Universidad de las Ciencias Informáticas

Facultad 5

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

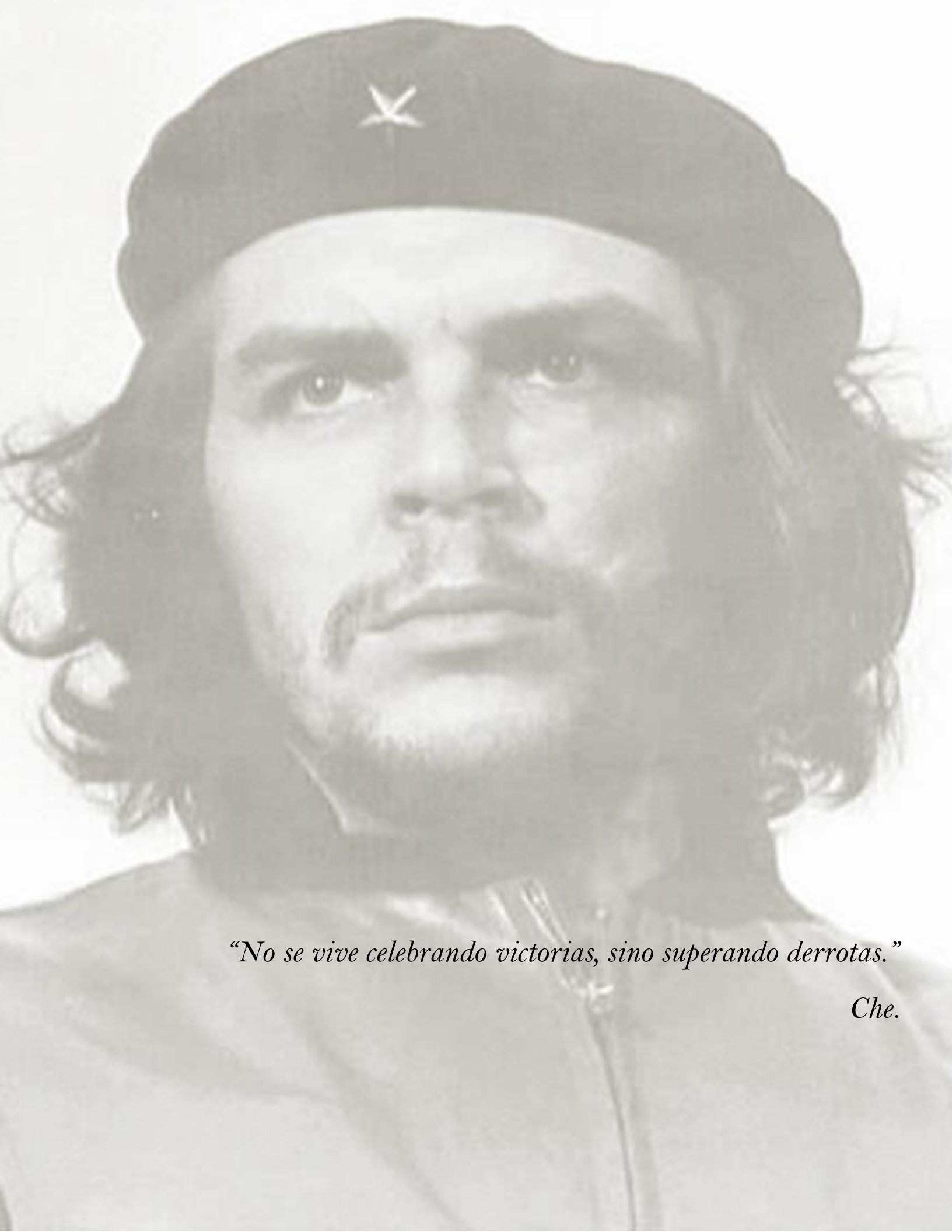
Título: Comunicación con los objetos del entorno virtual OpenSim

Autor: Yusleidy García Vázquez

Tutor: Ing. Minardo Gollún González López

Co-tutor: Ing. Manuel Alberto Ávila Solarana

Ciudad de la Habana, Junio 2012



“No se vive celebrando victorias, sino superando derrotas.”

Che.

DECLARACIÓN DE AUTORÍA

Declaro que soy la única autora de este trabajo y autorizo al tutor Minardo Gollún González López y al Centro 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 _____.

Yusleidy García Vázquez _____

Firma del Autor

Ing. Minardo Gollún González López _____

Firma del Tutor

Datos de Contacto

Nombre y Apellidos: 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 UCI, con ocho años de experiencia en el tema de realidad virtual.

Nombre y Apellidos: Ing. Manuel Alberto Ávila Solarana

Edad: 24 años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

E-mail: maavila

Graduado en la UCI con dos años de experiencia en el tema de Realidad Virtual.

Agradecimientos

A mis padres Magali y Gabriel, por haberme apoyado siempre, por confiar en que era capaz de llegar tan alto en la vida, por esperar de mí una persona mejor.

A mi hermano Yusliel, por aguantarme estos 5 años en la UCI, por estar presente cada vez que lo necesitaba.

A mi cuñada Mairenys, por hacer de mi hermano una mejor persona y por darme lo que más quería, mi bella sobrina.

A mis tíos Orlinda y Carito, por preocuparse tanto por mí como persona, y por mi carrera.

A mi novio Daniel, por ser la persona más especial de este mundo, por darme todo su amor y cariño, por planear una vida juntos y por toda su ayuda en todo momento.

A Javier por toda su ayuda, por aconsejarme y preocuparse por mí.

A Yoandy el pinareño por toda su ayuda.

A mi tutor Minardo y a Lorenzo por toda su ayuda y paciencia.

A mis amigos Miguel y Yoandy, por ser los únicos amigos desde primer año, por apoyarme en mis decisiones, por darme consejos, por ayudarme, por su cariño y amistad.

A mi bruja Enelis por compartir sus momentos conmigo y por dejarme compartir los míos con ella.

A mis compañeros de aula y de apartamento en los cinco años, aunque no los mencione por nombre, porque los tendría que mencionar todos, han sido muy importantes en mi vida, a ellos por los consejos, las ayudas, el apoyo, por hacerme sentir como en familia aun estando lejos de la mía.

A la UCI, por forjarme como ingeniera informática.

A Fidel, por darnos esta oportunidad.

En fin a todas las personas que de una forma u otra han estado presentes en mi vida.

Dedicatoria

Quiero dedicarle este trabajo de diploma especialmente a mi madre por estar siempre cuando la necesito, por ser la mejor madre del mundo.

A mi padre y a mi hermano, a mis tíos y a mi bella sobrina Ilsa.

Resumen

Actualmente la informática es una ciencia que evoluciona constantemente y trae consigo el surgimiento y desarrollo de nuevos campos que ayudan a aumentar los potenciales de esta ciencia, así es el caso de la Realidad Virtual (RV). Existen diversos servidores para el uso de la RV, uno de ellos es OpenSim el cual es un servidor de aplicaciones en 3 dimensiones (3D) que representa un ambiente de la realidad y permite crear mundos virtuales.

La documentación y creación de casos de estudios donde se posibilite la comunicación con los objetos del entorno virtual OpenSim supone un avance para los usuarios de la universidad de las ciencias informática, debido a la integración de un grupo de servicios en el entorno.

A través de la realización de este trabajo de diploma se desarrolló exitosamente dos casos de estudios donde se muestran las posibles aplicaciones virtuales que pudieran desarrollarse en la Universidad de las Ciencias Informáticas (UCI), los cuales cumplen con los requisitos funcionales planteados. Se documentaron procedimientos para lograr la comunicación desde el exterior con el interior de OpenSim para poder dar solución a peticiones realizadas por la dirección de extensión universitaria y la aduana de Cuba. La realización de estos casos de estudio contribuirá al uso de OpenSim en la UCI, haciendo difusión de este tipo de representación virtual en nuestro país.

Palabras clave: Entornos Virtuales, Metaverso, OpenSim

Índice

Índice de Figuras

Índice de Tablas

Introducción

Los adelantos en las Ciencias Informáticas permiten la aparición de tecnologías novedosas que podrán ser aplicadas en diversos campos, ejemplo de esto es la realidad virtual (RV) la cual ha adquirido un gran auge no solo en la informática, sino también en la vida diaria de las personas.

Dentro de las aplicaciones de la RV se encuentran: la medicina, educación, cine, arquitectura, ocio, entre otras. La RV se ha aplicado en la creación de distintos metaversos, éstos son entornos donde los humanos interactúan social y económicamente en un ciberespacio que es una abstracción del mundo real. Los metaversos amplifican las posibilidades y capacidades para el establecimiento de relaciones humanas y para el desarrollo de iniciativas empresariales, educativas y artísticas.

En la Universidad de las Ciencias Informáticas (UCI) existen una gran variedad de proyectos y centros de investigación y desarrollo, el Centro de Informática Industrial (CEDIN) es uno de ellos y en el cual se trabaja con el entorno virtual OpenSim como metaverso. En este metaverso se pueden realizar diversas actividades como realizar exhibiciones, juegos de roles, actuaciones de teatro, clases online, conferencias, talleres y proyectos colaborativos con otras entidades.

Muchas de las universidades prestigiosas en el mundo usan este tipo de metaverso, ejemplos de ellas son: Harvard, Oxford, Stamford, Salamanca, Edimburgo y la universidad politécnica de Madrid (UPM) con la cual la UCI ha tenido varios contactos, donde profesores han venido hasta aquí a demostrar sus resultados en este tipo de representación virtual.

Las personas que trabajan con este metaverso no cuentan con todos los conocimientos que el mismo exige, pero aun así se han desarrollado proyectos para la facultad como son la feria de soluciones informáticas (FESI) y actualmente está en proceso una tesis de doctorado y un paseo virtual de la residencia.

Debido al impacto que han tenido estas nuevas tecnologías se ha generado un creciente interés en clientes tales como la aduana de Cuba y la dirección de extensión universitaria de la UCI, los cuales han realizado peticiones para la creación de proyectos conjuntos para fomentar el entrenamiento del personal y el control de activos tangibles.

Hasta el momento no ha sido posible dar solución a estas peticiones, ejemplo de esto es una petición de la dirección de extensión universitaria la cual solicita registrar virtualmente el estado de los activos fijos tangibles de la universidad mediante objetos que representen los mismos en un entorno 3D.

Otro ejemplo es el de la aduana, la cual solicita realizar en OpenSim el actual proyecto Indicios Aduaneros, es decir, un estudiante aduanero que observe todos los pasajeros que se bajan del avión y según sus indicios (gesticulación, colores del rostro etc.) determine si es sospechoso de algún delito o no. Todo este ejercicio debe quedar registrado en alguna base de datos para que el profesor mediante una aplicación de escritorio evalúe al estudiante aduanero.

Las personas que han trabajado sobre el metaverso en este momento no cuentan con el conocimiento para lograr una comunicación desde el exterior con los objetos en el interior de OpenSim, por lo que se hace necesario elaborar un documento donde quede reflejado un procedimiento que de manera general proponga una solución para lograr el envío de datos a este entorno virtual, demostrando mediante casos de estudio el correcto funcionamiento del procedimiento planteado. Sería de suma importancia la documentación de este trabajo pues serviría de base para la creación de futuros proyectos que pudieran desarrollarse en la universidad sobre OpenSim.

Debido a la situación problemática planteada anteriormente se identifica el siguiente **problema de la investigación** ¿Cómo lograr una comunicación con los objetos de OpenSim? El **objeto de estudio** de la Investigación se enmarcó en el entorno virtual OpenSim. El **campo de acción** de este trabajo abarca la comunicación con los objetos de OpenSim.

Objetivo general: Crear un procedimiento para lograr la comunicación con los objetos del entorno virtual OpenSim.

Objetivo específico: Desarrollar dos casos de estudio donde se ponga en práctica los pasos propuestos para lograr la comunicación con los objetos de OpenSim.

Para cumplir con los objetivos propuestos se trazan siete tareas de la investigación las cuales son:

- Caracterización de los principales conceptos, herramientas, pasos y lenguajes asociados al objeto de estudio
- Caracterización de las diferentes técnicas de comunicación que se puedan desarrollar en OpenSim
- Selección de técnica para comunicarse con los objetos de OpenSim
- Documentación de procedimientos para comunicarse con los objetos de OpenSim
- Documentación procedimientos para los casos de estudios a desarrollar
- Desarrollo de dos casos de estudio, donde se ponga en práctica los pasos para lograr la comunicación con los objetos de OpenSim

- Evaluación de los resultados.

Métodos de investigación:

Métodos del nivel teórico:

- Método análisis histórico lógico se empleó para realizar un estudio valorativo de la evolución de los diferentes lenguajes, metodologías y software posibles a utilizar en el desarrollo de la herramienta y de sus funcionalidades.
- La inducción-deducción es utilizada en la investigación para sintetizar información de interés relacionada con el trabajo que se presenta.
- El analítico–sintético se utilizó al analizar toda la información relacionada con el tema de tesis, para extraer los elementos más importantes de cada documento consultado.

Métodos del nivel empírico:

- Análisis de todas las fuentes de información relacionadas con el tema.

Resultado esperado:

- Pasos para lograr la comunicación con los objetos de OpenSim
- Dos casos de estudio donde se ponga en práctica los pasos para lograr la comunicación con los objetos OpenSim.

Estructura capitular:

El contenido de este documento está estructurado en 3 capítulos de la forma siguiente:

Capítulo1. Fundamentación teórica.

Mediante un estudio del arte, se introducen conceptos básicos asociados a la realidad virtual (RV), OpenSim, y aspectos relacionados con la problemática planteada en la introducción del trabajo. Se caracterizan las principales herramientas, metodologías y lenguajes usados para el desarrollo de comunicar los objetos de OpenSim con el exterior de forma general, además de realizar un estudio de las diferentes técnicas, módulos y herramientas que permiten la comunicación con OpenSim. También se realiza un estudio sobre la biblioteca XML-RPC la cual será utilizada para darle solución al problema, pretendiendo dejar sentadas las bases teóricas.

Capítulo2. Solución propuesta.

Se muestra la propuesta de solución para el problema planteado y se describe todo el funcionamiento para los dos casos de estudio que se proponen, se realiza un resumen de la fase Análisis y Diseño para la implementación de los casos de estudio. Se definen los requisitos, tanto funcionales como no funcionales que deberán cumplir los casos de estudio. Se presentarán los modelos conceptuales, dígase diagramas y descripción de las clases, realización de casos de uso.

Captulo3.Resultados y discusión.

Se realizan los modelos de pruebas para cada caso de uso. Se proponen los pasos para realizar una comunicación general con los objetos de OpenSim y para los casos de estudio realizados.

Capítulo 1. Fundamentación teórica

1.1 Introducción

Mediante un estudio del arte, se introducen conceptos básicos asociados a la realidad virtual (RV), OpenSim, y aspectos relacionados con la problemática planteada en la introducción del trabajo. Se caracterizan las principales herramientas, metodologías y lenguajes usados para el desarrollo de comunicar los objetos de OpenSim con el exterior de forma general, además de realizar un estudio de las diferentes técnicas, módulos y herramientas que permiten la comunicación con OpenSim.

1.2 Realidad Virtual

Son muchas las compañías que desde la década de los 70 han usado la realidad virtual (RV) para diversos fines. A partir de este momento comienza la verdadera carrera comercial de la RV y cientos de productos empiezan a invadir la vida de las personas, llegando hasta el día de hoy en el que la tecnología sigue avanzando.

Definir RV, es un tanto difícil. Existen posiblemente tantas definiciones como investigadores haya, pues su reciente y rápida evolución no ha permitido establecer una definición clara. De este modo, no resulta extraño que la RV resulte ser relativa para diferentes personas y en diferentes situaciones. A continuación son citadas algunas definiciones:

"Medio que proporciona una visualización participativa en tres dimensiones y la simulación de mundos virtuales, siendo dichos mundos el elemento fundamental de un sistema de realidad virtual" [1].

"La realidad virtual es un entorno generado por computador en el que los participantes pueden entrar físicamente e interactuar con él, desplazándose por su interior o modificándolo de cualquier manera" [2].

La RV tiene diversas características, dentro de ellas las principales son:

- **Simulación:** capacidad de representar un sistema.
- **Interacción:** capacidad de tener el control de los sistemas creados.
- **Percepción:** es el factor más importante. Algunos sistemas de realidad virtual se dirigirán principalmente a los sentidos (visual, auditivo, táctil) por medio de elementos externos (Cascos de Visualización, Guantes de Datos, Cabinas, etc.), otros tratarán de llegar directamente al cerebro, evitando así las interfaces sensoriales externas, y otros, los más simples, recurrirán a toda la fuerza de la imaginación del hombre para experimentar una realidad virtual parcial [2].

