ESCUELA SUPERIOR DE INGENIERÍA

Modelo de accesibilidad para personas con discapacidad visual en juegos sociales multiterminal

Autor:
Javier Vidal Peña

Directores:
César Bernal Bravo
José Antonio Piedra Fernández

Fecha:
Julio de 2011

UNIVERSIDAD DE ALMERÍA

A mi familia, por su inmejorable apoyo durante estos últimos cinco años. Gracias.

ÍNDICE DE CONTENIDOS

1.1 Índice de ilustraciones

1.2 Índice de tablas

1.3 Índice de códigos

1.4 Índice de ejemplos de juego

2 Introducción

El propósito del proyecto es crear un juego social en red basado en texto que favorezca las relaciones sociales entre personas con discapacidad visual y el resto de la sociedad y que sea accesible por el mayor número posible de terminales.

El origen de este tipo de juegos se remonta a la era en la que las tarjetas gráficas de los ordenadores domésticos sólo eran capaces de visualizar texto y las “aplicaciones de consola” dominaban el entorno de la ofimática. En la actualidad este tipo de juegos sigue siendo una de las pocas alternativas que dispone un estudiante para desarrollar un juego completamente original sin la necesidad de grandes recursos.

Por otro lado cada día son más los dispositivos conectados a Internet. El juego, al estar basado en texto, favorece que sea accesible por cualquier terminal que disponga simplemente de una pantalla y un teclado. El objetivo es que cualquier persona con acceso a Internet disponga de un medio para poder jugar.

Finalmente, el colectivo de usuarios con discapacidad visual puede servirse de lectores de pantalla para beneficiarse de este tipo de juegos. Mientras que los videojuegos suelen ser poco accesibles o simplemente inaccesibles por su carácter visual, este tipo de juego presenta las características ideales para que una persona con problemas de visión pueda jugar sin grandes complicaciones. El usuario podrá utilizar lectores de pantalla para interpretar el texto que le envíe el juego y reproducirlo mediante sintetizadores de voz o una salida braille.

2.1 Finalidad

La meta del proyecto es ofrecer un juego gratuito en el que cada usuario juega a escribir un libro. La imagen que se desea ofrecer es la de un juego en el que el jugador narra su propia historia. Cada jugador tendrá un personaje y la libertad de interactuar con el resto de jugadores y el entorno mediante comandos. De esta forma el jugador irá describiendo una historia con sus acciones. Y al ser en red, todos los usuarios compartirán un mismo entorno y decidirán con sus acciones el porvenir del mundo.

Los usuarios deberán asociarse y cooperar para lograr objetivos que le reportarán fama y prestigio dentro de la comunidad. La historia continuará mientras existan jugadores, sin un final establecido. La finalidad de los jugadores será por tanto colaborar y competir para conseguir logros que los hagan destacar.

2.2 Objetivos

2.2.1 Objetivos generales:

1. Desarrollar un juego de rol multijugador en red que permita a los jugadores narrar la historia de sus personajes mediante sus acciones.
2. Utilizar las tecnologías de comunicación actuales para llevar el juego a la mayoría de dispositivos con conexión a Internet.
3. Adaptar el juego al colectivo de usuarios con discapacidad visual.

2.2.2 Objetivos específicos:

a) Crear una lógica de juego que ofrezca un elevado grado de libertad a los jugadores.
b) Ofrecer un mundo rico en contenidos.
c) Utilizar técnicas SIG para nutrir el entorno de información.
d) Conceder a los jugadores la posibilidad de construir el mundo donde se desarrolla el juego, como un sand-box.
e) Favorecer la creación de comunidades entre los usuarios del juego.
f) Establecer una curva de aprendizaje sencilla para los nuevos jugadores.
g) Crear un sistema de logros que atraiga a los jugadores y los incite a volver.
h) Ofrecer un medio de conexión al juego para cada uno de los terminales más usados con acceso a Internet.
i) Crear un lenguaje de alto nivel que permita la visualización del juego de forma similar independientemente del dispositivo.
j) Ofrecer una página web donde poder encontrar información en tiempo real sobre el desarrollo del juego.
k) Crear una aplicación que permita jugar a través del navegador web.
l) Cumplir los estándares web que certifican la accesibilidad frente a usuarios con problemas de visión.
m) Facilitar el uso de lectores de pantalla sobre la aplicación web desarrollada.
n) Adaptar la información ofrecida por el juego a los usuarios que utilicen lectores de pantalla para jugar.
o) Hacer accesible la tecnología web utilizada (AJAX, HTML5).

2.3 Metodología

El proceso de desarrollo del proyecto será dividido en tres etapas. Las etapas serán definidas por los principales objetivos del proyecto y tendrán completa independencia entre ellas. Cada etapa se subdividirá en tareas que corresponden con los objetivos más concretos que se deben cumplir para alcanzar los logros establecidos.

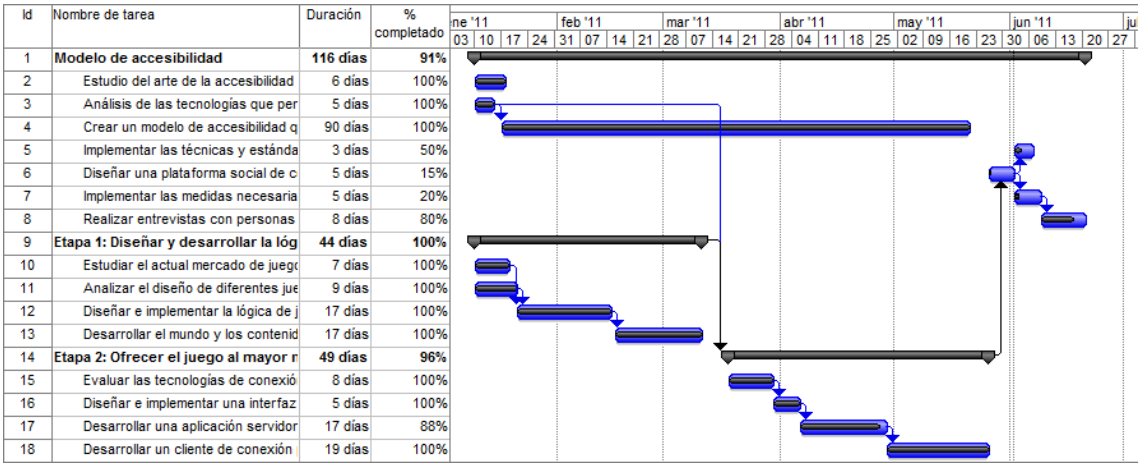
1) Etapa 1: Modelo de accesibilidad
   a) Estudio del arte de la accesibilidad en las nuevas tecnología [5].
   b) Análisis de las tecnologías que permiten realizar conexiones full-duplex sobre HTTP (AJAX,HTML5).
   c) Crear un modelo de accesibilidad que comprenda los diferentes tipos de discapacidad visual (daltonismo, resto de visión, ceguera total, …)
   d) Implementar las técnicas y estándares establecidos para mejorar la adaptación de lectores de pantalla.
   e) Diseñar una plataforma social de compartición de experiencias mediante RSS.
   f) Implementar las medidas necesarias para que la web del proyecto cumpla con los estándares de accesibilidad.
   g) Realizar entrevistas con personas con diferentes discapacidades visuales para evaluar el modelo de accesibilidad.
2) Etapa 2: Diseñar y desarrollar la lógica del juego que permita al jugador tener un elevado nivel de libertad.
   a) Estudiar el actual mercado de juegos de rol en línea y evaluar la situación de los juegos multijugador basados en texto.
   b) Analizar el diseño de diferentes juegos de rol de gran popularidad.
   c) Diseñar e implementar la lógica de juego.
   d) Desarrollar el mundo y los contenidos sobre los que prosperará el juego.
3) Etapa 3: Ofrecer el juego al mayor número posible de terminales.
   a) Evaluar las tecnologías de conexión que posibiliten el acceso al juego.
   b) Diseñar e implementar una interfaz que sirva como base a los diferentes métodos de conexión al juego.
   c) Desarrollar una aplicación servidor por cada protocolo de conexión admitido.
   d) Desarrollar un cliente de conexión para cada protocolo de conexión admitido.

2.4 Planificación

El proceso de la actividad durará seis meses y será dividido en dos etapas principales. La primera etapa corresponderá a la construcción del juego, mientras que la segunda etapa será destinada a ofrecer el juego al mayor número posible de terminales. Durante todo el proceso el desarrollo del modelo accesible será una actividad paralela pues dependerá de la evolución de las dos etapas. En líneas generales la planificación del proyecto se ha definido de la siguiente forma:

[1] Primera etapa: Enero – Marzo.
[2] Segunda etapa: Abril- Junio

El siguiente diagrama de Gantt muestra con detalle la planificación realizada y el estado actual de las tareas.



Id	Nombre de tarea	Duración	% completado
1	**Modelo de accesibilidad**	**116 días**	**91%**
2	Estudio del arte de la accesibilidad	6 días	100%
3	Análisis de las tecnologías que per	5 días	100%
4	Crear un modelo de accesibilidad q	90 días	100%
5	Implementar las técnicas y estánda	3 días	50%
6	Diseñar una plataforma social de c	5 días	15%
7	Implementar las medidas necesaria	5 días	20%
8	Realizar entrevistas con personas	8 días	80%
9	**Etapa 1: Diseñar y desarrollar la lóg**	**44 días**	**100%**
10	Estudiar el actual mercado de juego	7 días	100%
11	Analizar el diseño de diferentes jue	9 días	100%
12	Diseñar e implementar la lógica de j	17 días	100%
13	Desarrollar el mundo y los contenid	17 días	100%
14	**Etapa 2: Ofrecer el juego al mayor r**	**49 días**	**96%**
15	Evaluar las tecnologías de conexió	8 días	100%
16	Diseñar e implementar una interfaz	5 días	100%
17	Desarrollar una aplicación servidor	17 días	88%
18	Desarrollar un cliente de conexión	19 días	100%

Ilustración 1: Planificación del proyecto

2.5 Recursos y materiales

Uno de los puntos fuertes del proyecto consiste en la escasa cantidad de recursos necesarios para su realización. De cara al desarrollo se utilizará el entorno de desarrollo *Visual Studio 2010 Professional*, software que puede ser descargado de forma gratuita desde la web de la Universidad de Almería (http://msdn.ual.es/). La aplicación al completo será desarrollada como un *Sitio web ASP.NET* en sobre la plataforma *.NET Framework 4*. Para el desarrollo de las capas del SIG será utilizado cualquier editor de gráficos 2D gratuito.

En lo referente a los recursos hardware, en ninguna de las etapas del desarrollo del proyecto será necesario utilizar computadores de altas prestaciones. Cualquier ordenador personal actual posee capacidad suficiente para ser utilizado en el proyecto.

En cuanto a la implementación y puesta en marcha del juego será necesario utilizar un ordenador con conexión a Internet, con sistema operativo *Windows*, con la característica *Servicios World Wide Web* habilitada y soportar la versión 4 del entorno de trabajo *.NET Framework*. Los requisitos hardware del servidor no son elevados pues se trata de una aplicación que procesará pocas cantidades de datos y sobre los que se realizarán operaciones muy elementales.

Por último, para evaluar la experiencia de usuario del colectivo con discapacidad visual será necesario utilizar un lector de pantalla. En la actualidad la gran mayoría de internautas de este colectivo utilizan el software *JAWS*. El coste de su licencia de utilización es muy elevado, pero la compañía ofrece periodos de prueba de varios meses de duración; tiempo estimado suficiente para la realización de las pruebas. Software Aumento Pantalla.

2.6 Contenido de la memoria

Atendiendo a los tres pilares fundamentales del proyecto, este documento se estructura en tres capítulos fundamentales. El primer capítulo versa sobre el modelo de accesibilidad desarrollado, donde se muestran las pautas para desarrollar un juego de texto en red siguiendo la metodología *diseño para todos*. En el segundo capítulo se describe el desarrollo de juego y las principales diferencias entre la metodología de desarrollo de los contenidos utilizada y el resto de metodologías clásicas. En el tercer capítulo se realiza un análisis del carácter multiterminal del sistema, mostrando las tecnologías de conexión implementadas para conecta al juego. Finalmente se exponen las conclusiones al trabajo realizado y las propuestas para una futurible extensión del mismo. En los anexos del documento se incorpora, además, los diagramas creados durante la fase de diseño y un manual de usuario final de la aplicación.

3 Modelo de accesibilidad

Uno de los propósitos principales del proyecto es crear un patrón o molde de accesibilidad que pueda ser utilizado en cualquier juego en red basado en texto.

La naturaleza de estos juegos permite que su formato sea accesible por la gran mayoría de internautas, independientemente de su nivel o tipo de discapacidad. Mientras que los usuarios sin ningún tipo de discapacidad o con discapacidad auditiva pueden leer la pantalla, las personas con discapacidad visual pueden servirse de lectores de pantalla para acceder al juego e interactuar con él. Este formato de juego (MUD, Multi User Dungeon) presenta por tanto el mayor índice de accesibilidad para el colectivo de personas con discapacidad visual. Sin embargo, en los últimos años los juegos de texto han quedado relevados a un segundo plano. La industria sólo opta por crear videojuegos pues ofrecen un atractivo muy eficaz para la gran mayoría de jugadores, los gráficos. Actualmente, con el auge de la tecnología 3D y el acceso a sistemas gráficos de mayor potencia a nivel domésticos, el futuro de los juegos basados en texto es poco esperanzador. Pese a ello, el proyecto se embarca con el objetivo de ofrecer *un juego para todos*. Para ello ha sido necesario establecer dos enfoques: a) crear un juego único e igualmente atractivo que un videojuego y b) ofrecer un juego accesible para cualquier internauta y no centrado en personas con discapacidad visual. El primer enfoque será analizado en el tercer capítulo del documento, mientras que en este apartado se describe el modelo de accesibilidad que se ha implementado para lograrlo.

El modelo ha sido basado en la filosofía de *Diseño para Todos*. El concepto de Diseño para Todos (Design for All) persigue establecer soluciones de diseño para que todas las personas, con independencia de sus condiciones de edad, género o capacidades, puedan utilizar los espacios, productos y servicios de su entorno, tomando parte activa, al mismo tiempo, en la construcción de la sociedad (García de Sola, 2006). También es interesante la definición que se ofrece en (Technosite, 2006): "*El diseño para todos es el diseño de productos y entornos de fácil uso para el mayor número de personas posible, sin la necesidad de adaptarlos o rediseñarlos de una forma especial"*. Esta definición realza la idea de crear un único producto mediante una visión holística, sin la necesidad de crear productos alternativos para cada necesidad. Por otro lado, en lo referente al diseño para todos en las tecnologías de la información (Stephanidis, 2001) lo describe como: *"Diseño para Todos en la Sociedad de la Información es el esfuerzo consciente y sistemático para aplicar proactivamente métodos y herramientas, con el fin de desarrollar productos y servicios de la Tecnología de la Información que sean accesibles y utilizables por todos los ciudadanos, evitando así la necesidad de adaptaciones posteriores o un diseño especializado"*. Finalmente, a nivel nacional, la Ley 51/2003 de Igualdad de Oportunidades, no Discriminación y Accesibilidad Universal, define el término Diseño para Todos en su Artículo 2: Principios. Diseño para Todos: *"la actividad por la que se concibe o proyecta desde el origen, y siempre que ello sea posible, entornos, procesos, bienes, productos, servicios, objetos, instrumentos, dispositivos o herramientas, de tal forma que puedan ser utilizados por todas las personas, en la mayor extensión posible"*.

Este concepto se considera más una filosofía de trabajo que un objetivo tangible, pues no existen reglas específicas que determinen la consecución de logros. Sin embargo, el llamado *Diseño Universal*, concepto estrechamente relacionado, establece en su versión 2.0 (1 de abril de 1997) una serie de principios que han servido como estrategia de actuación para afrontar el modelo de accesibilidad. El primer apartado de este capítulo tratará de aclarar los siete

principios y su implicación con el modelo de forma genérica. Los siguientes apartados tratan de explicar con mayor detalle el modelo de accesibilidad referido a problemas y situaciones más específicas del proyecto.

Para realizar este modelo, además, se decidió colaborar con la Organización Nacional de Ciegos Españoles (ONCE). En un principio el objetivo fue acercar el proyecto a centros escolares o grupos de personas asociadas a la organización. Sin embargo esto no fue posible y finalmente se optó por la realización de entrevistas a personal del centro de dirección de la ONCE en Almería. Por motivos ajenos al proyecto, solo se pudo realizar una única entrevista a una persona de ese centro.

La persona entrevistada (se ha preferido mantener su anonimato) se encuentra trabajando en estos momentos en la sede de la ONCE en Almería. Además, se encuentra realizando su tesis doctoral. Esta persona sufre *hipovisión*, una privación parcial de la vista que no puede ser corregida adecuadamente con gafas convencionales. Durante la entrevista fue imposible ejecutar una prueba de la aplicación, pues los ordenadores que allí utilizan no disponen de la tecnología necesaria para realizar la conexión con el sistema. En general, esta persona nos informó de la problemática de que la tecnología que ellos utilizan suele estar desactualizada, tanto por la falta de medios que les permitan acceder a tecnología más cara y actual, tanto por la falta de compromiso por parte de los desarrolladores de tecnología de hacer accesible sus nuevos productos.

La hipovisión es una de las discapacidades visuales más comunes. En este proyecto se ha querido realizar un estudio de otras discapacidades y sus principales características, a la hora de establecer un diseño adecuado que abarque a esta clasificación. A continuación se incluye una tabla en la que se muestran las principales discapacidades visuales, su descripción y su posible efecto sobre el juego.

Nombre	Descripción	Tipos	Como afecta al diseño
Ceguera total	Pérdida total del sentido de la vista		El usuario no puede leer.
Hipovisión	Privación parcial de la vista	• Agudeza • Campos	El usuario debe utilizar magnificadores de pantalla.
Daltonismo	Imposibilidad de distinguir algunos colores.	• Acromático • Monocromático • Dicromático • Tricromático anómalo	Dificultad a la hora de percibir el color de la fuente.

3.1 Principios del Diseño Universal

En este apartado se describe cómo se han aplicado los siete principios del Diseño Universal al modelo de accesibilidad de una forma genérica, sin atender a la tecnología utilizada.

3.1.1 Uso equiparable

El objetivo de este apartado es ofrecer las mismas formas de uso, evitando segregar o estigmatizar a cualquier usuario y ofrecer un diseño atractivo para todos los usuarios.

En lo referente al uso y a la no segregación de la comunidad, el diseño del modelo se ha planteado para que todos los jugadores representen una misma entidad dentro del juego con las mismas oportunidades. Independientemente de la discapacidad del usuario, los jugadores pueden realizar las mismas acciones e interactuar entre ellos. De esta forma se consigue una única comunidad de jugadores, en la que todos pueden colaborar y formar grupos de usuarios con las mismas posibilidades de cumplimentar logros y conseguir renombre.

En cuanto al diseño, ha sido necesario crear reglas de visualización que ofrezcan un diseño accesible y atractivo para todos los usuarios. El objetivo no es solo ofrecer una visualización específica para cada terminal, sino adaptar la visualización al tipo de usuario que conecta al juego. El conjunto de reglas y estrategias se analiza con mayor detalle en el capítulo Entorno de juego.

Para ello se ha diseñado una lógica de juego independiente del tipo de usuario, pero también independiente del tipo de visualización. Un modelo organizado en capas ha sido la opción elegida. El diseño del modelo ha sido dividido en tres capas fundamentales: la lógica de juego, la capa accesibilidad y la capa de presentación.

