Universidad de las Ciencias Informáticas

Trabajo de Diploma para optar por el título de:

Ingeniero en Ciencias Informáticas

Fusión de la información de los servicios de la comunidad universitaria de la UCI con el metaverso OpenSim

Autor: *Enelis Blanca Cuba Rondón*

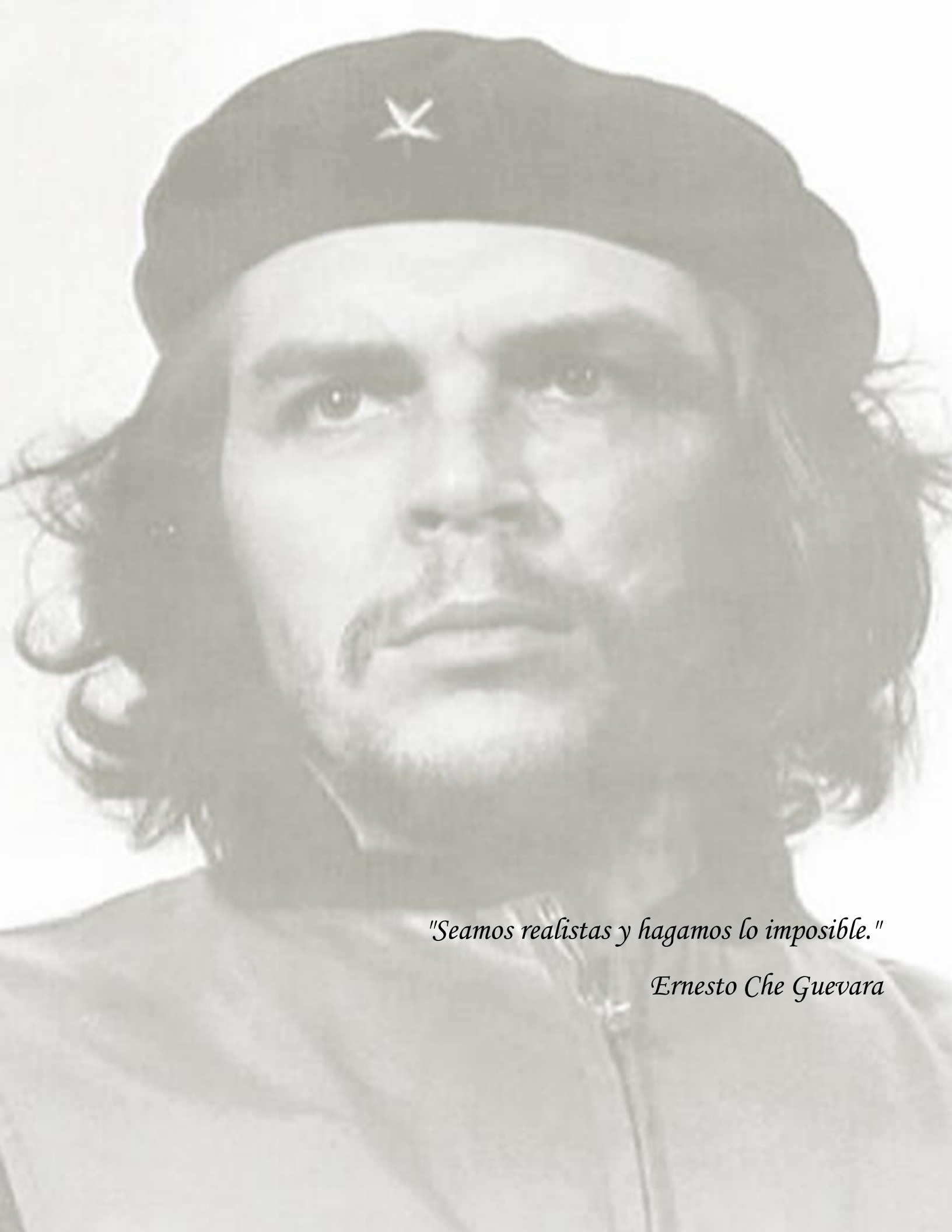
Tutores:

MSc. Lidiexy Alonso Hernández

Minardo Gollún González López

La Habana, junio del 2012

"Año 54 de la Revolución"



"Seamos realistas y hagamos lo imposible."

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo a los tutores Lidiexy Alonso Hernández, Minardo Gollún González López y al Centro de Desarrollo de Informática Industrial (CEDIN), de la Universidad de las Ciencias Informáticas, para que hagan el uso que estimen pertinente con este trabajo.

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

Enelis Blanca Cuba Rondón _____

Firma del Autor

MSc. Lidiexy Alonso Hernández _____

Firma del Tutor

Minardo Gollún González López _____

Firma del Tutor

Datos de Contacto

Nombre y Apellidos: MSc. Lidiexy Alonso Hernández

Edad: 32 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Títulos:

Licenciado en ciencias de la Informática

Máster en Informática Aplicada

Categoría Docente: Asistente

E-mail: lidiexy@uci.cu

Graduado de la Universidad de las Villas, con cuatro años de experiencia en el tema de realidad virtual y once años en el tema de tecnologías web.

Nombre y Apellidos: Ing. Minardo Gollún González López

Edad: 29 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

Categoría Docente: Asistente

E-mail: mgonzalezl@uci.cu

Graduado de la Universidad de las Ciencias Informáticas, con ocho años de experiencia en el tema de realidad virtual.

Agradecimientos

Mis agradecimientos primeramente para mi familia que de no ser por ellos no estaría aquí en estos momentos. Especialmente a mis dos viejitos, mi hermana del alma.

A mi amor Angel Omelio que siempre estuvo presente de todas las maneras posibles, apoyándome, dándome ánimos y amor.

A mi suegrita Marelis, Enmi, Omelio, Ildania, Santiago por su apoyo, consejo y ayuda en el transcurso de estos años.

A mis tutores y cliente por siempre poder contar con su apoyo, ayuda y guía.

A mis amigos Carlos Javier, Osnelvis y Yusleidy por todos estos años de excelente amistad.

A Toña y Roberto por su cariño mostrado hacia mi persona.

A mis amistades de estos 5 años en esta universidad y especialmente a Yaima Laugart, Lauren San Juan, Jacqueline Armesto, Aylén Goya, Angel Díaz, Ernesto Fuentes, Alián Ruiz, Rosbel por su ayuda, cariño, comprensión y apoyo recibido de su parte. También agradecer a todas mis compañeras que me han acompañado a los largo de la vida, entre ellas Rosalia, Eliannis, Lili, Irisleidis, La melli, Maite, Zurelis, Katia, Yamilé, Dunia y Evelín.

A todos los profesores que me impartieron clases por el conocimiento que me fue transmitido.

A la Revolución por permitirme estudiar en esta universidad y lograr una de mis metas.

Dedicatoria

A mi mamá y papá por todo el amor que no pude darles.

A mi abuelito del alma Pipo y a mi Quinito por darme todo el amor del mundo y hacerme feliz todo el tiempo.

A mis hermanas Katy y Yanelis, así como a mi sobrinita Annarita, que las considero mis tesoros.

A mi familia en general, que sin su apoyo no sería nada.

A mi novio Angel Omelio, por el amor que siempre me ha dedicado.

Resumen

Una de las innovaciones del año 2003 fueron los metaversos, mundos virtuales que permiten que un número de personas interactúen y puedan realizar actividades sociales, educacionales, científicas, entre otras. Un ejemplo de estos metaversos es OpenSim [OPENSIM 2012]. En el departamento de Visualización y Realidad Virtual, perteneciente al Centro de Informática Industrial de la Universidad de las Ciencias Informáticas está en proceso de estudio y establecimiento este metaverso para potenciar la educación, la investigación y el esparcimiento.

Con el desarrollo del presente trabajo se establece una aplicación web para la gestión de información del metaverso OpenSim. Para lograr el cumplimiento de la tesis fue necesario realizar un estudio de las diferentes tecnologías y herramientas necesarias para desarrollarla. Se realizó un estudio de las aplicaciones que existen actualmente que ofrecen funcionalidades parecidas a las que se necesita de la aplicación en cuestión. Luego de seleccionada la tecnología a trabajar se lleva a cabo un proceso de desarrollo del software.

Con la culminación de la tesis se obtuvo una aplicación capaz de gestionar y brindar servicios al metaverso OpenSim para lograr un mayor aprovechamiento de las facilidades que él ofrece al centro de Informática Industrial de la Universidad de las Ciencias Informáticas. La aplicación permite la creación de usuarios con los datos establecidos por los servidores de la Universidad, muestra aspectos del metaverso y gestiona eventos que se desarrollarán en dicho mundo.

Palabras clave: Metaversos, mundos virtuales, OpenSim.

Índice

Índice de figuras

Índice de tablas

Introducción

Una de las invenciones que tributa a la educación y a la investigación en el siglo XXI surge a partir del año 2003 y son llamados metaversos. Este término se creó a partir de la novela Snow Crash publicada en 1992 por Neal Stephenson. Los metaversos son entornos donde los humanos interactúan social y económicamente como íconos (avatares[1]) a través de un soporte lógico en un ciberespacio[2] que actúa como una metáfora del mundo real [CARLOS 2010].

En el departamento de Visualización y Realidad Virtual del Centro de Desarrollo de Informática Industrial (CEDIN) se trabaja en el estudio y establecimiento de un metaverso para la Universidad de las Ciencias Informáticas (UCI). Ese proyecto en desarrollo tiene como objetivo crear una comunidad virtual en la que se pueda llevar a cabo varias actividades, entre ellas: exposiciones de trabajos, seminarios, presentaciones de ferias, creación de grupos sociales, simulaciones de procesos y establecerlo como una vía alternativa para la formación profesional.

Actualmente para la creación de las cuentas con las que se accede al metaverso existe una aplicación web, pero no garantiza una confiabilidad en la información con que se registran las cuentas. Para lograr las metas por las que se estableciera OpenSim en la UCI, ya sean didácticas, recreativas o científicas, la información ingresada por los usuarios debe ser la que se controla en los diferentes servicios de la Universidad. En estos momentos, la información que está relacionada con el perfil de los usuarios, no se ajusta a los datos reales que son manejados en la UCI mediante servicios Protocolo Acceso a Objetos Simples (SOAP) y el Protocolo Ligero de Acceso a Directorios (LDAP[3]).

En la UCI se ha desarrollado la Feria Expositiva de Soluciones Informáticas (FESI), hoy convertida en una Feria Estudiantil. En la FESI 2011 se montó un mundo virtual que recreaba los estand[4] con información de los productos que se exponían en la feria real. De esta forma se facilitó la información a todas las personas que no pudieron acceder físicamente al evento. A pesar de que se creó un avatar que representara al encargado de dar información referente a los productos de un estand específico, no se tenía control de quién brindaba esta información, pues cualquier avatar podía dar una explicación incorrecta. La creación de

[1] Personas virtuales que representa a un individuo real en el metaverso.
[2] Ámbito artificial creado por medios informáticos.
[3] *Protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.*
4 Instalación o puesto en una exposición o mercado.

estos avatares se hacía a pedido y sin ningún registro de identificación de la persona que representaba en el mundo real.

Actualmente no se puede determinar si existe una persona en la UCI con el nombre establecido por un avatar. Otro ejemplo puede ser el caso de la transmisión y adquisición de cursos en el metaverso, en donde no se pueda determinar si la persona que lo imparte está calificada para hacerlo y no hay control de cuáles estudiantes son los que lo reciben.

Para mayor usabilidad del metaverso es necesario incorporar algunos módulos que brinden servicios para garantizar el control y organización de la comunidad de usuarios. De esta manera es necesario garantizar la creación y organización de los grupos en OpenSim. También existe la necesidad de brindar espacios para la gestión de los eventos que se desarrollen en el metaverso.

Debido a la situación problemática planteada se propone el siguiente **problema de la investigación**: ¿Cómo establecer una conexión segura y con datos confiables entre el metaverso OpenSim y los diferentes servicios que brinda la Universidad?

Presentado el problema se define como **objeto de estudio** la gestión de información a través de servicios web. Se establece como **campo de acción** las aplicaciones web que consumen servicios para garantizar la seguridad de la información.

Para dar solución al problema se define como **objetivo** de la presente investigación: desarrollar una aplicación web que permita la fusión de la información de los servicios de la comunidad universitaria de la UCI con el metaverso OpenSim y como **objetivos específicos**:

- Integrar el servicio LDAP con el sistema de acceso al metaverso OpenSim.
- Implementar servicios que permitan el acceso a la información que se genera en el metaverso, dígase gestionar eventos, mostrar grupo, mostrar amigos, crear avatar, mostrar perfil y mostrar mapa del mundo virtual.
- Configurar módulos que permitan el acceso a servicios de control y organización de información del metaverso.
- Comprobar el estado final de la aplicación.

Para poder lograr los objetivos planteados se definen las siguientes tareas:

- Comparación crítica de las plataformas virtuales existentes.

- Descripción de las tecnologías a utilizar en la construcción de la aplicación web.
- Preparación de la capa de datos para lograr un nivel elevado de integración.
- Implementación de la aplicación donde se muestre la integración lograda.
- Configuración de módulos que aporten servicios web al servidor OpenSim.
- Implementación de servicios que permitan el acceso a la información generada en el metaverso.
- Realización de pruebas de las funcionalidades implementadas.

Para el cumplimiento de estos objetivos se utilizaron varios métodos para la búsqueda y procesamiento de la información como son:

Métodos científicos:

Teóricos:

- **Análisis-síntesis:** para la búsqueda y análisis de los conceptos y documentos necesarios que permitieron la obtención de los elementos para la comprensión del objeto de estudio.
- **Análisis histórico–lógico:** permitió que se analizara el desarrollo histórico del objeto de estudio y encontrar algunas publicaciones sobre el desarrollo de los metaversos y de las aplicaciones web.

Empírico:

- **Observación:** Para observar y verificar los aportes de un metaverso a la Universidad. Para determinar cuántos problemas de la comunidad puede resolver una sola aplicación.

Con el objetivo de organizar y darle una estructura al trabajo se ha decidido dividir el trabajo en tres capítulos. A continuación se muestra como se encuentran estructurados:

Estructuración de los capítulos:

- **Capítulo 1 Fundamentación teórica:**

En el capítulo se describen los fundamentos teóricos de los procesos que se desarrollaron. Se trata el tema de las tendencias tecnológicas más avanzadas y homólogas al objeto de estudio. Se establece una breve descripción de los diferentes elementos posibles a utilizar en el desarrollo de la tesis, así como un análisis de las aplicaciones que existen en el mundo y su correspondencia con lo que se deseó lograr en la aplicación.

- **Capítulo 2 Exploración, planificación y diseño de la solución propuesta:**

En el capítulo se establece la solución al problema a través de la definición de las historias de usuarios y de la planificación del tiempo de trabajo. Muestra la descripción de las características fundamentales que posee el sistema, así como la definición de las iteraciones por las que pasó el proyecto para su desarrollo. Además se muestra una planificación de cómo se trabajaron las iteraciones y también el diseño de la aplicación. Se muestra como quedaron las relaciones entre las clases que se implementaron. Se da una breve explicación de cómo está estructurada la base de datos del sistema.

- **Capítulo 3 Implementación y pruebas del sistema:**

En el capítulo se puede visualizar como se llevó a cabo la implementación el sistema, definiéndose diferentes tareas por iteraciones. El capítulo culmina con las pruebas que se realizaron en conjunto con el cliente para definir si el sistema cumplía con los objetivos definidos y con la discusión de la solución.

Capítulo 1: Fundamentación teórica

Introducción

El desarrollo de tecnologías se ve aparejado a nuevos avances tecnológicos. Al aparecer los metaversos surgen variantes que hacen posible la programación y control de estos mundos virtuales. En el presente capítulo se describen los fundamentos teóricos de los procesos a desarrollar. Se tratará el tema de las tendencias tecnológicas más avanzadas y homólogas al objeto de estudio. Se establecerá una breve descripción de los diferentes elementos posibles a utilizar en el desarrollo de la tesis, así como un análisis de las aplicaciones que existen en el mundo y su correspondencia con lo que se quiere lograr en la aplicación.