1.2.1 Aplicaciones de la realidad virtual

Las aplicaciones de la RV que existen en la actualidad sobre actividades de la vida cotidiana son diversas: permite conocer elementos que de otro modo no estarían al alcance de las personas: recorrer las venas y arterias del cuerpo humano como en micro submarino, realizar cirugías, recreación de edificios y sitios de interés histórico o artístico, diseño de productos y maquinaria, simuladores aéreos, terrestres y marinos. Aquí es posible mezclar los conocimientos adquiridos con la fantasía y poner a prueba hipótesis sin el riesgo de destruir objetos o entornos delicados.

1.3 Metaversos

Los metaversos son entornos donde los humanos interactúan sociales y económicamente en un ciberespacio que es una abstracción del mundo real, [3]…Un punto importante es el relativo a la accesibilidad de los metaversos, para conseguir que las personas que tengan alguna minusvalía puedan interactuar libremente en ellos.

El Metaverso se compone de diferentes elementos, los básicos son los siguientes:

- **Avatar**: representación del usuario en el mundo virtual.
- **Regiones**: las regiones son pedazos de tierra delimitados por una determinada extensión.
- **Inventario**: característica esencial dentro de este mundo virtual, permite guardar los objetos.

1.4 OpenSim

Linden Lab (Propietaria de Second Life[1]) a principios de 2007, decidió liberar el código fuente de su cliente (visor) bajo la licencia GPL[2] para que fuera modificado y mejorado por la comunidad de desarrolladores independientes de Second Life. El mismo año nace el proyecto OpenSim, con la propuesta de crear un servidor de aplicaciones 3D, analizando la estructura del cliente de Second Life (ingeniería inversa)[3] eso permitió desarrollar OpenSim [6]. La llegada de OpenSim representó el sueño prometido, un ambiente de

[1] En 1999, se concibió Linden Lab, empresa que desarrolló un programa que permitía a sus usuarios la inmersión en un mundo virtual. Ya en el año 2003 nace Second Life, metaverso en el cual sus usuarios, conocidos como residentes, pueden explorar el mundo virtual e interactuar con otros residentes.

[2] Licencia Pública General, fue desarrollada por la FSF o Free Software Foundation y está diseñada específicamente para asegurar la cooperación con la comunidad en el caso de software que corra en servidores de red.

[3] Proceso de construir especificaciones de un mayor nivel de abstracción partiendo del código fuente de un sistema software o cualquier otro producto.

código abierto libre y escalable, por primera vez podrían desarrollar contenidos y actividades sin tener que pagar por terrenos ni por cada textura o animación que se sube al mundo las universidades.

Existen varias definiciones de OpenSim, a continuación se citan algunas de ellas:

"OpenSim es un servidor de aplicaciones 3D, presenta un entorno interactivo e inmersivo, convirtiéndose en un ambiente representativo de la realidad, se encuentra bajo licencia BSB..." [7].

"OpenSim es un servidor 3D de código abierto que permite crear ambientes virtuales (mundos virtuales) que pueden ser accedidos a través de una gran variedad de visores (clientes) o protocolos (software y web). OpenSim es configurable para suplir sus necesidades y puede ser extendido usando módulos" [8].

1.4.1 Actividades en OpenSim

En este metaverso se pueden realizar diversas actividades educativas, como son clases online, conferencias, talleres, capacitaciones entre usuarios, y proyectos colaborativos entre colegios.

Otras de las actividades comunes son:

- Se observa y se identifica dónde el usuario permanece
- Se puede movilizar los avatares y se puede visualizar las opciones del cliente Hippo Viewer[4]
- Se establece contactos con otros usuarios
- Se puede configurar la apariencia de los avatares
- Se conoce los diversos espacios para teletrasnportarse
- Se construyen objetos
- Se solicita un lugar para habitar y experimentar.

1.4.2 OpenSim en la educación

OpenSim cuenta con todas las bondades de un mundo virtual aplicado a la educación, pero su mayor potencial se encuentra en que las universidades pueden personalizar sus desarrollos, integrar usuarios existentes en sus LMS (Learning Management System) u otros sistemas con la base de datos de OpenSim, diseñar sistemas de administración y creación de contenidos que se adapten a las necesidades y a la metodología pedagógica de la universidad. Implementar OpenSim en las universidades indudablemente trae grandes ventajas al igual que implicaciones que son muy importantes tener en cuenta, en el modelo de implementación de mundos virtuales en la educación superior se mencionan los

[4] Cliente que permite acceder a mundos virtuales de simulación, como OpenSim o Second Life.

aspectos más importantes a la hora de diseñar un proyecto de educación, a continuación se describe cada uno de ellos: (6)

- **Infraestructura:** es muy importe que OpenSim corra en un servidor dedicado y no en uno virtualizado.
- **Administración:** es muy recomendable destinar a una persona dedicada a la administración de OpenSim.
- **Formación:** especialmente las personas que no tienen experiencias en videojuegos u otros mundos virtuales se hace necesario realizar una introducción en el uso básico.

OpenSim es una plataforma libre para la creación de mundos virtuales. Si lo comparamos con Second Life tiene unas ventajas muy claras para las empresas. IBM5 está apoyando con mucha intensidad a la comunidad de OpenSim. Un interesante artículo [9] describe la fuerza con la que IBM está apoyando a OpenSim y los usos que hace de esta plataforma. Prueba de que ambos mundos están muy próximos es el hecho de que IBM y Linden Labs han conseguido tele transportar avatares desde Second Life a un servidor de OpenSim.

El uso de OpenSim brinda funcionalidades a los usuarios para el manipulado y renderizado de los objetos que se construyan, lo que posibilita que todo quede lo más realista posible.

1.5 Biblioteca XML RPC

Se trata de una especificación y una serie de implementaciones que permiten al software que se ejecuta en sistemas operativos diferentes, correr en diferentes ambientes para hacer llamadas de procedimiento a través de internet.

Es llamada a procedimiento remoto a través de HTTP como el transporte y XML como la codificación. XML-RPC está diseñado para ser tan simple como sea posible, permitiendo al mismo tiempo las estructuras de datos complejas para ser transmitida, procesada y devuelta [10].

[5]**International Business Machines** (**IBM**), conocida coloquialmente como el Gigante Azul, es una empresa multinacional estadounidense que fabrica y comercializa herramientas, programas y servicios relacionados con la informática. IBM tiene su sede en Armonk (Nueva York, Estados Unidos) y está constituida desde el 15 de junio de 1911, pero lleva operando desde 1888.

1.5.1 XML RPC en OpenSim

XML RPC también se usa para realizar comunicación entre los objetos internos de OpenSim y el mundo externo lo que permite enviar mensajes a un objeto dentro de una región, siempre y cuando este se haya registrado para recibir dicha comunicación [11].

1.6 Módulos, técnicas y componentes que facilitan la comunicación de aplicaciones con OpenSim

- **SynEtTic**

SynEtTic es una plataforma multimedia de colaboración que utiliza herramientas en tiempo real. El proyecto completo incluye un mundo virtual, y videoconferencia. El espacio inmersivo utiliza tecnología OpenSim y servirá fundamentalmente para formación a distancia, eventos, reuniones de trabajo, desarrollo de habilidades profesionales por medio de simulaciones y visualización de datos [12].

- **DOT-AULA**

DOT está desarrollando sobre OpenSim el proyecto DOT-AULA, mundo virtual que permite a las personas y grupos en distintos lugares que trabajen juntos de manera más eficiente y natural, de forma que las empresas e instituciones se puedan beneficiar de los ahorros de costes que suponen los espacios virtuales [12].

- **Sloodle: LMS 3D**

Sloodle es un entorno de aprendizaje dinámico que une un entorno virtual 3D, OpenSim, a un LMS de código abierto, Moodle. El objetivo es describir las debilidades de los LMSs tradicionales y explorar las características de un entorno virtual 3D de aprendizaje, Sloodle, como una posible solución [13].

- **LMS**

LMS tiene funciones necesarias para el aprendizaje que no son posibles con ninguna de las plataformas por separado. Kemp[6] y Livingstone investigaron las necesidades y deseos sobre el uso de estos dos

[6] El trabajo del Dr. Kemp explora la intersección de espacios de aprendizaje digitales, físicos, las herramientas emergentes y las técnicas de apoyo a la innovación dentro de ellos. Más específicamente, sus intereses siguen los profesores, estudiantes y profesionales de la información, ya que identificar, utilizar y evaluar las tecnologías de la comunicación para la educación. Poco antes de unirse a la San José State University School of Library y la Facultad de Ciencias de la Información en 2006, fue co-fundador del proyecto Sloodle (Object Oriented Distancia Second Life Learning Environment) con el Dr. Daniel Livingstone de la Universidad de Oeste de

sistemas juntos mediante la realización de una encuesta a los educadores y encontraron este aprovechamiento integral útil [13].

- **Sloodle**

Los entornos virtuales proporcionan una nueva gama de oportunidades educativas. La naturaleza de estos entornos es generativa, permitiendo a los usuarios no sólo navegar e interactuar con un entorno 3D preexistente, sino también ampliar ese entorno. OpenSim es uno de estos entornos virtuales, los usuarios están incorporados con sus avatares [13].

- **Web-Intercom**

Posibilita conversaciones escritas por chat entre OpenSim y una sala de chat de Moodle. Esto tiene dos funciones clave. En primer lugar, proporciona acceso a una discusión en OpenSim para los usuarios que por alguna razón no puede acceder al propio OpenSim. En segundo lugar, permite el uso de una base de datos de Moodle para grabar y archivar las discusiones, y guardar este archivo en un entorno seguro protegido por contraseña [13].

- **Registration Booth**

Vincula los avatares de los estudiantes a sus cuentas de usuario Moodle, con el fin de hacer trabajar los componentes Sloodle.

- **Quiz Sloodle y Drop Box**

Es la sigla de evaluación en la lista OpenSim y del libro de calificaciones en Moodle establece cuestionarios o tareas de modelado en un atractivo entorno 3D [13].

- **Barra de herramientas multifuncional**

Esta herramienta mejora la interfaz de usuario de OpenSim.

- **Presenter Sloodle**

Permite hacer presentaciones en OpenSim de diapositivas y páginas web en Moodle. Presentar las diapositivas o páginas en OpenSim sin procesos largos para convertir o subir imágenes [13].

- **Quiz Chair**

Escocia. Este proyecto de becas ampliado el aprendizaje Moodle sistema de gestión con los avatares y construidos por el usuario entornos virtuales

Permite a los estudiantes intentar responder un cuestionario de opción múltiple dentro de OpenSim, con las respuestas que se almacene en Moodle.

Después de haber realizado un estudio de las diferentes técnicas, módulos, herramientas y biblioteca que permiten la comunicación con OpenSim de diversas formas, se decide utilizar la biblioteca XML RPC pues es la única que cumple con las necesidades actuales para el presente trabajo de diploma, ya que realiza envío de datos a los objetos de OpenSim desde el exterior. No se utilizarán las técnicas, módulos o componentes debido a que algunos de ellos como es el Soodle, están desarrollados con la biblioteca antes mencionada para cumplir otros propósitos como es unir aplicaciones con el mundo virtual, por lo que no cumplen con el objetivo del presente trabajo a pesar de que todos ellos logran una comunicación con OpenSim

1.7 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar automatizaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Está formado por un conjunto de símbolos y reglas y semánticas que definen su estructura y el significado de sus elementos y expresiones. En este epígrafe se describen los principales lenguajes, para en el siguiente capítulo escoger el adecuado para el desarrollo de la presente investigación.

1.7.1 Orientado a objeto

- **C#**

El lenguaje de programación C# fue creado por el danés Anders Hejlsberg que diseñó también los lenguajes Turbo Pascal y Delphi. Con este nuevo lenguaje se quiso mejorar con respecto de los dos lenguajes anteriores de los que deriva el C, y el C++ [14].

- **Java**

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principio de los años 90 [15]… Algunas de las principales características de este lenguaje son: es distribuido, interpretado y compilado a la vez, robusto, indiferente a la arquitectura, portable, de alto rendimiento y dinámico.

1.7.2 Lenguaje de programación Script

Un lenguaje interpretado es un lenguaje de programación que está diseñado para ser ejecutado por medio de un intérprete, en contraste con los lenguajes compilados. Teóricamente, cualquier lenguaje puede ser compilado o ser interpretado, así que esta designación es aplicada puramente debido a la práctica de implementación común y no a alguna característica subyacente de un lenguaje en particular. A ciertos lenguajes interpretados también se les conoce como lenguajes de script[16].

- **Script de OpenSim (LSL)**

El Linden Scripting Language (LSL) es el lenguaje que se utiliza para crear contenido interactivo en Second Life y OpenSim. Es un lenguaje utilizado para fijar los comportamientos de los objetos. Se ajusta a la sintaxis de los lenguajes C y Java, se maneja mediante eventos, utiliza variables tipo 3D, así como una variedad de funciones incorporadas para manipular la física y la interacción de los avatares [17]

1.8 Lenguajes y herramientas de modelado

El progreso de software ha tenido un gran avance en los últimos tiempos y las herramientas de modelado forman un componente muy significativo en el entorno de desarrollo, puesto que son esenciales para el análisis de sistemas. Las herramientas mejoran la forma en que ocurre el desarrollo y tiene influencia sobre la calidad del resultado final.

En la actualidad se han creado una serie de herramientas con este fin, las cuales han sido mejoradas por parte de sus desarrolladores con el fin de encontrar en ellas fiabilidad, eficiencia, entre estas se encuentra: visual Paradigm y Rational Rose Enterprise Edition, dichas herramientas usan el Lenguaje de Modelado (UML) para la especificación, la visualización, la construcción y la documentación de los artefactos de los sistemas de software y también para otros tipos de sistemas.

1.8.1 Lenguaje Unificado de Modelado 2.0 (UML)

UML (por sus siglas, del inglés Unified Modeling Language) es el lenguaje estándar de modelado para software. Es un lenguaje gráfico que visualiza, especifica, construye y documenta los artefactos del sistema. UML suministra mecanismos de extensibilidad, los cuales permiten a sus usuarios clarificar su sintaxis y su semántica. UML puede tanto ajustarse a un sistema, proyecto o proceso de desarrollo específico si es necesario [18].

UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

1.8.2 Herramientas CASE (Computer Assisted *Software* Engineering)

La herramientas CASE brindan una gran gama de componentes que incluyen todos o la mayoría de los requisitos necesarios para el desarrollo de los sistemas, además son creadas con una gran exactitud en torno a las necesidades de los desarrolladores de software para la automatización de procesos incluyendo el análisis, diseño e implementación. Ofrecen una gran plataforma de seguridad a sistemas que las usan, a continuación se describen algunas de ellas, como es Visual Paradigm for UML Enterprise Edition 8.0 y Rational Rose Enterprise Edition.

- **Visual Paradigm for UML Enterprise Edition 7.0**

Visual Paradigm es una herramienta CASE que usa UML como lenguaje de modelado, permite la generación automática de reportes en formato pdf y html, el reconocimiento de artefactos de ingeniería a partir de reconocimiento de textos formales o informales, implementa una actualización automática del modelo de diseño y código permitiendo mantener la documentación de ambos modelos actualizadas con los cambios que ocurran en ambos sentidos, optimizando la descripción textual de elementos de código a partir de la descripción visual. Es capaz de importar y exportar elementos de otras herramientas CASE como Rational Rose y presenta una interfaz de uso intuitiva y con increíbles facilidades a la hora de modelar los diagramas que soportan la ingeniería de requerimientos [19].

- **Rational Rose Enterprise Edition**

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML. Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software [20].

1.9 Metodologías de desarrollo de software

Las metodologías de desarrollo de software de forma general son usadas para planificar, estructurar y controlar el proceso de desarrollo de software. Se han definido un gran número de éstas en función de las dimensiones de los proyectos en lo que se aplicarán y otras muchas variables que han conllevado a un

desarrollo constante y fructífero, actualmente existen muchas como Scrum o Modelo de Diagnóstico Común (Common Diagnostic Model, CDM), entre las más utilizadas se encuentran las que se describen en los siguientes epígrafes.

