FACULTAD 5

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

GESTBOT

HERRAMIENTA PARA LA GESTIÓN DE INFORMACIÓN DE PRODUCTOS MEDIANTE AVATARES AUTÓNOMOS (BOTS) DENTRO DE UN METAVERSO.

Autor: Hermes Eduardo Ferrer Bustamante

Tutores: Ing. Lorenzo Domínguez García

Ing. Belkis G. Gonzales Rodríguez

La Habana, Cuba.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informática Industrial (CEDIN), de la Universidad de las Ciencias Informáticas (UCI), para que hagan el uso que estimen pertinente con el mismo.

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

Hermes Eduardo Ferrer Bustamante

_____
Firma del Autor

AGRADECIMIENTOS

Gracias a la vida por dame la oportunidad de contar con las personas que de una forma u otra me han apoyado y hecho una mejor persona a lo largo de estos 24 años y permitirme tener un montón de personas bellas a las cuales agradecerles por haber llegado hasta donde estoy y siendo quien soy.

A mis abuelos Josefa y Domingo por ser su "*negrito", darme todo el amor del mundo y también unas nalgadas cuando las merecía.*

A mis padres sin los cuales indiscutiblemente no estaría escribiendo estas letras y tener la dicha de contar siempre con ellos para guiarme. Mi mami la cual aun estando lejos siempre la he sentido a mi lado, y a mi pa, del cual me enorgullezco de parecerme tanto y ser mi amigo.

A mi familia en general porque me siento agradecido de tener una familia tan grande, unida y amorosa desde los tíos lejanos hasta mis tíos más tíos Busty y Carly que desde que nací han sido como unos padres uno recto el otro loco, y mi tía Mary "mi enfermera", a mis primas Yeny y Yamy las que han sido mi guía de estudiantes modelos y me utilizaban de alumno en el juego de "la escuelita", mi abuela Celeste, a mi tía Vivian y a mi padrastro Roberto por ser como un padre para mí, guiarme, aconsejarme en todo.

A mis amigos(as) con los que he compartido cincos (5) y dos (2) y se me ayudaron, especialmente a Yuly y Mayara que me venimos desde la primaria y tanto me regañaron en el preuniversitario. A los del barrio y de la universidad estos últimos que moleste a cualquier hora y siempre estaban para estudiar y FIESTAR (como me gusta) y hacer de mis días aquí un recuerdo importantísimo. Cerrando esta etapa de la vida.

A mis profesores desde la primaria hasta la universidad, sin los cuales no hubiera llegado a este momento y especialmente a mis tutores Lorenzo y Belkis.

A TODOS los que de un modo u otro han influido en lograr hacer de mí todo un INGENIERO EN CIENCIAS INFORMATICAS y MEJOR HOMBRE….

MIS ETERNOS AGRADECIMIENTOS!!!

MUCHAS GRACIAS!!!!

DEDICATORIA

A mis abuis Josefa y Domingo, a mis padres y a mis hermanos, que por ser el mayor espero poder servirles de guía para que puedan ser incluso mejores, en especial a Javi para el cual he sido un faro y trataré seguir brillando.
Para ustedes este trabajo.

Los quiero.

RESUMEN

Desde el 2003 se viene incursionando en el desarrollo de una nueva forma de interacción personal en internet, facilitado por la aparición de la web2.0, lo que trae consigo la aparición de nuevas formas de comunicarse, de compartir información y de interactuar que influyen en la vida cotidiana, a partir de los llamados Metaversos, siendo cada vez mayor su desarrollo y aplicación en la sociedad. En la Universidad de Ciencias Informáticas (UCI), se encuentra el Departamento de Visualización y Realidad Virtual del Centro de Informática Industrial (CEDIN), en el cual se viene desarrollando un Metaverso para la UCI: (UCIGRID), con el cual se pretende crear una comunidad virtual en la que se pueda llevar a cabo varias actividades, tanto educativas como sociales, entre ellas: exposiciones de trabajos, seminarios, creación de grupos sociales, simulaciones de procesos, presentaciones de ferias virtuales, entre otras.

Uno de los temas más complejos a desarrollar en un entorno virtual es la implementación de agentes autónomos (bots).

Por lo novedoso de esta tecnología, existen pocos ejemplos y documentación acerca del trabajo con bots en los metaversos, por tal motivo se hace necesario desarrollar nuevas herramientas que permitan el trabajo con los mismos para automatizar diversos procesos y dar mayor realismo y utilidad al metaverso.

Utilizando como herramienta base OpenSimulator (OpenSim), el cual es una plataforma orientada a crear ambientes 3D altamente interactivos, y que se viene desarrollando y expandiendo su uso a nivel mundial por sus amplias ventajas, entre las que destaca ser una herramienta de código abierto (open-source) y con la cual se desarrolló el metaverso de la universidad.

Como resultado de la implementación de la herramienta se desarrollaron Bots funcionales para la exposición de productos en la FESI Virtual de la Universidad y se asentaron las bases para el trabajo con los mismos en función de dar solución a algunas necesidades del metaverso UCIGRID.

Palabras clave: Metaverso, OpenSimulator, Bots.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La aparición de las actuales tendencias a partir de la denominada Web 2.0, hace posible la introducción de aplicaciones utilizadas diariamente por un gran número de usuarios y origina nuevas formas y canales de comunicación que cambian la manera de interactuar entre los usuarios de Internet, lo que influye en la vida cotidiana; a partir de lo cual se han producido durante la última década enormes progresos en el desarrollo de metaversos.

Estos mundos pueden definirse como entornos gráficos 3D simulados por ordenador, "cohabitados" por los usuarios a través de sus avatares. Tradicionalmente, los mundos virtuales se han estructurado a priori predefiniendo las tareas realizables por los usuarios. En la actualidad, en los mundos sociales virtuales, la interacción social posee un papel clave y los usuarios pueden determinar sus experiencias en el mundo virtual siguiendo sus propias decisiones. De este modo, los mundos virtuales se han transformado en verdaderas redes sociales útiles para la interacción entre personas de diferentes lugares que pueden socializar, aprender, entretenerse, entre otras. Debido al potencial social de los mundos virtuales, se han convertido en un atractivo para instituciones, empresas e investigadores, con la finalidad de desarrollar robots virtuales con las mismas apariencias y capacidades que los avatares correspondientes a usuarios humanos.

Con esta idea Lindens Labs desarrolló un programa que permitía a sus usuarios la inmersión en un mundo virtual, naciendo en el año 2003 Second Life, metaverso en el cual sus usuarios conocidos como residentes pueden explorar el mundo virtual e interactuar con otros residentes.

Teniendo como una de las características más interesantes, la posibilidad de construir objetos, lo que lo enlaza directamente con la Web 2.0, permitiendo a los usuarios interactuar y colaborar entre sí como creadores de contenido generado por los mismos en una comunidad virtual, a diferencia de sitios web donde los usuarios se limitan a la observación pasiva de los contenidos que se han creado para ellos. Ejemplos de la Web 2.0 son las comunidades web, las aplicaciones Web, los servicios de red social, los servicios de alojamiento de videos, las wikis, blogs, entre otros.

El uso de Second Life trae marcadas desventajas, figurando como principal ser privativo, y en segundo lugar los precios asociados al uso del mismo.

Second Life indudablemente es el padre de los mundos virtuales y lo seguirá siendo por un buen tiempo, pero no es una herramienta totalmente propicia para desarrollar actividades de aprendizaje con metas de crecimiento a largo plazo. Hoy existe un gran portafolio de servicios y productos en base a mundos virtuales orientados a la educación. Encontrándose una de las más grandes oportunidades en el mundo virtual de código abierto llamado *OpenSimulator (OpenSim)* que si bien no fue concebido para educación este permite crear un ambiente propicio para tal fin.

En Cuba, específicamente en la Universidad de las Ciencias Informáticas (UCI) se encuentra el Centro de Informática Industrial (CEDIN), donde un equipo del departamento de Visualización y Realidad Virtual, dentro del proyecto Escenarios 3D se viene desarrollando el metaverso "UCIGRID", con el cual se pretende crear una comunidad virtual en la que se pueda llevar a cabo varias actividades, tanto educativas como sociales, entre ellas: exposiciones de trabajos, seminarios, creación de grupos sociales, simulaciones de procesos y establecerlo además como una vía alternativa para la formación profesional y presentaciones de ferias virtuales.

Actualmente UCIGRID presenta un alto nivel de frialdad y desolación debido a la poca presencia de avatares persistentes, lo que disminuye el nivel de realismo e inmersión en dicho metaverso. Otra dificultad que presentan los mismos es que no son capaces de interactuar con otros avatares sin la manipulación de usuarios.

A partir de la situación polémica anteriormente planteada se tiene como **problema científico**.

La inexistencia de avatares autónomos capaces de manipular información dentro de un metaverso e interactuar con los usuarios.

Teniéndose como **objeto de estudio** los metaversos y **campo de acción** los bots para los metaversos.

Se plantea como **objetivo general** crear una herramienta para la gestión de información de productos mediante avatares autónomos (bots) dentro de un metaverso.

Obteniendo la siguiente **idea a defender:** si se crea una herramienta para la gestión de información de productos mediante avatares autónomos (bots) dentro de un metaverso, se podrá aumentar el nivel de realismo e inmersión en el mismo.

Entre las **tareas a desarrollar** para darle solución al objetivo planteado se encuentran las siguientes:

- Identificación de los principales conceptos, herramientas, pasos y lenguajes asociados al objeto de estudio.
- Determinación de las tendencias que han primado en las plataformas interactivas a nivel internacional y nacional.
- Evaluación del estado actual de los métodos y técnicas utilizados para la creación de bots en las plataformas interactivas.
- Realización de un estudio de las prestaciones que ofrece el entorno OpenSim para la creación de bots.
- Identificación de las ventajas de utilización de OpenSim sobre otras plataformas existentes.
- Implementación de bots para presentar los productos.
- Implementación de una herramienta para la gestión de los productos.

Resultados esperados:

Bases para la creación de bots en función de necesidades del metaverso de la universidad.

Bot funcional para la exposición de algún producto para la feria de productos virtuales del metaverso de la universidad UCIGRID.

Herramienta para la gestión de los productos a exponer en la FESI virtual de UCIGRID.

Para la realización de las tareas de investigación se emplean como **métodos de investigación** fundamentales a utilizar que se describen a continuación:

Métodos Teóricos:

Análisis-síntesis: Permite a través de la investigación de las teorías y documentos existentes el tema, hacer análisis más profundo y llegar así a su esencia, determinar los rasgos que las caracterizan y extraer los elementos más importantes que se relacionan con el mismo.

Análisis histórico – lógico: Para estudiar el comportamiento del objeto en su historia y determinar las tendencias actuales.

Empíricos:

Encuesta: Para ayudar adquirir conocimientos acerca de las necesidades que presenta la comunidad y poder reflejarlas en la aplicación.

Entrevista: Para obtener información a partir de conversaciones planificadas con especialistas en el tema, y diferentes representantes del proyecto.

Con el objetivo de organizar y darle una estructura al trabajo se ha decidido dividir el trabajo en tres capítulos:

Capítulo 1 “Fundamentación Teórica”, en este capítulo se especifican los conceptos más importantes asociados al dominio del problema para un mejor entendimiento de la investigación que se está llevando a cabo.

Capítulo 2 “Descripción de la Solución Propuesta”, en este capítulo se hace énfasis principalmente en las fases exploración, planificación de entrega e iteraciones. Se abordarán temas relacionados con el funcionamiento del sistema, los distintos tipos de usuarios, el flujo actual de la gestión de la información, y las descripciones de las distintas funcionalidades a desarrollar para finalmente obtener una propuesta final.

Capítulo 3 “Construcción de la solución propuesta”, en este capítulo se muestra detalladamente la evolución de la solución con las fases antes mencionadas. La implementación debe realizarse iterativa e incremental donde al finalizar cada una de las iteraciones se espera obtener un producto al cual se le corresponde aplicar las pruebas.

CAPÍTULO1: FUNDAMENTACIÓN TEÓRICA.

1.1 INTRODUCCIÓN.

En este capítulo se abordan los conceptos asociados al dominio del problema que resultan relevantes para una mejor comprensión del tema a desarrollar en este trabajo, se realiza un estudio del estado del arte, la reseña y el análisis de otras posibles soluciones existentes vinculadas al campo de acción.

1.2 METAVERSOS:

Esta palabra aparece por primera vez en la novela “Snow Crash”, publicada en 1992 por el escritor Neal Stephenson, en la cual la define como “un universo generado informáticamente, que el ordenador dibuja sobre el visor y le lanza a través de los auriculares”, un lugar imaginario que “no existe realmente, sino que es un protocolo infográfico escrito en papel en algún sitio” y que está formado por “fragmentos de software, puestos a disposición del público a través de la red mundial de fibra óptica” (1).

Con la materialización de esta idea y creación de metaversos se ha ido redefiniendo y enfocando el concepto. Entiéndase como mundos virtuales tridimensionales (3D), creados por computadoras en los cuales los usuarios interactúan a través de avatares. Otras definiciones interesantes se describen en (2), donde además el autor describe la posibilidad de distinguir cuatro posibles ***modelos de metaverso***::

1. **Juegos y mundos virtuales**: A este tipo pertenecen los más similares al comentado por Stephenson en Snow Crash. Son entornos virtuales inmersivos en los que el usuario se sumerge en una experiencia de contacto con otros usuarios y elementos dentro de un mundo virtual 3D. Este contacto puede estar orientado hacia el juego (como en World of Warcraft), o al aspecto social del meta-verso (como en Second Life, There, Habbo Hotel, Twinity, entre otros).

2. **Mundos espejo**: Son representaciones virtuales detalladas de uno o varios aspectos del mundo real. El ejemplo más claro es el de Google Earth, que representa la geografía mundial mediante imágenes aéreas.

3. **Realidad Aumentada**: Emplea la tecnología de mundos espejo en situaciones reales de nuestra vida cotidiana, ofreciéndonos información virtual sobre una realidad física ya existente. Por ejemplo, un individuo que está ante un monumento histórico puede consultar información sobre el mismo así como imágenes virtuales de su imagen re-construida.

4. **Lifelogging**: Son sistemas de registro digital que recogen datos sobre la vida cotidiana con el fin de ser analizados mediante estadísticas.

1.2.1 CARACTERÍSTICAS:

Según el especialista en mundos sintéticos Edward Castronova (2001), existen tres características fundamentales de los metaversos:

Interactividad. El usuario es capaz de comunicarse con el resto de usuarios y de interactuar con el metaverso. Esto implica que sus comportamientos pueden ejercer una influencia sobre los objetos y sobre los comportamientos y opiniones de otros usuarios, influencia que también puede ser recíproca.

Corporeidad. Los usuarios están representados por avatares y están limitados por una altura y un peso considerables. La corporeidad consiste en la presencia de ese avatar sobre ese espacio que también posee ciertos límites, ya que está sometido a ciertas leyes y tiene re-cursos limitados.

