas en inglés para Random Access Memory
[32] Siglas en inglés para Central Unit Processor

| | Tiempo total (s) | Peticiones fallidas | Peticiones por segundos | Tiempo por peticiones (ms) | Velocidad de transferencia (kb/s) |
|---|---|---|---|---|---|
| **Apache** | 3570.753 | 2617614 | 280.5 | 3.571 | 10.7 |
| **Node.js** | 1043.076 | 25227 | 958.7 | 1.043 | 76.8 |

**Tabla 1**. Comparación de rendimiento entre Apache y Node.JS

Otras pruebas de rendimiento[33] también subrayan el rendimiento de Node.js sobre sus similares, en este caso Nginx, destacando la cantidad de transacciones por segundo que es capaz de procesar Node.js (el doble), y el mayor tiempo para una transacción tomado por Node.js, 1.10s contra 13.95s de Nginx.

Por todo lo mencionado anteriormente, se elige como servidor web a utilizar Node.js, principalmente por su velocidad al procesar peticiones y el alto rendimiento demostrado, necesario para sostener una aplicación pensada para ser empleada por más de 5000 personas y brindar una experiencia satisfactoria al usuario final. La versión a utilizar será la 0.12.

### 1.4.8. Sistemas gestores de base de datos.

#### 1.4.8.1. MySQL.

Es un sistema de gestión de base de datos multiplataforma de código abierto desarrollado, distribuido y soportado por Oracle Corporation. Fue creado en 1995 por David Axmark, Allan Larsson y Michael Widenius.

MySQL se caracteriza fundamentalmente por su velocidad, rendimiento y fiabilidad, motivo por el cual es ampliamente utilizado en sitios web y aplicaciones de escritorio de baja y mediana complejidad. Muchas de las organizaciones más grandes y de más rápido crecimiento del mundo, incluyendo Facebook, Google, Adobe, Alcatel Lucent y Zappos basan sus sistemas en MySQL.

Tanto el *software* del servidor de MySQL en sí y las bibliotecas de cliente utilizan una doble licencia de distribución. Se ofrecen bajo licencia GPL, a partir del 28 de junio de 2000 (que Oracle ha ampliado con una excepción de licencia de *software* libre) (Oracle Corporation 2015a) o una licencia propietaria.

[33] http://centminmod.com/siegebenchmarks/2013/020313/index.html

Precisamente por esta razón el uso de MySQL no es recomendado para la solución informática propuesta, puesto que Oracle puede retirar en cualquier momento la licencia GPL del DBMS y se estarían incumpliendo las políticas de soberanía tecnológica de la universidad y del país.

#### 1.4.8.2. PostgreSQL.

Es un avanzado servidor de base de datos SQL, disponible en una amplia gama de plataformas. Uno de los beneficios más claros de PostgreSQL es que es de código abierto, además de que sus versiones son mantenidas por largos periodos de tiempo y requiere poco o ningún mantenimiento en muchos casos (Riggs y Krosign 2010).

Entre las características fundamentales de PostgreSQL se pueden encontrar (Riggs y Krosign 2010):

- Excelente cumplimiento de los estándares SQL 2008.
- Arquitectura cliente-servidor.
- Diseño altamente concurrentes donde los lectores y escritores no se bloquean entre sí.
- Altamente configurable y extensible para muchos tipos de aplicaciones.
- Integridad referencial (*Foreign Keys*).
- Triggers Vistas (*Views*).
- Control de versionado concurrente (MVCC[34]).

PostgreSQL se caracteriza además por ser un sistema de base de datos de propósito general que permite crear sus propias funciones de servidor en una docena de idiomas diferentes. Es altamente extensible, de modo tal que es posible agregar sus propios tipos de datos, operadores y los tipos de índices.

#### 1.4.8.3. MongoDB.

MongoDB (de la palabra en inglés *humongous* que significa enorme) es un sistema de base de datos de código abierto que lidera el movimiento NoSQL. Está escrito en C++ y ofrece alto rendimiento, alta disponibilidad y fácil escalabilidad.

[34] Siglas en inglés para Multiversion concurrency control

Entre sus principales características se incluyen (Mongodb Inc. 2014):

- Flexibilidad: MongoDB almacena los datos en documentos JSON[35] (los cuales son serializados a BSON[36]). JSON provee un modelo de datos rico que asigna a la perfección a los tipos de lenguaje de programación nativos, y el esquema dinámico hace evolucionar fácilmente el modelo de datos más que con un sistema con esquemas forzados como RDBMS[37].

- Potencia: MongoDB proporciona una gran cantidad de características que los RDBMS tradicionales brindan: índices secundarios, consultas dinámicas, clasificación, ricas actualizaciones, *upserts* (actualiza si existe el documento, inserta si no lo existe), y fácil agregación. Esto le da la amplitud de funcionalidades a la que estamos acostumbrados en un RDBMS, con la capacidad de flexibilidad y escalado que el modelo no relacional permite.

- Velocidad/Escalado: Al mantener los datos relacionados juntos en documentos, las consultas pueden ser mucho más rápidas que en una base de datos relacional donde los datos relacionados son separados en múltiples tablas y luego tienen que ser unidos más tarde. MongoDB también hace que sea fácil de escalar su base de datos. *Autosharding* permite ampliar o reducir el clúster de forma lineal mediante la adición de más máquinas. Lo cual es muy importante en la web cuando la carga puede aumentar de repente y dejar sin respuesta al sitio web.

Al emplear MongoDB con Node.js, este aporta más velocidad al servidor web cuando se ejecutan operaciones sobre la base de datos y brinda flexibilidad al modelar la solución. Por lo antes mencionado se elige como sistema gestor de base de datos MongoDB. La versión a emplear será la 3.0.

## 1.5. Conclusiones.

El estudio de las principales tendencias de los sistemas de planificación arrojó que:

- Todas las soluciones son web, lo cual implica que a partir de las tendencias y las características de la solución propuesta se decide que la misma debe estar desarrollada en una plataforma web que facilite el acceso y la disponibilidad de la información.

[35] Siglas en inglés para JavaScript Object Notation
[36] Siglas en inglés para Binary JSON
[37] Siglas en inglés para Relational Database Management System

- El uso de estándares definidos por el Grupo de Trabajo de Ingeniería de Internet (IETF por sus siglas en inglés) posibilita una correcta comunicación con otros sistemas y clientes similares.

- Las soluciones presentes la UCI (SIPAC, GESPRO y HorrPublisher) presentan problemas de comunicación entre ellas, mientras que las más usadas en el mundo (Google y Yahoo!) son privadas y su código no está disponible para desplegarlos.

Como parte de la definición de la propuesta de los elementos que formarán parte de la arquitectura base de la solución, se plantea el uso de herramientas libres que estén acordes con las políticas tecnológicas de la universidad y del país. Una vez realizadas las investigaciones pertinentes, se concluye que el producto debe ser una aplicación web cuyo proceso de desarrollo debe ser guiado por la metodología ágil XP, se debe utilizar el lenguaje UML para estandarizar la descripción del proyecto y Visual Paradigm como herramienta CASE.

Como lenguaje de programación se selecciona JavaScript del lado del servidor y del cliente. Además se utilizará como servidor web Node.js 0.12, MEAN.IO 0.8 como marco de trabajo de desarrollo, la librería AngularJS 1.3.15. NetBeans 8.0 será el IDE empleado en el desarrollo de la solución y como gestor de base de datos se utilizará MongoDB 3.0.

# CAPÍTULO 2:

# DESARROLLO DE LA SOLUCIÓN

## 2.1. Introducción.

Una vez analizadas las soluciones similares y elegidas las herramientas y la metodología a utilizar, se procede a plantear una solución al problema. Para el desarrollo de la solución se siguieron los pasos definidos por la metodología seleccionada que guían el proceso de desarrollo de *software*. En el presente capítulo se identifican los requisitos funcionales y no funcionales del *software*, se describen las fases por las que transcurre el desarrollo de la herramienta: la de implementación y la de pruebas, haciendo referencia a todo lo concerniente a ellas, así como una descripción de cada uno de los artefactos generados.

Uno de los pasos fundamentales en la fase de exploración de la metodología XP es definir una metáfora del sistema. En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto se solventan con la existencia de una metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo.

Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. La práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema. Cuando el sistema es pequeño la metáfora es extremadamente sencilla (Letelier y Penadés 2004).

Teniendo en cuenta los parámetros anteriormente expuestos, la metáfora del sistema que se propone es: “una herramienta donde las personas del dominio uci.cu puedan sincronizar sus calendarios existentes en diferentes sistemas informáticos de forma sencilla que ayude elevar la precisión al planificar tiempo y recursos disponibles, y una mejor gestión del tiempo”.

## 2.2. Descripción de la propuesta de solución.

Teniendo en cuenta los objetivos que persigue el presente trabajo de diploma y en aras de utilizar la información recopilada en el capítulo anterior, se propone como solución desarrollar una aplicación web que sincronice de los calendarios existentes en la UCI, permitiendo una mejor planificación y control del tiempo.

KLNDA, nombre que recibe el nuevo sistema, presenta entre sus principales características:

- Sincronización entre aplicaciones para la planificación operativa.
- Gestión de conflictos.
- Notificaciones mediante correo y mensajería instantánea.

La solución informática es capaz de sincronizar actividades que se encuentran almacenados en diferentes sistemas existentes en la universidad a través de funcionalidades que estos brindan o haciendo *scraping*[38]. Esta acción se realiza de forma manual o automáticamente en dependencia de la configuración establecida por el usuario.

El proceso de sincronización transcurre por dos etapas fundamentales: extracción y procesamiento.

- Extracción: mediante el módulo Aplicaciones, KLNDA se conecta a los diferentes sistemas (proveedores) para obtener las actividades de los usuarios, en el caso específico de SIPAC esta acción se debe realizar de forma manual pues este no cuenta con servicios web que permitan el intercambio de información.
- Procesamiento: después de obtenidas las actividades se transforman a JSON para ser analizadas, mediante el UID[39] de cada elemento se comprueba si existe o no en el sistema: si el identificador no se encuentra entonces se añade sino se actualiza la instancia correspondiente. Luego se comparan todos los UID existentes en el sistema según el proveedor y si alguno de estos no coincide con los recibidos entonces se elimina porque ha sido eliminado en su sistema fuente.

[38] Tecnología utilizada para obtener datos de una página web simulando la acción humana.
[39] Identificador universalmente único, estándar usado en el desarrollo de *software* en sistemas distribuidos. RFC 4122 de la IETF http://tools.ietf.org/html/rfc4122

Las acciones de crear y modificar actividades dan paso a la gestión de conflictos y la introducción del concepto de precisión.