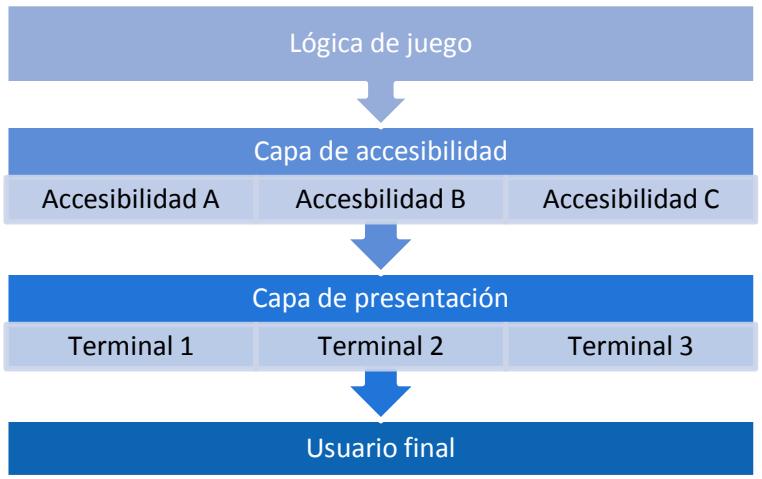


Ilustración 2: Modelo basado en capas

Las tres capas desempeñan su función de forma independiente, y así el usuario final recibe la información de juego completamente adaptada a sus necesidades. Las capas implementadas han sido:

- Lógica de juego: esta capa corresponde al funcionamiento interno del juego. En esta capa se encuentran las reglas de juego, así como los eventos y desencadenantes que rigen el funcionamiento del juego. Pese a que no ofrece ningún nivel de accesibilidad referido a discapacitados visuales, un diseño independiente del usuario final es fundamental para construir un modelo accesible por todos. Las reglas solo comprenden entidades genéricas de tipo *usuario*, y por tanto es imposible que el sistema favorezca o discrimine al jugador por su discapacidad. El funcionamiento concreto de esta capa y sus reglas será descrito más adelante en el capítulo de Desarrollo del juego social.
- Capa de accesibilidad: pese a que la naturaleza del juego lo convierte en un modelo accesible, es necesario tener en cuenta una serie de consideraciones adicionales para implementar un modelo diseñado para todos. Los problemas o discapacidades de los usuarios son muy dispares, y estos han de ser solventados mediante reglas de accesibilidad. Estas reglas se analizan en el apartado Entorno de juego.
- Capa de presentación: en un diseño para todos el concepto de sistema multiplataforma o multiterminal cobra especial relevancia. En un marco donde se pretende que el producto sea accesible para cualquier persona, es necesario ofrecer el juego en el mayor número posible de plataformas. El objetivo de esta capa es presentar la información de forma adecuada en cada terminal, cumpliendo los estándares de visualización y accesibilidad adecuados. Esta capa será analizada con mayor detenimiento en el capítulo Juego multiterminal.

El resultado de este modelo en capas puede visualizarse en el siguiente ejemplo. En él, dos jugadores introducen el mismo comando; sin embargo, uno de ellos tiene una discapacidad visual y por tanto la información que se muestra se adapta a sus características. En este caso, el juego elimina los colores y cualquier decoración textual:

```
>ficha
Datos generales
================================================================
Nombre:         Snaider
Entorno:        Estepa 209,193
Puntos
================================================================================
Vida:                   (200/200)     Energía:                 (20/20)
```

Ejemplo de juego 1: Mensaje de juego sin adaptación

```
>ficha
Nombre:         Jhanda
Entorno:        Pradera 219,196
Vida:                   (200/200)     Energía:                 (20/20)
```

Ejemplo de juego 2: Mensaje de juego adaptado

Por último indicar que este modelo basado en capas solo es aplicable al flujo de salida del juego, a los mensajes que el sistema emite a cada jugador. El flujo de entrada, compuesto por los mensajes que los jugadores envían al juego, no sufre ningún tipo de transformación. El motivo es evitar desequilibrios o medidas que favorezcan a unos jugadores frente a otros.

3.1.2 Uso flexible

El uso flexible hace referencia a crear un modelo que se acomode al rango de preferencias y habilidades individuales.

En primer lugar, ofrecer el juego en diferentes plataformas tecnológicas permite al usuario escoger y beneficiarse del entorno que más se adapte a sus preferencias. En este sentido, el usuario tiene una forma flexible de acceder al juego.

Por otro lado, el juego puede acomodarse en función de las necesidades que establece el propio usuario. Para ello se han establecido una serie de opciones dentro del juego que permiten la configuración de la visualización del juego. El jugador puede seleccionar de entre un conjunto de opciones predeterminadas de visualización o elegir específicamente las reglas de visualización que requiere para que sus habilidades no se vean mermadas, como se verá en el apartado Ofrecer la posibilidad de personalizar la información mostrada

3.1.3 Simple e intuitivo

Este principio pretende que el diseño sea fácil de entender, atendiendo a la experiencia, conocimientos, habilidades lingüísticas o grado de concentración actual del usuario. En este apartado es importante hacer mención a dos facetas del modelo:

- Hacer simple el juego: si se pretende crear un juego para todos, es importante desarrollar una lógica de juego sencilla y fácil de comprender. Sin embargo, no hay que confundir la sencillez del juego con su grado de dificultad. Si queremos desarrollar un juego social al que accedan diferentes perfiles de usuarios, no podemos crear un juego fácil. Un juego difícil favorecerá la colaboración entre los usuarios y la creación de comunidades sociales. Un juego social, por tanto, debe ofrecer un mínimo nivel de dificultad. Finalmente, el sistema de ayuda dentro de juego facilita el acceso a la información, ofreciendo un entorno amigable y de fácil aprendizaje.
- Hacer simple el acceso al juego: el acceso sencillo al juego es una pieza fundamental para ofrecer el juego al mayor número posible de usuarios. En un entorno basado en texto, no es atractivo ofrecer complejas interfaces. En este sentido el modelo elimina las complejidades innecesarias, ofreciendo un acceso sencillo e intuitivo a través de los diferentes terminales en los que se tiene acceso al juego.

3.1.4 Información perceptible

El objetivo de este principio es comunicar de manera eficaz la información necesaria para el usuario, atendiendo a las condiciones ambientales o a las capacidades sensoriales del usuario. Por un lao, el modelo de accesibilidad se ha diseñado para que la información sea ofrecida de forma similar en todos los casos de juego, ampliando la legibilidad de la información.

De forma externa al juego, la plataforma ofrece información accesible mediante la web. Esta información es retransmitida de forma redundante mediante diferentes técnicas de presentación, con el objetivo de que los usuarios puedan escoger y adaptarse al medio de comunicación más adecuado.

3.1.5 Con tolerancia a error

En este caso, el modelo de accesibilidad referido a este principio se centra en minimizar los riegos y las consecuencias adversas de acciones involuntarias o accidentales dentro del juego. En primer lugar y referido a la interfaz, el juego informa mediante el sistema de ayuda de ejemplos de uso, comandos más usados y acciones típicamente incorrectas. En segundo lugar, el juego ha sido preparado para responder ante la introducción de comandos incorrectos, permitiendo siempre la cancelación y vuelta atrás en el caso de realizar algún proceso guiado por pasos. Por otro lado, el sistema informa al jugador mediante advertencias sobre posibles peligros y amenazas. Esto último le permite reaccionar a tiempo y no verse sorprendido por algún evento inesperado. En último lugar el juego restringe de forma directa algunas acciones que podrían suponer un riesgo para el jugador. De esta forma el jugador pueden sentirse seguro durante el juego y disfrutar del contenido sin miedo a realizar alguna acción que ponga en peligro todo su progreso acumulado hasta la fecha.

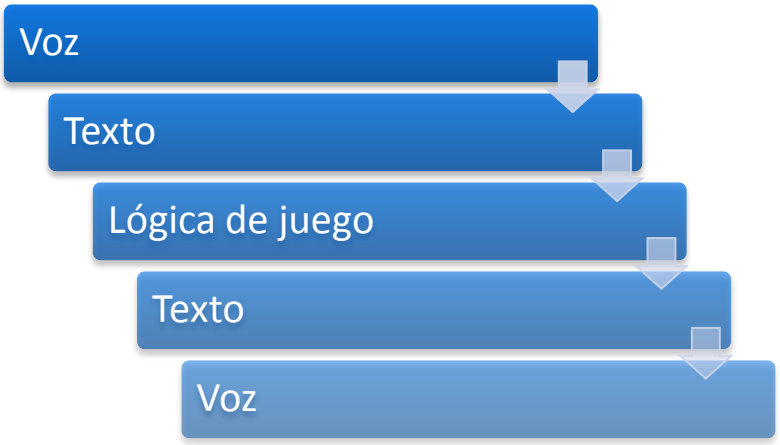
3.1.6 Que exija poco esfuerzo físico

Como es lógico, el acceso a este tipo de juegos requiere de muy poco esfuerzo físico. El entorno, al ser accesible desde múltiples terminales diferentes, puede ser usado de forma eficaz y confortable, siempre y cuando el terminal se adapte a las necesidades físicas del usuario. Siendo un juego basado en texto, los únicos requerimientos de tener un teclado y una pantalla al alcance permite que el usuario adopte cualquier tipo de posición corporal neutra. Por último, la posibilidad de utilizar lectores de pantalla e intérpretes de voz para jugar, incrementa aun más los posibles entornos de juego.

En cuanto al esfuerzo físico, su cantidad dependerá de la forma en la que se interactúa con el sistema. En primer lugar, hay que introducir comandos de texto en el juego, y la respuesta del juego se visualiza también mediante texto. Para personas sin problemas de movilidad en sus extremidades podrán teclear los comandos de juego mediante un teclado, mientras que aquellas que no disponen de la capacidad de teclear podrán utilizar intérpretes de voz para introducir de esta forma los comandos. En cuanto a la recepción del juego, el sistema envía la información en formato de texto. Las personas con discapacidad visual podrán utilizar lectores

de pantalla que conviertan la salida, o bien en sonido mediante intérpretes de voz, o bien a una salida braille.

El siguiente diagrama muestra, a modo de ejemplo, los tipos de medios de comunicación que intervendrían durante la sesión de un jugador que no puede teclear los comandos ni visualizar el texto que es enviado por el juego. En primer lugar utiliza un reconocer de habla para introducir los comandos que son enviados en formato textual (el sistema solo reconoce texto). Luego, el texto enviado por el sistema es retransmitido mediante un sintetizador de voz.



3.1.7 Tamaño y espacio para el acceso y uso

Este principio es referente a que el modelo proporcione un tamaño y espacio apropiados para el acceso, alcance, manipulación y uso, atendiendo al tamaño del cuerpo, la postura o la movilidad del usuario. Sin embargo, ya que se trata de una aplicación virtual, este principio no puede ser aplicado al modelo. El acceso y uso de la aplicación dependerá del terminal elegido por el usuario.

3.2 Entorno de juego

Si queremos realizar un modelo accesible por todos, deberemos atender de forma más específicas las necesidades del entorno de juego. Existen multitud de necesidades y características que diferencian a los usuarios posibles de la aplicación. Para crear un modelo que permita a todo tipo de persona acceder al juego será necesario crear un conjunto de reglas específicas para cada ámbito posible de actuación. A continuación se describen los principales puntos de actuación referidos al entorno de juego.

3.2.1 Opciones de percepción

Cuando la aplicación presenta la información de juego el usuario la recibe mediante su percepción. Pero no todos los usuarios podemos utilizar los mismos sentidos para ello. Por ello es necesario ofrecer al usuario, por un lado, la posibilidad de personalizar la información que quiere recibir, y por otro, el cómo recibirla.

3.2.1.1 Ofrecer la posibilidad de personalizar la información mostrada

Cuando un usuario con discapacidad visual accede al juego, probablemente prefiera recibir determinada información en primera lugar. Por defecto, el sistema envía la información del juego según un orden establecido. Sin embargo, en determinadas situaciones el usuario puede preferir recibir en primer lugar un tipo de información antes que otra, con el objetivo de poder reaccionar lo antes posible.

Supongamos una situación de juego en la que el usuario con discapacidad visual (que utiliza un lector de pantalla para jugar) se encuentra visitando una zona enemiga en busca de algún tesoro. Al entrar en una sala del mundo puede encontrarse esta visualización de juego:

```
Estepa 209,193 Salidas: todas
Ronin está aquí.
```

Ejemplo de juego 3: Visualizando una sala

Tras recibir esta información el lector de pantalla comenzará a leer el mensaje recibido de izquierda a derecha y de arriba abajo. La información sobre la sala a la que ha entrado el personaje se leerá en el siguiente orden: tipo de la sala, coordenadas de la sala, salidas disponibles y personajes que ya se encuentran en esa sala. Ahora supongamos que el personaje *Ronin* es un enemigo del bando contrario al usuario que ya se encontraba en la sala. Lo más probable es que el usuario prefiera salir de la sala lo antes posible y evitar así un posible combate. Sin embargo, dada la velocidad del lector de pantalla, puede transcurrir más de un segundo antes de que el usuario se perciba que hay un enemigo en la sala a la que acaba de entrar.

Por este motivo el juego se ha diseñado para que los usuarios puedan establecer el orden de visualización de determinadas situaciones de juego. Así, por ejemplo, un usuario con discapacidad visual podrá percibir en primer lugar la información más concreta de la sala y por último la más genérica. Siguiendo el ejemplo anterior, el usuario podría modificar el orden de percepción de la sala para recibir la información en el siguiente orden: personajes que ya se encuentran en la sala, salidas disponibles, tipo de sala y coordenadas de la sal.

```
Ronin está aquí.
Salidas: todas Estepa 209,193
```

Ejemplo de juego 4: Visualización ordenada de una sala

3.2.1.2 Ofrecer alternativas a la información visual

Este apartado corresponde a cualquier información visual que puede mostrar el sistema a los usuarios, tanto dentro como fuera del juego.

Por un lado, la información que se muestra dentro del juego solo se presenta mediante información textual, sin imágenes o sonidos. Esta única alternativa permite facilitar las comunicaciones y por ello extender el sistema a un mayor número de dispositivos. También permite que sea el usuario el que escoja un método apropiado a sus necesidades para transformar esa información visual (texto) a otro formato que sea más accesible para él. Así pues, desde un principio se descartó la posibilidad de que el sistema fuese capaz de interpretar el texto enviado y presentar la información mediante voz, pues cada usuario utiliza sus propios sintetizadores de voz con una configuración personalizada.

En cuanto a la información que se expone fuera del juego, el sistema ofrece gran cantidad de datos mediante la página web. Esta información en gran medida es estática, y por tanto puede ser preparada para ser presentada mediante alternativas. En lo referido a las imágenes mostradas en la web, los estándares web (analizados en el capítulo Usabilidad y accesibilidad web) establecen las pautas necesarias para que sean accesibles por personas con problemas de visión. Para el resto de contenidos web se propone un futuro trabajo que incluya la transcripción de esos contenidos a otros medios, como pistas de audio o videos explicativos.

3.2.2 Opciones de lenguaje

Una vez el usuario es capaz de acceder al juego y su percepción del mismo es correcta, es preciso preparar el entorno para que todos los usuarios utilicen un mismo lenguaje comprensible. Un gráfico que ilustra la información de un lugar puede ser de carácter informativo a un usuario y de difícil acceso o desconcertante a otro. Una foto o imagen que tiene un significado para algunos jugadores pueden tener significados muy diferentes para los jugadores de diferentes antecedentes culturales o familiares. Como resultado, las desigualdades surgen cuando se presenta la información a todos los usuarios a través de una sola forma de representación. Una estrategia de exposición importante es asegurarse de que las representaciones alternativas se proporcionan no sólo para la accesibilidad, sino para mayor claridad y comprensión de todos los usuarios.

3.2.2.1 Aclarar el vocabulario y símbolos

Los elementos semánticos a través del cual se presenta la información - las palabras, símbolos, números – no son igualmente accesibles a los usuarios con diferentes orígenes, lenguas y conocimiento léxico. Para garantizar la accesibilidad para todos, el vocabulario, las etiquetas y los símbolos deben ser vinculados o relacionados con las representaciones alternativas de su significado (por ejemplo, un glosario incorporado o definición, un equivalente gráfico, un gráfico o mapa). Modismos, expresiones arcaicas, frases cultural exclusiva, y el lenguaje popular, debe ser traducido.

Con el objetivo de lograr este principio la aplicación, en primer lugar, utiliza el menor número posible de símbolos y representaciones gráficas en su presentación e intenta representar de forma prosaica la mayor cantidad posible de situaciones. De esta forma, además, el colectivo con discapacidad visual puede percibir de forma más adecuada la información que es enviada por la aplicación. Por otro lado, el juego ofrece un fuerte sistema de ayuda basado en relaciones que vinculan los símbolos con un ambiente de elementos con definiciones similares. Así pues, el comando *'susurrar'* se relaciona con el comando *'decir'*. Finalmente, el usuario con discapacidad visual completa (y que utilice lectores de pantalla) puede configurar su cuenta de jugador para no recibir elementos visuales, símbolos o gráficos dentro del juego. Como alternativa recibe un texto alternativo que describe el elemento oculto.

3.2.2.2 Promover el entendimiento entre los idiomas

Aun que este aspecto no ha sido realmente implementado, es importante comprender que la accesibilidad de la información se reduce cuando no hay alternativas lingüísticas disponibles. Proporcionar alternativas, especialmente para la información clave o el vocabulario es un aspecto importante de la accesibilidad. Sin embargo, esto requiere de un gran trabajo que no ha sido implementado en el proyecto. En este sentido se ha decidido que la aplicación se ofrezca sólo en lengua castellana, tanto la información disponible dentro del juego como la accesible mediante la web.

3.2.2.3 Ilustrar a través de múltiples medios de comunicación

La presentación, tanto dentro como fuera del juego, está dominada por la información textual. Pero el texto es un formato débil para la presentación de muchos conceptos y la explicación de la mayoría de los procesos. Además, el texto es una forma particularmente débil de presentación para los usuarios que tienen discapacidades relacionadas con el texto o los idiomas. Proporcionar alternativas - sobre todo las ilustraciones, simulaciones, imágenes o gráficos interactivos - puede hacer que la información en texto sea más comprensible para cualquier usuario. Para ello, y en lo referente a la página web, se ha optado por incorporar en un futuro grabaciones de audio que reproducen el texto mostrado en las diferentes páginas de la aplicación. Es necesario recordar que el juego está basado en texto, y salvo pequeñas ilustraciones o dibujos realizados mediante símbolos, es imposible representar dentro del juego otro medio alternativo al texto.

3.3 Terminales

Este apartado pretende presentar la situación actual de los diferentes terminales con acceso al juego y su nivel de accesibilidad.

En primer lugar, se define un terminal con acceso al juego como un conjunto hardware y software que permite el uso del juego mediante una conexión a Internet. Así pues, un ordenador convencional puede considerarse un terminal posible de juego. Un móvil o una consola también pueden ser considerados terminales de juego. Dentro de cada terminal existen

varios métodos o formas de conexión. Las posibles formas de conexión y su desarrolló se explican en el capítulo Juego multiterminal.

En cuanto al papel de los terminales dentro del diseño para todos, son varios los aspectos que han de tenerse en consideración. Cuanto mayor sea el número de terminales con dichas características, mayor será el posible número de usuarios que tendrá acceso al juego.

3.3.1 Software independiente del hardware

Si consideramos un terminal que permite la conexión al juego independientemente de su soporte hardware, sus ventajas son evidentes. Esto permite que la mayoría de usuarios utilicen el mismo medio de conexión, ahorrando así gran cantidad de esfuerzo para distribuir el juego y dar soporte específico para cada medio. Por otro lado, los esfuerzos podrían centrarse en ofrecer mejoras y características especiales para dicho medio de conexión, ofreciendo una mejor experiencia de juego.

