UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Sistema de creación de mundos virtuales:

Módulo de administración de la información y comunicaciones

Alhvi Romancina Balcarcel Rodas

Guatemala

2010

Sistema de creación de mundos virtuales:

Módulo de administración de la información y comunicaciones

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Sistema de creación de mundos virtuales:
Módulo de administración de la información y comunicaciones

Trabajo de graduación presentado por Alhvi Romancina Balcarcel Rodas para optar al grado de Licenciada en Ingeniería en Ciencia de la Computación

Guatemala

2010

Vo. Bo.:

(f)_____

(Ing. Sergio Izquierdo)

Tribunal:

(f)_____

(Ing. Sergio Izquierdo)

(f)_____

(Ing. Luis Barrera)

(f)_____

(Ing. Erik Véliz)

Fecha de aprobación: Guatemala, 27 de mayo de 2010

PREFACIO

El sistema de creación de mundos virtuales desarrollado en este trabajo permite especificar y utilizar ambientes de simulación en tres dimensiones. La idea original del proyecto era desarrollar un sistema que permitiera crear mundos virtuales especializados para ser utilizados como herramienta educativa. Debido a la complicación del proyecto, a que se necesitaba asesoría de especialistas en el área de educación y a que esta asesoría podía aplicarse hasta que se tuviera una herramienta de software funcional, se decidió reducir el alcance del trabajo a la realización de dicha herramienta de software: un sistema de creación de mundos virtuales.

El plan original del trabajo fue propuesto por mi compañero Carlos Antonio Villagrán, a quien quiero agradecer por invitarme a trabajar con él en el desarrollo de este proyecto. También le agradezco a Carlos por la paciencia, por ayudarme a enfocarme en el trabajo y encontrar las soluciones cuando yo me enredaba sin razón en los problemas, por darme la oportunidad de trabajar en un proyecto interesante y con grandes aplicaciones y por los ánimos y el impulso constante que me ayudaron a terminar este trabajo.

Agradezco también a los creadores y documentadores de las herramientas de desarrollo, bibliotecas y aplicaciones gratuitas que utilizamos. Sin su enorme trabajo el desarrollo de esta aplicación no hubiera sido posible. Gracias también a toda la comunidad de internautas que comparten información y soluciones a problemas por medio de la web, a los autores de la bibliografía consultada y a las personas que nos dieron sugerencias para la implementación de este proyecto.

Agradezco a mi familia y amigos, por los consejos y el apoyo incondicional, especialmente a mis padres, y a mis amigos: Carlos y Henry. También a todas y cada una de las personas que ayudaron de alguna forma a mi formación académica y personal hasta este día.

ÍNDICE

PREFACIO IV
LISTA DE CUADROS VI
LISTA DE GRÁFICOS VII
RESUMEN IV
1. INTRODUCCIÓN 1
2. ESPECIFICACIÓN DEL SISTEMA 7
3. INTRODUCCIÓN MÓDULO DE ADMINISTRACIÓN DE LA INFORMACIÓN Y COMUNICACIONES 16
4. OBJETIVOS DEL MÓDULO DE ADMINISTRACIÓN DE LA INFORMACIÓN Y COMUNICACIONES . 17
5. MARCO TEÓRICO 18
6. DISEÑO DE LA REPRESENTACIÓN DE LOS DATOS 21
7. IMPLEMENTACIÓN DEL ALMACÉN DE DATOS 30
8. DISEÑO E IMPLEMENTACIÓN DEL API DEL SERVICIO WEB 31
9. FUNCIONALIDAD DEL SERVIDOR 32
10. CICLO DE VIDA DEL USUARIO DENTRO DE UN MUNDO VIRTUAL 42
11. CLIENTE ADMINISTRADOR 50
12. RESULTADOS 53
13. DISCUSIÓN 54
14. CONCLUSIONES 57
15. RECOMENDACIONES 58
16. BIBLIOGRAFÍA 59
17. APÉNDICE 1: GLOSARIO 61
18. APÉNDICE 2: REPRESENTACIÓN DE LOS DATOS 62
19. APÉNDICE 3: CÓDIGO FUENTE 83

LISTA DE CUADROS

Cuadro 1: Comparación entre las alternativas para la implementación del servidor 20
Cuadro 2: Tipos de usuario y operaciones de escritura permitidas 37
Cuadro 3: Tipos de usuario y operaciones de lectura permitidas 37
Cuadro 4: Funciones del API NaokWebService catalogadas como tipo de operación 38

LISTA DE GRÁFICOS

Figura 1 Captura de pantalla de MUD 2
Figura 2 Captura de pantalla del mundo virtual social: Second Life 3
Figura 3 Captura de pantalla del MMORPG World of Warcraft 4
Figura 4: Relación entre los componentes del sistema 8
Figura 5: Diagrama de ER de mundos 22
Figura 6: Diagrama de ER de mundos con ítems, estados y regiones 23
Figura 7: Diagrama de clases de mundos 24
Figura 8: Diagrama de ER de regiones 25
Figura 9: Diagrama de clases de regiones 26
Figura 10: Diagrama de ER de propiedades de los mundos 27
Figura 11: Diagrama de clases de propiedades del mundo 28
Figura 12: Diagrama de ER de ítems de los mundos 29
Figura 13: Diagrama de ER de mundos y usuarios conectados 29
Figura 14: Definición de la clase UserCredentialTR 33
Figura 15: Diagrama de clases del caché de usuarios 34
Figura 16: Relación entre el caché de usuarios y la clase NaokUserManager 36
Figura 17: Diagrama de clases del caché de mundos 40
Figura 18: Diagrama de clases de NaokWorldManager 41
Figura 19: Ciclo de vida del usuario dentro del sistema 42
Figura 20: Diagrama de clases de CharacterPositionTRSet 45
Figura 21: Diagrama de clases de ChatMessageTRSet 46
Figura 22: Diagrama de ER de DataUpdates 47
Figura 23: Definición de NaokUpdateTR 49
Figura 24: Relación entre los componentes del sistema 51
Figura 25: Ejemplo de las interfaces gráficas creadas en el cliente administrador 52
Figura 26: Diagrama de ER de ítems 62

Figura 27: Diagrama de clases de ítems 63
Figura 28: Diagrama de ER de propiedades de ítems 64
Figura 29: Diagrama de clases de propiedades de ítems 65
Figura 30: Diagrama de ER de trayectoria de ítems 66
Figura 31: Diagrama de clases de trayectoria de ítems 67
Figura 32: Diagrama de ER de personajes 68
Figura 33: Diagrama de clases de personajes 69
Figura 34: Diagrama de ER de propiedades de los personajes 70
Figura 35: Diagrama de clases de personajes 71
Figura 36: Diagrama de ER de misiones de los personajes 72
Figura 37: Diagrama de clases de misiones de los personajes 73
Figura 38: Diagrama de ER de misiones 74
Figura 39: Diagrama de clases de misiones 75
Figura 40: Diagrama de ER de eventos 76
Figura 41: Diagrama de clases de eventos 77
Figura 42: Diagrama de ER de condiciones de los eventos 78
Figura 43: Diagrama de clases de condiciones de los eventos 79
Figura 44: Diagrama de ER de acciones 80
Figura 45: Diagrama de clases de acciones 81
Figura 46: Diagrama de ER de parámetros de las acciones 82
Figura 47: Diagrama de clases de parámetros de las acciones 82

RESUMEN

El sistema de creación de mundos virtuales permite definir ambientes virtuales personalizados en tercera dimensión utilizando una interfaz gráfica de ventanas, donde un usuario administrador puede seleccionar y especificar las características de un mundo virtual. Esta especificación es almacenada en un servidor, para después ser retribuida por un cliente gráfico que representa visualmente el mundo y permite la interacción de los usuarios con el mismo. Para hacer posible esta interacción, el cliente gráfico y el servidor mantienen comunicación constante.

El sistema fue desarrollado en dos módulos: Módulo de administración de la información y comunicaciones y módulo de gráficas y simulación. El presente documento trata el tema del diseño e implementación del primero de dichos módulos.

El módulo de administración de la información y comunicaciones tuvo como objetivo crear el almacén físico de datos para los mundos virtuales. También crear un método de comunicación entre el servidor y varios clientes. Y además crear la interfaz que permite la creación de los mundos virtuales.

El almacén de datos se llevó a cabo utilizando una base de datos según el modelo relacional. La interfaz entre clientes y servidor se implementó por medio de un servicio web. El cliente que permite la creación de mundos virtuales, o cliente administrador, se implementó como una interfaz gráfica de ventanas.

El encargado de proveer la interfaz gráfica tridimensional que representa los mundos virtuales, simular las fuerzas físicas y eventos que suceden en el mundo y de manejar la interacción con el usuario es el módulo de gráficas y simulación. Para conocer los detalles de

implementación de este módulo consultar el trabajo: Sistema de creación de mundos virtuales – Módulo de gráficas y simulación, por Carlos Antonio Villagrán Juárez.

Para comprobar que se cumplieron los objetivos del proyecto se creó un mundo virtual de pruebas, el cual pudo definirse y utilizarse sin problemas a través del cliente administrador y el cliente gráfico.

Las comunicaciones del sistema se probaron a través de una red de área local y a través de Internet. Obteniendo una gran diferencia de rendimiento cuando se utiliza la LAN. Por lo que si se requiere implementar el sistema utilizando Internet como medio de comunicación, se sugiere realizar un estudio de desempeño, determinar las causas que provocan el comportamiento actual y hacer las correcciones necesarias para mejorarlo. También se sugiere evaluar otras alternativas para implementar la comunicación.

1. INTRODUCCIÓN

Al decir “mundo virtual”, la palabra “mundo” se refiere a un ambiente, el cual es considerado autónomo por parte de sus habitantes. La palabra “virtual” se refiere a algo que no existe, capaz de provocar el efecto de algo existente (Bartle 2003).

Los mundos virtuales son implementados por computadoras y simulan un ambiente. Algunas de las entidades en este ambiente actúan directamente bajo el control de personas individuales, siendo estos los “habitantes” del mundo. Generalmente una persona tiene un único habitante (llamado personaje o avatar) a quien controla y con quien se identifica. Con sus acciones, los habitantes realizan cambios en el ambiente. Varias personas pueden afectar el mismo ambiente al mismo tiempo. Por esta razón se dice que el mundo es compartido o multiusuario (Bartle 2003).

Para cumplir con la característica de que el mundo debe ser autónomo, el ambiente debe tener una serie de reglas por las cuales se rige y evoluciona. Estas reglas son las que permiten a los usuarios modificar el estado del mundo.

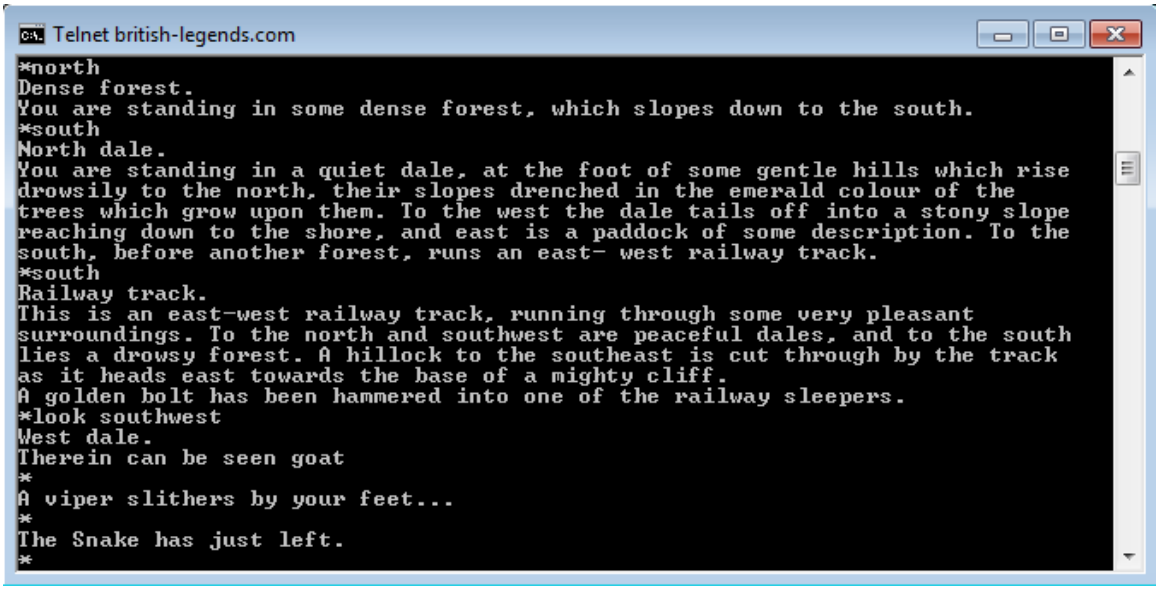
Un mundo virtual debe ser persistente, lo que significa que debe seguir funcionando aunque no se encuentren habitantes dentro de él. Otra de las características de un mundo virtual es que las respuestas a la interacción de los usuarios con el mismo se producen en tiempo real. Esto quiere decir que el sistema responde dentro del tiempo esperado por el usuario, según lo que tardaría realizar la operación en el mundo real. En este caso el tiempo de respuesta debe ser inmediato para la mayoría de operaciones.

1.1. Historia

Los mundos virtuales surgieron a finales de los años 70 como juegos de computadora originalmente conocidos como MUD’s (Multi User Dungeons), pues MUD era el nombre del primero de estos en prosperar (Bartle 2003). Estos juegos proponían aventuras en un mundo de fantasía, de las cuales el usuario era partícipe a través de su personaje. Inicialmente, los ambientes y los eventos que ocurrían dentro de ellos eran descritos utilizando texto (Figura 1).

Los MUD's lograron demostrar que aunque no se utilizara una tecnología de despliegue sofisticada, la representación rica y consistente de un espacio virtual puede ser vívidamente experimentada en la imaginación del usuario (Mitchel 1995).

Figura 1 Captura de pantalla de MUD



Debido a la aparición del Internet y a la evolución de la computación en los siguientes años, los MUD's se propagaron y varios programadores empezaron a implementar sus propios mundos virtuales. Uno de los que vale la pena mencionar es TinyMUD, el cual no tenía las características de un juego, sino permitía que los usuarios crearan elementos dentro del mundo virtual. Las personas pasaban el tiempo creando ambientes y objetos y hablando de sus creaciones (Bartle 2003). Los mundos virtuales inspirados en TinyMUD se llamaron "mundos virtuales sociales", y a diferencia de los "mundos virtuales de juego" no tienen una trama definida, no requieren que el usuario realice misiones ni reciba recompensas establecidas, sino se utilizan para que los usuarios creen sus propios objetos y se comuniquen y convivan con el resto de habitantes.

En la actualidad, los mundos virtuales basados en texto se han convertido en espacios digitales en tercera dimensión, con millones de usuarios, funcionamiento político y social complejo y con una gran variedad de características en su población (Peachey, y otros 2010). Como máximo exponente actual de los "mundos virtuales sociales" se puede mencionar a Second Life, desarrollado por Linden Lab (Figura 2). Al igual que TinyMUD, la base de Second

Life es el contenido creado por los usuarios, para esto incluye herramientas que permiten construir objetos tridimensionales a partir de formas básicas y además herramientas de programación que permiten diseñar contenido interactivo. Las personas pueden crear y distribuir objetos, incluso ponerlos a la venta y cambiar el dinero obtenido en el mundo virtual por dinero real. Los usuarios de Second Life aprovechan este sistema como medio de socialización, para hacer negocios, como medio de expresión artística, para educación y muchas otras actividades (Gollub 2007).

Figura 2 Captura de pantalla del mundo virtual social: Second Life

Los "mundos virtuales de juego" o mejor conocidos como MMORPG (Massive Multiplayer Online Role Playing Games) se han convertido en una industria multimillonaria del entretenimiento (Peachey, y otros 2010). Estos juegos siguen siendo una historia de fantasía donde el usuario crea un personaje que tiene un rol o trabajo específico y un conjunto de habilidades. Dentro del mundo virtual existen misiones y tareas que los personajes deben realizar para cumplir con su rol dentro del mundo. Al hacer las actividades el personaje puede subir de nivel, incrementar sus habilidades, conseguir nuevas armas y equipo y así especializarse en su trabajo y poder obtener mejores recompensas. Normalmente se necesita

que varias de las actividades sean realizadas en equipos donde participan usuarios con diferentes roles, por lo que se necesita que además de especializarse en las habilidades de su personaje, el usuario también sea capaz de integrar estas habilidades con las de sus compañeros de equipo (Gee s.f.). Uno de los MMORPG's más famosos es World of Warcraft (Figura 3) desarrollado por Blizzard Entertainment, que en la actualidad cuenta con más de doce millones de subscriptores (Peachey, y otros 2010).

Figura 3 Captura de pantalla del MMORPG World of Warcraft

1.2. Mundos virtuales y educación

Desde hace algunos años, varias universidades e instituciones académicas de nivel superior utilizan mundos virtuales sociales, como Second Life para realizar actividades educativas. El interés por esta herramienta radica en que además de permitir la socialización y el trabajo en equipo, permite que el usuario pueda construir objetos dentro del mundo virtual. Esto ha sido aprovechado por los educadores, quienes han utilizado Second Life para construir campus y aulas virtuales, bibliotecas y museos (Kemp y Livingstone 2006). También han utilizado esta herramienta para que los estudiantes realicen proyectos que serían muy difíciles

de realizar en el mundo real, como la implementación de un negocio o tienda virtual (Mason 2007), la creación de mini ciudades, y el entrenamiento para manipulación de equipo sofisticado (Kemp y Livingstone 2006).

La mayoría de interés hacia los mundos virtuales en el campo de la educación ha sido para los mundos virtuales sociales, como Second Life, pues Second Life es completamente libre de una narrativa, no tiene un escenario ni trama definida, por lo que los maestros tienen la libertad de construir sus propias metáforas y crear escenarios específicos (Kemp y Livingstone 2006). Esta es una gran ventaja para los educadores, pero si se utilizara un MMORPG específicamente diseñado para la educación, la trama, el escenario y los elementos de juego serían muy útiles como herramienta educativa. Dentro de un juego el usuario encuentra objetivos que debe cumplir dentro de la simulación, para alcanzar estos objetivos el jugador debe reconocer los problemas, determinar las reglas de la simulación y decidir qué elementos del juego puede utilizar para resolver estos problemas (Gee s.f.). Específicamente se mencionan las siguientes características de un video juego que pueden ser útiles para la educación :

- El crear una simulación temática de un sistema complejo y hacer que el usuario sea parte, y esté dentro de la simulación, hace que él sea capaz de entender el sistema como un todo y no sólo aprender eventos que parezcan locales o que suceden al azar (Gee s.f.).
- La mente humana funciona como un simulador, capaz de imaginar un escenario y colocar a la persona que lo imagina en diferentes roles dentro de la acción. Tal como se hace también dentro de un juego de video. Según el autor James Paul Gee, el aprendizaje se da al realizar generalizaciones de las diferentes experiencias y simulaciones de un sujeto. Por lo que los videojuegos son una herramienta natural para el aprendizaje (Gee s.f.)
- Los objetivos de un juego pueden alcanzarse por múltiples rutas, pero al decidir qué camino tomar, el usuario debe considerar las consecuencias y evaluar los diferentes trayectos para determinar cuál, según él, es el mejor camino a seguir. Al ejecutar su plan, si no obtiene los resultados esperados, el usuario debe de volver a empezar el proceso. Por lo que para complementar las tareas del juego se requiere la formulación de hipótesis, experimentación y análisis de los resultados (Mayo 2009).
- En un juego, la inteligencia puede distribuirse dentro de los diferentes elementos del mismo. Por ejemplo, el hacer que el personaje tenga ciertas habilidades necesarias para

completar una tarea, permite al usuario empezar la actividad con cierto grado de efectividad, sin que él propiamente tenga el completo conocimiento de lo que va a realizar, sino que lo tenga parcialmente a través del personaje. Utilizando el personaje el jugador aprende luego de prueba, error y retroalimentación (Gee s.f.).

1.3. Sistema de creación de mundos virtuales

La idea original de este proyecto era crear un MMORPG que pudiera ser utilizado para educación, pero debido a la dimensión muy grande del plan y a la falta de especialistas en el área de educación en el equipo de trabajo, se decidió reducir el alcance del proyecto a la realización de una herramienta de software que permita crear mundos virtuales. Con esta herramienta de software se podrá diseñar el MMORPG en el futuro.

El sistema de creación de mundos virtuales se realizó en dos módulos: Módulo de administración de la información y comunicaciones y Módulo de gráficas y simulación. En la siguiente sección de este documento se especifican las características que tiene el sistema de creación de mundos virtuales en conjunto.

.

2. ESPECIFICACIÓN DEL SISTEMA

El sistema de creación de mundos virtuales tiene como objetivos:

- Crear la infraestructura de software que permita definir mundos virtuales.
- Crear una interfaz gráfica que permita definir mundos virtuales.
- Crear una interfaz gráfica que permita utilizar los mundos virtuales.

Para esto se definen los siguientes componentes del sistema.

2.1. Componentes del sistema

2.1.1. Servidor El servidor es el lugar en el que "residen" los mundos virtuales, mantiene la copia que hace que el mundo virtual sea común a todos los usuarios. Los clientes se conectan al servidor a través de una red y obtienen de él la información del estado del mundo virtual y de los demás clientes conectados.

2.1.2. Cliente de mundos virtuales Es una interfaz gráfica en tres dimensiones que permite que un usuario interactúe con un mundo virtual. Obtiene los datos del estado del mundo virtual en el servidor, y con éstos representa el ambiente que es explorado por el usuario.

2.1.3. Cliente administrador Es una interfaz gráfica con el propósito de crear y administrar los mundos virtuales. Esta interfaz está disponible sólo para el administrador del sistema.

Todos los clientes interactúan con el mismo servidor. La interacción entre los componentes del sistema se muestra en la Figura 4.

Figura 4: Relación entre los componentes del sistema

2.2. Definiciones

En el sistema de creación de mundos virtuales se utilizan las siguientes definiciones.

2.2.1. Mundo virtual Es la simulación computarizada de un ambiente. En este caso es una representación gráfica tridimensional que contiene entidades y personajes, y permite que un individuo interactúe con estos a través de un avatar. La interacción entre varios individuos también es posible, compartiendo el mismo mundo virtual.

Los mundos virtuales cumplen leyes. Éstas pueden ser leyes físicas, como la gravedad, o leyes de la simulación, como una secuencia de actividades que deben realizarse para dar por terminada una misión.

Los mundos pueden ser compartidos por todos los usuarios, y también pueden ser instancias (copias) para un grupo de usuarios específico.

2.2.2. Entidad del mundo virtual (Ítem) Es cualquier objeto que se encuentre dentro del mundo virtual y no sea directamente controlado por el usuario. Se incluyen los objetos de escenario y decoración, los objetos que interactúan con los avatares y los personajes controlados por computadora NPC (Non Player Character). Los ítems pueden tener

características y propiedades específicas que definen su funcionalidad. Las entidades del mundo virtual pueden moverse dentro del ambiente siguiendo una trayectoria definida.