La gestión de conflictos presente en KLNDA parte del concepto de precisión al planificar tiempo definido por los autores de la investigación, el cual plantea que: *"en un mismo instante de tiempo no deben coincidir dos o más actividades para una misma persona".*

Bajo este principio, al crear o modificar una actividad el sistema siempre verifica que las personas asociadas estén disponibles en el rango de tiempo en el que se desea establecer dicha actividad notificando al creador la disponibilidad de cada una de ellas y proporcionando un "tiempo ideal" para llevarla a cabo en caso de que exista una afectación.

Este "tiempo ideal" tiene en cuenta las afectaciones en todos los calendarios del organizador y las personas implicadas en un día determinado y es generado a partir de la duración de la actividad.

En caso que una persona sea incluida en actividades que violan el concepto de precisión enunciado anteriormente se define la prioridad y por consiguiente la actividad a la que debería ir atendiendo a diversos criterios entre los que se encuentran: si el organizador es directivo, si es una asignación o una invitación y la categoría (académica, científica, productiva, extensionista y personal).

Mediante el uso del correo electrónico y la mensajería instantánea las personas se mantendrán informadas de las actividades a las que son invitadas y de los cambios que ocurran en ellas. Otra característica importante es la programación de alertas para recordar del advenimiento de una diligencia a las personas que deben asistir.

Desde su perfil los usuarios podrán especificar las vías por las que desean ser notificados y el tipo de notificación que desea recibir.

Al mismo tiempo, KLNDA es capaz de gestionar eventos y tareas en otros sistemas similares, si la actividad planificada pertenece a Zimbra, automáticamente se llevará a cabo la acción deseada en este. En el caso de SIPAC, el usuario tiene la opción de generar un archivo .sipac para luego importarlo en dicho sistema.

Al gestionar tiempo y comunicarse con otros sistemas, es vital brindar vías para que otros sistemas o clientes similares puedan obtener, insertar, modificar o eliminar eventos y tareas de los usuarios proporcionando su debida autenticación mediante usuario y contraseña o a través del intercambio de *tokens* de seguridad.

A través de los servicios web publicados se puede intercambiar información mediante tres vías, ellas son:

- iCal: estándar definido por la IETF.
- JSON: forma de representar objetos en JavaScript.
- SQLite: base de datos que contiene las actividades para SIPAC.

Desde la interfaz de la aplicación, las personas pueden exportar e importar su calendario en formato iCal o SIPAC.

La aplicación desarrollada puede ser utilizada desde cualquier estación sin limitaciones pues cuenta con diseño responsivo que se adapta al tamaño de la pantalla del dispositivo empleado.

Por último, el sistema mostró un alto rendimiento al procesar peticiones en pruebas realizadas. Los resultados de estas pruebas pueden ser vistas en el capítulo 3 de este documento.

Para llevar a cabo el desarrollo de la propuesta, se partió del análisis realizado, identificándose los siguientes requisitos funcionales:

F01- Gestionar eventos.

F02- Gestionar tareas.

F03- Gestionar conflictos entre actividades.

F04- Importar calendario en formato iCal.

F05- Exportar calendario en formato iCal.

F06- Importar calendario en una base de datos SQLite.

F07- Exportar calendario en una base de datos SQLite.

F08- Notificar a las personas sobre cambios en eventos y tareas mediante correo electrónico.

F09- Programar recordatorios.

F10- Obtener los datos del horario UCI.

F11- Gestionar eventos en Zimbra.

F12- Determinar tiempo libre de una persona.

F13- Proveer tiempo ideal para realizar una actividad.

F14- Obtener las estructuras de la UCI.

F15- Autenticar usuario mediante dominio UCI.

F16- Notificar de eventos próximos a su inicio mediante correo electrónico y mensajería instantánea.

F17- Notificar de tareas próximas a su inicio mediante correo electrónico y mensajería instantánea.

F18- Sugerir eventos y tareas a una persona.

F19- Sugerir eventos y tareas a una persona.

F20- Obtener personas desde el directorio activo de la UCI.

F20- Compartir eventos.

F21- Compartir tareas.

F22- Exportar calendario en formato .pdf.

F23- Exportar calendario en formato .doc.

F24- Generar *token* de seguridad.

F25- Brindar opciones de configuración.

F26- Buscar eventos.

F27- Buscar tareas.

F28- Brindar servicios web para el intercambio de información en formato iCal.

F29- Brindar servicios web para el intercambio de información en formato JSON.

F30- Brindar servicios web para el intercambio de información en base de datos SQLite.

## 2.3. Modelo conceptual.

Un modelo conceptual o modelo de dominio tiene como objetivo identificar y explicar los conceptos significativos en un dominio del problema, identificando los atributos y las asociaciones existentes entre ellos. Puede ser visto, también como una representación de las entidades, ideas, conceptos u objetos del "mundo real" o dominio de interés (Martínez 2010).

Después de analizar el contexto del sistema y apoyándose en Visual Paradigm y UML se obtuvo el siguiente modelo conceptual:

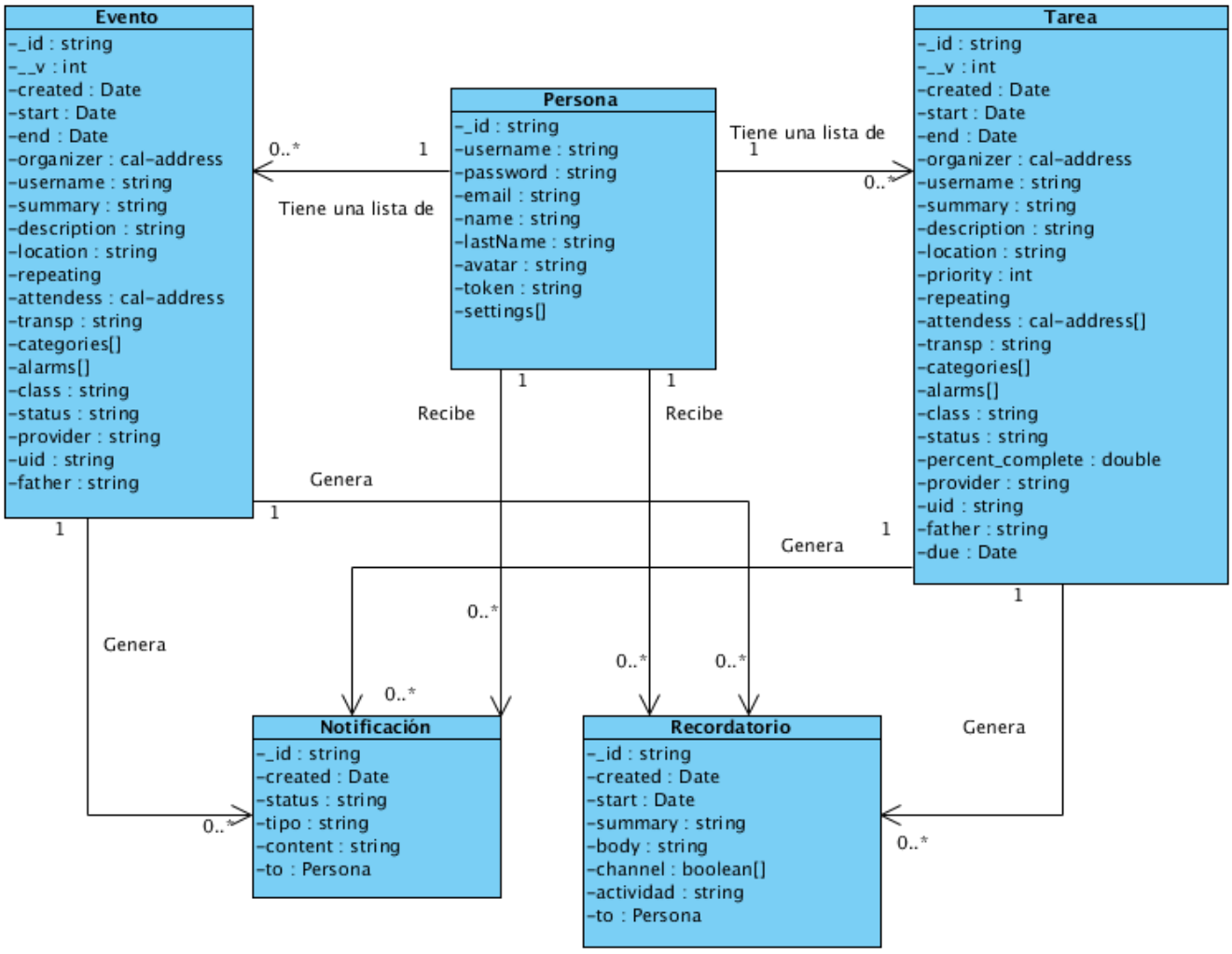


**Figura 1.** Diagrama del modelo conceptual de la solución

## 2.4. Patrón de arquitectura.

Un patrón de arquitectura expresa un esquema de organización estructural fundamental para sistemas de *software*. Proporciona un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y directrices para organizar las relaciones entre ellos (Sommerlad et al. 1996).

**Patrón Modelo-Vista-Controlador**

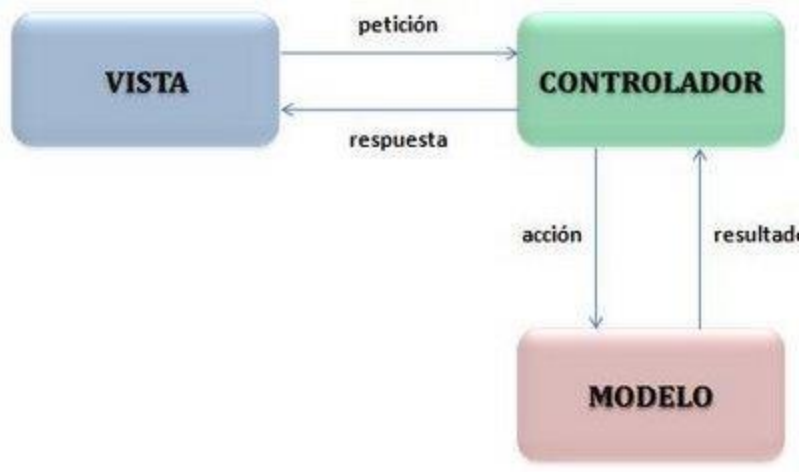


**Figura 2.** Modelo-Vista-Controlador (MVC).

El estilo arquitectónico o el patrón de arquitectura de *software* empleado para el desarrollo de la aplicación es la arquitectura MVC. Este separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento (López 2013).