Cuando se habla de software independiente del hardware es inevitable pensar en aplicaciones multiplataforma. La tecnología *Java* es un ejemplo de ello. Actualmente existen varios clientes Java que permiten la conexión mediante el protocolo *telnet* a este tipo de juegos basados en texto. Sin embargo, los clientes actuales no ofrecen una gran calidad en cuanto a la representación de texto. Por otro lado, las aplicaciones Java proporcionan la característica multiplataforma para sus aplicaciones gracias a que la ejecución de la aplicación se realiza dentro de una máquina virtual Java (Java Runtime) que interpreta el código de la aplicación Java. Hay diversos intérpretes de Java para cada plataforma Software. El problema de esta máquina virtual es que, por motivos de seguridad, se aísla del sistema operativo anfitrión, por lo que ciertas ayudas técnicas, como los lectores de pantalla, no pueden acceder al árbol de objetos del interfaz de la aplicación Java (Tyflos, Java y lectores de pantalla - Artículo de Weblogs de Discapnet, 2007) y estás dejan de ser accesibles para usuarios con discapacidad visual. Además, este hecho decremento el rendimiento de la aplicación frente a aplicaciones que aprovechan los recursos del sistema de forma más eficiente.

Existen tecnologías que también son independientes de la plataforma hardware. Así pues, el lenguaje HTML ha demostrado su gran potencial en los últimos años con el gran auge de las páginas web. Ofrecer un medio de conexión al juego mediante este lenguaje es, sin duda alguna, la forma actual más beneficiosa de ofrecer el juego. Sin embargo, HTML no está diseñado para permitir este tipo de conexión (dual) al juego. Por suerte, la nueva versión de HTML (*HTML5.0*) trae consigo una serie de utilidades para poder realizar esta conexión. Pese a que esta nueva versión aun no está disponible para la mayoría de navegadores, el proyecto incluye la una aplicación cliente escrita en HTML5.0, descrita en el apartado Cliente HTML5, que permita la conexión al juego desde cualquier navegador con soporte para este lenguaje.

3.3.2 Terminales configurables

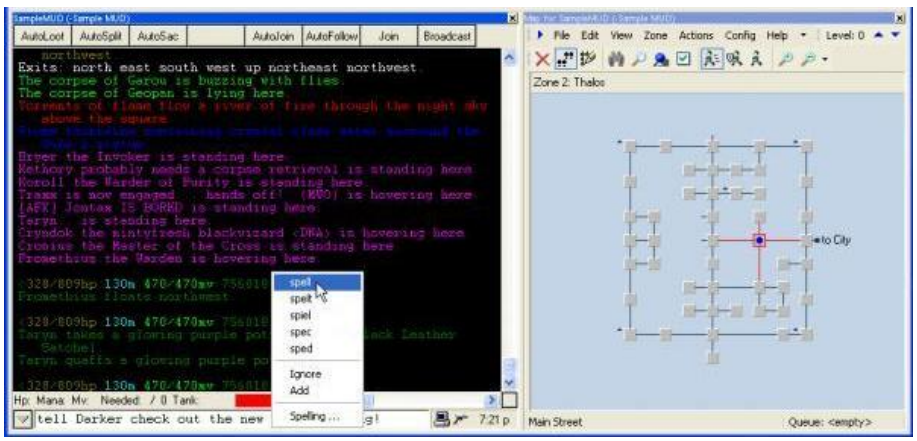
Cada usuario tiene sus necesidades y su modo de juego. Por ello, es fundamental que el terminal de conexión que utilice pueda ser configurado y adaptado a sus preferencias. En este sentido, la mayoría de aplicaciones permiten configurar el terminal. También es importante que dichas opciones se almacenen para su posterior aplicación, y así el usuario no debe establecer las mismas preferencias en cada sesión.

Dentro de las configuraciones posibles, es importante destacar dos tipos de configuración:

- Referidas a la presentación: permiten configurar el modo de visualización de la información que el juego envía. Esto es fundamental tanto para los usuarios con o sin discapacidad que prefieren visualizar la información mediante un formato específico, o bien para usuarios con discapacidad visual que requieren un formato accesible: tamaños de fuente, contraste de colores, síntesis de información,…
- Referidas a la usabilidad: muchas aplicaciones ofrecen preferencias de usuario referidas a su forma de interactuar con el sistema. En la mayoría de los casos se trata de opciones de interfaz de usuario, como abreviaturas, macros, disparadores, atajos de teclado o visualizadores de mapas,…. Todas ellas mejoran la experiencia de juego, pero en ninguna ocasión favorecen al jugador, pues la lógica del juego no tiene constancia si quiera del tipo de cliente que se encuentra conectado.

3.3.3 Que mejoren la experiencia de juego

El hecho de que el juego envíe únicamente información al usuario en formato textual no implica que la experiencia de juego se vea reducida a la interpretación de este formato. Multitud de clientes incorporan complementos que mejoran la experiencia de juego, añadiendo nuevas funcionalidades al modo de juego. Algunos clientes, por ejemplo, incorporan creadores de mapas. Estos complementos permiten al usuario crear mapas que ilustran el entorno de juego y hacer uso de ellos para conocer en cada momento la posición del jugador.



3.3.4 Que cumplan estándares de accesibilidad

Muchas aplicaciones se diseñan y desarrollan pensando en que su atractivo visual debe ser fundamental para poder así atraer al mayor número posible de usuarios. Sin embargo, esta filosofía empeora en muchos casos la accesibilidad de la misma. Si queremos un juego para todos, los terminales deberán ser accesibles al mayor número posible de personas.

Existen varias normas y estándares de accesibilidad establecidos, tanto como para páginas web como para aplicaciones de escritorio. En lo referido a los estándares web, y como se verá en el siguiente apartado Usabilidad y accesibilidad web, existe legislación, existen normativas y existen pautas para así proporcionar un acceso equitativo de la web. En cuanto a las aplicaciones de escritorio, cada gran plataforma de desarrollo (Microsoft para Windows, Apple para MAC) establece una serie de pautas de accesibilidad para las aplicaciones desarrolladas bajo su tecnología. En general, las aplicaciones con interfaz gráfica de usuario suponen un obstáculo para los lectores de pantalla. Por ello, los diseñadores de sistemas operativos y aplicaciones han intentado atacar estos problemas proporcionando vías de acceso a estos lectores para acceder a los contenidos sin tener que mantener ningún "off-screen model" (modelo fuera de pantalla). Esto implica la provisión de un acceso alternativo a lo que se muestra en pantalla a través de un API.

Por último es preciso recalcar que la legislación español establece en su artículo 8 del Real Decreto 1494/2007, de 12 de noviembre, el *Reglamento sobre las condiciones básicas para el acceso de las personas con discapacidad a las tecnologías, productos y servicios relacionados con la sociedad de la información y medios de comunicación social*. Dice así:

> ***Artículo 8. Condiciones básicas de accesibilidad a los equipos informáticos y a los programas de ordenador.***
>
> ***1. Los equipos informáticos y los programas de ordenador – independientemente de que sea libre o esté sometido a derechos de patente o al pago de derechos– utilizados por las administraciones públicas, cuyo destino sea el uso por el público en general, deberán ser accesibles a las personas mayores y personas con discapacidad, de acuerdo con el principio rector de «Diseño para todos» (2) y los requisitos concretos de accesibilidad exigidos, preferentemente en las normas técnicas nacionales que incorporen normas europeas, normas internacionales, otros sistemas de referencias técnicas elaborados por los organismos europeos de normalización o, en su defecto, normas nacionales (Normas UNE 139801:2003 y 139802:2003), y en los plazos establecidos en el apartado 1 de la disposición transitoria única del real decreto por el que se aprueba el presente reglamento***

3.3.5 Que faciliten el uso de programas de accesibilidad

La mayoría de usuarios con algún tipo de discapacidad recurren a programas de terceros que les facilitan el uso y manejo del ordenador. En concreto, los discapacitados visuales utilizan lectores de pantalla para acceder al juego y poder así interpretar la información que el juego envía.

Pero no basta con utilizar un lector de pantalla. Las aplicaciones utilizadas usando ese soporte deben estar preparadas o ser compatibles con estos programas de lectura. Como se vio anteriormente en el capítulo de

Software independiente del hardware, no todas las aplicaciones están preparadas para ser usadas junto con lectores de pantalla. Aplicaciones web creadas con *Adobe Flash Player* o *Java*, por ejemplo, siguen siendo inaccesibles por este tipo de programas. Así pues, es necesario que, si queremos un juego para todos, no solo desarrollemos aplicaciones para cada terminal, sino que nos aseguremos de su compatibilidad con los programas de accesibilidad.

En cuanto a programas de accesibilidad para personas con discapacidad visual, el programa *JAWS* es el máximo exponente del mercado. Actualmente se encuentra en su versión 12, exclusiva para sistemas operativos *Windows*. Su licencia es muy elevada (más de 1000 dólares), pero ha sido posible obtener una licencia de prueba gratuita durante el periodo de pruebas que ha requerido el proyecto. Dada su gran integración con el sistema operativo Windows, *JAWS* ha sido siempre el lector de pantalla más utilizado en esta plataforma. Como puede comprobarse en (ONCE, 2009), este programa es comúnmente utilizado por las organizaciones para evaluar la accesibilidad de los sistemas.

3.4 Navegadores y accesibilidad

Uno de los aspectos clave del sistema es la posibilidad de jugar mediante la página web. Además, mucha de la documentación de ayuda se encuentra también disponible en la misma página. Por este motivo es imprescindible analizar el estado actual respecto al diseño para todos y la accesibilidad de los navegadores.

En primer lugar, los navegadores web son los encargados de interpretar el código HTML y lo hacen, en muchos casos, tomando decisiones dispares. Por este motivo surgen algunos problemas a la hora de diseñar páginas. Gran cantidad de webs pueden ser visualizadas de forma distinta dependiendo del navegador con el que se acceda a ellas. Esto es un gran problema de accesibilidad. Si las webs no funcionan de la misma forma para todos los usuarios estos pueden quedar discriminados. Por suerte, las pautas creadas de accesibilidad para el contenido en la Web tienen en cuenta estas características y la responsabilidad recae exclusivamente en la laboral del administrador web. Podríamos decir que los navegadores no suponen un obstáculo a la accesibilidad de las webs, y sí el incumplimiento de las normativas por parte de los desarrolladores.

Por otro lado, al igual que ocurre con el resto de terminales, es importante analizar la compatibilidad de los navegadores web actuales con los lectores de pantalla. JAWS es el lector de pantalla más utilizado por el colectivo de personas con discapacidad visual, y solo Internet

Explorer es completamente compatible con esta aplicación. No obstante, el resto de navegadores (Firefox, Opera, Safari, Chrome,…) pueden ser utilizados usando JAWS. La diferencia radica en el nivel de compatibilidad. Internet Explorer, gracias a las APIs Microsoft Active Accessibility (MSAA), permite que JAWS acceda al Document Object Model (DOM) del la página web visitada, facilitando la lectura de la página (Tyflos, Navegar por Internet con Jaws - Artículo de Weblogs de Discapnet, 2007). Para el resto de navegadores, JAWS realiza una lectura de la web de forma predeterminada, como si se tratase de una aplicación de escritorio, siguiendo el orden de tabulación lógico establecido.

Sin embargo, como se verá en el siguiente capítulo, por motivos tecnológicos, actualmente sólo Google Chrome y Safari pueden ser utilizados para conectarse al juego mediante la página web. En el siguiente capítulo se analiza esta problemática y otras relacionadas con la accesibildad.

3.5 Interacciones que facilitan la accesibilidad

En primer lugar, el sistema permite acceder al juego mediante una página web creada. Esta página se sirve de la nueva tecnología *HTML5*, y en especial de *WebSockets*, para poder realizar la conexión dual necesaria para este tipo de juegos. Durante el transcurso del juego el usuario recibe información del sistema sin abandonar nunca la página, de forma que los contenidos de la web se actualizan de forma dinámica.

Dado el reciente lanzamiento de esta versión del lenguaje básico de la *World Wide Web*, sólo los navegadores *Google Chrome* (a partir de su versión 4) y *Safari* (a partir de su versión 5) soportan esta tecnología (Deveria, 2011).

Otra mala noticia es que no es posible hacer funcionar HTML5 en todas las tecnologías de asistencia conocidas, de la misma manera que no es posible utilizar HTML5 para trabajar con los navegadores más antiguos, pero la cuestión fundamental es: ¿cómo informar a los usuarios con lectores de pantalla de que un cambio ha tenido lugar en la página, y cómo pueden interactuar con el contenido?

Para entender los problemas de la accesibilidad de *WebSockets* respecto a los lectores de pantalla, es esencial tener una comprensión de cómo los lectores de pantalla realizan su trabajo. Para que los usuarios de lectores de pantalla puedan leer e interactuar con el contenido web, los lectores de pantalla toman una instantánea de la página web, y también de la estructura de la web en un modo virtual. El lector de pantalla utiliza el modo virtual para permitir al usuario navegar por el contenido siguiendo el Object Model (DOM) del la página. Sin el modo virtual, el lector de pantalla sólo tiene acceso a las partes de la página que se que pueden recibir el foco de usuario, como botones o cuadros de texto. Sin el modo virtual, el usuario no puede interactuar con otros elementos y sus nodos secundarios, como imágenes, listas, tablas, etc. Los lectores de pantalla tienen su propio nombre específico para el almacenamiento virtual, como el modo de *foco virtual* en *Supernova*.

El modo virtual se conoce como *Virtual PC Cursor* en JAWS. El modo Virtual PC Cursor está activado por defecto para ver documentos HTML en aplicaciones compatibles (como Internet Explorer y Firefox con JAWS 7.0), y puede activarse y desactivarse mediante la combinación

de teclas Insert + Z. En este modo, el usuario tiene acceso a los elementos HTML y sus atributos.

Cuando está en modo Virtual PC Cursor, JAWS responde de manera incorrecta a los eventos de script producidos por el cliente. JAWS es capaz de responder a los eventos fundamentales en el modo Virtual PC Cursor, tales como hacer clic u oprimir una tecla, y la instantánea tomada por el Virtual PC Cursor se actualizará para reflejar cualquier cambio que se haya producido en el contenido. El problema cuando se utiliza Ajax o WebSockets es que el contenido nuevo no se suele añadir directamente en respuesta a estos eventos, sino que el contenido se agrega generalmente a través del evento onreadystatechange del objeto XMLHttpRequest o en el caso de WebSockets a través del evento onmessage. Curiosamente, JAWS 7.0 responde al evento onreadystatechange con Firefox, pero no con Internet Explorer. El único método fiable de la adición de nuevos contenidos en el modo Virtual PC Cursor es como un resultado directo de un evento de clic, o de oprimir una tecla, o un evento mouseover, que obviamente no es una solución suficientemente buena para hacer accesible AJAX o WebSockets.

Desde la versión 6, JAWS introducido un comando para actualizar el Virtual PC Cursor utilizando la combinación de teclas Insert + Esc. Esto significa que un usuario de JAWS puede utilizar esta combinación de teclas para acceder al contenido que se ha añadido de forma dinámica. El problema es informar al usuario de que el contenido ha cambiado. Una solución que se consideró fue la de añadir contenido que pidiese al usuario que actualizase el buffer virtual como una respuesta directa a un evento de clic / oprimir una tecla, y reemplazar el contenido cuando el evento onreadystatechange o onmessage se desencadenara. La idea detrás de este enfoque es que los usuarios de JAWS 6 y versiones posteriores podrían utilizar el comando de actualización de Virtual PC Cursor, y los usuarios de versiones anteriores y otros lectores de pantalla que utilizan un modo virtual serían capaz de responder a este evento activando y desactivando el modo virtual - por ejemplo, mediante la combinación de teclas Insert + Z en JAWS.

Al probar esta técnica con Internet Explorer, el mensaje para actualizar el almacenamiento virtual es anunciado por JAWS, pero la actualización del buffer mediante la combinación de teclas Insert + Esc sólo anuncia que la pantalla se ha actualizado, y no empieza a leer el contenido actualizado. Activar y desactivar el modo Virtual PC Cursor repetidas veces solo implica que JAWS informe de que se ha cambiado el modo de lectura. Como JAWS con Firefox responde al evento onreadystatechange, el contenido del evento click es reemplazado por el contenido del evento onreadystatechange, y anunciado para el usuario. Como Firefox puede responder directamente al evento onreadystatechange, y alternar los modos no anuncia nuevos contenidos (Faulkner, 2006), no vale la pena indagar más en esta técnica.

El modo PC Cursor es lo contrario al modo Virtual PC Cursor, ya que no utiliza un buffer virtual. Este es el modo que utiliza JAWS cuando trabaja sobre navegadores diferentes a Internet Explorer y Firefox. En este modo, el usuario interactúa directamente con la aplicación, y sólo puede centrarse en los elementos que pueden recibir el foco. Si Ajax o WebSockets se utiliza en respuesta a un evento, entonces puede funcionar en este modo. A pesar de que el usuario tiene una capacidad muy limitada en el modo PC Cursor, se pueden activar los enlaces y botones. Si el contenido generado por el evento onreadystatechange puede recibir el foco, el usuario será notificado y puede acceder al contenido generado en este modo. La especificación

de HTML sólo permite que ciertos elementos reciban el foco de atención. Sin embargo, aun existe el problema de informar al lector de pantalla que el contenido ha cambiado.

Si el contenido ha cambiado mediante script, entonces este evento tiene que ser retransmitido al lector de pantalla. Sin un mecanismo para descubrir qué ha cambiado, un usuario con lector de pantalla no recibirá ninguna notificación de que el contenido ha cambiado en absoluto, o sólo le notificará que el contenido ha cambiado, pero será necesario leer todo el documento para descubrir exactamente lo que ha cambiado. Por ejemplo, cuando el contenido se inserta en el documento mediante script, Window-Eyes (otro lector de pantalla) responde lo siguiente:

Loading Page. Load Done. Looking for visible line.

La última línea visible es la línea que contiene el elemento que el usuario activó para producir el nuevo contenido, por lo que el usuario no tiene idea de cómo encontrar lo que ha cambiado. Para evitar esto, el método de enfoque de ECMAScript se puede utilizar para poner el foco en la parte de la página que ha cambiado. Para que esto funcione, el elemento de destino debe ser un elemento que pueda recibir el foco. En HTML (y XHTML) los únicos elementos que puede recibir el foco son: a, área, button, inputobject, select, y los elementos textarea.

En XHTML 2, todos los elementos serán capaces de recibir el foco, pero el problema que tenemos con las especificaciones de HTML 4.01 y XHTML 1.x es que no permiten que otros elementos de la interfaz puedan recibir el foco. Para solucionar este problema, *The Web Accessibility Initiative's Protocols and Formats working group* proponen *Dynamic Accessible Web Content Roadmap*, que sugiere el uso de un valor tabindex de -1 en elementos que no pueden recibir el foco de acuerdo a la especificación, y formalizar este con *States and Adaptable Properties Module*. La necesidad de hacer todos los elementos enfocables también ha sido reconocido por *Web Applications 1.0* (Faulkner, 2006).

El atributo tabindex acepta un valor entre 0 y 32767. Un valor positivo determina el orden en que el elemento será visitado durante la navegación mediante el teclado. El valor 0 significa que el elemento será visitado en el orden en el que se produciría naturalmente en el documento fuente. La asignación de un valor del atributo tabindex de 0 a un elemento de la interfaz significa que el elemento pueda recibir el foco con ECMAScript, pero puede ser confuso para los visitantes, ya que será capaz de navegar en el elemento con el teclado. La asignación de un valor del atributo tabindex de -1 significa que el elemento pueda recibir el foco con ECMAScript, pero no será colocado en el orden de tabulación. Internet Explorer y Firefox apoyan el valor del atributo tabindex de -1, pero esto no funciona con otros navegadores. Por ejemplo, el valor del atributo tabindex de -1 es ignorado por Safari, y el no recibe el foco con el método de enfoque de ECMAScript.

Cuando el foco se le da a un elemento que tiene un valor negativo tabindex y que normalmente no recibe el foco, como un elemento de párrafo, el comportamiento es diferente según el modo. En el modo de almacenamiento virtual, el lector de pantalla se centra en el elemento, pero no dará a conocer automáticamente el contenido del elemento. Si el modo de almacenamiento virtual está apagado, el lector de pantalla anuncia el contenido del elemento. En el modo de almacenamiento virtual, el usuario debe dar al lector de pantalla la orden de leer la línea actual

después de que un elemento reciba el foco (Insert + flecha hacia abajo en JAWS y flecha abajo en el Window-Eyes y Supernova).