Metodologías pesadas

- **Rational Unified Process (RUP)**

El Proceso Unificado de Desarrollo (RUP) es un proceso de desarrollo de software y a la vez, es un conjunto de actividades dirigidas a transformar los requerimientos del cliente en un sistema de software. Aumenta la productividad del equipo de desarrollo, permitiendo a cada miembro compartir un lenguaje común (UML[7]), un proceso y una vista de cómo revelar el software. RUP es un proceso configurable y es soportado por herramientas que automatizan partes grandes del proceso. Genera una amplia documentación de los elementos que la componen. En su modelación define como sus principales elementos a los trabajadores, actividades, artefactos y el flujo de actividades.

El ciclo de vida de RUP

La vida de un sistema transcurre a través de ciclos de desarrollo, desde que comienza hasta que termina, en cada ciclo se repite el proceso unificado de desarrollo. Cada uno consta de cuatro fases (inicio, elaboración, construcción, transición) y concluye con una versión del producto. Cada fase se subdivide en iteraciones. Una iteración es una secuencia de actividades con un plan establecido y criterios de evaluación, cuyo resultado es una versión del software.

El ciclo de vida de RUP se caracteriza por ser:

- Dirigido por casos de uso
- Centrado en la arquitectura
- Iterativo e Incremental [24].

Metodologías ágiles

- **Programación Extrema (XP)**

Se inició el 6 de marzo 1996. Programación Extrema es uno de los más populares procesos ágiles de desarrollo de software. Su mayor propósito es la satisfacción del cliente. XP faculta a los desarrolladores para responder con seguridad a las necesidades cambiantes de los clientes, incluso a finales del ciclo de

[7]Lenguaje Unificado de Modelado

vida [21]. Es una disciplina de desarrollo de software basado en los valores de la simplicidad, la comunicación y la retroalimentación. Su acción consiste en llevar todo el equipo junto con la presencia de prácticas sencillas, con suficiente información para que el equipo pueda ver dónde están y para ajustar las prácticas a su situación particular [22] Una de las características que destaca a esta metodología es que el volumen de información que describe es mucho menor que la generada por las metodologías pesadas [23].

- **Agile Unified Process (AUP)**

Agile Unified Process (AUP), es una versión simplificada del Proceso Unificado de Rational (RUP). Descrito en una forma simple, fácil de entender y brinda un enfoque de desarrollo de software utilizando técnicas ágiles y conceptos del RUP. El proceso unificado (Unified Process o UP) es un marco de desarrollo de software iterativo e incremental.

AUP abarca siete flujos de trabajos, cuatro ingenieriles y tres de apoyo: modelado, implementación, prueba, despliegue, gestión de configuración, gestión de proyectos y ambiente. El modelado agrupa los tres primeros flujos de RUP (modelamiento del negocio, requerimientos y análisis y diseño). Dispone de cuatros fases igual que RUP: creación, elaboración, construcción y transición.

AUP es una metodología de desarrollo ágil, la cual se basa en los siguientes principios:

El personal sabe lo que está haciendo. La gente no va a leer detallado el proceso de documentación, pero algunos quieren una orientación de alto nivel y/o formación de vez en cuando.

Simplicidad. Todo se describe concisamente utilizando un puñado de páginas, no miles de ellos.

Agilidad. El ajuste a los valores y principios de la alianza ágil.

Centrarse en actividades de alto valor. La atención se centra en las actividades esenciales para el de desarrollo, no todas las actividades que suceden forman parte del proyecto.

Herramienta de la independencia. Usted puede usar cualquier conjunto de herramientas que usted deseé con el ágil UP. Lo aconsejable es utilizar las herramientas más adecuadas para el trabajo, que a menudo son herramientas simples o incluso herramientas de código abierto [23].

1.10 Entorno integrado de desarrollo (IDE)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

En el presente epígrafe se describen los posibles IDE a utilizar en el desarrollo de la presente investigación.

- **NetBeans**

NetBeans es un reconocido entorno de desarrollo integrado, disponible para Windows, Mac, Linux y Solaris. El proyecto de NetBeans está formado por un IDE de código abierto y una plataforma de aplicación que permite a los desarrolladores crear con rapidez aplicaciones web, empresariales, de escritorio y móviles utilizando la plataforma Java, así como JavaFX, PHP, Java Script y Ajax, Ruby y C/C++ [25].

NetBeans es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE1 es un producto libre y gratuito sin restricciones de uso.

- **Visual Studio**

Visual Studio es un IDE que permite a los desarrolladores crear aplicaciones (Web, Escritorio), Aplicaciones para SmartPhone, Pocket PC, servicios web y otras utilidades. Este IDE tiene como característica que es multilenguaje, significa que soporta varios lenguajes de programación entre los que se encuentran C#.net, VisualBasic.net y Visual C++ [26].

1.11 Sistema Gestores de Bases de Datos

Un sistema de gestión de bases de datos (SGBD) es una herramienta que permite insertar, modificar y buscar eficazmente datos específicos en una gran masa de información. El usuario puede precisar estas investigaciones. Los resultados pueden clasificarse según criterios [27].

- **PostgreSQL**

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, con su código fuente disponible libremente sin costo, utiliza un modelo cliente/servidor y usa multiprocesos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Durante todo su desarrollo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema [28].

Es un servidor de base de datos objeto relacional libre, el cual está liberado bajo la licencia BSD, es decir, que puede ser utilizado, modificado y distribuido por todo el mundo, libre de cargo para cualquier propósito, sea comercial, privado, o académico.

- **MYSQL**

MySQL para Windows (versión de desarrollo) es un sistema de administración de una base de datos con soporte para múltiples usuarios, usa el lenguaje SQL estandarizado para el almacenamiento, actualización y acceso a información. Es muy rápido y capaz de almacenar grandes cantidades de datos. Soporta muchos lenguajes de programación distintos como: C, C++, Eiffel, Java, Perl, PHP, Python y TCL [29].

Consideraciones del capítulo

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 AUP facilita la creación de una documentación con los artefactos necesarios para la realización de los casos de estudio. Entre las herramientas estudiadas y teniendo en cuenta las peculiaridades de cada una, Visual Paradim, Netbeans y Postgres poseen las características necesarias para enfrentar el desarrollo de una solución al problema planteado. Se logró identificar que el uso de la biblioteca XML RPC diseñada para el envío de datos al entorno virtual, podría facilitar el cumplimiento de los objetivos de la presente investigación.

Capítulo 2. Solución propuesta

2.1 Introducción

En el presente capítulo se muestra la propuesta de solución para el problema planteado y se describe todo el funcionamiento para los dos casos de estudio que se proponen, se realiza un resumen de la fase Análisis y Diseño para la implementación de los casos de estudio. Se definirán los requisitos, tanto funcionales como no funcionales que deberán cumplir los casos de estudio. Se presentarán los modelos conceptuales, dígase diagramas, descripción de las clases y realización de casos de uso.

2.3 Propuesta de solución

Se propone como solución dos casos de estudios que muestran ejemplos de las posibles aplicaciones que pueden desarrollarse en la universidad sobre el metaverso OpenSim. Uno de los ejemplos que se propone es el control del consumo eléctrico que muestra una gráfica representando el gasto de energía de la universidad en 4 meses. Mediante una aplicación de escritorio el usuario podrá actualizar los datos del consumo eléctrico y los cambios se visualizarán en el mundo virtual. El otro ejemplo es sobre el control de los medios fijos tangibles de una computadora, la cual al ser actualizada mediante la aplicación de escritorio el usuario podrá observar sus datos en el mundo virtual al presionar clic encima de los medios que la componen. Para lograr la comunicación con los objetos de OpenSim, después de haber realizado un estudio de las diferentes técnicas de comunicación con este metaverso se decidió utilizar la biblioteca XML RPC que permite el envío de datos desde el exterior al interior del metaverso a través de un canal de comunicación que los objetos poseen. Es de suma importancia el uso de esta biblioteca pues permitirá el control virtual de los datos en varias aplicaciones de la universidad.

Dentro del mundo virtual no es necesario realizar la construcción de los objetos que se necesitarán para los casos de estudios pues ya forman parte de un inventario de OpenSim, lo que posibilita no realizar el renderizado y manipulado de los objetos.

Después de haber realizado un estudio sobre las metodologías pesadas y ágiles se seleccionó la metodología de desarrollo Agile Unified Process (AUP), esta metodología es muy semejante a RUP pero en una versión simplificada; sólo se utilizan los artefactos que son imprescindibles para la realización del producto, es seleccionada pues el objetivo de la investigación es generar un documento que permita guiar al usuario para lograr la comunicación con los objetos de OpenSim, además de poseer conocimientos ya que fue estudiada en clases desde el inicio . Se seleccionó el lenguaje de programación java ya que es un

lenguaje multiplataforma, la biblioteca XML RPC está desarrollada en java y es soportado por el IDE Netbeans, escogido por estar basado en código abierto y por estar escrito en java, también posee numerosas características que hacen que el IDE sea atractivo para cualquier desarrollador. Además se trabaja con el lenguaje de programación LSL (Script de OpenSim), que es el lenguaje interno del compilador del Engine que usa OpenSim para que los script de los objetos ejecuten funcionalidades implementadas por los usuarios. La herramienta CASE que se utilizará será el Visual Paradigm 8.0 por ser una herramienta multiplataforma que soporta el ciclo de vida completo de desarrollo de software, la UCI paga su licencia y además soporta el lenguaje de modelado UML el cual ha sido seleccionado para especificar y visualizar artefactos del sistema. Teniendo en cuenta las particulares de cada uno de los gestores de bases de datos, se llegó a la conclusión, que el más idóneo a utilizar en este trabajo de diploma es PostgreSQL pues es un potente motor de bases de datos y es libre, además de que la base de datos que maneja el consumo eléctrico de la uci está realizada sobre PostgreSQL, y ese es uno de los ejemplos que se propone como solución.

2.4 Especificación de los requisitos de software

Los requisitos son una especificación de lo que el sistema debe cumplir para satisfacer al cliente. Estos pueden dividirse en requisitos funcionales y requisitos no funcionales. En los próximos epígrafes se especifican los requisitos para cada uno de los ejemplos que se proponen.

2.4.1 Requisitos Funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir, para que las peticiones del cliente queden satisfechas. Para los ejemplos que se proponen el sistema debe cumplir:

Requisitos funcionales para los casos de estudio

N°	Nombre	Descripción
RF 1	Conectar con la Base de datos	El sistema debe mostrar una interfaz donde el usuario introduce los datos para realizar la conexión, cuando se introduce un dato incorrecto muestra un mensaje indicándolo, en caso contrario muestra un mensaje indicando el éxito de la operación.
RF 2	Mostrar información de la base	La aplicación debe mostrar los datos que representa la

	de datos.	tabla del consumo eléctrico en la base de datos.
RF 3	Actualizar datos de la gráfica.	Obtiene de la base de datos los datos de cada barra de la gráfica, y los envía mediante el canal de comunicación, si el usuario modifica un valor en la aplicación se muestra en OpenSim el nuevo valor.
RF 4	Enviar datos del consumo eléctrico a la gráfica de OpenSim.	El sistema se conecta a la base de datos, obtiene los datos y los envía a OpenSim mediante el canal de comunicación, la información se envía a cada objeto por separado.
RF 5	Actualizar computadora.	Obtiene de la base de datos los datos de cada medio que conforma la computadora y los envía mediante el canal de comunicación.
RF 6	Mostrar datos de los objetos de la computadora.	Muestra la información de los medios al darle clic encima mediante un mensaje.
RF 7	Enviar datos a los medios de la computadora en OpenSim.	Los datos se envían a OpenSim mediante el canal de comunicación.
RF 8	Cambiar estado de los medios de la computadora.	Cambia el color de los nombres de los medios de la computadora mientras les va llegando la información.

Tabla 1: Requisitos de los casos de estudio

Caso de estudio del control de medios

2.4.2 Requisitos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, además son aspectos importantes que el producto debe cumplir para lograr un producto atractivo, usable, rápido o confiable. A continuación se enuncian, separados en categorías, los diferentes requisitos no funcionales que el componente debe satisfacer.

- **RNF 1. Usabilidad.**

Utilización de sistema. El sistema solo será utilizado por aquellas personas que tengan relación con la información que se maneja y que tengan los conocimientos básicos para el manejo de OpenSim, además

contará con un manual de usuario, con instrucciones de tipo paso a paso, para entender el trabajo del sistema.

- **RNF 2. Portabilidad.**

Ejecución de la herramienta. La herramienta permitirá ser compilada y ejecutada solo sobre la plataforma Windows.

- **RNF 3. Restricciones en el diseño y la implementación.**

Lenguajes de programación. Se utilizará el lenguaje de programación java y el lenguaje interno de OpenSim LSL.

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.

Implementación del sistema. Se regirá por la filosofía de programación orientada a objetos (POO) y la programación estructurada (LSL script de OpenSim) para los objetos de OpenSim.

- **RNF 4. Hardware.**

Instalación de la herramienta. Para la instalación del servidor de OpenSim la computadora deberá cumplir con los siguientes requisitos:

- ✓ Memoria RAM de 1 Gb o Superior.
- ✓ Velocidad de Microprocesador a 2.0 GHz o superior.
- ✓ Recursos de Video de 256 Mb o superior.

- **RNF 5. Interfaz o apariencia externa.**

Interfaz de Usuario. La interfaz será sencilla para facilitar el uso de la misma, lo cual se evidencia en lo siguiente: las opciones de servicio tendrá el texto que muestre la opción en cuestión. Constará con una tabla que muestra los datos del consumo eléctrico de la base de datos, contará además con un campo textbox en el cual el usuario introduce el canal de comunicación.

2.5 Modelo del dominio

El modelo de dominio permite comprender el funcionamiento del negocio a través de la modelación de los procesos identificados, conceptos y sus relaciones. Mediante el modelo conceptual que se muestra a

continuación se explica los conceptos significativos del negocio con el objetivo de lograr un conocimiento básico del vocabulario y de los conceptos que se incluyen en los requerimientos.

- **Usuario:** Persona que interactúa con el sistema para la realización de los casos de uso.
- **Actualizador_Datos:** Aplicación para actualizar la computadora y la gráfica de consumo eléctrico.
- **Objetos_Secundarios:** Objetos de la gráfica y la computadora.
- **Objeto_Receptor:** Objeto que se encarga de enviar la información a los objetos secundarios a través del canal de comunicación.
- **Base_Datos:** Base de datos que almacena la información del consumo eléctrico y de los medios de la computadora.

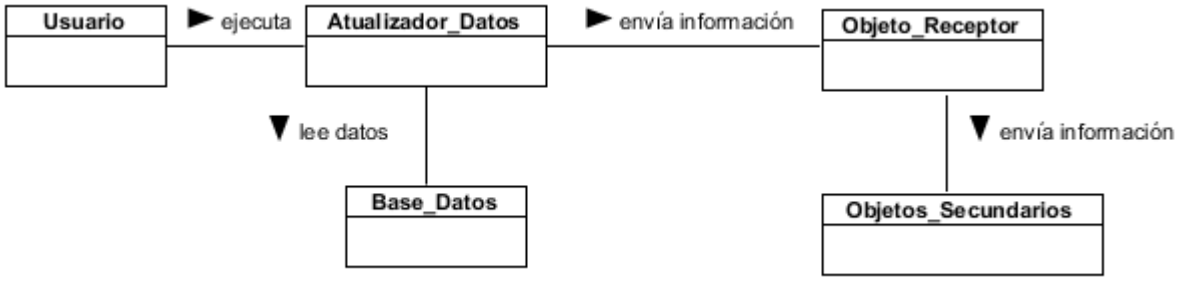


Figura 1: Modelo del Dominio

2.6 Definición de los casos de uso

Una vez recopilados los requisitos se pueden definir los casos de uso los cuales facilitarán una descripción de cómo el sistema se usará. El caso de uso describe la manera en que los actores interactúan con el sistema [18].

2.6.1 Definición de los actores

Un actor representa papeles que las personas (o dispositivos) juegan como impulsores del sistema. Los actores que interactuarán se describen a continuación.

Actores	Justificación
Usuario	Es la persona que interactúa con el sistema, la cual podrá conectarse a la base de datos, obtener el canal de comunicación, actualizar los datos del consumo eléctrico y los datos del control de medios.

Tabla 2: Descripción de los actores que interactúan con el sistema

Casos de uso (CU) para los casos de estudio

CU 1: Conectar con la Base de Datos.

CU 2: Actualizar gráfica.

CU 3: Actualizar computadora.

2.6.2 Diagrama de Casos de Uso del Sistema

Un diagrama de casos de uso del sistema describe parte del modelo de casos de uso y muestra un conjunto de casos de uso y actores con una asociación entre cada par actor/caso de uso que interactúan [30].