De manera genérica, los componentes de MVC se definen de la siguiente manera:

- El Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones.

- El Controlador: Responde a eventos e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información.

- La Vista: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

Entre sus principales ventajas y razón por la que se escoge este patrón arquitectónico podemos encontrar:

- Clara separación entre interfaz, lógica de negocio y de presentación.
- Sencillez para crear distintas representaciones de los mismos datos.
- Reutilización de los componentes.
- Simplicidad en el mantenimiento de los sistemas.

## 2.5. Patrones de Diseño.

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. De manera más simple, un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos (Larman 2003).

Para el diseño de la aplicación se tuvieron en cuenta una serie de patrones de gran utilidad que proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifican y describen formas de solucionar problemas que ocurren de forma frecuente durante el desarrollo.

### 2.5.1. Patrones GRASP[40].

Los patrones GRASP, en español Patrones Generales de *Software* para Asignación de Responsabilidades, describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable (Larman 2003).

**Experto:** Asignar una responsabilidad al experto en información - la clase que tiene la información necesaria para realizar la responsabilidad (Larman 2003).

Este patrón se evidencia en la clase Evento, la cual contiene la información referente a los eventos.

**Creador:** Asignar a la clase B la responsabilidad de crear una instancia de clase A si se cumple uno o más de los casos: *B agrega objetos de A, B contiene objetos de A, B registra instancias de objetos de A, B utiliza más estrechamente objetos de A, B tiene los datos de inicialización que se pasarán a un objeto de A cuando sea creado* (por tanto, B es un Experto con respecto a la creación de A) (Larman 2003).

El uso de este patrón se evidencia en la clase Evento que crea instancias de la clase Recordatorio y Notificación.

**Controlador:** Asignar la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase que representa una de las siguientes opciones: *Representa el sistema global, dispositivo o subsistema (controlador de fachada), representa un escenario de caso de uso en el que tiene lugar el evento del sistema* (Larman 2003).

Este patrón se ve reflejado en la clase Evento. La arquitectura MVC brinda una capa específicamente para los controladores, que son el núcleo de este y especifica la presencia de este patrón.

### 2.5.2. Patrones GoF.

Los patrones GoF (Gang of Four, en español Pandilla de los Cuatro), se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento. Para el diseño de la herramienta se tuvieron en cuenta los siguientes patrones creacionales:

**Prototipo/Prototype:** crea nuevos objetos clonándolos de una instancia ya existente (Larman 2003).

Este patrón se ve reflejado en la clase Evento, específicamente en la funcionalidad encargada de duplicar un evento.

**Factoría/Factory:** designa quién debe ser el responsable de la creación de los objetos cuando existen consideraciones especiales, como una lógica de creación compleja, el deseo de separar las responsabilidades de la creación para mejorar la cohesión (Larman 2003).

Este patrón se ve reflejado en la clase Horario, específicamente en la funcionalidad encargada de realizar las peticiones para sincronizar el sistema con el horario UCI.

[40] Siglas en inglés para General Responsibility Assignment Software Patterns

**Adaptador/Wrapper:** resuelve problemas de interfaces incompatibles, proporciona una interfaz estable para componentes parecidos con diferentes interfaces (Larman 2003).

Este patrón se ve reflejado en la clase Evento, específicamente en la funcionalidad encargada de importar o exportar el calendario al formato iCal.

**Instancia única/Singleton:** admite exactamente una instancia de una clase - es un "singleton" -. Los objetos necesitan un único punto de acceso global (Larman 2003).

Este patrón se ve reflejado en la clase Mongoose, encargada de realizar la conexión con la base de datos.

## 2.6. Prototipos no funcionales.

Para mostrar una vista preliminar de la solución se crearon prototipos de interfaz no funcionales que básicamente incluyen las características definidas de la propuesta de solución, y son de fácil modificación a partir de la retroalimentación del cliente. Los prototipos iniciales sufrieron muchos cambios a través de las iteraciones en el desarrollo del sistema.

A continuación se muestra el prototipo no funcional de la interfaz añadir evento.

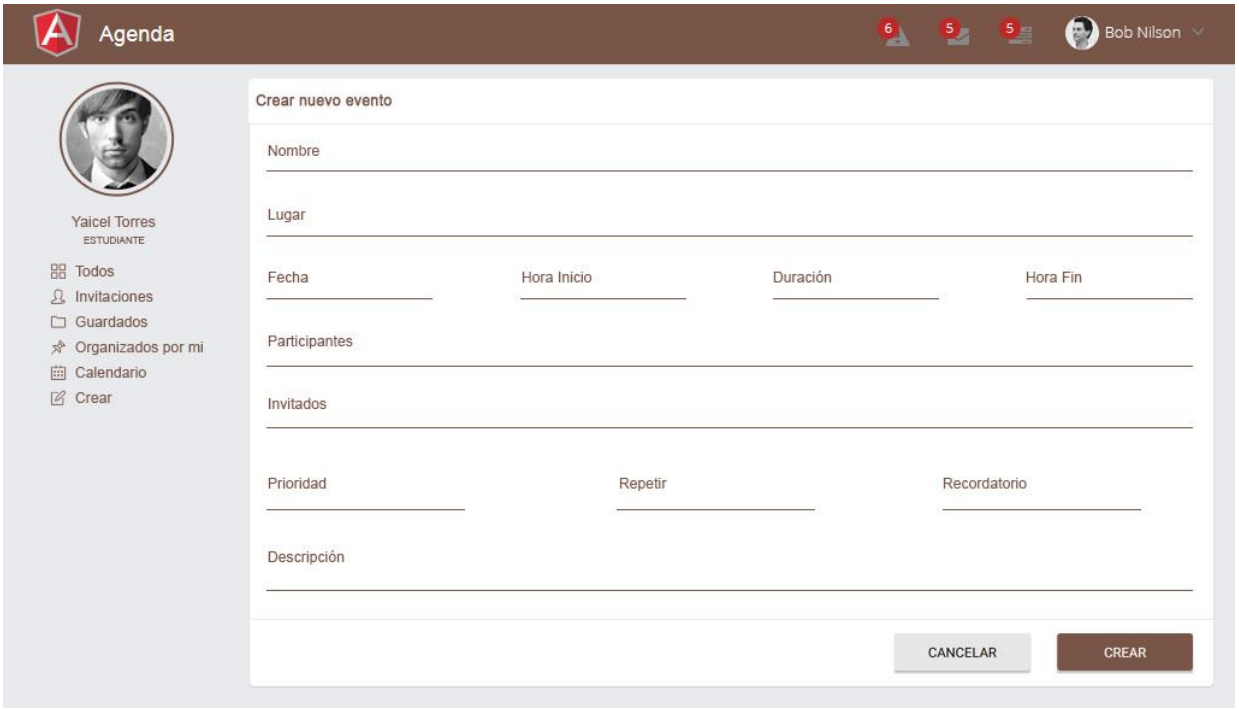


**Figura 3.** Prototipo no funcional añadir evento.

## 2.7. Ciclo de vida de la propuesta de solución.

El ciclo de vida de XP se enfatiza en el carácter iterativo e incremental del desarrollo. Una iteración de desarrollo es un período de tiempo en el que se realiza un conjunto de funcionalidades determinadas que en el caso de la metodología seleccionada corresponden a un conjunto de historias de usuario (HU). El ciclo de vida de XP consta de varias fases, a continuación se exponen algunas de ellas.

### 2.7.1. Fase de Planeación.

Durante esta fase se enmarca el alcance de la herramienta propuesta, que tiene como fin el desarrollo y la entrega de la misma. Para ello el cliente definió sus necesidades a través de las HU, a partir de las cuales los programadores estimaron el tiempo de desarrollo.

#### 2.7.1.1. Historias de usuario

Mediante las HU, escritas por el cliente en su propio lenguaje, se describe lo que la herramienta debe realizar. Las HU son la técnica utilizada para especificar los requisitos del *software* tanto funcionales como no funcionales. La diferencia más notable entre estas historias y los documentos de especificación de requisitos, utilizados por RUP, se encuentra en el nivel de detalle requerido. Dichas HU permiten a los programadores realizar una estimación poco riesgosa del tiempo que llevará el desarrollo (Pressman 2005).

Para definir las HU se utilizó la siguiente tabla, que contiene todos los datos necesarios para desarrollar la funcionalidad descrita.

| **Historia de Usuario** | |
|---|---|
| **Número:** *número de HU, incremental en el tiempo.* | **Nombre:** *el nombre de la HU, sirve para identificarla fácilmente entre los desarrolladores y los clientes.* |
| **Usuario:** *el usuario del sistema que utiliza o protagoniza la historia.* | |
| **Prioridad en negocio:** *que tan importante es para el cliente.* | **Riesgo en Desarrollo:** *que tan difícil es para el desarrollador.* |
| **Iteración Asignada:** *la iteración a la que corresponde.* | **Puntos estimados:** *estimación del tiempo de duración de la HU. Cuando el valor es 1* |

| | |
|---|---|
| | *equivale a una semana ideal de trabajo. En la metodología XP está definida una semana ideal como 5 días hábiles trabajando 40 horas, es decir, 8 horas diarias. Por lo que cuando el valor de dicho atributo es de 0.5 equivale a 2 días y medio de trabajo, lo que se traduce en 20 horas.* |
| **Descripción:** *la descripción de la historia, detallando las operaciones del usuario y opcionalmente las respuestas del sistema.* | |
| **Observaciones:** *algunas observaciones de interés, como glosario, información de usuarios, etc.* | |

**Tabla 2**. Representación de una HU.

Para las HU que describen las características y cualidades que debe cumplir la solución, los autores de la presente investigación decidieron excluir el aspecto: usuario.