Existen varios enfoques que se pueden utilizar para estructurar el contenido de una aplicación Ajax o WebSockets para que funcione con lectores de pantalla. El más simple, y probablemente la mejor opción para una aplicación, es hacer que las partes de la aplicación que deben ser activadas se situen en un formulario. Con este enfoque, el usuario puede ser informado del cambio, centrándose en la parte del documento donde se produjo el cambio. Esto requiere que el usuario alterne de modo virtual a modo cursor, pero al fin y al cabo esto es lo que el usuario espera de todos modos en la interacción con formularios. Sin embargo, si la parte modificada del contenido está en el propio formulario, esto presenta un problema. Aunque el usuario puede obtener el texto, no será capaz de interactuar con el contenido como lo haría si estuviera en el modo de almacenamiento virtual. Como ejemplo, considere una tabla de datos que está incrustada en un formulario (en sí mismo, esto es indicativo de que la estructura se puede mejorar, pero vale la pena considerarla, ya que es un escenario plausible). Si una celda de datos en una tabla que está incrustada en un elemento del formulario se actualiza en respuesta al evento onreadystatechange, el enfoque se puede dar a la celda de la tabla, y el lector de pantalla lo anunciará con éxito. El problema es que la tabla que contiene el texto no se reconoce, lo cual significa que el usuario no será capaz de determinar las cabeceras, o ser capaz de navegar por el resto de la tabla. Para hacer esto, el usuario tendrá que volver al modo de cursor virtual.

Si la interacción con la aplicación no está en un formulario, lo más importante es informar al usuario de que debe tener el modo de almacenamiento virtual apagado (como lo harían con el fin de utilizar un formulario). Esto no es tan simple como suena, ya que tendrá que estar redactado de modo que sea fácilmente comprensible por los usuarios de lectores de pantalla que no necesariamente comprendan los aspectos técnicos del software que están utilizando; de la misma manera que los usuarios de navegadores visuales no necesariamente saben cómo cambiar el tamaño del texto. Una solución sencilla sería tener una página de ayuda que es fácilmente detectable a partir de la aplicación que explicase los modos virtuales, con una tabla que contuviera los comandos para cambiar el modo virtual en todos los lectores de pantalla populares. La tabla siguiente muestra cómo cambiar el modo virtual en los lectores de pantalla más utilizados:

Tabla 1: Comandos para cambiar el modo virtual

Lector de pantalla	Convinación de teclas
JAWS	Insert + Z
Window-Eyes	Control + Shift + A
Supernova	Left Control + 4

Esta página sin embargo no será necesaria incluirla en nuestro proyecto, pues el modo de cursor virtual no está disponible para los navegadores Google Chrome y Firefox, únicos navegadores compatibles con la tecnología WebSockets a fecha de hoy.

Como conclusión, el beneficioso modo de uso de cursor virtual que algunos lectores de pantalla ofrecen, además de no estar disponible para todos los navegadores, no asegura la compatibilidad con tecnologías como AJAX o WebSockets. Si queremos notificar al lector de pantalla que un contenido ha sido modificado mediante script, deberemos utilizar el lector en el modo cursor y establecer el foco sobre el contenido que haya cambiado; siempre y cuando este elemento pueda recibir el foco.

Finalmente, este es el fragmento de código *JavaScript* utilizado en la aplicación para agregar contenido a la web y avisar a los lectores de pantalla del cambio que se ha producido.

```
var objAnchor = document.createElement('a');
objAnchor.setAttribute('href', '#target');
objAnchor.innerHTML = Text;

var LogContainer = document.getElementById("log");
LogContainer.appendChild(objAnchor);

LogContainer.scrollTop = LogContainer.scrollHeight;
objAnchor.focus();
```

Código 1: JavaScript para hacer accesible AJAX

3.6 Usabilidad y accesibilidad web

Para que un usuario pueda utilizar la web, ya sea para jugar o para obtener información del juego, es necesario que la página cumpla también los estándares de accesibilidad. En este sentido, *World Wide Web* Consortium ha establecido una serie de normativas referidas a la accesibilidad web.

Como se indica en (W3C, 2008), hablar de Accesibilidad Web es hablar de un acceso universal a la Web, independientemente del tipo de hardware, software, infraestructura de red, idioma, cultura, localización geográfica y capacidades de los usuarios.

Con esta idea de accesibilidad nace la Iniciativa de *Accesibilidad Web*, conocida como WAI (Web Accessibility Initiative). Se trata de una actividad desarrollada por el W3C, cuyo objetivo es facilitar el acceso de las personas con discapacidad, desarrollando pautas de accesibilidad, mejorando las herramientas para la evaluación y reparación de accesibilidad Web, llevando a cabo una labor educativa y de concienciación en relación a la importancia del diseño accesible de páginas Web, y abriendo nuevos campos en accesibilidad a través de la investigación en este área.

La idea principal radica en hacer la web más accesible para todos los usuarios independientemente de las circunstancias y los dispositivos involucrados a la hora de acceder a la información. Partiendo de esta idea, la página accesible lo será tanto para una persona con discapacidad, como para cualquier otra persona que se encuentre bajo circunstancias externas

que dificulten su acceso a la información (en caso de ruidos externos, en situaciones donde nuestra atención visual y auditiva no estén disponibles, pantallas con visibilidad reducida, etc.).

Para hacer el contenido Web accesible, se han desarrollado las denominadas Pautas de Accesibilidad al Contenido en la Web (WCAG), cuya función principal es guiar el diseño de páginas Web hacia un diseño accesible, reduciendo de esta forma barreras a la información. WCAG consiste en 14 pautas que proporcionan soluciones de diseño y que utilizan como ejemplo situaciones comunes en las que el diseño de una página puede producir problemas de acceso a la información. Las Pautas contienen además una serie de puntos de verificación que ayudan a detectar posibles errores (W3C, 2008).

Cada punto de verificación está asignado a uno de los tres niveles de prioridad establecidos por las pautas.

- **Prioridad 1**: son aquellos puntos que un desarrollador Web tiene que cumplir ya que, de otra manera, ciertos grupos de usuarios no podrían acceder a la información del sitio Web.
- **Prioridad 2**: son aquellos puntos que un desarrollador Web debería cumplir ya que, si no fuese así, sería muy difícil acceder a la información para ciertos grupos de usuarios.
- **Prioridad 3**: son aquellos puntos que un desarrollador Web debería cumplir ya que, de otra forma, algunos usuarios experimentarían ciertas dificultades para acceder a la información.

En función a estos puntos de verificación se establecen los niveles de conformidad:

- Nivel de Conformidad "A": todos los puntos de verificación de prioridad 1 se satisfacen.
- Nivel de Conformidad "Doble A": todos los puntos de verificación de prioridad 1 y 2 se satisfacen.
- Nivel de Conformidad "Triple A": todos los puntos de verificación de prioridad 1,2 y 3 se satisfacen.

Las pautas describen cómo hacer páginas Web accesibles sin sacrificar el diseño, ofreciendo esa flexibilidad que es necesaria para que la información sea accesible bajo diferentes situaciones y proporcionando métodos que permiten su transformación en páginas útiles e inteligibles.

Igualmente, se han desarrollado *Pautas de Accesibilidad para Herramientas de Autor*, cuyo objetivo es ayudar a los desarrolladores de software a la hora de crear herramientas de autor para producir contenido Web accesible. También se han desarrollado *Pautas de Accesibilidad para XML*, donde se explica cómo asegurar la accesibilidad de aplicaciones basadas en XML. Y por último, *Pautas de Accesibilidad para Agentes de Usuario 1.0*, donde se explica cómo hacer accesible los navegadores, reproductores multimedia y otras tecnologías asistivas.

Por otro lado, se han desarrollado otro tipo de documentos como las *Técnicas para Pautas de Accesibilidad al Contenido en la Web*, que ofrecen una serie de ejemplos de etiquetado y explicaciones muy detalladas de cómo implementar las Pautas de Accesibilidad al contenido en la Web. Entre ellas se pueden destacar *Técnicas esenciales para Pautas de Accesibilidad al*

Contenido en la Web 1.0, las *Técnicas HTML para Pautas de Accesibilidad al Contenido a la Web 1.0* y las *Técnicas CSS para Pautas de Accesibilidad al Contenido en la Web 1.0*.

4 Desarrollo del juego social

Para conseguir los objetivos de la filosofía *diseño para todos*, uno de los logros fundamentales del proyecto es desarrollar un juego social que, por sus características y jugabilidad, sea capaz de atraer a jugadores de igual forma que lo haría un videojuego. En este capítulo se describe el proceso que se ha seguido para alcanzar esta meta.

En primer lugar se presenta el estado del arte del mercado actual de juegos online masivos, con el objetivo de descubrir las principales características que los hacen tan atractivos. En segundo lugar se presenta el diseño del juego, acompañado de diagramas y esquemas. El tercer apartado versa sobre la lógica de juego implementada. A continuación se presenta el entorno sobre el que se desarrolla el juego, y cómo los jugadores interactúan con él. Como extensión al capítulo anterior se describe el proceso de generación del mundo donde se desarrolla el juego. En el sexto apartado se describe brevemente la economía, y finalmente se presenta el carácter social del entorno.

4.1 Estado del arte

El estado del arte corresponde al estudio de los principales juegos online masivos (massively multiplayer online game, MMOG), con el objetivo de obtener las características principales que hacen a estos juegos tan populares. La finalidad es poder realizar un diseño de juego que atienda a estas características, incorporando las decisiones más acertadas del mercado.

En primer lugar es necesario responder a algunas preguntas, como el número de juegos online actuales o el número de jugadores. En cuanto al número de juegos online, es muy difícil hacer una estimación concreta dada la inestabilidad de este tipo de juegos que dependen tanto del número de usuarios activos. También es difícil considerar el punto en el que un juego pasar a considerarse masivo. Si podemos realizar un pequeño análisis de los principales tipos de juegos online y la relevancia de cada uno de ellos. Más tarde concretaremos cuales son los principales juegos masivos dentro de esa categoría analizando su cantidad de usuarios activos.

En cuanto a los tipos de juegos online masivos, existe una gran variedad de géneros. Podemos hacer la siguiente distinción en un primer nivel:

- Acción
- De navegador
- De navegador en 3D
- Construcción
- Exploración
- Simulación de vuelo
- FPS
- Música
- Puzles
- Juegos de rol
- Estrategia en tiempo real

- Juegos sociales
- Estrategia basada en turnos

Si analizamos el mercado de juegos por géneros podemos encontrar un gran abanico de ellos. Sin embargo, existe una gran diferencia entre uno de los géneros y el resto. Los juegos online masivos de fantasía son los grandes triunfadores del mercado de juegos online. Como puede verse en (Geel, 2010), el 95,4% de las subscripciones corresponden a los juegos de fantasía. Habrá por tanto que analizar cuáles son los principales juegos de este género y sus características.

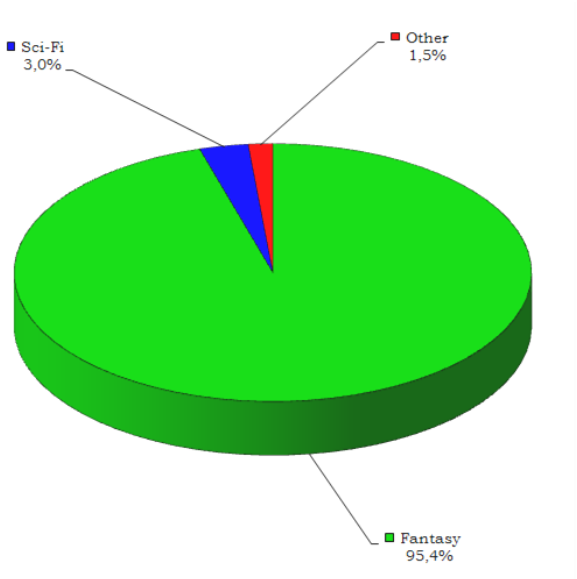


Ilustración 3: Géneros de MMO

Para concretar cuáles son los principales juegos de fantasía online es necesario analizar su número de *subscripciones*, concepto que será explicado a continuación. En la mayoría de juegos online masivos es necesario que el usuario se registre para poder acceder al juego. Sin embargo, en muchos casos el registro no es determinante a la hora de cuantificar la comunidad de usuarios activos de dicho juego, pues muchas cuentas de usuario se encuentran inactivas. Por esta razón no se utiliza la expresión *número de usuarios* o *usuarios registrados*, sino *subscripciones*, que representa al número de usuarios registrados que mantienen una actividad actualmente sobre la plataforma. Una vez más, (Geel, 2010) nos arroja datos sobre la cantidad de subscripciones de los principales juegos masivos de rol online:

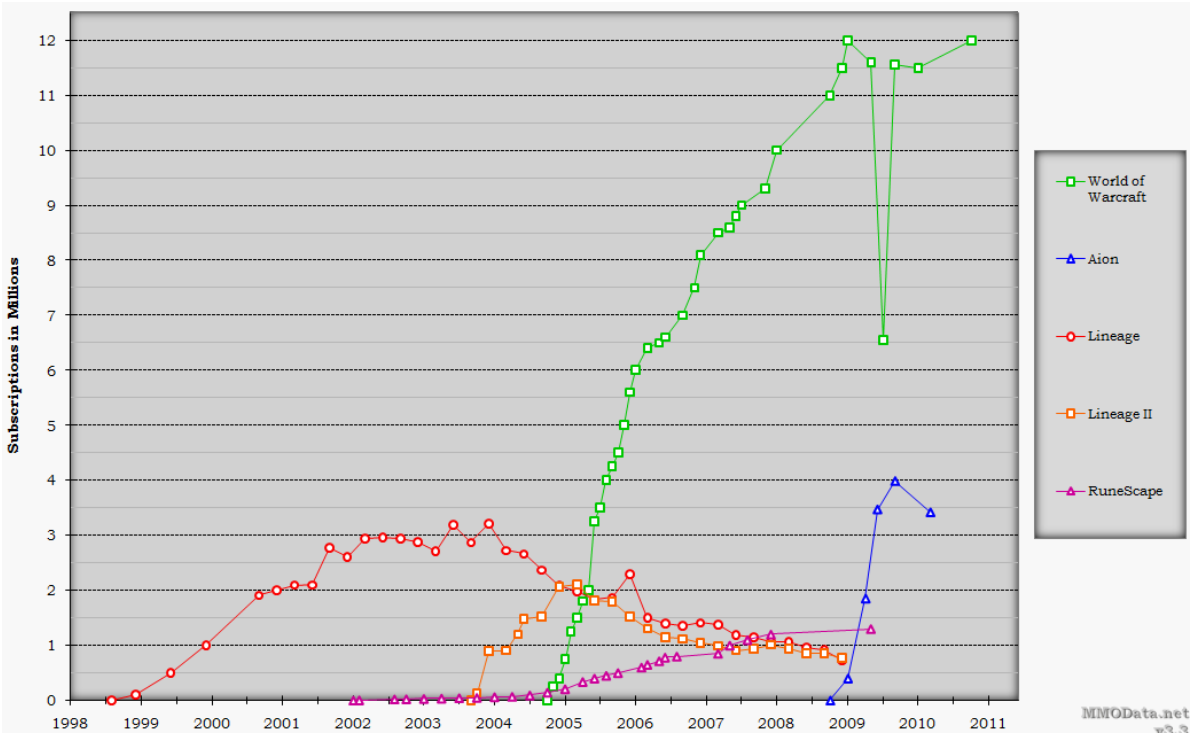


Ilustración 4: Gráfica de subscripciones por juego

Como puede verse, el juego *World of Warcraft* es el indiscutible juego de rol masivo online con mayor relevancia en el mercado mundial con aproximadamente 12 millones de subscripciones activas a finales de 2010. Desde su lanzamiento la mayoría de juegos que ya se encontraban en el mercado han visto reducido su número de subscripciones. Solo los juegos que han sido lanzados al mercado posteriormente a World of Warcraft han conseguido un aumento del número de subscripciones. Se entiende por tanto que este juego ha marcado un hito en la historia de los juegos online, incorporando por un lado un gran número de jugadores al mercado de juegos de rol online, y por otro, estableciendo unas políticas de juego que parecen ser las más adecuadas.

Teniendo en cuenta las estadísticas anteriores, es necesario analizar las características de los juegos de rol masivos en línea para conseguir un modelo de juego que se asemeje a ellos y pueda ofrecer una jugabilidad igualmente atractiva. Entre las características de estos juegos se destacan:

- El juego se desarrolla sobre un mundo persistente compartido por todos los usuarios.
- Un estilo de juego típico de los juegos de rol tradicionales, que incluye misiones y sistema de recompensa basado en la recogida de tesoros.
- Un sistema de desarrollo de estadísticas, normalmente relacionado con niveles y puntos de experiencia.
- Una economía basada en el trueque y una unidad monetaria o más de una.
- Organización de los jugadores en clanes, estén o no soportados directamente por el juego.
- Moderadores del juego y, en algunas ocasiones, personas retribuidas por vigilar el mundo.

Por otro lado, es conveniente analizar las características propias de World of Warcraft para concretar el motivo de su éxito. Sus características principales son:

- No se incluyen contenido para adultos. Es un juego destinado a todos los públicos.
- Un modelo de juego sencillo y fácil de aprender.
- Coste de tiempo logarítmico para la mejora y desarrollo del personaje.
- La imposibilidad de perder desarrollo o avances del personaje.
- La incorporación de modos de juego "jugador contra jugador" y "jugador contra entorno".
- La existencia de clases, razas y roles que aumentan la personalidad del personaje.

Todas estas características han sido incorporadas durante la fase del diseño del juego a razón de obtener, como mínimo, una lógica de juego parecida e igualmente atractiva. En el capítulo siguiente se muestran los diagramas utilizados durante la fase de diseño, aun que la explicación será dada en los siguientes apartados.

4.2 Diseño del juego

Para el diseño del juego se han tenido en cuenta, tanto el modelo creado en el capítulo Modelo de accesibilidad, como las características concretadas en el apartado anterior del capítulo. Para la representación del diseño se han utilizado diagramas UML, creados con la propia herramienta de desarrollo *Visual Studio 2010*, un diagrama de entidad-relación para representar la estructura de tablas de la base de datos y, en último lugar, un diagrama de secuencia desarrollado con la herramienta *Microsoft Visio 2007*.

4.2.1 Diagrama de entidades

En primer lugar se ha optado por un paradigma orientado a objetos, un método muy apropiado para representar un mundo de entidades y entornos. El primer diagrama que se en los Anexos I (Diagrama de entidades) presenta las entidades que se han diseñado y su relación entre ellas. Como puede verse, toda entidad del juego hereda de la clase objeto. Sucesivas herencias proporcionan métodos y propiedades que pretenden representar cualquier tipo de entidad que se encuentre dentro del juego. Desde un jarrón hasta un caballo, pasando por un barco o un trozo de leña recién cortado. Destaca la existencia de entidades que pueden ser *"rotas"*, pues el juego incorpora un sistema de combate que permite realizar batallas entre jugadores o entre jugadores y el entorno. Ya que no todo objeto *rompible* tiene porqué ser un ser vivo (por ejemplo, la puerta de un castillo o una catapulta), se han diseñado las entidades *inaninadas*. Por último, los objetos que no puedan ser destruidos podrán ser *salas* o *irrompibles*. Entre estos objetos irrompibles se encuentran las piezas de equipo que los jugadores podrán equipar y los materiales que podrán utilizar para construir entidades irrompibles. Por últimos, las salas representan el entorno principal en el que se encontrarán los jugadores.