2.2.3. Avatar (Personaje) Es una representación del usuario como un modelo en tres dimensiones dentro del mundo virtual. Es capaz de recorrer el espacio tridimensional y de interactuar con el mismo.

La interacción se realiza mediante mensajes de texto, animaciones e ingreso de datos por medio de mouse y teclado. De esta forma el avatar podrá realizar actividades y misiones dentro del mundo virtual.

Los personajes pueden poseer ítems. Los ítems pueden ser necesarios para realizar actividades dentro de la simulación o decorativos, como ropa y accesorios.

Los personajes tienen propiedades que definen como estos interactúan con el mundo virtual. Estas pueden ser físicas, como velocidad, fuerza y peso, y de control, como puntuación.

2.2.4. Actividades Las actividades que se realizan dentro de la simulación son:

- Exploración del mundo virtual: El personaje puede recorrer el mundo virtual y observar los ítems que se encuentran dentro de él.
- Información acerca de los ítems y NPC: Los ítems y NPC pueden darle información al usuario acerca de lo que estos representan dentro del mundo virtual.
- Misiones: Los ítems y NPC pueden pedirle al personaje que realice determinadas actividades a cambio de una recompensa. La recompensa puede ser un aumento en su puntuación o un ítem.
- Obtención de ítems: El personaje puede colectar ítems.

Las actividades pueden realizarse de manera individual, o en conjunto con varios personajes a la vez.

2.3. Descripción general del sistema

2.3.1. Requerimientos funcionales

2.3.1.1. Sistema

- Permitir la creación de mundos virtuales.
- Comunicar los datos de los mundos virtuales y usuarios a través de internet.
- Permitir la actividad de múltiples mundos virtuales al mismo tiempo.

2.3.1.2. Mundo virtual

- Representar el ambiente de la simulación en tres dimensiones.
- Cumplir leyes predefinidas, físicas y de simulación.
- Responder a la interacción con el usuario según la definición en la sección 2.2.3.
- Permitir la interacción de varios usuarios en el mismo mundo virtual.
- Implementar las actividades según su definición en la sección 2.2.4.

2.3.1.3. Personajes

- Representar al usuario en el mundo virtual.
- Explorar el mundo virtual en tres dimensiones.
- Realizar actividades dentro del mundo virtual según la definición de la sección 2.2.4.
- Interactuar con los elementos del ambiente y con otros personajes a través de texto, de animaciones y de ingreso de datos a través de teclado y mouse.
- Adquirir y administrar ítems.
- Adquirir y administrar misiones.

2.3.2. Tipos de usuarios

El sistema está dirigido a dos tipos de usuarios:

- Administrador del sistema: Encargado de administrar el sistema en general y crear los mundos virtuales.
- Usuario general: Utiliza los mundos virtuales.

2.3.3. Ambiente operativo El software está destinado a utilizarse en computadoras personales y debe poder ser ejecutado sobre los siguientes sistemas operativos: Windows, Linux, Mac OS.

2.3.4. Requerimientos no funcionales

- El software debe ser desarrollado con herramientas de distribución gratuita, ya que no se cuenta con recursos para la adquisición de herramientas de desarrollo de software.
- El software debe ser multiplataforma y de fácil instalación.

2.4. Requerimientos detallados de información

2.4.1. Mundos Un mundo debe tener:

- Un nombre que lo identifique.
- Un estado que indica si es:
    - Plantilla: Puede ser utilizado para generar copias de él mismo.
    - Instancia: Copia individual del mundo para un grupo de usuarios específico.
    - Único: No tiene copias y es compartido por todos los usuarios.

Un mundo puede tener:

- Propiedades
    - Leyes que físicas, como la gravedad.
    - Tipo de terreno.
    - Tipo de cielo.
- Ítems y NPC.
- Personajes de los usuarios.
- Regiones: Definen áreas específicas dentro del mapa. Se forman por un conjunto de puntos dentro del espacio tridimensional. Los puntos son de la forma [X, Y, Z].

2.4.2. Ítems Un ítem debe tener:

- Un nombre que lo identifique.
- Un estado que indica si es:
    - Plantilla: Puede ser utilizado para generar copias de él mismo.
    - Instancia: Copia individual del mundo o personaje que lo posee.
    - Único: Instancia común para todos los mundos y personajes que lo utilizan.

Un ítem puede tener:

- Una posición dentro del mundo virtual: Representada por una coordenada del espacio tridimensional en forma de punto [X, Y, Z] y un vector de rotación, de la forma [RX, RY, RZ].
- Propiedades.
- Un modelo tridimensional asociado.
- Indicador de la posición con respecto al ítem o personaje al que pertenece.
- Un conjunto de eventos a los que responde.
- Una trayectoria.
- Otros ítems.

2.4.3. Trayectoria Está formada por una serie de posiciones, por las cuales un ítem debe moverse constantemente. Una trayectoria tiene:

- Un nombre.
- Un estado, que puede ser plantilla, instancia o única.
- Un conjunto de puntos [X, Y, Z] , [RX, RY, RZ] que determinan la posición del objeto dentro del mundo.

2.4.4. Personajes Un personaje tiene:

- Un nombre que lo identifica.
- Una posición dentro del mundo virtual: Representada por una coordenada del espacio tridimensional en forma de punto [X, Y, Z] y un vector de rotación, de la forma [RX, RY, RZ].
- Un estado que indica si es:
    - Plantilla: Puede ser utilizado para generar copias de él mismo.

    - Instancia: Copia individual del usuario.
- Propiedades:
    - Un modelo tridimensional asociado.
    - Peso del objeto, para aplicarle gravedad.
    - Velocidad, para caminar.
    - Fuerza para saltar.
    - Puntos: Adquiridos por completar actividades.
- Ítems
- Misiones

2.4.5. Misiones Las misiones tienen:

- Un nombre.
- Un estado. Que puede ser plantilla, instancia y único.
- Un conjunto de eventos, que hay que cumplir para dar por terminada la misión.
- Un indicador del progreso de la misión, para cada personaje que la obtenga.

2.4.6. Eventos Un evento es un conjunto de condiciones que al cumplirse ejecutan una cadena de acciones. Los eventos tienen:

- Un estado: Indica si el evento está activo o inactivo.
- Condiciones: Las condiciones que deben cumplirse.
- Relación entre las condiciones: Permite establecer una relación lógica entre las condiciones listadas.
- Un conjunto de acciones, que se ejecutan al cumplirse las condiciones según la relación.

2.4.6.1. Condiciones de los eventos Estas necesitan de:

- Un tipo: Identifica a la condición y la información que debe de ser analizada para que la condición se cumpla. Ej. Mouse_Click
- Parámetros: Los datos que se utilizan en el análisis de la información.

2.4.7. Acciones Realizan operaciones que modifican las propiedades y funcionalidades del mundo y sus entidades. Las acciones tienen:

- Operación a realizar: Identifica la operación a realizar. Ej: Agregar ítem a un personaje.
- Parámetros: Los datos a utilizar en la operación.

2.4.8. Usuarios Los usuarios tienen:

- Nombre de usuario.
- Nombre y apellido.
- Género.
- Fecha de nacimiento.
- Estado, que indica si está activo o inactivo.
- Rol: Indica el papel del usuario dentro del sistema y determina los permisos que este posee.

2.5. Limitaciones del proyecto

2.5.1. Eventos y acciones Por el momento se cuenta con un conjunto predefinido de eventos y acciones que el sistema detecta y ejecuta. Estos son los básicos utilizados en la creación del mundo virtual de pruebas. En el futuro se piensa ampliar el conjunto de eventos y acciones o incluso proveer de alguna forma en la que los usuarios del sistema puedan definir estos eventos y acciones.

2.5.2. Seguridad informática Para hacer más sencillo el trabajo no se toman en cuenta todas las medidas de seguridad que se deberían de implementar en un proyecto en producción. Los aspectos de seguridad en los que se trabajan son los básicos para cumplir con la funcionalidad del proyecto.

2.5.3. SDK Se obtuvo como sugerencia la implementación de un SDK (Software Development Kit) para permitir que se desarrollen mundos virtuales y funcionalidad extra para el sistema por medio de programación. Pero, por el momento no se piensa desarrollar este SDK, pues la idea del proyecto es poder crear los mundos virtuales por medio de una interfaz gráfica y no mediante programación.

2.6. Diseño preliminar

2.6.1. Herramientas de desarrollo

- Lenguaje de programación: Java. Permite que el cliente sea ejecutado en cualquier plataforma.
- Ambiente de desarrollo de software: Eclipse. De distribución gratuita y de código abierto.

2.6.2. Arquitectura del sistema El sistema se separa en dos módulos:

2.6.2.1. Módulo de administración de la información y comunicaciones

Este módulo es el encargado de almacenar la información de los mundos virtuales en el servidor. También se encarga de crear un método para hacer que esta información pueda ser accedida por los clientes. Además implementa el cliente administrador definido en la sección 2.1.3.

2.6.2.2. Módulo de gráficas y simulación El encargado de proveer la interfaz gráfica tridimensional que representa los mundos virtuales, definida en la sección 2.1.2, simular las fuerzas físicas y eventos que suceden en el mundo y de manejar la interacción con el usuario.

3. INTRODUCCIÓN AL MÓDULO DE ADMINISTRACIÓN DE LA INFORMACIÓN Y COMUNICACIONES

Como se menciona en las secciones anteriores, el sistema de creación de mundos virtuales se desarrolló en dos módulos. El módulo de administración de la información y comunicaciones, comprendido en este trabajo y el módulo de gráficas y simulación realizado por Carlos Villagrán.

El desarrollo del módulo de administración de la información y comunicaciones comprendió la definición e implementación de un sistema de almacenamiento de datos para guardar la información de los mundos virtuales y sus usuarios, implementado como una base de datos utilizando el manejador MySQL. La definición e implementación de una aplicación de software que administrara la ejecución de los mundos virtuales del lado del servidor, la cual se desarrolló en el lenguaje de programación Java. La definición e implementación de un sistema de comunicación de datos a través de la red, la cual se implementó como un servicio web, desarrollado en Java, e implementado en el servidor Glassfish utilizando Metro. Este módulo también comprendió la elaboración de un cliente administrador que permitió definir los mundos virtuales.

Con las herramientas de software provistas por el módulo de administración de información y comunicaciones, el módulo de gráficas y simulación pudo generar una interfaz gráfica tridimensional, por medio de la cual los usuarios pueden interactuar con el mundo virtual.

A continuación se detallan los objetivos del módulo de Administración de la Información y Comunicaciones, los conocimientos necesarios para la realización del mismo y los procedimientos llevados a cabo para su implementación.

4. OBJETIVOS DEL MÓDULO DE ADMINISTRACIÓN DE LA INFORMACIÓN Y COMUNICACIONES

4.1. Objetivos generales

- Diseñar e implementar el sistema de almacenamiento de datos para la aplicación.
- Diseñar e implementar las funcionalidades del servidor
- Diseñar e implementar las comunicaciones del sistema.
- Diseñar e implementar el cliente administrador.

4.2. Objetivos específicos

- Definir el medio y las herramientas a utilizar para el almacén físico de datos.
- Diseñar la representación de los datos en el almacén físico.
- Implementar el almacén físico de datos.
- Programar las rutinas necesarias para agregar, modificar y eliminar datos del sistema.
- Definir la tecnología de comunicaciones a implementar.
- Diseñar el funcionamiento del servidor según los requerimientos.
- Programar las rutinas necesarias para la implementación del servidor.
- Implementar una interfaz que permita la interacción con el módulo gráfico y el cliente administrador.
- Diseñar las interfaces de usuario que permitan administrar el sistema.
- Programar la funcionalidad del cliente administrador utilizando la interfaz provista por el servidor.

5. MARCO TEÓRICO

Para comprender lo que se desarrolla en este módulo es necesario tener conocimiento acerca de los temas de almacenamiento de información y redes de computadoras.

5.1. Almacenamiento de información

Una de las características que debe tener el sistema de creación de mundos virtuales, es que debe de almacenar información de manera persistente. Para esto se utiliza un sistema de bases de datos.

Un sistema de bases de datos es básicamente un sistema de mantenimiento de registros computarizado, cuyo propósito es almacenar información y permitir que usuarios obtengan y modifiquen esta información cuando lo necesiten. (Date 2000)

Los componentes de un sistema de bases de datos son: información, hardware, software y usuarios. La información se refiere a los datos que se tiene interés en almacenar. El hardware es el medio físico en el cual se almacenan los datos, por ejemplo discos duros. Dentro del software se incluye un intermediario entre el hardware y los usuarios, que se encarga de manejar las solicitudes de acceso a la base de datos, esta capa de software se conoce como manejador de base de datos o DBMS (Database Management System). (Date 2000)

El manejador de base de datos que se utiliza en este proyecto es MySQL. Este es un manejador gratuito y de código abierto que implementa el modelo relacional. El modelo relacional es un modelo de representación de datos basado en lógica y matemáticas, donde la información está organizada en tablas (relaciones) con columnas y filas definidas y existen operaciones que se pueden realizar sobre las tablas para obtener información. Una característica importante del modelo es que estas operaciones dan como resultado nuevas tablas. Utilizando este modelo se diseñó la representación de los datos, la cual puede verse en la sección 6.

5.2. Redes de computadoras

Una red de computadoras es una colección de ordenadores autónomos interconectados por una sola tecnología. Se dice que dos computadoras están conectadas si son capaces de intercambiar información. La conexión puede ser vía cable, fibra óptica, microondas, rayos infrarrojos o comunicación satelital (Tanenbaum 2003).

Debido a que se necesita que múltiples usuarios intercambien información e interactúen en el mismo mundo virtual, se utilizará una red de computadoras. El modelo de red a utilizar es el conocido como cliente-servidor.

Los clientes y servidores son entidades separadas lógicamente que trabajan juntas a través de una red para cumplir una tarea. El servidor es un proveedor de servicios. El cliente es el consumidor de estos servicios. Un servidor puede atender varios clientes al mismo tiempo y regular el acceso a recursos compartidos entre ellos. (Orfali, Harkey y Edwards 1999)

Los clientes siempre inician el diálogo solicitando el servicio. Los servidores están pasivamente esperando las solicitudes de los clientes. El cliente y el servidor interactúan a través de un mecanismo de intercambio de mensajes. El mensaje es el medio por el que se realizan las solicitudes y respuestas. (Orfali, Harkey y Edwards 1999)

Las alternativas que se evaluaron para la implementación del servidor son: comunicación por sockets y servicios web.

5.2.1. Comunicación por sockets Los sockets representan extremos en una línea de comunicación (Gay 2000). Abstraen los detalles de bajo nivel de las redes y permiten que un programador trate una conexión de red como una cadena donde puede escribir y leer mensajes en forma de bytes (Harold 2004), proveyendo dos partes o “canales” para intercambio de información: uno para escuchar (leer) y otro para mandar (escribir) (Walton 2001).

La comunicación se hace entre un cliente y un servidor, cada uno cuenta con un socket de dos canales. El servidor es el encargado de administrar las conexiones, pudiendo tener varios sockets asignados a distintos clientes.

Como sólo se cuenta con un punto de entrada y un punto de salida y la información se representa como una cadena de bytes, es necesario definir un protocolo de comunicación, es decir la sintaxis y semántica de los mensajes.

5.2.2. Servicios Web Un servicio veb es un sistema de software diseñado para permitir la interoperabilidad entre máquinas a través de una red. Tiene una interfaz descrita en un formato procesable por máquina (específicamente WSDL). Otros sistemas interactúan con el servicio web en una manera prescrita por su descripción usando mensajes SOAP, típicamente transmitido utilizando HTTP con una serialización XML en conjunto con otros estándares relacionados con la Web (W3C Working Group 2004). Los servicios web son básicamente un API accesible a través de la red.

Cuadro 1: Comparación entre las alternativas para la implementación del servidor

	Sockets	Servicios Web
Desarrollo	Es necesario definir un protocolo de comunicación. El cliente y el servidor deben interpretar el significado de los mensajes. El servidor debe administrar las conexiones de los clientes.	Se programa como un API común, el marco de desarrollo utilizado abstrae la implementación de redes. Puede utilizarse directamente en los clientes sin requerir programación extra.
Desempeño	Al desarrollar un servidor específico se contaría con un mejor desempeño.	Puede existir un sobrecosto en la comunicación.

Para la realización de este proyecto se utilizó un servicio web. Para que el servicio web pueda ejecutarse debe residir en un servidor de aplicación. El servidor que se utilizó es Glassfish. Glassfish es un servidor de aplicación multiplataforma, gratuito y de código abierto, desarrollado por Sun Microsystems para la plataforma Java. Este servidor de aplicación provee de un marco de desarrollo para servicios web llamado Metro.

6. DISEÑO DE LA REPRESENTACIÓN DE LOS DATOS

Para diseñar la representación física de los datos se utilizan diagramas de entidad-relación (ER) y para representar la estructura de datos en memoria se utilizan diagramas de clases. A continuación se presenta la especificación de los datos para los mundos virtuales según la especificación de requerimientos de información. Se utiliza la estructura diseñada para los mundos como un ejemplo de la metodología utilizada. Los diagramas de entidad-relación y de clases para los demás datos necesarios (ítems, personajes, misiones, eventos, acciones, etc.) pueden consultarse en el Apéndice 2: Representación de los datos.

Los mundos se representan con la entidad llamada Worlds. El nombre y el estado son dos entidades separadas ya que pueden ser compartidos por varias copias de un mismo mundo. Se definen entonces las entidades WorldNames y WorldStatus, para estos propósitos (Figura 5)

Como se especificó en los requerimientos (sección 2.4.1), un mundo puede tener regiones, ítems y propiedades. Para vincular estos con el mundo virtual se crean las tablas: WorldRegions, WorldAssignedProperties y WorldItems, como se muestra en la Figura 6.

La representación en objetos equivalente a la Figura 6 se muestra la Figura 7. Donde se muestra como un mundo virtual contiene un nombre, un estado, un conjunto de ítems, un conjunto de regiones, un conjunto de propiedades y un conjunto de personajes.

Figura 5: Diagrama de ER de mundos

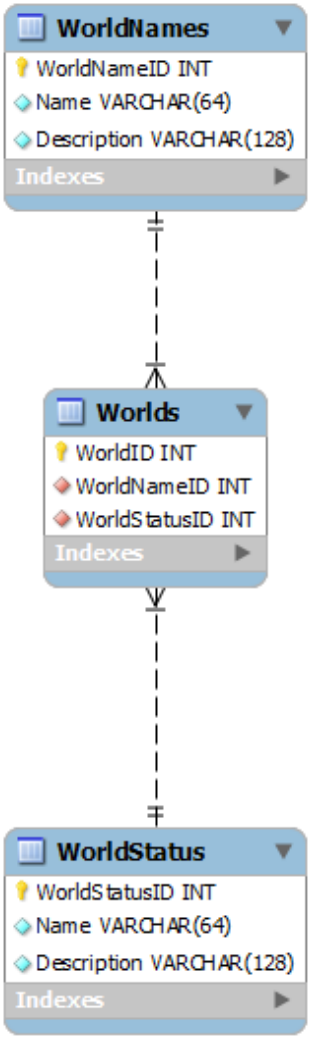


Figura 6: Diagrama de ER de mundos con ítems, estados y regiones

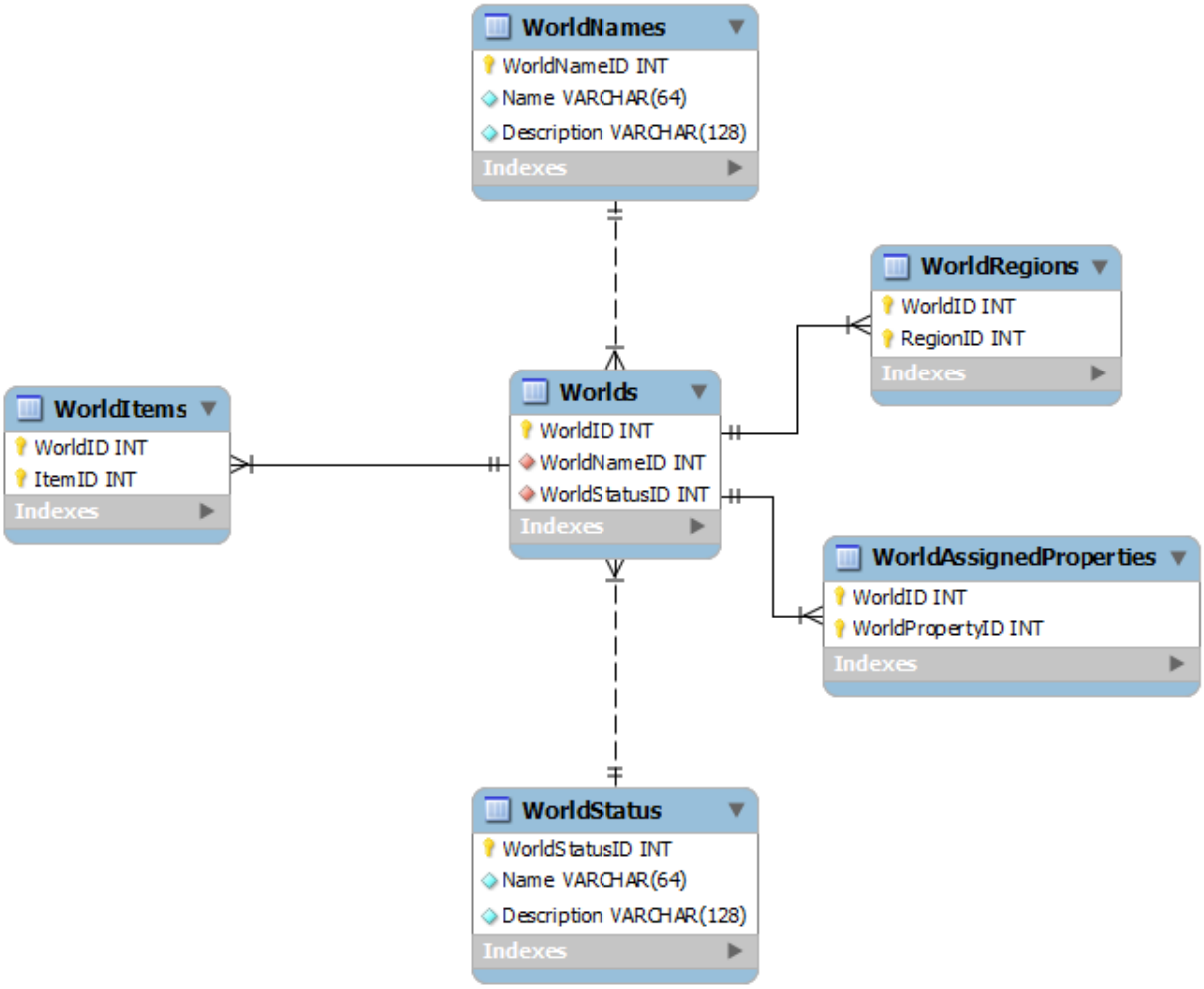
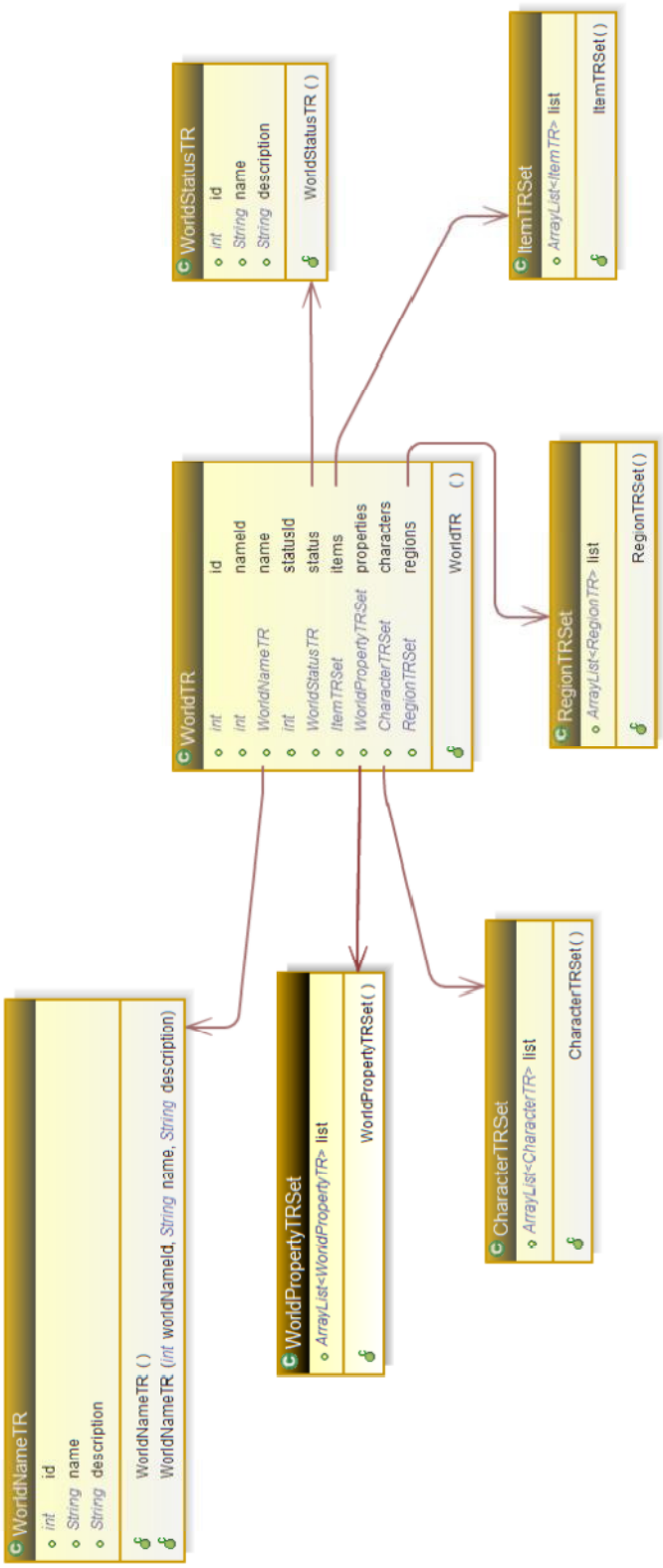