Durante todo el proceso se identificaron 23 HU, las cuales se mencionan a continuación:

| No. | Historias de Usuario |
|---|---|
| 1 | Gestionar eventos. |
| 2 | Gestionar tareas. |
| 3 | Compartir eventos. |
| 4 | Compartir tareas. |
| 5 | Asignar eventos a una o un grupo de personas |
| 6 | Asignar tareas a una o un grupo de personas. |
| 7 | Invitar a eventos a una o un grupo de personas. |
| 8 | Invitar a tareas a una o un grupo de personas. |

| | |
|---|---|
| 9 | Buscar eventos. |
| 10 | Buscar tareas. |
| 11 | Sincronizar calendarios iCal. |
| 12 | Sincronizar horario docente UCI. |
| 13 | Exportar calendario personal en formato ICS (iCal) |
| 14 | Importar calendario en formato ICS (iCal) |
| 15 | Notificar eventos próximos a su inicio. |
| 16 | Notificar tareas próximas a su inicio. |
| 17 | Notificar eventos solapados. |
| 18 | Notificar tareas solapadas. |
| 19 | Determinar tiempo libre de una persona. |
| 20 | Sugerir eventos a una persona. |
| 21 | Brindar opciones de configuración. |
| 22 | Sincronizar SIPAC. |
| 23 | Proveer servicios web. |

**Tabla 3.** Historias de Usuario.

### 2.7.2. Fase de Diseño.

En esta fase se establecieron las tarjetas Clase-Responsabilidad-Colaborador (CRC) y la prioridad de cada historia de usuario por parte del cliente, momento a partir del cual los programadores realizaron una estimación del esfuerzo necesario para desarrollar cada una de ellas.

### 2.7.2.1. Tarjetas Clase-Responsabilidad-Colaborador

La tarjeta CRC es una técnica de modelado orientado a objetos que permite identificar las clases y sus responsabilidades. El nombre de la clase se coloca en forma de título en la tarjeta, en la parte izquierda las funcionalidades (responsabilidades) y en la parte derecha las clases que se implican en cada funcionalidad (colaboración). A continuación se muestra la plantilla que corresponde a las tarjetas CRC, las que permitieron definir y simular los escenarios que garantizan el buen funcionamiento del diseño:

| **Nombre de la clase** | |
|---|---|
| **Responsabilidad** | **Colaborador** |
| *Las HU que le corresponde desempeñar.* | *Las clases del sistema con que interactúa la clase especificada.* |

**Tabla 4.** Tarjeta CRC.

### 2.7.2.2. Estimación de esfuerzo por Historias de Usuario

Para el buen desarrollo de la herramienta propuesta, se realizó una estimación de cada una de las HU identificadas que definen las funcionalidades, la cual arrojó los resultados que se muestran a continuación:

| **Historia de Usuario** | **Estimación** |
|---|---|
| Gestionar eventos. | 1 |
| Gestionar tareas. | 1 |
| Compartir eventos. | 1 |
| Compartir tareas. | 1 |
| Asignar eventos a una o un grupo de personas | 1 |
| Asignar tareas a una o un grupo de personas. | 1 |
| Invitar a eventos a una o un grupo de personas. | 1 |

| | |
|---|---|
| Invitar a tareas a una o un grupo de personas. | 1 |
| Buscar eventos. | 1 |
| Buscar tareas. | 1 |
| Sincronizar calendarios iCal. | 2 |
| Sincronizar horario docente UCI. | 2 |
| Exportar calendario personal en formato ICS (iCal) | 2 |
| Importar calendario en formato ICS (iCal) | 2 |
| Notificar eventos próximos a su inicio. | 2 |
| Notificar tareas próximas a su inicio. | 1 |
| Notificar eventos solapados. | 1 |
| Notificar tareas solapadas. | 1 |
| Determinar tiempo libre de una persona. | 0.2 |
| Sugerir eventos a una persona. | 1 |
| Brindar opciones de configuración. | 1 |
| Sincronizar SIPAC. | 2 |
| Proveer servicios web. | 2 |

**Tabla 5**. Estimación de esfuerzo por HU.

### 2.7.2.3. Plan de iteraciones

Una vez descritas las HU que definen las funcionalidades por parte del cliente y estimado el esfuerzo por los desarrolladores para la realización de las mismas, se procede a realizar la planificación de la etapa de implementación del sistema. Este plan define las HU que serán implementadas en cada iteración y las posibles fechas de sus entregas. En relación con lo antes mencionado se decide implementar el sistema en tres iteraciones, las cuales se describen a continuación:

Iteración 1: En esta iteración se implementan las HU que tienen prioridad alta en el negocio, de esta forma, se van creando las funcionalidades principales de la herramienta que dan soporte a la implementación de las demás funcionalidades. Estas HU (de la 1 a la 9) hacen alusión de modo general al código base del sistema para permitir la gestión de eventos y tareas, actividades fundamentales del negocio. Al finalizar dicha iteración se realizará la primera entrega de la herramienta con el fin de recibir retroalimentación por parte del cliente.

Iteración 2: Esta iteración tiene como objetivo realizar el segundo grupo de funcionalidades requeridas por la herramienta. Estas nuevas características añaden la posibilidad de comunicarse con otros sistemas, obtener sus datos y modificarlos si es posible. Al finalizarla se tendrán implementadas las peticiones del cliente descritas en las HU (de la 10 a la 19) y se contará con la versión final del producto. A partir de este momento la herramienta será puesta a prueba para evaluar el desempeño de la misma y corregir no conformidades.

Iteración 3: Esta iteración tiene como objetivo llevar a cabo las restantes funcionalidades requeridas por la herramienta. Al finalizarla se tendrán implementadas las peticiones del cliente descritas en las HU (de la 20 a la 23) y se contará con la versión final candidata del producto. A partir de este momento la herramienta comenzará su etapa final de pruebas y pulir impurezas.

#### 2.7.2.4. Plan de duración de las iteraciones

| Iteración | Historias de Usuario | Duración total de la iteración |
|---|---|---|
| Iteración 1 | – Gestionar eventos.<br>– Gestionar tareas.<br>– Compartir eventos.<br>– Compartir tareas.<br>– Asignar eventos a una o un grupo de personas<br>– Asignar tareas a una o un grupo de personas.<br>– Invitar a eventos a una o un grupo de personas.<br>– Invitar a tareas a una o un grupo de personas. | 8 semanas |

| | | |
|---|---|---|
| | – Buscar eventos. | |
| Iteración 2 | – Buscar tareas.<br>– Sincronizar calendarios iCal.<br>– Sincronizar horario docente UCI.<br>– Exportar calendario personal en formato ICS (iCal).<br>– Importar calendario en formato ICS (iCal).<br>– Notificar eventos próximos a su inicio.<br>– Notificar tareas próximas a su inicio.<br>– Notificar eventos solapados.<br>– Notificar tareas solapadas.<br>– Determinar tiempo libre de una persona. | 12 semanas |
| Iteración 3 | – Sugerir eventos a una persona.<br>– Brindar opciones de configuración.<br>– Sincronizar SIPAC.<br>– Proveer servicios web. | 4 semanas |

**Tabla 6.** Plan de duración de las iteraciones.

### 2.7.2.5. Plan de entregas

El propósito del plan de entregas es establecer las HU que serán agrupadas para conformar una entrega y el orden de las mismas. En este plan se unen las funcionalidades referentes a un mismo tema en paquetes, esto permite un mayor entendimiento en la fase de implementación. A continuación se presenta el plan de entrega elaborado para la fase de implementación:

| **Paquete** | **Historias de Usuario** |
|---|---|
| Evento | Gestionar eventos. |
| Tarea | Gestionar tareas. |
| Evento | Compartir eventos. |
| Tarea | Compartir tareas. |

| | |
|---|---|
| Evento | Asignar eventos a una o un grupo de personas |
| Tarea | Asignar tareas a una o un grupo de personas. |
| Evento | Invitar a eventos a una o un grupo de personas. |
| Tarea | Invitar a tareas a una o un grupo de personas. |
| Evento | Buscar eventos. |
| Evento | Buscar tareas. |
| Aplicaciones | Sincronizar calendarios iCal. |
| Aplicaciones | Sincronizar horario docente UCI. |
| Evento, Tarea | Exportar calendario personal en formato ICS (iCal). |
| Evento, Tarea | Importar calendario en formato ICS (iCal). |
| Centro de Notificaciones | Notificar eventos próximos a su inicio. |
| Centro de Notificaciones | Notificar tareas próximas a su inicio. |
| Centro de Notificaciones, Integrador | Notificar eventos solapados. |
| Centro de Notificaciones, Integrador | Notificar tareas solapadas. |
| Integrador | Determinar tiempo libre de una persona. |
| Integrador | Sugerir eventos a una persona. |
| Usuario | Brindar opciones de configuración. |
| Aplicaciones | Sincronizar SIPAC. |

| | |
|---|---|
| Servicios | Proveer servicios web. |

**Tabla 7.** Funcionalidades por paquetes.

| **Paquetes** | **Final 1ra iteración (1ra quincena de marzo)** | **Final 2da iteración (2da quincena de abril)** | **Final 3ra iteración (1ra quincena de mayo)** |
|---|---|---|---|
| Evento | v0.1 | v1.0 Final | Finalizado |
| Tarea | v0.1 | v1.0 Final | Finalizado |
| Aplicaciones | - | v0.1 | v1.0 Final |
| Servicios | - | - | v1.0 Final |
| Centro de Notificaciones | v0.1 | v1.0 Final | Finalizado |
| Estructuras | - | v1.0 Final | |
| Integrador | - | - | v1.0 Final |
| Usuario | v0.1 | v0.1 | v1.0 Final |
| Sistema Base | v0.1 | v0.1 | v1.0 Final |

**Tabla 8**. Plan de duración de entrega.

### 2.7.3. Fase de Codificación.

En esta fase se realiza la implementación de las HU que definen las funcionalidades que fueron seleccionadas por cada iteración. Al inicio se lleva a cabo un chequeo del plan de iteraciones por si es necesario realizar modificaciones.

### 2.7.3.1. Estándares de codificación.

En la actualidad existen diferentes estándares de codificación cuya utilización favorece la comunicación fluida y directa entre los desarrolladores, permitiendo el aumento de la reutilización y el mantenimiento de los sistemas.

Para implementar la solución propuesta se hacen uso de los siguientes estándares establecidos por los autores de esta investigación. Dichos estándares se describen a continuación:

Nombre de las funciones (métodos o funcionalidades de los controladores):

- Dentro de las especificaciones del marco de trabajo está que cada una de los controladores deben finalizar con la palabra Controller. (ejemplo: EventController).
- En el caso de los controladores de la vista, los nombres de acciones deben comenzar con $scope. seguido por su nombre en notación lowerCamelCase. (ejemplo: $scope.create).
- En el caso de los controladores del servidor, los nombres de acciones deben comenzar con exports. seguido por su nombre en notación lowerCamelCase. (ejemplo: exports.synchronizeCalendar).
- Los nombres de las acciones deben especificar con la menor cantidad de palabras cual es el objetivo de la acción, de ser posible estar en infinitivo.
- Antes de cada función se debe especificar su propósito a través de comentarios por vías que el desarrollador desee. Se recomienda usar: /** nueva línea * nueva línea */.