1.1 Metaversos

El término metaverso aparece descripto por primera vez en la novela de ciencia ficción *Snow Crash* (1992) de Neal Stephenson. A partir del año 2003 surgen los metaverso que se caracterizan por ser mundos virtuales en donde los humanos, como avatares, interactúan entre sí y con software agentes[5], en un espacio tridimensional que usa la metáfora del mundo real. Entre algunas de las características de los metaversos se pueden destacar [CARLOS 2010]:

- **Interactividad:** existe en una o varias máquinas a las que se puede acceder remota y simultáneamente por multitud de personas. Es un mundo compartido en el que las acciones de un usuario pueden ser percibidas e influir en el resto de usuarios.
- **Corporeidad:** las personas acceden al programa a través de un interfaz que simula un entorno. Simulado en primera persona, sometido a las Leyes de la Física y en el que los recursos son escasos.
- **Persistencia:** el programa existe independientemente de que los usuarios estén conectados y recuerda la localización de personas y objetos. Se encuentra en una plataforma mantenida por el publicador[6] del Metaverso.

Algunos de los metaversos más conocidos son Second Life y OpenSim. A continuación se da una breve descripción de mundos virtuales.

[5] *Individuo virtual con inteligencia artificial que realiza actividades predeterminadas por el programa.*
[6] *Administrador del servidor de OpenSim.*

1.1.1 Second Life

Es un mundo virtual basado en Internet, lanzado en el 2003, desarrollado por Linden Research, Inc. (referido como Linden Lab). Tiene un programa de cliente descargable Second Life viewer, que permite a los residentes, interactuar entre sí a través de avatares [LINDEN_LAB 2012]. El objetivo de Linden Lab ha sido crear un mundo definido por el usuario en el cual la gente puede interactuar, jugar, hacer negocios y comunicarse.

1.1.2 OpenSim

Es un proyecto creado a partir de la liberación del código fuente del cliente de Second Life. El código original fue revisado, modificado y compilado bajo la licencia de GNU. OpenSim nace con la propuesta de crear un servidor de aplicaciones 3D. Este servidor presenta un entorno interactivo e inmersivo[7], convirtiéndose en un ambiente representativo de la realidad. Una de las ventajas que posee es que se encuentra bajo la licencia BSD [OPENSIM 2012]. En este metaverso se pueden llevar a cabo varias actividades como por ejemplo:

- Simulaciones.
- Realizar exhibiciones (esculturas 3D, fotos, pinturas, etc.).
- Dinámica y juego de roles.
- Clases online, conferencias, talleres, capacitaciones.
- Proyectos colaborativos.

EL uso de estas tecnologías en cualquier sector aporta facilidades. Son aplicaciones que integran varias funcionalidades que le pueden dar solución a varios problemas en un solo espacio. En la UCI existen varios servicios que necesitan de un monitoreo y administración por diferentes personas, se pueden poner de ejemplo: la gestión del consumo energético en la Universidad, la gestión académica, exposiciones, espacios de esparcimiento y de apoyo al aprendizaje de otros idiomas. Los metaversos permiten que todos esos servicios sean integrados y que una persona con solo acceder con su avatar al mundo pueda realizar cualquiera de esas actividades. Importante destacar que se necesita solo de una persona para administrar el servidor del metaverso.

[7] Referente a los metaversos, significa el nivel de abstracción que poseen las personas dentro del mundo virtual, en donde las actividades que ocurren dentro de OpenSim, realmente influyen en la vida cotidiana de las personas.

1.2 Análisis de las soluciones existentes

El proyecto metaverso se desarrolla desde el año 2003 a nivel mundial. Primero con la aparición de *Second Life* y luego con OpenSim. Estas grandes comunidades poseen aplicaciones que de una forma u otra les posibilita la administración de las cuentas para poder interactuar en ellos. Se tiene como ejemplo:

- ***Second Life web_interface*:** interfaz visual de la aplicación web de Second Life que gestiona la creación y diseño de los avatares de ese mundo virtual. También permite comprar tierras, ropas y conocer acerca de lo que se realice en el metaverso [LINDEN_LAB 2012]. En la Figura 4 de los anexos se muestra un ejemplo de la interfaz principal de la aplicación.
- **OpenSim *Web_interface* (Redux)**: es un software de código abierto que le permitirá realizar tareas de administración de la información del metaverso. Es gratis para uso personal y comercial. Permite acciones como crear y listar los avatares, mostrar mapa y mostrar eventos [OPENSIMULATOR 2012b]. La programación que fue usada para elaborarlo es sencilla, por lo tanto, para entender las funciones que utiliza no se requiere de un alto nivel de conocimiento en programación.
- **módulo Xoopensim**: Módulo que se encuentra integrado al sistema gestor de contenidos XoopsCube. Fue creado por *Fumi.Iseki*, *Saki Sakura* y BK201 Se creó sobre la base de OpenSim Redux 0.32.Brinda servicios de [TUIS 2011].
    - crear grupos,
    - realizar búsquedas,
    - listar y crear los avatares que existen en OpenSim,
    - lista y crear eventos.

En nuestro país aún no se ha establecido ningún mundo virtual. En estos momentos la selección y asimilación de un metaverso está en proceso de estudio en la Universidad de la Habana y en la UCI. Actualmente no se ha desarrollado un servicio que permita la creación de cuentas del mundo virtual vinculado a un dominio de datos. El análisis de las soluciones existentes definió como resultado que algunos de los algoritmos y funciones que se usaron en el desarrollo de estas herramientas se pueden reutilizar para la implementación de la aplicación de que es objeto esta tesis.

1.3 Metodologías de software

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos. Imponen un proceso disciplinado sobre el desarrollo de

software con el fin de hacerlo más predecible y eficiente [MURCIA 2011]. Desarrollan un proceso detallado con un fuerte énfasis en planificar las actividades por las que debe pasar un proceso de desarrollo de software. Algunos ejemplos de metodologías usadas en la UCI son:

1.3.1 Proceso unificado de desarrollo (RUP):

RUP va más allá del mero análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. El resultado es un proceso basado en componentes, dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental [JACOBSON 2000]. Se utiliza mayormente para proyectos largos y en los que son necesarios establecer una detallada documentación.

1.3.2 Programación extrema (XP):

Es usada para proyectos cortos y con necesidad de alterar los objetivos de desarrollo de un proyecto en cualquier momento que lo necesite. Esta metodología establece que el cliente es un integrante más del equipo de trabajo. Dirigida a equipos pequeños. Se utiliza en proyectos de poca duración, organiza el trabajo mediante fases, es flexible a los cambios que puedan ocurrir en las funcionalidades del sistema. Está organizado por historias de usuarios, que son tarjetas escritas de forma sencilla que muestran las funcionalidades del software. La filosofía de desarrollo incluye la realización de pruebas unitarias que permite la validación de la implementación. XP permite entregas pequeñas del software a medida que se elabora [PRESSMAN 2005].

1.3.3 Valoración de selección

Para el desarrollo de la aplicación es necesario escoger una metodología que permita realizar el trabajo en poco tiempo y que sea adaptable a cualquier cambio de requisito que pueda ocurrir en el proceso. No obstante en los inicios de la investigación se pueden utilizar algunos elementos de RUP, como levantamiento de requisitos y el diagrama de clases del diseño, que expliquen de manera formal elementos de apoyo para la comunicación con el cliente.

La introducción del concepto de metaversos es nuevo para la Universidad y existen muy pocos especialistas involucrados con OpenSim; de ahí que los requisitos de la aplicación pueden variar en dependencia de lo que realmente el cliente desee incorporar o cambiar, para aumentar la funcionalidad de la solución que se propone. Como estrategia para lograr un resultado coherente se involucra al cliente como parte del equipo de trabajo, de esta manera se adquiere un experto en la tecnología que involucra al

metaverso. Además se desea establecer un orden en las fases del proceso de desarrollo. Estas son las razones fundamentales por las que se ha seleccionado la metodología XP, la cual garantiza que se generen solo los artefactos necesarios para describir los aspectos fundamentales de la aplicación como las historias de usuarios y la planificación para el desarrollo de la solución [LETELIER,PENADÉS 2007].

1.4 Herramientas de Ingeniería de Software Asistida por Computadora (CASE)

CASE es una filosofía que se orienta a la mejor comprensión de los modelos de empresa, sus actividades y el desarrollo de los sistemas de información. Esta filosofía involucra además el uso de programas que permiten [SOMMERVILLE 2005a]:

- Construir los modelos que describen la empresa.
- Describir el medio en el que se realizan las actividades.
- Llevar a cabo la planificación.

En el desarrollo del proyecto se hace necesario crear diagramas que apoyen a la descripción de cómo se establecen los procesos. Es necesario utilizar herramientas que permitan la gestión de los artefactos de un software. Desde el curso 2008-2009 la Universidad ha orientado el uso de la herramienta Visual Paradigm para el proceso docente y productivo con la justificación de ser un producto basado en software libre aunque se pagó la licencia que se establece para su uso [CALIDAD 2008a]. A continuación se describen algunas características.

1.4.1 Visual Paradigm

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. Algunas características de esta herramienta son [PARADIMG 2012]:

- Posibilita la creación de diagramas de flujo de datos.
- Posee distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.

1.5 Sistemas gestores de contenidos (CMS)

Es una herramienta que permite a un editor crear, clasificar y publicar cualquier tipo de información en una página web. Generalmente trabajan sobre una base de datos de modo que el editor simplemente lo que hace es actualizarla, en donde incluye una nueva información o se edita la existente. Algunos ejemplos de CMS son [FORMANTÍN, SIERRA 2007]:

1.5.1 Drupal

Es uno de los CMS más populares, gratuito y de código abierto. Fue creado en PHP y con posibilidad de utilizar varias bases de datos distintas, por defecto MySQL. Este CMS facilita la organización, administración y publicación de contenidos, con una gran variedad de personalización. Para su uso requiere de servidor web Apache y como servidor de base de datos puede ser MySQL, PostgresSQL, SQLite [*Drupal* 2012].

1.5.2 Joomla

Es un CMS de código libre, creado en PHP. Muchos aspectos incluyendo el fácil uso y la extensibilidad hacen de Joomla el CMS más popular en la actualidad. Es usado para crear aplicaciones web de todas las formas y tamaños. Con él se puede lograr crear portales o sitios web sociales, páginas de negocios online, periódicos digitales, entre otros [JOOMLA 2012].

1.5.3 XOOPS Cube

Es una plataforma de aplicaciones web de código abierto que se encuentra bajo la licencia GPL. Está escrito en PHP y utiliza como Base de datos a MySQL. Algunas características de XOOPS Cube son [*XOOPS Cube* 2012] :

- Modularizado: Los módulos pueden ser instalados/desinstalados/activados/desactivados con un simple *clic* .Posee sistema de administración de módulos.
- Posee un módulo llamado Xoopensim que es usado para gestionar información del metaverso OpenSim.
- Personalización extensible.

1.5.4 Valoración de selección

Todos los CMS que fueron estudiados aportan funcionalidades para la realización de la aplicación. Permiten crear y editar aplicaciones sin mucha dificultad. Todos son de código abierto y cada uno brinda aspectos importantes a la aplicación. Aunque se profundizó en los gestores Drupal y Joomla, que son los más usados en la Universidad [CALIDAD 2008b], fue seleccionado el CMS XoopsCube. Este último no cuenta con la popularidad de los anteriores, pero posee un módulo llamado Xoopensim que permite la gestión de datos pertenecientes al metaverso OpenSim, que lo convierte en una propuesta más llamativa para el desarrollo.

1.6 Marcos de trabajo

Es una estructura de soporte definido que permite organizar y desarrollar un proyecto. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado para ayudar a desarrollar y unir los diferentes componentes de un proyecto [GUTIERREZ]. Los objetivos principales que persigue un marco de trabajo son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. A continuación se da alusión a alguno de ellos:

1.6.1 Zend Frameworks

Requiere PHP 5 e incorpora el patrón Modelo Vista Controlador (MVC). Alguna de las ventajas que posee son [VASWANI 2010]:

- El marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL.
- Robustas clases para autenticación y filtrado de entrada.
- Es un software con licencia BSD.

1.6.2 Yet it is! (Yii)

Es basado en componentes de alto rendimiento para desarrollar aplicaciones web de gran escala. Basado en PHP. Permite la máxima reutilización en la programación web y puede acelerar el proceso de desarrollo. La persona que trabaje con este marco de trabajo posee el control total sobre la configuración (desde la presentación hasta la persistencia). Viene empaquetado con herramientas para ayudar a probar y depurar la aplicación y tiene una documentación clara y completa. Posee patrón de diseño MVC, integración de JQuery, *widgets* de Ajax como por ejemplo el autocompletado de campos de texto. También posee

generación automática de código para el esqueleto de la aplicación, aplicaciones CRUD, entre otras [*YIIframework* 2012].

1.6.3 Symfony

Incluye varias técnicas para acelerar el desarrollo de tus aplicaciones: existen multitud de comandos de consola para generar clases y código PHP, toda la configuración de la aplicación se puede realizar con archivos YAML[8] en vez de XML, soporta anotaciones para definir toda la configuración de la aplicación en las propias clases PHP en vez de en archivos de configuración externos [*Symfony* 2012].

1.6.4 YUI

Es una herramienta para crear aplicaciones web enriquecidas del lado del cliente. Ofrece una serie de librerías que permiten crear scripts en JavaScript compatibles con los navegadores más habituales. YUI 3 contiene además una serie de herramientas, controles y componentes ya listos para implementar muchos de los dinamismos típicos de las páginas web [WELLMAN 2008].

1.6.5 JQuery

Marco de trabajo para el lenguaje JavaScript que implementa una serie de clases (de programación orientada a objetos) que permiten programar sin preocuparse del navegador con el que se visita el usuario. Ofrece una infraestructura para la creación de aplicaciones complejas del lado del cliente. Con jQuery existe una ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de Ajax[9]. Es un producto bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización [ALVAREZ 2010].

1.6.6 Valoración de selección

Para el desarrollo de la aplicación se necesita un marco de trabajo sencillo, para ahorrar tiempo en pos de familiarizarse con él. También se desea que establezca un orden de los componentes del sistema para hacer más simple y organizado el desarrollo de la aplicación. Se seleccionó el marco de trabajo Yii pues ofrece todas estas facilidades y además integra a JQuery. Yii cuenta con una amplia documentación y ofrece componentes visuales que optimizan el trabajo. Posee un administrador de componentes que

[8] Formato de serialización de datos legible por humanos inspirado en lenguajes como XML, C, Python, Perl.
[9] Técnica de desarrollo web para crear aplicaciones interactivas.

permite la creación instantánea de modelos, módulos y brinda la posibilidad de generar el patrón CRUD[10] completo [*YIIframework* 2012].

1.7 Herramientas de desarrollo web

Algunos ejemplos de herramientas (o IDE de desarrollo como son llamadas) que son usadas para el desarrollo web son:

- **Zend Studio:** Sistema que posee editor de textos para páginas PHP, ayudas que facilitan la creación y gestión de las aplicaciones hasta la fase de depuración del código [26]. Zend Studio implementa además unas interesantes opciones para trabajar en grupo, al integrar el sistema de trabajo conocido como sistema de control de versiones (CVS) [*Zend* 2012].
- **Aptana:** Entorno de desarrollo integrado (IDE) de desarrollo para aplicaciones de la web 2.0, gratuito, código libre, con soporte Ajax, PHP, Ruby, Adobe Air, iPhone. Aptana está basado en el conocido entorno de desarrollo Eclipse, también de código fuente abierto. Es una distribución focalizada en el desarrollo web, con soporte a HTML, CSS y JavaScript [APTANA 2012].
- **Eclipse:** proyecto de desarrollo de software de código fuente abierto, cuyo objetivo es la construcción de herramientas integradas para el desarrollo de aplicaciones. Es neutral y adaptable a cualquier tipo de lenguaje, por ejemplo C/C++, Cobol, C#, XML, entre otros. La característica clave es la extensibilidad [MEDNIEKS 2011].
- **NetBeans 7.1.2:** Entorno de código abierto de desarrollo. Es un sistema modular, escrito en el lenguaje de programación Java. Puede ser utilizado como un marco genérico para construir cualquier tipo de aplicación. Es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Permite crear aplicaciones web con PHP 5, posee un potente depurador integrado y además viene con soporte para diferentes marcos de trabajos. Entre algunas de las características de este software están: la administración de las interfaces de usuario, de las configuraciones y del almacenamiento [BOCH 2011].