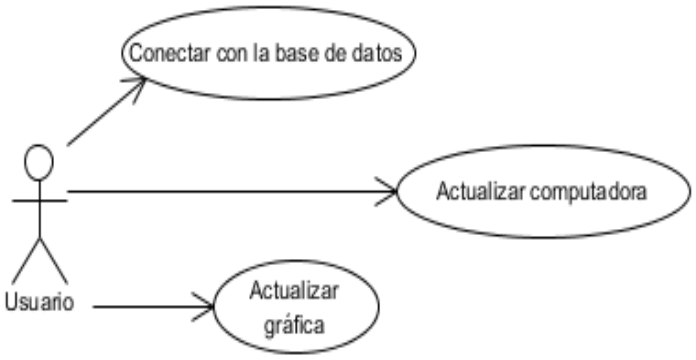


Figura 2: Diagrama de casos de uso del sistema.

2.6.3 Descripción textual de los Casos de Uso del Sistema

Un caso de uso es una unidad coherente de funcionalidad, externamente visible, proporcionada por una unidad del sistema y expresada por secuencias de mensajes intercambiados por la unidad del sistema y uno o más actores. El propósito de un caso de uso es definir una pieza de comportamiento coherente, sin revelar la estructura interna del sistema. Desde el punto de vista de los usuarios, éstas pueden ser situaciones anormales. Desde el punto de vista de los sistemas, son las variaciones adicionales que deben ser descritas y manejadas [31].

Caso de uso	
CU-1	Conectar con la base de datos.

Propósito	El objetivo del caso de uso es que el usuario pueda conectarse con la base de datos para obtener información tanto de la gráfica del consumo eléctrico como de los medios de la computadora.
Prioridad	Crítico.
Actores: Usuario	
Resumen: El usuario inicializa el caso de uso cuando ejecuta la aplicación para conectarse a la base de datos, el mismo procede a introducir los datos como son nombre del servidor, puerto, nombre de la BD, usuario y contraseña. La aplicación se conecta a la base de datos en caso de introducir todos los datos correctos, sino muestra mensajes de error, también muestra un mensaje cuando la conexión es correcta.	
Referencias	RF 1
Precondiciones	Existencia de la base de datos a conectar.

Curso Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El actor ejecuta la aplicación para conectarse a la base de datos. 2. El actor introduce los datos para la conexión y presiona la opción conectar	1.1 El sistema muestra la interfaz de conexión a la base de datos con los siguientes campos: Nombre del servidor. Puerto. Nombre de la base de datos. Usuario. Contraseña. 2.1 El sistema verifica la existencia de la base de datos y que todos los datos sean correctos. 2.2 El sistema realiza la conexión. 2.3Finaliza el caso de uso.
Curso Alterno de Eventos	
Acción del actor	**Respuesta del sistema**

1. En el paso 2 del flujo normal de eventos el actor introduce datos incorrectos.	1.1 El sistema muestra un mensaje de error indicando: *"Datos incorrectos".* 1.2 El sistema muestra nuevamente la interfaz: Conectar a la base de datos.

Tabla 3: Descripción del caso de uso Conectar con la base de datos

Caso de uso	
CU-3	Actualizar gráfica.
Propósito	El objetivo del caso de uso es que el usuario pueda actualizar los valores de la gráfica y en OpenSim se muestren los cambios realizados.
Prioridad	Medio.
Actores: Usuario.	
Resumen: El usuario inicializa el caso de uso después de haber logrado la conexión con la base de datos, posteriormente obtiene el canal de comunicación en OpenSim y lo introduce en la aplicación "Actualizador de datos", el caso de uso permite modificar los valores de la gráfica en la aplicación y mostrar los cambios realizados en OpenSim.	
Referencias	RF 1, RF 2, RF 3, RF 5.
Precondiciones	Haber realizado la conexión con la base de datos. Haber obtenido el canal de comunicación.
Curso Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El actor introduce el canal de comunicación en el campo: Canal de Comunicación, modifica los campos deseados de la tabla referente al consumo eléctrico y presiona la opción Actualizar Gráfica.	1.1 El sistema verifica que el campo del canal de comunicación no esté vacío. 1.2 El sistema envía los datos a OpenSim y actualiza la gráfica. 1.3 Finaliza el caso de uso.

Curso Alterno de Eventos	
Acción del actor	**Respuesta del sistema**
1. En el paso 1 del flujo normal de eventos el actor no introduce un canal de comunicación y presiona el botón Actualizar Gráfica.	1.1 El sistema muestra un mensaje de error indicando:*"Debe introducir el canal de comunicación "* 1.2 El sistema muestra nuevamente la interfaz Actualizador de datos para introducir el canal de comunicación.

Tabla 4: Descripción del caso de uso Actualizar gráfica

Caso de uso	
CU-4	Actualizar computadora.
Propósito	El objetivo del caso de uso es mostrar los datos de los medios de la computadora y cambiarle el color a los nombres de los medios informando así cada vez que le llegue la información a cada uno.
Prioridad	Alto.
Actores: Usuario.	
Resumen: El usuario inicializa el caso de uso después de haber logrado la conexión con la base de datos, posteriormente obtiene el canal de comunicación en OpenSim y lo introduce en la aplicación "Actualizador de datos", el caso de uso permite mostrar los datos de los medios de la computadora al darles clic encima, y además de cambiar de color su nombre a la medida que la información les llegue.	
Referencias	RF 1, RF 2, RF 5, RF 6, RF 7.
Precondiciones	Haber realizado la conexión con la base de datos. Haber obtenido el canal de comunicación.

Curso Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El actor introduce el canal de comunicación en el campo: Canal de Comunicación y presiona la opción Actualizar Gráfica.	1.1 El sistema verifica que el campo del canal de comunicación no esté vacío. 1.2 El sistema envía los datos a OpenSim y actualiza los medios de la computadora. 1.3 Finaliza el caso de uso.

Curso Alterno de Eventos	
Acción del actor	**Respuesta del sistema**
1. En el paso 1 del flujo normal de eventos el actor no introduce un canal de comunicación y presiona el botón Actualizar Computadora.	1.1 El sistema muestra un mensaje de error indicando: *"Debe introducir el canal de comunicación"* 1.2 El sistema muestra nuevamente la interfaz Actualizador de datos para introducir el canal de comunicación.

Tabla 5: Descripción del caso de uso Actualizar computadora

2.7 Análisis y diseño del sistema

El análisis proporciona una visión general del sistema que puede ser más difícil de obtener mediante el estudio de los resultados del diseño y la implementación debido a que contienen demasiados detalles. El modelo de diseño se crea tomando el modelo de análisis como punto de partida. En el presente trabajo solamente se realiza el modelo de diseño, pues la metodología AUP es una versión simplificada de RUP que solamente utiliza los artefactos que son imprescindibles y realmente necesarios para la realización del producto.

2.7.1 Vista lógica

Con el propósito de organizar los distintos elementos que toman acción dentro del sistema se ha designado una variedad de paquetes en donde se agrupan dichos elementos. Cada paquete está compuesto por clases que tienen responsabilidades y atributos dentro de sí. En la siguiente figura se representa la organización del sistema en capas utilizando el patrón 3 capas que permite simplificar y comprender el desarrollo del sistema reduciendo la dependencia de forma que las capas más bajas no sean conscientes de ningún detalle de las superiores. En cada una de las capas se representan las clases arquitectónicamente significativas organizadas en paquetes.

Capa de presentación: Paquete interfaz de usuario: Contiene las clases arquitectónicamente significativas que representan las interfaces de usuario encargadas de enviar las peticiones del cliente a las clases que se encuentran en la capa de aplicación.

Capa de negocio: Paquete lógica del negocio: Contiene las clases y funcionalidades de control y ejecución de las peticiones que llegan desde la capa de interfaz de usuario. Sirve como intermediario o mediado entre la lógica de presentación y la lógica de negocio.

Capa de datos: Paquete acceso a datos: contiene la clase y funcionalidades de acceso a los datos es decir la información almacenada en base datos. Sirve como intermediario o mediado entre la lógica de datos y la lógica de negocio.

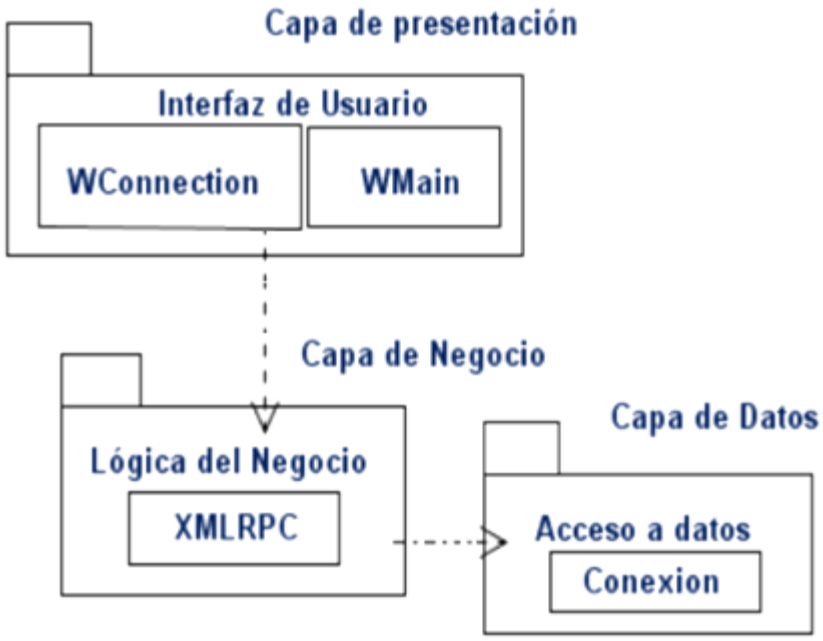


Figura 3: Vista Lógica del sistema

2.7.3 Diagramas de clases

Los diagramas de clases se utilizan para mostrar clases y sus relaciones, para saber las clases que participarán en la realización de un caso de uso y las responsabilidades que deberán cumplir. El diagrama de clases del diseño incluye más detalles que el diagrama de clases del modelo de análisis lo cual es necesario para la adaptación del modelo de diseño al entorno de la implementación. Las clases que intervienen en la realización de los casos de uso de los ejemplos implementados estarán estructuradas según el patrón tres capas, ya que permite simplificar y comprender el desarrollo del sistema reduciendo la dependencia de forma que las capas más bajas no sean conscientes de ningún detalle de las superiores.

2.7.3.1 Descripción de las clases

- **Conexión:** Clase encargada de realizar la conexión entre la aplicación y la base de datos.
- **WConnection:** Clase interfaz que permite introducir los datos para realizar la conexión con la base de datos.

- **XMLRPC:** Clase controladora que tiene la responsabilidad de enviar la información al Metaverso OpenSim. Contiene las funcionalidades que dan respuesta a los casos de uso del sistema acepto el caso de uso obtener canal de comunicación.
- **Wmain:** Clase interfaz que permite introducir el canal de comunicación, actualizar tanto la grafica como la computadora.
- **javax.swing.JFrame:** Se encarga de la gestión de los componentes visuales.
- **Cis69mc.jar:** Es la biblioteca que se usa para el envío de datos a OpenSim (XML RPC).
- **Postgresql-8.3-603.jdbc.jar:** Es la biblioteca que se usa para la conexión con la base de datos.

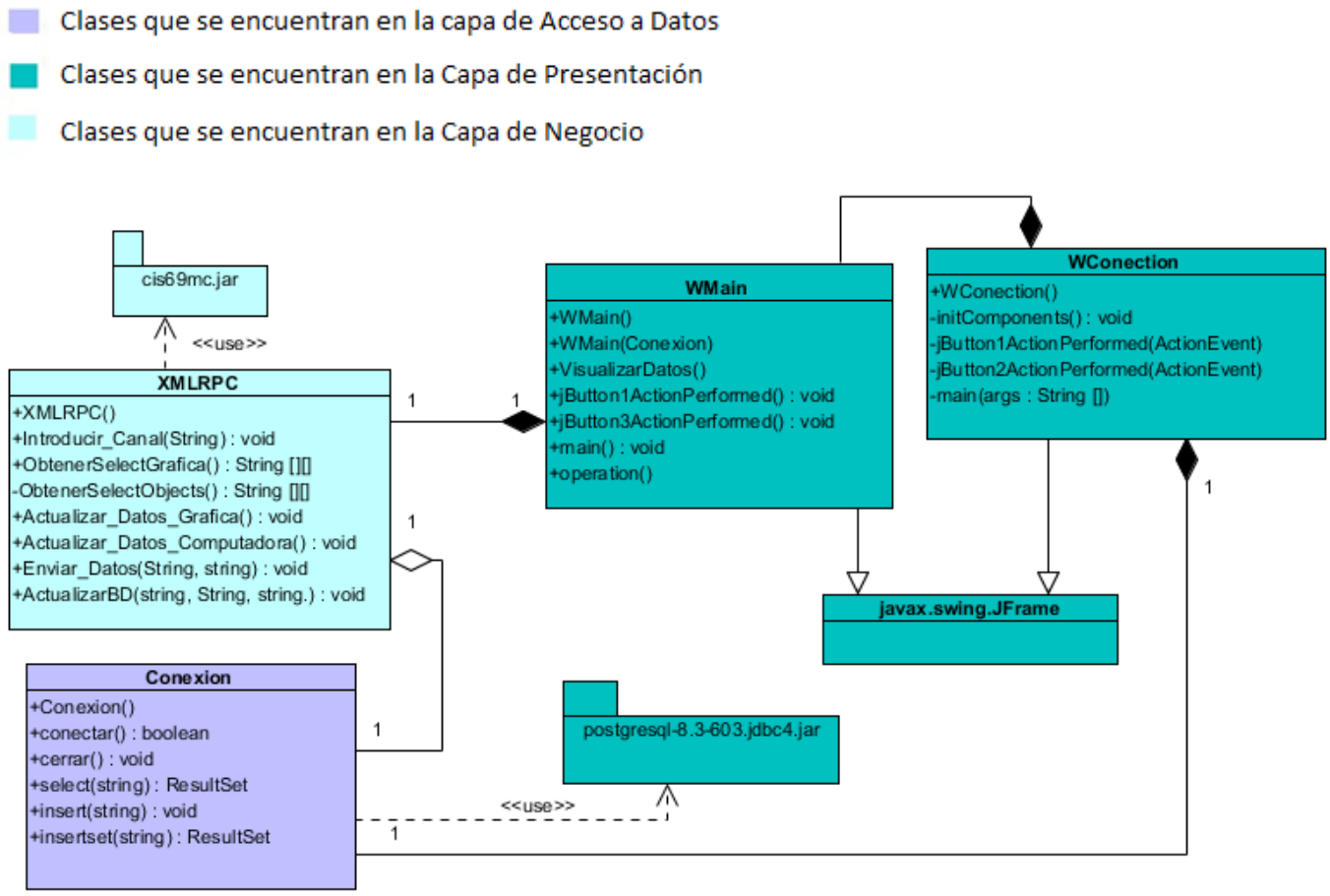


Figura 4: Diagrama de clases del diseño

2.7.4 Realización de casos de uso

La realización de los casos de uso se describe durante el análisis y el diseño a través de los diagramas de colaboración y los diagramas de secuencia. La realización de un caso de uso en el modelo de diseño describe como se realiza el caso de uso en términos de las clases de diseño correspondientes. Para modelar las interacciones entre los objetos del diseño se utiliza el diagrama de secuencia. En las siguientes figuras se representan los diagramas de secuencia de los casos de uso que fueron realizados durante el diseño.