Persistencia. Esto significa que el programa sigue funcionando y desarrollándose a pesar de que algunos o todos sus miembros no estén conectados. Además, las posiciones en las que se encontraban los usuarios al cerrar sus sesiones, así como sus conversaciones, objetos de propiedad, entre otros; siempre son guardados, lo que permite recuperarlos cuando se reconecten.

Estas características se encuentran como elemento común en los disimiles metaversos que están en constante crecimiento cada día, con variadas funciones desde educación, juegos, hasta moda, dentro de los que destacan:

World of Warcraft:

Este juego de rol online multijugador masivo en el que personas de todo el mundo exploran e interactúan este mundo de fantasía.

Fly Bar:

Es una iniciativa de RTVE que recrea el bar de la serie "Cuéntame". (*Anexo1).*

Entropía Universo:

Es un universo virtual en crecimiento. Los participantes pueden acceder a los Entropia Universe con una sola cuenta, viajando entre todos los planetas disponibles en el juego. La magnífica calidad gráfica de los contenidos es uno de los pilares de Entropia Universe. (Anexo2)

Twinity

Mundo virtual que recrea ciudades como Berlín, Singapore y Londres. Cualquiera persona que quiera podrá crearse una cuenta para poder testear este mundo virtual. En este momento dentro del mundo virtual existe una recreación de la ciudad de Berlín de gran detalle. (Anexo3)

Nurien

Nurien Software es el creador de un mundo virtual dedicado a la moda con avatares extremadamente realistas que habitan entornos urbanos. (Anexo4)

1.3 SECOND LIFE

En 1999, Philip Rosedale 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.

Una de las características más interesantes de Second Life es la posibilidad de construir objetos, lo que enlaza directamente con la Web 2.0. Aunque existe la posibilidad de crear objetos en otros programas y exportar el resultado a Second Life, el resultado suele ser pobre. Es preferible construir las cosas desde el principio, recoger material gratuito o usar objetos de la librería y remodelarlo o reformarlo.

En Second Life se ofrecen clases virtuales para distintas universidades e instituciones educativas, algunas de las actividades primarias que los educadores realizan en Second Life:

- Andanzas al azar
- Escucha de presentaciones y conferencias
- Se conoce nuevas personas
- Participación en encuentros
- Construcción de objetos.

Second Life no es una herramienta totalmente propicia para desarrollar actividades de aprendizaje con metas de crecimiento a largo plazo. El uso de este trae marcadas desventajas, entre las que se encuentra el modelo de negocio basado en la compra de regiones o islas, lo que obliga a las instituciones a comprar terrenos o islas y a pagar una renta anual o semestral. Hoy existe un gran

portafolio de servicios y productos en base a mundos virtuales orientados a la educación una de las más grandes oportunidades se encuentra en el mundo virtual de código abierto llamado *OpenSimulator* que si bien no fue concebido para educación este si permite crear un ambiente propicio para tal fin.

1.4 OPENSIMULATOR (OPENSIM)

1.4.1 ¿QUÉ ES OPENSIM?

OpenSimulator es un servidor de aplicaciones 3D que surge a partir de la liberación del código fuente del cliente de Second Life. Pudiéndose utilizar para operar mundos virtuales 3D a los que se acceden a través de una variedad de clientes, en múltiples protocolos. Este permite desarrollar su propio entorno utilizando las tecnologías que mejor se ajusten a su trabajo se ha diseñado el software para ser fácilmente ampliable a través de módulos cargables para construir completamente configuraciones personalizadas. Soporta regiones independientes y múltiples conectadas a un Grid[1] centralizado. Esto es similar a una web donde cualquiera puede correr su propio servidor, a través de internet. También se puede usar para crear grid privado, análogo a una intranet privada. Es liberado bajo una Licencia BSD[2] por lo que su código es abierto, tanto para uso comercial como doméstico. Consta de regiones, grids, avatares, primitivas y su composición da como resultado objetos que conforman las estructuras del mundo.

1.4.2 ¿QUÉ ES UN AVATAR?

Es la simulación del propio usuario. Esta simulación, llamada avatar, es literalmente una representación del usuario en el mundo virtual, que no tiene sólo características corporales humanas, sino también gestos, actitudes e incluso acciones que el humano "convencional" no podría hacer en el mundo real, como volar sin la asistencia de algún aparato.

El usuario puede personalizar su avatar tanto como le sea necesario, o incluso falsificarlo para que no se parezca en lo absoluto al original. Un grupo de herramientas de personificación permite agregar o quitar cabello, cambiar el tipo de cara, ojos y nariz, amoldar el cuerpo del avatar como uno desee y vestirlo con las telas y estilos que el usuario prefiera. Los avatares "viven" en la Internet 3D.

[1] Grid (cuadricula en español), es el nivel que organiza las regiones y sus posiciones en el mundo, y maneja las cosas que necesita para existir la región, como el inventario de usuarios. Puede pensarlo similar al mapa del mundial.

[2] Berkeley Software Distribution: licencia de software libre permisiva, tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre.

1.4.3 ¿QUÉ SE HACE CON OPENSIM?

- ✓ Se observa e identifica el lugar.
- ✓ Se moviliza el avatar y visualiza las opciones del cliente que se use.
- ✓ Se establece contactos con otros.
- ✓ Se configura la apariencia del avatar.
- ✓ Se conocen los diversos espacios para tele-transportarse.
- ✓ Se construyen objetos.
- ✓ Se solicita lugar para habitar y experimentar.

1.4.4 ¿POR QUÉ USAR OPENSIM?

- ✓ Está bajo licencia BSD, código abierto.
- ✓ Ofrece facilidades para configurar un entorno privado (restringido), es atractivo por su parte porque el espacio virtual no implica gasto.
- ✓ Creación de servidores propios.
- ✓ Creación de contenido en tiempo real.
- ✓ Incluye herramientas para personalizar avatares, chatear con otros usuarios, construcción de elementos 3D dentro del entorno.
- ✓ La plataforma muestra un mundo virtual completo que permite a los profesores y estudiantes de distintos lugares trabajar juntos de manera más eficiente y natural.

1.5 BOT

Bot (abreviatura de robot), palabra que parte de la jerga informática, referente a Avatares con funciones programadas para cumplir con tareas específicas de forma autónoma, sin la manipulación de las acciones por un usuario.

Es importante distinguir que bot es una definición funcional, y no hace diferencias en cuanto a su implementación. Un bot puede estar diseñado en cualquier lenguaje de programación, funcionar en un servidor o en un cliente, o ser un agente móvil, entre otros. A veces son llamados Sistemas Expertos pues muchos se especializan en una función específica.

En sitios como Wikipedia, un bot puede realizar funciones rutinarias de edición. En otros sitios, como Encarta, el bot puede responder a cuestiones sobre el propio contenido del sitio (bots conversacionales). Dentro de los metaversos se programan a los avatares para disimiles funciones aras de hacer más real el ambiente.

1.5.1 BOTS EN METAVERSOS

El uso de los bots en los metaversos es tan diverso como la necesidad que se tenga de automatizar un avatar para realizar determinada función ejemplo encontramos en simplemente estar en un sitio para generar concurrencia, como conversar con otros avatares. La mayoría de los ejemplos q existen son de código privativo siendo ejemplos comerciales q grupos y empresas crean principalmente para metaversos privativos como Second Life.
Ejemplo de estos tenemos:

Privativos:

- PikkuBot (3): Es una herramienta para manejar bots para Second Life, con determinadas acciones entre las cuales están obtener dinero en las sillas de camping, se teleporte a ti, se Auto-Siente en los stios de camping entre otros.
- SmartBots (4): Es una herramienta para manenipular bots y grupos de Second Life con una amplia gama de habilidades. En estos casos no se describe información referente a la creación.

Libres:

- **PetBot (bot mascota)** (5)**:** Tiene la función de seguir al dueño a donde valla dentro del metaverso.
- **BotConvensional** (6)**:** Responde a preguntas dentro de una determinada tarea y se controlar un mediante la utilización de la voz.

La forma de crear estos bots no está enfocada a las buenas prácticas de programación como es el uso de patrones.

1.5.2 BOTS EN UCIGRID

En este caso será usado para dar información de productos, sin la necesidad de una persona manipulando el avatar en todo momento, siendo persistente a la par del metaverso. Existiendo dos tipos de bots, bot presentador general y bot producto.

- ✓ **Bot presentador general:** Encargado de dar la introducción a la Feria, mostrando la lista de productos disponibles en la misma, una breve descripción de los mismos y las localizaciones de cada cual.
- ✓ **Bot producto:** Encargado de mostrar la información completa sobre el producto específico.

Para el desarrollo de los mismos se utilizaron varias herramientas y la metodología que a continuación se describen.

1.6 MARCO DE TRABAJO

La solución en cuestión se guiara principalmente para su realización por la metodología Programación Extrema (XP), con la utilización de algunos artefactos generados principalmente en otras metodologías para el desarrollo de software como Rational Unified Process (RUP).

1.6.1 PROGRAMACIÓN EXTREMA

La programación Extrema o XP (por sus siglas en inglés Extreming Programing) es una metodología ágil que se centra principalmente en potenciar las relaciones interpersonales para el éxito durante el desarrollo del software. Se centra en promover el trabajo en equipo preocupándose por el aprendizaje de los desarrolladores. XP se basa en la retroalimentación continua entre cliente y el equipo de trabajo, además de ser apta para proyectos que cuenten con requisitos imprecisos y muy cambiantes. XP le da al cliente el software que necesita en el momento que lo necesita. Presenta características esenciales como son: Historias de Usuario (HU), roles, proceso y por último, prácticas.

Consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo.

Las HU son unas sencilla tarjetas en las cuales el cliente describe brevemente las características que el software a desarrollar debe tener. Estas HU son muy flexibles, son lo suficientemente comprensibles para que los programadores puedan implementarlas en unas pocas semanas.

1.6.1.1 ROLES QUE UTILIZA LA METODOLOGÍA XP:

- ✓ **Programador:** Es el que realiza las estimaciones sobre las historias de usuario, el que define tareas a partir de dichas historias, el encargado de implementar y de realizar las pruebas unitarias.
- ✓ **Cliente:** Es el encargado de escribir las HU, además de asignar la prioridad a cada HU a implementar en cada una de las iteraciones.
- ✓ **Encargado de Pruebas:** Es el encomendado de ayudar al cliente a escribir las pruebas funcionales, además de difundir los resultados al equipo.
- ✓ **Consultor:** Es un miembro externo del equipo con un conocimiento en algún tema necesario para el proyecto, con el objetivo de solucionar algún problema que pueda surgir durante la elaboración del software.
- ✓ **Entrenador:** Verifica las estimaciones realizadas, evalúa el progreso de cada iteración así como la factibilidad de los objetivos con las restricciones de tiempos y recursos presentes.

Se decide seleccionar la metodología ágil XP por el modo en que sus características favorecen la realización de la Herramienta, entre las que resaltan el plazo de entrega del proyecto, producto o software es de un corto plazo. Se hace inminente seleccionar una metodología que permita en un

futuro realizar cambios; XP brinda esa posibilidad ya que es flexible a futuros cambios administrando las modificaciones de forma óptima. Es muy importante destacar que esta metodología es principalmente para proyectos que no tengan una gran envergadura, es decir, para proyectos sencillos como es el caso de la realización de los bots. Por todo lo anteriormente planteado es que se decide seleccionar dicha metodología ya que cumple con los requisitos suficientes y necesarios que ayuden a la realización de la solución.

1.7 HERRAMIENTAS A UTILIZAR.

Las herramientas a utilizar van a estar dirigidas principalmente por las usadas actualmente en el proyecto Escenarios 3D (quien es el cliente de la solución). Estas fueron estudiadas y analizadas por sus características como las más convenientes.

1.7.1 LENGUAJES DE PROGRAMACIÓN

Los bots pueden ser programados tanto del lado del cliente como del lado del servidor:

Para programar del lado del cliente se utiliza:

1.7.1.1 SCRIPT DE SECOND LIFE Y OPENSIM:

El Linden Scripting Language (LSL) es el lenguaje que se utiliza para crear contenido interactivo en Secondlife 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, características de los estados, utiliza variables tipo 3D, así como una variedad de funciones incorporadas para manipular la física y la interacción de los avatares (7).

Para programar del lado del servidor se utiliza:

1.7.1.2 C#

Es un lenguaje de alto nivel, orientado a objetos, potente y fácil de aprender. Este lenguaje facilita la vida a los programadores por su gran capacidad para manipular errores. C#, incorpora las ventajas o mejoras de lenguajes ya existentes tales como C, Java, Visual Basic y C++, lográndose un lenguaje flexible y poderoso. También contiene una librería de clases muy completa y bien diseñada, lo que hace que sea uno de los lenguajes predilectos y más utilizados.

1.7.2 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.

1.7.2.1 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++. (8)

1.7.3 XAMPP

Es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de **X** (para cualquiera de los diferentes sistemas operativos), **A**pache, **M**ySQL, **P**HP, **P**erl.

El programa está liberado bajo la licencia GNU[3] y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y MacOS X.

1.7.4 SISTEMAS DE GESTIÓN DE BASES DE DATOS (SGBD).

Los sistemas de Gestión de Bases de Datos, son aplicaciones que permiten a los usuarios definir, crear y mantener la base de datos y proporciona un acceso controlado a la misma.

1.7.4.1 MYSQL

Está bajo la licencia GPL (***General Public License***). Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. Fue creada por la empresa sueca MySQL AB, que mantiene el derecho de autor del código fuente del servidor SQL, así como también de la marca. Entre las características principales están:

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

Estableciendo relación entre las diferentes tecnologías a usar, se decide utilizar MySql ya que se definió anteriormente como sistema gestor de contenido a usar para la base de datos del servidor

[3] Orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

montado de OpenSim, y ser la recomendada para cualquier uso más allá de la experimentación o pequeñas aplicaciones independientes.

1.7.4.2 DEVART DOTCONNECT FOR MYSQL

Devart DotConnect para MySQL, anteriormente conocido como MyDirect NET, es un proveedor de datos mejorado para MySQL, que se basa en la tecnología ADO.NET para presentar una solución completa para el desarrollo de MySQL de aplicaciones basadas en bases de datos. Como parte de la estructura de base de datos el desarrollo de aplicaciones Devart, dotConnect para MySQL ofrece, tanto la conectividad de alto rendimiento nativo de la base de datos MySQL y un número de herramientas de desarrollo y tecnologías innovadoras. dotConnect para MySQL introduce nuevos enfoques para el diseño de arquitectura de la aplicación, aumenta la productividad, y aprovecha la implementación de aplicaciones de bases de datos.

La versión usada está disponible en (9).

1.7.5 LIBRERÍA OPENMETAVERSE

La librería LibOpenMetaverse es una librería .Net basada en Cliente/servidor, usada para acceder y crear mundos virtuales en 3D. Es compatible con el protocolo de Second Life y puede ser usado para crear clientes y autómatas en cualquier mundo virtual que use el mismo protocolo.