1.7.1 Valoración de selección

La entrega de la aplicación debe ser en tiempos cortos, es necesario de un IDE de desarrollo que permita al programador completar código, cargar librerías, utilizar marcos de trabajo, entre otros. Luego de analizados los IDE, se descarta Zend Studio por ser un software privativo y porque existen software libres que poseen

[10] Patrón de casos de uso gestionar.

sus mismas funcionalidades. En ese caso se tiene a Aptana, Eclipse y NetBeans. De estos tres IDE se seleccionó el NetBeans porque aparte de que posee todas las funcionalidades mencionadas al inicio de la valoración ofrece un entorno amigable y un potente depurador que permite al usuario poder verificar paso a paso como es llevada a cabo la ejecución del proyecto. Además es uno de los IDE de desarrollo más usado en la Universidad [TECNOLÓGICOS 2009].

1.8 Sistemas Gestores de Base de Datos (SGBD)

Son un conjunto de programas que permiten crear y mantener una base de datos. Permiten asegurar la integridad, confidencialidad y seguridad. Para el desarrollo de la aplicación es necesario utilizar un gestor que este dirigido a la base de datos que utiliza OpenSim, por lo tanto se selecciona el gestor base de datos Mysql. Aunque actualmente existan otros gestores como el PostgresSQL que permiten funcionalidades similares, se utilizará Mysql para evitar problemas de conflictos entre las bases de datos. Señalar que ambas se mantienen como gestores libres y cuentan con una marcada aceptación el desarrollo de software en la UCI [CALIDAD 2008b]. A continuación se da una breve especificación del software:

Está bajo la licencia GPL. Su diseño multihilo[11] le permite soportar una gran carga de forma muy eficiente. Fue creada por la empresa sueca MySQL AB, que mantiene el derecho de autor del código fuente del servidor SQL, así como también de la marca. Entre las características principales de este gestor están [ORACLE 2012]:

- Aprovecha la potencia de sistemas multiprocesador.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de Interfaz de Programación de Aplicaciones (API) en gran cantidad de lenguajes (C, C++, Java, PHP).
- Gran portabilidad entre sistemas.
- Gestión de usuarios y contraseñas, mantiene un muy buen nivel de seguridad en los datos.

1.9 Lenguajes de modelado y de programación web

Actualmente existen diferentes lenguajes de programación para desarrollar en la web, estos han surgido debido a las tendencias y necesidades de las plataformas. Con el paso del tiempo las tecnologías se han desarrollado y han surgido nuevos problemas que necesitan ser solucionados. Esto ha dado lugar al

[11] *Multitarea o multiproceso, puede realizar varios procesos a la misma vez.*

desarrollo de lenguajes de programación dinámicos para la web, que permiten interactuar con los usuarios y utilizar sistemas de bases de datos. Los lenguajes web se clasifican en dos tipos: los del lado del cliente y lado del servidor. Los del lado del servidor son aquellos reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible. Ejemplo de estos se puede nombrar a PHP y Java. Mientras que los del lado del cliente son aquellos que pueden ser directamente asimilados por el navegador y no necesitan un pretratamiento. Entre estos se encuentran JavaScript y CSS [VALDÉS 2010].

Otros usados en el desarrollo de un software son los lenguajes de modelado. Estos permiten realizar modelos necesarios en el proceso de desarrollo de un software. Entre ellos se puede nombrar a Lenguaje Unificado de Modelado (UML) y la Notación para modelado del proceso de negocio (BPMN) [LARMAN 1999].

1.10 Módulos de servicios para OpenSim

Al ser OpenSim una versión libre desarrollada a partir de la liberación del código fuente del cliente de Second Life muchos de los servicios que se puedan brindar en el metaverso no se encuentran implementados. A medida que la comunidad de desarrolladores OpenSimulator se expande se han incorporado módulos que le ofrecen a OpenSim más funcionalidades. Actualmente necesita de algunos servicios que son obtenidos a partir de aplicaciones web que han sido implementadas por integrantes del grupo de soporte del metaverso. Ejemplo de estos servicios son los módulos de búsqueda *Ossearch* y el de control y gestión de grupos de usuarios *XmlRpcGroup* [TUIS 2011].

1.10.1 Módulo *Osseach*:

Fue creado por el grupo de desarrollo de OpenSimulator con el objetivo de permitir en el metaverso realizar búsquedas en general. Es un servicio que se integra a la plataforma web que se encargue de la gestión dela información de OpenSim. Posibilita hacer búsquedas de usuarios, grupos, lugares, eventos, entre otros. Permite crear una arquitectura escalable, abierta para la búsqueda de mundos basados en OpenSim. Cuenta con cuatro ficheros *.php (*query*, *register*, *parser* y *databaseinfo*), una *.dll (*OpenSimSearch*) y un fichero *.sql (*search*) [OPENSIMULATOR 2012b]. En la siguiente imagen se muestra un ejemplo de su uso en el metaverso.

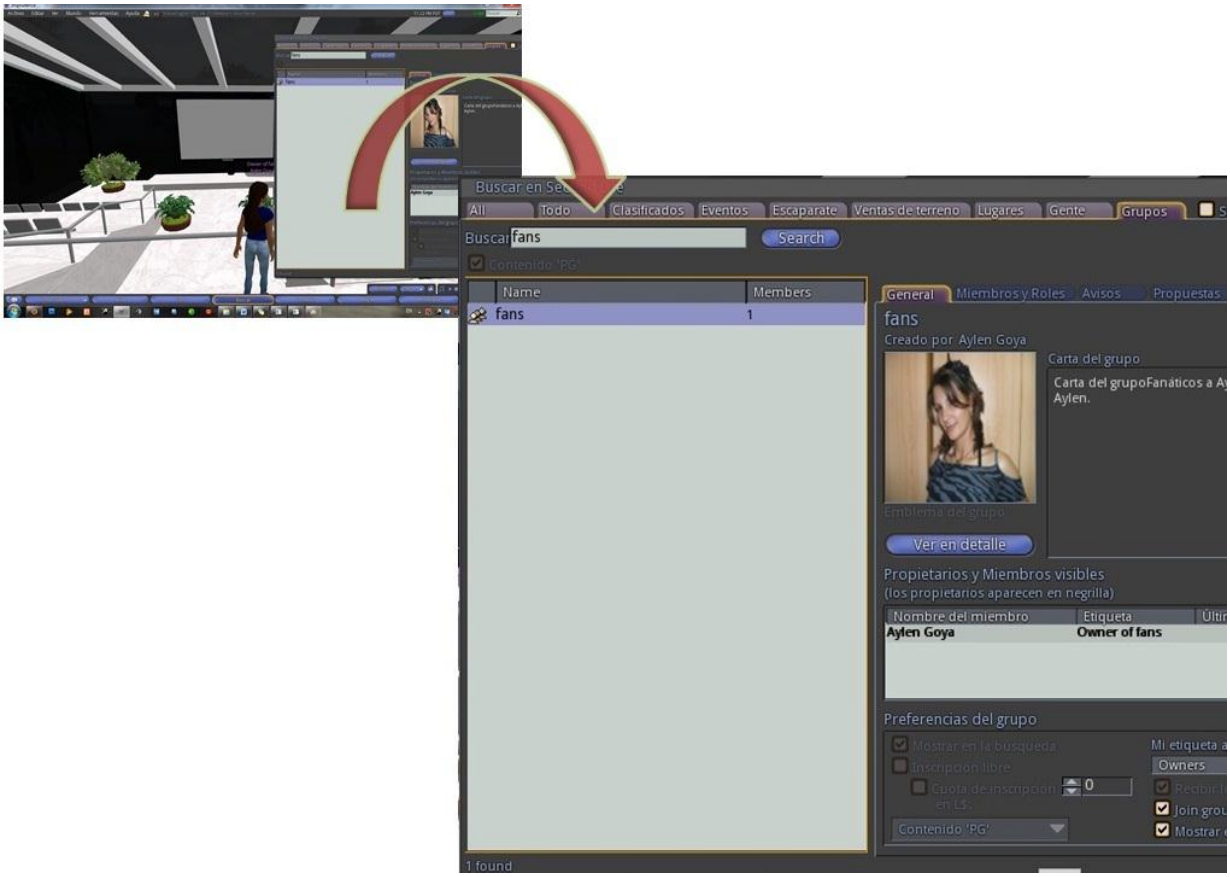


Figura 1.Módulo Ossearch en OpenSim.

1.10.2 Módulo XmlRpcGroup:

Permite crear una arquitectura escalable, abierta para los grupos que se generen en el OpenSim. Está formado por tres componentes: el servicio de grupo, el servicio conector de los grupos y el módulo grupo. En la Figura 1 se muestra la relación que existen entren estos tres componentes. El servicio de grupo guarda los datos de los grupos del mundo virtual. Esta información es guardada en una base de datos. El servicio conector de los grupos es quien permite la comunicación entre el servicio de grupo y OpenSim mediante el uso del protocolo establecido por el servicio. El módulo grupo es quien realiza las peticiones relacionadas con los grupos.

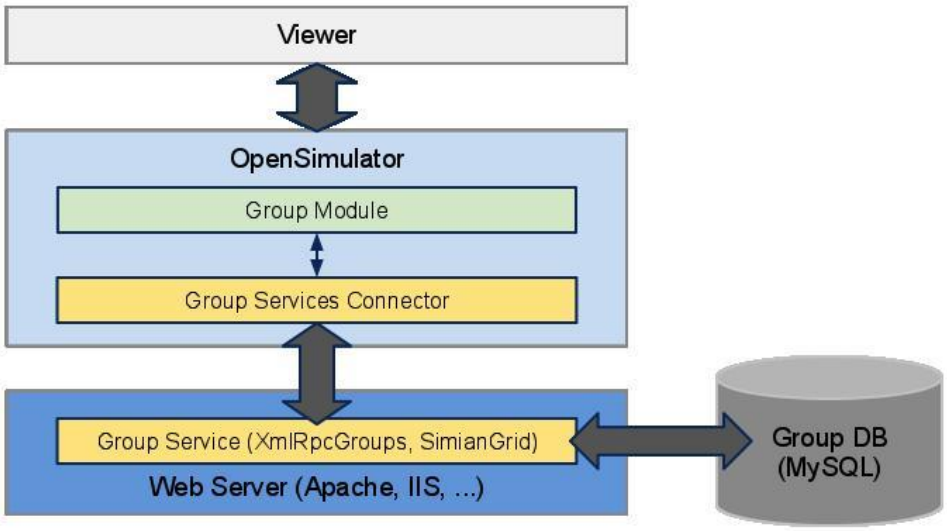


Figura 2. Relación entre los componentes del módulo XmlRpcGroup.[12]

Luego que se establece la comunicación entre los componentes de este módulo el resultado final es mostrado en el metaverso a la hora en la que el usuario pueda crear un grupo [OPENSIMULATOR 2012a; 2012b]. En la siguiente imagen se muestra cómo funciona el módulo dentro de OpenSim.

[12] Tomado de http://opensimulator.org/wiki/Enabling_Groups.

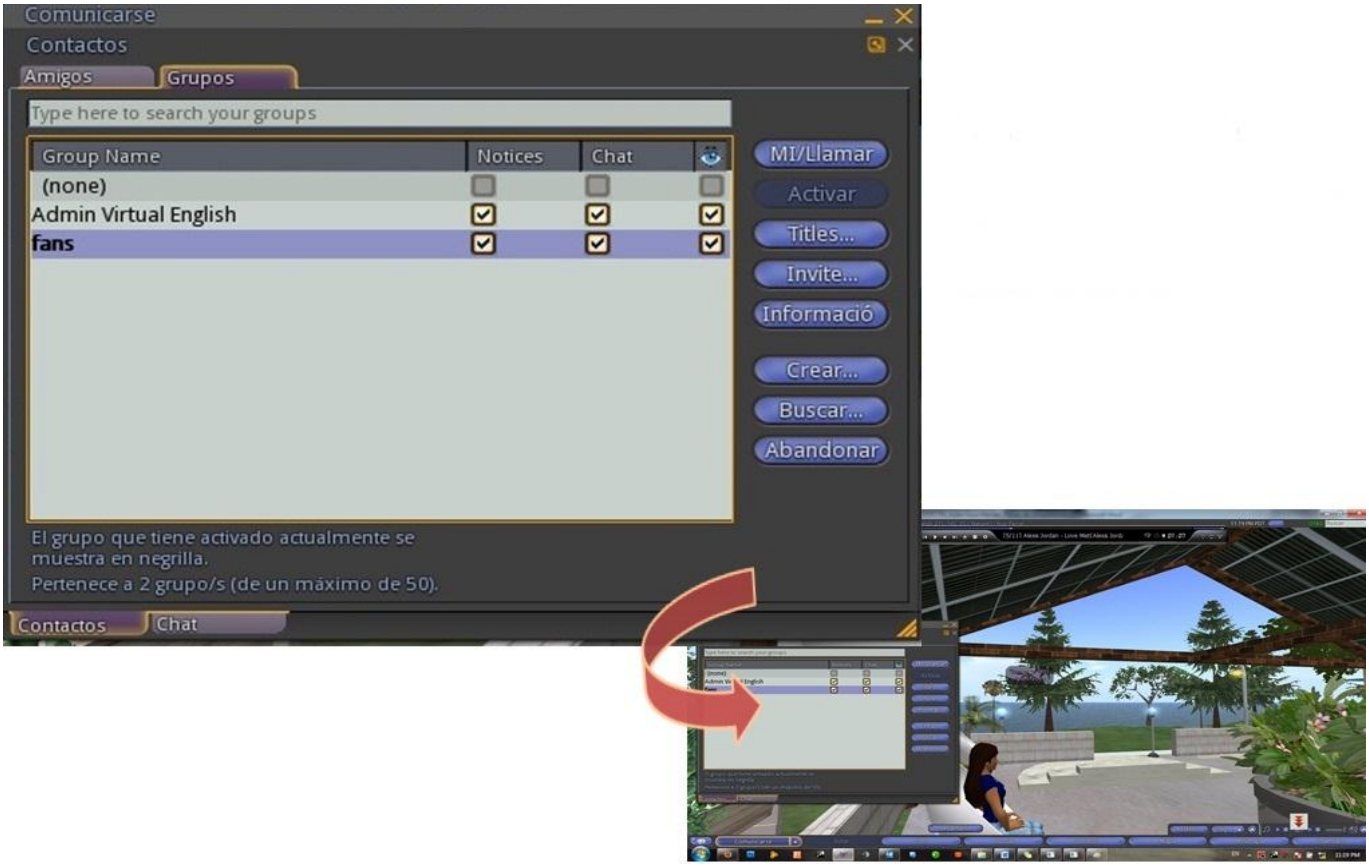


Figura 3. Módulo XmlRpcGroup en OpenSim.

Conclusiones parciales

Con el estudio del estado del arte se logró conformar un marco teórico que permite la comprensión de decisiones tomadas a lo largo del trabajo para darle cumplimiento a los objetivos planteados. El análisis de las diferentes metodologías, herramientas y lenguajes sugiere que para el desarrollo del presente trabajo utilizar la metodología XP facilite el desarrollo ordenado de la aplicación. Entre las herramientas estudiadas y teniéndose en cuenta las peculiaridades de cada una, Visual Paradigm, XoopsCube, Yii, Netbeans y Mysql poseen las características necesarias para enfrentar el desarrollo de una solución al problema planteado. Se logró identificar que el uso de módulos como el *Ossearch* y *XmlRpcGroup* podría agregar funcionalidades al metaverso OpenSim.