Nombres de los modelos:

- Los nombres de los modelos deben estar expresados en notación UpperCamelCase y seguidos de la palabra Schema. (ejemplo: EventSchema).
- No se deben utilizar guiones bajos en su nombre.
- Deben expresar con claridad cuál es el alcance y la responsabilidad de la clase.
- Los nombres de las clases no deben estar atados a las clases de las que se deriva, cada clase debe tener un significado por ella misma, no en dependencia de la clase de la que deriva.

Variables:

- Los nombres de las variables deben expresar claramente el contenido de la misma.
- Deben seguir la nomenclatura lowerCamelCase.
- Pueden estar referidas en singular o plural.
- Se definen al principio de las estructuras donde son utilizadas.
- En caso de que no se le asigne un valor inicial se deben inicializar con un valor que indique el tipo de dato más general al que debe pertenecer.

### 2.7.3.2. Tratamiento de errores.

Un aspecto importante a tener en cuenta a la hora de desarrollar un software es el tratamiento de errores, tanto los generados por los usuarios como los producidos por el sistema.

Las validaciones del negocio se encargan de controlar el flujo de los datos recibidos en el controlador para evitar la inconsistencia de la información almacenada en la base de datos.

Las validaciones realizadas en las vistas juegan un papel primordial en la entrada correcta de los datos a la aplicación. Tienen como función principal evitar que se introduzcan datos incorrectos en los diferentes campos de los formularios para impedir que datos inconsistentes o incorrectos lleguen al servidor. Con el objetivo de detectar y mostrar las alertas y mensajes de notificación de errores en la interfaz del sistema se utilizan las validaciones que provee AngularJS.

Por otro lado en cada acción implementada en MEAN.IO se realiza un riguroso escrutinio de los valores de las variables que llegan a la funcionalidad en el servidor y en caso de detectarse errores son enviados en forma de cadena JSON a la interfaz para que sean notificados al usuario.

## 2.8. Conclusiones.

Al finalizar el capítulo se arriba a las siguientes conclusiones:

- El conjunto de artefactos propuestos por las fases de Planeación, Diseño y Codificación según la metodología XP permitieron establecer claridad en aspectos sumamente importantes como el alcance de la herramienta, el tiempo estimado para implementar cada HU y los planes de entregas de las iteraciones.

- La descripción se la solución a desarrollar propició una mejor comprensión de las características con que cuenta KLNDA.
- El modelo conceptual facilitó comprender y conocer aún más los principales conceptos y relaciones del nuevo sistema para sincronizar calendarios.
- La utilización de patrones de diseño favorece la reutilización de clases existentes, la creación de una estructura sólida y minimizan el riesgo de generar un mal diseño.
- Los estándares de codificación empleados posibilitaron un estilo homogéneo fácil de entender en menos tiempo y que el código en consecuencia sea mantenible.

# CAPÍTULO 3:

# VALIDACIÓN DE LA PROPUESTA

## 3.1. Introducción.

El desarrollo de un producto informático es un proceso altamente complejo y no está exento a la existencia de errores, por lo que el desarrollo del mismo debe estar acompañado de actividades o subprocesos que garantice la calidad. Entre estas actividades podemos mencionar la gestión de riesgos y las pruebas.

Las pruebas de *software* son el proceso de identificar las fallas o defectos en el sistema y asegúrese de que el mismo cumple con los requisitos del cliente antes de lanzarlo al mercado. Con el fin de mejorar la calidad del producto final, ciertos protocolos y reglas necesitan ser establecidos para complementar la eficiencia del *software*.

En este capítulo se muestra el diagrama de despliegue de la solución y el resultado de las pruebas aplicadas, de manera que se pueda comprobar la eficiencia de las clases u operaciones utilizadas para dar respuesta a las necesidades de los clientes.

## 3.2. Diagrama de despliegue.

A pesar de que la metodología utilizada no define ningún artefacto del proceso de desarrollo para visualizar la relación de los elementos que arman un sistema, los autores de la presente investigación consideran útil mantener un diagrama que modela la disposición física de los distintos nodos que componen un sistema y sus relaciones.

Los elementos de diseño al nivel de despliegue asignan la arquitectura, sus componentes y las interfaces a la configuración física que albergará el software. Esto significa que el diagrama de despliegue muestra el entorno computacional, pero no indica de manera explícita detalles de configuración (Pressman 2005).

A continuación se muestra el diagrama de despliegue para KLNDA.

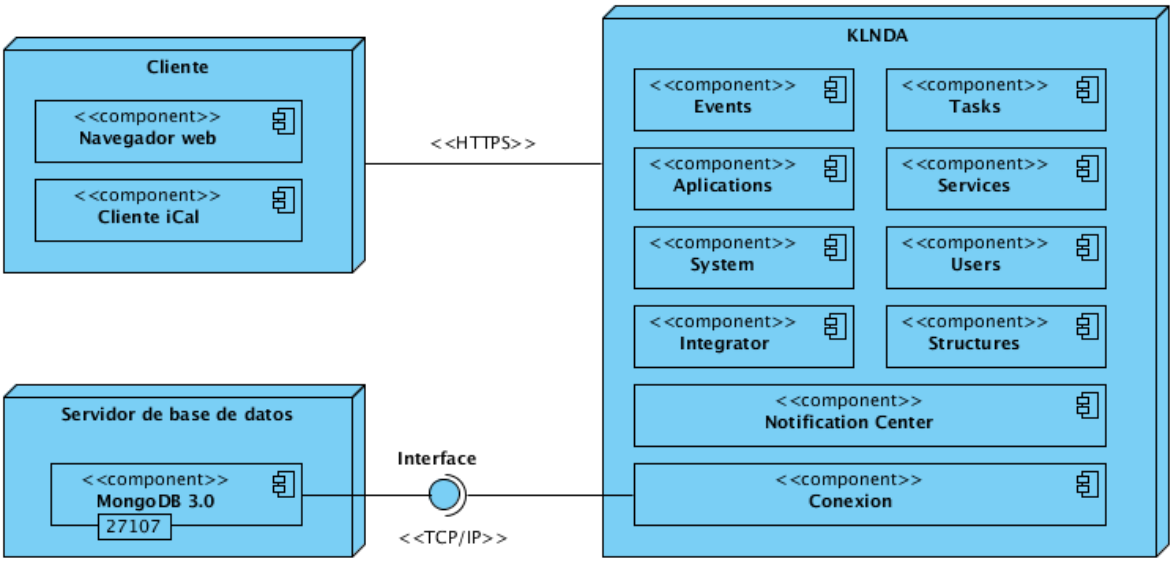


**Figura 4.** Diagrama de despliegue de la solución.

## 3.3. Fase de pruebas.

En XP uno de los pilares más importantes es el proceso de pruebas, en el cual los desarrolladores prueban tanto como sea posible. Mediante esta filosofía se reduce el número de errores no detectados, así como el tiempo de introducción de este en el sistema y su detección. Todo esto permite una mejor calidad en los productos desarrollados y la seguridad de los programadores a la hora de introducir algún cambio o modificación en el sistema.

La metodología XP divide las pruebas en dos grupos: las pruebas unitarias, que son realizadas por los programadores, encargadas de verificar el código y las pruebas de aceptación, destinadas a verificar si al final de cada iteración se obtuvo la funcionalidad que se requiere, además de comprobar que dicha funcionalidad haga lo que el cliente quiere (Cuervo 2014).

### 3.3.1. Pruebas unitarias.

Las pruebas unitarias o también llamadas pruebas de caja blanca se basan en realizar pruebas al código del sistema. Para llevar a cabo esta tarea se comprueban los caminos lógicos de la aplicación mediante casos de prueba, que pongan a prueba los algoritmos implementados. Las pruebas unitarias no se le pueden realizar a todo el código de la aplicación, ya que el número de caminos lógicos puede llegar a crecer de manera exponencial lo cual imposibilita realizar casos de pruebas para todos estos

caminos y mucho menos procesarlos todos (Jeffries 1999). Por este motivo las pruebas de caja blanca se realizan a los principales algoritmos o procedimientos.

Uno de los métodos o técnicas de prueba unitarias es el camino básico. Esta técnica permite obtener una medida de la complejidad de un procedimiento o algoritmo y un conjunto básico de caminos de ejecución de este, los cuales luego se utilizan para obtener los casos de prueba.

Esta técnica será la utilizada para desarrollar los casos de pruebas unitarias de la herramienta desarrollada. De igual manera existen varias métricas de *software* para realizar pruebas unitarias, entre estas se encuentra la complejidad ciclomática, la cual será utilizada junto a la técnica explicada anteriormente. Esta métrica proporciona una medición cuantitativa de la complejidad lógica de un procedimiento.

La complejidad ciclomática cuando se utiliza en el contexto del método de prueba del camino básico, el valor que se calcula como complejidad ciclomática define el número de caminos independientes dentro de un fragmento de código y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Pressman 2005).

Para realizar las pruebas unitarias en Node.js existe mocha[41], una herramienta rica en funciones que se ejecutan tanto en el servidor como en el navegador. Mocha ejecuta los casos de prueba en serie, lo cual permite la presentación de informes flexibles y precisos (TJ Holowaychuk 2015).

A continuación se explica, mediante un ejemplo, todo el procedimiento que se siguió para obtener los casos de prueba utilizando la técnica del camino básico junto con la métrica complejidad ciclomática. El algoritmo escogido para aplicar estas pruebas fue *crearEvento()* el cual tiene como responsabilidad la creación de un nuevo evento en el sistema para el organizador y las personas que deben asistir, así como la notificación de invitación, recordatorios y otras instancias en caso de ser repetitivo.

[41] Marco de trabajo para realizar pruebas unitarias en Node.js http://mochajs.org/

A partir del algoritmo, se obtuvo el siguiente grafo de flujo asociado.