1.7.6 SMART DEVELOPMENT ENVIRONMENT (SDE) DE VISUAL PARADIM (VP) PARA VISUALSTUDIO.

SDE hereda todas las características de gran alcance de la Visual Paradim (VP)-UML y se integra con su IDE favorito a la perfección. Por lo tanto la SDE, acelera todo el modelo- código-despliegue en el proceso de desarrollo de software.

1.7.7 CLIENTES/VISORES

El *Visor* o *Cliente* es un software de navegación 3D que se necesita para conectarse, explorar y comunicarse en el Mundo Virtual. A continuación se hace una breve descripción de algunos de ellos:

- ✓ **Imprudence/Kokua Viewer** es un proyecto de código abierto bajo los términos de la GNU. Su objetivo es el de mejorar en gran medida la utilidad del espectador a través de la participación comunitaria, diseño inteligente, los métodos modernos de desarrollo, y un ambiente a favor del cambio.

- ✓ **RealXtend Viewer** es un navegador basado en el visor Second Life de Linden Labs. Las modificaciones realizadas incluyen procesamiento de OGRE (compatible con mallas 3D de Mundo Virtual) además del procesamiento original y un sistema global de avatar.

- ✓ **Hippo OpenSim Viewer** es un visor de Second Life modificado, dirigida a los usuarios OpenSim. Se permite la construcción hasta una altura de 10.000 metros, prims escala hasta 256x256x256 metros y otras interesantes funciones.

1.8 CONCLUSIONES PARCIALES

Con el estudio del estado del arte se logró conformar un marco teórico que permite la comprensión de decisiones tomadas a lo largo del trabajo para darle cumplimiento a los objetivos planteados. El análisis de las diferentes metodologías, herramientas y lenguajes sugiere que para el desarrollo del presente trabajo la utilización de un marco de trabajo que define XP como metodología de desarrollo de software con algunas adaptaciones. Entre las herramientas estudiadas y teniendo en cuenta las peculiaridades de cada una, Visual Paradigm, Visual Estudio y MySQL poseen las características necesarias para enfrentar el desarrollo de una solución al problema planteado. Se logó identificar que el uso de la biblioteca Openmetaverselib para la configuración de bots en entornos virtuales, podría facilitar el cumplimiento de los objetivos de la presente investigación.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA.

2.1 INTRODUCCIÓN

El presente capítulo se centra un marco de trabajo dirigido principalmente por la metodología Extreme Programing (XP) (10), usada para la realización de la herramienta abordando las fases de Exploración, Planificación de la Entrega, Iteraciones y Pruebas en las cuales se tocaran temas acerca de las necesidades del software y la planificación del proyecto. Se realiza una descripción del flujo actual para así poder conformar una propuesta del sistema. Se muestran las Historias de Usuarios definidas por el cliente, construyéndose también el plan de entrega con la cantidad de iteraciones que va a tener cada uno.

2.2 FLUJO ACTUAL DEL PROCESO

En la actualidad UCIGRID no cuenta con avatares que persistan dentro del mundo durante todo el tiempo en el cual esté activo; dependiendo de la manipulación de una persona tras ellos para interactuar en el mismo, lo que provoca un enorme vacío. Esto hace que los usuarios que entren en momentos donde no exista más ningún avatar online, sientan cierta incomodidad y disminuya el interés por el metaverso. Lo planteado anteriormente constituye un obstáculo para el desarrollo del metaverso.

2.3 PROPUESTA DE SOLUCIÓN

Tras haber analizado el flujo actual de UCIGRID se ha llegado a la conclusión de que es de mucha utilidad e importancia crear unas bases sólidas para la creación de bots para el metaverso en cuestión y los que puedan surgir en un futuro en la universidad utilizando las herramientas analizadas en el capítulo anterior.

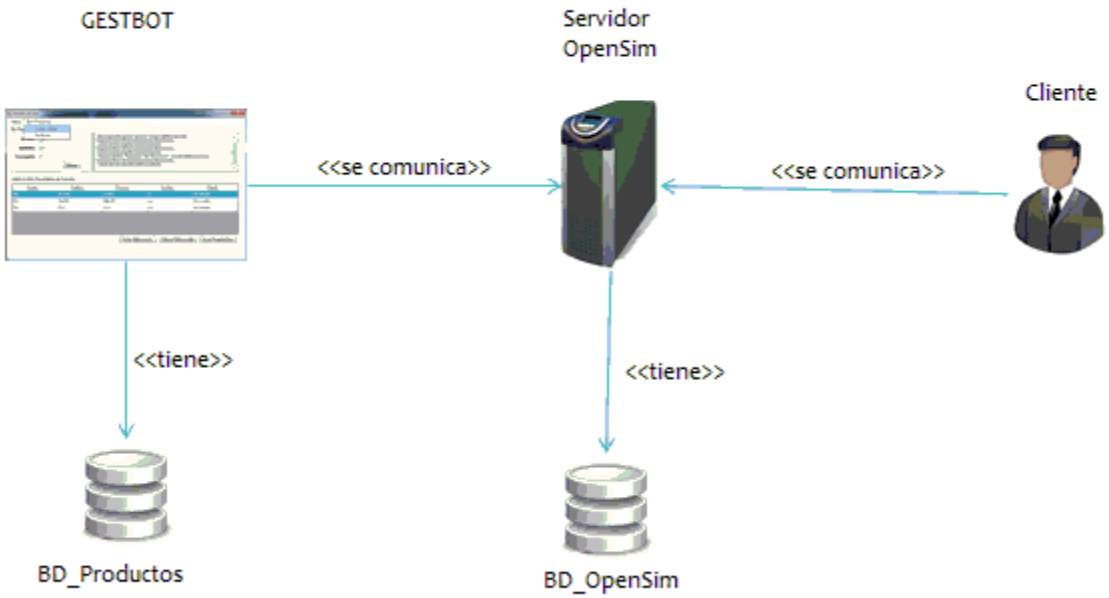
Se propone crear una herramienta para manipular bots que muestren información de productos que en UCIGRID se expongan.

Se dispondrá de un bot principal que será el encargado del saludo, la introducción y mostrar la lista de productos disponibles, mostrando un breve resumen sobre el producto deseado así como la ubicación (X, Y, Z) y brindar la opción de tele-transportación a la ubicación de dicho producto. Además se tendrán tantos bots como productos existan y cada uno encargado de la completa información del producto correspondiente.

Finalmente tras haber implementado cada una de las funcionalidades de gestión (crear, modificar y eliminar) de productos, se obtendrán los bots encargados de dar la información de los mismos y aumentar el nivel de realismo de UCIGRID, permitiendo contar con las bases sólidas para la implementación de futuros bots para disímiles funciones incluyendo animales.

2.4 REPRESENTACIÓN GRÁFICA DEL SISTEMA

FIGURA 1 REPRESENTACION DEL SISTEMA



2.4.1 REQUISITOS NO FUNCIONALES

Los requisitos no funcionales son propiedades o cualidades que el producto debe poseer. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Normalmente están vinculados a requisitos funcionales. A continuación se presentan los definidos para la realización de la aplicación a desarrollar.

Soporte

RNF 1. Utilizar una metodología de desarrollo ágil que permita sacar versiones de manera regular.

RNF 2. Un sistema gestor de base de datos con soporte para grandes volúmenes de datos y alta velocidad de procesamiento. Con un tiempo de respuesta rápido en accesos concurrentes.

Restricciones en el diseño y la implementación

RNF 3. Utilizar programación orientada a objetos.

RNF 4. Lenguajes de programación. Se utilizará el lenguaje C# para programar las acciones del bot.

Usabilidad

RNF 5. Podrá ser utilizada por usuarios que posean avatares creados en los servidores.

RNF 6. La aplicación podrá ser instalada por cualquier persona.

Portabilidad

RNF 7. El código podrá ser compilado para Windows.

Software

RNF 8. Se requiere de un cliente de aplicaciones 3D para entrar al entorno.

RNF 9. Se requiere como sistema gestor de base de datos MySQL.

RNF 10. Se requiere tener el instalador de la aplicación en cuestión (GESTBOT).

Hardware

RNF 11. Para tener un mejor rendimiento se recomienda tener como requerimientos mínimos de sistema:

- Memoria RAM de un 1 Gb o superior.
- Velocidad de Microprocesador a 2.00 GHz o superior.
- Recursos de video de 256 Mb o superior.
- Contar con un almacenamiento en disco de 1Gb.

Apariencia o interfaz externa

RNF 12. Diseño orientado a llamar la atención del usuario y con una navegación lo más sencilla posible.

Seguridad:

RNF 13. Se requiere de tener avatares creados por el administrador del servidor.

RNF 14. Verificación sobre acciones irreversibles. (Cerrar, eliminar).

2.5 PERSONAS RELACIONADAS CON EL SISTEMA.

Cuando se menciona el término de personas relacionadas con el sistema se hace referencia a todo aquel sujeto que de una manera u otra puede acceder a la información: usuario común, administrador, bot presentador general y bot producto.

TABLA 1 PERSONAS RELACIONADAS CON EL SISTEMA.

Personal relacionado con el sistema	Explicación
Administrador	Es la persona encargada de gestionar todas las funcionalidades dentro del metaverso. Administra las cuentas de usuarios, restringe los permisos de los otros usuarios, además puede gestionar la información de los productos.
Bot presentador general	Avatar creado por el Administrador, encargado de brindar la información sobre los productos existentes y dar un breve resumen del seleccionado. Además de dar la localización de dicho producto y ofrecer teleport hacia la misma.
Bot producto	Avatar creado por el Administrador, encargado de brindar toda la información sobre el producto.
Usuario común	Tiene permisos de interacción y construcción en las áreas determinadas por el administrador.

2.6 FASES DE XP

El ciclo de vida ideal de XP consta de 6 fases (Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto).

2.6.1 FASE DE EXPLORACIÓN.

En esta fase se definen las HU (Historias de Usuario) especificándose las características que van a tener cada una de ellas. También el equipo de desarrollo se familiariza con las herramientas y tecnologías a utilizar. Esta fase toma de pocas semanas a pocos meses dependiendo de la familiaridad que tengan los programadores con las tecnologías a utilizar.

2.6.1.1 HISTORIAS DE USUARIO.

Las historias de usuario son escritas por los propios clientes, tal y como quieren ellos que funcione el sistema, no quita que los desarrolladores los puedan ayudar en la identificación de las mismas. Estas emplean la terminología del cliente sin lenguaje técnico. Las historias de usuario se emplean para hacer estimaciones de la parte de la aplicación que se esté desarrollando. El tiempo de desarrollo ideal para cada una de las historias de usuario son de 1 a 3 semanas, si el tiempo estimado dura más de 3 semanas debe dividirse en 2 o más historias de usuarios. A continuación se muestran las HU definidas para el desarrollo de este trabajo:

HU 1: Configurar conexión del Metaverso

HU 2: Configurar conexión a la Base de Datos (BD)

- HU 2.1: Nombre de la BD
- HU 2.2: Usuario administrador de la BD.
- HU 2.3: Contraseña de la BD
- HU 2.4: Dirección de la BD

HU 3: Comunicar bot y usuario.

- HU 3.1: Comunicación bot presentador general
- HU 3.2: Comunicación bot producto
- HU 3.3: Comunicación avatar

HU 4: Autenticar bot presentador general.

HU 5: Gestionar productos

HU 5.1: Adicionar producto

HU 5.2: Modificar producto

HU 5.3: Eliminar producto

HU 6: Listar los bots productos y sus productos asignados.

HU 7: Configurar distancia máxima de escucha

TABLA 2-HISTORIA DE USUARIO CONEXIÓN AL METAVERSO

Historia de Usuario	
Número: 1	**Título:** Configurar conexión al Metaverso
Usuario: Administrador	
Prioridad: Alta.	**Iteración:** 1
Descripción: El sistema debe comunicarse con el servidor de OpenSim.	
Observaciones: El sistema debe permitir que se introduzcan la dirección donde se ubica el servidor del metaverso	
Prototipo de interfaces: TextBox para introducir la dirección.	

TABLA 3 -HISTORIA DE USUARIO CONFIGURAR CONEXIÓN A LA BASE DE DATOS (BD)

Historia de Usuario	
Número: 2	**Título:** Configurar conexión a la BD
Usuario: Administrador	
Prioridad: Alta.	**Iteración:** 1
Descripción: El sistema debe permitir configurar los datos de la BD destinada para los productos. En caso de no existir permitir crearla.	

Observaciones: Esto permite que la BD pueda estar en servidor distinto al del metaverso.
Prototipo de interfaces: Interfaz visual para introducir los datos de la BD: Nombre, Usuario Administrador, contraseña, dirección.

TABLA 4 - HISTORIA DE USUARIO COMUNICACIÓN ENTRE BOT Y AVATAR (BOT PRESENTADOR GENERAL)

Historia de Usuario	
Número: 3.1	**Título:** Comunicación entre Bot y avatar
Usuario: Bot presentador general	
Prioridad: Alta	**Iteración:** 1
Descripción: El bot presentador general cumple con determinadas opciones de conversación, en las que figuran: • Saludar. • Mostrar la lista con los productos existentes. • Dar resumen del producto deseado por el usuario y su localización • Ofrecer teleport al sitio del producto	
Observaciones: El bot debe haber sido autenticado. El usuario debe ser el iniciador de la conversación.	
Prototipo de interfaces: Se realiza a través del cliente / visor.	

TABLA 5- HISTORIA DE USUARIO COMUNICACIÓN (BOT PRODUCTO)

Historia de Usuario	
Número: 3.2	**Título:** Comunicación entre Bot y avatar
Usuario: Bot producto	
Prioridad: Alta	**Iteración:** 1
Descripción: El bot producto cumple con determinadas opciones de conversación, en las que figuran: • Saludar. • Mostrar la descripción del producto que expone.	
Observaciones: El bot debe haber sido autenticado. El usuario debe ser el iniciador de la conversación.	
Prototipo de interfaces: Se realiza a través del cliente / visor.	

TABLA 6- HISTORIA DE USUARIO COMUNICACIÓN (USUARIO)

Historia de Usuario	
Número: 3.3.	**Título:** Comunicación entre Bot y avatar
Usuario: Usuario	
Prioridad: Alta	**Iteración:** 1
Descripción: El sistema debe reconocer las palabras de comunicación: Con el *Bot presentador general:* Puede saludar (*hola*), y si desea saber sobre los productos de la feria, escribe: *productos* (plural), para recibir la lista con los productos disponibles y la localización de los mismos, así como la invitación de teleport al sitio del producto. Con algún *Bot producto*: Puede saludar (*hola*) y si desea saber sobre el producto en cuestión, se escribe: *producto* (singular), para recibir la información completa sobre dicho producto.	

Observaciones: • Deben estar autenticado los bots en el metaverso • Usuario es el iniciador de la conversación a partir de las palabras indicadas: • Saludo: Hola, Adiós. • Información: Productos (refiriéndose al Bot presentador general) • Información: Producto (refiriéndose al Bot producto)
Prototipo de interfaces: Se realiza a través del cliente / visor.