Capítulo 2: Exploración, planificación y diseño de la solución propuesta

Introducción

La exploración de un proceso, la planificación de las actividades y un diseño correcto son parte fundamental para lograr el éxito de una aplicación. En este capítulo se establecerá la solución al problema a través de la definición de las historias de usuarios y de la planificación del tiempo de trabajo. Presenta como objetivo principal describir las características fundamentales que poseerá el sistema, así como establecer las iteraciones por las que tiene que pasar el proyecto para su desarrollo. Además se mostrará una planificación de cómo se trabajaran las iteraciones y se describirá el diseño de la aplicación.

2.1 Definición del cliente y usuario del sistema

Cliente: Persona interesada en el desarrollo del producto en un tiempo establecido y que el resultado del mismo cumpla con un número de requisitos determinados por él con anterioridad. Realiza reuniones constantes con el equipo de desarrollo en donde enfrasca su interés en fraguar las dudas que puedan tener. Se encuentra presente a lo largo del desarrollo de la aplicación. Para esta investigación se define como cliente al Ing. Lorenzo Domínguez quien cumple el rol de especialista responsable del montaje de la plataforma OpenSim en la UCI.

Usuario: Persona que pueda interactuar con la aplicación. Se define que debe estar registrado dentro del dominio de la Universidad. Puede cumplir cualquier rol, dígase profesor, trabajador o estudiante siempre y cuando cumpla con las especificaciones de estar registrado en los sistemas que establezca la UCI.

2.2 Proceso de captura de requisitos del sistema

Antes de establecer una solución al problema siempre se hace necesario llevar a cabo una reunión con el cliente del software o aplicación, para llegar a acuerdos y establecer los requisitos que se deseen que posea el proyecto. De ahí es común establecer en los primeros encuentros que se tienen con el cliente, peticiones informales de requisitos de software y diseños que a la larga aportarán mucho a la concepción del software.

XP propone que se realice un levantamiento de requisito en las primeras reuniones, en las que se tomen todos los aspectos relacionados con las necesidades del sistema a desarrollar. En estos encuentros se pueden llevar a cabo prácticas muy efectivas de captura de requisitos como:

- Entrevistas frecuentes para aclarar dudas que de manera continua puedan aparecer.
- Observación de los procesos a informatizar para lograr mantener una presencia lo más imperceptible posible para no obstaculizarlos.
- Juegos de rol donde el cliente simule ser un usuario que interactúa con el sistema, este segundo simulado por el desarrollador.

Luego de realizados los encuentros se realiza la captura de requisitos que se muestra a continuación:

2.2.1 Requisitos funcionales del sistema

Luego de una detallada investigación sobre el objeto de estudio, se analiza qué debe hacer el sistema para darle cumplimiento a los objetivos planteados. Para ello se enumeran a través de requisitos funcionales, que son capacidades o condiciones que el sistema debe cumplir [SOMMERVILLE 2005b]. Los requisitos funcionales del sistema propuesto son:

RF1- Autenticar el usuario en el sistema.

RF2- Administrar avatar para metaverso OpenSim.

RF2.1- Crear avatar para OpenSim.

RF2.2- Eliminar avatar de OpenSim.

RF3- Configurar de módulos que brindan servicios web a OpenSim.

RF3.1- Configurar del módulo ossearch.

RF3.3- Configurar del módulo XmlRpcGroup.

RF4- Administrar información referente al metaverso.

RF4.1- Mostrar amigos de usuarios.

RF4.2- Mostrar grupos de usuarios.

RF4.3- Mostrar integrantes de un grupo.

RF4.3- Gestionar eventos.

RF4.4- Mostrar perfiles.

RF4.5- Mostrar mapa del metaverso.

2.2.2 Requisitos no funcionales del sistema

Los requisitos no funcionales del sistema son propiedades que el producto debe poseer según lo que establezca el cliente [SOMMERVILLE 2005a]. Son características que hacen a la aplicación más atractiva, confiable y usable.

Diseño e implementación

RnF1: Lenguajes de programación: Se utilizará el lenguaje de programación php 5.

RnF2: Herramienta de desarrollo: Para la implantación del sistema se utilizará la herramienta de desarrollo NetBeans. Se utilizará el Visual Paradigm como herramienta CASE para el modelado del sistema.

RnF 3: Diseño del sistema: Aplicar arquitectura en capa con un patrón de diseño MVC.

RnF 4: Implementación del sistema: Utilizar programación orientada a objetos. Usar una metodología de desarrollo ágil que permita un rápido y eficiente desarrollo del sistema.

Apariencia o interfaz externa

RnF 5: Interfaz con un diseño con vista a acelerar la velocidad de respuesta; sin dejar de ser amigable, legible, interactiva, fácil de usar, profesional, clara y sencilla.

Usabilidad

RnF 6: El sistema debe proporcionar una interfaz sencilla para que todos los usuarios puedan utilizarla sin necesidad de poseer habilidades técnicas.

Software

RnF 7: Multiplataforma. Debe ejecutarse sin problemas tanto en Internet Explorer como en Mozilla Firefox.

2.3 Descripción del sistema propuesto

En conjunto con el cliente se decidió llamar a la aplicación OpenSim con el nombre de "UCIgrid" para que representara los principios por los que fue establecido en el centro. Este nombre significa red de la UCI o llevándolo a un mejor consenso podría significar espacio UCI. Los principales objetivos de la aplicación serán el de crear avatares para poder acceder al metaverso OpenSim, brindar funcionalidades que

permitan conocer datos del metaverso y establecer módulos que aportan las funcionalidades las funciones de crear y buscar grupos al mundo virtual.

Luego de establecidos las funcionalidades necesarias a tener la aplicación, se definió en conjunto con el cliente un primer diseño de como seria la interfaz visual de la aplicación, que se muestra a continuación:

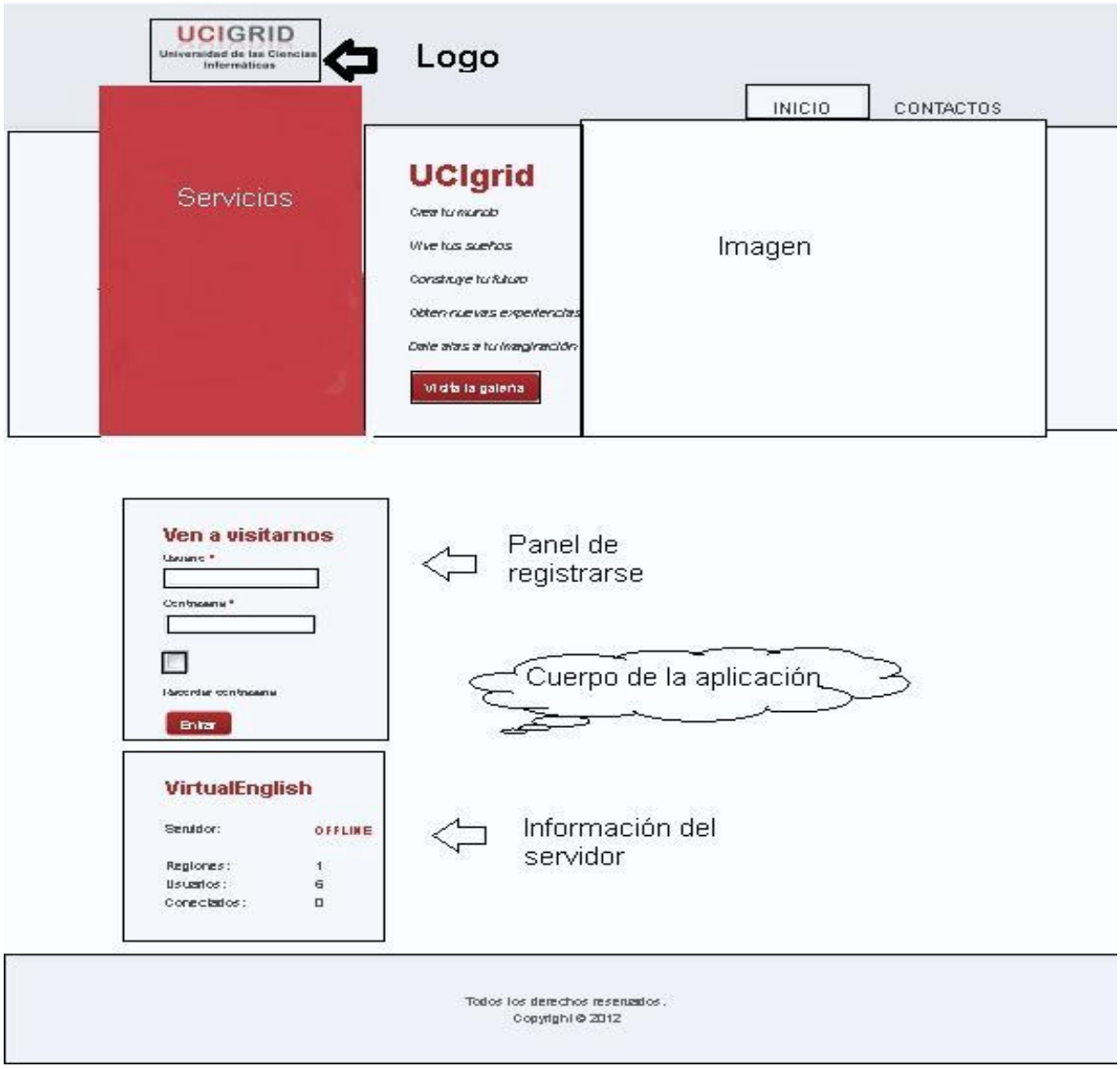


Figura 4. Primer diseño de la interfaz

La aplicación contará con la siguiente estructura funcional:

- **Interfaz para autenticarse:** en la aplicación dado que su uso es solo dentro de la Universidad se usará para la autenticación con el servicio LDAP de la UCI para un mejor control en la conexión de los usuarios.
- **Nombre de la región:** Opción general que permite mostrar a los usuarios datos de metaverso como: el estado del servidor, la cantidad de usuarios en el mundo y cuantos están en línea.
- **Contactar:** tendrá un servicio que permitirá al usuario escribir alguna queja o sugerencia sobre la aplicación, así como si existiera algún problema a la hora de trabajar con la misma tendrá la oportunidad de notificarlo.
- **Crear_avatar:** servicio que le brindara al usuario la posibilidad de crearse una cuenta en el metaverso OpenSim. Da acceso a que el usuario pueda acceder a los demás funcionalidades de la aplicación.
- **Gestionar_avatar:** le permitirá al usuario eliminar su avatar una vez creado, así como listar los avatares que existen en el mundo virtual.
- **Eventos:** le facilitará al usuario poder realizar funcionalidades referentes a los eventos que se puedan realizar en el metaverso. En esta opción se podrá realizar una completa gestión de los eventos del usuario que se encuentre registrado en la aplicación en ese momento. Mediante esta funcionalidad se podrá gestionar un evento ya sea eliminar, modificar y observar parámetros de un evento en específico. Además se mostrara un listado con los eventos que existen en OpenSim en ese momento.
- **Grupos:** en esta funcionalidad el usuario podrá observar cuantos grupos existen en el metaverso y algunos de sus datos. También se podrá apreciar si se desea el listado de integrantes por grupo.
- **Amigos:** permitirá al usuario verificar sus amigos y ver el estado en el que se encuentran en el mundo virtual en ese momento.
- **Mapa:** esta funcionalidad permitirá al usuario observar cual es el mapa de la región sobre la que se encuentra establecido.
- **Perfil:** se define en esta funcionalidad la opción de mostrarle al usuario el perfil de uno de sus amigos.
- **Galería:** Permite al usuario observar un grupo de fotos referentes a las funcionalidades que se pueden llevar a cabo en el metaverso y fotos del mundo virtual en el que se encuentre.

- El usuario solo podrá poseer acceso a las de más funcionalidades si hubo creado su avatar con anterioridad, pues las funcionalidades trabajan en función de la información que genere ese avatar en el metaverso.

2.4 Fase de exploración

La exploración es la etapa del proceso de desarrollo de software que propone XP para comenzar la construcción de un producto. Cuando sean entregadas las propuestas del cliente al equipo de trabajo, comienza el análisis, la tormenta de ideas y la conceptualización del software. Un aspecto de gran importancia es que el cliente debe estar inmerso en cada uno de las actividades antes expuestas [PRESSMAN 2005]. En esta etapa se realizará un estudio de que funcionalidades se deberían desarrollar para lograr primeramente el objetivo principal de la aplicación, así como las que se complementarían para lograr el éxito del proyecto. Además se expone un diseño de manera general de la relación que van a tener las clases que conformaran la solución del problema. A continuación se expondrán las diferentes etapas por las que transita esta fase:

2.4.1 Historias de usuarios

Las historias de usuarios (HU) son como los casos de uso en RUP, con la diferencia de que deben ser escritos por el cliente con un lenguaje sencillo. En esencia, no son más que las ideas del cliente organizadas y agrupadas de acuerdo a su funcionalidad, estableciéndose un orden que permita priorizar sus necesidades, así como definir las que resultan críticas o claves en el momento de desarrollo de la solución [PRESSMAN 2005]. A continuación aparecen descritas las HU de la presente solución:

Historia del cliente	
No.: 1- Acceso a la aplicación	
Prioridad: alta	**Nivel de complejidad:** alta
Estimación: 15 días	**Iteración asignada:** 1
Descripción de la tarjeta: Se debe acceder a la aplicación mediante el servicio LDAP con la contraseña y usuarios UCI. Si el usuario no pone los datos correctos no se le dejará acceder a la aplicación.	

Observaciones:

Tabla 1. HU Acceso a la aplicación.

Historia del cliente	
No.: 2- Gestión de avatar para OpenSim	
Prioridad: alta	**Nivel de complejidad:** alta
Estimación: 15 días	**Iteración asignada:** 2
Descripción de la tarjeta: La aplicación tendrá una opción que permitirá crearse un avatar para poder acceder al metaverso OpenSim. Los datos que se podrán poner será la contraseña, ya que su nombre y apellido, serán por defecto las que aparecen en los servidores de la Universidad. Además tendrá las opciones de eliminar avatar y modificar la contraseña de la cuenta que fue registrada al crear el avatar.	
Observaciones:	

Tabla 2. HU Gestión de avatar para OpenSim.

Historia del cliente	
No.: 3- Configuración de módulos que brindan servicios web a OpenSim	
Prioridad: alta	**Nivel de complejidad:** alta
Estimación: 10 días	**Iteración asignada:** 3
Descripción de la tarjeta: Se establecerá la configuración de los módulos ossearch y XmlRpcGroup. Se debe redefinir la estructura y configuración de los módulos de manera que el metaverso pueda obtenerlos a partir de la aplicación.	
Observaciones:	

Tabla 3. HU Configuración de módulos.

Historia del cliente	
No.: 4- Servicio de muestra de amigos de usuarios	
Prioridad: media	**Nivel de complejidad:** media

Estimación: 5 días	**Iteración asignada:** 4
Descripción de la tarjeta: Se establecerá un modelo que permita mostrar los amigos del usuario registrado en ese momento.	
Observaciones:	

Tabla 4. HU Servicio de muestra de amigos.

Historia del cliente	
No.: 5- Servicio de muestra de información de los grupos de usuarios	
Prioridad: media	**Nivel de complejidad:** media
Estimación: 5 días	**Iteración asignada:** 4
Descripción de la tarjeta: Se establecerá un modelo que permita mostrar los diferentes grupos de usuarios que existen en el metaverso.	
Observaciones:	

Tabla 5. HU Servicio de muestra de grupos.

Historia del cliente	
No.: 6- Servicio de administración de eventos	
Prioridad: media	**Nivel de complejidad:** media
Estimación: 5 días	**Iteración asignada:** 5
Descripción de la tarjeta: Se establecerá un modelo que permita mostrar los diferentes eventos que se desarrollan en el metaverso.	
Observaciones:	

Tabla 6. HU Servicio de muestra de eventos.