4.2.2 Diagrama del entorno

El siguiente diagrama UML mostrado en los Anexos I muestra las clases que se han creado para implementar el entorno en el que se desarrolla el juego. Pese a que la lógica de juego se ha diseñado para que cualquier objeto pueda ser entorno de otro (por ejemplo, el entorno de una espada puede ser un personaje), es necesario definir clases que representen el entorno básico y lógico de los objetos. Por este motivo se definen las *SalasArtesanas* y las *SalasExteriores*. Cada sala tiene un conjunto de *Salidas* que permiten el acceso a otras salas. Todo esto se describe con más detalle en el apartado Entorno de interacción.

.

4.2.3 Diagrama del mundo

El tercer diagrama de los Anexos I corresponde a las clases que hacen posible la generación dinámica del mundo y sus contenidos. Como se verá en el apartado El mundo, el entorno en el que se desarrolla el juego parte de varias imágenes o capas. El sistema genera salas con las características que se definen en las diferentes imágenes. Seguidamente las salas son rellenadas mediante generadores de contenido. Estos generadores utilizan las características de las salas para determinar los elementos que se incorporarán a las salas. Por ello la clase *Mundo* consta, por un lado, de las capas que definen las características de la sala, y por otro, de una lista de generadores. Además, el mundo consta de una matriz de salas cargadas. Esto último permite ahorrar una gran cantidad de memoria, pues solo se cargan en memoria aquellas salas que son visitadas por los jugadores. Por último la interfaz *ILocalizacion* permite hacer referencia a una sala única del mundo, independientemente de su tipo.

4.2.4 Diagrama de conectividad

El siguiente diagrama de clases UML corresponde a la parte del sistema que permite los diferentes métodos de conexión al juego. Es en este punto, además, donde se incorporan los diseños definidos en el capitulo Modelo de accesibilidad, y por eso se muestran las clases e interfaces necesarias para su implementación.

Se ha creado un modelo basado en capas con el objetivo de cumplir los logros de accesibilidad y de funcionamiento multiterminal; desde que el juego genera un mensaje para un jugador hasta que esa información es enviada en forma de bytes al cliente. En primer lugar, se ha creado una clase *Servidor* que contiene la lista de todos los usuarios que están conectados actualmente al juego. La forma en la que los usuarios están conectados no se define en el servidor, y por tanto la información fluye hacia ellos sin ningún tipo de tratamiento. Los usuarios son los que tratan la información que les llega en función de su nivel de accesibilidad y su forma de conexión. Por un lado, cada usuario tiene asociado una propiedad de tipo *IAccesibilidad* que transforma la información según se ha definido en el apartado Entorno de juego. Por otro, el usuario tiene una propiedad de tipo *IConexion* que permite el envío y recepción de mensajes. Esta clase se adapta al tipo de terminal de conexión que utiliza cada

usuario. De esta forma, por ejemplo, el formato del texto (color, decoración) pueden ser enviados en diferentes formatos (HTML, texto enriquecido o ansi codes). Finalmente, se ha diseñado una clase para cada uno de los servidores creados. Esto último será analizado con mayor detalle en el capítulo Juego multiterminal.

4.2.5 Diagrama general

A razón de unificar todos los subsistemas explicados anteriormente, en el último apartado del Anexo I se presenta un diagrama de clases que engloba la mayoría de clases que forman la lógica de juego. Se han omitido todas aquellas clases que forman grupos, como las clases de los comandos de juego y las clases de las habilidades. En este diagrama se puede visualizar la relación entre los usuarios del juego y los personajes, así como las piezas de equipo de que disponen o las órdenes que puede recibir un animal domesticado.

4.2.6 Diagrama entidad-relación

Finalmente el diagrama entidad-relación incorporado en los Anexos II modela los datos del sistema. El modelo expresa las entidades relevantes para el sistema que deben persistir aun que el juego no se encuentre disponible. Las entidades se dividen en dos categorías: entidades propias del juego y entidades creadas por los jugadores.

En primer lugar, las entidades propias del juego corresponden a los datos necesarios para la generación del mundo y su configuración. Un ejemplo de ello son los tipos de salas que se pueden generar o los diferentes tipos de climas.

Por otro lado, las entidades creadas por los jugadores son los datos que generan los jugadores con sus acciones. En primer lugar, la entidad *Personajes* almacena toda la información referida a los personajes creados por los usuarios. Estos personajes pueden adquirir habilidades, y esto se ve reflejado en las tablas *Habilidades Personaje* y *Habilidades*. El juego puede considerarse social puesto que los jugadores pueden asociarse en grupos creados por ellos mismos. Estos grupos se denominan *Hermandades*. Un jugador solo puede pertenecer a una hermandad al mismo tiempo. Finalmente, los personajes son los protagonistas de su propia historia, y esto se ve reflejado en capítulos que son almacenados mediante la tabla *Capitulos*. La explicación más detallada de la existencia de cada una de las tablas se descubre en el apartado Lógica del juego.

4.2.7 Diagrama de flujo

El siguiente diagrama de secuencia muestra el flujo de mensajes y llamadas que se realiza entre las diferentes clases creadas siguiendo el modelo por capas propuesto con anterioridad. En este modelo se tiene en cuenta, por un lado, el nulo tratamiento que se hace a la información de entrada al sistema, y por otro, la transformación que sufren los informes de

juego antes de ser mostrados al usuario. Como puede verse, la lógica desconoce por completo el tipo de conexión utilizada por el usuario o su nivel de accesibilidad. Por otro lado, la interfaz de conexión tampoco es capaz de interpretar la información de entrada o salida, y se limita a transmitirla. El objeto *usuario* es por tanto el elemento fundamental en la comunicación; esta clase es la encargada de transformar los informes enviados por el juego, tanto a nivel de accesibilidad, como a nivel de presentación.

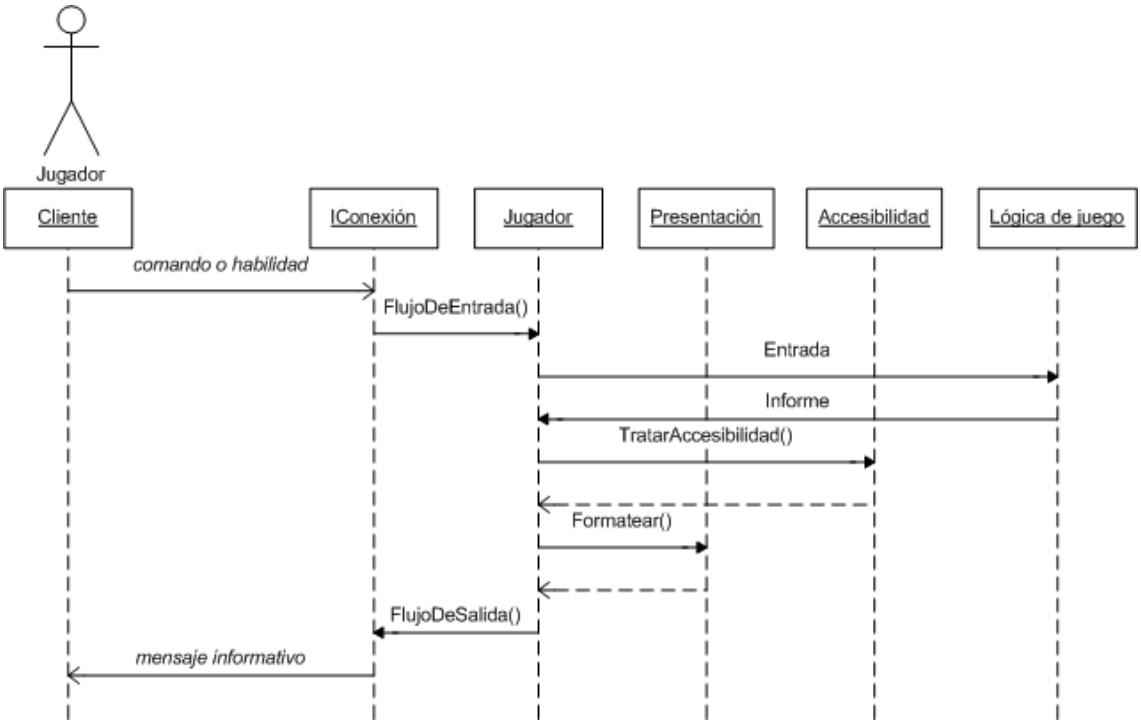


Ilustración 5: Diagrama de secuencia

4.3 Lógica del juego

La lógica del juego se centra en ofrecer un modelo de juego claro y conciso: un juego donde tú eres el narrador de tu propia historia. El objetivo es transmitir al usuario que es el único responsable de sus acciones y que el mundo que le rodea cambiará en función de lo que decida hacer en cada momento.

En primer lugar, un usuario deberá identificarse respecto al juego mediante un personaje. Si el usuario no dispone de un personaje almacenado en la base de datos deberá realizar un proceso de creación de personajes. Se ha barajado la posibilidad de crear una entidad *Visitante* que no pueda interactuar con el mundo, pero si visualizarlo, de forma que pueda ser testigo de lo que ocurra sin tomar parte de la historia; sin embargo, se ha decidido desviar los recursos a implementar otras medidas más oportunas. Todos los personajes comparten un mismo entorno, aun que por las características y las acciones de cada uno de ellos, normalmente se encontrarán en zonas diferentes del mundo.

Una vez el usuario se ha identificado, para interactuar con el mundo y realizar acciones el usuario consta de *comandos* y *habilidades*. El usuario envía mediante su cliente mensajes de texto que deben ser reconocidos como un comando o una habilidad. Como respuesta a esos comandos y habilidades, o a la acción de otro usuario, el jugador recibirá mensajes de respuesta que le informarán de los cambios producidos a su alrededor.

```
>chat hola
[Chat] Jhanda: hola
>dfasfs
¿Cómo? Consulta tus 'comandos' y 'habilidades'.
```

Ejemplo de juego 5: Ejemplo de envíos

Para interactuar con el mundo de la misma forma que otro jugador se utilizan los comandos. Los comandos permiten, sin gasto de recursos alguno por parte del personaje, realizar acciones de la misma forma que haría otro usuario. Estos comandos, aun que no se muestran categorizados a los usuarios, se pueden dividir en:

- Comandos de entorno: permiten interactuar con el entorno que rodea al personaje. Desde aquellos que permiten coger una piedra hasta los que permiten cerrar una puerta. Ejemplos: mirar, coger, leer,…
- Comandos de comunicación: estos comandos permiten comunicarse con otros personajes. Los personajes pueden comunicarse con otros personajes que se encuentren en su misma sala, pero además los usuarios disponen de varios canales de mensajería. Ejemplos: decir, susurrar, chat,…
- Comandos de juego: estos comandos permiten configurar el modo de juego u obtener información sobre el mismo. Ejemplos: ficha, ayuda, opciones,…

Por otro lado, los jugadores tienen la opción de mejorar sus personajes e interactuar con el mundo de forma diferente, dependiendo de su esfuerzo. Por ello se han creado las habilidades. Las diferencias principales entre los comandos y las habilidades son:

- Las habilidades tienen *enfriamiento*: a diferencia de los comandos, un usuario no puede ejecutar dos habilidades encadenadas. Un usuario solo puede ejecutar una habilidad cada segundo. Pero además, cada habilidad tiene un periodo de enfriamiento propio que no permite ejecutar la misma habilidad hasta que transcurra un tiempo determinado.
- Las habilidades gastan energía: los personajes tienen energía, y utilizar habilidades consume parte de esa energía. Pese a que la energía se regenera con el tiempo, este concepto limita la repetición de habilidades consecutivas y favorece el progreso de jugadores casuales.
- Las habilidades permiten realizar acciones diferentes dependiendo del nivel que tenga el jugador en dicha habilidad: cada jugador tiene un conjunto de habilidades y un nivel asociado a cada una de ellas. Al conjunto habilidad-nivel se le ha denominado *talento*. Así pues, un talento mediocre en la habilidad *Cabalgar* permitirá tomar las riendas de monturas de nivel inferior a otras monturas que sólo podrían ser montadas utilizando

un nivel superior en la habilidad. Además, los talentos mejoran con el uso de la habilidad. De esta forma, por ejemplo, un usuario que disfrute con la construcción de casas se podrá considerar con el paso del tiempo un maestro de construcción, viéndose esto reflejado en su nivel de habilidad. Y esto le permitirá edificar mejores construcciones. Así, el jugado será el único responsable de sus progresos y de la configuración que finalmente adopte su personaje.

```
>habilidades
Habilidades:

 rastrear   [####                ](20,100)
 esconderse [######              ] 30,100)
 combatir   [##########          ] 50,100)
 cabalgar   [##################  ](90,100)
```

Ejemplo de juego 6: Visualización de habilidades

Finalmente, una vez que el usuario se ha identificado comienza a narrarse un nuevo capítulo de su historia. Este capítulo está formado tanto por los mensajes que envía el jugador como por los mensajes que son recibidos del juego. Al finalizar la sesión el capitulo es almacenado en la base de datos y este puede ser consultado por cualquier usuario mediante el portal web. De esta forma los jugadores van creando una historia en paralelo, pública, que describe desde diferentes puntos de vista el porvenir del mundo.

4.4 Entorno de interacción

El juego y la historia se desarrollan en un mundo habitado por jugadores y personajes controlados por el sistema. Este mundo puede ser modificado y cambiado por las acciones de los usuarios. El cómo se modifica y la forma en la que un usuario puede interactuar con este mundo se le ha denominado *Entorno de interacción*.

En primer lugar hay que destacar las bases que fundamentan las acciones que pueden realizar los usuarios. Cualquier entidad de juego (hereda de la clase *Objeto*) puede ser un entorno de otra. Así, se genera una estructura jerárquica en la que quedan organizados los objetos que han sido instanciados en el juego. Además, cada objeto dispone de un nombre y una descripción. Con estas dos reglas puede construirse cualquier situación de juego. Supongamos, por ejemplo, que un jugador dispone de una flecha dentro de un carcaj de flechas. La organización jerárquica de entornos sería la siguiente:

Ilustración 6: Jerarquía de entornos

Siempre existirá al menos una entidad sin entorno, y normalmente esta entidad será una sala. El siguiente ejemplo es un poco más complejo. Representa una posible situación de una herrería repleta de entidades mediante un esquema jerárquico:

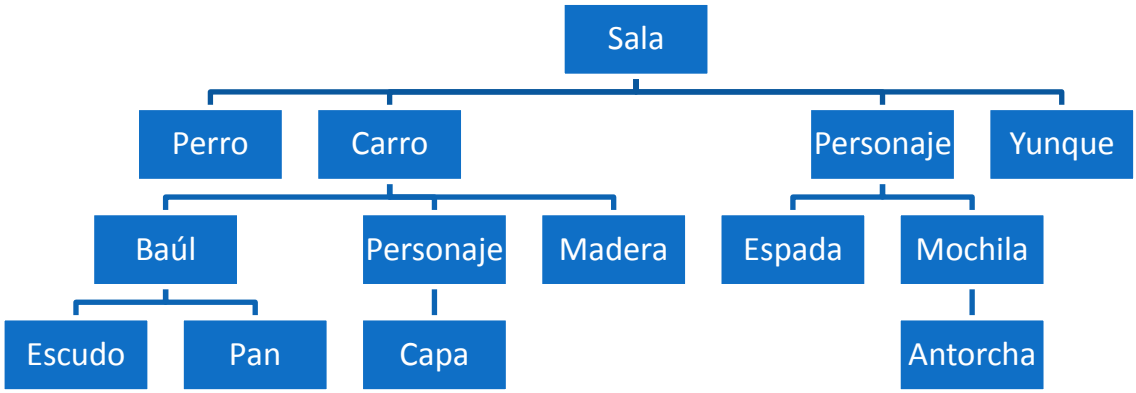


Ilustración 7: Ejemplo de jerarquía de entornos

El mundo, representado de esta forma, cambia cuando esta organización jerárquica es modificada por los jugadores, por los personajes controlados por el sistema o simplemente por eventos del juego. La asignación de entornos, sin embargo, tiene varias restricciones. Estas restricciones se han establecido para asegurar el realismo del juego y mejorar la jugabilidad. Así pues, una *sala* nunca podrá ser entorno de otra *sala*, o un *pan* nunca podrá ser el entorno de un jugador. Establecidas estas reglas, el jugador dispone de los comandos y habilidades descritos en el apartado Lógica del juego para interactuar con el mundo; cogiendo objetos, moviéndose de sala o depositando un material dentro de un contenedor.

Pero además, los jugadores tienen la posibilidad de eliminar, crear o modificar entidades para formar otras. Un jugador, por ejemplo, que disponga de talento suficiente en la habilidad *carpintería*, podrá utilizar la madera parar crear un arco, y de esta forma elimina la entidad *madera* para engendrar un nuevo objeto. De esta forma podrá también edificar construcciones, permitiendo así que los jugadores formen pequeños poblados o ciudades. La necesidad de materiales para la construcción implica que algún jugador deberá extraer primero los materiales del entorno, y por tanto se verán implicadas varias habilidades en la consecución de una construcción. Esta forma de interacción con el entorno propicia la creación de comunidades,

como será visto en el apartado Juego social; pero además dará pie al establecimiento de una economía interna dentro del juego, como será descrito en el apartado La economía.

Finalmente hay que destacar que el entorno habitual de los personajes (entidades controladas por los jugadores) serán las salas. Existen dos tipos de salas: salas exteriores y salas artesanales. Los tipos de salas y sus diferencias serán descritos en el apartado El mundo. Los jugadores podrán moverse de sala utilizando comandos o habilidades. Para determinar la forma en la que se mueven y la dirección (la sala de destino), las salas incorporan el concepto de *salida*, elemento solo disponible para este tipo de entidad. Las salidas permiten conectar salas de forma bidireccional. Así, si un bosque tiene una salida en dirección este a una sala de tipo montaña, la sala montañosa tendrá una salida en dirección oeste que referencie al bosque. De esta forma el jugador puede moverse por el mundo con la posibilidad de volver siempre sobre sus pasos y encontrar el mismo camino entre dos puntos. Por último, el tipo de salida implicará una restricción en los enfriamientos de habilidad, con el objetivo de representar la posible distancia entre dos salas. El siguiente ejemplo muestra el resultado de moverse en dirección este y volver en dirección oeste:

```
>e
Pradera 218,193 Salidas: todas
Vaca está aquí.
>o
Estepa 217,193 Salidas: todas
Caballo está aquí.
```

Ejemplo de juego 7: Movimiento mediante salidas

4.5 El mundo

Como se ha visto en el apartado anterior, el mundo está formado en su mayoría por salas rellenas de contenido. Las salas se conectan mediante salidas, formando redes o mallas que los jugadores pueden imaginar y formar representaciones del entorno que les rodea.

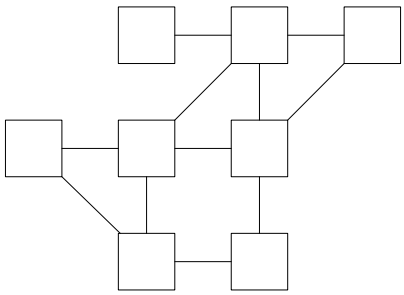


Ilustración 8: Red habitual de salas

La organización de las salas debe ser lógica y debe respetar en cada momento las restricciones espaciales del mundo que representa. No sería correcto, por ejemplo, encontrar un conjunto de salas que siguiese está distribución de salidas:

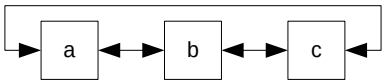


Ilustración 9: Distribución incorrecta de salas-salidas

Sin embargo, esta medida ha sido muy difícil de cumplir en los grandes MUDs actuales. Por la forma en la que se diseña el mundo y su forma de implementarlo (muchas veces de forma iterativa, por zonas), es fácil encontrarse situaciones en las que no se cumple la lógica espacial. Es habitual, por ejemplo, encontrar tres zonas diferentes del mundo conectadas de forma incorrecta. El siguiente ejemplo es una prueba de ello:

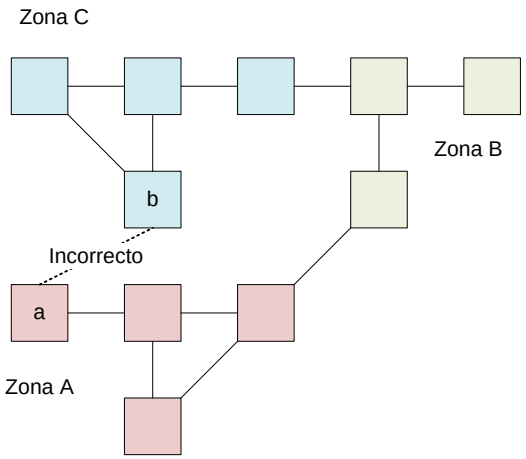


Ilustración 10: Error en la unión de zonas

El diagrama anterior muestra un problema de conexión entre la sala *a* y la sala *b*. Las salidas entre las salas se han establecido como la combinación *norte-sur*, pero sin embargo la representación espacial del entorno demuestra que la combinación correcta sería *noreste-sudoeste*. En algunos casos, algunos juegos justifican estos fallos argumentando que la distancia entre las salidas o el tamaño de las salas no es siempre el mismo. Para algunos jugadores esto no supone un problema pues su representación mental del entorno se amolda sin dificultad a este tipo de imperfecciones de carácter geométrico. Sin embargo, el jugador que utiliza herramientas de representación gráfica del mapa para orientarse encuentra graves problemas que no pueden solventarse con facilidad.