TABLA 7- HISTORIA DE USUARIO AUTENTICAR BOT PRESENTADOR GENERAL

Historia de Usuario	
Número: 4	**Título:** Autenticar Bot presentador general
Usuario: Administrador	
Prioridad: Media	**Iteración:** 2
Descripción: El sistema debe permitir autenticar el avatar que será bot presentador general.	
Observaciones: Este bot es el encargado de dar la introducción a la feria y mostrar el listado de los productos existentes y sus posiciones.	
Prototipo de interfaces: Campos en el componente visual para la autenticación del bot presentador general: Nombre, Apellido, contraseña.	

TABLA 8- HISTORIA DE USUARIO GESTIONAR PRODUCTO

Historia de Usuario	
Número: 5	**Título:** Gestionar producto
Usuario: Administrador	
Prioridad: Media.	**Iteración:** 2

Descripción: El sistema debe permitir gestionar (adicionar, modificar y eliminar) productos
Observaciones: El avatar que será bot, tiene que estar previamente creado.
Prototipo de interfaces: Campos en el componente visual para: Gestión del Producto: Nombre, Facultad, Posición, Resumen, Descripción Bot producto: Nombre, Apellido, Contraseña

TABLA 9- HISTORIA DE USUARIO LISTAR BOTS PRODUCTOS Y SUS PRODUCTOS ASIGNADOS

Historia de Usuario	
Número: 6	**Título:** Listar los bots productos y sus productos asignados
Usuario: Administrador	
Prioridad: Baja.	**Iteración:** 2
Descripción: El sistema mostrar el listado de los bots asignados a cada producto.	
Observaciones: Deben haberse gestionado productos.	
Prototipo de interfaces: Componente visual para mostrar el listado de bots con sus productos respectivos.	

TABLA 10- HISTORIA DE USUARIO CONFIGURACIÓN DE LA DISTANCIA MAX. DE CHAT

Historia de Usuario	
Número: 7	**Título:** Configurar distancia máxima de chat
Usuario: Administrador	
Prioridad: baja	**Iteración:** 3
Descripción: Esta funcionalidad limita el radio de escucha del bot, permitiendo colocar bots cercanos, y que no respondan varios al mismo tiempo.	

Observaciones: La distancia máxima. será 10m
Prototipo de interfaces: Componente para seleccionar la distancia máxima de escucha.

2.6.2 FASE DE PLANIFICACIÓN

En esta fase el artefacto que se construye es la Estimación por Esfuerzo aquí es donde el cliente establece la prioridad de cada una de las HU y es donde los programadores realizan una estimación del esfuerzo que se puede tomar en implementarlas, es importante tener en cuenta los requerimientos del sistema para poder hacer una correcta estimación del tiempo de desarrollo del producto.

La Estimación por Esfuerzo consiste en asignarle puntos a las HU y cada uno de estos puntos son equivalentes a una semana. Los programadores se basan para realizar la estimación en la complejidad que pueden tener las HU y en el tiempo hábil para el desarrollo de la aplicación, para la realización de la herramienta se analizarán cada una de las HU para la correcta estimación.

Para la duración de las semanas que se tienen en cuenta en las estimaciones de las historias de usuario anteriores, es necesario aclarar que una (1) semana equivale a los cinco (5) días laborales de la misma. Se considera válida esta aclaración ya que generalmente, los cálculos erróneos de estimación de los tiempos de desarrollo se realizan en base a los siete (7) días de la semana.

TABLA 11- ESTIMACIÓN DE ESFUERZO.

Mes de ejemplo						
Domingo	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Días feriados.	Fin de semana.	Próximo mes.	Días laborales.

TABLA 12- ESTIMACION DE ESFUERZO

Iteraciones	Orden de las Historias de Usuario a implementar	Cantidad de tiempo de Trabajo (semanas)
1	Configurar conexión al metaverso	2
1	Configurar conexión a la Base de Datos (BD)	1
1	Comunicación bot y usuario	3
2	Autenticar Bot presentador general.	1
2	Gestionar productos	1
3	Listar los Bots productos y sus productos asignados.	1
3	Configurar distancia máxima de escucha	1
Total		**10**

2.6.3 FASE DE ITERACIÓN.

En esta fase se realizan varias iteraciones al sistema antes de ser entregado al cliente, las iteraciones están compuestas por no más de tres semanas. El cliente es el que decide que HU se implementarán en cada una de las iteraciones. Al final de la última iteración el sistema estará listo para entrar en la fase de producción.

Plan de Iteraciones.

Iteración 1:

En esta iteración se implementarán las HU 1, 2.1, 2.2, 2.3 y 3 las cuales son las encargadas de las configuraciones pertinentes para la comunicación del bot con el metaverso y la conexión a las BD.

Iteración 2:

En esta iteración se estarán implementando la HU 4 la cual se encarga de la configuración de autenticación del bot presentador general y la HU 5 encargada de la gestión de la información de los productos asociados a los bots.

Iteración 3:

En esta iteración se estarán implementando las HU 6 encargada de listar los bots existentes y sus productos respectivos y la HU 7 para la configuración de la distancia máxima de escucha de los bots que permite acercar los bots en dependencia del rango de escucha.

2.6.4 FASE DE PLAN DE ENTREGAS

El plan de entregas es el compromiso final del equipo de desarrollo con los clientes. Es una cuestión de vital importancia para el negocio entre ambas partes, ya que la entrega tardía de la solución, repercute notablemente de manera negativa en el desarrollo del producto creando insatisfacción en el cliente. La estimación es uno de los temas más complicados del desarrollo de un proyecto de software y es por ello que resulta fundamental tener bien claros los requerimientos del cliente y el estilo de trabajo del equipo de desarrollo para realizar una entrega de la solución con un máximo de calidad. En el más extremo de los casos, la honestidad debe prevalecer en lugar de la incertidumbre y saber decir cuando se puede terminar el proyecto a tiempo o no. Es mejor ser claros con los clientes que defraudarlos luego de haber malgastado su tiempo y sus recursos. La siguiente tabla muestra el control de versiones que se debe tener al final de cada iteración, que se realizan en la cuarta semana de los meses de febrero, abril y mayo respectivamente.

TABLA 13 - PLAN DE ENTREGA.

Entregas	**Historias de Usuario.**
Entrega 1.	Configurar conexión del Metaverso Configurar conexión a la Base de Datos (BD) Comunicar Bot y usuario.

	○ Comunicación Bot presentador general ○ Comunicación Bot producto ○ Comunicación Avatar
Entrega 2.	Autenticar Bot presentador general. Gestionar Productos
Entrega 3.	Listar los Bots productos y sus productos asignados. Configurar distancia máxima de escucha

2.6.5 FASE DE PRODUCCIÓN

Realizar el diseño de las aplicaciones utilizando la metodología XP no requiere de la representación de diagramas de clases utilizando notación UML (aunque es beneficioso su uso para una mejor descripción y comprensión de las relaciones entre clases y sus posibles métodos); en su lugar esta metodología propone usar las tarjetas **C.R.C** (Contenido, Responsabilidad y Colaboración).

2.6.5.1 PATRONES DE DISEÑO DE SOFTWARE

Una de las cuestiones más complicadas en la Orientación a Objeto es elegir adecuadamente las clases y decidir cómo estas deben interactuar, incluso cuando utilizamos metodologías rápidas como XP y centramos el proceso en el desarrollo continuo. No basta con definir la arquitectura, es necesario además establecer directrices que permitan lograr un sistema bien estructurado, para así construir una solución eficaz. Estas directrices son los llamados Patrones de Diseño de Software.

El uso de patrones, entre otras ventajas proporciona al usuario vistas siempre actualizadas sin que el programador tenga que estar preocupado por esa tarea, además de facilitar el trabajo a los desarrolladores en caso de cambio.

En el presente trabajo se emplearán cinco patrones GRASP[4], estos son:

- Experto
- Creador
- Bajo Acoplamiento

[4] **GRASP** (***G****eneral* ***R****esponsibility* ***A****ssignment* ***S****oftware* ***P****atterns*): Patrones generales para asignar responsabilidades.

- Alta Cohesión
- Controlador

Empleando el patrón *Experto* se garantiza que cada clase del sistema asuma las responsabilidades que le conciernen, según las funcionalidades que se quieren implementar y a partir de la información que posee; por lo que cada clase contendrá la información necesaria para cumplir su responsabilidad. Evidenciándose en las clases: Bot, Cálculos, Animaciones, Cadena, Mensajes y Productos

Al utilizar el patrón *Creador* cada clase instancia y crea las clases la misma para cumplir su responsabilidad, tratando de lograr siempre un *Bajo Acoplamiento* y una *Alta cohesión*. Con el cumplimiento de estos dos últimos patrones se logra que cada clase recurra solamente a las clases que son imprescindibles para su trabajo y tenga asociada las responsabilidades que le corresponden de acuerdo con su comportamiento cumpliendo con este patrón las clases: BotProducto y BotPresentadorGeneral y GestionarProducto. Al emplear el patrón *Controlador*, se estructura el sistema con una clase controladora para cada caso de uso, para que esta se encargue de los eventos y funcionalidades que representan dicho caso de uso, mostrándose en las clases: Configuración, GestionarProducto.

También se apoyó el diseño con la utilización de patrones GoF[5]:

- Singleton

El patrón de diseño Singleton es una clasificación de los *patrones de creación*[6]. Este patrón garantiza que una clase u objeto solo tengan una sola instancia de la misma, de modo que solo exista un punto de acceso global. El uso de este patrón permite el refinamiento en las operaciones y en la representación mediante la especialización por herencia, siguiendo el mismo la clase Bot.

- Chain of Responsability (Cadena de Responsabilidades):

El patrón de diseño Chain of Responsability es una clasificación de los *patrones de comportamiento*[7]. Se encarga de otorgar a más de un objeto la capacidad de atender una petición x, conformando una cadena en la cual cada objeto satisface la petición o pasa a la siguiente. El uso de este patrón aporta una mayor flexibilidad puesto que se puede modificar la capacidad de atender una solicitud,

[5] Gang of Four: Patrones de Diseño en el campo del Diseño Orientado a Objetos más conocidos y usados en la actualidad.

[6] Tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

[7] Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos

modificando la cadena de responsabilidades evidenciado en las clases BotProducto y BotPresentadorGeneral.

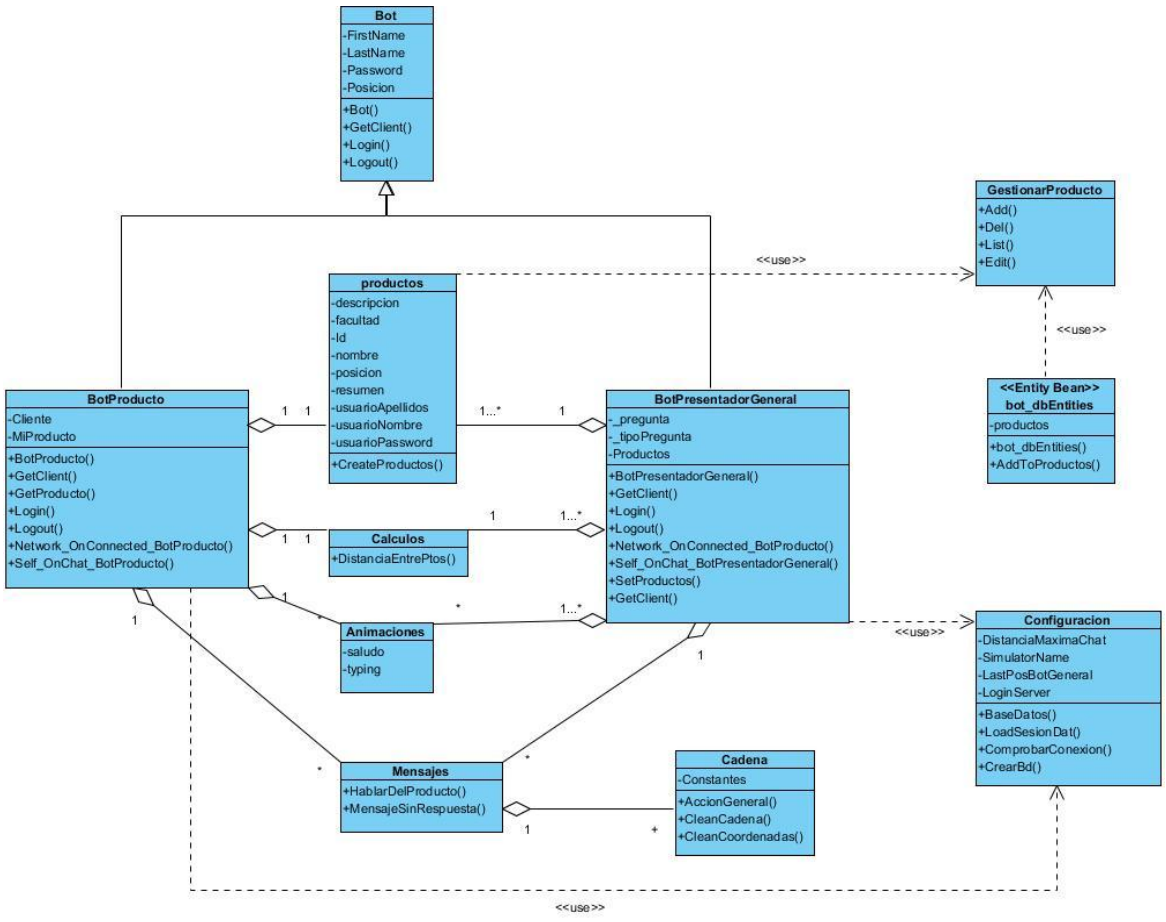
2.6.5.2 UML

Lenguaje Unificado de Modelado (LUM o **UML**, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

A continuación se presenta el diagrama de clases para la solución en cuestión.

FIGURA 2 DIAGRAMA DE CLASES



2.6.5.3 TARJETAS C.R.C.

Son unas simples tarjetas que sustituyen a los diagramas en la representación de modelos. Se comienza la realización del proceso creando dichas tarjetas, inicialmente escribiendo el nombre de ellas, después se irán completando y se sitúan próximas a las que comparten interfaces o llamadas.

Las tarjetas C.R.C permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación clásica. Dichas tarjetas representan objetos, las clases a las que pertenecen dichos objetos. El nombre de la clase se escribe arriba de la tarjeta, en el lado izquierdo en una columna se pueden escribir las responsabilidades y objetivos que deben cumplir el objeto y a la derecha las clases que colaboran con cada responsabilidad.

Una tarjeta CRC representa un objeto. El nombre de la clase se coloca a modo de título en la tarjeta, las responsabilidades se colocan a la izquierda, y las clases que se implican en cada responsabilidad a la derecha, en la misma línea que su requerimiento correspondiente, tal y como muestra la figura.