Historia del cliente
No.: 7- Servicio de muestra del perfil de amigos

Prioridad: media	**Nivel de complejidad:** media
Estimación: 5 días	**Iteración asignada:** 5
Descripción de la tarjeta: Se establecerá un modelo que permita mostrar los daros del perfil que posee el usuario en el mundo virtual.	
Observaciones:	

Tabla 7. HU Servicio de muestra del perfil.

Historia del cliente	
No.: 8- Mostrar mapa	
Prioridad: media	**Nivel de complejidad:** media
Estimación: 10 días	**Iteración asignada:** 5
Descripción de la tarjeta: Se establecerá un módulo que muestre el mapa del mundo de OpenSim.	
Observaciones:	

Tabla 8. HU Mostrar mapa.

Historia del cliente	
No.: 9- Desarrollo de la interfaz de la aplicación	
Prioridad: media	**Nivel de complejidad:** media
Estimación: 5 días	**Iteración asignada:** 5
Descripción de la tarjeta: Establecer una interfaz amena y adecuada para los usuarios que accedan a ella. Ya sea en la composición de los elementos, como en el diseño de la aplicación.	
Observaciones:	

Tabla 9. HU Desarrollar interfaz de la aplicación.

2.5 Fase de planificación

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar todas las historias de usuario, se parte del tiempo asignado a cada una en la fase de exploración [PRESSMAN 2005]. Una vez terminado esto, se procede a organizarlas en las iteraciones correspondientes, teniéndose en cuenta la prioridad especificada por el cliente y del tiempo de desarrollo de cada una. A continuación se explica el proceso que se llevó a cabo en el desarrollo de la aplicación.

2.5.1 Plan de iteraciones

Una iteración es una parte esencial del proyecto que cuando culmina el cliente obtiene un resultado parcial. Se cumple que al concluir la última iteración, el mismo quedará completamente satisfecho, pues culmina la producción del proyecto. En la organización de las iteraciones se debe evitar extender más de un mes laboral; por lo general se proponen entre 20 y 21 días. La demora de un producto puede atentar con los objetivos que tenga el cliente con la aplicación. Una manera de evitar que esto ocurra se lleva a cabo mediante las pruebas de los usuarios finales, las correcciones a las que son sometidas la aplicación en las revisiones y así se podrá definir si el software va por un buen camino.

A continuación se presentan las iteraciones que se determinaron necesarias para la producción del software:

Iteraciones	Historias de Usuario a implementar	Tiempo de trabajo
Iteración 1	Acceso a la aplicación	15 días
Iteración 2	Gestión de avatar para OpenSim	15 días
Iteración 3	Configuración de módulos que brindan servicios web a OpenSim.	10 días
Iteración 4	Servicio de muestra de amigos de usuarios. Servicio de muestra de grupos de usuarios. Servicio de muestra del perfil de los usuarios.	15 días
Iteración 5	Servicio de muestra de eventos. Mostrar mapa.	20 días

	Desarrollar la interfaz de la aplicación

Tabla 10. Distribución de iteraciones por Historias de usuarios.

2.6 Plan de entregas

El plan de entrega es el compromiso final del equipo de desarrollo con el cliente, pues en él se define cuando será entregado el producto. Representa un factor importante para el proyecto, pues la demora en la entrega del proyecto trae consigo insatisfacción con el cliente. En el plan de entrega se lleva a cabo la estimación del tiempo necesario para entregar al cliente las versiones del producto a medida que se cumplan los requisitos del software.

La siguiente tabla muestra la entrega de los objetivos que se deben haber cumplido al final de cada iteración, que se realizan en la cuarta semana de los meses de enero, febrero, abril y mayo respectivamente.

Producto	1ra Iteración	2da Iteración	3ra Iteración	4ta Iteración	5ta Iteración
Aplicación OpenSim	-------	--------	Versión 0.1 del producto	Versión 0.2 del producto	Versión 1.0 del producto

Tabla 11. Plan de entrega del producto.

2.7 Fase de diseño

En la fase de diseño la metodología XP sugiere que hay que diseñar el software sencillo y fácil de implementar, ya que esto permite realizar el trabajo en menos tiempo y con menos esfuerzo. El diseño de la aplicación establece conformar un diseño sencillo de la arquitectura y de las tarjetas de clase, responsabilidad y colaboración (CRC). A continuación se describen las etapas en las que se trabajó en esta fase.

2.7.1 Descripción de la arquitectura

La arquitectura de un sistema de software es la organización o estructura de los componentes importantes que interactúan en el mismo. La aplicación está diseñada bajo un patrón MVC. Este patrón separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes diferentes

[MAQUINA; PARRA 2008]. En la siguiente figura se muestra como se aplica lo anteriormente explicado al sistema UCIgrid. El patrón MVC se aprecia en la imagen, donde la vista es la página HTML y el código que provee de datos dinámicos a la página; el modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista. El sistema tiene un aspecto que lo identifica y es el uso de dos bases de datos, la del servidor OpenSim y la de la aplicación.

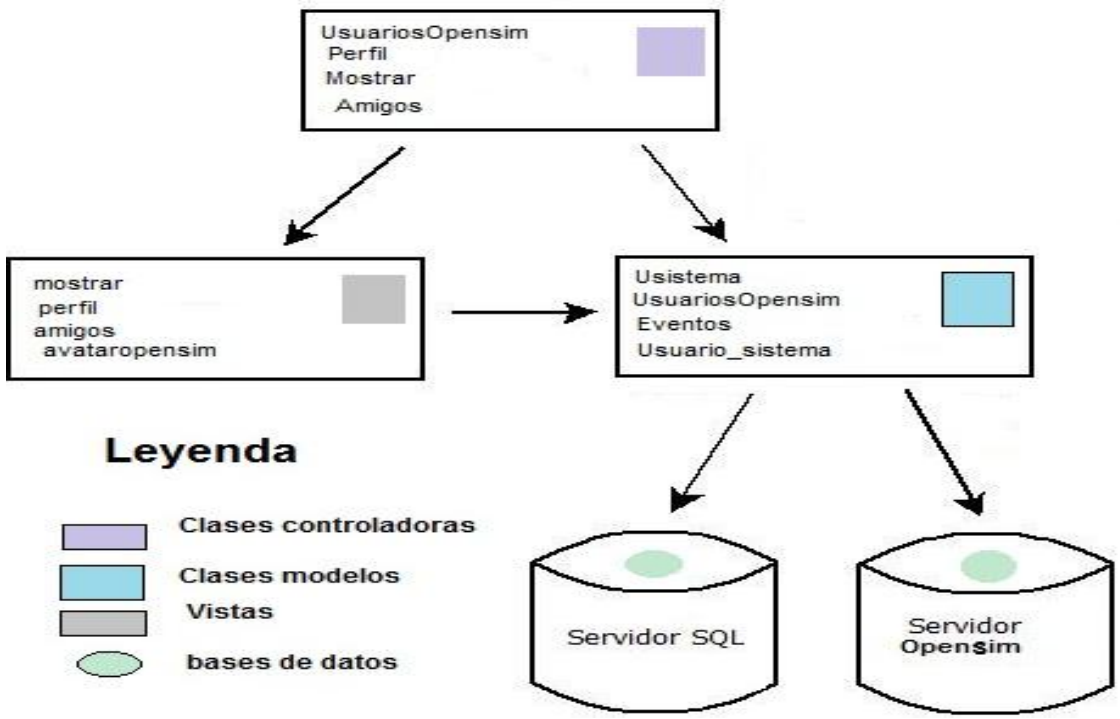


Figura 5. Descripción de la arquitectura del sistema.

Definida la arquitectura se expone un diagrama de clases para mostrar una vista general del funcionamiento de la aplicación.

2.7.2 Diagrama de clases del sistema

Para una mayor comprensión de cómo estará estructurado el sistema se define el siguiente diagrama de clases:

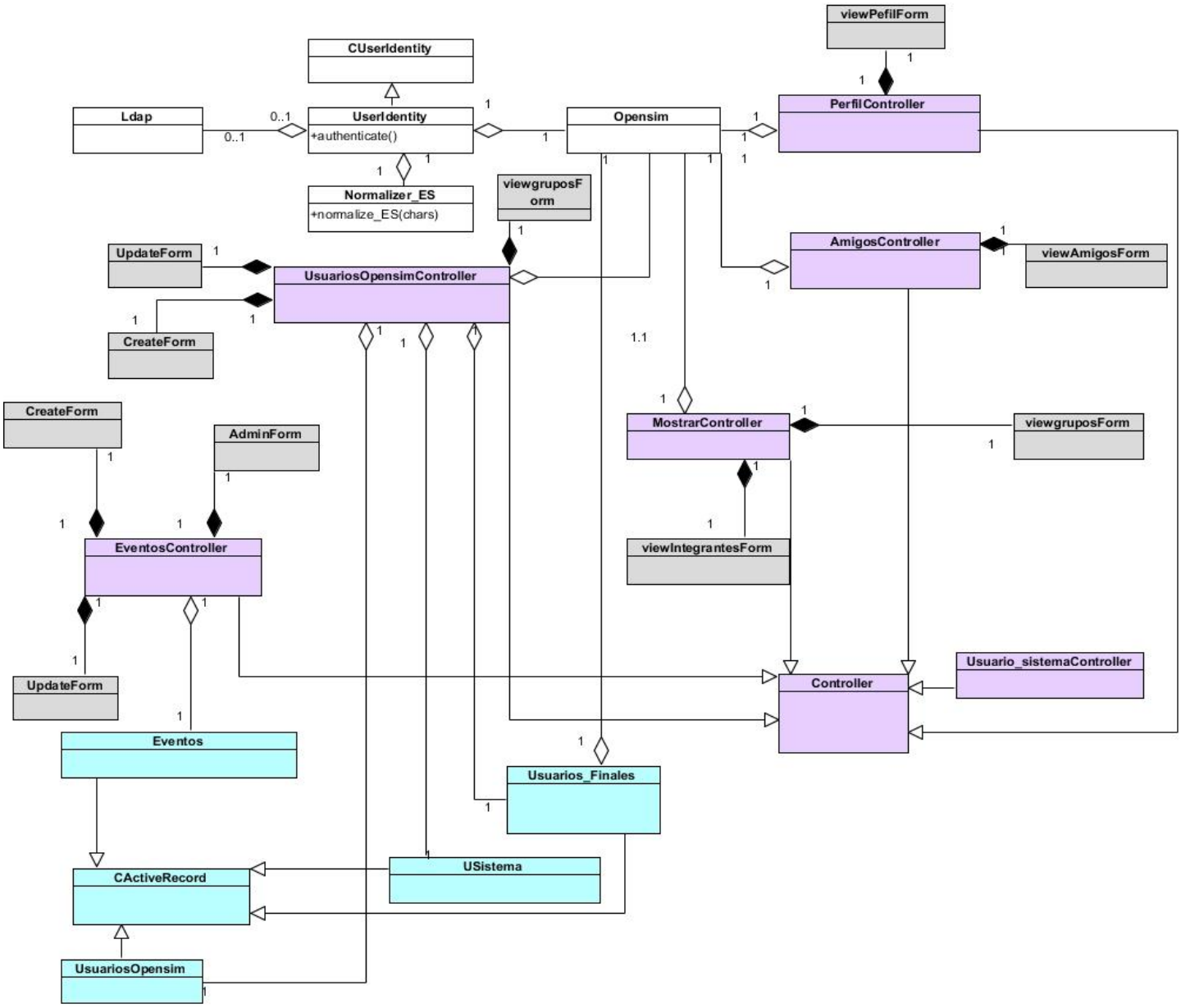


Figura 6. Diagrama de clases del sistema.

Para obtener una mayor descripción de las clases a continuación se muestran las tarjetas CRC:

2.7.3 Descripción de las tarjetas CRC

El principal objetivo de las tarjetas CRC es dejar el pensamiento procedimental para enfocarse al orientado a objetos. Cada tarjeta representa una clase con su nombre en la parte superior, en la sección

inferior izquierda están descritas las responsabilidades y a la derecha las clases que le sirven de soporte. A continuación se presentan las tarjetas que fueron necesarias crear para el desarrollo de la aplicación.

OpenSim	
Funcionalidades:	**Clases relacionadas:**
Permite crear y eliminar avatar directamente en la base de datos. También crear los directorios necesarios para el inventario en OpenSim y chequear datos de usuarios del metaverso. Además posibilita obtener datos de amigos y grupos del mundo virtual.	➢ UserIdentity ➢ Usuarios_Finales ➢ UsuariosOpenSimController ➢ PerfilController ➢ MostrarController ➢ AmigosController

Tabla 12. Tarjeta CRC OpenSim.

UserIdentity	
Funcionalidades:	**Clases relacionadas:**
Permite el realizar la autenticación del usuario y obtener datos necesarios de los servicios de la UCI.	➢ OpenSim ➢ Ldap ➢ Normalizer_ES ➢ Usistema

Tabla 13. Tarjeta CRC UserIdentity.

Normalizer_ES	
Funcionalidades:	**Clases relacionadas:**
Permite modificar en un texto las letras con tilde a sin tilde.	➢ UserIdentity ➢ PerfilController ➢ MostrarController ➢ AmigosController

Tabla 14. Tarjeta CRC Normalizer_ES.

Ldap	
Funcionalidades:	**Clases relacionadas:**
Permite la comunicación con el servicio LDAP y obtener de él la información que se desee.	➢ UserIdentity

Tabla 15. Tarjeta CRC Ldap.

UsuariosOpenSim	
Funcionalidades:	**Clases relacionadas:**
Es la representación de la tabla usuarios_OpenSim Permite realizar operaciones sobre los datos que esta contiene.	➢ UsuariosOpenSimController

Tabla 16. Tarjeta CRC UsuariosOpenSim.

Usuarios_Finales	
Funcionalidades:	**Clases relacionadas:**
Permite referenciar la función que crear usuarios y hacer un trabajo con los datos que son necesarios para crearlos y enviarlos al método de la clase OpenSim.	➢ OpenSim ➢ UsuariosOpenSimController

Tabla 17. Tarjeta CRC Usuarios_Finales.

Usistema	
Funcionalidades:	**Clases relacionadas:**
Representación de la tabla usuario y sobre ella permite realizar operaciones sobre los datos que esta contiene.	➢ UserIdentity ➢ UsuariosOpenSimController ➢ Usuario_sistemaController

Tabla 18. Tarjeta CRC Usistema.

Eventos	
Funcionalidades:	**Clases relacionadas:**
Permite que se pueda realizar la gestión de los datos de ella a partir de un modelo creado de la tabla events de la base de datos.	➢ EventosController

Tabla 19. Tarjeta CRC Eventos.

UsuariosOpenSimController	
Funcionalidades:	**Clases relacionadas:**
Permite realizar las acciones de insertar, eliminar y modificar usuarios del metaverso una vez sea llamada mediante la vista.	➢ Usistema ➢ Usuarios_Finales ➢ OpenSim ➢ UsuarioOpenSim

Tabla 20. Tarjeta CRC UsuariosOpenSimController.

Usuario_sistemaController	
Funcionalidades:	**Clases relacionadas:**
Permite realizar las acciones de insertar, eliminar y modificar usuarios de la aplicación una vez sea llamada mediante la vista.	➢ Usistema

Tabla 21. Tarjeta CRC Usuario_sistemaController.

MostrarController	
Funcionalidades:	**Clases relacionadas:**
Clase controladora de la funcionalidad grupos. Permite que cuando sea llamada la acción de grupos la clase mostrara la vista que posee los grupos que existen en el metaverso y los	➢ Normalizer_ES ➢ OpenSim

integrantes de cada uno de ellos.

Tabla 22. Tarjeta CRC MostrarController.

EventosController	
Funcionalidades:	**Clases relacionadas:**
Clase controladora de la funcionalidad Eventos. Permite realizar acciones de mostrar, insertar, elimina y modificar un evento.	➢ Eventos

Tabla 23. Tarjeta CRC EventosController.

AmigosController	
Funcionalidades:	**Clases relacionadas:**
Clase controladora de la funcionalidad amigos. Permite que cuando sea llamada la acción de amigos la clase permitirá obtener los datos que hagan falta y mostrar la vista correspondiente a los amigos.	➢ Normalizer_ES ➢ OpenSim

Tabla 24. Tarjeta CRC AmigosController.