Ya que estos problemas no deben encontrarse en un juego de renombre, es conveniente analizar la metodología de construcción clásica de mundos utilizada por los MUDs actuales y proponer un cambio que permita solventar este y otros problemas que se verán a continuación.

En primer lugar, las características principales de la metodología de construcción de este tipo de juegos son:

- Desarrollo iterativo por zonas: el mundo se desarrolla por etapas. La expansión del mundo no es constante y está determinada por el ritmo de trabajo del equipo de desarrollo. El beneficio de este método es poder ofrecer a la comunidad segmentos del juego (zonas) sin que el mundo haya sido desarrollado por completo. Además, la incorporación de nuevos contenidos al juego es un aliciente que atrae a los jugadores. Sin embargo, esto provoca problemas de descoordinación y desactualización. Crear zonas en periodos diferentes de tiempo puede producir que las políticas de juego hayan cambiado y algunas zonas queden obsoletas en cuanto a contenidos o lógica de juego. Además, en muchas ocasiones las zonas quedan unidas por sólo una o dos salas, lo cual provoca una generación de un mundo basado en *islas* o zonas aisladas.
- Un archivo por sala: en la mayoría de los MUDs se opta por crear las salas mediante una base de datos formada por archivos. De esta forma, cada sala es representada por un archivo en el que se indican sus propiedades y sus salidas. El siguiente ejemplo muestra un posible archivo que representa la plaza mayor de un pueblo:

```
Archivo: plaza_mayor.c
Nombre: Plaza Mayor
Luz: 20
Salidas:
        norte calle_1.c
        sur calle_2.c
        este calle_7.c
        oeste calle_6.c
```

Código 2: Definición manual de una sala

  Esto provoca problemas de actualización de la base de datos de salas. Modificar una zona implica modificar todos los archivos de dicha zona, lo cual no suele ser una tarea fácil para los desarrolladores.
- Creación manual de las salas: la mayoría de equipos de desarrollo están exentos de aplicaciones que faciliten o automaticen la generación de los archivos que definen las salas. Además, las políticas internas de desarrollo implican un nivel de personalización de las salas que no facilita la automatización del proceso. En muchos casos se pretenden que cada sala sea única en cuanto a contenido, y esto implica que sea una persona humana la que configure manualmente las características de la sala mediante la edición del archivo asociado. Este hecho involucra dos problemas: a) se producen gran cantidad de fallos humanos en la edición de las salas y b) el esfuerzo de creación es enorme, lo que restringe el tamaño final de mundo desarrollado.

El modo de creación clásico aflora otros problemas que no se han analizado anteriormente. Con el objetivo de estudiar una posible solución global se enumeran a continuación los problemas encontrados:

- Problemas de incoherencia entre zonas del juego: algunas zonas del juego son desarrolladas en periodos de tiempo diferentes, en los que las políticas de juego han cambiado y estos efectos no se ven reflejados por igual en todo el mundo.
- Mundo basado en islas: el desarrollo iterativo favorece la creación de zonas aisladas del mundo, lo cual resta credibilidad y jugabilidad.
- Grandes recursos para su expansión: la creación manual de las salas mediante archivos de texto es una tarea que requiere grandes recursos.
- Dificultad a la hora de actualizar contenidos: la base de datos de salas organizada mediante ficheros dificulta enormemente la actualización de los contenidos.
- Errores humanos: la creación manual de salas genera errores humanos que son difíciles de controlar.

Por estos problemas surge la idea de automatizar la creación del mundo y la gestión de sus contenidos. ¿Por qué no desarrollar una aplicación que permita la generación de una base de datos de salas diferente? Y así nace la idea de *1pixel-1sala*. Esta idea consiste en utilizar una imagen bidimensional en la que cada pixel de la imagen representa una sala. Así, por ejemplo, con sólo una imagen de 500 píxeles de ancho y 500 píxeles de ancho podemos generar un mundo de 250000 salas. El color de cada pixel nos puede indicar las características de la sala. Por ejemplo, una tonalidad verde podría indicar al sistema que se trata de una sala de tipo *bosque*. Podríamos de esta manera, dibujar un mapa real del mundo mediante una imagen y preparar al juego para que generase las salas pertinentes a partir de esa imagen o mapa del mundo. Con la definición suficiente de colores podríamos además tener un abanico enorme de salas diferentes.

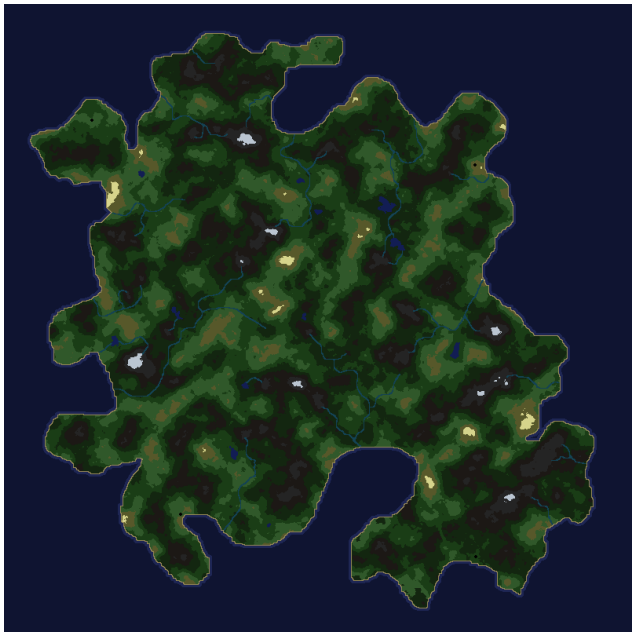


Ilustración 11: Posible imagen geográfica

Esta idea novedosa tiene numerosos beneficios en comparación con el método clásico de construcción del mundo. A continuación se explican los más importantes:

- El mundo se convierte en un continente denso: ya no existen islas o zonas separadas, todo el mundo es explorable y todos los movimientos están permitidos. La sensación de libertad es mayor.
- Los accidentes geográficos son reales: si definimos que algunos tipos de sala no son transitables entonces conseguimos un mundo donde los usuarios deben solventar obstáculos reales. Los pasos montañosos ya no serán una mera hilera de salas conectadas, sino un camino obligatorio a seguir entre dos puntos de una cordillera. Además, las fronteras políticas cobran un mayor sentido.
- El mapa es intuitivo: la creación del mundo basado en una imagen posibilita la opción de que los propios jugadores puedan visualizar dicha imagen como una representación fidedigna del mundo.
- Un mundo equilibrado: si la generación ya no se hace de forma manual, bajo el criterio de un humano, y todo el mundo es creado mediante un mismo generador de salas, con las mismas reglas, es imposible crear desequilibrios entre zonas del mundo.
- Un sistema basado en coordenadas: actualmente las salas sólo se identificaban por su nombre, ahora cada sala puede ser representada por un par de números. Esto permite la creación de sistemas que utilicen dichas coordenadas, como sistemas de rutas, de trayectorias,…
- El mantenimiento es instantáneo: en esta metodología se aprovecha la orientación a objetos para instanciar miles y miles de salas de una misma clase. Por tanto, en este caso solo hay que modificar un solo archivo para realizar un cambio en todas las salas del mundo.
- La generación de la imagen es sencilla: para crear la imagen sobre la que se generará el mundo nos basta con utilizar cualquier editor de imágenes.

- Los recursos de creación del mundo son mínimos: teniendo en cuenta la capacidad de recursos de un estudiante, podríamos afirmar que este método es infinitamente más eficiente a la hora de generar salas que el método tradicional de creación manual de archivos.
- El gasto de memoria es despreciable: en comparación con el método clásico de generación de salas, donde cada sala estaba definida por un archivo en disco, en este caso una sola imagen es suficiente para la generación de todo el vasto mundo. La comparación es enormemente favorecedora para esta metodología.

Sin embargo, no todas las características de esta metodología parecen ser favorecedoras. En primer lugar, el mapa parece estar ahora vacío de contenido. Si un simple color define el tipo de sala a generar, es fácil percatarse de que pocos serán las características diferentes que se pueden generar. Es cierto que, si utilizásemos un modelo simple de color como *RGB*, el número posible de colores asciende a casi 17 millones de colores, y por tanto tendríamos casi 17 millones de salas posibles para distinguir. Sin embargo, esta alternativa es inabarcable, pues cuanto mayor sea el número de colores que utilicemos para dibujar el mapa mayor será la dificultad de su creación y mantenimiento; y por tanto podríamos volver a los problemas de la generación manual de salas. Es por tanto necesario alcanzar un compromiso entre la personalización de la sala y el uso de los colores. Para solucionar este problema se ha utilizado un sistema basado en capas, el cual será descrito en el siguiente apartado denominado El sistema SIG.

Por otro lado, la relación *número de salas/jugador* es una proporción que nunca ha sido llevada a dicho extremo. Nos encontramos ante una situación en la que, suponiendo una cuota de 100 usuarios conectados simultáneamente de media, si dividiésemos el mundo entre los jugadores, cada jugador tendría 2500 salas; casi el número de salas creadas para un MUD, normalmente. Esto se traduce en que será muy complicado la concurrencia de dos jugadores en una misma sala. Este factor, sin embargo, se ve reducido en primer lugar por la gran cantidad de zonas inaccesibles del mundo para los jugadores, como salas anegadas por agua (ríos, lagos, mares,...) o cumbres escarpadas. Por otro lado, la división del mundo en zonas con dificultades dispares hará que los jugadores se concentren en zonas de dificultad baja y zonas de máximo nivel. Finalmente, la incorporación de ciudades hará que estas sirvan como punto de encuentro entre jugadores.

La siguiente tabla muestra un resumen de las ventajas e inconvenientes de ambas metodologías:

Tabla 2: Ventajas e inconvenientes de las metodologías de desarrollo

	Generación clásica basada en archivos	Generación basada en mapas densos
Ventajas	• Alto nivel de detalle • Control total sobre las salas • No hay dos salas iguales	• Fácil orientación • Mundo equilibrado • Coordenadas • Recursos de creación mínimos • Actualización centralizada • Requisitos del sistema menores
Inconvenientes	• Costoso de desarrollar • Difícil de actualizar • Desequilibrios entre zonas • Errores humanos • Mundo de islas	• Mapas vacios de contenido • Demasiadas salas por jugador

4.5.1 Las ciudades

En el apartado anterior se ha descrito cómo utilizar una nueva metodología de desarrollo para generar un enorme mundo de forma dinámica partiendo de una sola imagen. Al finalizar el apartado se ha descrito la posibilidad de incorporar *ciudades* para concentrar allí a los jugadores. El objetivo es crear puntos estratégicos donde los jugadores puedan reunirse y encontrar servicios.

Antes de describir la metodología implementada para desarrollar las ciudades conviene analizar cuáles son las principales características que definen una ciudad respecto del mundo exterior.

- Forma regular: a diferencia del vasto mundo exterior, una ciudad se caracteriza por estar construida siguiendo un patrón más o menos regular, con numerosos obstáculos (construcciones) que limitan el movimiento. Las calles forman retículas, mallas, anillos,… formas que distan de la densidad *8 vecinos* del mapa denso.
- Gran nivel de detalle: ya que se trata de una zona muy transitada, se espera que su nivel de detalle y ambientación sea superior al resto. En una ciudad pueden encontrarse estatuas, guardias, pasacalles,….
- Alto grado de personalización: frente a una pradera o un bosque, en una ciudad pocas calles son iguales. Por este motivo casi todas las salas deben parecer únicas.
- Salas de servicios: la ciudad es el único lugar en el que un jugador puede encontrar salas destinadas a dar servicios. Como por ejemplo bancos, caballerizas o herrerías.

Partiendo de estas características, la primera aproximación que se estudió fue utilizar el mismo sistema de generación del mundo para la incorporación de ciudades. Esto significaba, por ejemplo, utilizar la escala del color amarillo para definir sobre la imagen del mundo una ciudad al completo. La siguiente imagen muestra una ciudad implementada de esta forma como prueba:

Ilustración 12: Ciudad generada sobre la capa geográfica

Sin embargo este método tiene varios inconvenientes. El primero de ellos es que la ciudad ocupa un lugar demasiado grande en el mundo. En un mundo donde las salas representan grandes extensiones de terreno, las ciudades parecen ser desproporcionadas por su enorme tamaño. Por otro lado, la necesidad de utilizar gran cantidad de colores diferentes reduce los beneficios de esta metodología. Además, el método debe poder ser extensible a otros tipos de zonas, como cuevas, puentes, fortalezas,…. Y por tanto el número de salas diferentes se eleva a más de 255 salas. Además, una fortaleza o una simple torre están formadas por varios niveles. Mediante el sistema de mapas densos es imposible definir una edificación con varias alturas sobre un mismo pixel. Los mapas densos proporcionan un conjunto de salas distribuidas en un plano bidimensional. Finalmente, la imposibilidad de realizar cambios personalizados en salas concretas de la ciudad empobrece su ambientación.

Por este motivo se ha optado finalmente por utilizar la antigua metodología de desarrollo basada en ficheros para la generación de estas salas. Pero para ello se ha decidido incorporar algunos cambios en la metodología que permitan solventar los problemas existentes y adaptarla a las nuevas tecnologías. A partir de este punto se hace patente la necesidad de diferenciar dos tipos de salas: las salas exteriores y las salas artesanales. Las salas exteriores son aquellas salas generadas dinámicamente a partir de una imagen, mientras que las salas artesanales son aquellas salas que se generan a partir de un fichero, y que han sido creadas utilizando la metodología clásica.

A continuación se describen los principales cambios realizados en la metodología clásica para la generación de zonas.

4.5.1.1 XML como lenguaje

El principal cambio en la metodología es el uso del lenguaje XML. Este lenguaje ha demostrado ser un excelente medio de almacenamiento de información organizada. Su estructura jerárquica permite la definición de manera inequívoca cualquier situación, tal y como se ha descrito en el apartado

Entorno de interacción. Además, la gran mayoría de lenguajes de programación emergentes incorporan muy buenas herramientas para la lectura y escritura de estos tipos de archivos de texto. El siguiente fragmento corresponde a un segmento de un archivo XML correspondiente a una zona del juego:

```
<?xml version="1.0" encoding="utf-8"?>
<zona id="torre_amarilla" nombre="Torre de Vigilancia">
  <conexiones>
    <conexion x="217" y="192" id="id0" />
  </conexiones>
  <salas>
    <sala id="id0">
      <nombre>Sendero hacia la torre</nombre>
```

Código 3: XML de una zona

Como puede verse, sólo será necesario generar un fichero por zona. En lugar de crear un archivo por sala, solo será necesario escribir un archivo para definir por completo toda una zona. De esta forma las zonas quedan más controladas y sus variables pueden ser editadas desde un mismo archivo.

El juego es capaz de reconocer este tipo de archivos y generar dinámicamente las salas de una zona en función de las características que se indique en dicho archivo. Para que el juego reconozca estos archivos se ha incorporado una carpeta denominada *Zonas*. En dicha carpeta el sistema busca aquellos ficheros XML que cumplen la estructura de zona y los carga en memoria en forma de salas. Si un desarrollador desea incorporar una nueva zona al juego, basta con que éste incorpore el archivo de la zona a la carpeta correspondiente y la zona quedará automáticamente incorporada al mundo.

4.5.1.2 La aplicación generadora

El segundo cambio realizado es la utilización de una aplicación para automatizar la generación del archivo XML. El objetivo consiste en permitir al usuario describir de una forma visual e interactiva una zona para más tarde generar a partir de esa información un archivo XML completo.

Para ello se ha desarrollado una aplicación de escritorio denominada *Editor Zonas Artesanales* que permite mediante el uso del ratón y el teclado generar mapas bidimensionales y transformarlos en documentos XML legibles por el juego.

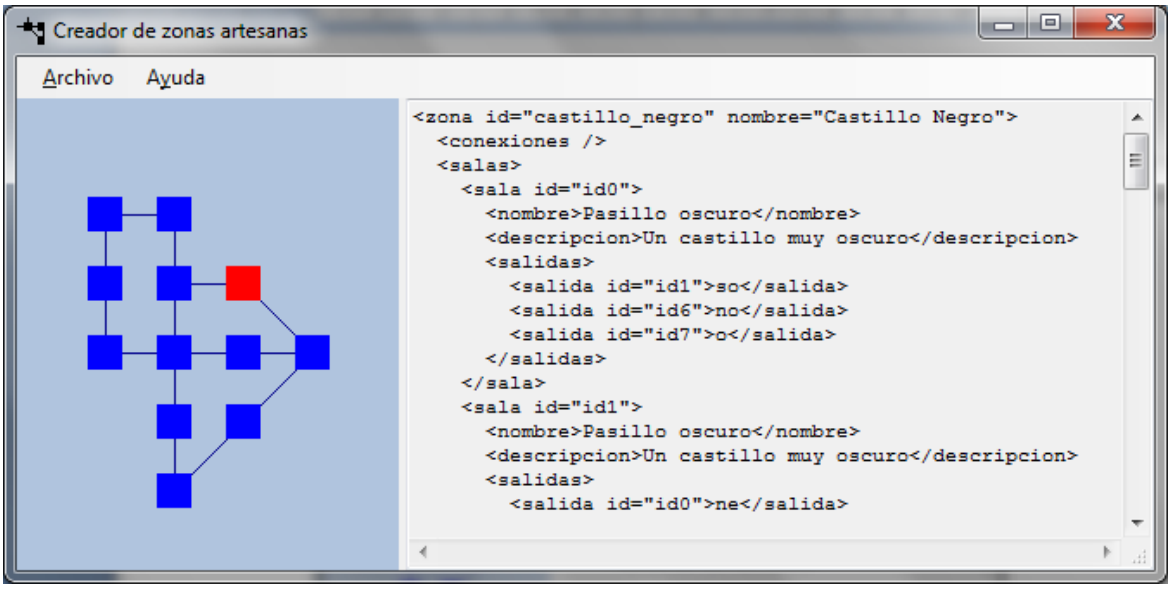


Ilustración 13: Aplicación de generación de zonas

La captura anterior muestra una captura de pantalla de la aplicación. Como puede verse, la aplicación se divide en dos secciones: la parte visual y editable, y la parte correspondiente al archivo XML de salida (no editable). El usuario puede utilizar el teclado numérico para editar el mapa bidimensional y más tarde guardar dicho mapa en un archivo de formato XML.

4.5.1.3 La unión con el mundo exterior

Uno de los problemas que surgen al utilizar las dos metodologías de desarrollo al mismo tiempo consiste en unir los dos tipos de salas. Es decir, permitir que desde el mundo exterior se puedan acceder a zonas artesanales y viceversa. Así, un usuario podría caminar por una montaña y encontrar una entrada especial a una gruta. O un jugador podría utilizar una zona artesanal para atravesar un rio.