TABLA 14 - ESTRUCTURA DE LA TARJETA CRC

Nombre de la clase.	
Responsabilidades	Colaboradores

Clase: es cualquier persona, cosa, evento, concepto, pantalla o reporte.

Responsabilidades: las responsabilidades de una clase son sus atributos y métodos.

Colaboradores: los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

Este mecanismo ayuda a que todas las personas participen aportando sus ideas en el diseño, moviendo las tarjetas encima de la mesa según este va progresando

TABLA 15 -PLANTILLA PARA LAS TARJETAS CRC.

Tarjeta CRC	
Clase: Nombre de la clase que se está modelando.	
Súper Clase: Nombre de la clase padre en la herencia.	
Sub Clase(s): Nombre de la(s) clase(s) hija en la herencia.	
Responsabilidades: Es una descripción de alto nivel del propósito de la clase.	**Colaboraciones:** Indica con cuáles otras clases se requiere relación para cumplir la responsabilidad.

TABLA 16-TARJETA CRC BOT.

Tarjeta CRC	
Clase: Bot	
Súper Clase:	
Sub Clase(s): BotProducto, BotPresentadorGeneral	
Responsabilidades: Clase padre con todas las responsabilidades generales de los bots, tanto de los	**Colaboraciones:**

presentadores de productos como el presentador general.	

TABLA 17 - TARJETA CRC BOT PRESENTADOR GENERAL.

Tarjeta CRC	
Clase: BotPesentadorGeneral	
Súper Clase: Bot	
Sub Clase(s):	
Responsabilidades: • Bot presentador general tiene un listado de productos. Y obtiene las características generales de su padre. Bot.	**Colaboraciones:** • Bot • Mensaje • Cálculos • Producto

TABLA 18- TARJETA CRC BOT PRODUCTO.

Tarjeta CRC
Clase: BotProducto
Súper Clase: Bot
Sub Clase(s):

Responsabilidades:	**Colaboraciones:**
Contener todas las clases de Bots productos. Bot presentador de un producto en específico. Es la clase encargada de asignarle un producto a un bot y que el mismo sea capaz de exponer todas las características del producto en cuestión.	• Bot • Producto • Mensaje • Cálculos

TABLA 19 - TARJETA CRC GESTIONAR PRODUCTO.

Tarjeta CRC	
Clase: GestionarProducto	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: • Para la gestión de los productos (CRUD[8]). • Add: Adicionar un nuevo producto en la base de datos. • Edit: Modificar el producto especificado • Del: Eliminar el producto especificado del sistema. • List: Listado de los productos almacenados.	**Colaboraciones:** • Producto • bot_dbEntities

[8]**CRUD** es el acrónimo de Crear, Obtener, Actualizar y Borrar (del original en inglés: **C**reate, **R**ead, **U**pdate and **D**elete). Es usado para referirse a las funciones básicas en bases de datos o la capa de persistencia en un sistema de software.

TABLA 20 TARJETA CRC PRODUCTO

Tarjeta CRC	
Clase: Productos	
Súper Clase: Bot	
Sub Clase(s):	
Responsabilidades: Clase espejo de la entidad producto; como no se pueden modificar los métodos de la entidad, se crea esta clase para poder modificar la forma de obtener los datos de los productos, a la hora de pedir un atributo se puede sobrescribir la forma de hacerlo.	**Colaboraciones:** ✓ Gestionar Producto

TABLA 21- TARJETA CRC ANIMACIONES

Tarjeta CRC	
Clase: Animaciones	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: ✓ Encargada de almacenar las animaciones que puede efectuar el bot.	**Colaboraciones:** ✓ Bot producto ✓ Bot presentador general

TABLA 22- TARJETA CRC CADENA

Tarjeta CRC	
Clase: Cadena	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: ✓ Encargada de la limpieza de las cadenas, es decir, hacer el texto indiferente a mayúsculas o minúsculas y tílde. ✓ Crear constantes para las cadenas deseadas.	**Colaboraciones:** ✓ Mensajes

TABLA 23 - TARJETA CRC CALCULOS

Tarjeta CRC	
Clase: Cálculos	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: Encargada de almacenar los cálculos que se van a realizar en la solución.	**Colaboraciones:** • Bot producto • Bot presentador general

TABLA 24 - TARJETA CRC MENSAJES

Tarjeta CRC	
Clase: Mensajes	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: Contener todas las posibles opciones de conversación que el bot puede tener y las respuestas para cada una de ellas.	**Colaboraciones:** • Bot producto • Bot presentador general

TABLA 25 - TARJETA CRC CONFIGURATION

Tarjeta CRC	
Clase: Configuración	
Súper Clase:	
Sub Clase(s):	
Responsabilidades: Almacenar los datos de la configuración. Como son: • LastPosBotGeneral: Para cargar la última posición en la que se inició el	**Colaboraciones:** • Bot producto • Bot presentador general

bot presentador general. • LoginServer: Dirección del servidor de autenticación para autenticarse en OpenSim. • SimulatorName: Nombre del Metaverso donde van a aparecer los bots. • BaseDeDatos: Parámetros de conexión a la base de datos de donde van a ser cargados los productos que se van a exponer. • DistanciaMaximaChat: A la hora de los bots hablar dentro del metaverso, se determina a que distancia pueden asumir que se está hablando con ellos.	

2.7 CONCLUSIONES PARCIALES

Con la conclusión del capítulo se obtuvo una el flujo actual del proceso referente a los metaversos dentro de la universidad, a partir del cual se pudo conformar una propuesta del sistema, analizándose los requerimientos para el software, así como el marco de desarrollo para el desarrollo de la herramienta, guiado por la metodología XP y las fases de exploración, planificación, iteraciones, plan de entrega y producción con las cuales se logró describir la solución.

CAPÍTULO 3

VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 INTRODUCCIÓN.

La metodología XP plantea que la implementación de un software debe ser iterativa e incremental, y al finalizar debe obtenerse un producto que debe ser probado antes de mostrárselo al cliente y de esa forma mantener la retroalimentación entre el equipo de desarrollo y el cliente. Durante el presente capítulo se lleva a cabo la realización de las tareas generadas por cada HU, así como las pruebas realizadas al sistema.

3.2 IMPLEMENTACIÓN.

Durante la fase de implementación como principal objetivo se implementan las historias de usuarios en cada una de las iteraciones correspondientes. Debido a esto se lleva a cabo el plan de iteraciones y se modifican en caso de ser necesario. Como parte de este plan se crean tareas para ayudar a organizar la implementación de cada una de las historias de usuarios. Teniendo en cuenta la planificación realizada con anterioridad se desarrollaron tres iteraciones, para finalmente obtener un producto que esté de acuerdo con las exigencias del cliente al terminar la última iteración. A continuación se detallan cada una de las iteraciones:

3.2.1 ITERACIÓN 1.

La primera iteración tendrá como objetivo darle cumplimiento a las HU 1, 2, 3 que representan un mayor valor para el cliente, pues con las mismas se conformará la base de la estructura del negocio. Estas recogen funcionalidades de gran importancia para el proyecto, pues a través de ellas se definen aspectos que serán utilizados luego por las demás funcionalidades como son la conexión al metaverso, la comunicación entre los bots y los avatares y la conexión a las respectivas bases de datos (productos y avatares).

Para ello se trazaron las tareas que se indican a continuación:

- Tarea No1: Análisis e instalación de la librería OpenMetaverse para la utilización de los métodos necesarios.
- Tarea No2: Implementación de las clases necesarias para la configuración de conexión con el Metaverso.
- Tarea No3: Implementación de las clases necesarias para lograr la conexión con las BD del metaverso y de productos.
- Tarea No4: Implementación de las clases necesarias para lograr comunicar los bots y los avatares.
- Tarea No5: Implementar componente visual para la introducción de los datos necesarios para la Dirección del metaverso.
- Tarea No6: Implementar componente visual para la introducción de los datos necesarios para la Dirección de la BD de los productos. (Usuario y contraseña administrador de la BD de productos.)

TABLA 26 - TAREA 1

Tarea	
Número de tarea: 1	**Número de HU**:3
Nombre de la tarea: Análisis e instalación de la librería OpenMetaverse para la utilización de los métodos necesarios.	
Tipo de tarea: Instalación	**Estimación:** 10 días
Descripción: Analizar el modo de instalación y las funcionalidades más favorables para la utilización de la librería OpenMetaverse para la solución en cuestión.	

Lo más significativo en la realización de la tarea es la vital importancia de tras instalar la librería openmetaverselib (versión 0.7), añadir las referencias:

OpenMetaverse.dll: Es el núcleo de la biblioteca OpenMetaverse, que puede ser usado para crear aplicaciones clientes y servidor.

OpenMetaverseTypes.dll: Funciones específicas 3D y bibliotecas matemáticas.

OpenMetaverse.StructuredData.dll: Conjunto de librerías para dar un completo soporte a (Linden Lab Structured Data (LLSD) y JSON (JavaScript Object Notation).

TABLA 27 - TAREA 2

Tarea	
Número de tarea: 2	**Número de HU:** 1
Nombre de la tarea: Implementación de las clases necesarias para la configuración de conexión con el Metaverso.	
Tipo de tarea: Implementación	**Estimación:** 5 días
Descripción: Implementar las clases necesarias para lograr conectar la solución con un metaverso.	

En esta tarea los métodos más significativos son:

FIGURA 3- LOGIN SERVER. CLASS CONFIGURATION

```
/// <summary>
/// Salvar la dirección del server de autenticación
/// </summary>
public static string LoginServer
{
    get
    {
        return Session.Instance.Get("LoginServer").ToString();
    }
    set
    {
        Session.Instance.Set("LoginServer", value);
        var config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
        config.AppSettings.Settings.Remove("LoginServer");
        config.AppSettings.Settings.Add("LoginServer", value);
        config.Save(ConfigurationSaveMode.Full);
    }
}
```

FIGURA 4- SIMULATOR NAME. CLASS CONFIGURATION

```
/// <summary>
/// Nombre del simulador o región donde aparecera el bot
/// </summary>
public static string SimulatorName
{
    get
    {
        return Session.Instance.Get("SimulatorName").ToString();
    }
    set
    {
        Session.Instance.Set("SimulatorName", value);
        var config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
        config.AppSettings.Settings.Remove("SimulatorName");
        config.AppSettings.Settings.Add("SimulatorName", value);
        config.Save(ConfigurationSaveMode.Full);
    }
}
```

Para que cada bot se conecte con el metaverso se usa:

FIGURA 5- LOGIN BOTS

```
/// <summary>
/// Para iniciar al bot dentro del Metaverso
/// </summary>
/// <returns></returns>
public override bool Login()
{
    Client1 = new GridClient
    {
        Settings = { LOGIN_SERVER = Session.Instance.Get("LoginServer").ToString() }
    };
    Client1.Network.OnConnected += Network_OnConnected_BotProducto1;
    var startLocation = NetworkManager.StartLocation(Configuration.SimulatorName, (int)Posicion.X, (int)Posicion.Y, (int)Posicion.Z);
    if (!Client1.Network.Login(FirstName, LastName, Password, MiProducto1.Nombre, startLocation, MiProducto1.Id.ToString()))
    {
        return false;
    }
    Client1.Appearance.SetPreviousAppearance(true);
    Client1.Self.OnChat += Self_OnChat_BotProducto1;
    return true;
}
```

Este método es el mismo para todos los bots, teniendo en el cambio de los nombres de los métodos:

- Network_OnConnected_**XXXX**, que envía un mensaje público cuando se conecta.
- Self_OnChat_**XXXX**, para acceder al método de chat, que varia en dependencia de si es **bot presentador general** o **bot producto**.

TABLA 28 - TAREA 3

Tarea	
Número de tarea: 4	**Número de HU:** 2
• **Nombre de la tarea:** Implementación de las clases necesarias para lograr la conexión con las BD del metaverso y la de productos.	
Tipo de tarea: Implementación	**Estimación:** 4 días
Descripción: Implementar las clases necesarias para conectar la solución con las BD de avatares y de los productos que serán asignados a cada bot. Teniendo en cuenta que si no existe la BD, brinde la opción de crearla.	

En esta tarea los métodos más significativos son:

FIGURA 6- COMPROBAR CONEXIÓN A BD PRODUCTOS. CLASS CONFIGURATION

```
/// <summary>
/// Comprueba la conexión con la base de datos
/// Posibles resultados:
/// 0 - No existe la base de datos
/// 1 - Si existe la base de datos
/// 2 - No se puede autenticar con el servidor
/// </summary>
/// <returns></returns>
public static int ComprobarConexion()
{
    try
    {
        using (var db = new bot_dbEntities())
        {
            try
            {
                if (db.DatabaseExists())
                {
                    db.DetectChanges();
                    return 1;
                }
            }
            catch (Exception)
            {
                return 0;
            }
        }
        return 2;
    }
    catch (Exception)
    {
        return 2;
    }
}
```

FIGURA 7 BD_CONECTION. CLASS CONFIGURATION

```
/// <summary>
/// Nombre de la base de datos
/// </summary>
public static void BaseDeDatos(string pserver, string pbd, string puser, string ppass)
{
    var config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
    string server = pserver, bd = pbd, user = puser, pass = ppass;

    //APPConfig XML
    config.AppSettings.Settings.Remove("server");
    config.AppSettings.Settings.Add("server", server);

    config.AppSettings.Settings.Remove("bd");
    config.AppSettings.Settings.Add("bd", bd);

    config.AppSettings.Settings.Remove("user");
    config.AppSettings.Settings.Add("user", user);

    config.AppSettings.Settings.Remove("pass");
    config.AppSettings.Settings.Add("pass", pass);

    Session.Instance.Set("server", server);
    Session.Instance.Set("bd", bd);
    Session.Instance.Set("user",user);
    Session.Instance.Set("pass", pass);

    var auxDb = "metadata=res://*/Modelo.Model.csdl|res://*/Modelo.Model.ssdl|
                        res://*/Modelo.Model.msl;provider=Devart.Data.MySql;
                        provider connection string=';User Id="+user+";Host="+server+";I
                        Database="+bd+";Password="+pass+";Persist Security Info=True';";

    var a = new ConnectionStringSettings("bot_dbEntities", auxDb, "System.Data.EntityClient");
    config.ConnectionStrings.ConnectionStrings.Clear();
    config.ConnectionStrings.ConnectionStrings.Add(a);

    config.Save(ConfigurationSaveMode.Full);
    ConfigurationManager.RefreshSection("connectionStrings");
}
```

Este método es usado principalmente para la configuración de la base de datos de los productos.