Figura 7: Diagrama de clases de mundos



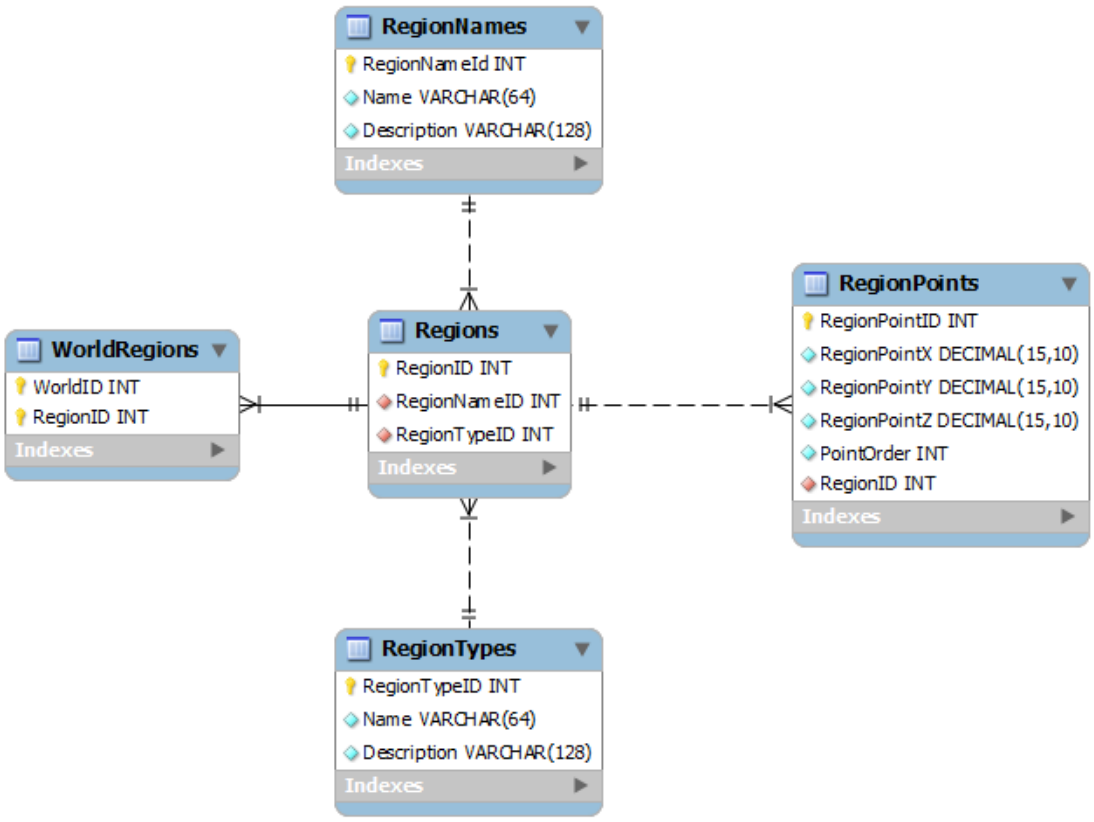
6.1.1.1. Regiones Para administrar los datos de las regiones se define la tabla Regions. Debido a que pueden existir copias de regiones y los nombres son compartidos por estas copias, se crea una entidad separada RegionNames que almacena los nombres de las regiones.

Las regiones son utilizadas para definir fronteras dentro del mundo virtual. Pensando en que estas podrían tener otra aplicación en el futuro, se creó la entidad RegionTypes, que indica el tipo de la región.

Como se define en la especificación de las regiones, en la sección 2.4.1, estas se forman de un conjunto de puntos en el espacio tridimensional. Este conjunto de puntos se almacena en la tabla RegionPoints.

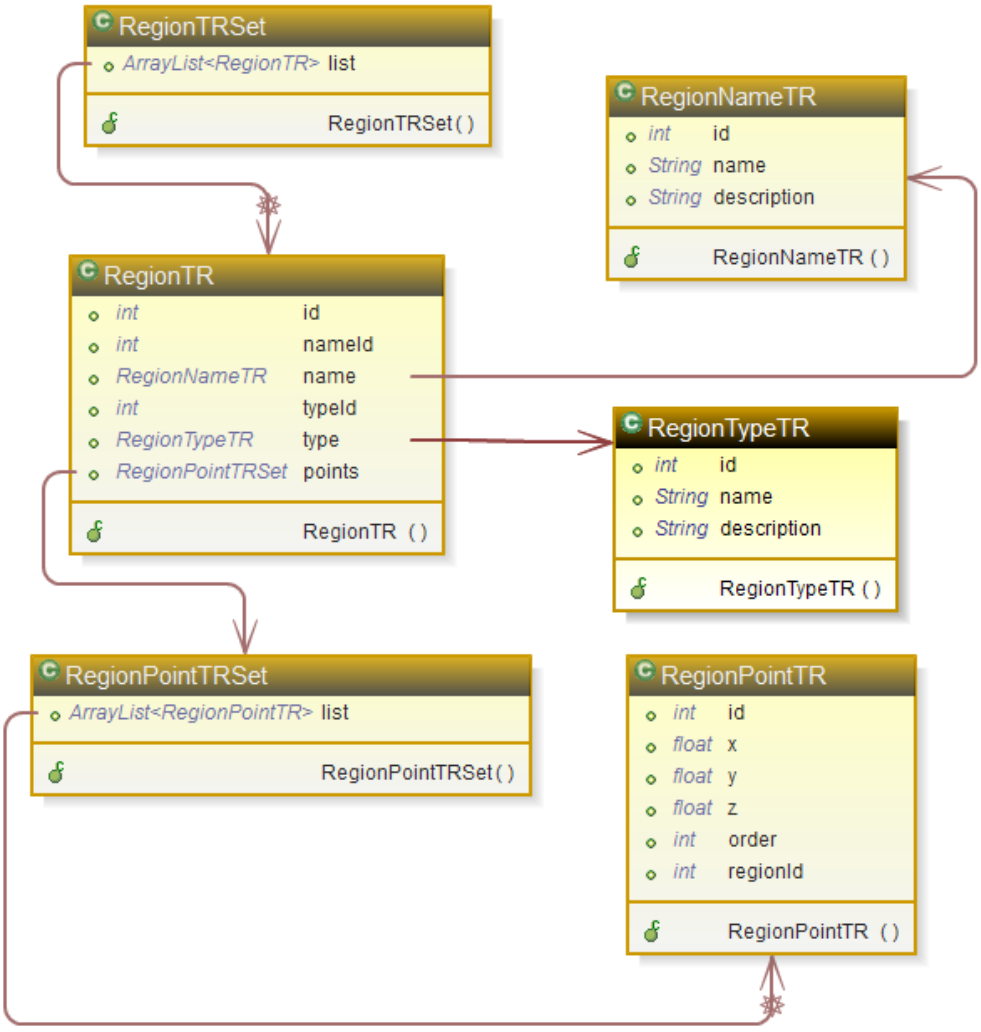
Un mundo virtual puede tener varias regiones. Estas se especifican en la entidad WorldRegions. Todas las entidades mencionadas anteriormente pueden verse en la Figura 8.

Figura 8: Diagrama de ER de regiones



El diagrama de clases para las regiones se muestra en la Figura 9.

Figura 9: Diagrama de clases de regiones

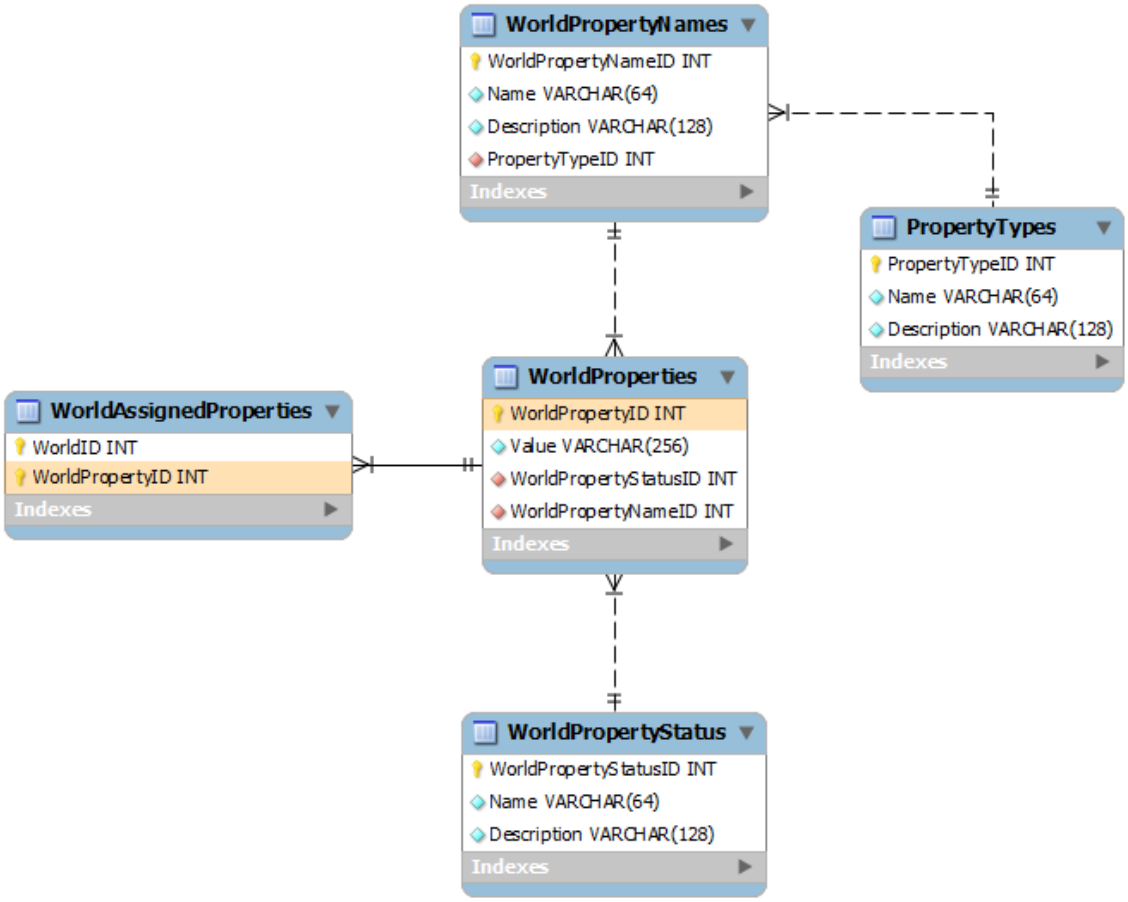


6.1.1.2. Propiedades de los mundos Las propiedades de los mundos se almacenan en la tabla WorldProperties (Figura 10). El esquema sigue la forma de las representaciones anteriores, con una tabla de nombres (WorldPropertyNames) separada y una tabla de estados (WorldPropertyStatus).

Se define un tipo para las propiedades, este especifica el tipo de dato que tiene la propiedad, el cual puede ser: número entero, número decimal, cadena de caracteres, etc. El tipo está relacionado directamente con el nombre.

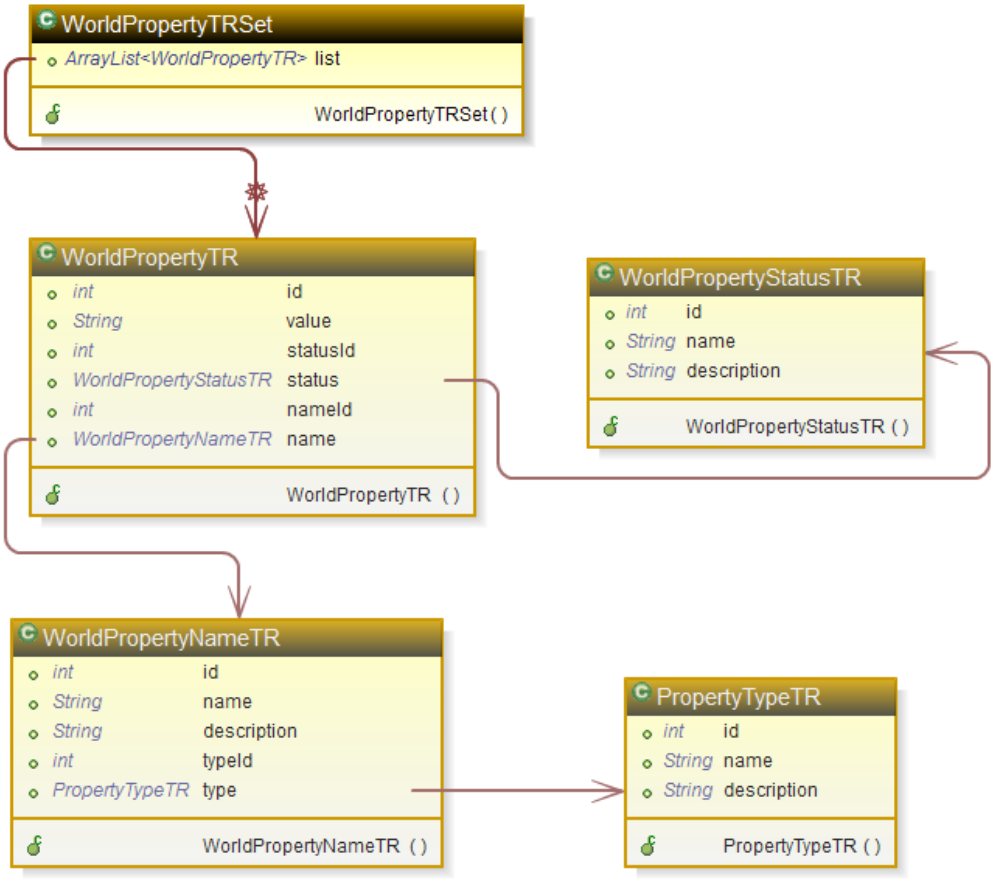
El valor de la propiedad se define en el campo Value de la tabla WorldProperties. Las propiedades se relacionan con el mundo por medio de la tabla WorldAssignedProperties.

Figura 10: Diagrama de ER de propiedades de los mundos



En la Figura 11 se puede observar la representación en clases equivalente al diagrama anterior.

Figura 11: Diagrama de clases de propiedades del mundo



6.1.1.3. Ítems y personajes La relación entre un mundo y sus ítems se especifica utilizando la entidad WorldItems, como se muestra en la Figura 12. La especificación detallada de la entidad Ítems, mostrada en la misma figura, puede encontrarse en el Apéndice 2: Representación de los datos.

Según la definición de un mundo virtual, este puede tener personajes que lo habitan. Los habitantes del mundo son los usuarios conectados, autenticados con el sistema, que hayan escogido entrar al mundo virtual en cuestión. Esto se representa según la entidad LoggedUsers, como se muestra en la Figura 13.

Figura 12: Diagrama de ER de ítems de los mundos

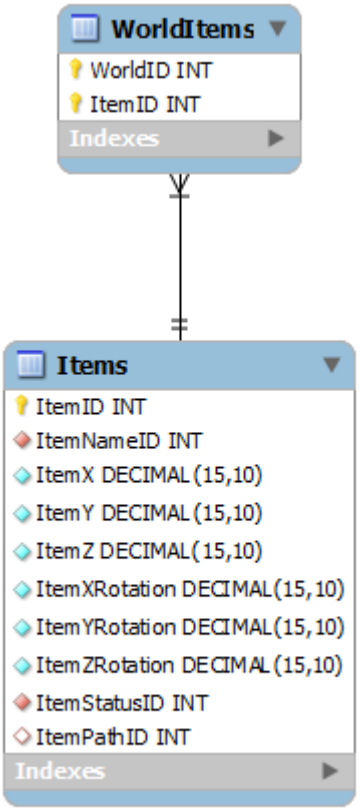
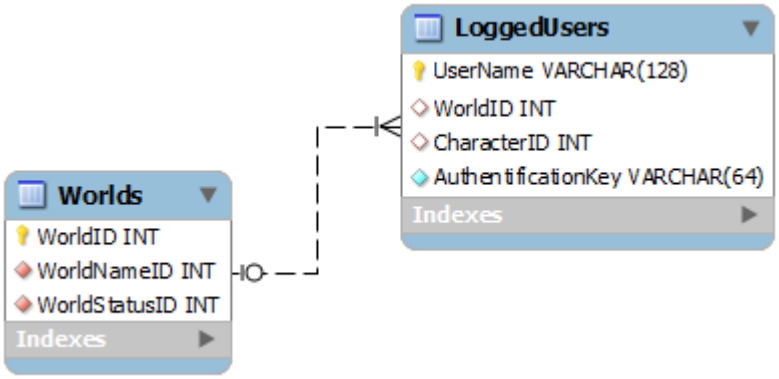


Figura 13: Diagrama de ER de mundos y usuarios conectados



7. IMPLEMENTACIÓN DEL ALMACÉN DE DATOS

Se utilizó MySQL como manejador de bases de datos, por ser un manejador gratuito, de código abierto, comúnmente utilizado en las comunidades de desarrollo de software y por tener amplia documentación y soporte.

La implementación de la funcionalidad del almacén de datos se realizó en el lenguaje de programación Java. Todas las rutinas que involucran el acceso a la base de datos están dentro de la clase NaokDataBase. Dentro de esta clase se definieron funciones básicas para insertar, modificar, retribuir y eliminar datos. Estas funciones básicas son utilizadas para convertir la información entre el manejador de bases de datos y la estructura de clases que representa a los objetos del sistema, definidos por los requerimientos de información en la sección 2.4 y diseñados según el esquema presentado en la sección 2.

8. DISEÑO E IMPLEMENTACIÓN DEL API DEL SERVICIO WEB

En el API del servicio web básicamente encapsula el funcionamiento del almacén de datos y hace que las funcionalidades de este sean accesibles por los clientes que se conectan al servidor. El servicio web fue implementado en una única clase, `NaokWebService`. Esta clase provee el API accesible a través de la red. Toda la interacción con el servicio web se realiza a través de esta clase.

Se crearon dos estructuras de clases equivalentes, según las clases definidas en la sección 6. Estas son: clases de transferencia de datos, a las cuales se les agregó el sufijo TR, ej. `WorldTR` y clases de almacenamiento y uso de datos, a las cuales se les agregó el sufijo DT, ej. `WorldDT`. La interacción con el servicio web se realiza únicamente con las clases de transferencia de datos TR. Las clases DT son utilizadas por los clientes para el manejo interno de la información.

Las demás funciones del servidor se especifican en las secciones siguientes.

9. FUNCIONALIDAD DEL SERVIDOR

Las funcionalidades principales del servidor son: Proveer de un medio de autentificación para que el usuario pueda utilizar su personaje propio, proporcionar a los clientes una forma de acceder al API del servicio web y controlar el movimiento de los objetos dentro del mundo virtual. A continuación se especifican a detalle las funcionalidades mencionadas.

9.1. Autentificación de usuarios (Login)

Para que un usuario pueda utilizar el API del servicio web `NaokWebService` debe de estar autentificado con el sistema. El usuario debe de solicitar la autentificación mediante la llamada a la función `login` del servicio web. Esta función recibe dos parámetros: un objeto `UserCredential`, que se define según el diagrama de clases representado en la figura 14 y una cadena de caracteres que contiene la contraseña del usuario. La firma de la función se presenta a continuación:

```
public UserCredentialTR login(UserCredentialTR userData, String
password)
```

Esta función llama a `login` de la clase `NaokUserManager`, quien es la encargada de realizar el proceso de autentificación.