PerfilController	
Funcionalidades:	**Clases relacionadas:**
Clase controladora de la funcionalidad perfil. Permite que cuando sea llamada la acción de perfil la clase facilitará obtener los datos que hagan falta y mostrar la vista correspondiente al perfil.	➢ Normalizer_ES ➢ OpenSim

Tabla 25. Tarjeta CRC AmigosController.

2.7.4 Análisis de la base de datos del sistema

El sistema para poder realizar todas las funcionalidades que se requieren tiene que trabajar con dos bases de datos, la del servidor OpenSim y la de la aplicación. A la hora de definir la base de datos de la aplicación se realizó el diseño a partir de las tablas que eran necesarias para brindar los servicios al servidor. Cada uno de los módulos que brinda la aplicación cuenta con un número de tablas necesarias para su correcto funcionamiento. Para mejor entendimiento se dará una explicación de las tablas que se necesitan por módulo.

Para el módulo XmlRpcGroup se estableció el uso de las tablas:

- osgroup: guarda la información de los grupos.
- osgroupinvite: almacena las invitaciones que fueron hechas a usuarios para incorporarse a los grupos.
- osgroupmembership: para guardar información de las personas integradas a un grupo.
- osgroupnotice: recoge las noticias de los grupos.
- osgrouprolemembership: para almacenar la definición de los roles de los integrantes de los grupos
- osrole: para guardar los roles que se darán en los grupos.

Con el control de estos datos se puede llevar a cabo una mayor organización de los datos de los grupos que se generen en el metaverso. En el módulo ossearch se definen:

- parcels: almacena los datos de un espacio de terreno
- parcelsales: guarda los datos de los espacios vendidos.
- populasplaces: se definen los lugares populares.
- clasiffieds: guarda anuncios clasificados que son creados por un usuario.
- events: posee los datos de los eventos que se realizaran en el mundo virtual. Como en la aplicación se implementa la funcionalidad de administrar eventos, esta tabla es propia además de la aplicación.

Explicada la estructura que posee cada uno de los módulos se define que para la aplicación fue necesario agregar las tablas categorías, duración_eventos, contactos, usuario, usuario_OpenSim, y usuario_OpenSim_sistema. Dichas tablas son necesarias para gestionar la creación de usuarios en el metaverso, así como para gestionar los eventos que se realicen en este mundo inmersivo.

Conclusiones parciales

Luego de culminado el capítulo se puede definir que la exploración, planificación y diseño fue dirigida a la organización que tendrá luego el desarrollo del proyecto. Se destacan como funcionalidades más importantes, registrarse con el dominio a la aplicación (para garantizar la seguridad de la información con la que se trabaje en la aplicación), crear avatar con los datos que se obtengan de esos servicios (lo que garantiza la confiabilidad de la información que van a poseer las cuentas creadas) y la configuración de los módulos Ossearch y XmlRpcGroup que permite agregar funcionalidades al metaverso OpenSim. Se definieron las iteraciones más importantes por iteración, estableciéndose que las iteraciones de más peso son la 1, 2 y 3. También la planificación que se llevó a cabo permitió asignar el tiempo necesario para realizar cada historia de usuario y evita que se tengan que agregar más iteraciones de las que se habían previsto. Al establecer el diseño de la aplicación la estructura dada a la lógica de las clases mediante el patrón MVC permitirá una optimización del tiempo y un orden a la hora de desarrollar la solución. También se puede concluir que en el desarrollo de la aplicación el uso de dos bases de datos, permite un mejor funcionamiento y control a la hora de trabajar con los datos que se generan de la aplicación y de metaverso OpenSim.

Capítulo 3: Implementación y pruebas del sistema

Introducción

Luego de realizada el diseño de un proyecto, es necesario establecer tareas en las iteraciones de implementación para guiar el trabajo. En el desarrollo de este capítulo se apreciará como se llevará a cabo la implementación el sistema, definiéndose diferentes tareas por iteraciones. El capítulo culminará con las pruebas que se realizan en conjunto con el cliente para definir si el sistema cumple con los objetivos definidos.

3.1 Fase de implementación

La metodología XP propone comenzar la implementación de la solución a partir de una arquitectura lo más flexible posible, con el propósito de que los desarrolladores puedan reestructurar el sistema sin cambiar su comportamiento y así remover duplicaciones de código, mejorar la comunicación, simplificar el código o agregar flexibilidad [PRESSMAN 2005].

3.1.1 Primera iteración

El objetivo de la primera iteración es el desarrollo de la historia de usuario No 1: Acceso a la aplicación. Esta iteración culmina con una aplicación web básica que permita acceder correctamente mediante usuario y contraseña del dominio UCI.

Para ello se trazaron cuatro (4) tareas, que se indican a continuación:

- Tarea No1: Diseño de la página principal de la aplicación.
- Tarea No2: Desarrollo de la aplicación básica.
- Tarea No3: Estudio y selección de los servicios que brinda la UCI para la recopilación de datos.
- Tarea No4: Programación de las clases para el correcto acceso a la aplicación.

Tareas	
Número de tarea: 1	**Número de HU:**1
Nombre de la tarea: Diseño de la página principal de la aplicación.	
Tipo de tarea: Diseño	**Estimación:** 3 días

Diseñador responsable: Lorenzo
Descripción: Realizar el diseño de la aplicación.

Tabla 26. Tarea 1 Iteración 1.

Tareas	
Número de tarea: 2	**Número de HU:**1
Nombre de la tarea: Desarrollo de la aplicación básica.	
Tipo de tarea: Desarrollo	**Estimación:** 5 días
Programador responsable: Enelis Blanca Cuba Rondón.	
Descripción: Realizar mediante el framework Yii una aplicación básica y realizarle algunos cambios para ajustarla a los requisitos que debe tener la aplicación final a desarrollar.	

Tabla 27. Tarea 2 Iteración 1.

Tareas	
Número de tarea: 3	**Número de HU:**1
Nombre de la tarea: Estudio y selección de los servicios que brinda la UCI para la recopilación de datos.	
Tipo de tarea: Investigativa	**Estimación:** 2 días
Programador responsable: Enelis Blanca Cuba Rondón.	
Descripción: Estudiar los diferentes servicios que brinda la Universidad como LDAP y SOAP. Definir cuáles son los necesarios para obtener los datos requeridos para registrarse correctamente.	

Tabla 28. Tarea 3 Iteración 1.

Tareas	
Número de tarea: 4	**Número de HU:**1
Nombre de la tarea: Programación de las clases para el correcto acceso a la aplicación	

Tipo de tarea: Desarrollo	**Estimación:** 5 días
Programador responsable: Enelis Blanca Cuba Rondón.	
Descripción: Desarrollar las clases necesarias para la autenticación correcta del usuario, dígase UserIdentity y LDAP.	

Tabla 29. Tarea 4 Iteración 1.

Al culminar el desarrollo de la iteración se realizó las pruebas de aceptación No 1 y que se explica en el acápite 3.3. El desarrollo de esta iteración estuvo en tiempo.

3.1.2 Segunda iteración

El objetivo de la segunda iteración es el desarrollo de la historia de usuario No 2: Creación de avatar para OpenSim. Esta iteración culmina con una aplicación web básica que permita una vez registrado crear un avatar para poder acceder a la aplicación OpenSim, así como opciones de modificación y eliminación de avatar. Luego de obtener todo esto se tendría el objetivo principal del producto.

Para desarrollar la segunda iteración se definieron cinco (5) tareas:

- Tarea No1: Desarrollo de los modelos necesarios para gestionar la información de usuarios.
- Tarea No2: Establecimiento de la relación entre los modelos y los datos que se necesitan obtener de los servicios de la Universidad.
- Tarea No3: Diseño del modelo de datos del sistema.
- Tarea No4: Desarrollo de funciones necesarias para guardar la información obtenida.
- Tarea No5: Integración de funcionalidades y modelos.

Tareas	
Número de tarea: 1	**Número de HU:**2
Nombre de la tarea: Desarrollo de los modelos necesarios para gestionar la información de usuarios.	
Tipo de tarea: Desarrollo	**Estimación: 3** días
Programador responsable: Enelis Blanca Cuba Rondón.	

Descripción: Programación de los modelos Usistema, UsuariosFinales y UsuariosOpenSim. Estos modelos son necesarios para obtener datos y garantizar la seguridad de los datos a través de funciones definidas.

Tabla 30. Tarea 1 Iteración 2.

Tareas	
Número de tarea: 2	**Número de HU:**2
Nombre de la tarea: Establecimiento de la relación entre los modelos y los datos que se necesitan obtener de los servicios de la Universidad.	
Tipo de tarea: Desarrollo	**Estimación:** 3 días
Programador responsable: Enelis Blanca Cuba Rondón**.**	
Descripción: Programación de funciones que permitan el envío de los datos obtenidos de los servicios de la Universidad hasta los modelos.	

Tabla 31. Tarea 2 Iteración 2.

Tareas	
Número de tarea: 3	**Número de HU:**2
Nombre de la tarea: Diseño del modelo de datos del sistema.	
Tipo de tarea: Diseño	**Estimación:** 3 días
Programador responsable: Enelis Blanca Cuba Rondón.	
Descripción: Diseño del modelo de datos de la aplicación. Establecimiento de las tablas necesarias para cumplir con los objetivos de la aplicación.	

Tabla 32. Tarea 3 Iteración 2.

Tareas	
Número de tarea: 4	**Número de HU:**2
Nombre de la tarea: Desarrollo de funciones necesarias para guardar la información obtenida.	

Tipo de tarea: Desarrollo	**Estimación:** 3 días
Programador responsable: Enelis Blanca Cuba Rondón.	
Descripción: Desarrollo de la clase OpenSim, en donde se establecen funciones encargadas de guardar y obtener datos del servidor del metaverso. Además se desarrollaron en los modelos las diferentes funciones necesarias para guardar información en la base de datos de la aplicación.	

Tabla 33. Tarea 4 Iteración 2.

Tareas	
Número de tarea: 5	**Número de HU:**2
Nombre de la tarea: Integración de funcionalidades y modelos.	
Tipo de tarea: Desarrollo	**Estimación:** 3 días
Programador responsable: Enelis Blanca Cuba Rondón	
Descripción: Integración de las funciones programadas con los modelos para lograr el objetivo planteado en la iteración, que es la gestión de avatar del metaverso OpenSim.	

Tabla 34. Tarea 5 Iteración 2.

Al culminar el desarrollo de la iteración se realizó las pruebas de aceptación No 2 y 3 que se explican en el acápite 3.3. El desarrollo de esta iteración estuvo en tiempo.

3.1.3 Tercera iteración

La tercera iteración se centra en el desarrollo de las HU No 3: Configuración de módulos que brindan servicios web a OpenSim. Con el desarrollo de esta iteración se establecerán en la aplicación cuatro módulos que permitirá al servidor OpenSim poseer servicios que le permitirán gestionar la información de la comunidad de usuarios.

Para el desarrollo de esta iteración fue necesario desarrollar cuatro (4) tareas:

- Tarea No 1: Análisis del módulo XmlRpcGroup.
- Tarea No 2: Configuración del módulo XmlRpcGroup.
- Tarea No 3: Análisis del módulo ossearch.
- Tarea No 4: Configuración del módulo ossearch.

Tareas	
Número de tarea: 1	**Número de HU:**3
Nombre de la tarea: Análisis del módulo XmlRpcGroup.	
Tipo de tarea: Configuración**.**	**Estimación:** 2 día
Programador responsable: Enelis Blanca Cuba Rondón.	
Descripción: Se realizó un análisis con todos los módulos XmlRpcGroup que han existido. Se llevó a cabo un estudio de las clases que lo conforman para un mejor entendimiento a la hora de establecer la relación entre el metaverso OpenSim y la aplicación.	

Tabla 35. Tarea 1 Iteración 3.

Tareas	
Número de tarea: 2	**Número de HU:**3
Nombre de la tarea: Configuración del módulo XmlRpcGroup.	
Tipo de tarea: Configuración	**Estimación:** 2 día
Programador responsable: Enelis Blanca Cuba Rondón	
Descripción: Se agregó el módulo XmlRpcGroup al proyecto. Se modificó el archivo config y se establecieron como parámetro los datos correspondiente a la base de datos de la aplicación. Se modificó el archivo OpenSim.ini, en donde se establecieron parámetros en la función (Group) que son necesarios para acceder a este servicio web y para poder levantarlo.	

Tabla 36. Tarea 2 Iteración 3.

Tareas	
Número de tarea: 3	**Número de HU:**3
Nombre de la tarea: Análisis del módulo ossearch.	
Tipo de tarea: Configuración	**Estimación:** 3 días

Programador responsable: Enelis Blanca Cuba Rondón
Descripción: Se realizó un análisis de las clases que conformaban al módulo para poder lograr la relación entre el metaverso y la aplicación. Se hizo un análisis de cómo trabaja el componente ossearch con el archivo sql.

Tabla 37. Tarea 3 Iteración 3.

Tareas	
Número de tarea: 4	**Número de HU:**3
Nombre de la tarea: Configuración del módulo ossearch.	
Tipo de tarea: Configuración	**Estimación:** 3 día
Programador responsable: Enelis Blanca Cuba Rondón	
Descripción: Se agregó el módulo ossearch al proyecto. Se modificó el archivo search_config y se establecieron como parámetro los datos correspondiente a la base de datos de la aplicación. Se agregó la biblioteca OpenSimSearch.Modules.dll en los archivos del servidor y se configuró la función [Search] del fichero OpenSim.ini para permitir que el servidor acceda al servicio web que brinda este módulo.	

Tabla 38. Tarea 4 Iteración 3.

Al culminar el desarrollo de la iteración se realizó las pruebas de aceptación No 4 y 5 que se explican en el acápite 3.3. El desarrollo de esta iteración estuvo en tiempo.

3.1.4 Cuarta iteración

La cuarta iteración centra su desarrollo en el cumplimiento de las HU No 4: Servicio de muestra de amigos de usuarios, No5: Servicio de muestra de grupos de usuarios y No6: Servicio de muestra del perfil usuarios. Con el desarrollo de esta iteración la aplicación poseerá funcionalidades que permitirá a los usuarios ver quiénes son sus amigos y si están conectados, cuáles son los grupos que existen en el metaverso, los integrantes de cada grupos y poder acceder al perfil que poseen en el mundo virtual.

Para el desarrollo de esta iteración fue necesario desarrollar cuatro (4) tareas:

- Tarea No 1: Implementación de funciones para mostrar amigos.
- Tarea No 2: Desarrollo de métodos para mostrar grupos e integrantes de grupos.
- Tarea No 3: Integración de funciones de amigos y grupos con visual.
- Tarea No 4: Desarrollo de funcionalidad mostrar perfil.

Tareas	
Número de tarea: 1	**Número de HU:**4
Nombre de la tarea: Implementación de funciones para mostrar amigos.	
Tipo de tarea: Implementación	**Estimación:** 4 días
Programador responsable: Enelis Blanca Cuba Rondón	
Descripción: Implementación de la función Amigos que se encuentra en el archivo php OpenSim.mysql. Esta función permite obtener los amigos de la persona que se encuentra registrada en ese momento en la aplicación. Además permite obtener el estado de los amigos de esa persona, dígase en línea o desconectado.	

Tabla 39. Tarea 1 Iteración 4.

Tareas	
Número de tarea: 2	**Número de HU:**5
Nombre de la tarea: Desarrollo de métodos para mostrar grupos e integrantes de grupos.	
Tipo de tarea: Implementación	**Estimación:** 4 días
Programador responsable: Enelis Blanca Cuba Rondón	
Descripción: Desarrollo de las funciones Grupos e Integrantes_Grupos presentes en el archivo OpenSim.mysql.php. Dichas funciones permiten obtener los grupos que están presentes en el mundo virtual y las personas que los integran.	

Tabla 40. Tarea 2 Iteración 4.