FIGURA 8- CREAR BD_PRODUCTOS. CLASS CONFIGURATION

```
/// <summary>
/// Para crear la base de datos configurada en la conexión,
/// en caso de que no se pueda crear porque hay problemas con
/// la conexión devuelve false
/// </summary>
/// <returns></returns>
public static bool CrearBd()
{
    using (var db = new bot_dbEntities())
    {
        try
        {
            if (!db.DatabaseExists())
            {
                db.CreateDatabase();
                db.CreateDatabaseScript();
                return true;
            }
        }
        catch (Exception)
        {
            try
            {
                db.CreateDatabase();
                db.CreateDatabaseScript();
                return true;
            }
            catch (Exception)
            {
            }
            return false;
        }
    }
    return false;
}
```

TABLA 28 TAREA 4

Tarea	
Número de tarea: 4	**Número de HU:** 3
Nombre de la tarea: Implementación de las clases necesarias para lograr comunicar los bots y los avatares.	
Tipo de tarea: Implementación	**Estimación:** 10 días
Descripción: Implementar las clases necesarias para lograr el entendimiento entre bots y avatares.	

Las palabras reconocibles y las acciones a ejecutar para cada una de estas. Ya sea para bot presentador general como para bot presentador de producto. Destacar que el bot presentador general debe dar una lista de los productos existentes, mostrar el resumen del producto que el usuario seleccione y la posición respectiva del bot presentador del mismo, quien dispondrá de la descripción total de dicho producto.

En esta tarea los métodos más significativos son:

FIGURA 9- CHAT BOT PRODUCTO. CLASS BOTPRODUCTO

```
/// <summary>
/// Para responder las preguntas que formulen en el chat
/// </summary>
/// <param name="message"></param>
/// <param name="audible"></param>
/// <param name="type"></param>
/// <param name="sourceType"></param>
/// <param name="fromName"></param>
/// <param name="id"></param>
/// <param name="ownerid"></param>
/// <param name="position"></param>
static void Self_OnChat_BotProducto1(string message, ChatAudibleLevel audible, ChatType type, ChatSourceType sourceType,
                                                                    string fromName, UUID id, UUID ownerid, Vector3 position)
{
    if (message.Length == 0 || type == ChatType.StartTyping || type == ChatType.StopTyping ||
           Calculos.DistanciaEntrePuntos(Client1.Self.SimPosition, position) > Configuration.DistanciaMaximaChat)

        return;

    var mensajeMinuscula = Cadena.CleanCadena(message);
    if (Cadena.AccionGeneral(mensajeMinuscula))
    {
        Mensajes.MensajeSinRespuesta(Client1, mensajeMinuscula, fromName, position);
        return;
    }

    switch (mensajeMinuscula)
    {
        case Cadena.Producto:
            {
                Mensajes.HablarDelProducto(Client1, MiProducto1);
            } break;
    }
}
```

Este método es el mismo para cada bot producto, variando indiscutiblemente los nombres de los parámetros.

FIGURA 10-CHAT BOT PRESENTADOR GENERAL. CLASS BOT PRESENTADOR GENERAL

```
/// <summary>
/// Para responder las preguntas que formulen en el chat
/// </summary>
/// <param name="message"></param>
/// <param name="audible"></param>
/// <param name="type"></param>
/// <param name="sourceType"></param>
/// <param name="fromName"></param>
/// <param name="id"></param>
/// <param name="ownerid"></param>
/// <param name="position"></param>
static void Self_OnChat_BotPresentadorGeneral(string message, ChatAudibleLevel audible, ChatType type, ChatSourceType sourceType,
                                                string fromName, UUID id, UUID ownerid, Vector3 position)

{
    if (message.Length == 0 || type == ChatType.StartTyping || type == ChatType.StopTyping ||
                            Calculos.DistanciaEntrePuntos(Client.Self.SimPosition, position) > Configuration.DistanciaMaximaChat)
        return;

    var mensajeMinuscula = Cadena.CleanCadena(message);
    if (Cadena.CleanCadena(Cadena.NombreIdProducto) == Cadena.CleanCadena(mensajeMinuscula)) return;

    // preguntó si quería saber algo más acerca de algún producto en especifico y la respuesta fue sí
    if (_pregunta && _tipoPregunta == 1 && mensajeMinuscula == Cadena.Si)
    {
        Client.Self.AnimationStart(Animaciones.Typing, true);
        Client.Self.Chat(Cadena.NombreIdProducto, 0, ChatType.Normal);
        Client.Self.AnimationStop(Animaciones.Typing, true);
        _pregunta = true;
        _tipoPregunta = 2; // significa que espera o un nombre del producto o el id
        return;
    }
    // preguntеó si quería saber algo más acerca de algún producto en especifico y dijo que no
    if (_pregunta && _tipoPregunta == 1 && mensajeMinuscula == "no")
    {
        _tipoPregunta = -1;
        _pregunta = false;
        return;
    }
    // había preguntado primero si queria saber algo más de algún producto y la respuesta fue sí y
    // se le preguntó el nombre del producto o el id
    if (_pregunta && _tipoPregunta == 2)
    {
        _tipoPregunta = -1;
        _pregunta = false;
        // la respuesta que dió fue que realizara una acción
        if (Cadena.AccionGeneral(mensajeMinuscula))
        {
            Client.Self.AnimationStart(Animaciones.Typing, true);
            Client.Self.Chat(Cadena.NoAccionComoRespuesta, 0, ChatType.Normal);
            Client.Self.AnimationStop(Animaciones.Typing, true);
            return;
        }
        Mensajes.InfoProductos(Client, Productos, mensajeMinuscula,id,position);
        return;
    }

    if (Cadena.AccionGeneral(mensajeMinuscula))
    {
        Mensajes.MensajeSinRespuesta(Client, mensajeMinuscula, fromName,position);
        return;
    }

    switch (mensajeMinuscula)
    {
        case Cadena.Productos:
            {
                Client.Self.AnimationStart(Animaciones.Typing, true);
                var result = Productos.Count > 0
                                ? ""
                                : "Disculpe, todavía no tengo la lista de productos de la exposición.";
                result = Productos.Aggregate(result,
                                             (current, t) =>
                                             current +
                                             (t.Id + ": \"" + t.Nombre + "\"" + "\n"));
                Client.Self.Chat(result, 0, ChatType.Whisper);

                if (Productos.Count > 0)
                {
                    Client.Self.Chat("¿Desea saber algo más acerca de un producto? ¿Sí o No?", 0,
                                     ChatType.Whisper);

                    _pregunta = true;
                    _tipoPregunta = 1; // respuesta si o no
                }
                Client.Self.AnimationStop(Animaciones.Typing, false);
            }
            break;
    }
}
```

FIGURA 11 - MENSAJES SIN INFORMACION DE PRODUCTO. CLASS MENSAJE

```
/// <summary>
/// Utilizado por los bots para escribir mensajes que no esperan respuestas.
/// Lo utiliza tanto los bots presentadores de productos como que el bot presentador general
/// </summary>
/// <param name="gridClient">Instancia del GridClient del bot que llama la función</param>
/// <param name="mensaje">Mensaje que dijo el bot de OpenSim</param>
/// <param name="fromName">Nombre del bot que envio el mensaje</param>
/// <param name="posAvatar">Posición del avatar con el que se conversa</param>
public static void MensajeSinRespuesta(GridClient gridClient, string mensaje, string fromName,Vector3 posAvatar)
{
    gridClient.Self.Movement.TurnToward(posAvatar);
    switch (mensaje)
    {
        case Cadena.Hola:
            {
                gridClient.Self.AnimationStart(Animaciones.Typing, true);
                System.Threading.Thread.Sleep(1500);
                gridClient.Self.Chat("Hola Sr(a). " + fromName, 0, ChatType.Normal);
                gridClient.Self.AnimationStop(Animaciones.Typing, true);
            }
            break;
        case Cadena.Adios:
            {
                gridClient.Self.AnimationStart(Animaciones.Saludo, true);
                System.Threading.Thread.Sleep(1500);
                gridClient.Self.Chat("Adiós Sr(a). " + fromName, 0, ChatType.Normal);
                gridClient.Self.AnimationStop(Animaciones.Typing, true);
            }
            break;
        case Cadena.Volar:
            {
                gridClient.Self.Fly(true);
            }
            break;
        case Cadena.DejarDeVolar:
            {
                gridClient.Self.Fly(false);
            }
            break;
        case Cadena.Levantarse:
            {
                gridClient.Self.Stand();
            }
            break;
        case Cadena.Sentarse:
            {
                gridClient.Self.SitOnGround();
            }
            break;
    }
}
```

Este método es usado por los bots para la interacción no referente a los productos como es el saludo y algunas acciones como sentarse y volar.

FIGURA 12- INFORMACION DE PRODUCTOS Y BOTS. CLASS MENSAJE

```
/// <summary>
/// Hablar acerca del producto que me especifica en nombre o id del producto
/// </summary>
/// <param name="gridClient">Instancia del GridClient del bot que llama la función</param>
/// <param name="productos">Listado de produtos que presenta el bot</param>
/// <param name="idNombreProduto">Nombre o identificador del producto del que desea que el bot hable</param>
/// <param name="idAvatar">Identificador del avatar con el que se conversa</param>
/// <param name="posAvatar">Posición del avatar con el que se conversa</param>
public static void InfoProductos(GridClient gridClient, List<MProducto> productos, string idNombreProduto,UUID idAvatar,Vector3 posAvatar)
{
    gridClient.Self.Movement.TurnToward(posAvatar);
    bool existe = false, esId = false;
    var vector3 = new Vector3();
    try
    {
        var auxId = int.Parse(idNombreProduto);
        foreach (var p in productos.Where(p => p.Id == auxId))
        {
            HablarDelProducto(gridClient, p,2);
            existe = true;
            vector3 = p.PosicionStand;
            break;
        }
        esId = true;
    }
    catch (Exception)
    {
    }
    if (!esId)
    {
        foreach (var p in productos.Where(p => Cadena.CleanCadena(p.Nombre) == idNombreProduto))
        {
            HablarDelProducto(gridClient, p,2);
            vector3 = p.PosicionStand;
            existe = true;
            break;
        }
    }
    if (existe)
    {
        gridClient.Self.AnimationStart(Animaciones.Typing, true);
        System.Threading.Thread.Sleep(1500);
        var listadoAux = (List<Bot>)Session.Instance.Get("BotsPresentadoresProducto");
        foreach (var t in
            listadoAux.Where(t => (esId && t.GetProducto().Id == int.Parse(idNombreProduto)) ||
                                  (!esId && Cadena.CleanCadena(t.GetProducto().Nombre) == idNombreProduto)))
        {
            if (t.GetClient() != null)
            {
                gridClient.Self.Chat(string.Format("El presentador del producto se encuentra disponible en
                                    la posición {0}.", Cadena.CleanCoordenadas(vector3)), 0, ChatType.Normal);
                t.GetClient().Self.SendTeleportLure(idAvatar, "¿Desea teletransportarse al punto de exposición?");
            }
            else
            {
                gridClient.Self.Chat(string.Format("El presentador del producto no se encuentra disponible.
                                    Su posición es {0}.", Cadena.CleanCoordenadas(vector3)), 0, ChatType.Normal);
            }
        }
        gridClient.Self.AnimationStop(Animaciones.Typing, true);
        return;
    }
    gridClient.Self.AnimationStart(Animaciones.Typing, true);
    System.Threading.Thread.Sleep(1500);
    gridClient.Self.Chat("Disculpe ese producto no lo tengo registrado.", 0, ChatType.Normal);
    gridClient.Self.AnimationStop(Animaciones.Typing, true);
}
```

Este método es usado exclusivamente por el bot presentador general en su comunicación con algún avatar.

FIGURA 13- HABLAR DEL PRODUCTOS Y BOTS. CLASS MENSAJE

```
/// <summary>
/// Metodo utilizado por los bots para hablar hacerca del producto
/// </summary>
/// <param name="gridClient">Instancia del GridClient del bot que llama la función</param>
/// <param name="producto">Producto del cual se desea hablar</param>
///  <param name="tipoExposicion">Cuando el tipo de exposicion es 1 significa que es decir la descripcion completa,
///  si es 2 se dice solo el resumen</param>
public static void HablarDelProducto(GridClient gridClient, MProducto producto, int tipoExposicion = 1)
{
    gridClient.Self.AnimationStart(Animaciones.Typing, true);
    System.Threading.Thread.Sleep(1500);
    gridClient.Self.Chat(producto.Nombre + ": " + (tipoExposicion == 1 ? producto.Descripcion : producto.Resumen), 0,
                         ChatType.Whisper);
    gridClient.Self.AnimationStop(Animaciones.Typing, true);
    gridClient.Self.Chat("El resto de la informacion la encontrara en nuestro STAND",0,ChatType.Whisper);
}
```

Este método se utiliza por los bots. En caso del *bot presentador general* el tipoExposicion = 2 pues da el resumen del producto del cual el avatar quiere conocer, y los *bot producto* muestran la descripción del producto.

TABLA 29 TAREA 5

Tarea	
Número de tarea: 5	**Número de HU:** 1
Nombre de la tarea: Implementación de componentes visuales para la introducción de los datos necesarios en las tareas anteriores: Dirección del metaverso, Dirección de la BD de los productos. Usuario y contraseña administrador de la BD de productos.	
Tipo de tarea: Implementación	**Estimación:** ½ día
Descripción: Implementar los componentes visuales que permitirán introducir los datos necesarios para lograr la conexión al metaverso. (Dirección y Nombre)	

TABLA 30 TAREA 6

Tarea	
Número de tarea: 6	**Número de HU:** 2
Nombre de la tarea: Implementación de componentes visuales para la introducción de los datos necesarios para la Dirección de la BD de los productos. (Usuario y contraseña administrador de la BD de productos.)	
Tipo de tarea: Implementación	**Estimación:** ½ día
Descripción: Implementar los componentes visuales que permitirán introducir los datos necesarios para lograr la conexión con la BD (dirección) donde se almacenaran los productos que se gestionen, así como los datos (usuario y contraseña) de administración de la misma, para poder manipularla.	

3.2.2 ITERACIÓN 2.

La segunda iteración tendrá como objetivo darle cumplimiento a las HU 4 y 5 que representan un valor medio para el cliente, pues con las mismas se va dando forma a la estructura final y validan cuestiones de selección. Recogen funcionalidades de autenticación de los avatares que funcionaran como bots. Además de la adición, modificación y eliminación de la información referente a los productos.

Para ello se debe dar cumplimiento a las tareas, que se indican a continuación:

- Tarea No1: Implementar componente visual para introducir los datos del Bot Presentador General.
- Tarea No2: Implementar componente visual gestionar los productos.

TABLA 31 TAREA 1

Tarea	
Número de tarea: 1	**Número de HU:** 4
Nombre de la tarea: Implementar componente visual para introducir los datos del Bot Presentador General	
Tipo de tarea: Implementación	**Estimación:** 3 días
Descripción: Implementar componente visual para introducir los datos del avatar que será el bot	

presentador general. (Nombre, Apellido, contraseña).

TABLA 32 TAREA 2

Tarea	
Número de tarea: 2	**Número de HU:** 5
Nombre de la tarea: Implementar componente visual para gestionar los productos.	
Tipo de tarea:	**Estimación: 7**
Descripción: Implementar los componentes visuales que permitan la gestión de los productos y los datos del avatar que será el Bot producto, del mismo. Del Avatar: Nombre, Apellido, contraseña Del Producto: Nombre, Facultad, Resumen, Descripción, Posición	

3.2.3 ITERACIÓN 3.

La tercera iteración tendrá como objetivo darle cumplimiento a las HU 6 y 7 que representan un valor bajo para el cliente, las mismas son para perfeccionar y acomodar las acciones. Recogen funcionalidades de información y para lograr ubicar lo más cerca posible un bot de otro sin que interfieran en las conversaciones.