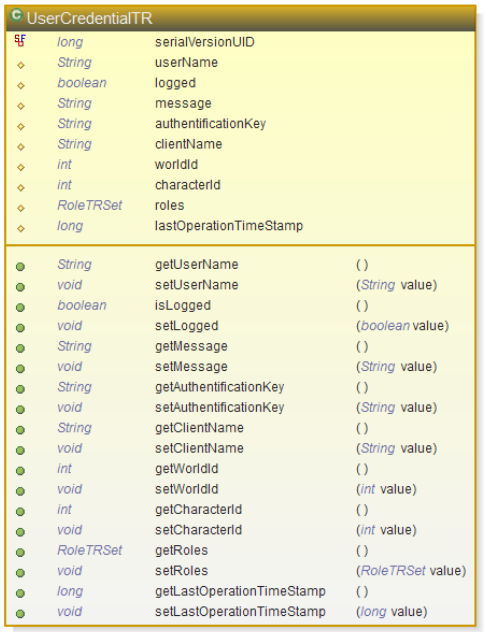
9.1.1. Proceso de autentificación

El proceso de autentificación de usuarios se describe según el siguiente algoritmo:

Entradas de Información:

- Objeto `UserCredential` que contiene el nombre de usuario y una cadena de caracteres que identifica al cliente desde donde se conecta.
- Cadena de caracteres que contiene la contraseña del usuario.

Figura 14: Definición de la clase UserCredentialTR



Salidas de Información:

- Objeto `UserCredential` que contiene el resultado del proceso de autentificación. El resultado puede ser una autentificación positiva o negativa, estados que se representan con los valores verdadero y falso de la variable booleana `logged` de objeto `UserCredential`.

Procedimiento:

- Se verifica que el usuario exista en la base de datos.
- Se verifica la contraseña en la base de datos.
- Si es un usuario válido:
    - Se le asigna el valor verdadero a la variable `logged` del objeto `UserCredential` a retornar.
    - Se genera una cadena de caracteres aleatorios que se asigna a la variable `authentificationKey` del objeto `UserCredential`.
    - El usuario se guarda en la base de datos, en la tabla LoggedUsers.
    - Se obtienen los roles del usuario, de las tablas UserRoles y Roles, y se guardan en el objeto `UserCredential`.

- El `UserCredential` se almacena en el *caché* de usuarios conectados.

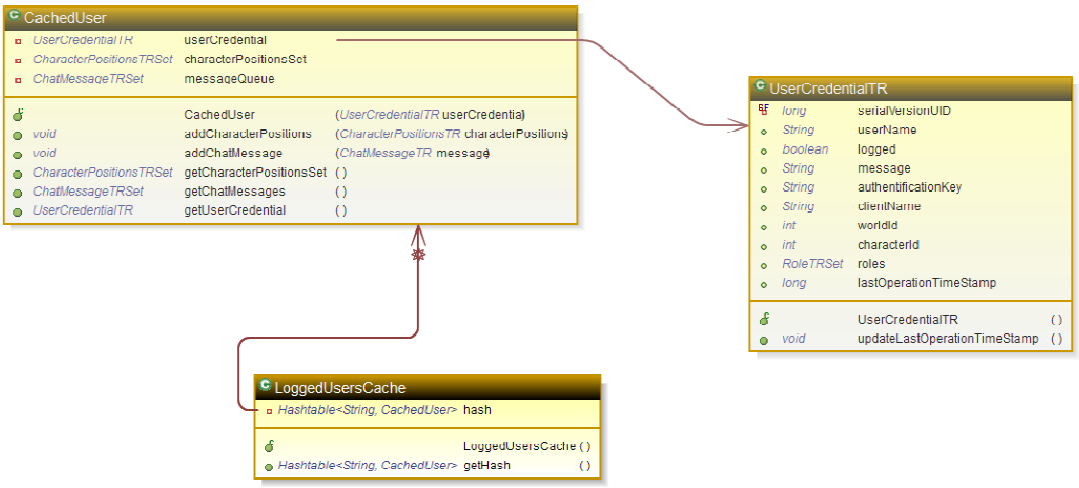
- Si el usuario no es válido:
  - Se le asigna el valor falso a la variable `logged` del objeto `UserCredential` a retornar.
  - Se coloca un mensaje de error en la variable `message` del objeto `UserCredential`.

La funcionalidad del objeto UserCredential retornado, el authentificationKey, el caché de usuarios conectados y los roles de los usuarios se explican en las siguientes secciones.

9.1.2. Cache de usuarios conectados Es un espacio de memoria donde se guardan los usuarios conectados al sistema. Está representado como un HashTable cuya llave es una cadena de caracteres que representa el nombre de usuario y el objeto que almacena es un `CachedUser`.

En el `CachedUser` se guarda un `UserCredential`, y dos listas para almacenar información para el usuario. Estas son: lista de mensajes de chat y lista de posiciones de otros usuarios. El uso de estas listas será especificado en las secciones 10.4.2 y 10.4.3. La relación entre las clases anteriormente mencionadas se muestra en la Figura 15.

Figura 15: Diagrama de clases del caché de usuarios



El caché de usuarios conectados es manejado por el objeto `NaokUserManager`. La ventaja de tener la información en un caché es que se permite el acceso rápido a la misma y se provee de un lugar donde almacenar datos relevantes a la autenticación (`UserCredential`) y datos de actualizaciones para el usuario (listas de posiciones y de mensajes de chat).

9.2. Uso del API del servicio web

Estando autentificado con el sistema el usuario puede utilizar las funciones del API del servicio web. Cada vez que un cliente manda a llamar a una función del servicio web se manda en la llamada un objeto `UserCredential`, el objeto retornado por la función de `login`. Por ejemplo, la función `getWorldName` se define como:

```
public WorldNameTR getWorldName(int worldId, UserCredentialTR
userCredential)
```

El `UserCredential` se utiliza para verificar que el usuario esté conectado y que tenga los privilegios necesarios para realizar la operación, de la siguiente manera:

9.2.1. Proceso de verificación de privilegios

Entradas de información:

- `UserCredential` que se envía en la solicitud de información.
- Tipo de operación a realizar. Determinado según el nombre de la función, en este caso `getWorldName`.

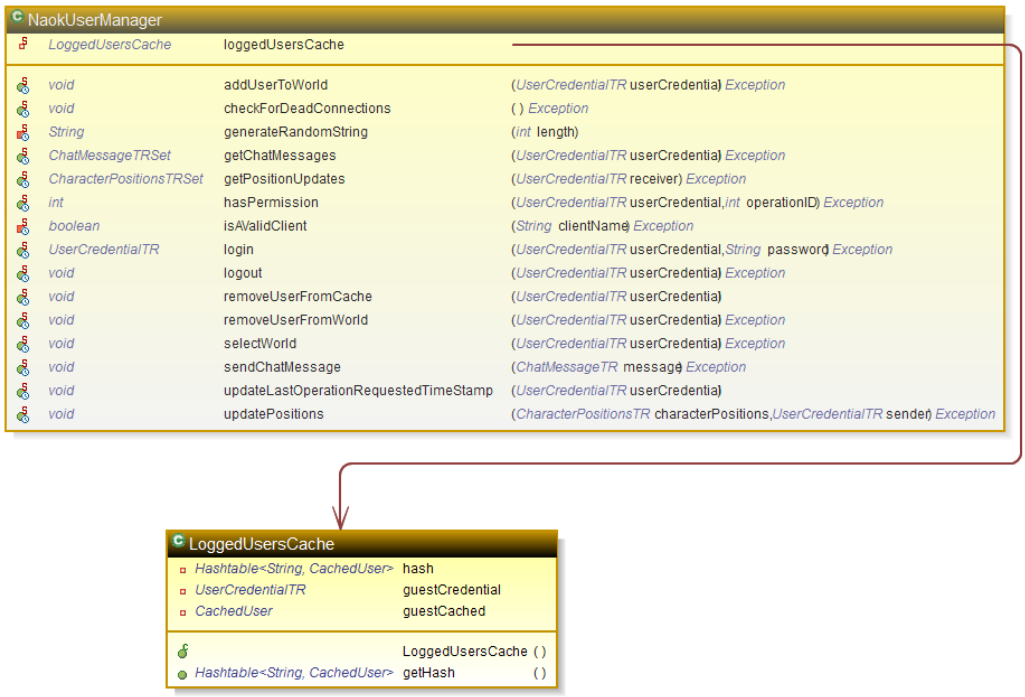
Salidas de información:

- Código de resultado. Indica si el usuario tiene permiso de realizar la operación o el código de error correspondiente.

Proceso:

1. Se chequea que el usuario se encuentre en el caché de usuarios. De esta manera se revisa que el usuario haya pasado por el proceso de autentificación.

Figura 16: Relación entre el caché de usuarios y la clase NaokUserManager



2. Se revisa que el `authentificationKey` que se encuentra en el `UserCredential` de entrada sea el mismo que el `authentificationKey` del `UserCredential` almacenado en el caché de usuarios.
3. Se obtienen los roles que tienen permiso de realizar la operación. Estos son obtenidos de las tablas Roles, OperationRoles y Operations de la base de datos.
4. Se revisa que el usuario tenga uno de los roles con permiso, comparando los roles obtenidos en el paso 3 con los roles del usuario, que es encuentran en el `UserCredential` del caché.
5. Se asigna el código de resultado según la conclusión de la verificación.

9.2.2. Especificación de roles y operaciones permitidas

Aunque los tipos de usuarios se definieron como administrador del sistema y usuario general, se agregó un rol extra, con la idea de agregar un nuevo tipo de usuario a futuro, el administrador de mundos. Los roles definidos son:

- Super User: Administrador general del sistema.
- World Admin: Administrador y editor de mundos virtuales.
- User: Usuario general.
- Everyone: Usuarios no registrados en el sistema.

Las operaciones que se definen son:

- Write Base Catalog: Modificar catálogos de funcionalidad del sistema.
- Write User Data: Modificar catálogos de datos de usuarios.
- Write World Data: Modificar catálogos que contienen la definición de los mundos virtuales.
- Read Base Catalog: Obtener datos de catálogos de funcionalidad del sistema.
- Read User Data: Obtener datos de los catálogos de usuarios.
- Read World Data: Obtener datos de los catálogos de mundos virtuales.

Las operaciones permitidas para cada rol se definen en el Cuadro 2 y el Cuadro 3 donde una operación permitida se identifica con un 1 y una operación no permitida se identifica con un 0.

Cuadro 2: Tipos de usuario y operaciones de escritura permitidas

Usuarios/Operaciones	WriteBaseCatalog	WriteUserData	WriteWorldData
SuperUser	1	1	1
WorldAdmin	0	1	1
User	0	1	0
Everyone	0	0	0

Cuadro 3: Tipos de usuario y operaciones de lectura permitidas

Usuarios/Operaciones	ReadBaseCatalog	ReadUserData	ReadWorldData
SuperUser	1	1	1
WorldAdmin	1	1	1
User	1	1	1
Everyone	1	1	1

Cada función del API expuesta por el servicio web se cataloga como un tipo de operación, según el Cuadro 4.

Cuadro 4: Funciones del API NaokWebService catalogadas como tipo de operación

	Add/Update/Remove	Get
Action	writeWorldData	readWorldData
ActionParameter	writeWorldData	readWorldData
ActionParameterName	writeWorldData	readWorldData
ActionStatus	writeWorldData	readWorldData
Character	writeUserData	readUserData
CharacterAssignedProperty	writeUserData	readUserData
CharacterItem	writeUserData	readUserData
CharacterPosition	notAvailable	readUserData
CharacterProperty	writeWorldData	readWorldData
CharacterPropertyName	writeWorldData	readWorldData
CharacterPropertyStatus	writeWorldData	readWorldData
CharacterQuest	writeUserData	readUserData
CharacterQuestEvent	writeUserData	readUserData
CharacterStatus	writeWorldData	readUserData
Country	writeBaseCatalog	readBaseCatalog
Event	writeWorldData	readWorldData
EventAction	writeWorldData	readWorldData
EventParameter	writeWorldData	readWorldData
EventParameterName	writeWorldData	readWorldData
EventStatus	writeWorldData	readWorldData
EventType	writeWorldData	readWorldData
Gender	writeBaseCatalog	readBaseCatalog
Item	writeWorldData	readWorldData
ItemAssignedProperty	writeWorldData	readWorldData
ItemEvent	writeWorldData	readWorldData
ItemName	writeWorldData	readWorldData
ItemPath	writeWorldData	readWorldData
ItemPathName	writeWorldData	readWorldData
ItemPathPosition	writeWorldData	readWorldData
ItemPathStatus	writeWorldData	readWorldData

Continuación Cuadro 4: Funciones del API NaokWebService catalogadas como tipo de operación

	Add/Update/Remove	Get
ItemProperty	writeWorldData	readWorldData
ItemPropertyName	writeWorldData	readWorldData
ItemPropertyStatus	writeWorldData	readWorldData
ItemStatus	writeWorldData	readWorldData
Operation	writeBaseCatalog	notAvailable
OperationRole	writeBaseCatalog	notAvailable
OperationType	writeWorldData	readWorldData
ParameterType	writeWorldData	readWorldData
PropertyType	writeWorldData	readWorldData
Quest	writeWorldData	readWorldData
QuestEvent	writeWorldData	readWorldData
QuestItemReward	writeWorldData	readWorldData
QuestName	writeWorldData	readWorldData
QuestPropertyReward	writeWorldData	readWorldData
QuestStatus	writeWorldData	readWorldData
Role	writeBaseCatalog	notAvailable
User	writeUserData	readUserData
UserCharacter	writeUserData	readUserData
UserRole	notAvailable	readUserData
UserStatus	writeBaseCatalog	readUserData
World	writeWorldData	readWorldData
WorldAssignedProperty	writeWorldData	readWorldData
WorldItem	writeWorldData	readWorldData
WorldName	writeWorldData	readWorldData
WorldProperty	writeWorldData	readWorldData
WorldPropertyName	writeWorldData	readWorldData
WorldPropertyStatus	writeWorldData	readWorldData
WorldStatus	writeWorldData	readWorldData

9.3. Control de movimiento de entidades dentro del mundo

El servidor es responsable de controlar los ítems y NPC's que se mueven dentro de una trayectoria definida en el mundo virtual. Para esto se tiene en memoria un caché de mundos donde a su vez se tiene un caché de ítems, representado por los objetos de la Figura 17.

La posición de los ítems en el caché es calculada constantemente por un hilo de procesamiento (thread) en el servidor. El hilo se define en la clase `ItemPositionsThread`. Cada vez que se cambia la posición del objeto, este es actualizado dentro del caché de mundos.

El hilo y el caché de mundos están definidos dentro de la clase **`WorldManager`**, y se representan según la

Figura 18.

Figura 17: Diagrama de clases del caché de mundos

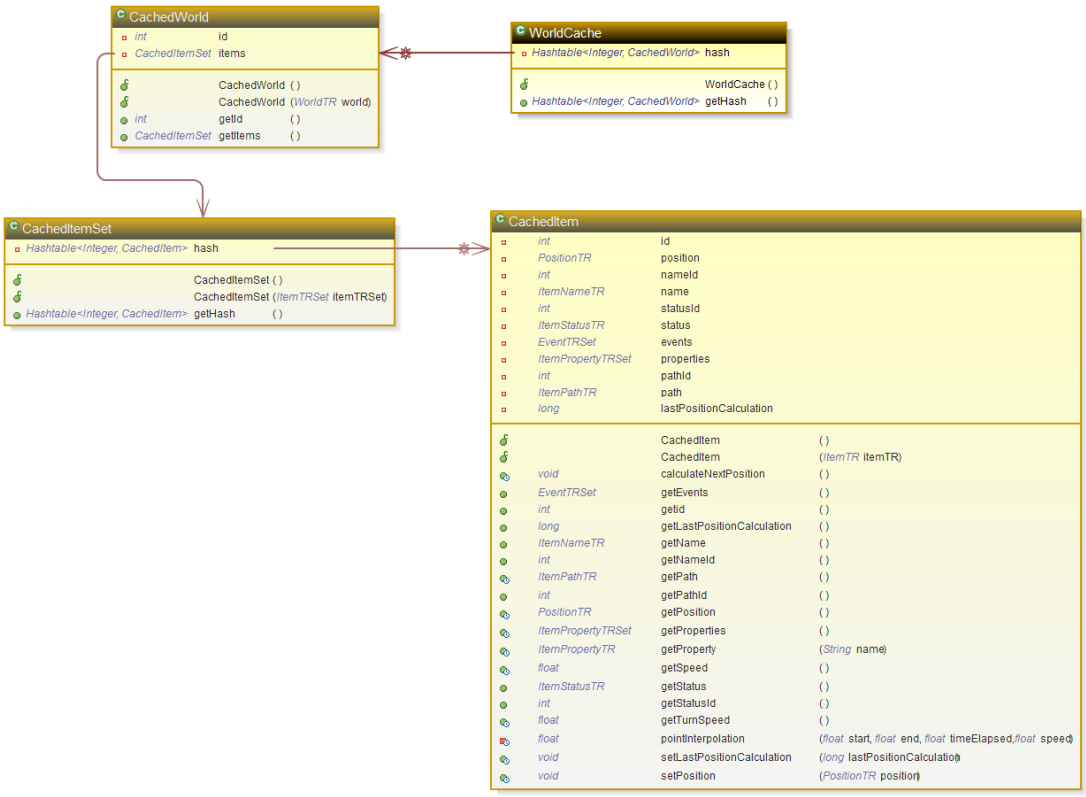
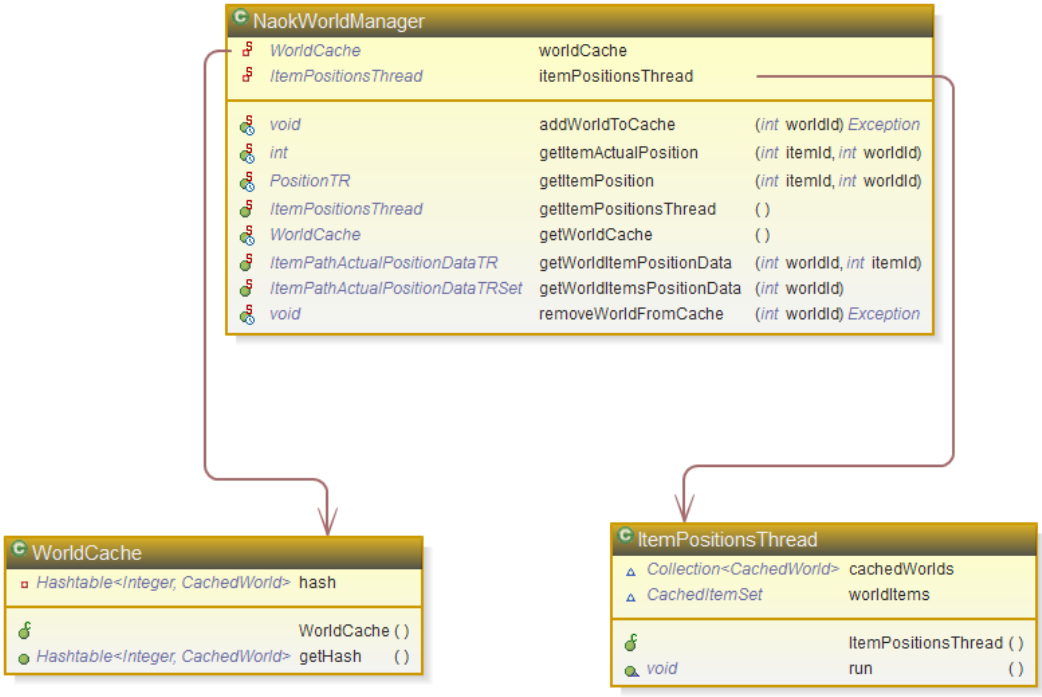


Figura 18: Diagrama de clases de NaokWorldManager



10. CICLO DE VIDA DEL USUARIO DENTRO DE UN MUNDO VIRTUAL

El ciclo de vida de un usuario dentro del sistema se representa según la Figura 19 y se detalla en las secciones que continúan.

Figura 19: Ciclo de vida del usuario dentro del sistema

10.1. Inicio de sesión (login)

Para iniciar una sesión en el sistema el usuario debe de autentificarse, la explicación de este proceso se encuentra en la sección 9.1. El resultado de la autentificación es un objeto `UserCredential` que se utiliza durante toda la sesión para hacer llamadas al servidor.

10.2. Selección de personaje

Antes de seleccionar un personaje un usuario debe de poder saber qué personajes tiene disponibles. Para esto basta con llamar a la función `getUserCharacters` del `NaokWebService`. Sabiendo ya qué personajes tiene disponibles, el usuario puede escoger uno llamando a la función `selectCharacter` del mismo servicio web. Esta función actualiza la tabla LoggedUsers de la base de datos, con el personaje seleccionado.

10.3. Selección de mundo

Al igual que en la selección de un personaje, el usuario debe saber qué mundos se encuentran disponibles. Para esto se llama a la función `getWorlds` del `NaokWebService` que regresa un listado de mundos. De los mundos del listado, el usuario puede seleccionar uno llamando a la función `selectWorld` de `NaokWebService`. La función selectWorld procesa la solicitud de la siguiente manera.

1. Chequea que el mundo esté en el caché de mundos activos. Si no fuera así, se agrega el mundo al caché y se activa el hilo que calcula las posiciones de los objetos del mundo que lo necesiten.
2. Actualiza la tabla LoggedUsers en la base de datos con el mundo seleccionado por el personaje.
3. Notifica a los usuarios que se encuentren en el mundo virtual seleccionado, que un usuario nuevo ha ingresado. Ver sección 10.4.5.
4. Si el usuario estuviera cambiándose de mundo virtual, se notifica a los habitantes del mundo abandonado que el usuario lo ha dejado. Si no hubieran habitantes dentro del mundo, se remueve el mundo del caché de mundos activos.

10.4. Dentro del mundo

10.4.1. Solicitud de información del mundo Al ingresar a un mundo virtual, el usuario necesita obtener la información acerca de las propiedades, ítems y personajes del nuevo mundo. Para esto debe de llamar a la función `getWorld` del `NaokWebService`. Esta función regresa un objeto de tipo `WorldTR`. El cliente es responsable de convertir este objeto a un `WorldDT` y de obtener las propiedades, ítems y personajes que se encuentren dentro de él

cuando los necesite, mediante las funciones `getWorldAssignedProperties`, `getWorldCharacters` y `getWorldItems`. De la misma manera el cliente debe de obtener las propiedades y demás información de los personajes e ítems en el mundo.

10.4.2. Actualización de posiciones de personajes Un avatar dentro de un mundo virtual debe de ser capaz de conocer las acciones de otros personajes, dentro de las que se encuentran la posición en el mundo y animación activa. Este problema se resuelve al hacer que cada uno de los avatares mande constantemente esta información al servidor y haciendo al servidor responsable de distribuir la misma a los demás usuarios conectados en el mismo mundo.

Para simplificar la operación las posiciones de un personaje se guardan en una cola de posiciones. Cuando otro usuario lee las posiciones del personaje, estas se eliminan de la cola para dar paso a nueva información.

El problema con esta implementación es que cada uno de los usuarios debe de tener su propia cola de posiciones para todos los demás usuarios dentro del mundo. Ya que la información es eliminada cuando se consume. Esto puede verse en la Figura 20, la clase que representa la lista es `CharacterPositionTRSet`.