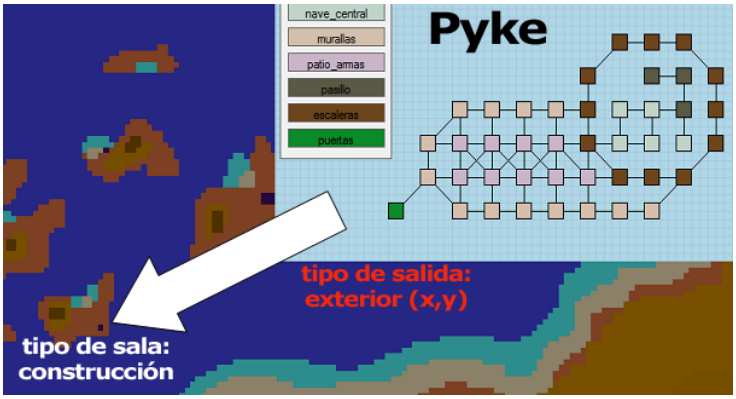


Ilustración 14: Conexiones entre zonas artesanales y el mundo exterior

Una primera aproximación consintió en utilizar una nueva capa del sistema SIG (descrito en el apartado El sistema SIG) para indicar en qué puntos del mundo exterior se incluían entradas a zonas artesanales. Sin embargo, esta solución no era la más adecuada pues utilizaba grandes recursos estáticos (una imagen) para una cantidad de información mínima y cambiante. Si la incorporación de zonas al juego va a ser constante, debería existir un método de unión que permitiese adaptarse fácilmente a esas nuevas incorporaciones.

Por ese motivo se decidió que las uniones entre las zonas artesanales y el mundo exterior se definiesen en las propias zonas artesanales. Así, cuando una zona es incorporada al sistema mediante su copia en la carpeta *Zonas*, el sistema analiza las entradas que se han definido en el archivo XML y las refleja en el mundo exterior. De esta forma el uso de recursos es mínimo y el impacto producido por la incorporación, modificación o eliminación de zonas se reduce enormemente. El siguiente ejemplo muestra el segmento de un archivo XML de una zona que representa un puente:

```
<conexiones>
  <conexion x="219" y="201" id="id0" />
  <conexion x="222" y="201" id="id3" />
</conexiones>
```

Código 4: Conexiones de una zona

4.5.2 Rellenando el vacio

Unos de los problemas vistos en este apartado es la problemática de generar un mundo gigante repleto de contenidos diferentes. Suponiendo una sola imagen como base de datos sobre las salas del mundo exterior, sólo tenemos información referida al color de un pixel (el tipo de sala) y sus coordenadas. Con esa información difícilmente podemos rellenar el mundo de contenidos de una forma interesante y atractiva. No podemos, por ejemplo, aplicar un buen sistema climático, realizar una segmentación del territorio por fronteras políticas irregulares o

hacer crecer un determinado árbol en una zona. Esto produce en el usuario la sensación de estar explorando un mundo vacio de contenidos, que no incita a la exploración.

Para solventar este problema se propone una solución basada en dos pilares:

- Incorporar más variables por sala: el objetivo es conseguir más parámetros sobre una misma sala, y no solo su tipo. De esta forma la probabilidad de encontrar dos salas con las mismas características es mucho menor. Para incorporar más información sobre las salas sin incumplir el compromiso en el uso de colores se propone el Sistema de Información Geográfica que será analizado en el siguiente apartado.
- Crear generadores de contenido: suponiendo que las salas tienen varios parámetros, podemos crear generadores de contenidos que, utilizando filtros, sean capaces dar una rico ambientación las salas del mundo exterior.

4.5.2.1 El sistema SIG

La primera medida que se propone como solución al problema de ambientar el mundo exterior es el Sistema de Información Geográfica (SIG o GIS, en su acrónimo inglés [Geographic Information System]). El objetivo es enriquecer el mundo mediante la incorporación de más parámetros a las salas utilizando la misma metodología propuesta para la creación del mundo vista al principio de este apartado.

Hasta ahora se había comentado la idea de utilizar una única imagen para generar a partir de ella todas las salas del mundo exterior. Sin embargo, esta imagen solo nos permite otorgar el valor a un parámetro de la sala. El sistema SIG consiste en utilizar varias imágenes en paralelo en lugar de una sola. De esta forma, dado un par de coordenadas, no se obtiene un único valor de color (el de la imagen inicial), sino que se obtienen tantos valores de color como imágenes auxiliares se quieran incorporar al sistema. Cada imagen otorgará el valor a un parámetro de la sala, en función del color del pixeles de la coordenada seleccionada. Por tanto, existirá una imagen por cada uno de los parámetros que se configuren en las salas. La base de datos de las salas pasa ahora a estar formada por un conjunto de imágenes de las mismas dimensiones junto a los valores asociados a ellas.

Con el objetivo de ampliar la información disponible en las salas se ha decidido añadir varios parámetros auxiliares a la información existente; que hasta ahora sólo estaba formada por las coordenadas de la sala y el tipo de sala. Los parámetros actuales que definen una sala son:

- Coordenadas: cada sala exterior tiene un par de coordenadas que la identifican de manera única.
- Tipo de sala: representa el bioma del entorno. Por ejemplo: bosque, pradera, océano,…
- Dificultad: indica en nivel de dificultad del entorno, y por tanto de todos los contenidos que se generen en su interior.
- Altura: representa la altura respecto al nivel del mar de la sala.
- Dominio: simboliza a qué facción pertenece el territorio.
- Clima: establece el clima de la zona.

Por tanto, para establecer el valor de cada uno de estos parámetros se ha creado una imagen del tamaño del continente. A la combinación de una imagen junto a sus valores asociados se le ha denominado *capa*. A continuación se muestran las capas creadas:

4.5.2.1.1 Capa geográfica

La capa geográfica es la capa que permite designar el bioma de la sala generada. Un mundo completo dispone de gran cantidad de biomas diferentes. Sin embargo, por motivos de jugabilidad se ha definido acotar el número de biomas a 15. A continuación se adjunta la tabla de información que define los diferentes biomas que se pueden encontrar en el juego:

Tabla 3: Biomas

Id.	Nombre	Descriptor	Rojo	Transitable
1	Oceano	el oceano	15	False
2	Costa	la costa	30	True
3	Playa	una playa	150	True
4	Desierto	un desolado desierto	214	True
5	Estepa	una extensa estepa	87	True
6	Pradera	una verde pradera	48	True
8	Bosque	un bosque	26	True
9	BosqueDenso	un denso bosque	19	True
10	Montaña	una montaña	28	True
11	Escarpado	una zona escarpada	36	False
12	Cumbre	la cumbre de una montaña	186	False
13	Rio	un rio	20	False
14	Lago	un lago	18	False
15	Construccion	una construcción	0	False

La primera columna de la tabla indica el identificador del tipo del bioma. En segundo lugar se muestra el nombre real del bioma. El tercer parámetro es el descriptor que puede utilizarse para narrar información sobre la sala. La columna *Rojo* indica el valor del color rojo que debe tener un pixel determinado de la imagen asociada a la capa geográfica para que el bioma de la sala corresponda a este tipo. Finalmente la última columna indica si la sala puede ser transitada por jugadores o se presenta como un obstáculo para la movilidad.

A continuación se muestra la imagen generada mediante un programa de edición de imágenes que está asociada a esta capa:

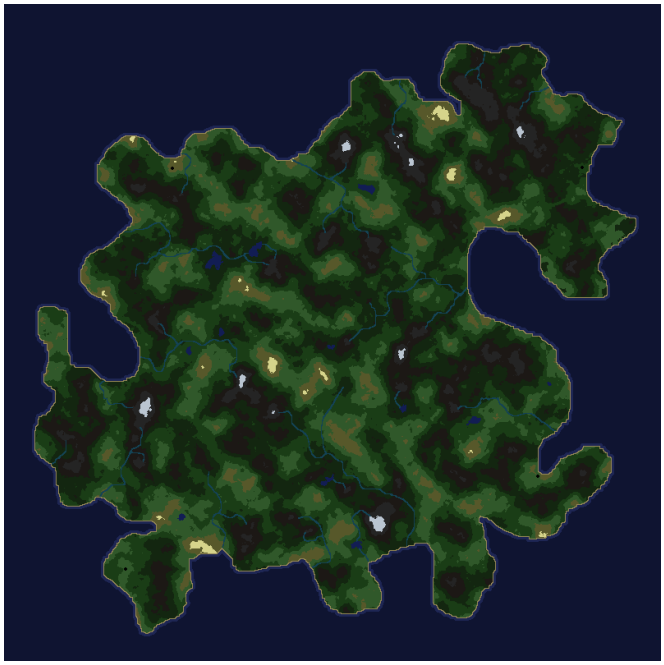


Ilustración 15 : Imagen de la capa geográfica

El mundo está formado por un único continente rodeado por agua. El océano, al no ser transitable, establece las fronteras reales del mundo conocido. A simple vista pueden visualizarse cuatro puntos negros en los extremos del continente. Estos puntos son salas de tipo construcción y representan el lugar que ocupan las cuatro capitales (una por cada facción).

4.5.2.1.2 Capa de niveles

Poder dividir el mundo en zonas con niveles de dificultad favorece que el jugador visite diferentes zonas del continente a lo largo del progreso de su personaje, con los grandes beneficios que esto conlleva. Establecer el nivel de una sala es por tanto un parámetro fundamental. De la manera en la que estos valores se asignen dependerá en gran medida el equilibrio del juego y, por tanto, su atractivo.

La idea consiste en establecer valores de 0 a 12. El nivel 0 corresponden a zonas en las que no se requiere experiencia de juego ni progreso del personaje para poder superar los contenidos que se encuentran. El nivel 10 es la dificultad máxima que un jugador puede superar con sus propios medios. Las zonas de nivel 11 o superior se reservan a contenidos que deben ser superados con la colaboración de varios personajes de nivel máximo.

Para ello se ha decidido optar por lo siguiente. Suponiendo que las ciudades es el punto de encuentro de los jugadores de un mismo bando y los jugadores novatos tienen personajes con gran dependencia hacia estas ciudades, ¿por qué no establecer el nivel de una zona en función de la distancia a una ciudad? De esta forma, el usuario avanzado, con un personaje con grandes recursos, puede viajar lejos de su ciudad en busca de contenidos difíciles y complejos. Además, al alejarse de su ciudad aumentará la probabilidad de encontrar jugadores de otra facción enemiga, lo que incrementará aun más la dificultad. Mientras, los usuarios

novatos pueden aprovecharse de la cercanía de las ciudades (protección, servicios) para mejorar su experiencia de aprendizaje.

La siguiente imagen corresponde a la capa de niveles. Puede observarse como se han utilizado 11 niveles de gris diferentes para representar la dificultad de las salas. El color más oscuro representa las zonas de menor dificultad, mientras que el color más claro las salas con dificultad mayor. El centro de los círculos concéntricos corresponde a la posición de las ciudades, tal y como se ha podido ver en la imagen geográfica. El centro del continente se convierte, lógicamente, en la zona más difícil del juego al estar muy alejada de todas las ciudades y ser un punto equidistante respecto a las cuatro facciones del juego.

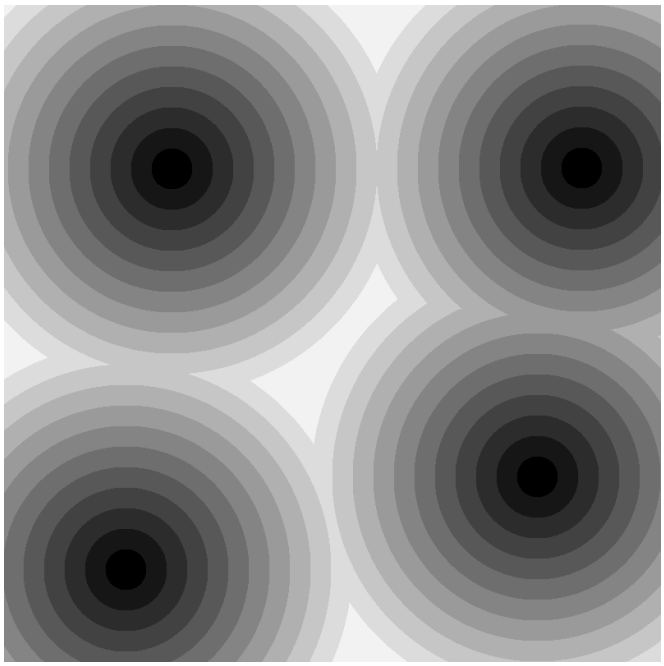


Ilustración 16: Imagen de la capa de niveles

La siguiente imagen es un montaje meramente orientativo que permite visualizar la capa de niveles sobre la capa geográfica.

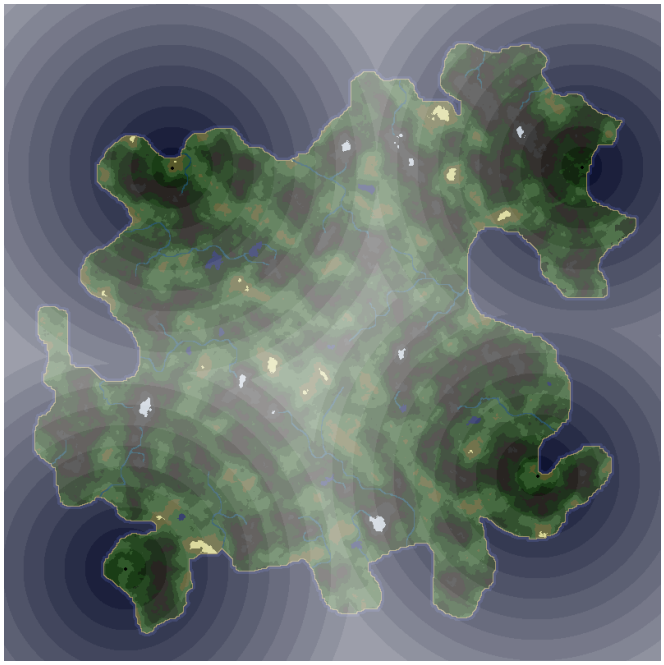


Ilustración 17: Montaje de la imagen geográfica y la de niveles

4.5.2.1.3 Capa de alturas

Poder definir la altura de una sala respecto al nivel del mar es un gran aporte tanto para la ambientación como para la jugabilidad. Por un lado, el bioma definido en una sala puede adoptar tantas formas como alturas se definan. De esta forma, no encontraríamos los mismos tipos de árboles en un bosque a nivel del mar como en un bosque a gran altitud. Por otro lado, conocer la altura de una sala puede permitir crear sistemas muy interesantes de cara a la jugabilidad. Imaginemos poder colocar tesoros solo en aquellos puntos del océano que se encuentren a una determinada profundidad, o provocar avalanchas de nieve que arrasen todo colina abajo. Finalmente, también puede usarse como métodos de filtrado. Así, una facción acostumbrada a habitar cercana al mar, podría tener una restricción que impidiese a los jugadores construir edificaciones más allá de una cierta altura. Por lo contrario, otra facción más afín a las alturas, podría tener una restricción que impidiese construir en alturas inferiores a una dada.

Esta es la tabla almacenada en la base de datos que permite obtener los datos de la sala a partir de la imagen asociada a la capa.

Tabla 4: Alturas

Id	Nombre	Nivel	Rojo	Descriptor
1	Bajo	-1	226	por debajo del nivel del mar
2	NivelDelMar	0	191	despreciable, al nivel del mar
3	Macrotermico	1	153	ligeramente elevada sobre el nivel del mar
4	Mesotermico	2	115	elevada
5	Microtermico	3	77	muy elevada

7	Gelido	4	38	impresionante, desde donde puedes contemplar grandes extensiones de terreno
9	Extremadamente Gelido	5	0	titánica, este lugar parece la cima del mundo

La siguiente imagen corresponde a la capa de alturas. Los colores más oscuros representan las zonas más altas en tierra y más profundas en el océano. Esto es así porque se ha decidido utilizar una misma escala de colores tanto para las alturas negativas como para las positivas, pero siempre teniendo en cuenta que los colores más claros corresponden a una altura al nivel del mar.

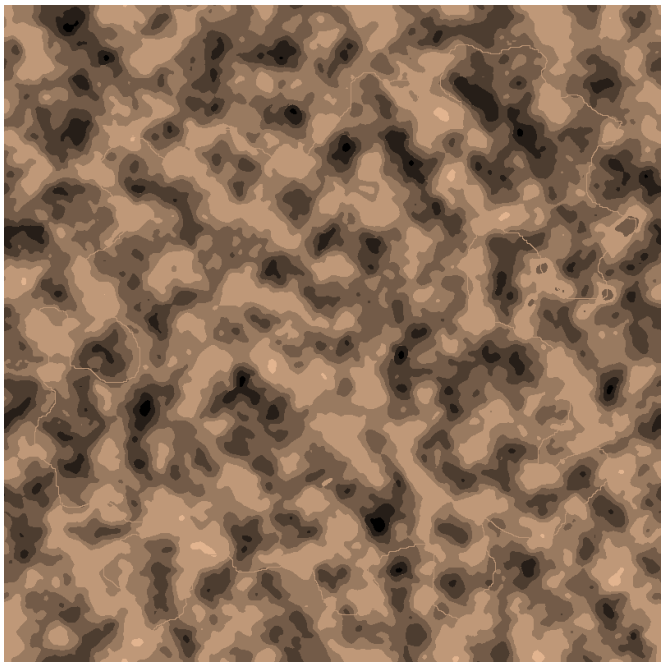


Ilustración 18: Imagn de la capa de alturas

4.5.2.1.4 Capa política

Una de las características más importantes del desarrollo de un personaje es su afiliación con una de las cuatro facciones que se encuentran en el continente. Poder representar la influencia de dicha facciones en las salas exteriores es una idea que conlleva muchísimas ventajas, de nuevo, de cara a la ambientación y a la jugabilidad. En cuanto a la ambientación, saber que una sala pertenece al territorio de una facción permite decorarla con objetos típicos de dicho bando. Así, un camino podría estar empedrado de una forma determinada, o podrían encontrarse cultivos típicos de los habitantes de la facción. En cuanto a la jugabilidad, se puede aprovechar este parámetro para varias alternativas. La primera de ellas es restringir, por ejemplo, el lugar donde los jugadores pueden edificar construcciones a aquellos territorios que pertenezcan a su facción. Por otro lado, favorecer la protección de los jugadores que se encuentren dentro del territorio de su facción, mediante la inserción de PNJ (personajes no

jugadores) que ataquen a personajes enemigos. Finalmente, permitir que exista un canal de comunicación directo entre los jugadores que se encuentran dentro del mismo territorio.

La siguiente tabla muestra la información referida a esta capa. En un futuro será fundamental incorporar nuevos datos a las tablas referidos a su historia así como modificar el nombre de las facciones. Es importante destacar la incorporación de la facción *Neutral*, que se corresponde con aquellos territorios que no pertenecen a ninguna facción.

Tabla 5: Facciones

Id	Nombre	Rojo	Descriptor
1	Verdes	0	los Verdes
2	Azules	1	los Azules
3	Amarillos	253	los Amarillos
4	Rojos	254	los Rojos
5	Neutral	255	neutrales

La siguiente imagen corresponde a la capa política. Por cada facción se ha establecido un color. Una vez más, el color rojo es suficiente para determinar la facción de la sala. Se le ha incorporado un contorno a la imagen parar mejorar su visualización.