Para ello se debe dar cumplimiento a las tareas, que se indican a continuación:

- Tarea No1: Implementar componente visual mostrar el listado de BotsPresentadores y sus productos asignados.
- Tarea No2: Implementar componente visual para elegir la distancia máx. de escucha de los bots.

TABLA 33 TAREA 1

Tarea	
Número de tarea: 1	**Número de HU:** 6
Nombre de la tarea: Implementar componente visual mostrar el listado de BotsPresentadores y sus productos asignados.	
Tipo de tarea: Implementación	**Estimación:** 5 días
Descripción: Implementar componente visual para visualizar el listado de Bot productoes, con sus productos respectivos y poder elegir cuál y en qué momento inicializarlo dentro del metaverso.	

TABLA 34 TAREA 2

Tarea	
Número de tarea: 2	**Número de HU:** 7
Nombre de la tarea: Implementar componente visual para elegir la distancia máx. de escucha de los bots.	
Tipo de tarea: Implementación	**Estimación:** 5 días
Descripción: Implementar componente visual para elegir la distancia máx. de escucha de los bots. Para permitir ubicar, mínimo, dicha distancia + 1, a un bot de otro sin que interactúen juntos con un mismo avatar	

3.3 PRUEBAS

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación (11). Tienen como objetivo principal el descubrir errores en el software realizado, el éxito de dichas pruebas está en encontrar alguna deficiencia al producto que se esté desarrollando, estas deben planificarse mucho antes de que comiencen a realizarse y deben comenzar desde lo pequeño y progresan hasta lo más grande.

3.3.1 PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación son realizadas una vez terminado e integrado el producto que se esté llevando a cabo, pero lo que la diferencia de las demás pruebas que se le realiza a un software es que estas están concebidas para que sea el cliente o usuario final quien detecte los posibles errores que pueda presentar el producto. Estas pruebas están enfocadas principalmente a probar las historias de usuario, es decir a demostrar que dichas historias de usuarios no cumplen con lo acordado con el cliente.

Destacar que se toma como **resultado satisfactorio** en la **evaluación de la prueba**, los casos donde el **resultado esperado** y **el resultado de la evaluación** sean el mismo.

A continuación, aparecen las pruebas de aceptación realizadas a la solución propuesta:

TABLA 35 CASO PRUEBA

Caso de Prueba de Aceptación
Código: P1_H1
Nombre HU: Configurar conexión al Metaverso
Descripción Prueba: Se introduce la dirección del metaverso en el componente visual.
Condiciones de Ejecución: ✓ Debe estar corriendo el servidor del metaverso.
Entrada/Pasos de ejecución: • Se accede a la función mediante el Menú, submenú Configuración. • Llenar los valores correctos de la Dirección y el Nombre del Metaverso
Resultado esperado: Datos llenados correctamente
Evaluación de la prueba: Resultado satisfactorio.

TABLA 36 CASO PRUEBA

Caso de Prueba de Aceptación
Código: P1_H2
Nombre: Configurar conexión a BD de productos
Descripción: Probar el correcto funcionamiento de la conexión y creación de BD para la gestión de los productos.
Condiciones de Ejecución: • Debe estar corriendo el servidor del metaverso y el servidor de BD.
Entrada/Pasos de ejecución: • Se accede a la función mediante el Menú, submenú Configuración. • Llenar los valores correctos del nombre BD productos, usuario administrador de la BD, contraseña del usuario administrador y dirección de la BD.
Resultado esperado: Datos llenados correctamente
Evaluación de la prueba: Resultado satisfactorio.

TABLA 37 CASO PRUEBA

Caso de Prueba de Aceptación
Código: P1_H3
Nombre: Comunicación Bot presentador general y Avatar
Descripción: Probar el correcto entendimiento de las palabras.
Condiciones de Ejecución: ✓ Corriendo el servidor del metaverso y el servidor de BD. ✓ Estén autenticados el bot presentador general y el Avatar (usuario).

Entrada/Pasos de ejecución: • Usuario saluda • Usuario escribe productos (indistinto a mayúsculas o minúsculas) (PLURAL)
Resultado esperado: ✓ Bot responde: Hola Sr “NombreAvatar” ✓ Bot muestra listado de productos disponibles ✓ Bot pregunta si se quiere conocer sobre alguno especifico ✓ En caso de seleccionar alguno: Muestra Resumen del mismo y da la ubicación además de ofrecer teleport a la misma.
Evaluación de la prueba: Resultado satisfactorio.

TABLA 38 CASO PRUEBA

Caso de Prueba de Aceptación
Código: P2_H3
Nombre: Comunicación Bot producto y Avatar
Descripción: Probar el correcto entendimiento de las palabras.
Condiciones de Ejecución: ✓ Corriendo el servidor del metaverso y el servidor de BD. ✓ Estén autenticados el Bot producto y el Avatar(usuario).
Entrada/Pasos de ejecución: ✓ Usuario saluda ✓ Usuario escribe producto (indistinto a mayúsculas o minúsculas) (SINGULAR)
Resultado esperado: ✓ Bot responde: Hola Sr “NombreAvatar”

✓ Muestra Descripción del producto
Evaluación de la prueba: Resultado satisfactorio.

TABLA 39 CASO PRUEBA

Caso de Prueba de Aceptación
Código: P1_H4
Nombre: Autenticar Bot presentador general
Descripción: Autenticar correctamente un bot
Condiciones de Ejecución: ✓ Bot existente en la BD del servidor ✓ Datos de configuración correctamente
Entrada/Pasos de ejecución: • Autenticar avatar para bot general: Nombre, Apellido, contraseña.
Resultado esperado: ✓ Corriendo el servidor del metaverso y el servidor de BD. ✓ Iniciado Bot presentador general correctamente
Evaluación de la prueba: Resultado satisfactorio.

TABLA 40 CASO PRUEBA

Caso de Prueba de Aceptación
Código: P1_H5
Nombre: Gestionar Productos
Descripción: Gestión (adicionar, modificar, eliminar) de productos y el bot asociado a cada producto.
Condiciones de Ejecución: ✓ Corriendo el servidor del metaverso y el servidor de BD. ✓ Datos de configuración correctamente
Entrada/Pasos de ejecución: ✓ Datos del bot que se asociara al producto ✓ Llenar datos para el producto
Resultado esperado: ✓ Producto gestionado satisfactoriamente
Evaluación de la prueba: Resultado satisfactorio.

TABLA 41 CASO PRUEBA

Caso de Prueba de Aceptación
Código: P1_H6
Nombre: Listar los Bots Presentadores y sus productos asignados.

Descripción: Mostrar listado de los productos existentes y el bot asociado a cada uno de ellos en un componente visual.
Condiciones de Ejecución: ✓ Datos de configuración correctamente ✓ Corriendo el servidor del metaverso y el servidor de BD.
Entrada/Pasos de ejecución: ✓ Datos del bot que se asociará al producto ✓ Llenar datos para el producto
Resultado esperado: ✓ Producto gestionado satisfactoriamente
Evaluación de la prueba: Resultado satisfactorio.

TABLA 42 CASO PRUEBA

Caso de Prueba de Aceptación
Código: P1_H7
Nombre: Configurar Distancia Máxima de Escucha
Descripción: El bot no debe escuchar ninguna conversación superior al rango de escucha determinado.
Condiciones de Ejecución: ✓ Estar autenticado bot y avatar ✓ Distancia inferior a la definida en el componente visual de configuración.

Entrada/Pasos de ejecución: ✓ Posicionar el avatar a una distancia inferior o igual a la definida ✓ Escribir algo valido
Resultado esperado: ✓ Bot responde correctamente
Evaluación de la prueba: Resultado satisfactorio.

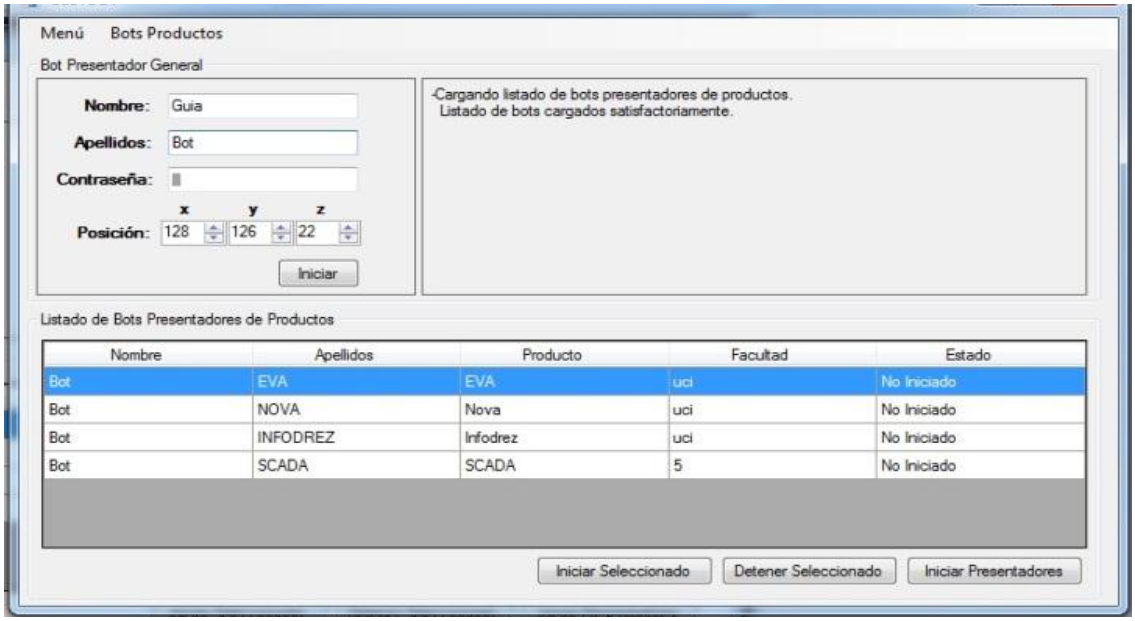
3.4 SOLUCIÓN

3.4.1 PANTALLA DE PRINCIPAL

Menús:

- **Menú:** Contiene los submenús **Configuración y Salir.**
- **Bot Productos:** Contiene los *submenús* **Cargar Listado y Gestionar.**
  - **Cargar Listado:** Muestra el listado de los Bots y sus productos asociados en el área: *Listado de Bots Presentadores de Productos.*

FIGURA 14- GESTBOT. PANTALLA PRINCIPAL



3.4.2 PANTALLA DE CONFIGURACIÓN

Campos:

Dirección Metaverso: para poner la dirección del servidor del metaverso al cual se conectaran los bots.

Nombre Metaverso: para poner el nombre del metaverso al cual se conectaran los bots.

Base de Datos: para poner el nombre de la base de datos que contendrá la información que se almacenará de los productos.

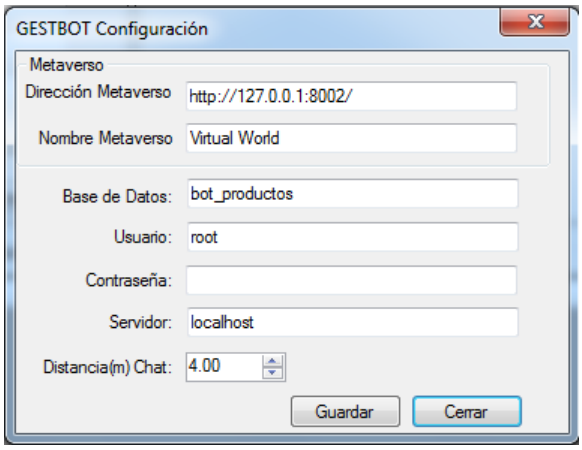
Usuario: para poner el nombre del usuario administrador de la base de datos que contendrá la información que se almacenará de los productos.

Contraseña: para poner la contraseña del usuario administrador de la base de datos que contendrá la información que se almacenará de los productos.

Servidor: para poner la dirección del servidor de la base de datos que contendrá la información que se almacenará de los productos.

Distancia (m) Chat: para especificar la distancia máxima de escucha de los bots.

FIGURA 15-PANTALLA DE CONFIGURACION



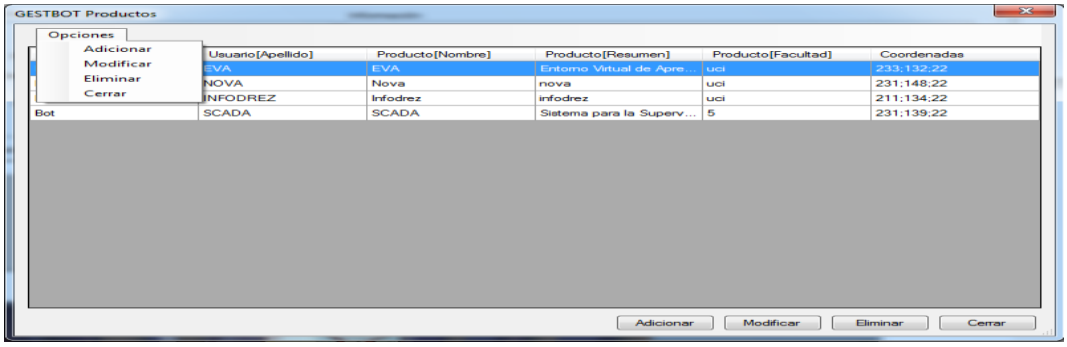
3.4.3 PANTALLA DE GESTIONAR

Muestra el listado de los bots y sus productos asignados existentes en la base de datos de productos.

Menú:

Opciones: Contiene los submenús: **Adicionar, Modificar, Eliminar y Cerrar.**

FIGURA 16-PANTALLA DE GESTIONAR



3.4.4 PANTALLA DE ADICIONAR

Muestra el listado de los bots y sus productos asignados existentes en la base de datos de productos.

Campos:

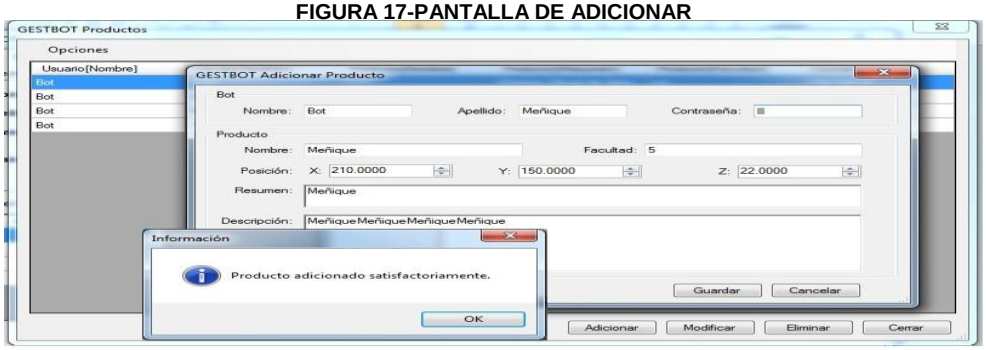
Bot:

- **Nombre:** Nombre del avatar que se utilizara de bot para manipular la información del producto.
- **Apellido:** Apellido del avatar que se utilizara de bot para manipular la información del producto.
- **Contraseña:** Contraseña del avatar que se utilizara de bot para manipular la información del producto.