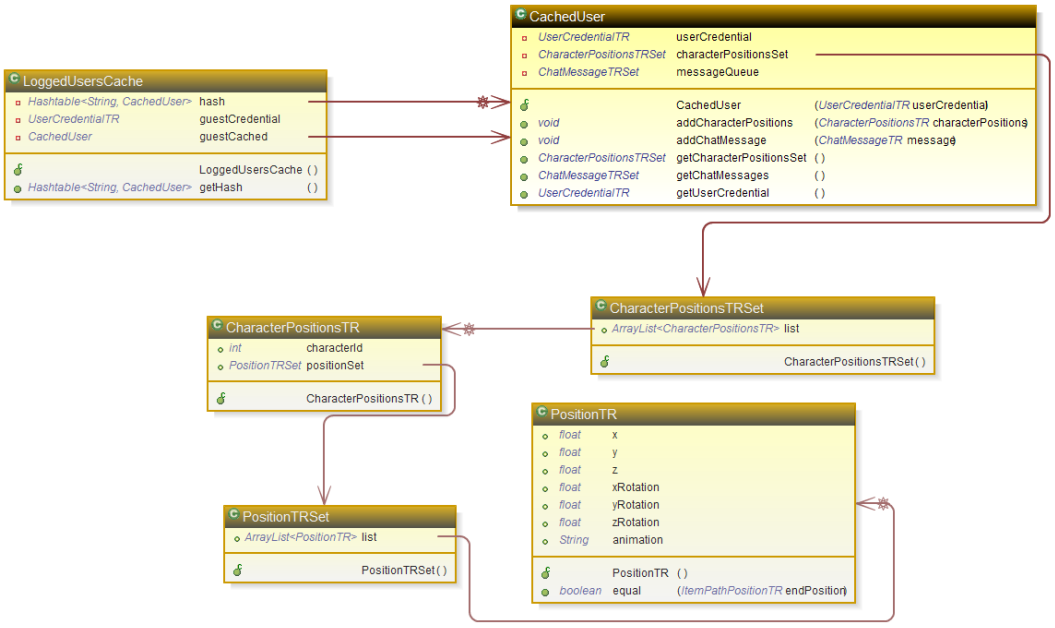
De otra manera, teniendo una lista de posiciones única para cada usuario se complicaría el problema al tener que estar llevando un control de cuales posiciones ya han sido leídas por todos los usuarios para poder eliminarlas. Las posiciones deben eliminarse constantemente ya que la frecuencia con que se actualizan es muy alta y dejar toda esa información en memoria puede sobrecargar el servidor.

Además tener una lista como `CharacterPositionTRSet` facilita la operación de obtener las actualizaciones de todos los usuarios, ya que toda la información que el cliente necesita se encuentra en un solo lugar.

El procedimiento que se sigue para actualizar un conjunto de posiciones es el siguiente:

1. El cliente envía sus posiciones al servidor.
2. Al recibir la actualización el servidor busca los usuarios conectados en el mundo en la tabla LoggedUsers de la base de datos.

Figura 20: Diagrama de clases de CharacterPositionTRSet



3. Cada uno de los usuarios encontrados en el mundo se busca en el caché de usuarios conectados.
4. En cada uno de los usuarios encontrados en el caché se busca el registro (`CharacterPositionTR`) para el cliente que envió la actualización de posiciones en el `CharacterPositionTRSet` y se agregan a él las actualizaciones.

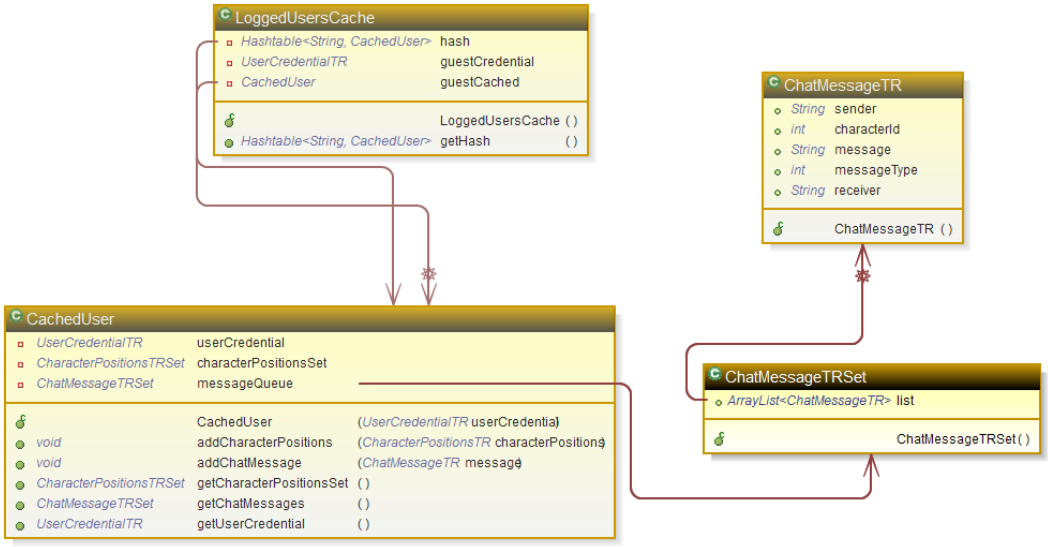
Para pedir las actualizaciones,

1. El cliente solicita las actualizaciones de todos los personajes en el mundo.
2. El servidor busca el registro correspondiente al usuario en el caché de usuarios conectados y regresa una copia del objeto `CharacterPositionTRSet` que se encuentra dentro del caché.
3. El servidor elimina los datos del `CharacterPositionTRSet` del caché.

10.4.3. Chat De la misma manera que se tiene el `CharacterPositionTRSet` dentro del `CachedUser` se tiene también una cola para almacenar mensajes de chat. Esta vez no es necesario tener una cola separada por cada usuario, ya que cada mensaje tiene la información

del usuario que lo envió. La cola de mensajes de chat `messageQueue` también es una cola temporal de mensajes, donde los mensajes leídos son eliminados.

Figura 21: Diagrama de clases de ChatMessageTRSet



El proceso para enviar y recibir mensajes de chat es el siguiente:

Para enviar:

1. El usuario llama a la función `sendChatMessage` del `NaokWebService`, pasándole de parámetro un objeto `ChatMessageTR`.
2. El servidor busca el usuario destinatario en el caché de usuarios conectados. Al encontrarlo agrega el mensaje de chat a su cola personal de mensajes.

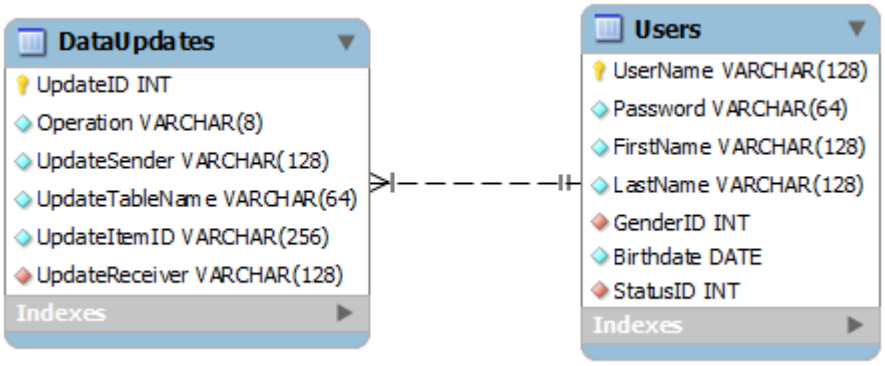
Para recibir:

1. El usuario llama a la función `getChatMessages` del `NaokWebService`.
2. El servidor regresa un `ChatMessageTRSet` que contiene todos los mensajes en la cola del usuario.
3. El servidor limpia la cola de mensajes.

10.4.4. Actualización de posiciones de ítems El servidor controla la posición de los ítems que cumplen una trayectoria dentro del mundo. La implementación de este procedimiento se plantea en la sección 9.3. Para recibir las actualizaciones, el cliente debe de llamar a la función `getWorldItemsPositionData` del `NaokWebService`. Al recibir la llamada el servidor retorna un listado items con la posición en la que se encuentran.

10.4.5. Actualización de información de otros usuarios Cada vez que un usuario entra o sale de un mundo virtual, cambia una propiedad o cambia un ítem propio, el sistema debe informar al resto de usuarios que se encuentren dentro del mundo. Estas actualizaciones no son tan frecuentes como las de posiciones, por lo que se utiliza una tabla en la base de datos para guardar los mensajes de actualización. La tabla es DataUpdates y el diagrama puede verse en la Figura 22.

Figura 22: Diagrama de ER de DataUpdates



El procedimiento para el control de las actualizaciones es el siguiente:

1. El servidor recibe la solicitud para entrar o salir de un mundo virtual, cambio de una propiedad o ítem en el mundo, personaje u otro ítem.
2. El servidor busca los usuarios conectados en el mundo en el que se realizó el cambio en la tabla LoggedUsers de la base de datos.
3. Para cada uno de los usuarios se ingresa un registro en la tabla de DataUpdates, que contiene la operación a realizar, que puede ser agregar (ADD), eliminar (REMOVE) o actualizar (UPDATE), el nombre del usuario que realizó el cambio (UpdateSender), el nombre de la tabla en donde se realizó la actualización (UpdateTableName), el

identificador del registro en la tabla que se modificó (UpdateItemID) y el nombre del usuario que recibirá el mensaje (UpdateReceiver).

El cliente pide un conjunto de actualizaciones llamando a la función `getUpdates` del `NaokWebService`:

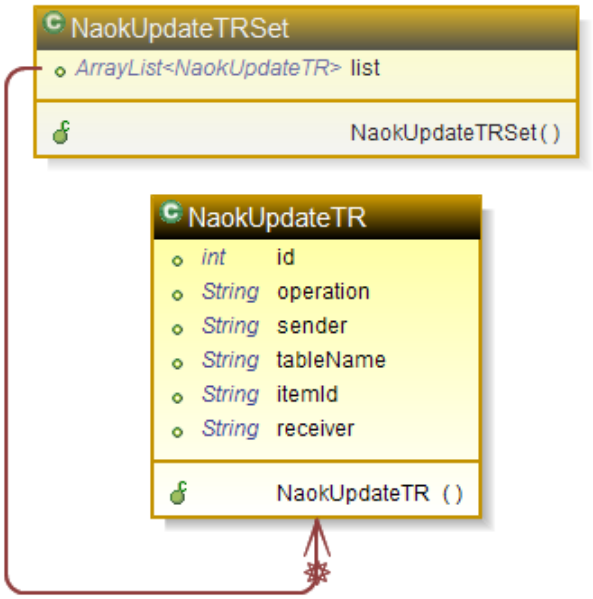
1. El servidor hace una consulta en la tabla DataUpdates de la base de datos, acerca de las actualizaciones pendientes para el usuario.
2. Con el conjunto de actualizaciones se guarda en un objeto `NaokUpdateTRSet`, representado en la Figura 23.
3. El servidor elimina las actualizaciones leídas de la tabla. Esto se hace en una transacción junto con la lectura del paso 1.

El cliente debe interpretar los mensajes actualizaciones que se encuentran en `NaokUpdateTRSet`, ya que estos sólo contienen el nombre de la tabla y el identificador del ítem actualizado y no la actualización en sí. Por ejemplo, si se actualizara una propiedad del mundo, el objeto `NaokUpdateTR` recibido tendría en la variable `operation` la cadena de caracteres “UPDATE”, en la variable `tableName` el valor “WorldProperties” y en la variable itemId el identificador de la propiedad, por ejemplo 5. Con esta información el cliente debe de solicitar la nueva propiedad llamando a la función `getWorldAssignedProperty` con el identificador de la propiedad como parámetro.

10.5. Cierre de sesión (logout)

Cuando el usuario se desconecta voluntariamente del sistema, debe llamar a la función `logout` de `NaokUserManager`, expuesta a través del API de servicio web `NaokWebService`. Dicha función borra la información del usuario del caché y de la tabla LoggedUsers en la base de datos.

Figura 23: Definición de NaokUpdateTR



11. CLIENTE ADMINISTRADOR

El cliente administrador es una interfaz gráfica que interactúa con el servidor y permite al usuario administrador crear y manipular mundos virtuales mediante el manejo de la información de éstos.

Al igual que el cliente gráfico, el cliente administrador debe poder conectarse al servidor e intercambiar información con este. Para este fin se creó la biblioteca `ClientLibrary`, que sirve como intermediaria entre los clientes y el servidor. Esta biblioteca provee la conexión con el servidor por medio de la clase `ServerConnection` y hace las transformaciones de los datos entre clases TR y DT. Dejando listos los objetos DT para que el cliente pueda utilizarlos. La funcionalidad de esta biblioteca se muestra en la Figura 24.

Para la realización de la interfaz gráfica se utilizó Jigloo. Esta herramienta es una extensión para el ambiente de desarrollo Eclipse que permite crear interfaces gráficas de forma visual, generando el código necesario para que estas sean ejecutadas. Jigloo genera código para Swing y SWT, que son dos API's que se utilizan para el desarrollo de interfaces gráficas en Java. Se utilizó Jigloo debido a que se integra fácilmente con el ambiente de desarrollo utilizado y es gratuita para el uso en aplicaciones no comerciales.

El cliente administrador provee interfaces gráficas para manipular cada uno de los siguientes objetos: usuarios, mundos, ítems, personajes, misiones, eventos y acciones. En la Figura 25 se muestra la ventana de mundos como ejemplo de las interfaces realizadas.

Figura 24: Relación entre los componentes del sistema

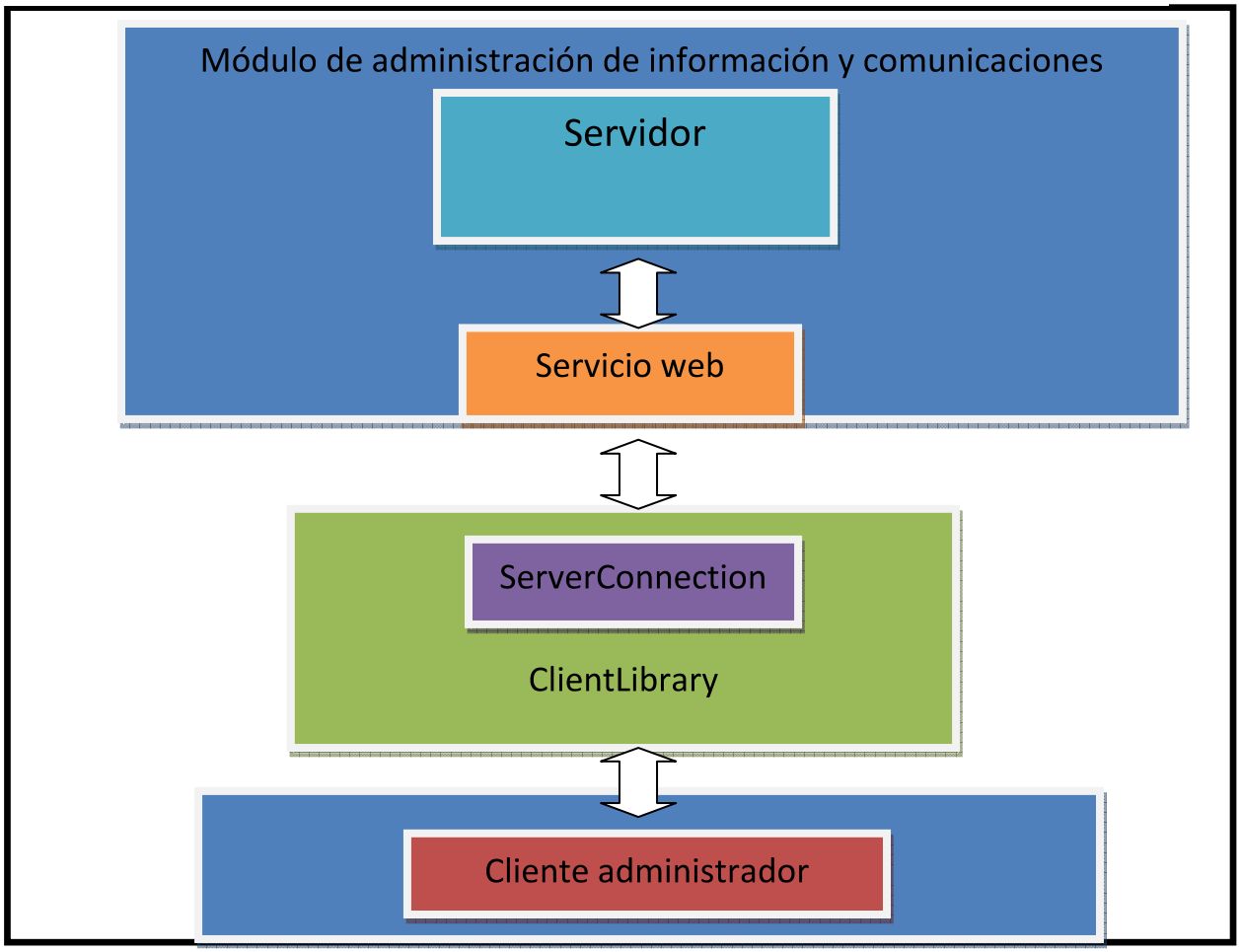
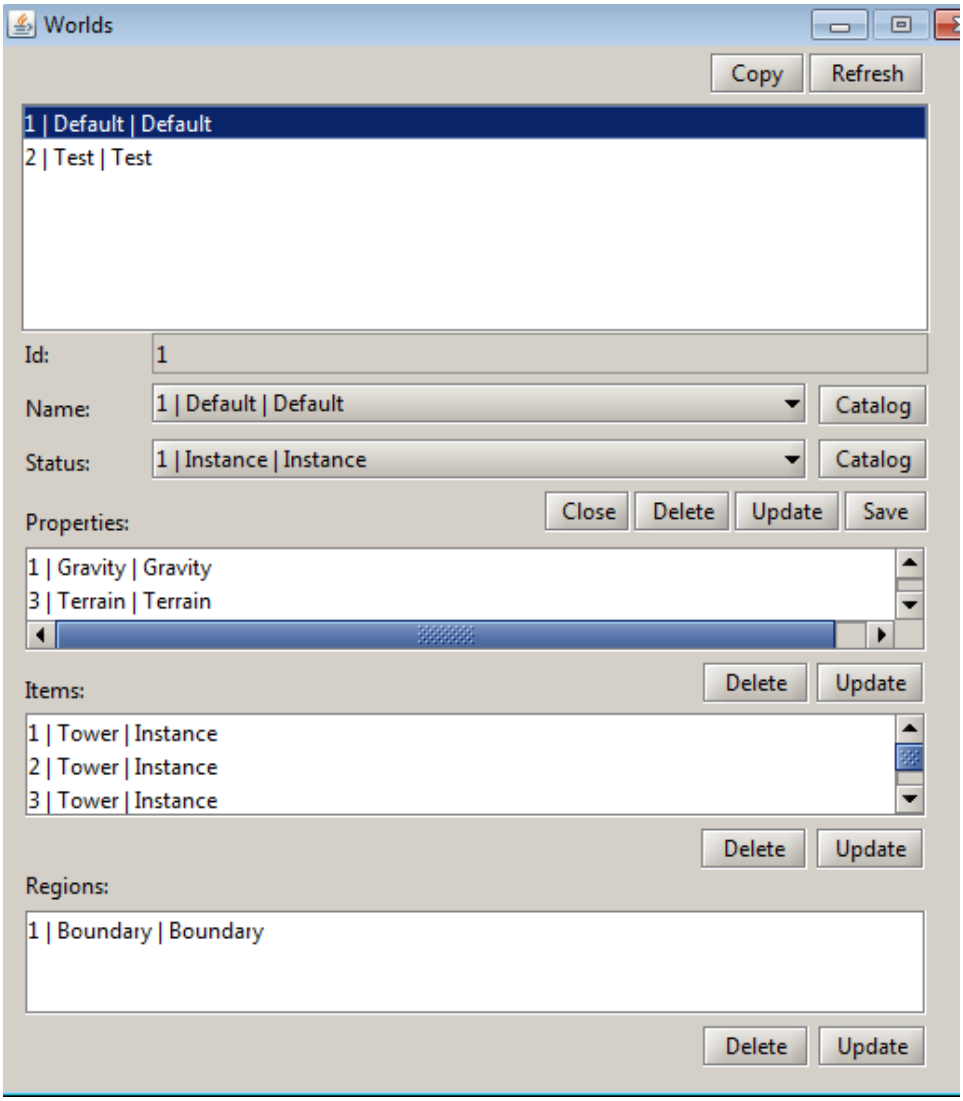


Figura 25: Ejemplo de las interfaces gráficas creadas en el cliente administrador



12. RESULTADOS

Al finalizar este trabajo se tuvo como resultado una definición teórica de un mundo virtual, según su descripción en la sección 2 Especificación del Sistema. Y la elaboración de una herramienta de software capaz de crear mundos virtuales mediante esta definición teórica.

La creación de los mundos virtuales se realiza a través del cliente administrador. Este software pide al usuario los datos necesarios para construir un mundo y se comunica con el servidor para guardar los datos que definen al mundo virtual en el almacén diseñado específicamente para esta función.

Los mundos virtuales se utilizan mediante la ejecución de código en el servidor, como se define en la sección 9 y mediante la utilización de un servicio web, que permite que el cliente gráfico utilice los datos que definen a un mundo virtual.

Para demostrar la funcionalidad de este módulo y su integración con el Módulo de gráficas y simulación se diseñó y creó un mundo virtual de pruebas, el cual fue utilizado en la presentación de este trabajo de graduación.

13. DISCUSIÓN

El módulo de administración de la información y comunicaciones cumplió con todos los objetivos planteados y con las necesidades de la prueba escogida. Pero aunque se obtuvo un rendimiento satisfactorio según los objetivos y la prueba, este no necesariamente es el adecuado para utilizar el sistema de forma definitiva.

El servidor se implementó utilizando servicios web debido a la facilidad que este enfoque provee en la etapa de desarrollo del sistema al abstraer la complejidad de la programación para redes, permitiendo que la funcionalidad del servidor se programe como un API común (ver Cuadro 1).

Para montar un servicio web se necesita de un servidor de aplicación. En este caso se utiliza Glassfish, un servidor desarrollado por Sun Microsystems, y su implementación del marco de desarrollo para servicios web llamada Metro. Se escogió esta alternativa debido a que Glassfish es un servidor especialmente diseñado para utilizarse con el lenguaje de programación Java, es multiplataforma además de ser gratuito y de código abierto.

Es posible que la implementación de un servicio web tenga un sobrecosto demasiado alto en la transferencia de la información, debido al estándar que hay que cumplir de envío de mensajes según SOAP y utilizando XML. Aunque no se midió el desempeño del sistema en cuanto a transferencia de información, fue notoria la diferencia al tener el cliente y el servidor conectados en una red local comparado con la conexión de los mismos a través de internet. Si se quiere implementar el sistema para una aplicación real debe de realizarse una evaluación del desempeño, tomando en cuenta la capacidad del equipo en el cual se piensa ejecutar y la cantidad de usuarios que se desea tener. Si el desempeño fuera muy pobre se deben determinar las causas de este comportamiento y diseñar mejoras que permitan un rendimiento adecuado. Para el almacenamiento de datos se utilizó el manejador MySQL, debido a que es una herramienta gratuita, de código abierto, de sólido prestigio, bien documentada y con una con una gran comunidad de usuarios que lo respalda.