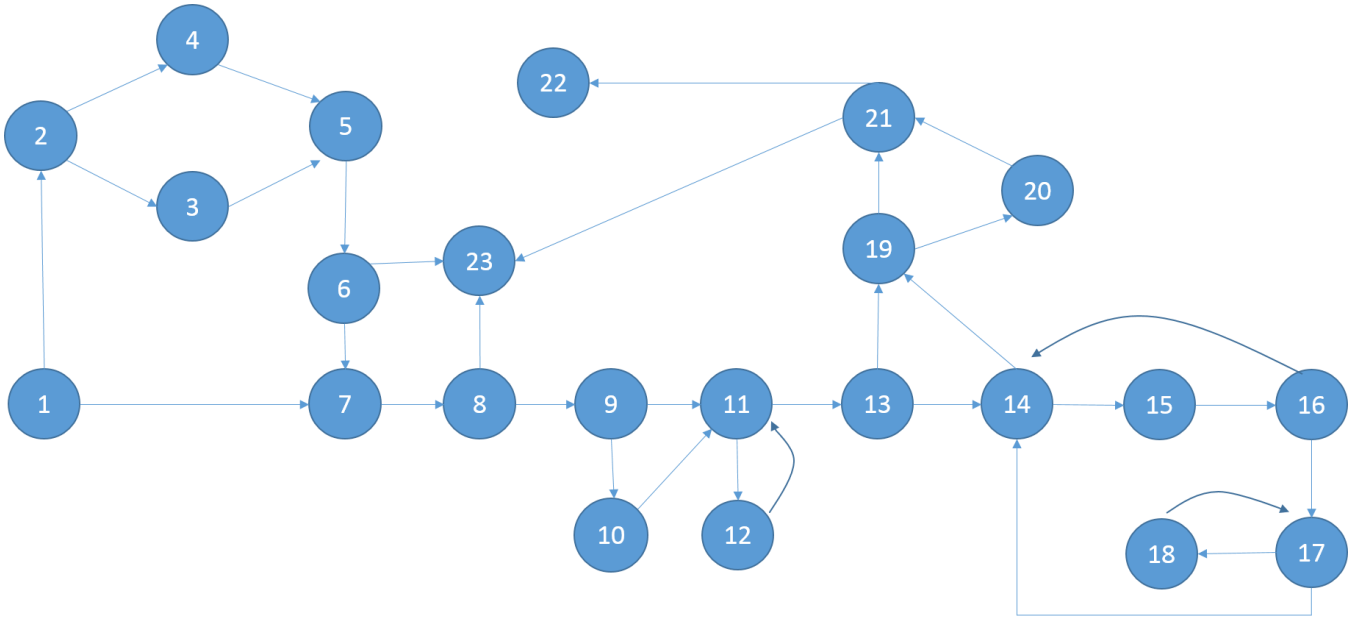


**Figura 5.** Grafo de flujo de la funcionalidad *crearEvento*.

Luego se determina la complejidad ciclomática V(G) del grafo resultante G, para ello se recurrió a las siguientes fórmulas:

1) V(G) = A – N + 2 (donde A es el número de aristas del grafo y N el número de nodos).

2) V(G) = P + 1 (donde P es la cantidad nodos Predicados, un nodo Predicado es aquel del cual parten 2 o más aristas).

3) V(G) = número de regiones de la gráfica del flujo.

Despejando las fórmulas en el grafo obtenido en 1) se tiene que:

V(G) = A – N + 2

V(G) = 34 – 23 + 2

V(G) = 13

Despejando las fórmulas en el grafo obtenido en 2) se tiene que:

V(G) = P + 1

V(G) = 12 + 1

V(G) = 13

El número de regiones de la gráfica se puede observar que es trece (13) y que la complejidad ciclomática[42] en todos los casos es igual, por lo tanto podemos afirmar que trece (13) son los caminos independientes. Los caminos independientes son:

**Camino 1:** 1, 2, 3, 5, 6, 23

**Camino 2:** 1, 2, 3, 5, 6, 7, 8, 23

**Camino 3:** 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 19, 20, 21, 22

**Camino 4:** 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 13, 19, 20, 21, 22

**Camino 5:** 1, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 14, 19, 21, 22

**Camino 6:** 1, 7, 8, 9, 11, 12, 13, 14, 15, 16, 14, 19, 20, 21, 23

**Camino 7:** 1, 7, 8, 9, 11, 12, 13, 14, 19, 20, 21, 22

**Camino 8:** 1, 2, 4, 6, 7, 8, 9, 11, 13, 19, 21, 23

**Camino 9:** 1, 2, 4, 6, 7, 8, 9, 10, 11, 13, 19, 20, 21, 22

**Camino 10:** 1, 2, 4, 6, 7, 8, 9, 10, 11, 13, 14, 19, 20, 21, 22

**Camino 11:** 1, 2, 4, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 14, 19, 20, 21, 22

**Camino 12:** 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 14, 19, 20, 21, 22

**Camino 13:** 1, 7, 8, 23

Un camino exitoso sería aquel que finaliza en el nodo veintidós (22) y donde el evento se ha creado satisfactoriamente, uno fallido sería aquel que concluye en el nodo veintitrés (23) y significa que ocurrió un error crear el evento. Luego se definieron trece (13) casos de prueba para el código del algoritmo, uno para cada camino. Para realizar estos casos de prueba se utilizó la siguiente plantilla.

| Caso de prueba unitaria |
|---|
| **Camino:** *nombre.* |
| **Caso de prueba:** *nombre.* |
| **Entrada:** *descripción textual de lo que ocurre en el mundo real que hace necesario ejecutar el caso de prueba, precisando la data de entrada y los comandos a dar por el actor. Descripción textual del estado de la información almacenada.* |
| **Resultado:** *descripción textual del estado en el que queda la información y las alertas que puedan generarse, una vez ejecutado el caso de uso con los valores y el estado especificado en la entrada.* |
| **Condiciones:** *condiciones que deben cumplirse mientras se ejecuta el caso de prueba.* |

**Tabla 9.** Caso de prueba unitaria

### Resultados de las pruebas unitarias

Una vez detectados los trece (13) caminos básicos aplicando dicha prueba, estos sirvieron como entrada para la confección de los tests en mocha y medir el tiempo de respuesta del sistema.

| Camino | Tiempo (ms) | Resultado esperado |
|---|---|---|
| 1 | 45 | Si |
| 2 | 59 | Si |
| 3 | 89 | Si |
| 4 | 68 | Si |
| 5 | 90 | Si |

[42] Métrica de calidad de software basada en el cálculo del número de caminos independientes que tiene nuestro código (Oriente 2012)

| | | |
|---|---|---|
| 6 | 74 | Si |
| 7 | 66 | Si |
| 8 | 72 | Si |
| 9 | 78 | Si |
| 10 | 90 | Si |
| 11 | 96 | Si |
| 12 | 84 | Si |
| 13 | 53 | Si |

**Tabla 10.** Resultados de las pruebas unitarias

### 3.3.2. Pruebas de aceptación.

Las pruebas de aceptación son un tipo de prueba de caja negra orientadas a evaluar las distintas tareas en las que ha sido dividida una HU. Para asegurar el funcionamiento final de una determinada historia estos experimentos son creados y usados por los clientes para comprobar que estas cumplen su cometido.

Un relato de usuario puede tener una o varias pruebas de aceptación y no está completo hasta no haberlas pasado todas. El cliente es el máximo responsable de verificar cada una de las pruebas y de priorizar la corrección de las pruebas fallidas.

La utilización de estas pruebas fue de vital importancia en el proceso de desarrollo ya que permitió a los programadores tener una idea más clara e inequívoca de la calidad del trabajo, a la vez se garantizó la entrega de un producto en elevada correspondencia con las necesidades del usuario final.

A continuación se muestra la plantilla de caso de prueba de aceptación definida por el cliente:

| **Caso de prueba de aceptación** | |
|---|---|
| **Código:** *código que identifica la prueba.* | **Historia de Usuario:** *número de la historia de usuario relacionada con la prueba que se realiza.* |
| **Nombre:** *definición de la prueba que se realiza.* | |
| **Descripción:** *breve descripción del objetivo con que se realiza la prueba.* | |
| **Condiciones de ejecución:** *condiciones que deben satisfacerse para que pueda realizarse la prueba.* | |
| **Entrada/Pasos de ejecución:** *se describen los pasos de ejecución de la prueba en cuestión.* | |
| **Resultado esperado:** *proporciona las expectativas ideales para las cuales fue pensada la prueba.* | |
| **Evaluación de la prueba:** *calificación que recibe la prueba de acuerdo con los resultados obtenidos.* | |

**Tabla 11.** Caso de prueba de aceptación

Iteración 1:

Para la primera iteración se realizaron catorce (14) casos de pruebas de aceptación, identificando cinco (5) no conformidades, cuatro (4) no significativas y una significativa. La mayoría de estas no conformidades presentaban complejidad mínima y fueron resueltas tres de ellas, quedando pendientes dos de ellas para la siguiente iteración.

Iteración 2:

Para esta iteración se realizaron 18 casos de pruebas de aceptación, identificando seis (6) no conformidades, de ellas dos (2) son significativas y cuatro (4) no significativa. Cinco de ellas fueron resueltas y solamente una pendiente al comenzar la tercera y última iteración.

Iteración 3:

Para esta iteración se realizaron ocho (8) casos de pruebas de aceptación, identificando tres (3) no conformidades, todas son significativas y solucionadas. Al finalizar la corrección de estas no conformidades se puso fin al proceso de pruebas de aceptación, obteniendo resultados satisfactorios.

Las no conformidades, no significativas, se centraron en errores ortográficos como: omisiones de tildes, paréntesis y no envío de notificaciones, y las significativas, en errores relacionados con la gestión de eventos y tareas.