Producto:

- **Nombre:** Nombre del producto.
- **Facultad:** Nombre del dueño del producto.
- **Posición:** Posición en la que aparecerá el bot que dispone la información del producto.
- **Resumen:** Resumen del producto.
- **Descripción:** Descripción del producto.

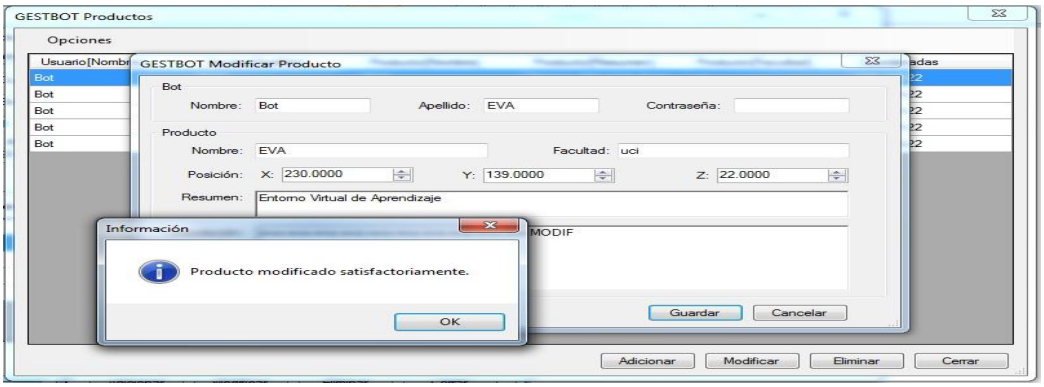
FIGURA 17-PANTALLA DE ADICIONAR



3.4.5 PANTALLA DE MODIFICAR

Muestra la información referente al producto seleccionado para modificar.

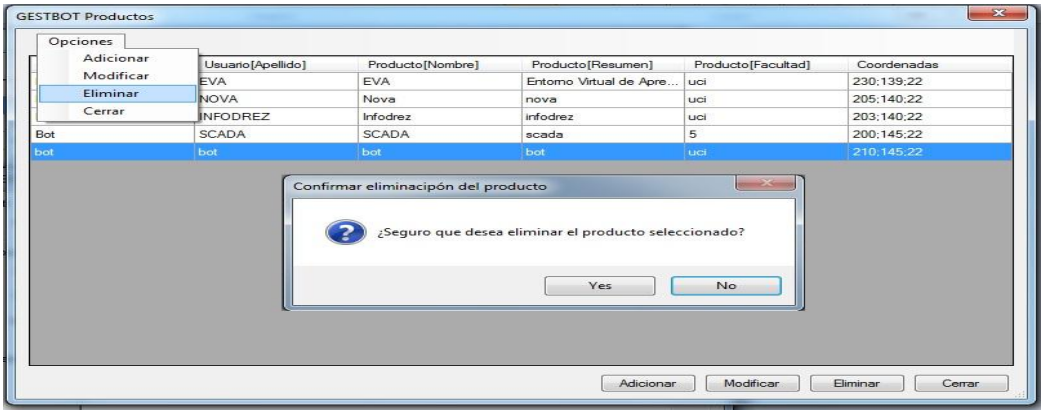
FIGURA 18-PANTALLA DE MODIFICAR



3.4.6 PANTALLA DE ELIMINAR

Tras seleccionar el producto que se va a eliminar, se pide confirmación de eliminación del mismo.

FIGURA 19-PANTALLA DE ELIMINAR



3.4.7 EJEMPLO DE LA APLICACIÓN FUNCIONANDO:

Anexos, imágenes 5, 6 y 7.

3.5 CONCLUSIONES PARCIALES

Con la conclusión del presente capítulo se logró validar la solución, plasmando el desarrollo de cada iteración y el desarrollo de las pruebas para la satisfacción del cliente.

CONCLUSIONES GENERALES

El desarrollo de los metaversos representa todo un nuevo mundo y ofrece amplias oportunidades para el modo de interacción en internet que se pueden incorporar a las interfaces tradicionales.

Con la realización del presente trabajo logró crear las bases para la creación de bots en función de necesidades del metaverso de la universidad siguiendo las buenas prácticas de la programación orientada a objetos y crear bots funcionales para la exposición de algún producto para la feria de productos virtuales del metaverso.

Esto permite la implementación de este modo de próximos bots enfocados a otras tareas que tributen a nuestros metaversos.

RECOMENDACIONES

Para futuros resultados se recomienda:

- Explotar las ventajas que nos brindan los bots en los metaversos, implementándolas en metaversos de la universidad.
- Incluir estas posibilidades en bot animales también para nuestros metaversos.

BIBLIOGRAFÍA

1. **Stephenson, Neal.** *"Snow Crash".* Barcelona : Editorial Gigamesh, 2005.

2. *METAVERSOS Y EDUCACIÓN Second Life como plataforma educativa.* **Márquez, Israel V.** Madrid (España) : REVISTA ICONO 14, 2011, Vol. 2. pp. 151-166. ISSN 1697-8293..

3. Pikubot-The Second Life Bot. [Online] 2012. http://www.pikkubot.de/dokuwiki/doku.php?id=es:faq.

4. SmatBots. [Online] http://www.smartbots2life.com/.

5. Open Simulator. [Online] http://opensimulator.org/wiki/Building_a_bot.

6. **Sánchez., Eduardo Rojo.** *Aplicación de la Herramienta Open Source Sloodle y las Tecnologías del Procesamiento del lenguaje Natural para el Desarrollo de una Plataforma de Virtual Learning en la Universidad Carlos III de Madrid.* Madrid : Universidad Carlos III de Madrid, 2010.

7. **Spad, Fiona.** Virtual-spain. [Online] Abril martes, 2010. [Cited: abril 1, 2012.] http://www.virtual-spain.es/component/content/article/39-blogs/92.

8. solocsharp. [Online] febrero 26, 2010. [Cited: Marzo 10, 2012.] http://solocsharp.blogspot.com/2010/02/concepto-versiones-visual-studio.html.

9. dc devart. [Online] http://www.devart.com/dotconnect/mysql .

10. **Penadés, Patricio Letelier y Mª Carmen.** *Métodologías ágiles para el desarrollo de software.* Universidad Politécnica de Valencia. : Laboratorio de Sistemas de Información. Departamento de Sistemas Informáticos y Computación.

11. **Pressman, Roger S.** *Ingeniería del Software. Un enfoque práctico.* Madrid : Concepción Femández , 2002. 0-07-709677-0.

12. *Métodologías Ágiles en el Desarrollo de Software.* **José H. Canós, Patricio Letelier y Mª Carmen Penadés.** Universidad Politécnica de Valencia : s.n.

13. 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..

14. **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..

15. 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..

16. **Mestras, Juan Pavón.** *Patrones de diseño orientado a objetos. .* Madrid : Dep. Ingeniería del Software e Inteligencia Artificial Universidad Complutense, 2004.

17. **Rey, Eduardo Mosqueira.** *Programacion Orientada a Objetos. Patrones de Diseño.* Coruña, España : LIDIA. Departamanto de Computacion. Universidad de Coruña.

18. Wikipedia. [Online] enero 2001. [Cited: febrero 18, 2012.] http://es.wikipedia.org/wiki/Avatar.

19. Wikipedia. [Online] enero 2001. [Cited: febrero 20, 2012.] http://es.wikipedia.org/wiki/Metaverso.

20. Wikipedia. [Online] enero 2001. [Cited: febrero 20, 2012.] http://es.wikipedia.org/wiki/Mundo_virtual.

21. Wikipedia. [Online] enero 2001. [Cited: febrero 15, 2012.] http://es.wikipedia.org/wiki/Second_Life.

22. **IEEE.** *Standard Glossary of Software Engineering Terminology.* 1990. IEEE Std 610.12-1990.

23. OpenMetaverse Fundation. [Online] [Cited: marzo 5, 2012.] http://openmetaverse.org/projects/libopenmetaverse.

24. **Dvorski, Dalibor D.** *INSTALLING, CONFIGURING, AND DEVELOPING WITH XAMPP.* Skills Canada – Ontario : s.n., 2007.

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

26. **Barres, David Griol.** *OpenCourseWare, Moodle y Mundos Virtuales: Experiencias educativas en la UC3M.* Granada : s.n., 2011.

27. **Mendez, Manuel.** El pais. [Online] diciembre 4, 2008. http://www.elpais.com/articulo/red/mundos/virtuales/abren/paso/colaboracion/empresarial/elpeputeccib/20081204elpcibenr_3/Tes.

RECOMENDACIONES

- Continuar la implementación de bots en UCIGRID con otras funciones.
- Implementar bot animales para incluirlos en los metaversos.
- Realizar un manual de ayuda a la aplicación desarrollada.

ANEXOS

IMÁGENES:

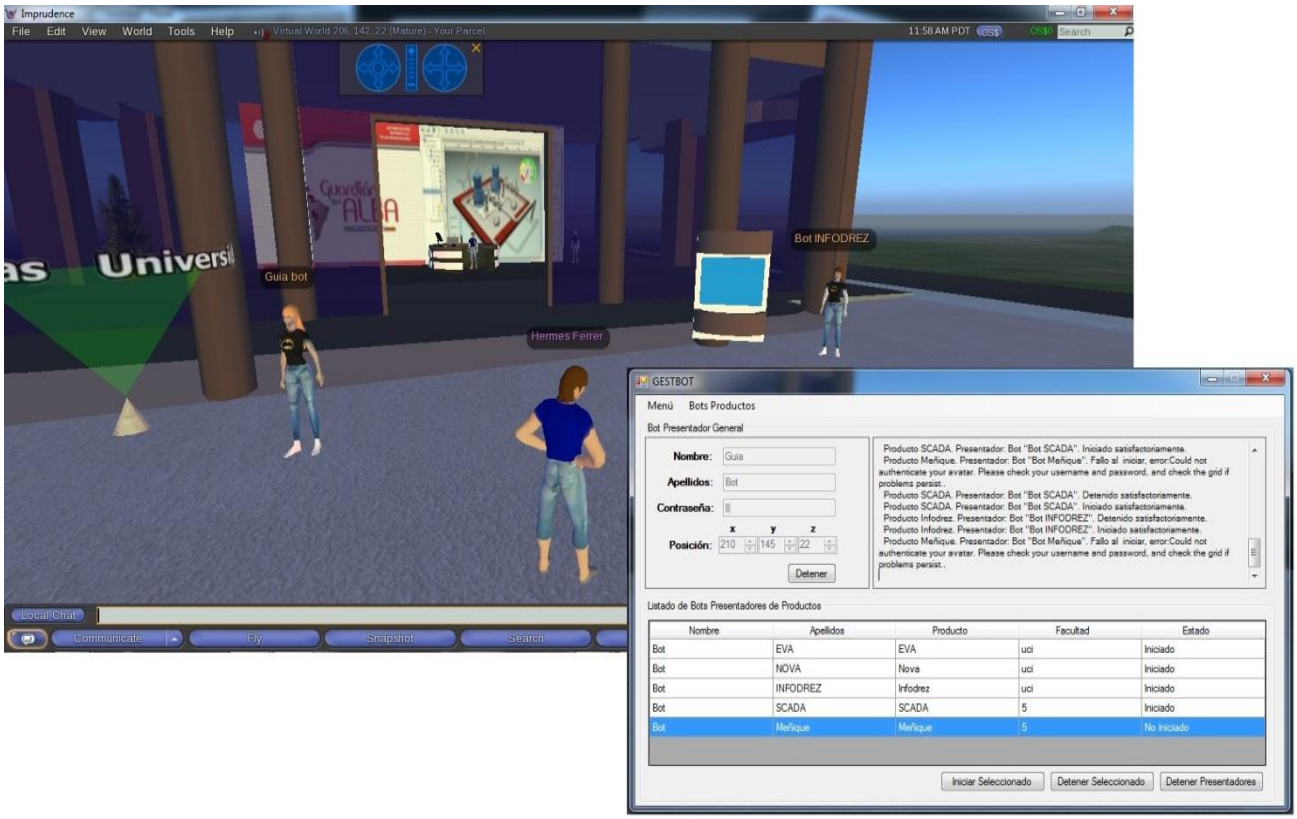
1- http://flybar.rtve.es

2- www.entropiauniverse.com

3- http://www.twinity.com

4- http://www.nurien.com

5-



6-


Imprudence
File Edit View World Tools Help
Virtual World 207, 145, 22 (Mature) - Your Parcel
12:26 PM PDT
Search
Bot SCADA has offered to teleport you to his or her location:
¿Desea teletransportarse al punto de exposición?
Teleport
Cancel
Bot INFODREZ
Guia bot
Hermes Ferrer
Hermes Ferrer: hola
Guia bot: Hola Sr(a). Hermes Ferrer
Hermes Ferrer: productos
Guia bot whispers: 5: "EVA"
8: "Nova"
9: "Infodrez"
10: "SCADA"
12: "Meñique"
Guia bot whispers: ¿Desea saber algo más acerca de un producto? ¿Sí o No?
Hermes Ferrer: si
Guia bot: Escriba el nombre o el identificador del producto.
Hermes Ferrer: SCADA
Guia bot: SCADA: Sistema para la Supervision Control y Adquisicion de Datos
Guia bot: El presentador del producto se encuentra disponible en la posición <231,139,22>.
Local Chat
Say
Gestures
Editor
Sky
Communicate
Fly
Snapshot
Search
Build
Map
Mini-Map
Inventory

7-


Imprudence
File Edit View World Tools Help
Virtual World 228, 139, 22 (Mature) - Your Parcel
12:28 PM PDT
Search
Bot SCADA
Bot EVA
Hermes Ferrer
Hermes Ferrer: hola
Bot SCADA: Hola Sr(a). Hermes Ferrer
Hermes Ferrer: producto
Bot SCADA: SCADA: Concebido por el grupo tecnico del proyecto que mas que un solo SCADA, fuese un configurador de SCADAs, de forma tal que pueda ser aplicado a cualquier proceso ndustrial o de cualquier indole que quiera ser automatizado.
Cuenta con varios modulos que interactuan entre si, entre ellos: Configuracion, Seguridad, Recoleccion, BD de Tiempo Real, entre otros.
Local Chat
Say
Gestures
Editor
Sky
Communicate
Fly
Snapshot
Search
Build
Map
Mini-Map
Inventory