Al hacer las pruebas de implementación del sistema debe de evaluarse también si la cantidad de consultas a la base de datos está teniendo un impacto en el desempeño. Al revisar el programa para la elaboración de este documento, se pudo notar que existen algunas rutinas que utilizan la base de datos que podrían implementarse en un caché de memoria.

Una de estas rutinas es la obtención de roles y operaciones permitidas en el proceso de verificación de privilegios (sección 9.2.1). Este procedimiento se utiliza cada vez que un usuario realiza una llamada al sistema, es una función utilizada muy frecuentemente, y cada vez que se ejecuta hace una lectura a la base de datos para obtener la información necesaria. Los datos de privilegios no varían mucho durante el tiempo, por lo que la mayoría de veces se estaría leyendo la misma información de la base de datos cada vez que el proceso es ejecutado. Esta información podría leerse una vez al iniciar el sistema y tenerse en memoria para futuras referencias.

Otra de las rutinas que puede optimizarse al tener la información en memoria es la que determina qué usuarios están conectados a un determinado mundo. En este momento la información se obtiene de la tabla LoggedUsers en la base de datos, pero podría obtenerse del caché de usuarios conectados, o incluso tener una referencia a los usuarios en el caché de mundos. Este es un cambio fácil de realizar en el código, que podría mejorar el desempeño.

Un aspecto que hay que mejorar en el sistema es la seguridad del mismo. Por el momento las contraseñas de los usuarios se almacenan como texto claro y deberían de almacenarse por lo menos codificadas según algún algoritmo de encriptamiento. Otro problema de seguridad es que los usuarios que tienen el permiso WriteUserData (escribir datos de usuario) por el momento pueden modificar la información de cualquier usuario del sistema, cuando deberían de poder modificar solamente la información propia. Estos problemas son simples de detectar, pero como esta pueden existir otros que amenacen la seguridad del sistema. Se recomienda realizar un estudio de vulnerabilidades y amenazas y atacar los problemas encontrados.

La interfaz gráfica del cliente administrador podría mejorarse, para esto se debe realizar un estudio adecuado de usabilidad de la aplicación y diseñar cambios en la interfaz que permitan una mejor interacción con el usuario.

Se debe tomar en cuenta que la parte de seguridad y de diseño de interfaces gráficas están fuera del alcance de este proyecto pero se sugiere tomar en cuenta estos aspectos si se quiere implementar el sistema en un ambiente real.

Se cree que en este momento la aplicación está lista para empezar a implementar las funcionalidades necesarias para que pueda llegar a ser una herramienta educativa.

14. CONCLUSIONES

Se creó el almacén de datos para el servidor utilizando el manejador de bases de datos MySQL. Se programó la funcionalidad necesaria para poder administrar los datos utilizando el lenguaje de programación Java.

Se utilizó un servicio web para permitir la comunicación entre el servidor y los clientes. El servicio web se implementó utilizando el servidor de aplicación Glassfish y el marco de desarrollo para servicios web Metro.

Se crearon las interfaces gráficas necesarias para el manejo de la información en el cliente administrador, mediante el uso de la herramienta Jigloo que genera código para Swing y SWT.

Se creó un mundo virtual de pruebas mediante el uso del cliente administrador. El sistema permitió que el mundo virtual de pruebas fuera definido correctamente, pudiendo ingresar, modificar, eliminar y retribuir su información utilizando el cliente creado específicamente para esto.

La interacción con los clientes (administrador y gráfico) y el servidor fue posible, al proveer una biblioteca intermediaria para el manejo de la conexión con el servicio web. Esta biblioteca es la ClientLibrary, de la cual se habla en la sección 11.

La interacción entre varios usuarios dentro del mismo mundo virtual fue posible debido al uso adecuado de las rutinas de actualización de datos en el cliente. En resumen, se cumplieron los objetivos propuestos para la aplicación.

15. RECOMENDACIONES

- Se recomienda realizar una evaluación de desempeño en el ambiente en el que se desee utilizar el sistema y adecuar el mismo a las necesidades de los usuarios.
- Si se llegara a determinar que la implementación de servicios web es la que afecta al rendimiento del sistema, se puede crear un servidor propio del sistema utilizando sockets o evaluar alternativas de servidores especializados para mundos virtuales.
- Se recomienda evaluar la funcionalidad del sistema, tratar de minimizar el acceso a la base de datos en operaciones recurrentes, utilizando un caché en memoria RAM.
- Si se desea darle uso al sistema para una aplicación real se debe de tomar en cuenta la seguridad del mismo, hacer una evaluación de vulnerabilidades y amenazas y resolver los problemas encontrados.
- Para que el sistema cumpla con la funcionalidad educativa para la que se pensó inicialmente, se debe solicitar la asesoría de expertos en el campo de la educación y continuar su desarrollo.

16. BIBLIOGRAFÍA

Bartle, Richard. *Designing Virtual Worlds.* New Riders Publishing, 2003.

Cerami, Ethan. *Web Services Essentials.* O'Reilly Media, Inc., 2002.

Cherbakov, Luba, Robi Brunner, Rob Smart, y Charisse Lu. *Virtual Spaces: Enabling Immersive Collaborative Enterprise.* 30 de Junio de 2009. http://www.ibm.com/developerworks/webservices/library/ws-virtualspaces/index.html (último acceso: 13 de Agosto de 2010).

Date, C. J. *An Introduction to database systems.* Addison Wesley, 2000.

Gay, Warren W. *Linux Socket Programming by Example.* Que, 2000.

Gee, James Paul. «Why are Video Games Good for Learning?» *Academic Advanced Distributed Learning Co-Lab.* http://www.academiccolab.org/resources/documents/MacArthur.pdf (último acceso: 16 de Septiembre de 2010).

Gollub, Rachel. «Second Life and Education.» *Crossroads* (ACM) 14, nº 1 (2007).

Harold, Elliotte Rusty. *Java Network Programming.* 3ra. O'Reilly Media, Inc., 2004.

Kemp, Jeremy, y Daniel Livingstone. «Putting a Second Life Metaverse Skin on Learning Management Systems.» *Second Life Education Workshop 2006.* San Francisco, 2006.

Mason, Hilary. «Experiential Education in Second Life.» *Second Life Education Workshop 2007.* 2007.

Mayo, Merrilea J. «Video Games: A Route to Large-Scale STEM Education?» *Science Magazine*, nº 323 (Enero 2009).

Mitchel, Don. *From MUD's to Virtual Worlds.* 23 de Marzo de 1995. http://www.mentallandscape.com/Papers_95vworlds.htm (último acceso: 12 de Agosto de 2010).

Orfali, Robert, Dan Harkey, y Jeri Edwards. *Client/Server Survival Guide.* 3ra. John Wiley & Sons, 1999.

Peachey, Anna, Julia Gillen, Livingstone Daniel, y Sarah Smith-Robins, . *Researching Learning in Virtual Worlds.* Springer London, 2010.

Sommerville, Ian. *Ingeniería de Software.* 6ta. Addison Wesley, 2002.

Tanenbaum, Andrew S. *Computer Networks.* 4ta. Prentice Hall, 2003.

Thaller, Michelle. *Whyville: the place girls love to go for science.* 16 de Agosto de 2002. http://www.csmonitor.com/2002/0816/p25s01-lecs.html (último acceso: 11 de Agosto de 2010).

W3C Working Group. *Web Services Architecture.* 11 de Febrero de 2004. http://www.w3.org/TR/ws-arch (último acceso: 06 de Abril de 2010).

Walton, Sean. *Linux Socket Programming.* Sams, 2001.

17. APÉNDICE 1: GLOSARIO

API: Application Program Interface. Es una interfaz (conjunto de funciones) que ofrece una biblioteca para ser utilizada por otro componente de software.

Avatar: Es una representación del usuario como un modelo en tres dimensiones dentro del mundo virtual.

LAN: Local Area Network. Es una red de computadoras que cubre un área física pequeña, como una casa, oficina, o un pequeño grupo de edificios. Comparadas con las WAN (Wide Area Network), las LAN generalmente tienen una tasa de transferencia de datos más alta y cubren un área geográfica menor.

SOAP: Es un protocolo (conjunto de reglas) que especifica la forma de intercambiar información estructurada en la implementación de un servicio web. Está basado en XML.

WSDL: Es un archivo en formato XML que se utiliza para describir un servicio web. La descripción incluye las operaciones disponibles y la forma en que deben de comunicarse los mensajes entre los dos puntos de comunicación.

XML: Es un conjunto de reglas definidas para codificar documentos y estructuras de datos de forma electrónica en formato de texto.

18. APÉNDICE 2: REPRESENTACIÓN DE LOS DATOS

18.1. Ítems

Figura 26: Diagrama de ER de ítems

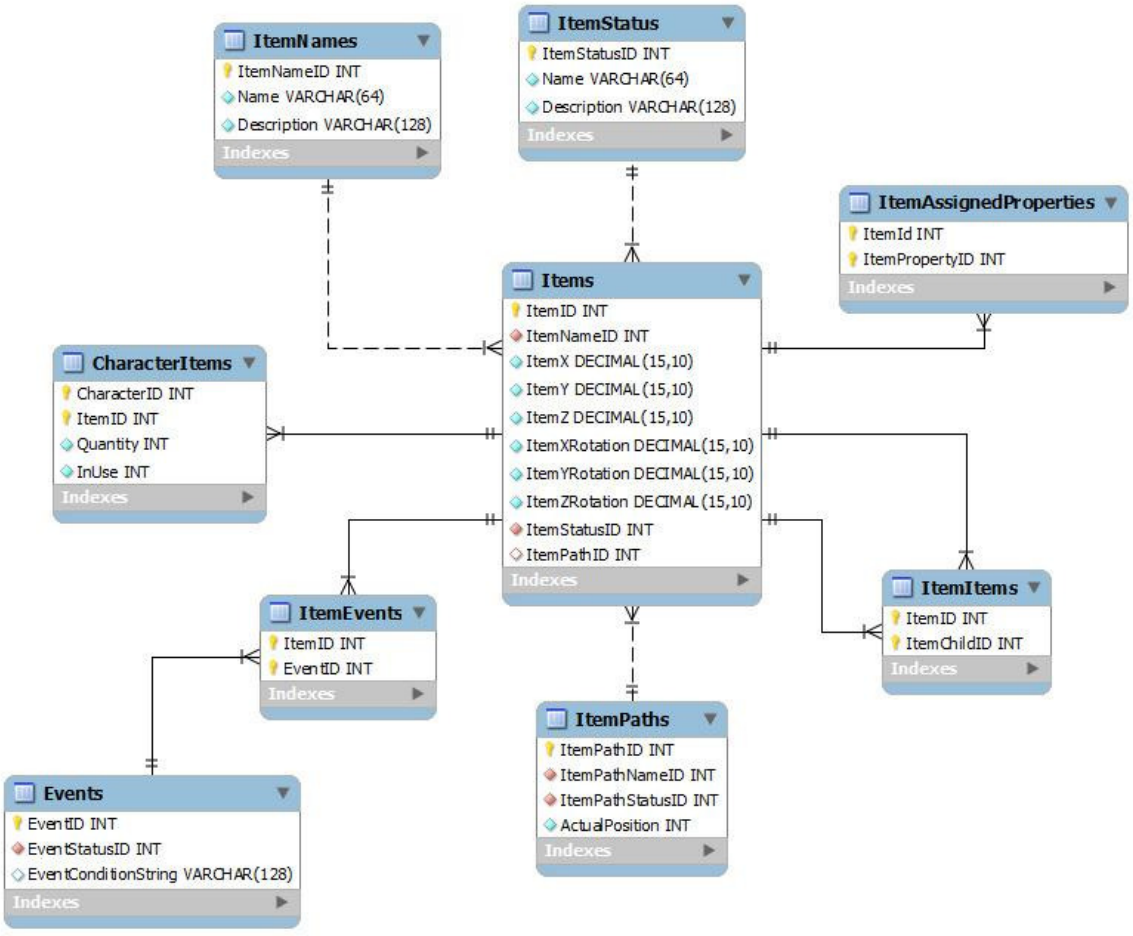


Figura 27: Diagrama de clases de ítems

Figura 28: Diagrama de ER de propiedades de ítems

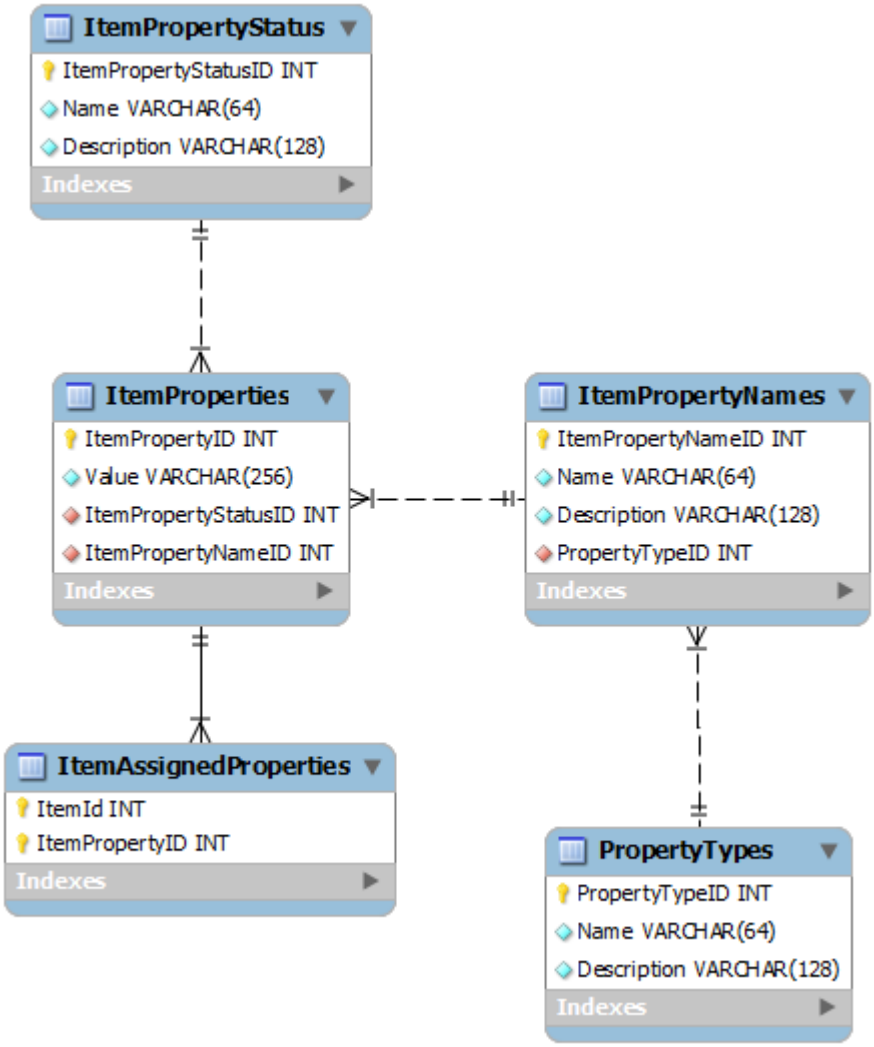


Figura 29: Diagrama de clases de propiedades de ítems

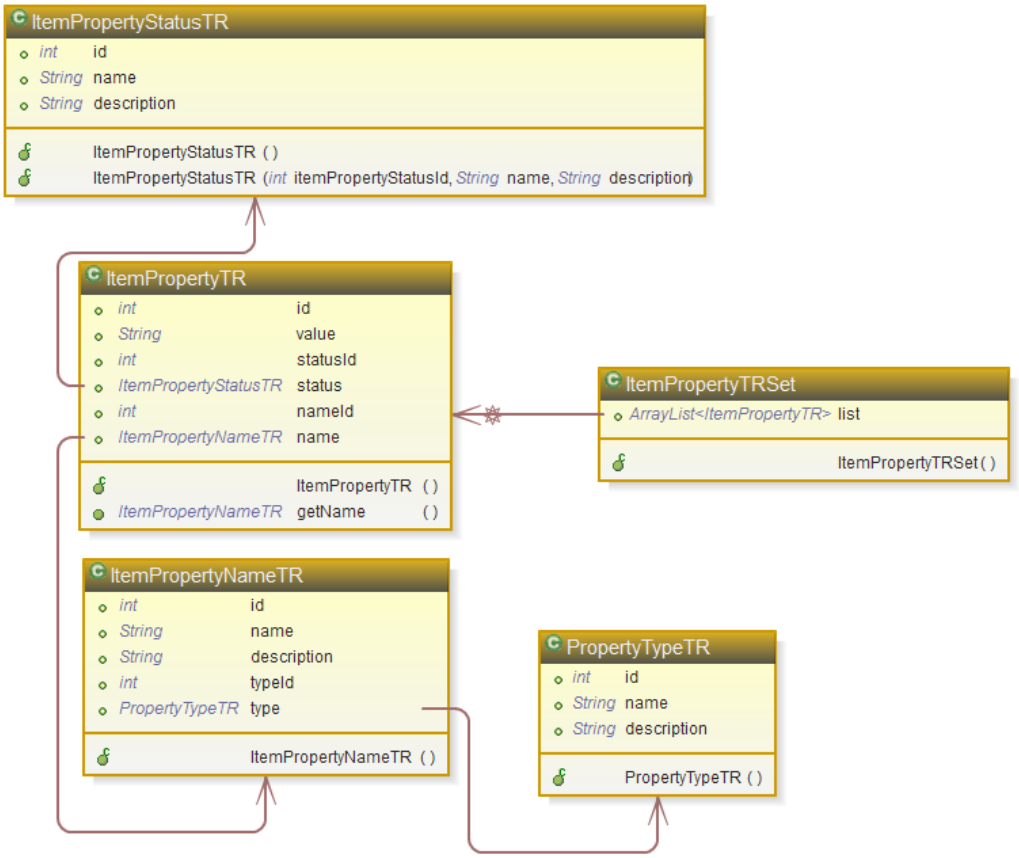


Figura 30: Diagrama de ER de trayectoria de ítems

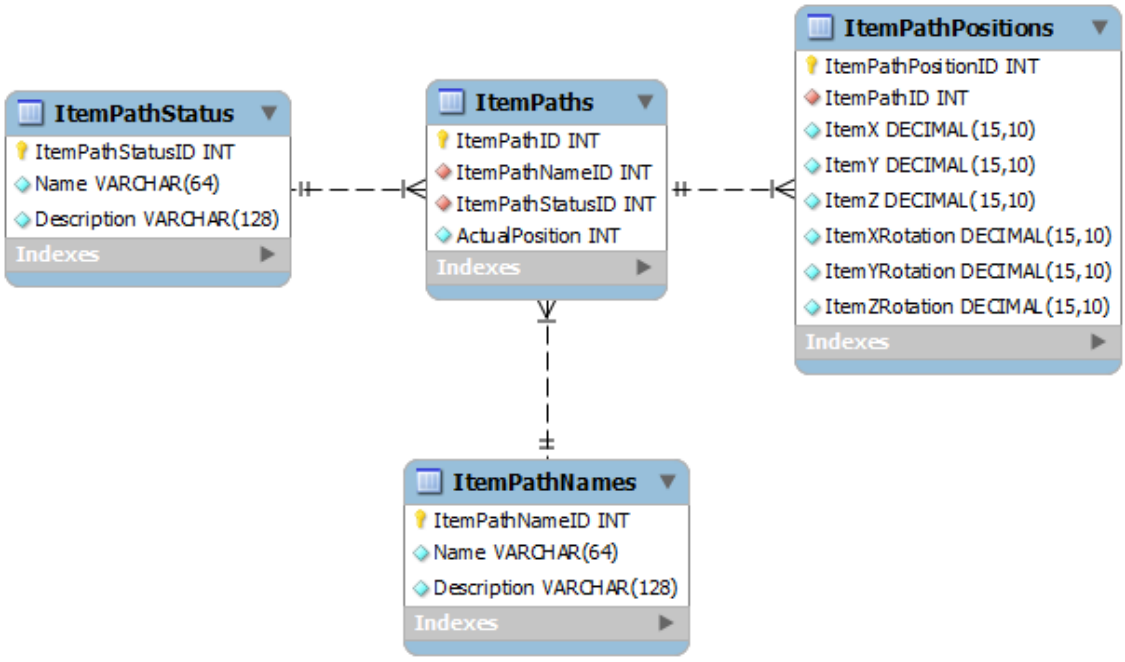
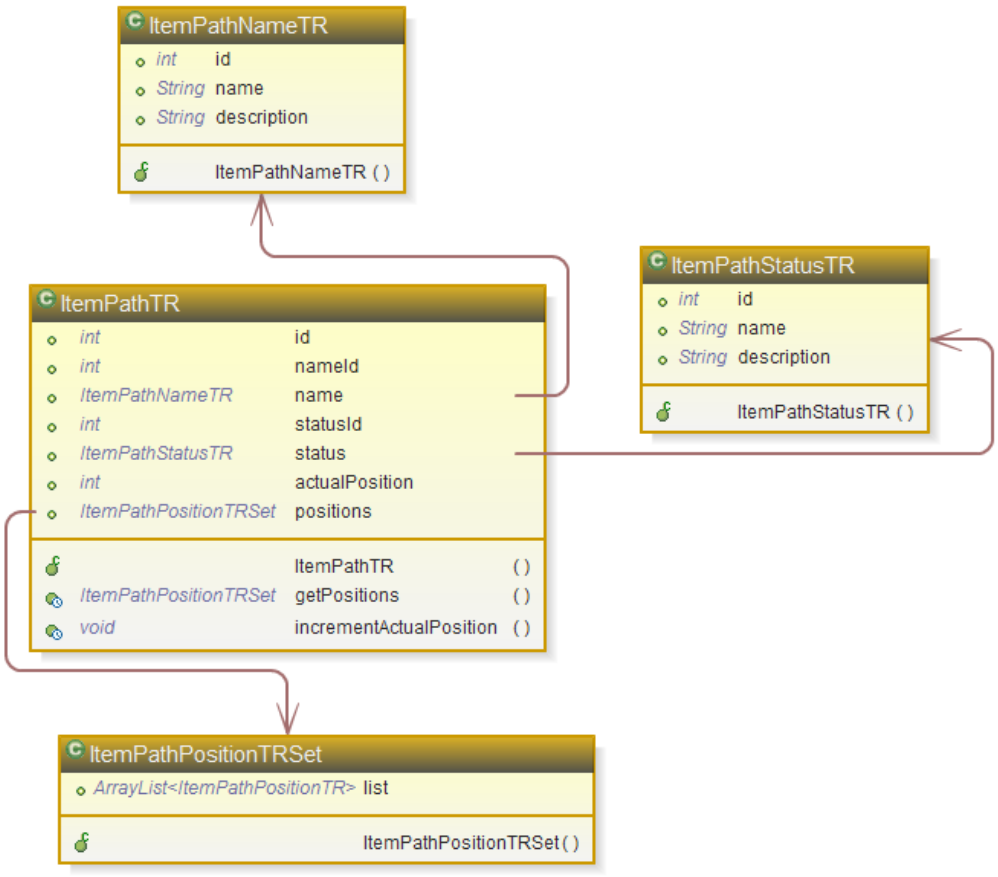