2.7.4.1 Diagramas de secuencia

Diagrama de secuencia para el caso de uso conectar con la base de datos

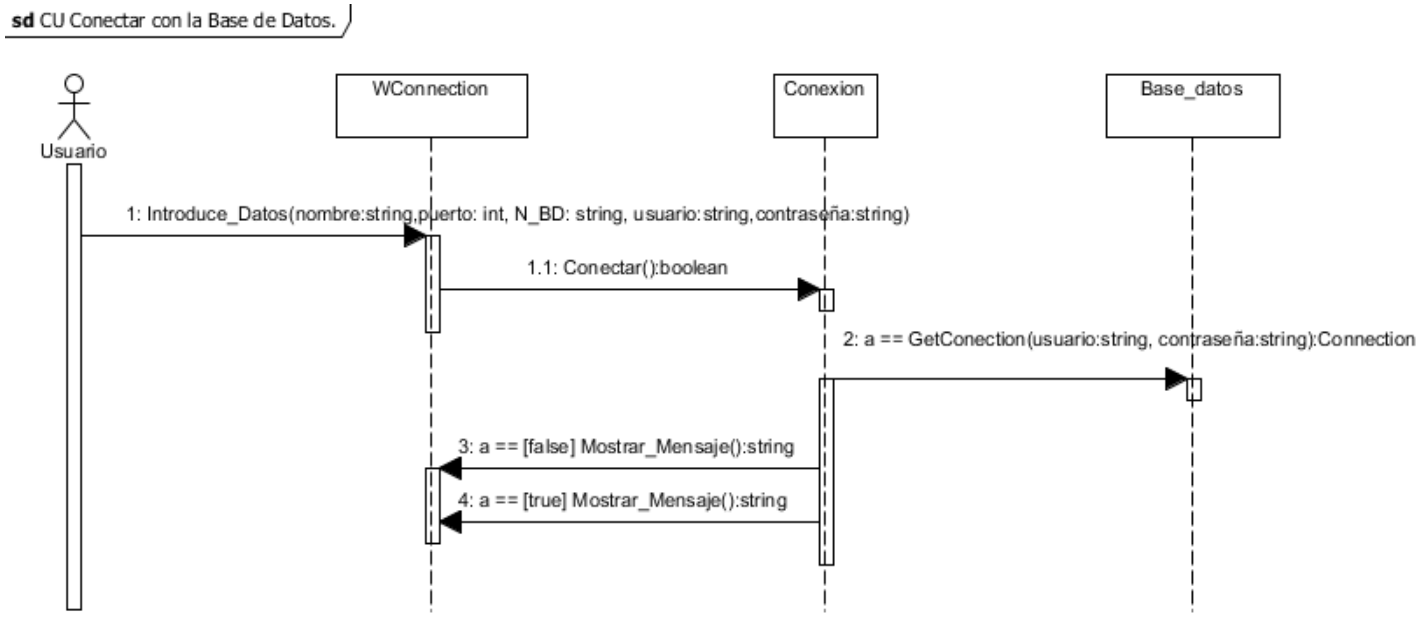


Figura 1: Diagrama de secuencia para el caso de uso conectar con la base de datos

Diagrama de secuencia para el caso de uso actualizar gráfica

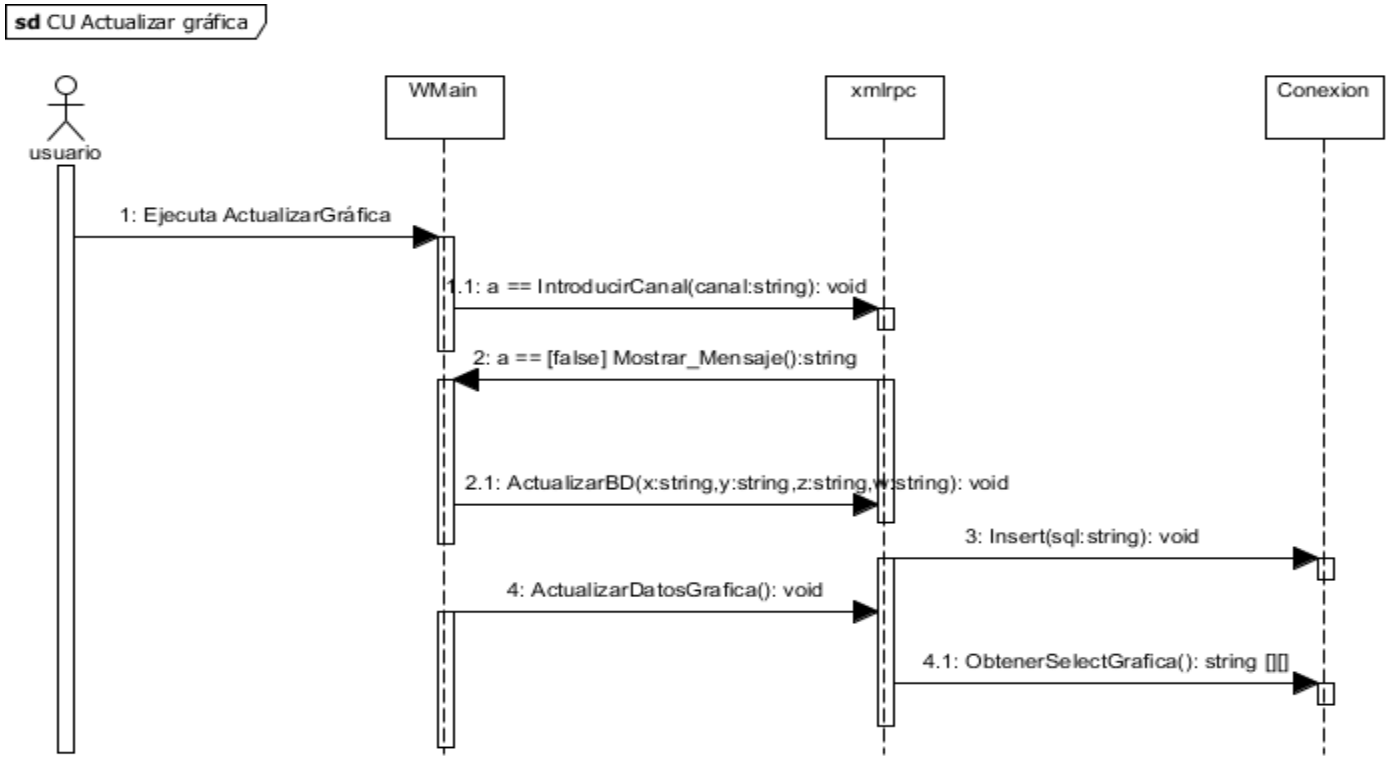


Figura 2: Diagrama de secuencia para el caso de uso actualizar gráfica

Diagrama de secuencia para el caso de uso actualizar computadora

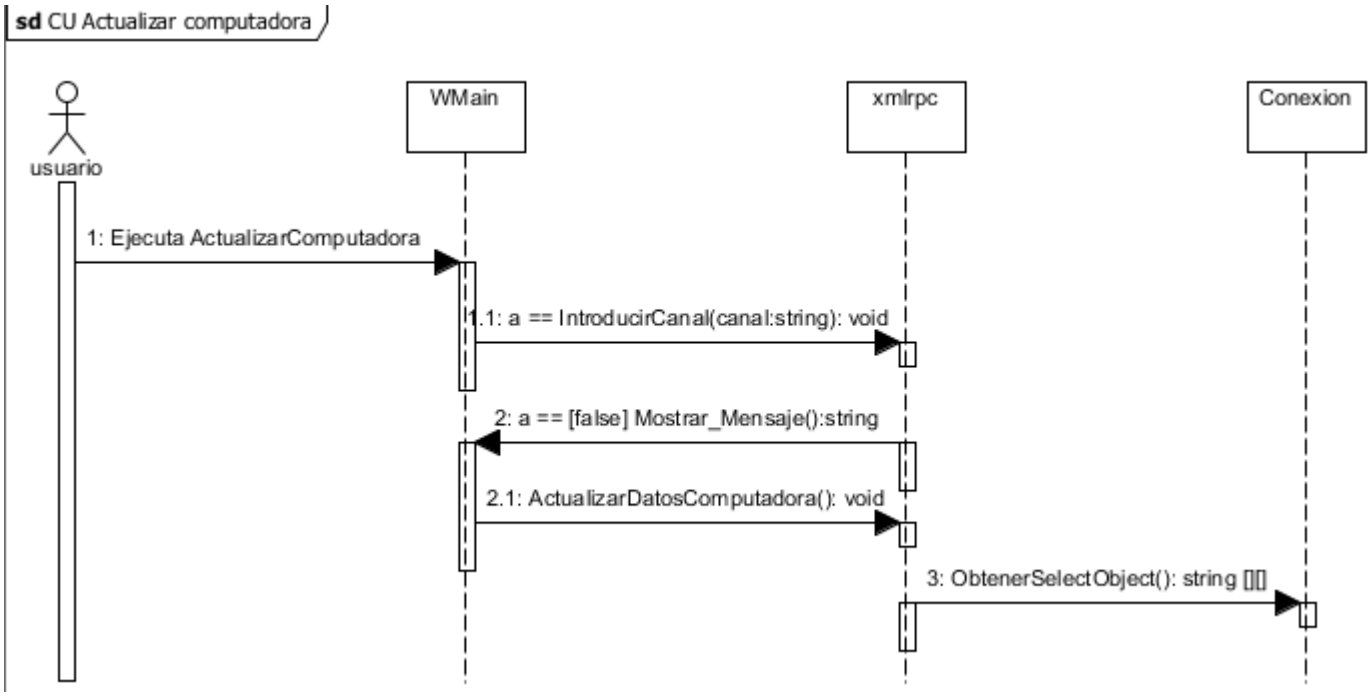


Figura 3: Diagrama de secuencia para el caso de uso actualizar computadora

2.7.4.2 Diagrama de actividad

Los diagramas de actividades son un tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.

Diagrama de actividad para obtener el canal de comunicación.

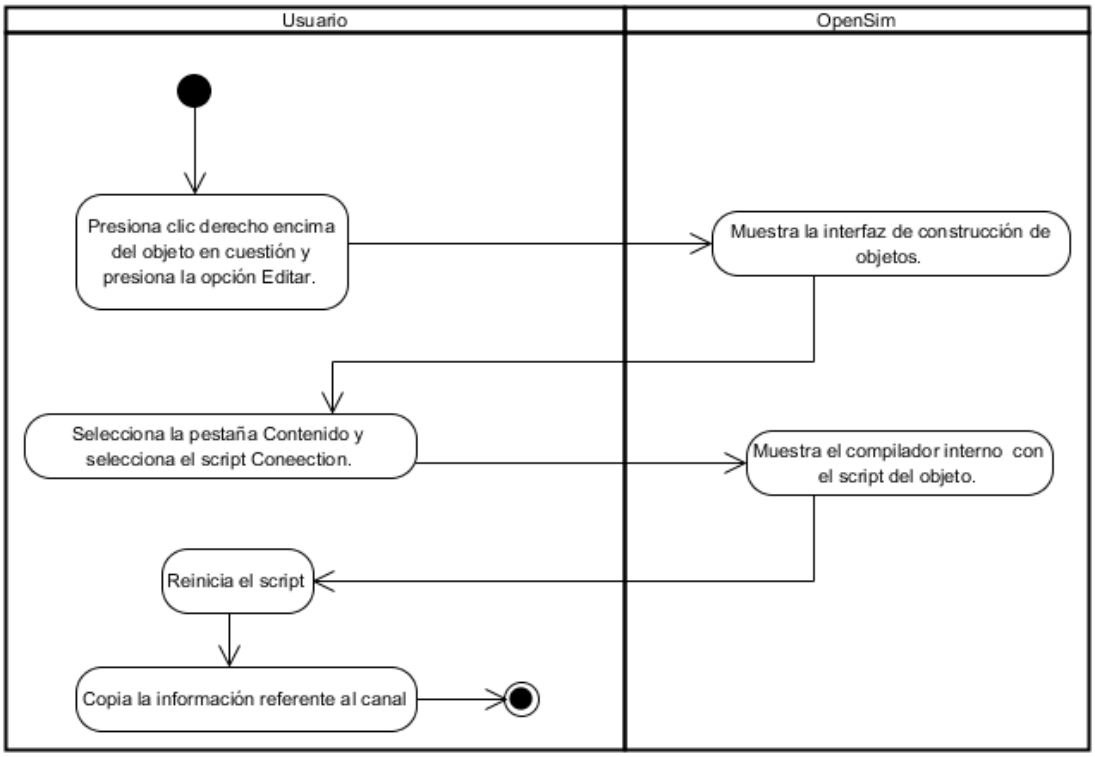


Figura 4: Diagrama de actividad para obtener canal de comunicación

2.7.4.3 Diagramas de flujos

Un diagrama de flujo representa la esquematización gráfica de un algoritmo, el cual muestra gráficamente los pasos o procesos a seguir para alcanzar la solución de un problema. Su correcta construcción es sumamente importante porque, a partir del mismo momento se escribe un programa en algún lenguaje de programación [32].