Tareas	
Número de tarea: 3	**Número de HU:**3

Nombre de la tarea: Integración de funciones de amigos y grupos con visual.	
Tipo de tarea: Implementación	**Estimación:** 3 días
Programador responsable: Enelis Blanca Cuba Rondón	
Descripción: Se crea el controlador AmigosController y la vista Amigos. Se establece una relación entre ellos y se determina en el controlador la llamada al método implementados en la clase OpenSim, luego se define en el archivo index.php de la vista Amigos como se va a mostrar el contenido al usuario.	

Tabla 41. Tarea 3 Iteración 4.

Tareas	
Número de tarea: 4	**Número de HU:**3
Nombre de la tarea: Desarrollo de funcionalidad mostrar perfil.	
Tipo de tarea: Implementación	**Estimación:** 4 días
Programador responsable: Enelis Blanca Cuba Rondón	
Descripción: Para el desarrollo de la funcionalidad mostrar perfil se hizo necesario, primeramente a la hora de que el usuario se registra por primera vez obtener los datos. Esta primera parte se logra con la implementación de la primera iteración. Luego se crea una clase controladora llamada PerfilController y en ella se definen con una llamada a la base de datos cuales son los parámetros que serán mostrados en el visual. En la vista Perfil se define el diseño de cómo se mostrarán los datos referentes a los datos obtenidos del servicio LDA-UCI y los obtenidos a partir de la información creada en el servidor OpenSim.	

Tabla 42. Tarea 4 Iteración 4.

Al culminar el desarrollo de la iteración se realizó las pruebas de aceptación No 9 y 11 que se explica en el acápite 3.3. El desarrollo de esta iteración estuvo en tiempo.

3.1.5 Quinta iteración

La tercera iteración se centra en el desarrollo de las HU No 6: Servicio de muestra de eventos, la No7: Mostrar mapa y la No8: Desarrollar la interfaz de la aplicación. Con el desarrollo de esta iteración se establecerán en la aplicación dos funcionalidades que permitirán al usuario ver los eventos que se ofrecen y observar el mapa del metaverso. Además esta es la iteración que da cumplimiento con todos los objetivos establecidos.

Para el desarrollo de esta iteración fue necesario desarrollar cinco (5) tareas:

- Tarea No 1: Implementación de funcionalidades para gestionar los eventos.
- Tarea No 2: Integración de funciones de eventos con elementos visuales.
- Tarea No 3: Desarrollo de la funcionalidad Mostrar mapa.
- Tarea No 4: Desarrollo de la interfaz visual de la aplicación en general.

Tareas	
Número de tarea: 1	**Número de HU:**6
Nombre de la tarea: Implementación de funcionalidades para gestionar los eventos.	
Tipo de tarea: Diseño	**Estimación:** 5 días
Programador responsable: Enelis Blanca Cuba Rondón	
Descripción: Para el desarrollo de la funcionalidad eventos se creó un modelo Eventos y un CRUD para la gestión de toda la información referente a ellos. Para obtener datos necesarios a mostrar en los eventos se desarrollaron los métodos GetDuración, GetCategoría y GetNombre en el modelo Evento.	

Tabla 43. Tarea 1 Iteración 5.

Tareas	
Número de tarea: 2	**Número de HU:**6
Nombre de la tarea: Integración de funciones de eventos con elementos visuales.	
Tipo de tarea: Implementación	**Estimación:** 5 días

Programador responsable: Enelis Blanca Cuba Rondón
Descripción: Para la visualización de los datos de los eventos y la creación se hizo necesario hacer una modificación al CRUD que se genera automáticamente con el marco de trabajo Yii. Se definió la manera en que se iba a mostrar las categorías, la duración del evento y la hora al usuario. Se desarrollaron elementos visuales para mostrar los datos como los: CGridView (para la gestión de los eventos), CListView (listar datos de los eventos) y CDetailView (para listar datos de un elemento en específico).

Tabla 44. Tarea 2 Iteración 5.

Tareas	
Número de tarea: 3	**Número de HU:**7
Nombre de la tarea: Desarrollo de la funcionalidad Mostrar mapa.	
Tipo de tarea: Implementación	**Estimación:** 5 días
Programador responsable: Enelis Blanca Cuba Rondón	
Descripción: Para el desarrollo de la funcionalidad mostrar mapa se desarrolló un visual Mapa, en donde se obtiene el mapa del servidor OpenSim mediante una dirección que envía el servidor. El mapa cuenta con una leyenda que informa al usuario algo de los objetos que componen dicha imagen.	

Tabla 45. Tarea 3 Iteración 5.

Tareas	
Número de tarea: 4	**Número de HU:**8
Nombre de la tarea: Desarrollo de la interfaz visual de la aplicación en general.	
Tipo de tarea: Implementación	**Estimación:** 5 días
Programador responsable: Enelis Blanca Cuba Rondón	

Descripción: Para la culminación del desarrollo de la aplicación se lleva a cabo un establecimiento del CSS en general. Se integraran todos los componentes de la aplicación dígase vistas. Se establece el banner de la aplicación. Los colores con los que se presentaran los datos.

Tabla 46. Tarea 4 Iteración 5.

Al culminar el desarrollo de la iteración se realizó las pruebas de aceptación No 6, 7, 8 y 10 que se explican en el acápite 3.3. El desarrollo de esta iteración estuvo en tiempo.

3.2 Fase de pruebas

El proceso de pruebas es uno de los pilares fundamentales de la metodología XP, el cual ayuda al cliente a verificar y concretar las funcionalidades de las HU, por lo que favorece la comunicación entre el cliente y el equipo de desarrollo. Esta filosofía ayuda a identificar y corregir fallos u omisiones cometidas en las mismas, por lo que se reduce el número de errores no detectados así como el tiempo entre la introducción de éste en el sistema y su detección [PRESSMAN 2005].

3.2.1 Pruebas de aceptación

Las pruebas de aceptación son realizadas por el propio cliente en compañía de uno de los representantes del equipo de desarrollo y se orientan a las funcionalidades del sistema. Su objetivo es comprobar, desde la perspectiva del usuario final, el cumplimiento de las especificaciones de la lista de reservas del producto. A continuación, aparecen las pruebas de aceptación realizadas a la solución propuesta:

Caso de Prueba de Aceptación	
Código: P1H1	**Número de HU:**1
Nombre: Registrarse en la aplicación con el dominio	
Descripción: Demostrar que el usuario puede acceder a la aplicación mediante su cuenta del dominio UCI.	
Condiciones de Ejecución: Debe estar levantado el servidor web donde está montada la aplicación.	
Entrada/Pasos de ejecución: ➢ El usuario debe poner en el explorador la dirección de la aplicación.	

➢ En el panel de registrarse el usuario debe poner su usuario y contraseña donde se le indica. ➢ Luego debe dar sobre el botón entrar.
Resultado esperado: Se espera que cuando se hayan cumplido los pasos de ejecución el usuario entre a la aplicación correctamente y se le sea dada la bienvenida.
Evaluación de la prueba: Resultado satisfactorio.

Tabla 47. Prueba de aceptación No 1.

Caso de Prueba de Aceptación	
Código: P2H2	**Número de HU:**2
Nombre: Crear avatar	
Descripción: Se debe probar que el usuario se crea correctamente.	
Condiciones de Ejecución: Estar registrado en la aplicación.	
Entrada/Pasos de ejecución: ➢ El usuario se registra. ➢ Accede a la funcionalidad mediante el menú Crear Avatar. ➢ Pasa a introducir los datos que son necesarios como contraseña y seleccionar la región. ➢ Se debe culminar la funcionalidad al acceder a la opción Crear.	
Resultado esperado: Se espera que se cree un avatar capaz de entrar al metaverso OpenSim y pueda realizar todas las funcionalidades que el mundo ofrece. Además la aplicación web una vez creada el avatar debe mostrar todos los datos que el usuario uso para crear el avatar.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 48. Prueba de aceptación No 2.

Caso de Prueba de Aceptación	
Código: P3H2	**Número de HU:**2
Nombre: Eliminar avatar	

Descripción: Se debe probar que el usuario se elimine correctamente.
Condiciones de Ejecución: Estar registrado en la aplicación y tener un usuario creado.
Entrada/Pasos de ejecución: ➢ El usuario se registra. ➢ Accede a la funcionalidad mediante el menú Gestionar Avatar. ➢ Luego selecciona la opción Eliminar avatar que se encuentra en el panel derecho de la interfaz a la que se accedió. ➢ Cuando se elimina el avatar se muestra la opción de crear uno nuevo.
Resultado esperado: Se espera que se elimine el avatar correctamente del servidor y de la base de datos de la aplicación.
Evaluación de la prueba: Resultado satisfactorio.

Tabla 49. Prueba de aceptación No 3.

Caso de Prueba de Aceptación	
Código: P4H3	**Número de HU:**3
Nombre: Funcionamiento del módulo XmlRpcGroup.	
Descripción: Se debe comprobar que en OpenSim se puedan crear grupos.	
Condiciones de Ejecución: Tener un usuario creado en la aplicación para poder acceder a OpenSim y estar dentro del mundo virtual.	
Entrada/Pasos de ejecución: ➢ El usuario entra al mundo virtual OpenSim. ➢ Dar clic derecho sobre el avatar y seleccionar la funcionalidad Grupos. ➢ Seleccionar de la pantalla que aparecerá la opción crear. ➢ Crear un grupo con los parámetros que especifica el metaverso y guardar los datos. ➢ Verificar en la pantalla del tercer paso que el grupo se allá creado.	
Resultado esperado: Se espera que se cree un grupo en el metaverso y que aparezcan los datos	

guardados en la base de datos de la aplicación.
Evaluación de la prueba: Resultado satisfactorio.

Tabla 50. Prueba de aceptación No 4.

Caso de Prueba de Aceptación	
Código: P5H3	**Número de HU:**3
Nombre: Funcionamiento del módulo ossearch.	
Descripción: Se debe comprobar que en OpenSim se puedan llevar a cabo búsquedas de grupos.	
Condiciones de Ejecución: Tener un usuario creado en la aplicación para poder acceder a OpenSim y estar dentro del mundo virtual.	
Entrada/Pasos de ejecución: ➢ El usuario entra al mundo virtual OpenSim. ➢ Selecciona la opción “Buscar” que se encuentra en el panel inferior de la aplicación Imprudence. ➢ Selecciona el panel grupos e introduce el nombre del grupo que desea buscar. ➢ Dar en la opción OK para obtener los datos del grupo.	
Resultado esperado: Aparezca la información del grupo deseado y en caso de no estar el grupo informar con un mensaje.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 51. Prueba de aceptación No 5.

Caso de Prueba de Aceptación	
Código: P6H6	**Número de HU:**6
Nombre: Crear evento.	
Descripción: Se verificará si en la aplicación se puede crear un evento.	
Condiciones de Ejecución: Tener un usuario creado y estar en línea en la aplicación.	

Entrada/Pasos de ejecución: ➢ El usuario accede a la funcionalidad evento. ➢ Llena los campos que le son solicitados. ➢ Dar clic sobre el botón crear. ➢ Aparece un listado todos los eventos incluyendo el que acaba de ser creado.
Resultado esperado: Se espera aparezca el evento como creado.
Evaluación de la prueba: Resultado satisfactorio.

Tabla 52. Prueba de aceptación No 6.

Caso de Prueba de Aceptación	
Código: P7H6	**Número de HU:** 6
Nombre: Modificar evento.	
Descripción: Se verificará si en la aplicación se puede modificar un evento.	
Condiciones de Ejecución: Tener un usuario creado y estar en línea en la aplicación.	
Entrada/Pasos de ejecución: ➢ El usuario accede a la funcionalidad avatar. ➢ Dar clic sobre la operación administrar avatar. ➢ Luego aparecerá un panel con todos los eventos del usuario, seleccionar la opción que está definida por un lápiz en el evento que desea modificar. ➢ Sobrescribir los datos que desea modificar. ➢ Dar clic sobre el botón Enviar. ➢ Se mostrara como ha quedado el evento luego de ser modificado.	
Resultado esperado: Se espera que se modifiquen los valores de un evento específico.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 53. Prueba de aceptación No 7.

Caso de Prueba de Aceptación

Código: P8H5	**Número de HU:** 5
Nombre: Mostrar grupo.	
Descripción: Se verificará si la aplicación muestra los grupos que existen en el metaverso.	
Condiciones de Ejecución: Tener un usuario creado y estar en línea en la aplicación.	
Entrada/Pasos de ejecución: ➢ El usuario accede a la funcionalidad “Grupo”. ➢ Acceder al panel de grupos y obtener los datos.	
Resultado esperado: Se espera que se muestren datos de los grupos que existen en el metaverso tales como el nombre, el creador, el tema y la cantidad de personal que posee.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 54. Prueba de aceptación No 8.

Caso de Prueba de Aceptación	
Código: P9H4	**Número de HU:** 4
Nombre: Mostrar amigos.	
Descripción: Se verificará si la aplicación muestra los amigos que posee el usuario en el metaverso y el estado en que se encuentran.	
Condiciones de Ejecución: Tener un usuario creado y estar en línea en la aplicación.	
Entrada/Pasos de ejecución: ➢ El usuario accede a la funcionalidad “Amigos”. ➢ Presenciar el panel “Amigos”.	
Resultado esperado: Se espera que se muestren los nombres de los amigos que posee el usuario en el metaverso y el estado en que se encuentra en ese momento ya sea en línea o desconectado.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 55. Prueba de aceptación No 9.

Caso de Prueba de Aceptación	
Código: P10H8	**Número de HU:** 8
Nombre: Mostrar mapa.	
Descripción: Se verificará si la aplicación muestra una interfaz que refleje el mapa del metaverso**.**	
Condiciones de Ejecución: Tener un usuario creado y estar en línea en la aplicación.	
Entrada/Pasos de ejecución: ➢ El usuario accede a la funcionalidad “Mapa”.	
Resultado esperado: Debe aparecer el mapa del metaverso en la vista y debe aumentar una vez se pase el ratón de la pc encima de la imagen.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 56. Prueba de aceptación No 10.

Caso de Prueba de Aceptación	
Código: P11H7	**Número de HU:** 7
Nombre: Mostrar perfil de amigos.	
Descripción: Se debe mostrar en la aplicación una interfaz que muestre los datos de un amigo seleccionado por el usuario.	
Condiciones de Ejecución: Tener un usuario creado y estar en línea en la aplicación.	
Entrada/Pasos de ejecución: ➢ El usuario accede a la funcionalidad “Perfil”. ➢ Seleccionar en la vista que aparece el amigo del que desea obtener los datos del perfil.	
Resultado esperado: Se espera que se muestre una vista con los datos del perfil del amigo y en caso de no tenerlo se muestra un mensaje que indica que el usuario no posee amigos en el metaverso.	

Evaluación de la prueba: Resultado satisfactorio.

Tabla 57. Prueba de aceptación No 11.

Las pruebas realizadas anteriormente fueron enfocadas al correcto funcionamiento de los requisitos del sistema. Además de ellas se realizaron pruebas dirigidas a demostrar la seguridad y confiabilidad de los datos. A continuación se muestran las más importantes de las que se realizaron:

Caso de Prueba de Aceptación	
Código: P12H1	**Número de HU:** 1
Nombre: Validar el registro de usuarios a la aplicación.	
Descripción: Se deben poner usuarios y contraseñas incorrectas en el papel de registro para verificar la respuesta de la aplicación.	
Condiciones de Ejecución: Acceder a la aplicación por su dirección URL.	
Entrada/Pasos de ejecución: ➢ El usuario pone un usuario que no se encuentra en el dominio UCI. ➢ El usuario pone una contraseña incorrecta. ➢ Luego se le da al botón entrar.	
Resultado esperado: Se espera que la aplicación muestre el mensaje “Usuario o Contraseña Incorrecta”.	
Evaluación de la prueba: Resultado satisfactorio.	

Tabla 58. Prueba de aceptación No 12.

Caso de Prueba de Aceptación	
Código: P13H2	**Número de HU:** 2
Nombre: Validar el registro directos de datos a través del método POST y GET.	
Descripción: Pasar valores de una funcionalidad a través de la URL sin estar registrado.	