Figura 31: Diagrama de clases de trayectoria de ítems



18.2. Personajes

Figura 32: Diagrama de ER de personajes

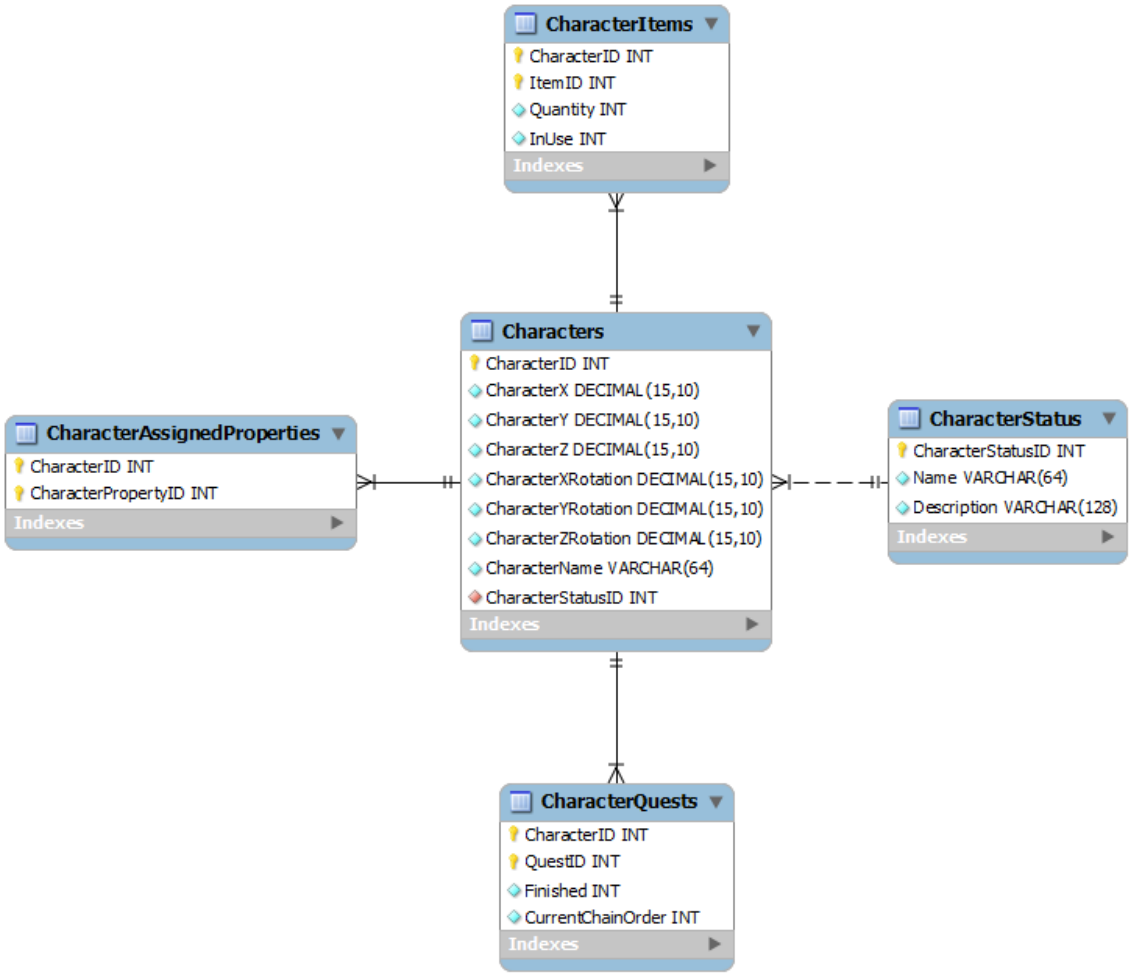


Figura 33: Diagrama de clases de personajes

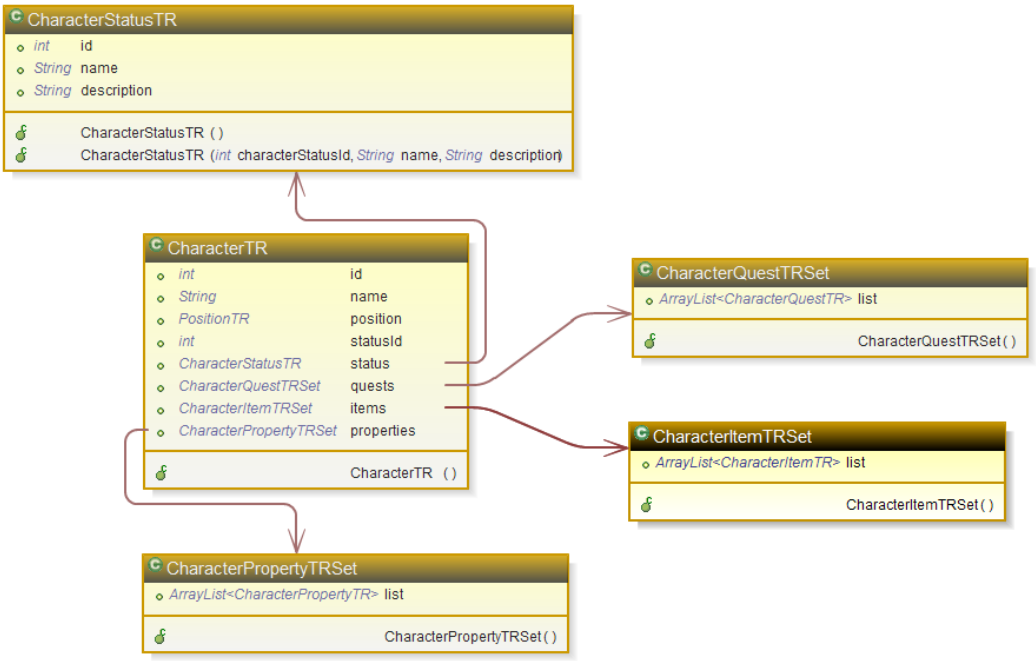


Figura 34: Diagrama de ER de propiedades de los personajes

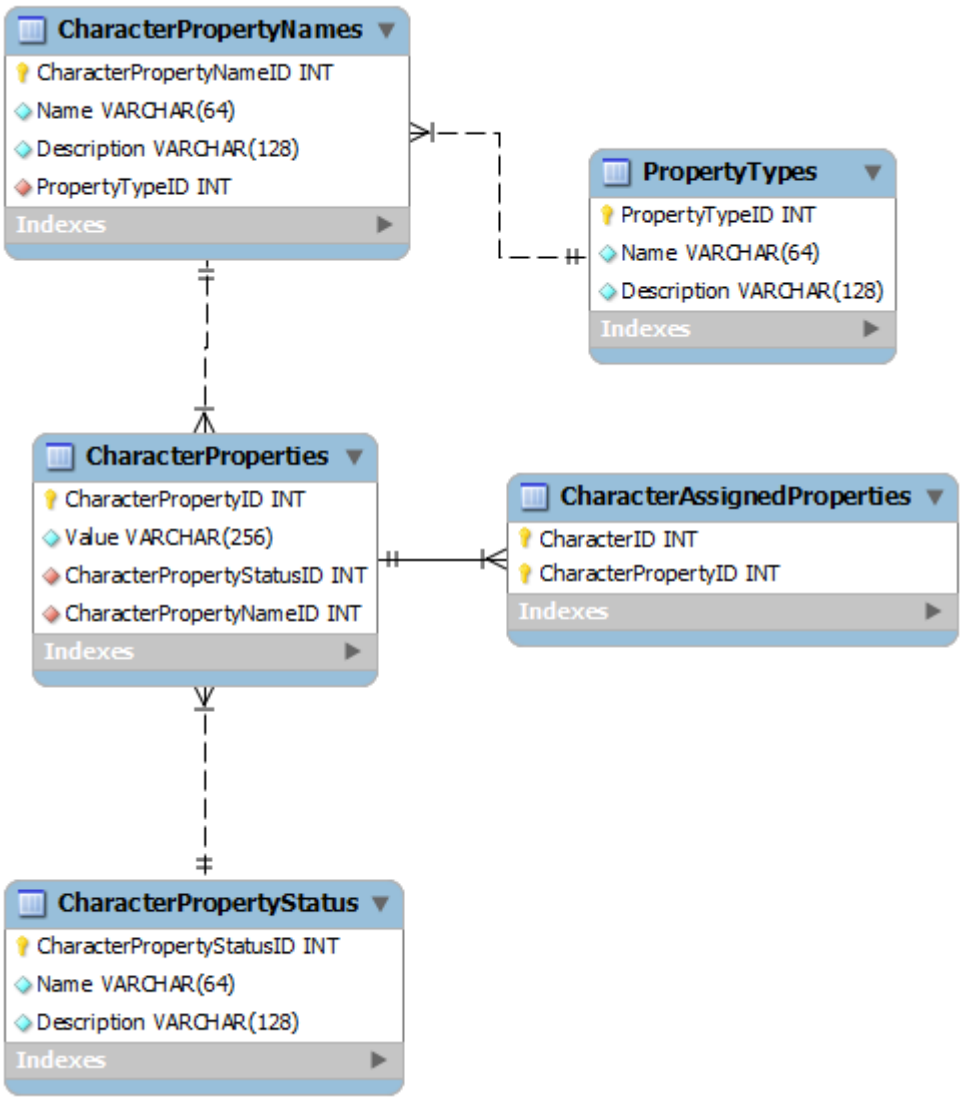


Figura 35: Diagrama de clases de personajes

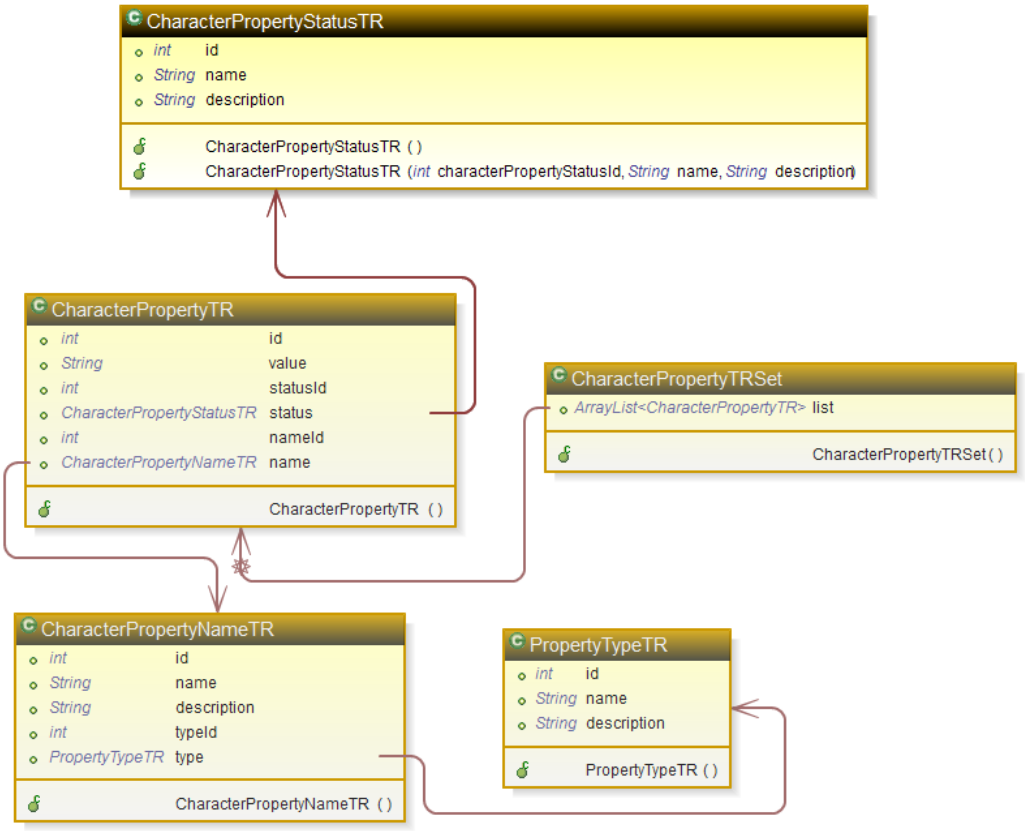


Figura 36: Diagrama de ER de misiones de los personajes

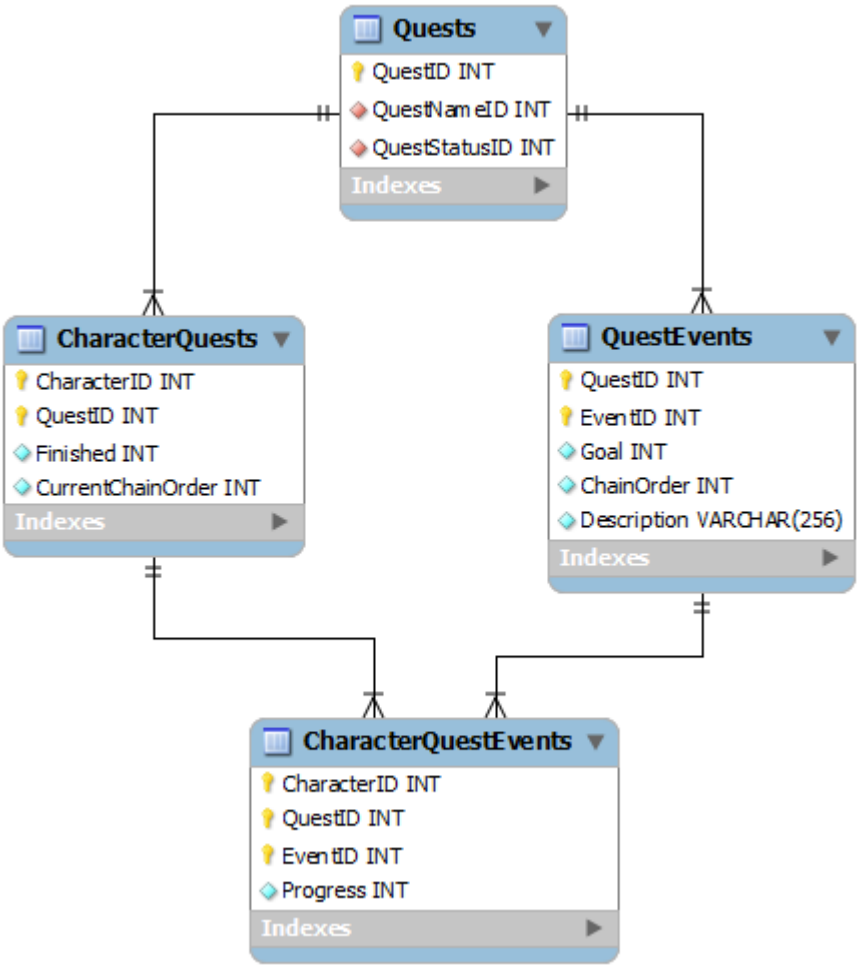
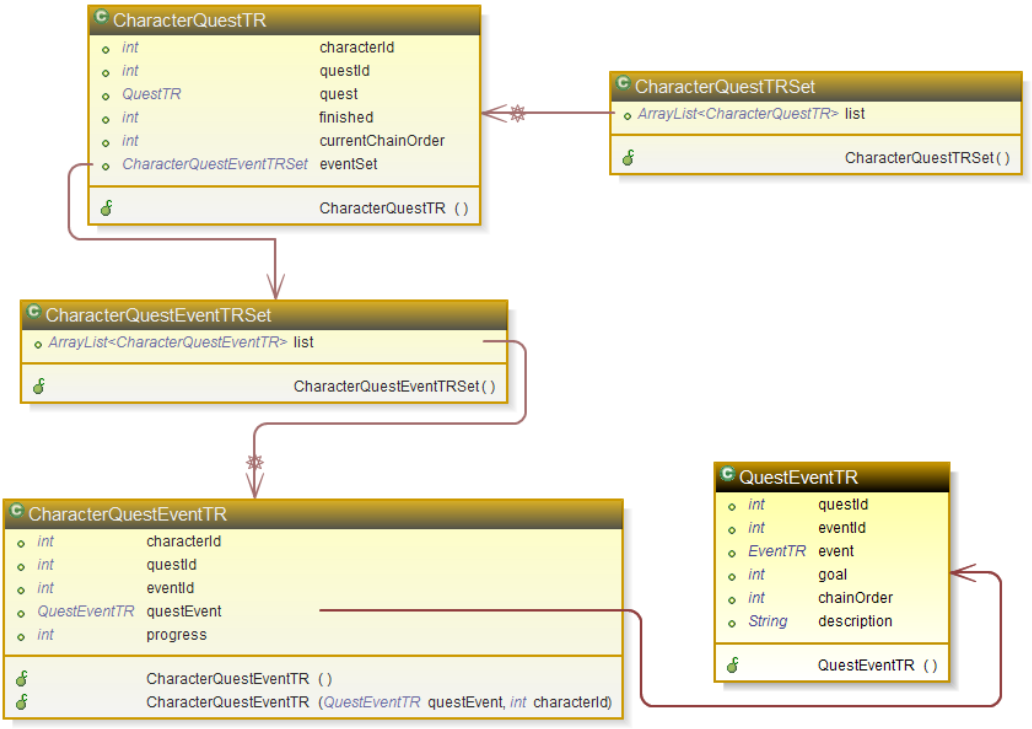


Figura 37: Diagrama de clases de misiones de los personajes



18.3. Misiones

Figura 38: Diagrama de ER de misiones

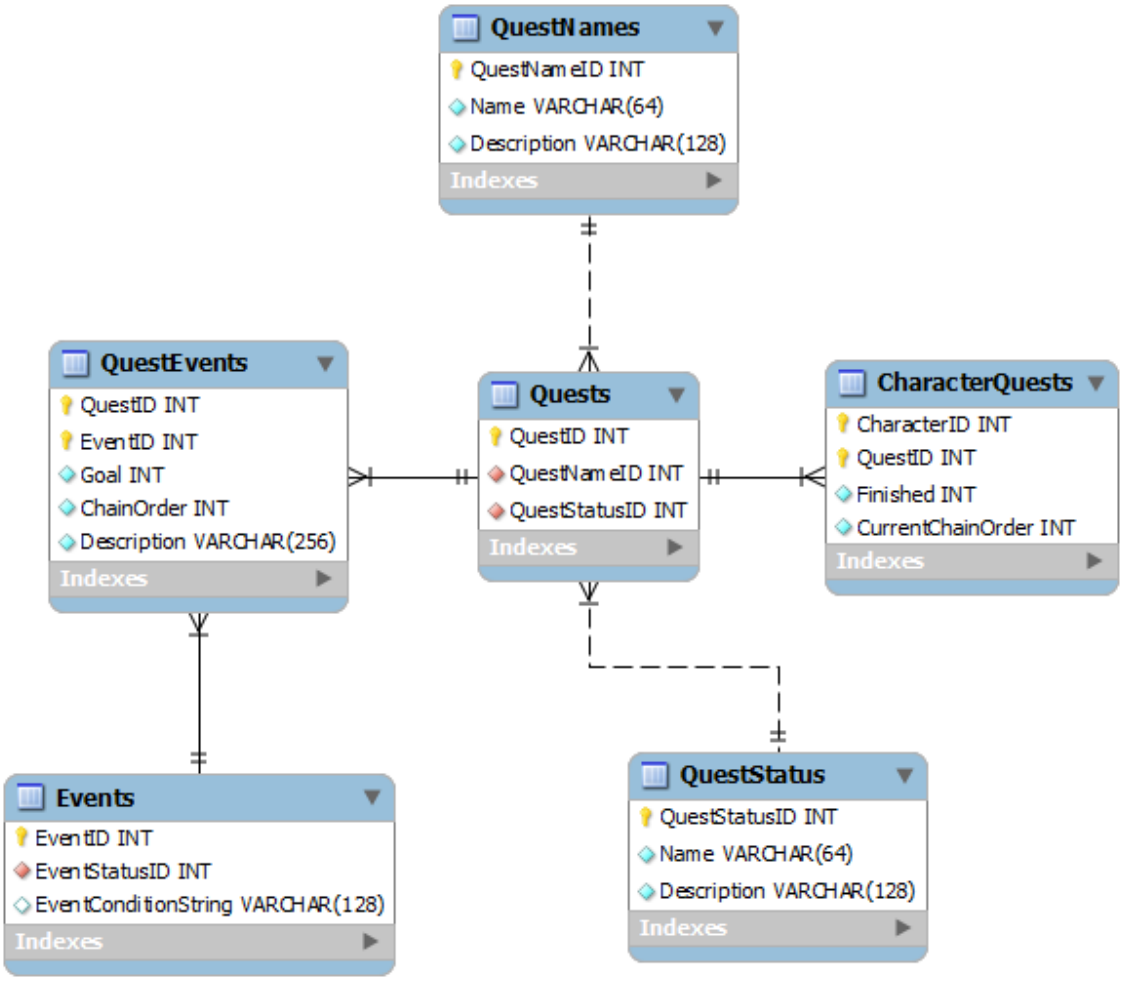
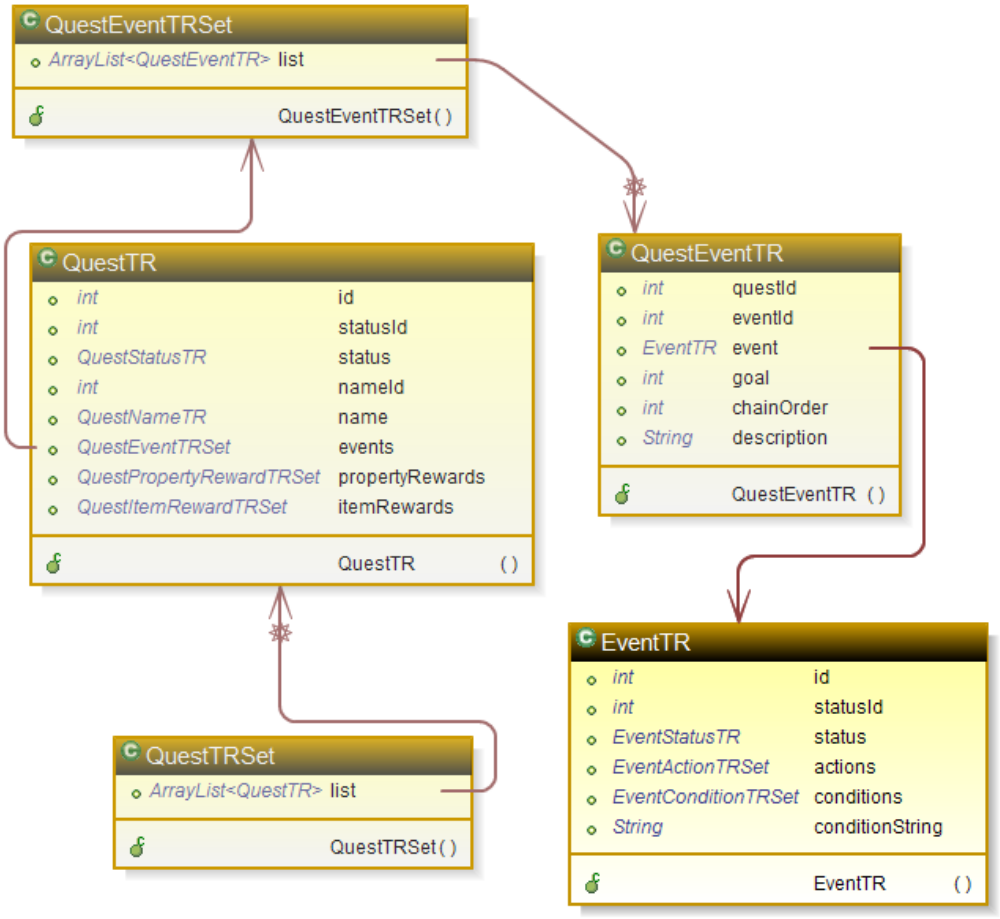