**Resultados de las pruebas de aceptación.**

| | **Iteración 1** | **Iteración 2** | **Iteración 3** |
|---|---|---|---|
| CP de Aceptación | 14 | 18 | 8 |
| No conformidades | 5 | 6 | 3 |
| Significativas | 1 | 2 | 0 |
| No Significativas | 4 | 4 | 3 |
| Resueltas | 3 | 5 | 3 |
| Pendientes | 2 | 1 | 0 |

**Tabla 12.** Resultados de las pruebas de aceptación

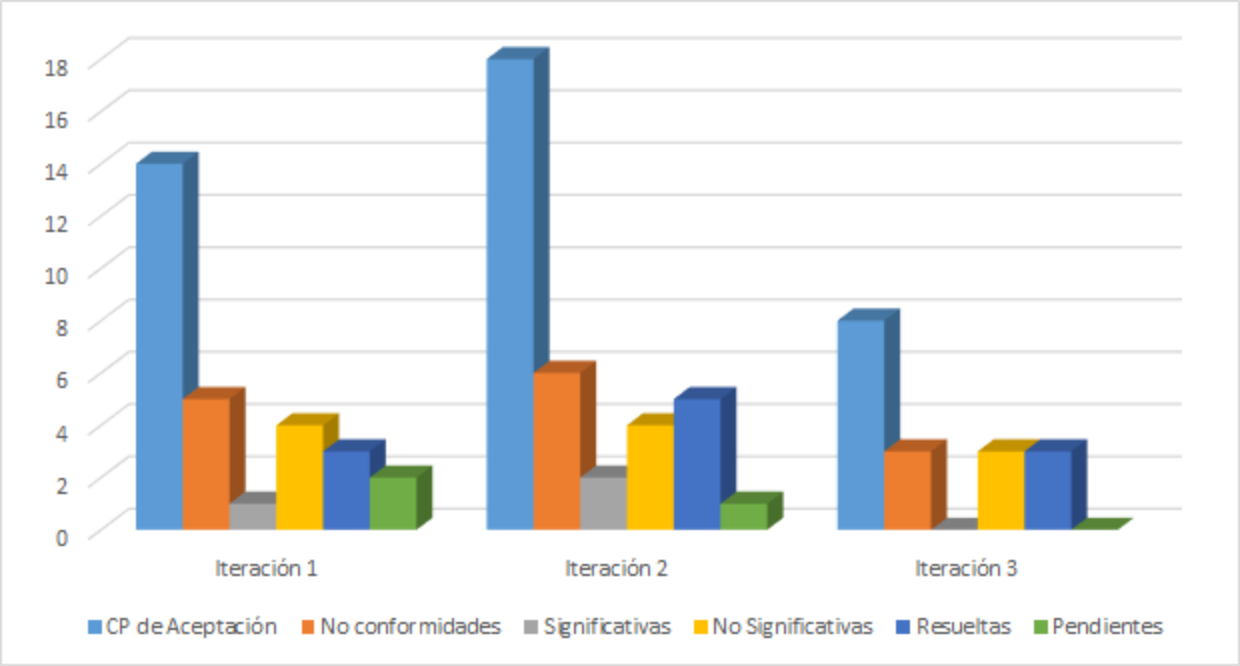


**Figura 6.** Resultados de las pruebas de aceptación.

### 3.3.3. Pruebas de validez.

Luego de haber realizado las pruebas que define la metodología de desarrollo de software seleccionada, se decide además, acorde a las peticiones del cliente realizar pruebas en un ambiente real y aplicar la prueba estadística de McNemar para medir la magnitud de los cambios. El objetivo principal de este tipo de pruebas es garantizarle al cliente un producto terminado que posea como principal característica la confiabilidad.

**Ejecución de la Prueba Piloto.**

Para ejecutar las pruebas se escogió un proyecto priorizado de la Facultad 3, en la misma participaron 19 profesionales que interactuaron con el sistema durante dos horas. Ello permitió medir el comportamiento del sistema en cuanto a: tiempo de respuesta del servidor, tiempo de cargar en el navegador, consumo de memoria RAM y uso del CPU.

Como servidor se empleó una computadora portátil VIT modelo P2402 con las siguientes características:

- Microprocesador Intel Corei3 de tercera generación a 3.1 GHz de velocidad.

- 2 GB de memoria RAM (DDR3).
- Sistema operativo Debian 8.0.
- Disco duro de 320 GB a 5600 RPM.

Durante la ejecución del *Test Pilot*, los participantes pudieron comprobar el correcto funcionamiento de las funcionalidades existentes en la aplicación y detectar no conformidades previamente identificadas por el equipo de desarrollo o el cliente.

**Resultados de la Prueba Piloto.**

| | **Tiempo promedio de respuesta (ms)** | **Consumo promedio de memoria RAM (MB)** | **Consumo promedio de CPU (%)** |
|---|---|---|---|
| **Valor** | 536.938 | 256.3 | 62.21 |

**Tabla 13.** Resultados de la Prueba Piloto.

**Aplicación de la prueba estadística McNemar.**

McNemar es una prueba no paramétrica de comparación de proporciones para dos muestras relacionadas. Su función es comparar el cambio en la distribución de proporciones entre dos mediciones de una variable dicotómica y determinar que la diferencia no se deba al azar (que las diferencia sea estadísticamente significativa) (Juárez, Villatorio y López 2011).

Se utiliza para decidir si se puede o no aceptarse que determinado “tratamiento” induce un cambio en la respuesta de los elementos sometidos al mismo, y es aplicables a los diseños tipo “antes-después” en los que cada elemento actúa como su propio control (Guzmán 2009).

La prueba consiste en n observaciones de una v.a bidimensional $(X_i, Y_i)$ para i = 1,.., n, donde se define una hipótesis $H_0$ que niega los cambios y una regla de decisión que rechaza $H_0$ si se cumple el valor esperado.

En la dócima de McNemar, los resultados se representan en una tabla 2x2 en la siguiente forma:

| | | Clasificación $Y_i$ | |
|---|---|---|---|
| | | (+)<br>$Y_i = 0$ | (-)<br>$Y_i = 1$ |
| Clasificación $X_i$ | (+)<br>$X_i = 0$ | **(A)**<br>(0,0) | **(B)**<br>(0,1) |
| | (-)<br>$X_i = 1$ | **(C)**<br>(1,0) | **(D)**<br>(1,1) |

**Tabla 14.** Dócima de McNemar.

Los casos que muestran cambios entre la primera y la segunda respuesta aparecen en las celdillas B y C. Un individuo es clasificado en la celdilla B si se cambió de más (+) a menos (–) y es clasificado en la celdilla C si se cambió de menos (–) a más (+). Sino es observado ningún cambio va a la celdilla A (respuestas de más (+) antes y después) o a la celdilla D (respuestas de menos (–) antes y después).

Para aplicar McNemar se definió lo siguiente:

$H_0$: No hay diferencias en la proporción en sentirse satisfecho de la forma de gestionar las actividades por parte de las personas antes y después de usar el sistema.

Regla de decisión: Si $p \leq 0.05$ se rechaza $H_0$.

Para obtener los datos a evaluar, antes de la prueba piloto se le preguntó a cada uno de los participantes Si estaban insatisfechos con la forma de gestionar sus actividades. Al finalizar el ensayo se volvió a encuestar a los presentes, alcanzándose los siguientes valores:

| Instante | Insatisfecho | Satisfecho |
|---|---|---|
| Antes | 14 | 5 |
| Después | 4 | 15 |

**Tabla 15.** Resultados de la encuesta de satisfacción.

Apoyándose en el programa estadístico SPSS se procedió al análisis de la encuesta. Obteniéndose como resultando las gráficas que se muestran a continuación:

**Test Statistics[b]**

| | Satisfecho de la forma de gestionar sus actividades Antes & Satisfecho de la forma de gestionar sus actividades Después |
|---|---|
| N | 19 |
| Exact Sig. (2-tailed) | ,002[a] |

a. Binomial distribution used.

b. McNemar Test

**Figura 7.** Tabla de frecuencias que se mantienen y cambian.

**Satisfecho de la forma de gestionar sus actividades Antes & Satisfecho de la forma de gestionar sus actividades Después**

| Satisfecho de la forma de gestionar sus actividades Antes | Satisfecho de la forma de gestionar sus actividades Después | |
|---|---|---|
| | No | Si |
| No | 4 | 10 |
| Si | 0 | 5 |

**Figura 8.** Nivel de significancia.

Los resultados obtenidos se traducen en que la aplicación desarrollada ha causado impacto en las personas que han realizado los exámenes pues p no sobrepasa el valor esperado en la regla de decisión (0.002 < 0.05). Por lo tanto se rechaza $H_0$ y queda demostrado que el sistema contribuye a que las personas se sientan satisfechas en la forma de gestionar sus actividades.

**Resultados generales de la Prueba de Validez.**

Los resultados obtenidos en la Prueba piloto y la aplicación del método estadístico McNemar aportan una valiosa información a la investigación pues ha sido demostrado que el nuevo Sistema para Sincronizar Calendarios en el dominio uci.cu contribuye al rendimiento de los usuarios al contar con un

buen rendimiento y mejorar el sentimiento que tienen las personas sobre la forma en que gestionan sus actividades.

Al comunicarse con herramientas similares desplegadas en el dominio uci.cu y contar con servicios web que posibilitan el intercambio de datos fácilmente, se reducen los tiempos que las personas deben dedicar para chequear los eventos o tareas planificadas en las aplicaciones que son empleadas para gestionar este tipo de información.

De esta forma queda reafirmada la hipótesis planteada al inicio de esta investigación, alcanzando como resultado final un sistema que sincroniza las aplicaciones para la planificación operativa existentes en la UCI y la satisfacción de los usuarios.

## 3.4. Conclusiones.

En el presente capítulo se abordó el proceso de validación de la propuesta, uno de los más importantes para garantizar el éxito de la aplicación. Una vez concluidas la fase de pruebas se arriba a las siguientes conclusiones:

- Las pruebas de aceptación ejecutadas al final de cada iteración posibilitaron corregir no conformidades sobre la marcha.
- Las pruebas unitarias permitieron examinar individualmente parte de las funcionalidades de la aplicación, así como la validez de los algoritmos implementados para la gestión y sincronización de los eventos.
- El diagrama de despliegue facilita el conocimiento de los elementos necesarios para emplear e implantar la nueva solución informática.
- La prueba piloto realizada evidenció un alto rendimiento por parte de KLNDA, resultado que satisface a los usuarios y a los autores de la presente investigación.
- La aplicación de la prueba no paramétrica de McNemar demostró que la diferencia entre la satisfacción al gestionar actividades antes y un después del uso de KLNDA se considera estadísticamente significativa.

# Conclusiones

Al finalizar el presente trabajo de diploma se logró cumplir de manera satisfactoria los objetivos planteados al inicio del mismo, obteniéndose como resultado directo el Sistema para la Sincronización de Calendarios. Por lo que se considera concluida la investigación y se arriba a las siguientes conclusiones:

- El estudio de los principales sistemas para la gestión de actividades a nivel mundial y local permitió identificar tendencias, limitaciones y funcionalidades a tener en cuenta, entre las que destacan el uso de estándares para representar la información, el empleo del correo electrónico y de la mensajería instantánea para notificar a las personas y la generación de identificadores universalmente únicos para evitar colisiones.

- El uso de la metodología XP para guiar el proceso de desarrollo del *software* y los artefactos generados propició una mayor comunicación entre el equipo de desarrollo y el cliente, beneficiando el cumplimiento de los objetivos trazados.

- La propuesta de sincronización entre aplicaciones que gestionan actividades permite un mejor control del tiempo para las personas del dominio uci.cu.

- Se demostró la validez de un sistema capaz de sincronizar los diversos sistemas para la planificación operativa empleados en la Universidad las Ciencias Informáticas.