Diagrama de flujo para el objeto receptor

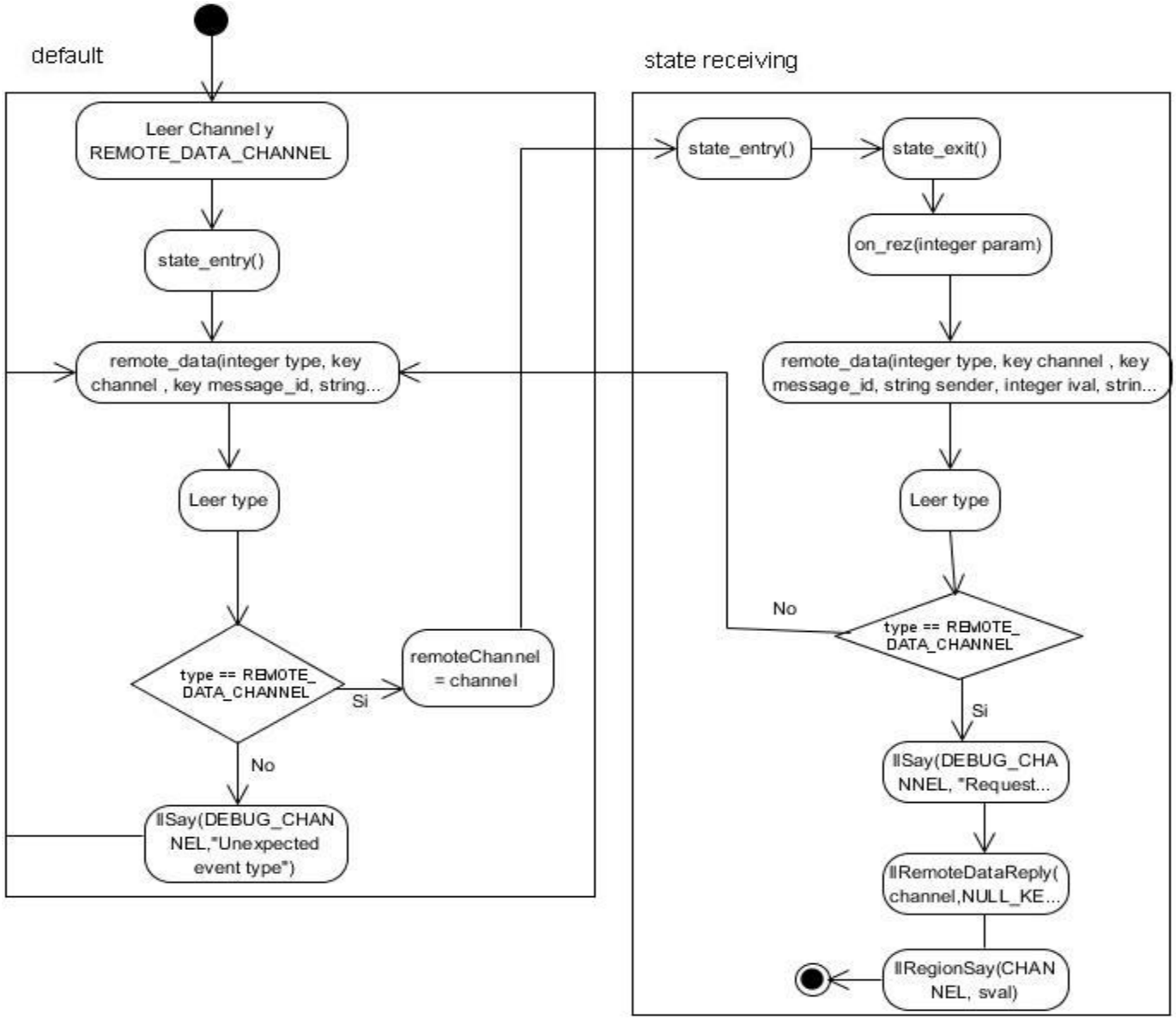


Figura 5: Diagrama de flujo para el objeto receptor

Diagrama de flujo para los objetos de la gráfica

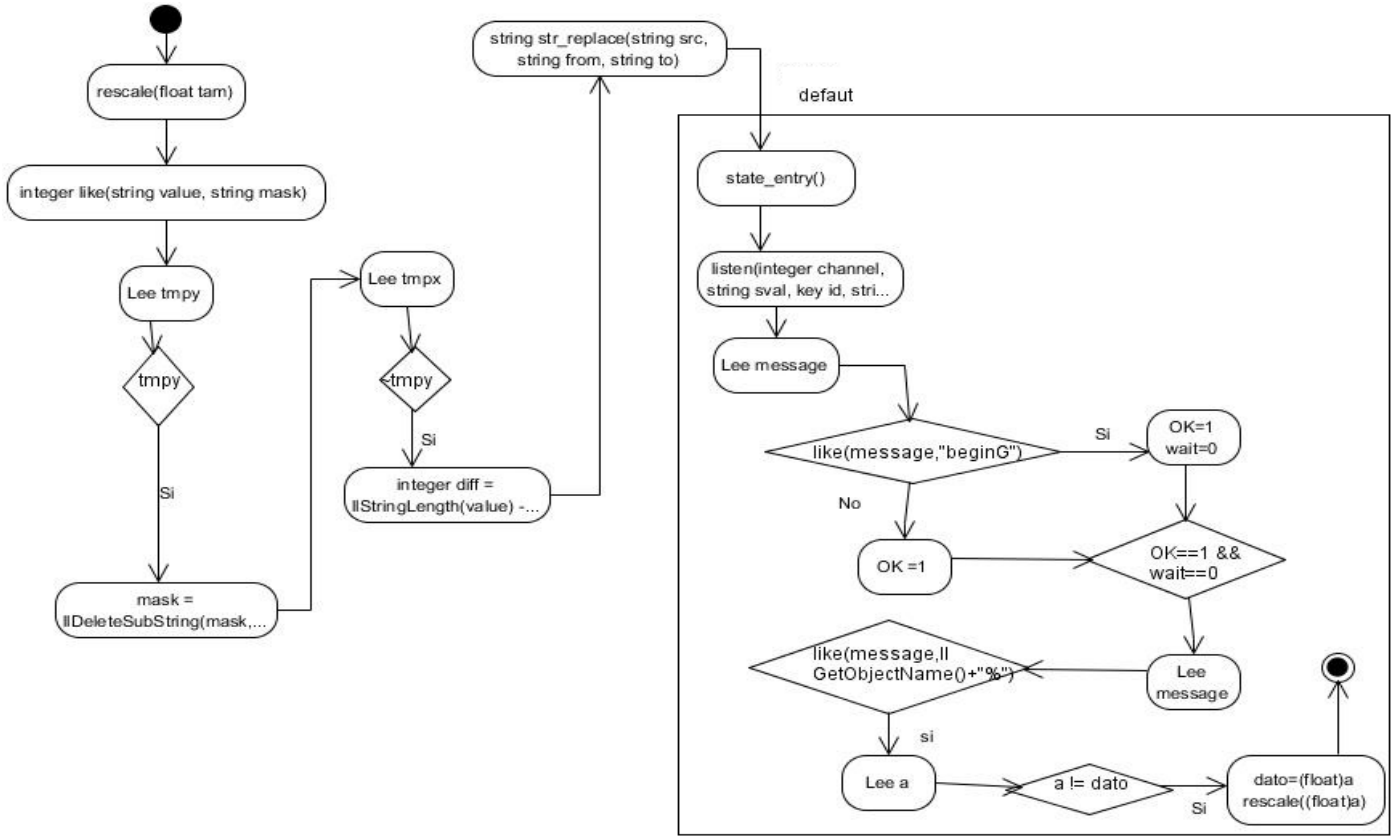


Figura 6: Diagrama de flujo para los objetos de la gráfica

Diagrama de flujo para los objetos de la computadora

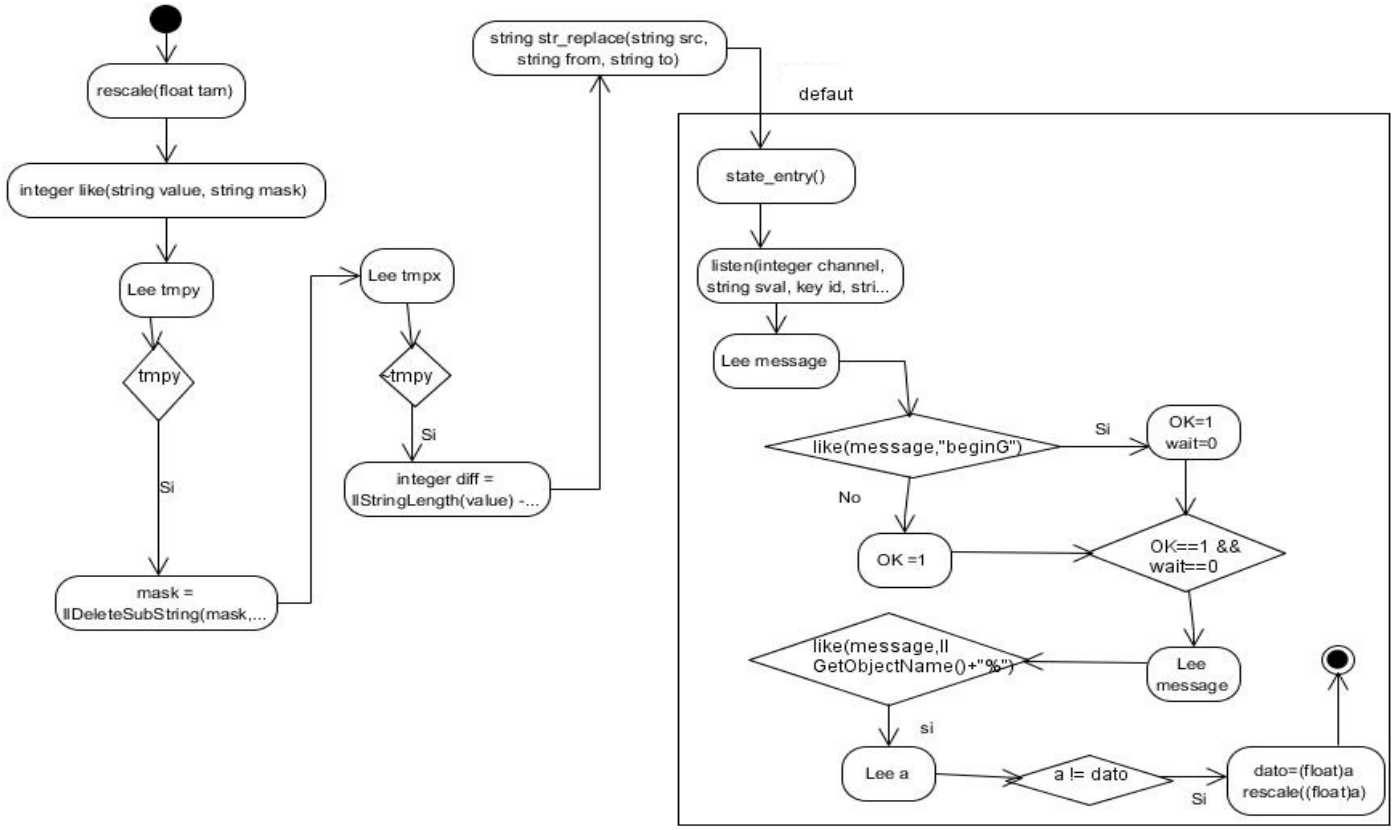


Figura 7: Diagrama de flujo para los objetos de la computadora

2.8 Patrones de diseño.

Los patrones de diseño son una solución reutilizable de problemas recurrentes que ocurren durante el desarrollo del software. Ayudan a entender las soluciones del problema con un vocabulario igual lo que permite un mejor entendimiento [33].

Los patrones de diseño usado de describen a continuación:

2.8.1 Patrones GRASP

- Creador: este patrón tiene la responsabilidad de crear objetos de las clases, se puede ver en la clase Wmain que crea objetos de la clase xmlrpc y la clase WConnection que crea objetos de la clase Conexión
- Bajo acoplamiento: acoplamiento es la medida de cuánto una clase está conectada (tiene conocimiento) de otras clases, permite que el diseño de clases sea más independiente. El patrón

está presente pues existe una mínima relación entre las clases, las cuales tienen responsabilidades diferentes. La clase conexión es la encargada de realizar la conexión a la base de datos, y la clase xmlrpc es la controladora, donde se realizan todos los requisitos funcionales planteados excepto el de conectarse a la base de datos.

2.9 El modelo de implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos [35].

2.9.1 Diagrama de Despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación [36].

Figura 8: Diagrama de Despliegue

2.9.2 Implementación de los casos de uso.

Implementación del CU Conectar con la base de datos

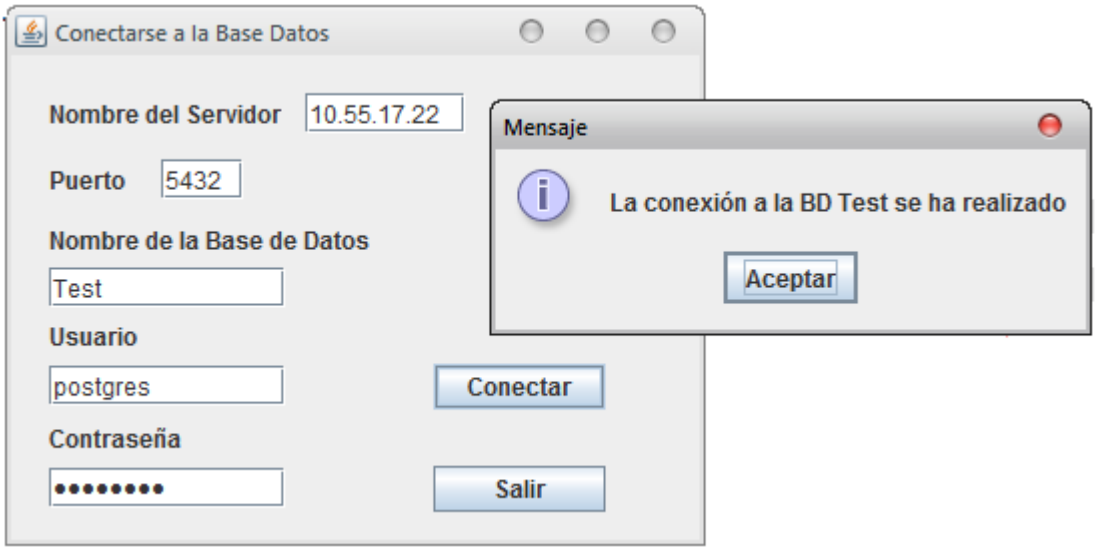


Figura 9: Implementación de caso de uso Conectar con la Base Datos

Implementación del CU Actualizar gráfica

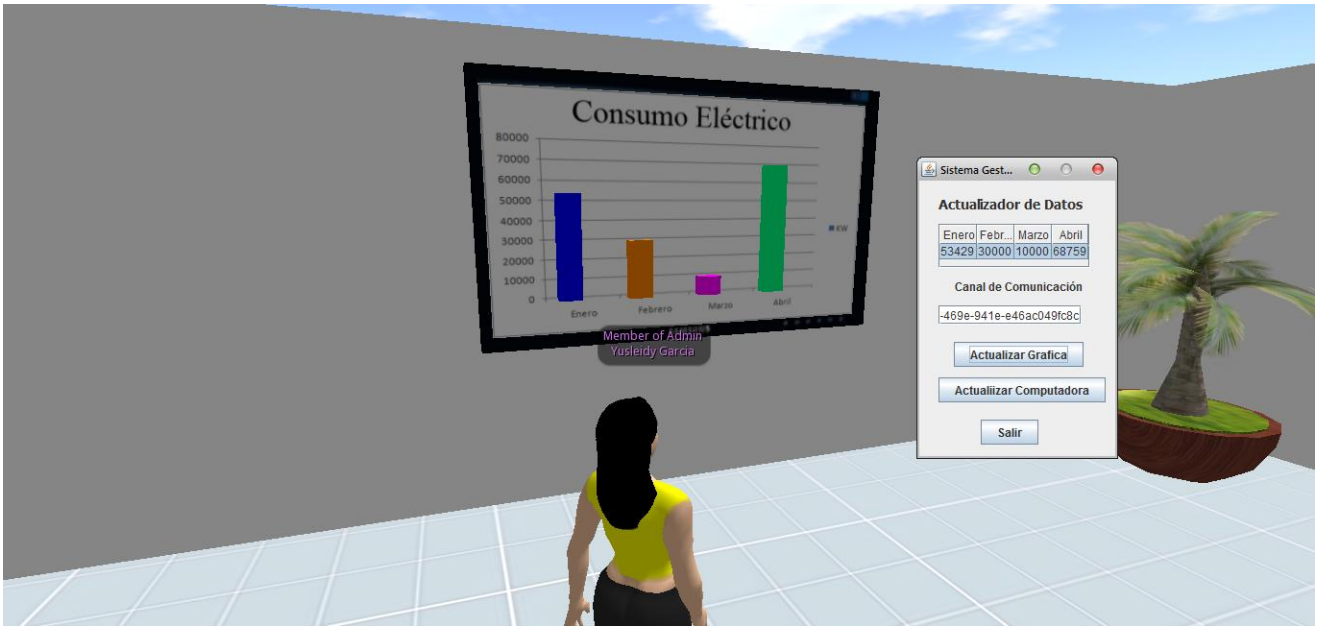


Figura 10: Implementación de caso de uso Actualizar Gráfica

Implementación del CU Actualizar computadora

Figura 11: Implementación de caso de uso Actualizar computadora

Consideraciones del capítulo

En el capítulo se seleccionaron los lenguajes, metodologías y herramientas a utilizar. Mediante la descripción de la propuesta de solución se le muestra al usuario como se cumplirán los objetivos trazados. Se especificaron los requisitos de software, se crearon y representaron los diagramas de clases del diseño y diagramas de secuencia del diseño para cada caso de uso para obtener de forma más detallada las interacciones entre los objetos del diseño, fueron realizados diagramas de flujo para representar el código de los objetos de OpenSim. Con la realización del modelo de dominio se obtuvo una mejor comprensión de los conceptos y procesos que se manejan. La definición, representación y descripción textual de los casos de uso del sistema permitió especificar las funcionalidades del sistema y los actores que interactuarán con el mismo. Se describieron paso a paso la secuencia de eventos que los actores realizarán para inicializar un caso de uso y la respuesta del sistema ante cada acción del actor, se definieron dos casos de uso tanto para el caso de estudio del consumo eléctrico como para el caso de estudio de actualizar la computadora. De forma general con la conclusión del presente capítulo se logró describir el sistema que se desea construir lo cual dará paso a una nueva etapa del desarrollo del trabajo para darle cumplimiento a los objetivos trazados.

Capítulo 3. Resultados y discusión

3.1 Introducción

En el presente capítulo se realizan los modelos de pruebas para cada caso de uso. Se proponen procedimientos para realizar una comunicación general con los objetos de OpenSim y para los casos de estudio realizados.

3.2 Discusión de los resultados

Una forma para realizar comunicación entre los objetos internos de OpenSim y el mundo externo es a través del uso de la biblioteca XML-RPC la cual permite enviar mensajes a un objeto dentro de una región, siempre que éste se haya registrado para recibir dicha comunicación.

En este epígrafe se describirán los pasos necesarios para lograr tal comunicación a través de un ejemplo sencillo, lo cual servirá posteriormente para lograr cualquier comunicación con este mundo virtual, dando así respuesta a la problemática planteada.