Figura 39: Diagrama de clases de misiones



18.4. Eventos

Figura 40: Diagrama de ER de eventos

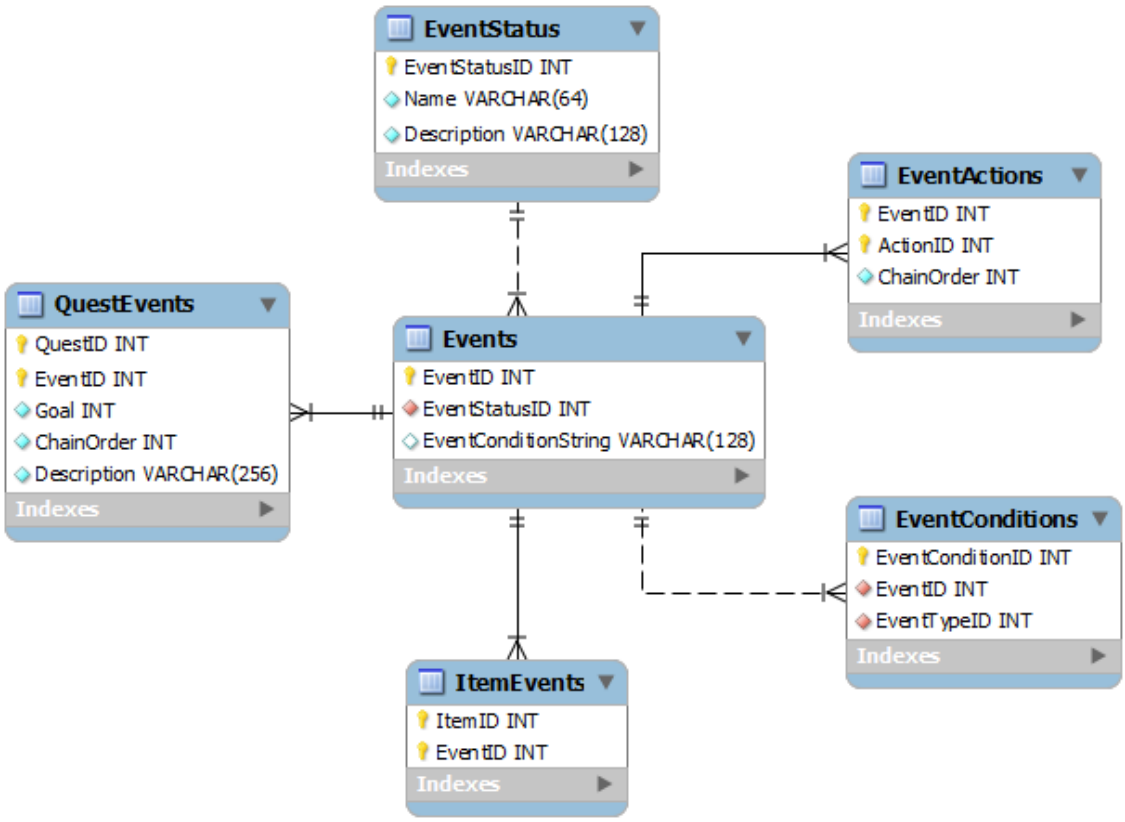


Figura 41: Diagrama de clases de eventos

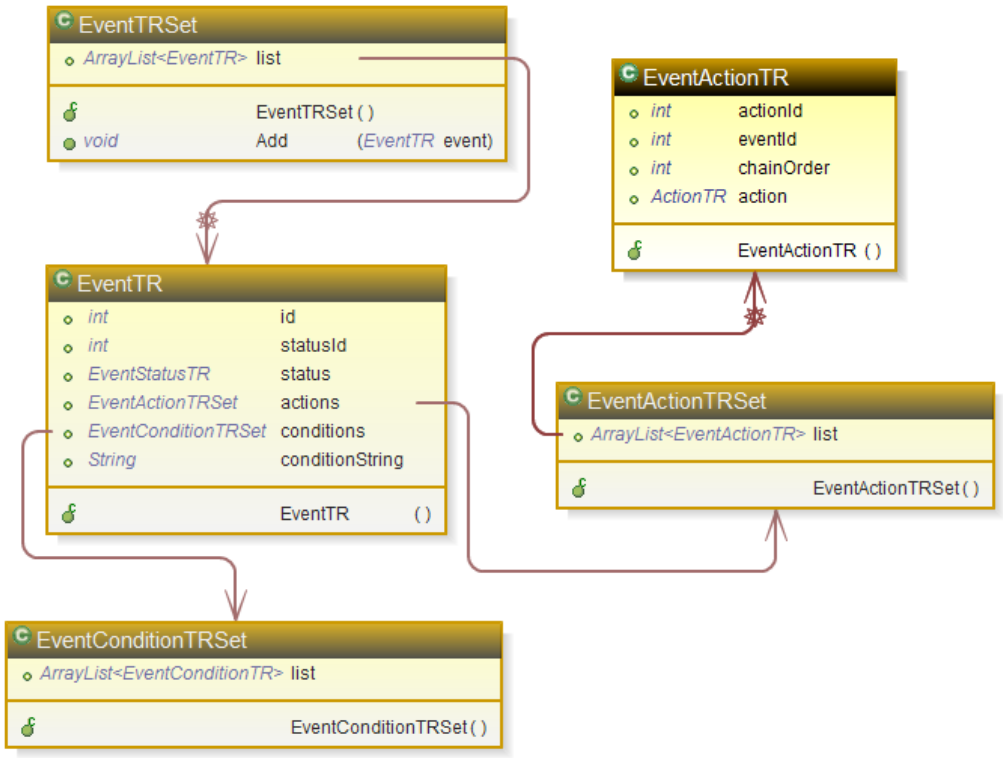


Figura 42: Diagrama de ER de condiciones de los eventos

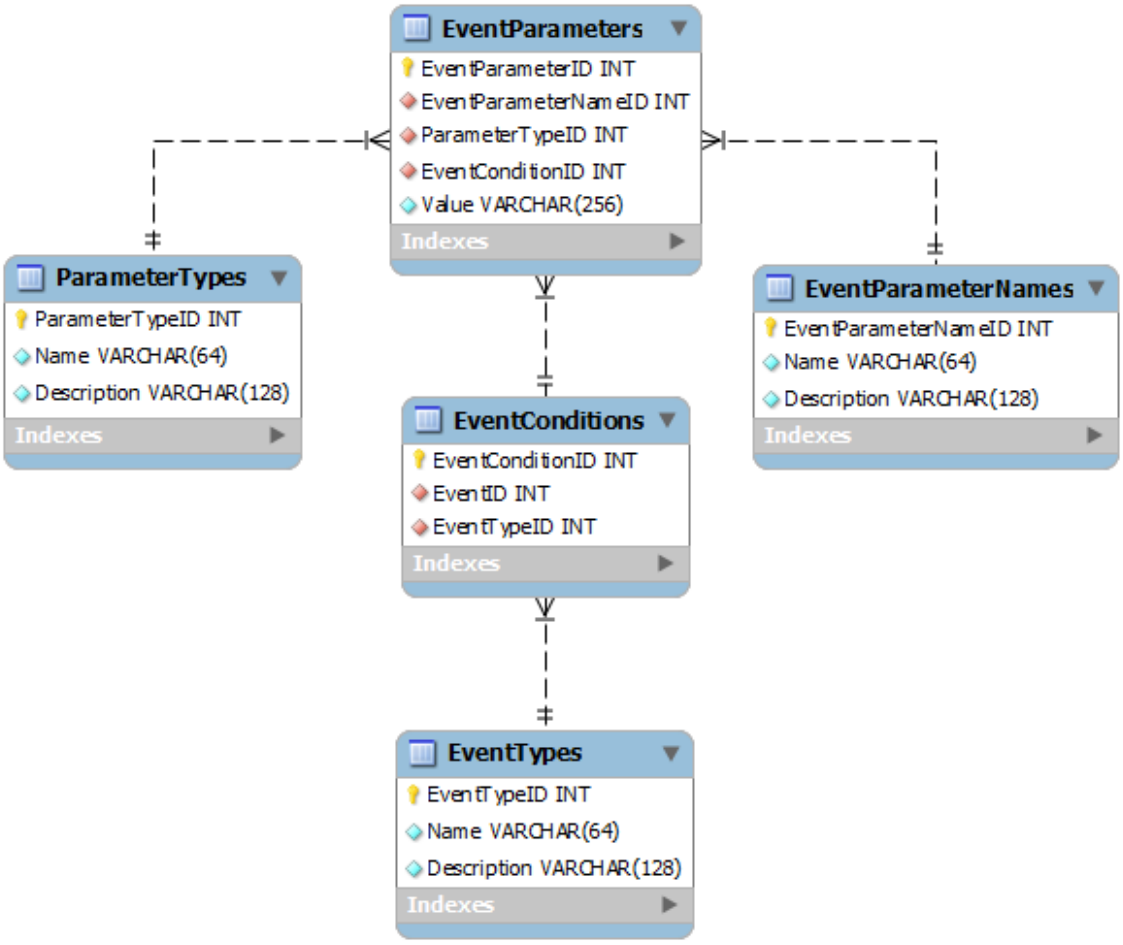
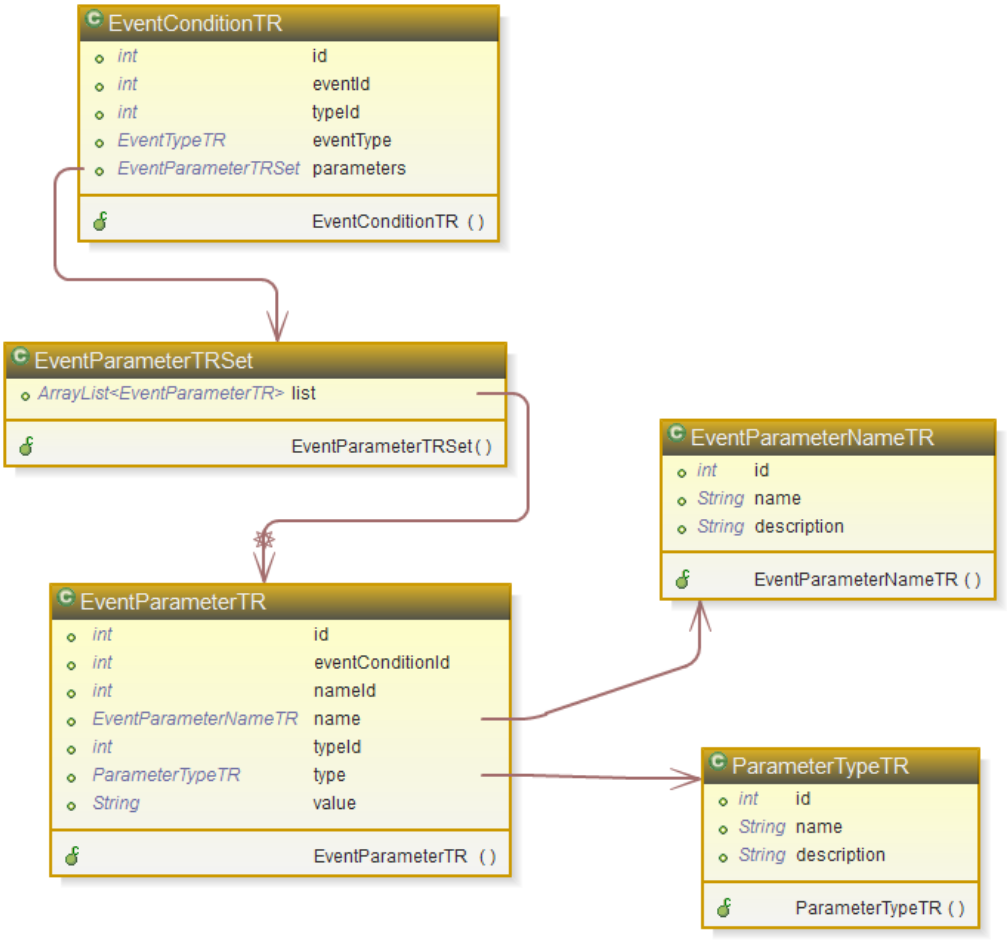


Figura 43: Diagrama de clases de condiciones de los eventos



18.5. Acciones

Figura 44: Diagrama de ER de acciones

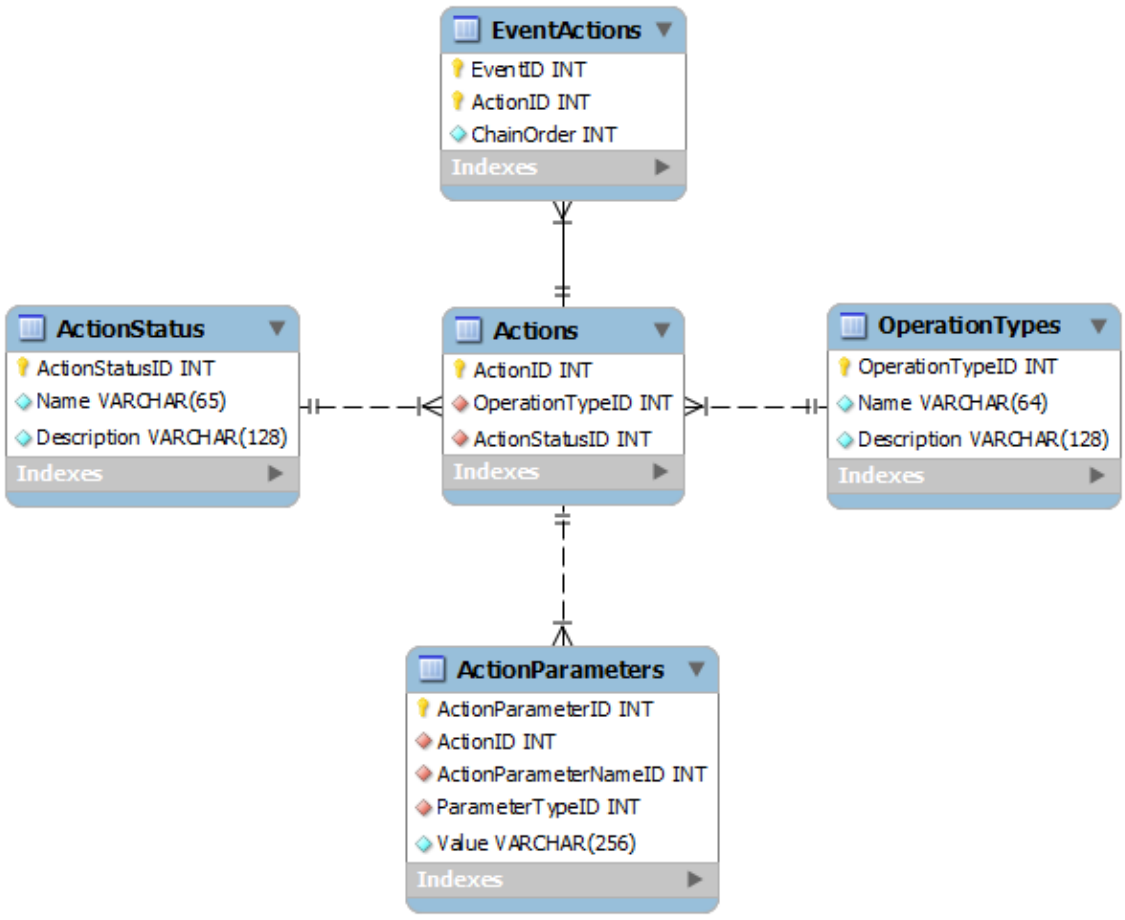


Figura 45: Diagrama de clases de acciones

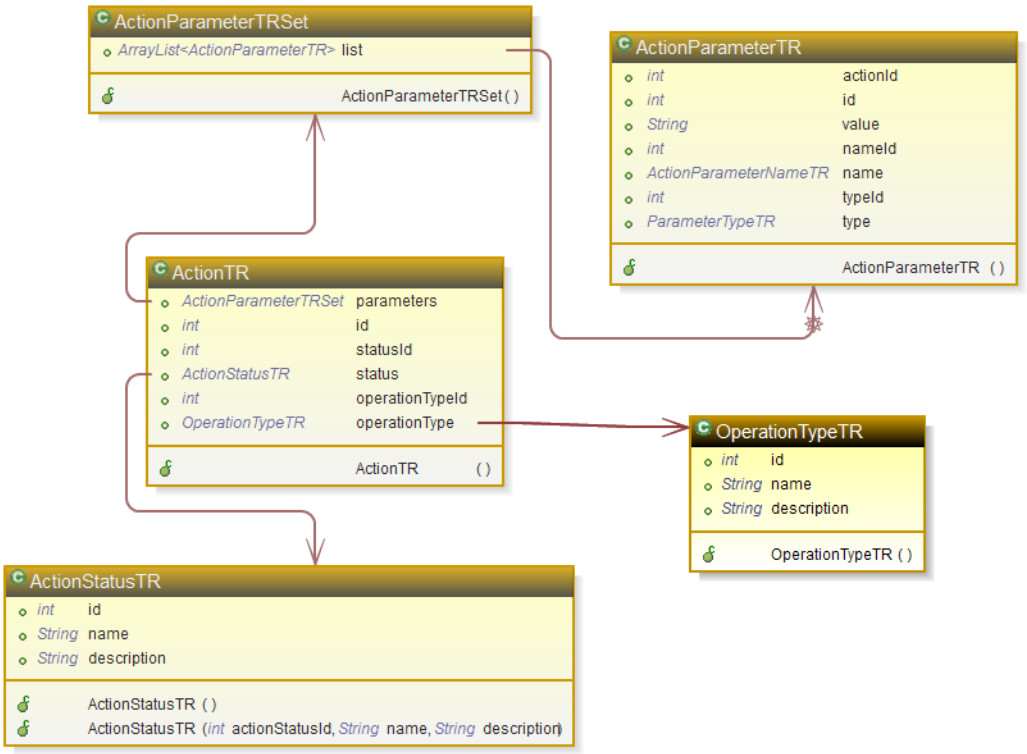


Figura 46: Diagrama de ER de parámetros de las acciones

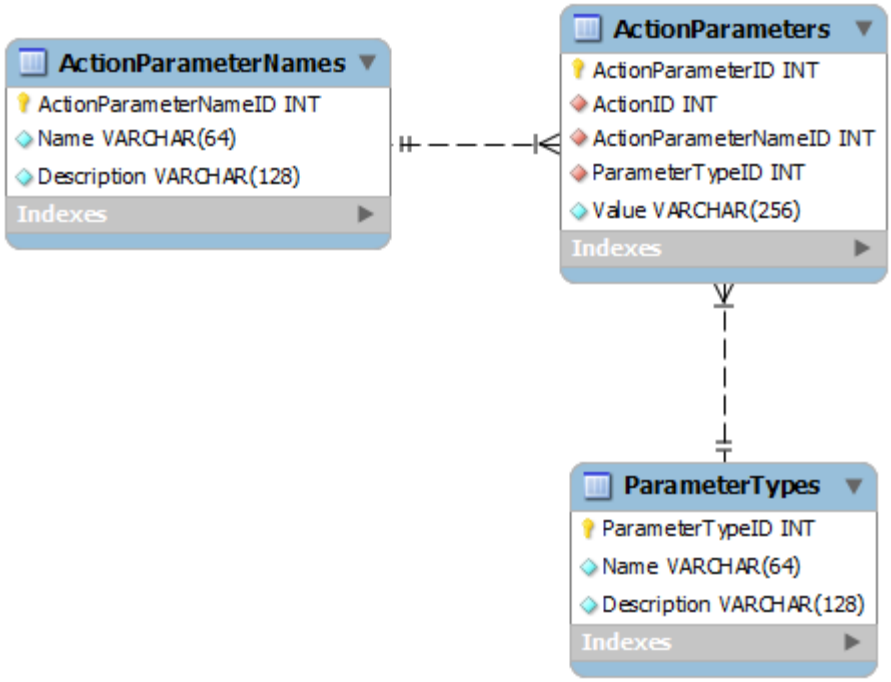
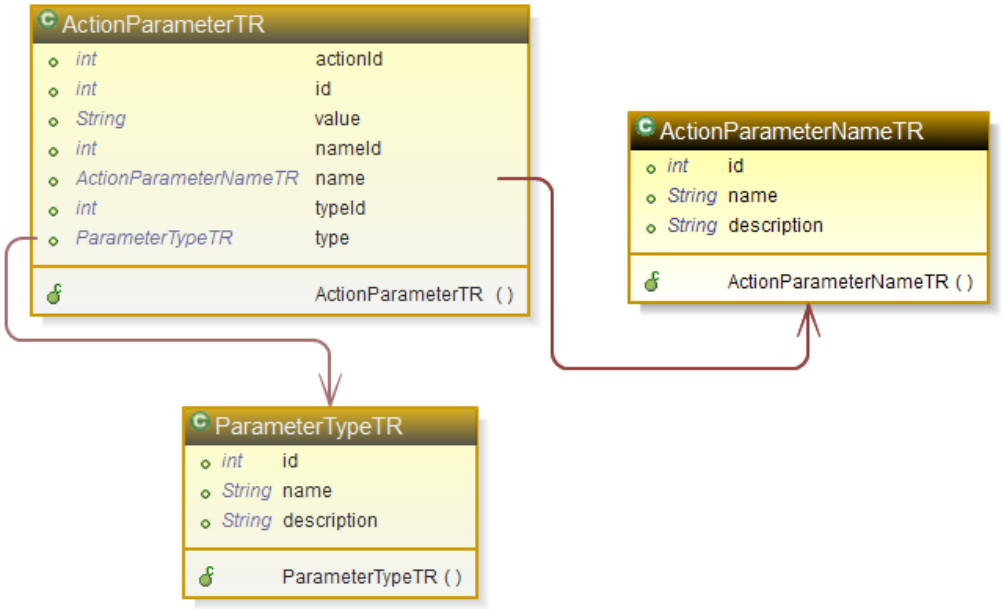


Figura 47: Diagrama de clases de parámetros de las acciones



19. APÉNDICE 3: CÓDIGO FUENTE