Condiciones de Ejecución: Tener levantada la aplicación.
Entrada/Pasos de ejecución: ➢ Pasar por la URL de la aplicación los valores de una funcionalidad mediante los métodos POST y GET
Resultado esperado: Se espera que no suceda nada, es decir que no se guarden valores a la base de datos mediante este método.
Evaluación de la prueba: Resultado satisfactorio.
Observación: La seguridad en la prueba está dada porque cada controlador contiene un método de reglas de acceso que define que permiso poseen los tipos de usuarios que interactúan con la aplicación.

Tabla 59. Prueba de aceptación No 13.

Conclusiones parciales

Luego de realizadas las iteraciones, se reconoció la importancia que se tuvo al organizar su desarrollo por tareas, pues permitieron evitar duplicación de esfuerzos por parte del desarrollador. A medida que se implementaban las tareas de las iteraciones, se lograba un avance y orden en el proceso de desarrollo de la solución; el tiempo que se le asignó a cada una de estas iteraciones fue justo el que se necesitó para su desarrollo. Cada vez que se terminaba una iteración se realizaban pruebas de aceptación, lo que posibilitó que el cliente obtuviera resultados desde etapas tempranas del desarrollo de la aplicación. Todas las pruebas que se realizaron tuvieron éxito.

Conclusiones

A través del desarrollo de este trabajo se obtuvo una aplicación que permite la creación de cuentas del metaverso OpenSim con la información que se gestiona en los servicios de la Universidad. Además permite brindar servicios web al mundo virtual que le agregan funcionalidades. Con el uso de esta aplicación el usuario tendrá acceso a información que se gestiona en el mundo virtual en tiempo real.

Luego de analizado el proceso se llegaron a las siguientes conclusiones:

- El proceso de instalación y montaje de un metaverso posibilitó un conocimiento necesario para entender los procesos que intervienen en la fusión de información del mundo virtual con los servicios que brinda la Universidad; lo que demostró la posibilidad de integrar el metaverso OpenSim con una aplicación web que permitiera la creación de cuentas definidas con la información adquirida de los servicios de la comunidad y que ofreciera información al usuario de las actividades que ocurrieran en el metaverso.
- El uso de un proceso ágil de desarrollo con elementos de RUP como son: la descripción de los requisitos y el diagrama de clases permitió una organización y claridad a la hora de implementar las funcionalidades de la solución.
- Se determinó que al consumir los servicios de LDAP y los publicados en la UDDI de la Universidad se garantiza la seguridad y confiabilidad de los datos a la hora de crear avatares para el metaverso.
- Al lograr la integración de los módulos *Ossearch* y *XmlRpcGroup* se agregan funcionalidades al metaverso, lo que permite gestionar un mayor volumen de información referente a la actividad del mundo virtual.

Recomendaciones

Para dar continuación con el desarrollo de la aplicación se recomienda:

- Agregar nuevos módulos de servicios a la aplicación para aumentar la funcionalidad del metaverso OpenSim, como por ejemplo, *Profile* para modificar el perfil y MessageOffline para guardar mensajes que fueron enviados a un usuario que se hubo desconectado.
- Desarrollar un sistema de ayuda para la interacción del usuario con la aplicación.

Referencias bibliográficas

ALVAREZ, M. A. *Manual de JQuery*.DESARROLLOWEB.COM, 2010.

APTANA. *Aptana*, 2012. [2012]. Disponible en: http://aptana.com/

BOCH, H. *The Definitive Guide to NetBeans Platfoms 7*. Jim Freeman, 2011. 5-8 p. 978-1-4302-2418-1

CALIDAD, D. D. *Diagnostico 2008 - Resultado de la revisión UCI - Herramientas de modelado*, Universidad de las Ciencias Informáticas, 2008a. 3.5.4.

CALIDAD, D. D. *Diagnóstico 2008 - Resultado de la revisión UCI - Herramientas para la gestión documental*. La habana, Universidad de las Ciencias Informáticas., 2008b. 3.5.2.

CARLOS, J. *Intercambios virtuales en busca del conocimiento.*, 2010

Drupal 2012. [2012]. Disponible en: http://drupal.org/

FORMANTÍN, Y. P.; I. P. SIERRA. *GUIA PARA EL TRABAJO CON EL CMS DRUPAL*. La habana, Universidad de las Ciencias Informáticas., 2007. 5. p.

GUTIERREZ, A. F. *Kumbia PHP Framework*. 19 p.

JACOBSON, I.; G. BOCH*, et al. El Proceso Unificado del Desarrollo de Software*. Malaga, 2000. 3-8 p. 0-201-57169-2

JOOMLA. *Joomla*, 2012. [2012]. Disponible en: http://www.joomla.org/

LARMAN, C. *UML y patrones*. 1ra .1999. 4,5 p. 970-1 7-0261-1

LETELIER, P.; M. C. PENADÉS. *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. VALENCIA, U. P. D. España, 2007.

LINDEN_LAB. *SecondLife*, 2012. [2012]. Disponible en: www.secondlife.com

MAQUINA, E. ; J. D. PARRA. *Guía de Patrones, Prácticas y Arquitectura .NET*, 2008. 2.5:5.

MEDNIEKS, Z.; L. DORNIN*, et al. Programing Android*, 2011. 112.p. 978-1-449-38969-7

MURCIA, U. D. *Metodología de desarrollo de software*, 2011.

OPENSIM. *OpenSimulator*, 2012. [2012]. Disponible en: http://opensimulator.org/wiki/Main_Page

OPENSIMULATOR. *Enabling Groups*. *Groups Overview*, 2012a.

OPENSIMULATOR. *Forge*, 2012b. [2012]. Disponible en: http://forge.opensimulator.org/gf/project/opensimwi/

ORACLE. *Oracle y MySQL*, 2012. [2012]. Disponible en: http://www.oracle.com/partners/es/knowledge-zone/server-storage/mysql-050746-es.html

PARADIMG, V. *Visual Paradimg*, Visual Paradimg, 2012. [2012]. Disponible en: http://www.visual-paradigm.com/

PRESSMAN, *Ingienería del software. Un enfoque práctico*,2005. 4: 84-86.

SOMMERVILLE, I. *Ingienería de Software*. 7ma. Madrid, 2005a. p.84-7829-074-5

SOMMERVILLE, I. *Ingienería de Software*. 7ma. Madrid, 2005b. 108-110 p. 84-7829-074-5

Symfony. 2012. [2012]. Disponible en: http://www.symfony.es/categoria/symfony2/

TECNOLÓGICOS, C. D. E. P. E. D. D. P. *Libro de diagnóstico-IDE o herramientas de desarrollo*. La habana, Universidad de las Ciencias Informáticas, 2009.

TUIS, N. S. L. O. *Opensim/ WEB interface (E)/ Common Simple Settings*, 2011. [2012]. Disponible en: http://www.nsl.tuis.ac.jp/xoops/modules/xpwiki/?plugin=related&page=OpenSim%2FWEB%20Interface%20%28E%29%2FCommon%20Simple%20Settings

VALDÉS, D. P. *Maestros del web*, 2010. [2012]. Disponible en: http://www.maestrosdelweb.com/principiantes/los-diferentes-lenguajes-de-programacion-para-la-web/

VASWANI, V. *Zend Framework: A Beginner's Guide*. The McGraw-Hill Companies, 2010. 1-5 p. 978-0-07-163940-8

WELLMAN, D. *Get started; get to grips with the YUI JavaScript development library!* , 2008. 7-9 p.

XOOPS Cube. 2012. [2012]. Disponible en: http://xoopscube.org/

YIIframework. 2012. [2012]. Disponible en: http://www.yiiframework.com

Zend. 2012. [2012]. Disponible en: http://www.zend.com/en/

Bibliografía

ALVAREZ, M. A. *Manual de JQuery*, 2010.p.

BOCH, H. *The Definitive Guide to NetBeans Platfoms 7*. Jim Freeman, 2011,p. 978-1-4302-2418-1

CALIDAD, D. D. *Diagnostico 2008 - Resultado de la revisión UCI.p.*

FORMANTÍN, Y. P.; I. P. SIERRA. *GUÌA PARA EL TRABAJO CON EL CMS DRUPAL*. La habana, Universidad de las Ciencias Informáticas., 2007.p.

GILFILLAN, I. La biblia MySQL. SYBEX, p.

GUTIERREZ, A. F. *Kumbia PHP Framework*.p.

JACOBSON, I.; G. BOCH*, et al. El Proceso Unificado del Desarrollo de Software*. Malaga, 2000. p. 0-201-57169-2

LARMAN, C. *UML y patrones*. 1999. p. 970-1 7-0261-1

LETELIER, P.; M. C. PENADÉS. *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. VALENCIA, U. P. D. España, 2007.p.

MAQUINA, E. ; J. D. PARRA. *Guía de Patrones, Prácticas y Arquitectura .NET*, 2008.p.

MAKAROV, A. Yii 1.1 Application Development Cookbook. BIRMINGHAM - MUMBAI, Packt Publishing, 2011. p. 978-1-849515-48-1

MATEU, C. Desarrollo de aplicaciones web. Universidad de Cataluña Barcelona, 2004. p. 84-9788-118-4

MEDNIEKS, Z.; L. DORNIN*, et al. Programing Android*, 2011. 112.p. 978-1-449-38969-7

MURCIA, U. D. *Metodología de desarrollo de software*, 2011.p.

PARADIGM., V. Business Process Visual ARCHITECT 2.0 User's Guide. Visual Paradigm., 2007. p.

PÉREZ, J. E. Introducción a JavaScript. 2008. p.

PRESSMAN. *Desarrollo ágil Ingienería del software. Un enfoque práctico*, 2005p.

SKLAR, D.; A. TRACHTENBERG. PHP Cookbook. O'Reilly, 2002. p. 1-56592-681-1

SOMMERVILLE, I. *Ingienería de Software*. 7ma. Madrid, 2005. p. 84-7829-074-5

WELLMAN, D. jQuery UI 1.6 The User Interface Library for jQuery. BIRMINGHAM - MUMBAI, Packt Publishing, 2009. p.

WINESETT, J. Agile Web Application Development with Yii 1.1 and PHP5. BIRMINGHAM - MUMBAI, Packt Publishing, 2010. p. 978-1-847199-58-4

Anexos:

Figura 7. Página principal de Second Life.

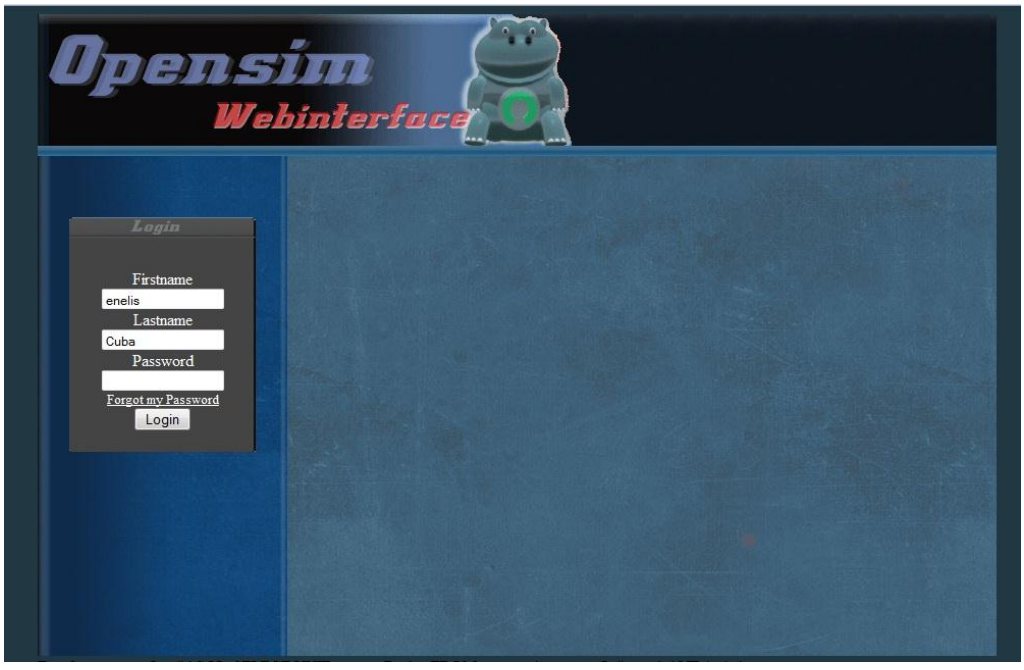


Figura 8. Interfaz visual de OpenSim_Web_interface.

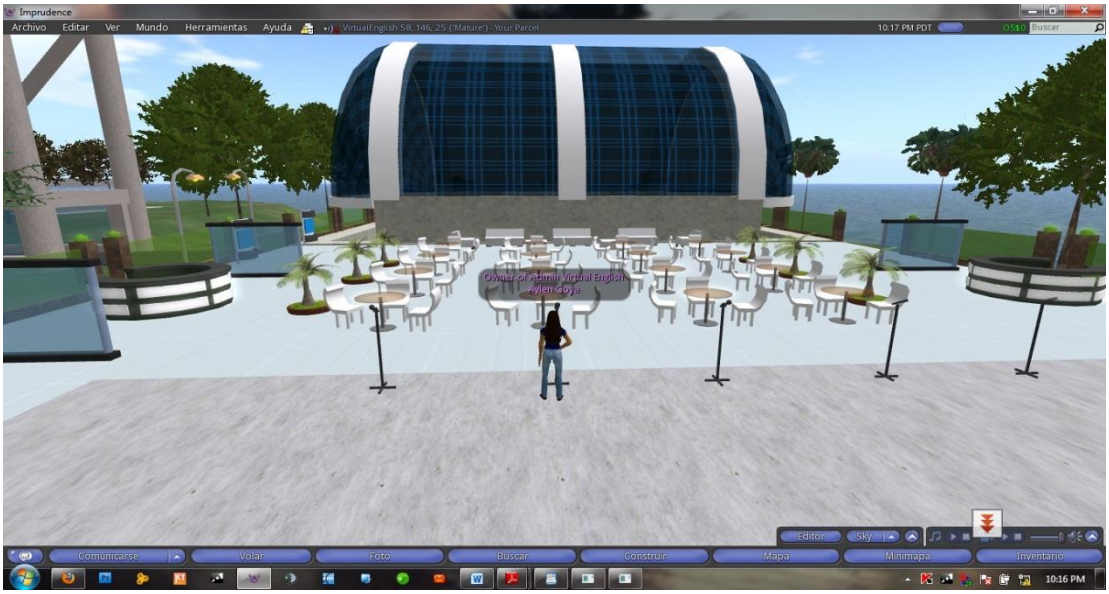


Figura 9. Cafetería en UClgrid.

Figura 10. Piscina olímpica en UClgrid

Glosario de términos

Avatar: Representación gráfica, generalmente humana, que se asocia a un usuario para su identificación. Los avatares pueden ser fotografías o dibujos artísticos, y algunas tecnologías permiten el uso de representaciones tridimensionales.

Biblioteca: En ciencias de la computación, una biblioteca es un conjunto de subprogramas utilizados para desarrollar software.

Ciberespacio: Realidad simulada que se encuentra dentro de los ordenadores y redes del mundo.

Código abierto (Open Source): término con el que se conoce al software distribuido y desarrollado libremente.

Comunidad virtual: Comunidad cuyos vínculos, interacciones y relaciones tienen lugar no en un espacio físico sino en un espacio virtual como Internet.

LDAP: Son las siglas Protocolo Ligero de Acceso a Directorios que hacen referencia a un protocolo a nivel de aplicación el cual permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.

SOAP (Simple Object Access Protocol): Se trata de un protocolo que te permite la comunicación entre aplicaciones a través de mensajes por medio de Internet. Está basado en XML y es la base de los Web Services. Es independiente de la plataforma y del lenguaje.

Soporte: comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas.

Widgets: Componente con el cual el usuario interactúa, ya sea gráfico o de control. Son ejemplo de widget las ventanas, cajas de texto, checkboxs, listbox, entre otros. Son utilizados por los programadores para hacer interfaces gráficas de usuario.