Paso # 1

Crear un proyecto en NetBeans de tipo Java y Java Application como se muestra a continuación:

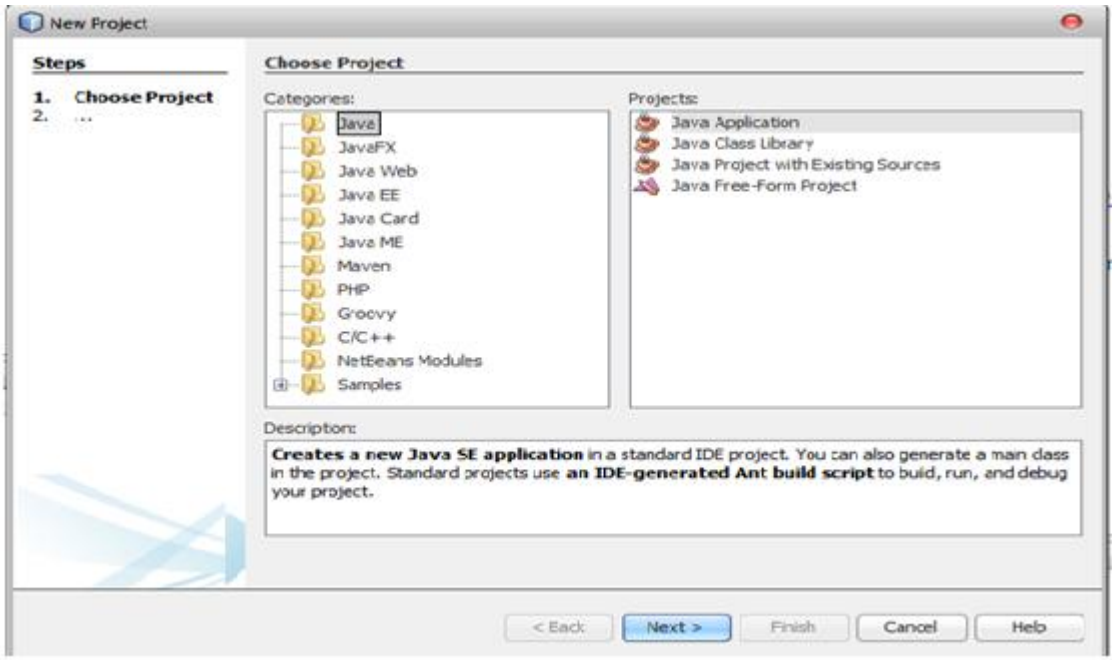


Figura 12: Crear un proyecto en NetBeans

Seguidamente se llenan todos los campos especificando el nombre del proyecto y se le da finalizar.

Paso # 2

Después de creado el proyecto se copia el siguiente código.

Se incluyen todas las bibliotecas necesarias que exige el netbeans y la biblioteca XML RPC para el envío de los datos.

```
public class XMLRPC {
public static void main (String [] args) {
try {
@SuppressWarnings("Use OfObsolete Collection Type")
HashtabletheData = new Hashtable();
XmlRpcClient server = new XmlRpcClient("http://10.55.18.100:20800/cgi-bin/xmlrpc.cgi"); //servidor donde se encuentra la aplicación.
theData.put("Channel","a02f84dd-f8e7-4161-bf40-9d9aadddbf3c");
theData.put("IntValue",2483);
theData.put("String Value","La fecha es: "+ (new Date()).toString() );
Vector params = new Vector();
params.add(theData);
 Object result = server.execute("llRemoteData", params);
System.out.println(result); }
catch (XmlRpcException | IOException e) {
e.toString(); } } }
```

Paso # 3

Después en el fichero OpenSimDefaults.ini se habilitan el módulo y el puerto como se muestra en la imagen y se ejecuta el OpenSim.exe

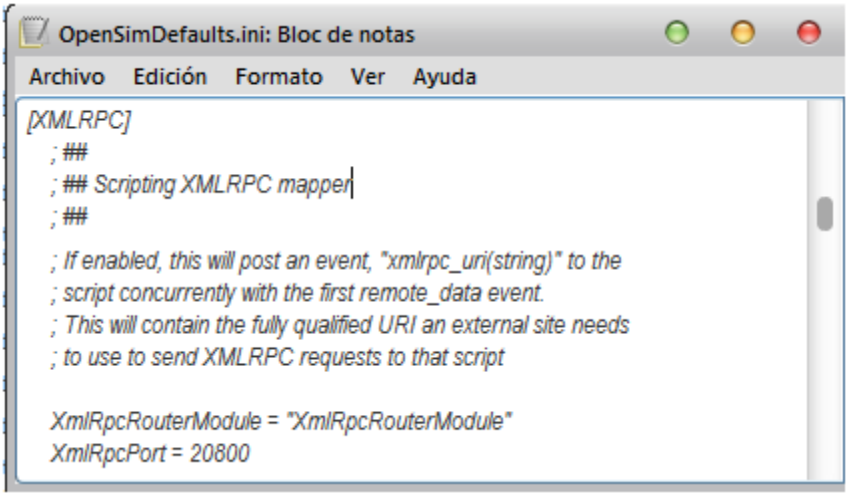


Figura 13: Habilitar el módulo y puerto

Paso # 4

Se abre el navegador de OpenSim y se crea un objeto para pasarle los datos. Para crear un objeto se va a la barra inferior en donde dice construir, o directamente a través de la tecla B. Seguidamente se edita el objeto dando clic derecho encima y seleccionando la opción editar, después le aparece la siguiente ventana:

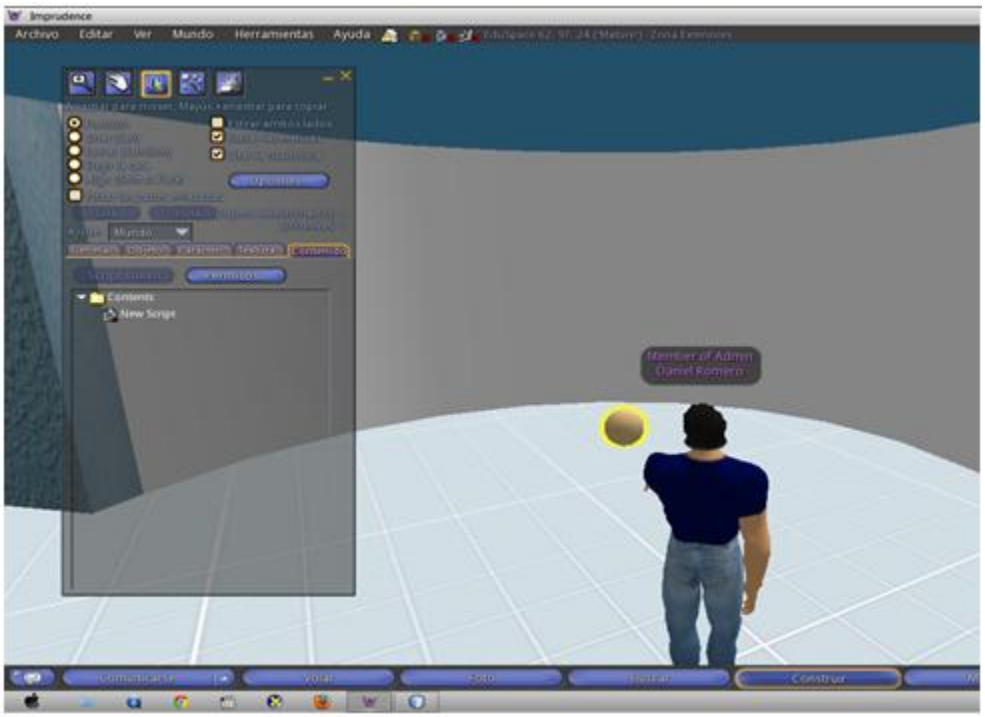


Figura 14: Crear un Objeto en OpenSim

Se busca la pestaña que dice Contenido y en New Script se copia el siguiente código:

```
key remoteChannel;
init(){
llOpenRemoteDataChannel();
llOwnerSay("My key is "+ (string)llGetKey()); }
default{
state_entry()
{ init(); }
state_exit(){
return;
on_rez(integer param){
llResetScript(); }
remote_data(integer type, key channel , key message_id, string sender, integer ival, string sval)
{ if(type == REMOTE_DATA_CHANNEL) {
llSay(DEBUG_CHANNEL, "Channel open for REMOTE_DATA_CHANNEL " + (string)channel + " " +
(string)message_id + " " + (string)sender + " " + (string)ival + " " + (string)sval);
remoteChannel = channel;
llOwnerSay("Ready to receive requests on channel \""+(string)channel + "\"");
state receiving; }
else{
llSay(DEBUG_CHANNEL,"Unexpected event type"); } } }
state receiving{
state_entry() {
llOwnerSay("Ready to receive information from outside SL"); }
state_exit(){
llOwnerSay("No longer receiving information from outside Sl.");
llCloseRemoteDataChannel(remoteChannel); }
on_rez(integer param){
```

```
llResetScript(); }
remote_data(integer type, key channel , key message_id, string sender, integer ival, string sval) {

 if(type == REMOTE_DATA_REQUEST) {
llSay(DEBUG_CHANNEL, "Request received for REMOTE_DATA_REQUEST " + (string)channel + " " +
(string)message_id + " " + (string)sender + " " + (string)ival + " " + (string)sval);
llRemoteDataReply(channel,NULL_KEY,"I got it", 2008);
llOwnerSay("I just received data in "+ llGetRegionName() + " at position " + (string)llGetPos() + "\n" + " The
string was " + sval + "\n The number was " + (string)ival + ".");  } } }
```

A continuación se muestra una imagen con el script.

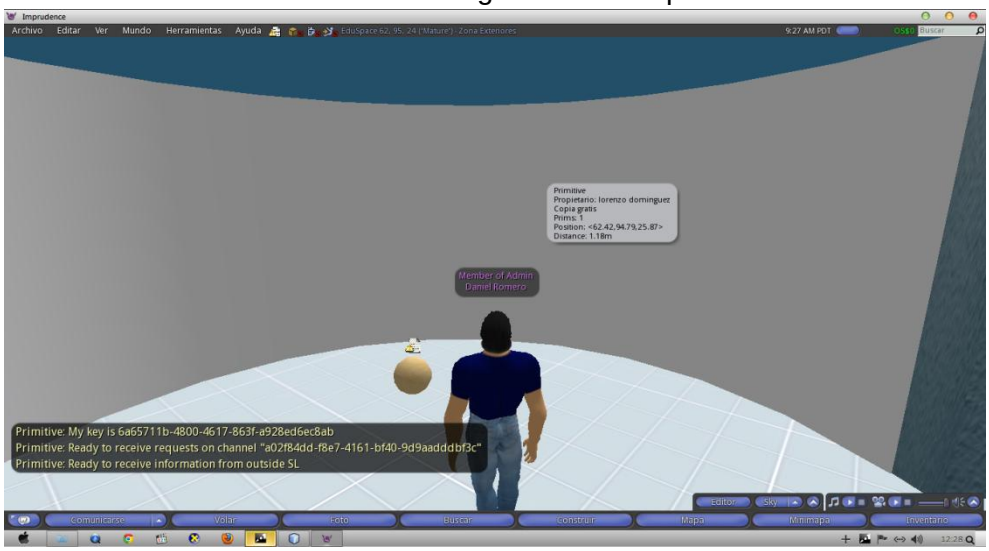


Figura 15: Datos del objeto

Una vez ejecutado el script se muestran los datos exitosamente.

3.3 Pasos para los casos de estudios desarrollados.

Este epígrafe propone los pasos necesarios para desarrollar los casos de estudios que se proponen como resultado.

Paso 1: Incluir las siguientes bibliotecas al Netbeans para el envío de datos a los objetos y la conexión con el servidor de base de datos.

- Postgresql-8.3-603.jdbc4.jar

- cis69mc.jar

Paso 2: Incluir las clases necesarias para realizar la conexión a la base de datos. En la clase controladora los métodos de suma importancia serán mostrados en el presente epígrafe.

Primeramente incluir todas las bibliotecas que sean necesarias para el trabajo con netbeans. Se declaran los atributos necesarios y se crea un objeto de la clase encargada de realizar la conexión. En el ejemplo quedaría de la siguiente forma:

```
Public class Controladora {
Private String graphics; // ejemplo de declaración de atributos.
Conexión conection; //ejemplo de declaración de atributos.
```

Para obtener los datos de la base de datos realizar las consultas necesarias, a continuación se muestra un ejemplo.

```
graphics = "SELECT \"Dato\".\"Enero\",\"Dato\".\"Febrero\", \"Dato\".\"Marzo\",\"Dato\".\"Abril\" FROM \"Dato\";";
```

Métodos importantes en la clase controladora.

El siguiente método permite actualizar los medios de la computadora en el mundo virtual.

```
Public void Actualizar_Datos_Computadora() {
Enviar_Datos("beginPC", ChannelObject);
String [][] datas = ObtenerSelectObjects();
for (int i = 0; i <datas.length; i++) {
String info = datas[i][0]+" Serie: "+datas[i][1]+"\n Estado: "+datas[i][2]+"\n Ubicación: "+datas[i][3];
Enviar_Datos(info, ChannelObject); }
Enviar_Datos("endPC", ChannelObject) }
```

El siguiente método permite actualizar la gráfica que representa el control de consumo eléctrico en el mundo virtual;

```
Public void Actualizar_Datos_Grafica(){
Enviar_Datos("beginG", ChannelObject);
```

```
String[][] data = ObtenerSelectGrafica();
for (int i = 0; i < 4; i++) {
float a = (float) ((Float.valueOf(data[0][i])*1.3)/80000);
Enviar_Datos("Dato"+String.valueOf(i+1)+String.valueOf(a), ChannelObject);}
Enviar_Datos("endG", ChannelObject);}
```

El siguiente método es imprescindible, pues es el encargado de enviar la información a los objetos de OpenSim a través del canal de comunicación ChannelObject.

```
public void Enviar_Datos(String valor, String channel){
@SuppressWarnings("Use OfObsolete CollectionType")
HashtabletheData = new Hashtable();
XmlRpcClient server = null;
Object put = theData.put("Channel",channel);
theData.put("IntValue",1.3);
theData.put("StringValue",valor/*"La Fecha es: "+ (new Date()).toString() */);
Vector params = new Vector();
params.add(theData);{
Object result = server.execute("llRemoteData", params);}
```

Dentro del mundo virtual no es necesario realizar todos los pasos involucrados pues en el ejemplo generar del epígrafe discusión de los resultado se proponen todos los pasos para realizar la comunicación, a continuación se muestran los pasos detallados para los casos de estudio que se proponen.

Paso 1: En cada barra que compone la gráfica copiar el siguiente script:

```
integer CHANNEL = 99; vector startingPos=<64.209, 89.885, 25.794>;integerchannel_dialog;integer OK = 0;integerwait=0;float dato=-1; //declaración de las variables necesarias.
```

El siguiente método permite rescalar las barras de la gráfica para que represente el nuevo valor en caso de ser modificada en la aplicación.

```
rescale(float tam){
```

```
llSetScale(<0.0, 0.0, 0.0>);
llSetPos(startingPos);
float z= tam/2;
llSetPos(llGetPos() + <0.0,0.0,z>);
llSetScale(<0.2, 0.1, tam>);}
```

El siguiente método es para saber si una cadena es subcadena de otra, para cuando el objeto encargado de enviar la información a los demás objetos estos verifiquen si su nombre está dentro de la cadena de información que envía, para de esa forma tomar la información correspondiente a cada objeto según su nombre.

```
integer like(string value, string mask) {
integertmpy = (llGetSubString(mask, 0, 0) == "%") | ((llGetSubString(mask, -1, -1) == "%") << 1);
if(tmpy)
mask = llDeleteSubString(mask, (tmpy / -2), -(tmpy == 2));
integertmpx = llSubStringIndex(value, mask);
if(~tmpx) {
integer diff = llStringLength(value) - llStringLength(mask);
return ((!tmpy&& !diff)|| ((tmpy == 1) && (tmpx == diff))|| ((tmpy == 2) && !tmpx)|| (tmpy == 3)); }
return FALSE; }
```

Después de que cada objeto lee la información una vez que coincida su nombre con algún String de la cadena éste sustituye su nombre por vacio y lo toma, el resto de la cadena la muestra como mensaje. A continuación el código encargado de realizar esta operación.

```
stringstr_replace(stringsrc, stringfrom, stringto){
integerlen = (~-(llStringLength(from)));
if(~len){
string buffer = src;
integerb_pos = -1;
integerto_len = (~-(llStringLength(to)));
```

```
integerto_pos = ~llSubStringIndex(buffer, from);
  @loop;
if(to_pos){
b_pos -= to_pos;
src = llInsertString(llDeleteSubString(src, b_pos, b_pos + len), b_pos, to);
b_pos += to_len;
buffer = llGetSubString(src, (-~(b_pos)), 0x8000);
jump loop; }}
returnsrc;}
```

El siguiente método es un estado que espera a que el objeto receptor le envíe la información mediante su canal de comunicación.

```
default{
state_entry() {
llListen(CHANNEL, "", NULL_KEY, ""); // listen for dialog answers (from multiple users)
listen(integer channel, string sval, key id, string message) {
if(like(message,"beginG")){
OK=1;
wait=0;}
else if(like(message,"endG")) {
OK=0; }
if(OK==1 && wait==0){
if(like(message,llGetObjectName()+"%")) {
string a="";
a=str_replace(message,llGetObjectName(),"");
if((float)a != dato) {
dato=(float)a;
rescale((float)a);}
```

wait=1;} }}}

Paso 2: En cada objeto que componen la computadora copiar el siguiente script:

integer CHANNEL = 99; list colourchoices = ["OK"];key ToucherID;integer channel_dialog;string info ="";integer OK = 0;integer wait=0; //declaración de variables.

integerlike() //igual que el método del script de la gráfica.
stringstr_replace()//igual que el método del script de la gráfica.