# Recomendaciones

Los objetivos del presente trabajo de diploma fueron alcanzados satisfactoriamente; sin embargo se recomienda:

- Desplegar el Sistema para la Sincronización de Calendarios.
- Adaptar las aplicaciones informáticas existentes en el entorno de manera que hagan provecho de las funcionalidades que brinda la herramienta informática desarrollada.
- Continuar el mantenimiento y desarrollo de nuevas características en aras de ampliar y enriquecer el Sistema para la Sincronización de Calendarios.

# Referencias bibliográficas


ALTER-NOS COMUNICACIONES 2014. Controlando nuestro tiempo I. [en línea]. [Consulta: 14 febrero 2015]. Disponible en: https://alternoscomunicaciones.wordpress.com/2014/04/06/controlando-nuestro-tiempo-i/.

ANDERS, V. 2014. CALENDARIO. *Diccionario Etimológico* [en línea]. [Consulta: 3 mayo 2015]. Disponible en: http://etimologias.dechile.net/?calendario.

ANTÚNEZ NARANJO, L.D. 2012. *Implementación del Sub-sistema de gestión de información personal de la Facultad 3*. La Habana: Universidad de las Ciencias Informáticas. 005.12-Ant-I

APPELO, J. 2008. The Definitive List of Software Development Methodologies. [en línea]. [Consulta: 21 noviembre 2014]. Disponible en: http://noop.nl/2008/07/the-definitive-list-of-software-development-methodologies.html.

CASTLEDINE, E. y SHARKIE, C. 2010. *jQuery: Novice to Ninja*. S.l.: Cambridge. ISBN 978-0-9805768-5-6.

CEBALLOS, D. 2003. *Control del TIEMPO: la organización social del TIEMPO* [en línea]. S.l.: s.n. Disponible en: http://www.ub.edu/iafi/Membres/DCeballos/Control%20del%20TIEMPO.pdf.

CUERVO, J. 2014. AYUDA EN LÍNEA PARA EL SISTEMA SENTAI DE LA CORPORACIÓN DE IMPORTADORES Y EXPORTADORES DE CUBA (CIMEX). *Informática Jurídica* [en línea]. [Consulta: 13 febrero 2015]. Disponible en: http://www.informatica-juridica.com/trabajos/ayuda-en-linea-para-el-sistema-sentai-de-la-corporacion-de-importadores-y-exportadores-de-cuba-cimex/.

ECLIPSE FOUNDATION 2015. Eclipse: The Eclipse Foundation. About the Eclipse Foundation. [en línea]. [Consulta: 17 enero 2015]. Disponible en: http://www.eclipse.org/org/.

FINCH, J., FREEMAN, R. y GILBERT, D. 1996. *Administración* [en línea]. S.l.: Prentice-Hall Hispanoamericana. ISBN 9789688806852. Disponible en: http://books.google.com.cu/books?id=g_nweMjueSkC.

GESPRO 2013. CEIGE - Centro de Informatización de Entidades. [en línea]. [Consulta: 17 abril 2015]. Disponible en: http://gespro.ceige.prod.uci.cu/gespro/about.

GOOGLE INC. 2014. Te damos la bienvenida a Google Calendar - Ayuda de Calendar. *Google* [en línea]. [Consulta: 11 junio 2014]. Disponible en: https://support.google.com/calendar/answer/2465776?hl=es&ref_topic=3417969.

GOSLING, J., JOY, B., STEELE, G., BRACHA, G. y BUCKLEY, A. 2015. *The Java® Language Specification* [en línea]. Java SE 8 Edition. Redwood City: Oracle Corporation. [Consulta: 4 mayo 2015]. Disponible en: http://docs.oracle.com/javase/specs/jls/se8/jls8.pdf.

GUZMÁN, R. 2009. McNemar. . S.l.

JACOBSON, I. 1998. *Applying UML in The Unified Process*. S.l.: s.n.

JEFFRIES, R., 1999. *Extreme Testing* [en línea]. 1999. S.l.: s.n. Disponible en: http://www.xprogramming.com/publications/SP99%20Extreme%20for%20Web.pdf.

JOYENT INC. 2014. node.js. *node.js* [en línea]. [Consulta: 11 noviembre 2014]. Disponible en: http://nodejs.org/.

JUÁREZ, F., VILLATORIO, J. y LÓPEZ, E. 2011. McNemar. [en línea]. S.l. [Consulta: 29 abril 2015]. Disponible en: http://www.rincondepaco.com.mx/rincon/Inicio/Apuntes/Proyecto/archivos/Documentos/McNemar.pdf.

KABIR, M. 2003. *La Biblia Del Servidor Apache 2* [en línea]. S.l.: Anaya Multimedia. Disponible en: http://es.scribd.com/doc/24039203/Mohammed-Kabir-La-Biblia-Del-Servidor-Apache- 2.

LARMAN, C. 2003. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado*. 2. Buenos Aires: Prentice Hall. ISBN 8420534382.

LETELIER, P. y PENADÉS, M.C. 2004. Metodologías Ágiles para el Desarrollo de Software: eXtreme Programming (XP). . S.l.: Universidad Politécnica de Valencia.

LINNOVATE 2014. MEAN.IO Documentation. *MEAN.IO* [en línea]. [Consulta: 11 noviembre 2014]. Disponible en: http://learn.mean.io/.

LÓPEZ, U. 2013. PATRONES DE DISEÑO: MODELO VISTA CONTROLADOR. *PATRONES DE DISEÑO* [en línea]. [Consulta: 13 abril 2015]. Disponible en: http://thelozu.blogspot.com/2013/07/modelo-vista-controlador.html.

MARTÍNEZ, I.C., 2010. *MODELO CONCEPTUAL/MODELO DE DOMINIO* [en línea]. 19 mayo 2010. S.l.: Universidad Simón Bolívar. Disponible en: http://ldc.usb.ve/~martinez/cursos/ci3715/clase6_AJ2010.pdf.

MONGODB INC. 2014. Introduction to MongoDB. *MongoDB.org* [en línea]. [Consulta: 11 noviembre 2014]. Disponible en: http://www.mongodb.org/about/introduction/.

MOZILLA y COLABORADORES INDIVIDUALES 2015. JavaScript | MDN. *Mozilla Developer Network* [en línea]. [Consulta: 17 enero 2015]. Disponible en: https://developer.mozilla.org/es/docs/Web/JavaScript.

NEDELCU, C. 2010. *Nginx HTTP Server*. S.l.: Aanchal Kumar.

NÚÑEZ PARADA, A. 2010. *Administración de empresas* [en línea]. S.l.: s.n. Disponible en: https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxhZG1jY3NjZnR8Z3g6MjEyMTE1YTczYTllYmM1Zg.

ORACLE CORPORATION 2015a. FOSS License Exception. *MySQL* [en línea]. Disponible en: http://www.mysql.com/about/legal/licensing/foss-exception/.

ORACLE CORPORATION 2015b. NetBeans IDE. *NetBeans* [en línea]. Disponible en: http://netbeans.org/features/index.html.

ORIENTE, J. 2012. Complejidad ciclomática ¿Nuestro código es fácil de mantener y probar? *Formandobits* [en línea]. [Consulta: 21 abril 2015]. Disponible en: http://formandobits.com/2012/11/complejidad-ciclomatica-nuestro-codigo-es-facil-de-mantener-y-probar/.

PRESSMAN, R. 2005. *Ingeniería del Software. Un enfoque práctico*. Sexta. New York: McGraw Hill. ISBN 9701054733.

REAL ACADEMIA ESPAÑOLA 2012. *Diccionario de la lengua española | Real Academia Española* [en línea]. Madrid: Real Academia Española. Disponible en: http://lema.rae.es/drae/?val=tiempo.

RIGGS, S. y KROSIGN, H. 2010. *PostgreSQL 9 Administration Cookbook*. Birmingham: Packt Publishing Ltd. ISBN 978-1-849510-28-8.

SANDEEP PANDA 2014. *AngularJS: Novice to Ninja*. S.l.: s.n.

SILVA BARRERA, D., 2011. *Manual de usuario del Sistema de Planificación y publicación de horarios docentes UCI* [en línea]. 1 octubre 2011. S.l.: s.n. Disponible en: http://horario.uci.cu/Manual%20de%20usuario.pdf.

SIPAC, 2014. *MANUAL DE USUARIO DE PLANIFICACIÓN*. noviembre 2014. S.l.: s.n.

SOMMERLAD, P., BUSCHMANN, F., ROHNERT, H. y STAHL, M. 1996. *Pattern-Oriented Software Architecture-A System of Patterns*. New York: John Wiley and Sons, Inc.

SOUSA, S. 2012. The Advantages and Disadvantages of Agile Software Development. [en línea]. [Consulta: 2 julio 2015]. Disponible en: http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-agile-software-development.html.

TELLIGENT SYSTEMS INC. 2013. Ayuda de ZWC. [en línea]. [Consulta: 11 noviembre 2014]. Disponible en: https://correo.estudiantes.uci.cu/help/es/advanced/zimbra_user_help.htm.

THE JQUERY FOUNDATION 2015. jQuery. *jQuery* [en línea]. Disponible en: http://jquery.com/.

THE PHP GROUP 2014. PHP: Hypertext Preprocessor. [en línea]. [Consulta: 12 abril 2014]. Disponible en: http://php.net/.

TJ HOLOWAYCHUK 2015. Mocha - the fun, simple, flexible JavaScript test framework. [en línea]. [Consulta: 12 mayo 2015]. Disponible en: http://mochajs.org/.

UCI 2015. Misión | Portal de la Universidad de las Ciencias Informáticas. *Portal UCI* [en línea]. [Consulta: 12 marzo 2015]. Disponible en: http://www.uci.cu/?q=mision.

VISUAL PARADIGM 2014. Visual Paradigm. UML, BPMN and Database Tool for Software Development. *Visual Paradigm* [en línea]. [Consulta: 12 abril 2014]. Disponible en: http://www.visual-paradigm.com.

WELLS, D. 2013. Extreme Programming: A Gentle Introduction. [en línea]. [Consulta: 21 noviembre 2014]. Disponible en: http://www.extremeprogramming.org/.

YAHOO! INC. 2015. Yahoo! Agenda. [en línea]. [Consulta: 2 noviembre 2014]. Disponible en: http://info.yahoo.com/privacy/es/yahoo/calendar.

ZAPATA, C., LUCÍA, G., PORTILLA, B., GÓMEZ, D., NARANJO, M. y CARMONA, P. 2009. Aproximación a una ontología para lenguajes de modelado gráfico. , vol. 29, pp. 10. ISSN 0121-4993.

ZAVALA RUIZ, J. 2008. *Análisis organizacional de empresas de software. Una primera aproximación a una visión alterna de la Ingeniería de Software*. S.l.: s.n.