El siguiente mótodo inicializa los valores y las funcionalidades

```
state_entry() {
llSetText(llGetObjectName(),<1,0,0>,1.0);
llListen(CHANNEL, "", NULL_KEY, "");
channel_dialog = ( -1 * (integer)("0x"+llGetSubString((string)llGetKey(),-5,-1)) );}
touch_start(integer total_number){
ToucherID = llDetectedKey(0);
llDialog(ToucherID, info,colourchoices, channel_dialog);}
listen(integer channel, string name, key id, string message) {
if(like(message,"beginPC")){
OK=1;wait=0;
llSetText(llGetObjectName(),<1,0,0>,1.0); }
else if(like(message,"endPC")) {
OK=0;}
if(OK==1 && wait==0){
if(like(message,llGetObjectName()+"%")){
string a="";
a=str_replace(message,llGetObjectName(),"");
info ="La última información referente a este objeto es: \n"+a;
```

```
llSetText(llGetObjectName(),<0,1,0>,1.0);
wait=1;}}}}
```

Paso 3: En el script del objeto receptor (objeto encargado de enviar información a los objetos secundarios que son los objetos de la gráfica y la computadora.)

```
keyremoteChannel;integer CHANNEL = 99; // dialog channel
```

El siguiente método se encarga de inicializar datos.

```
init(){
llOpenRemoteDataChannel();
llOwnerSay("My key is "+ (string)llGetKey());}
```

El método defaut espera a que se active el envío de datos desde el exterior de OpenSim

```
default{
    state_entry(){
    init();}
    state_exit(){
    return;

    on_rez(integer param){
    llResetScript();}

    remote_data(integer type, key channel , key message_id, string sender, integer ival, string sval){
    if(type == REMOTE_DATA_CHANNEL){
    llOwnerSay("Ready to receive requests on channel \""+(string)channel + "\"");
    state receiving;}
    else{
    llSay(DEBUG_CHANNEL,"Unexpected event type");}}}
```

El siguiente método espera los datos, y cuando los recibe los envía.

```
    state receiving{
```

```
state_entry(){
        state_exit(){
        llCloseRemoteDataChannel(remoteChannel);}
        on_rez(integer param){
        llResetScript();}
        remote_data(integer type, key channel , key message_id, string sender, integer ival, string
sval){
if(type == REMOTE_DATA_REQUEST){
llRemoteDataReply(channel,NULL_KEY,"I got it", 2008);
        llRegionSay(CHANNEL, sval); }}}
```

3.4 Modelo de prueba

Los modelos de pruebas responden a la implementación de los casos de uso más importantes de la aplicación. Para documentarlos se utilizó el formato de los documentos oficiales del CEDIN. Se realizan pruebas de caja negra, especialmente dirigidas a interacción del usuario con el sistema y las respuestas de este mediante la interfaz.

3.4.1 Diseño de caso de prueba. CU Conectar con la base de datos

Descripción general:

El usuario inicializa el caso de uso cuando ejecuta la aplicación para conectarse a la base de datos, el mismo procede a introducir los datos como son nombre del servidor, puerto, nombre de la BD, usuario y contraseña. La aplicación se conecta a la base de datos en caso de introducir todos los datos correctos, sino muestra mensajes de error, también muestra un mensaje cuando la conexión es correcta.

Condiciones de ejecución:

Debe existir la base de datos a conectar.

Secciones a probar en el caso de uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central

SC 1: Conectar con la base de datos.	**EC 1.1:** Conectar con la base de datos.	Al ejecutarse el sistema, este se conecta con la base de datos y muestra la interfaz de conexión con la base de datos, permitiendo al usuario introducir los datos necesarios para realizar la conexión.	Se ejecuta el sistema. Se conecta con la base de datos. Muestra interfaz de conexión. Se introducen los datos. Se muestra un mensaje de conexión satisfactoria.

Tabla 6: Diseño de caso de prueba.CU Conectar con la base de datos

3.4.2 Diseño de caso de prueba. CU Actualizar gráfica

Descripción general:

El usuario inicializa el caso de uso después de haber logrado la conexión con la base de datos, posteriormente obtiene el canal de comunicación en OpenSim y lo introduce en la aplicación “Actualizador de datos”, el caso de uso permite modificar los valores de la gráfica en la aplicación y mostrar los cambios realizados en OpenSim.

Condiciones de ejecución:

Haber realizado la conexión a la base de datos.

Haber obtenido el canal de comunicación.

Secciones a probar en el caso de uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Actualizar gráfica.	**EC 1.1:** Actualizar gráfica.	Al introducir el canal de comunicación se da clic encima del botón Actualizar Gráfica. En OpenSim se actualiza los valores de la gráfica si estos fueron modificados en la aplicación, sino	Se introduce el canal. Se presiona el botón Actualizar Gráfica. Se actualizan valores de la gráfica y se muestran los cambios en OpenSim.

		la gráfica no cambia.	

Tabla 7: Diseño de caso de prueba.CU Actualizar Gráfica

3.4.3 Diseño de caso de prueba. CU Actualizar computadora

Descripción general:

El usuario inicializa el caso de uso después de haber logrado la conexión con la base de datos, posteriormente obtiene el canal de comunicación en OpenSim y lo introduce en la aplicación "Actualizador de datos", el caso de uso permite modificar los valores de la gráfica en la aplicación y mostrar los cambios realizados en OpenSim.

Condiciones de ejecución:

Haber realizado la conexión a la base de datos.

Haber obtenido el canal de comunicación.

Secciones a probar en el caso de uso:

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC 1: Actualizar computadora.	**EC 1.1:** Actualizar computadora.	Al introducir el canal de comunicación se da clic encima del botón Actualizar Computadora. En OpenSim se actualiza los nombres de los medios de la computadora mientras les llega la información. Al hacer clic encima de un medio se muestra un mensaje con todos sus datos.	Se introduce el canal. Se presionar el botón Actualizar Computadora. Se actualizan los medios. Se muestran los datos de los medios en OpenSim.

Tabla 8: Diseño de caso de prueba.CU Actualizar computadora

Consideraciones del capítulo

En el capítulo se realizaron los modelos de prueba para caso de uso con el objetivo de verificar el funcionamiento del sistema. Se describieron los procedimientos para lograr la comunicación con los objetos de OpenSim de manera general y específica para cada caso de estudio, dándole así solución a la problemática planteada.

Conclusiones

- Con el procedimiento propuesto se logra la comunicación desde el exterior con el entorno virtual de OpenSim.
- Con el desarrollo de los casos de estudio del consumo eléctrico y el control de los medios de las computadoras se demostró el funcionamiento de la biblioteca XML RPC y la efectividad del proceso propuesto.

Recomendaciones

Para futuros resultados del metaverso OpenSim se recomienda:

- Aplicar la investigación a los proyectos para la Aduana de Cuba y la dirección de extensión universitaria.
- Aplicar la investigación a otras entidades que necesiten amplificar las posibilidades para el establecimiento de relaciones humanas y para la creación de proyectos tanto educativos como de otras esferas.
- Aplicar la investigación como base para el desarrollo de iniciativas empresariales, artísticas y educativas.
- Investigar como cargar diferentes formatos de video en el entorno del metaverso.

Trabajos citados

1. **Rojo Sánchez, Don Eduardo.** scribd. *scribd.* [En línea] Octubre de 2010. [Citado el: 12 de Marzo de 2012.] http://es.scribd.com/doc/64113855/5/Analisis-de-las-alternativas-de-los-mundos-virtuales#outer_page_106.

2. campusvirtual. *campusvirtual.* [En línea] 4 de Octubre de 2012. [Citado el: 13 de Marzo de 2012.] http://campusvirtual.unex.es/cala/epistemowikia/index.php?title=Mundos_virtuales#Concepto_de_Realidad_Virtual.

3. **Angel-Mendez, Manuel.** elpais. *elpais.* [En línea] 4 de Diciembre de 2008. [Citado el: 15 de Marzo de 2012.] http://www.elpais.com/articulo/red/mundos/virtuales/abren/paso/colaboracion/empresarial/elpeputeccib/20081204elpcibenr_3/Tes.

4. larevistainformatica. *larevistainformatica.* [En línea] 2006. [Citado el: 15 de Marzo de 2012.] http://www.larevistainformatica.com/C1.htm.

5. cad.com.mx. *cad.com.mx.* [En línea] [Citado el: 15 de Marzo de 2012.] http://www.cad.com.mx/historia_del_lenguaje_java.htm.

6. **Wells.** extremeprogramming. *extremeprogramming.* [En línea] 1999-2009. [Citado el: 16 de Marzo de 2012.] http://www.extremeprogramming.org/.

7. **Jeffries, Ron.** xprogramming. *xprogramming.* [En línea] 8 de Noviembre de 2001. [Citado el: 15 de Marzo de 2012.] http://xprogramming.com/book/whatisxp/.8solocsharpsolocsharphttp://solocsharp.blogspot.com/2010/02/concepto-versiones-visual-studio.html

9. **Guamán Salazar, Lilian.** scribd. *scribd.* [En línea] http://www.scribd.com/lily_guam%C3%A1n/d/82649610-Rup.

10. sistemas.uniandes. *sistemas.uniandes.* [En línea] [Citado el: 19 de Abril de 2012.] http://sistemas.uniandes.edu.co/~isis2701/dokuwiki/lib/exe/fetch.php?media=isis2701-patronesgrasp.pdf.

11. ecured. *ecured.* [En línea] [Citado el: 1 de Mayo de 2012.] http://www.ecured.cu/index.php/Patrones_de_Casos_de_Uso.

12. marlonj. *marlonj.* [En línea] 1 de Mayo de 2012. http://www.marlonj.com/blog/2008/12/lsl-xml-rpc-en-opensimulator/.

13. xmlrpc. *xmlrpc.* [En línea] [Citado el: 1 de Abril de 2012.] http://xmlrpc.scripting.com/.

14. merinde. *merinde.* [En línea] 12 de Mayo de 2012. [Citado el: 15 de Abril de 2012.] http://merinde.net/index.php?option=com_content&task=view&id=495&Itemid=291.

15. **Jacobson, y otros.***El Proceso Unificado de Desarrollo de Software.* 2004.

16. diflu. *diflu.* [En línea] [Citado el: 16 de Mayo de 2012.] http://diflu.shtml.htm.com.

17. [En línea] [Citado el: 18 de Marzo de 2012.] http://www. Fvirtual.usalesiana.edu.bo

18. **S.Coop, ISEA.***INTERNET 3D,Análisis prospectivo de las potenciales aplicaciones asociadas a los Mundos Virttuales.* 2008.

19. o3Dsoft Blog. *o3Dsoft Blog.* [En línea] [Citado el: 12 de 1 de 2012.] http://o3Dsoft.com/blog/es/2011/04/analisis-02-opensim-en-la-educacion/.

20. **Guillermo.** slideshare. *slideshare.* [En línea] 13 de Noviembre de 2008. [Citado el: 5 de Marzo de 2012.] http://www.slideshare.net/guillermo/opensim-presentation.

21. Devi. *Devi.* [En línea] 5 de Julio de 2011. [Citado el: 2 de Febrero de 2012.] http://devi.com.co/mundos-virtuales/red-de-educacion-y-negocios-virtuales-en-opensim/.

22. OpenSimulator. *OpenSimulator.* [En línea] [Citado el: 6 de 1 de 2012.] http://opensimulator.org/wiki/Portada.

23. **Mojica, Raul.** Virtual World. *Virtual World.* [En línea] 23 de Mayo de 2010. [Citado el: 2 de 2 de 2012.] http://virtualworlds3D.wordpress.com/tag/opensim/.

24*Metodologías Agiles. Proceso Unificado Ágil (AUP).*Bolivia:Universidad Unión Bolivariana

25*El Lenguaje Unificado de Modelado.Manual de Referencia*MadridPearson Educación2005

26. NetBeans. *Información de la versión de NetBeans IDE 6.8. NetBeans.* [En línea] [Citado el: 25 de Abril de 2012.] http://netbeans.org/community/releases/68/index_es.html..

27*Gestión de la participación y los resultados obtenidos por los estudiantes en las actividades extracurriculares*2008

28. postgresql. *postgresql.* [En línea] 22 de Marzo de 2009. [Citado el: 25 de Abril de 2012.] http://www.postgresql-es.org. www.postgresql-es.org..

29. **softonic, Equipo de.** La mejor alternativa gratuita a Oracle y a SQL Server. *La mejor alternativa gratuita a Oracle y a SQL Server.* [En línea] [Citado el: 20 de Marzo de 2012.] http://mysql.softonic.com/.

30. Visual Paradigm for UML. *sitio Web de Visual Paradigm for UML*. [En línea] 2009. [Citado el: 29 de Marzo de 2012.] http://www.visual-paradigm.com..

31. **LaFuente, G.J.** Herramientas CASE. [En línea] [Citado el: 29 de Marzo de 2012.] http://gidis.ing.unlpam.edu.ar/personas/glafuente/uml/uml.html..

32. IBM. *Rational Rose Enterprise. www.ibm.com. .* [En línea] 2011. [Citado el: 29 de Marzo de 2012.] http://www-142.ibm.com/software/products/es/es/enterprise/..

33. **Jacobson, Ivar, Booch, Grady y Rumbauch, James.**El Proceso Unificado de Desarrollo de Software. *s.l. : Félix Varela.* [En línea] 2004.

34. secondlifedata*secondlifedata*http://secondlifedata.wordpress.com/category/second-life-historia/

35. scribd. *Lenguaje de programación interpretado.* [En línea] [Citado el: 26 de Marzo de 2012.] http://es.scribd.com/doc/27030831/Lenguaje-de-programacion-interpretado-syntaxis-etc-resumen-basico.

36. **Spad, Fiona.** virtual-spain. *virtual-spain.* [En línea] martes de Abril de 2010. [Citado el: 27 de Marzo de 2012.] http://www.virtual-spain.es/component/content/article/39/92.

Glosario de términos

- **Chanel**: Canal de comunicación de los objetos de OpenSim.
- **RV:** Abreviatura de Realidad Virtual.
- **Virtual**: Abstracción del mundo real.
- **UCI:** Abreviatura de Universidad de las Ciencias Informáticas.
- **Avatar:** usuario dentro del mundo virtual que representa a las personas que interactúan con el mismo.
- **Visor:** Se utiliza para conectarse a los mundos virtuales. El visor para OpenSim es el Imprudence.
- **Región:** Espacios dentro de OpenSim.
- **3D:** Abreviatura de 3 dimensiones.

Anexos

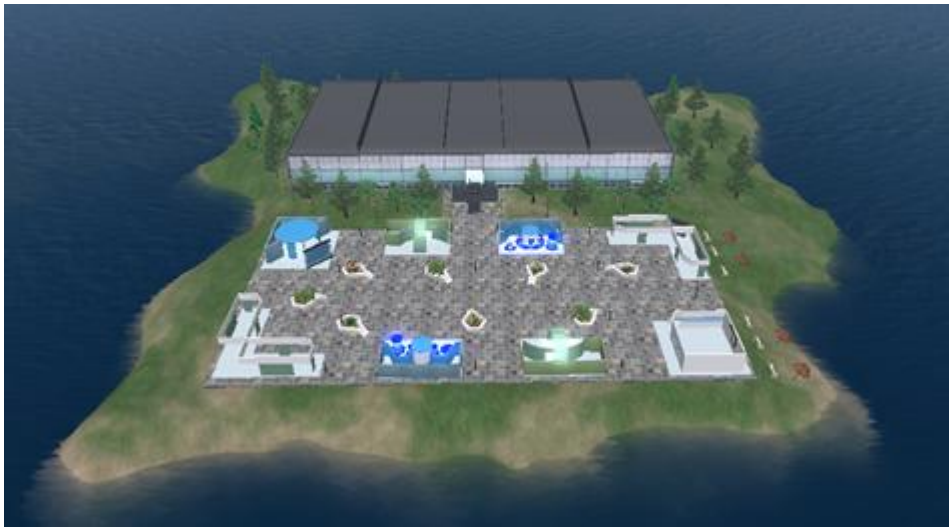


Figura 16: Mundo Virtual OpenSim en la UCI

Figura 17: Piscina en OpenSim

Figura 18: Restaurante en OpenSim

Figura 19: Parque en OpenSim