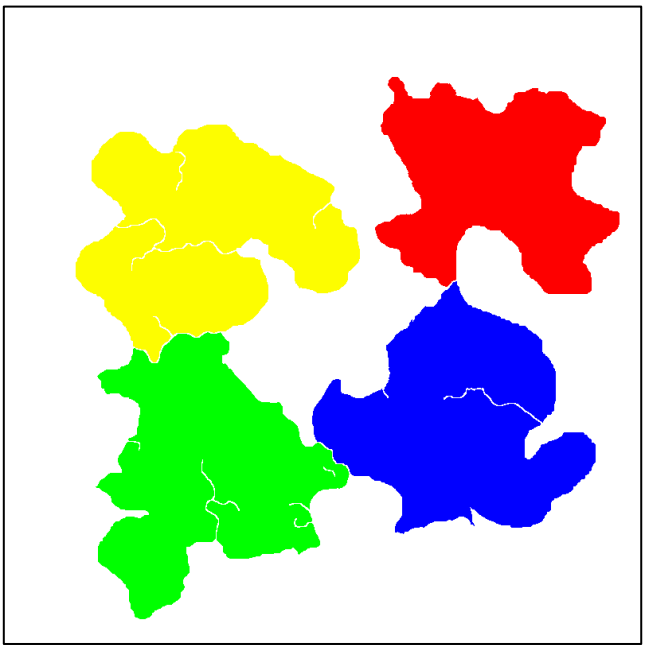


Ilustración 19: Imagen de la capa política

En la imagen anterior resalta la irregularidad de las fronteras, tal y como se han establecido en el mundo real. En la mayoría de los casos su forma viene determinada por la distribución de los

accidentes geográficos descritos en la capa geográfica. Para entender mejor este concepto se presenta a continuación un montaje en el que se superpone la capa política a la imagen geográfica.

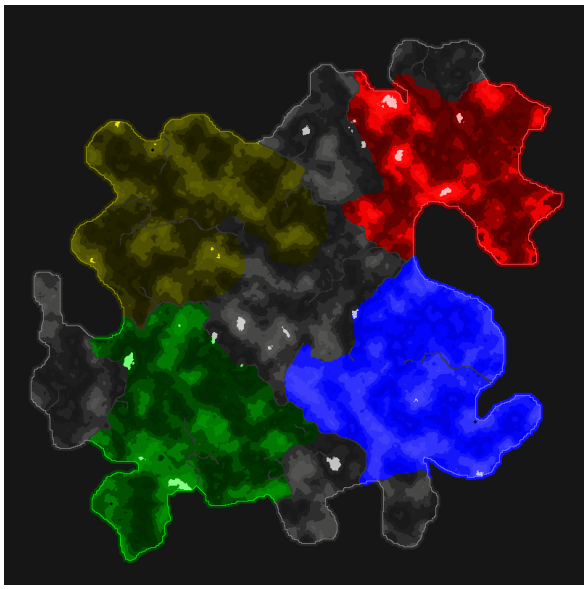


Ilustración 20: Montaje usando la imagen geográfica y la política

4.5.2.1.5 Capa climática

Uno de los efectos más interesantes que se pueden aplicar a las salas son los cambios climáticos. Poder representar una tormenta en alta mar o una ventisca de nieve en la cima de una montaña provoca en el jugador la sensación de estar explorando un mundo vivo y cambiante.

Una primera aproximación consistió en generar una imagen aleatoria mediante un efecto de *nubes* que ofreciese un efecto climático más o menos parecido a una imagen de satélite. El resultado fue muy bueno. A continuación se muestra la imagen utilizada. Los colores claros representan el mal tiempo, mientras que los colores oscuros el buen tiempo.

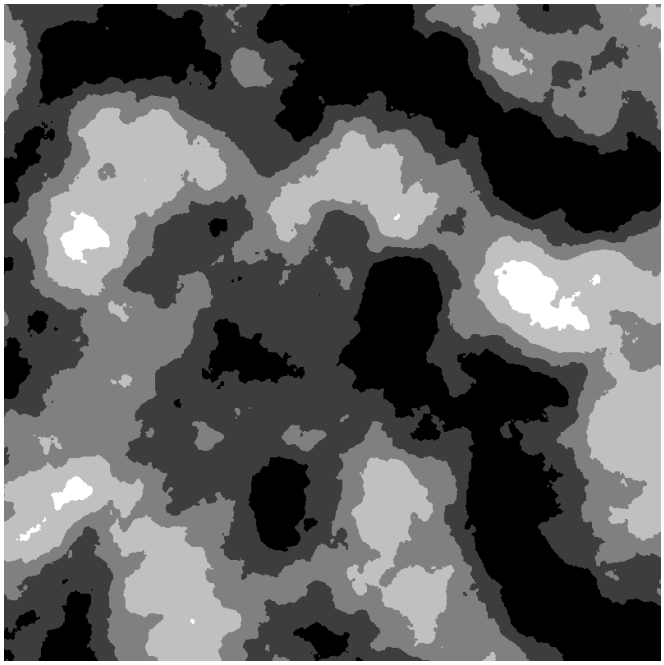


Ilustración 21: Imagen inicial de la capa climática

Sin embargo, esta imagen tiene un problema evidente. El clima cambia. Hasta ahora hemos descrito capas del modelo SIG formadas por imágenes estáticas que no varían en el tiempo. Es lógico, por ejemplo, que un bioma de tipo bosque no cambie a otro diferente. El clima, por lo contrario, debe cambiar, y mediante esta simple aproximación las salas siempre tendrían el mismo clima.

Para solucionar este problema se estudió una segunda alternativa. Se trataba de crear algoritmos con memoria que fuesen capaces de generar imágenes climáticas secuencialmente siguiendo una relación lógica entre las imágenes consecutivas. Sin embargo este método rápidamente fue descartado por la enorme dificultad que suponían crear este tipo de algoritmos.

En tercer lugar se pensó en crear una imagen cilíndrica, es decir, que el borde derecho de la imagen pudiese unirse al borde izquierdo sin perder la lógica del clima. Un vez creada, un algoritmo se encargaría se establecer una máscara de recorte constante sobre la imagen e iría rotando dicha máscara a lo largo de la superficie del cilindro. De esta forma se conseguiría cambiar el clima de la zona, pero se establecerían ciclos en los que siempre se encuentra el mismo clima.

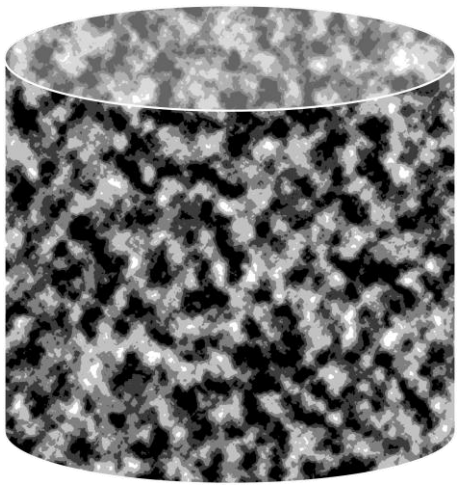


Ilustración 22: Imagen climática cilíndrica

Descartada esta alternativa se pensó en crear una imagen esférica. Es esta forma la máscara de recorte podía no solo trasladarse por la superficie de la esfera, sino que también podía rotar para conseguir que el movimiento del clima no fuese siempre en la misma dirección.

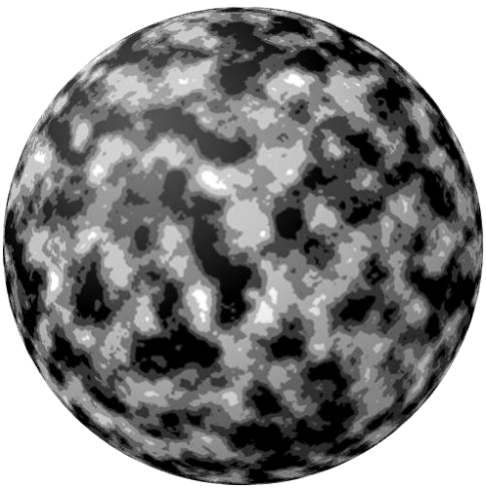


Ilustración 23: Imagen climática esférica

Sin embargo, esta alternativa tiene dos grandes inconvenientes:

- La representación de una imagen esférica en memoria.
- La proyección de una esfera sobre un plano. Si la máscara de recorte es un plano de tamaño del mundo, debemos proyectar la superficie de una esfera sobre un plano, lo cual es un enorme problema de proyección cartográfica cuyas dimensiones no permiten que sea abarcado por este proyecto.

Finalmente la solución elegida ha consistido en utilizar una imagen cuadrada de enormes dimensiones y hacer trasladar y rotar la máscara de recorte sobre ella de manera aleatoria, reduciendo así al mínimo la posibilidad de encontrar repeticiones climáticas perceptibles por los jugadores.

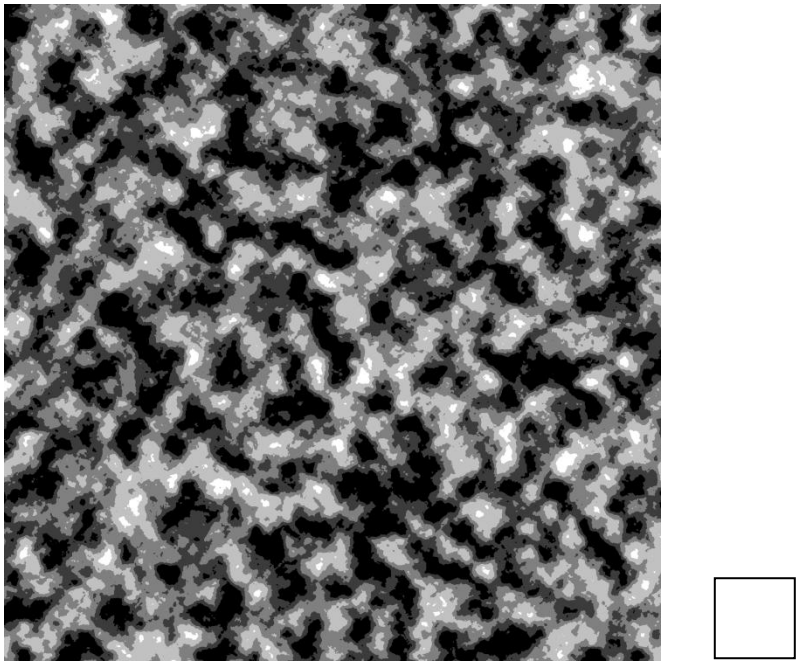


Ilustración 24: Imagen final de la capa climática y su máscara de recorte

De esta forma, para cada instante de tiempo determinado la máscara de recorte nos ofrece una imagen cuadrada de las dimensiones del mundo tal y como necesita la capa climática para poder establecer el valor del clima de las salas. Se ha programado, además, un algoritmo que mueve la máscara de recorte a lo largo de la imagen y que recorre las salas exteriores del mundo cargadas en memoria actualizando el nuevo valor del parámetro climático. Este algoritmo es lanzado cada 15 minutos.

4.5.2.2 Generadores

De manera paralela al sistema de información geográfica por capas se ha creado un conjunto de *generadores* como solución al problema de la ambientación del mundo. El objetivo, una vez más, es otorgar a las salas exteriores del mundo una ambientación muy elaborada y evitar la similitud entre las salas.

En este caso el sistema consiste en utilizar generadores de contenidos especializados que evalúan los parámetros de una sala y en función de estos incorporan unos contenidos u otros. El esquema general del proceso de creación de una sala puede verse en la siguiente figura. El jugador comienza el proceso de creación solicitando una sala al controlador de salas. La sala es cargada en memoria con solo un par de parámetros, sus coordenadas. Cuando la sala accede al modelo SIG, a partir de sus coordenadas se obtiene el valor de los parámetros vistos

anteriormente. Finalmente, en su última etapa los generadores se encargan de dar ambientación a la sala incorporando contenidos.

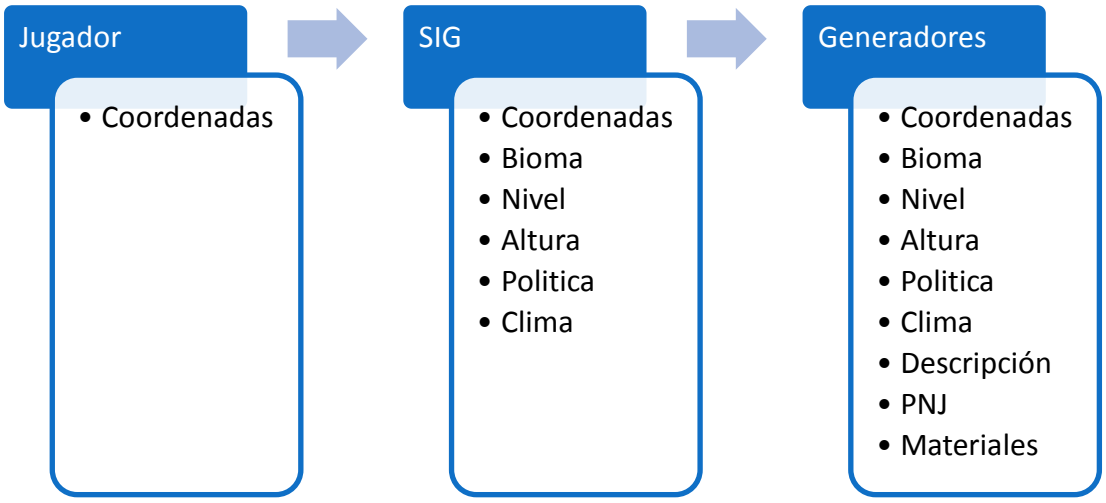


Ilustración 25: Esquema de generación de salas

Los generadores actuales implementados son:

- Generador de descripciones: es el encargado de poner el nombre, la descripción y el resto de información textual propia de la sala. Este generador utiliza cada uno de los parámetros de la sala para realizar una descripción personalizada y única de la sala.
- Generador de PNJs: este generador se encarga de introducir en la sala los PNJs que dan vida al entorno que rodea a los jugadores. Este generador utiliza todos los parámetros de la sala excepto el clima para establecer los seres que se pueden encontrar en las salas. Desde una ardilla en un bosque de al nivel del mar en una zona de poca dificultad, hasta una patrulla de guardias en un camino perteneciente a una determinada facción.
- Generador de materiales: una de las posibilidades de juego más atrayentes para los jugadores es la posibilidad de utilizar materiales extraídos del mundo para la construcción de objetos o edificios. Este generador es el encargado de repartir los materiales de forma equitativa por todo el continente. Para ello este generador también utiliza todos los parámetros de la sala exceptuando el clima. De esta forma, por ejemplo, un jugador novato podrá encontrar minerales de bajo nivel en montañas cerca de su ciudad, mientras que un jugador experimentado podrá conseguir madera rara y de mucho nivel en bosques a gran altura que se encuentren en territorio neutral.

4.6 La economía

Muchos MMORPGs ofrecen economías vivas. Objetos virtuales y monedas se obtienen durante el juego. Además, un mercado basado en la oferta y la demanda ha demostrado ofrecer la mejor alternativa para conseguir crear una base económica estable.

En este sentido, el juego se ha creado para incorporar las características básicas que tantos buenos resultados han dado y que se ofertan con ligeras alteraciones entre unos juegos y otros:

- La posibilidad de conseguir objetos virtuales y monedas mediante el esfuerzo del jugador. Existen muchas formas de conseguir objetos y monedas: extrayendo materiales del entorno mediante habilidades, combatiendo contra PNJs, completando misiones,…
- Permitir el intercambio comercial entre jugadores. Es fundamental que los jugadores tengan una herramienta para intercambiar objetos y monedas de una forma segura y fiable para ambos.
- Favorecer el mercado entre jugadores estableciendo valores de venta muy bajos en tiendas controladas por el juego. Esta medida, además, controla en gran medida el crecimiento de dinero virtual en circulación.

4.7 Juego social

En primer lugar podemos definir un juego social como cualquier tipo de juego que implique la interacción entre dos o más personas. El juego parte de ser un juego en línea donde todos los usuarios comparten un mismo mundo y por tanto es obvio pensar que se trata de un juego social. Sin embargo, es importante resaltar la socialización del juego en cuatro sentidos que se describen a continuación.

El primero de ellos es referido al modelo de accesibilidad definido en el capitulo Modelo de accesibilidad en el que se ha diseñado un juego siguiendo la filosofía de *diseño para todos*. Esta filosofía es de por si social, pues pretende que el sistema pueda ser utilizado por el mayor número posible de personas independientemente de sus necesidades. Al ser un juego en red, sirve como punto de encuentro para todas estas personas y permite que puedan comunicarse mediante muchos de los comandos de comunicación existentes.

Por otro lado, la necesidad de colaborar entre jugadores para superar ciertos contenidos otorga al juego un carácter social colaborativo. En muchas ocasiones, un solo jugador no es capaz de realizar cierta acción o superar cierto reto, y necesita de la ayuda de otro usuario o varios para poder realizarla. Este hecho favorece la cooperación entre usuarios y el nacimiento de lazos de amistad que perduran en el tiempo más allá de las necesidades puntuales.

En la línea anterior también se encuentra el mercadeo. En un mundo donde los jugadores se especializan en realizar determinados tipos de acciones, en muchos casos se requieren

transacciones y acuerdos de negocio entre compradores y vendedores. Esto favorece también el establecimiento de relaciones afectivas entre usuarios del juego. Al igual que en la vida real, los trueques e intercambios económicos satisfactorios producen relaciones amistosas entre las partes implicadas.

Por último, un sistema de hermandades favorece el nacimiento de pequeñas comunidades organizadas. Una hermandad es un conjunto de jugadores que se unen para crear un ambiente de colaboración colectiva. Existen varios tipos de hermandades. Algunas están relacionadas con aprender y enseñar a nuevos jugadores. Otras hermandades se crean con el objetivo de cumplir logros y ganar renombre entre la comunidad. Otras, por otro lado, buscan en las hermandades una oportunidad para mejora su economía virtual y conseguir mejores oportunidades de mercado.

5 Juego multiterminal

Uno de los pilares fundamentales del proyecto y que se ajusta a la filosofía de *diseño para todos* vista en el capitulo Modelo de accesibilidad corresponde a lograr un juego multiterminal. La palabra “multiterminal” pretende abarcar el concepto de que el juego pueda ser utilizado desde el mayor número posible de dispositivos, independientemente de su hardware y su software.

Es importante destacar que la palabra “multiplataforma” no se ha utilizado para definir el sistema pues el software programado no cumple estas características. Tanto en el lado del servidor como en el lado de los clientes, se han utilizado diferentes lenguajes para permitir que los usuarios se conecten al juego desde plataformas dispares, pero esto no implica que el software desarrollado sea multiplataforma.

Ya que se trata de un sistema multiusuario distribuido por la red, se ha decidido utilizar una arquitectura cliente-servidor. Por motivos de seguridad y eficiencia, la capacidad de proceso se ha centrado en el servidor. El cliente solo envía al servidor los comandos de texto que el usuario introduce y visualiza aquellos mensajes que son enviados al cliente por el servidor. El servidor, por otro lado, interpreta esos mensajes mediante la lógica de juego y envía la respuesta a los diferentes clientes. En ningún caso los clientes pueden comunicarse directamente entre ellos.

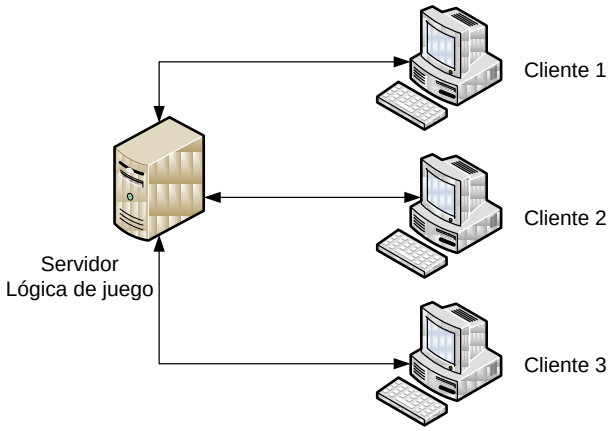


Ilustración 26: Diagrama cliente-servidor multiusuario

En un primer lugar se pensó en utilizar exclusivamente la tecnología *Windows Comunication Fundation* de *Microsoft* para permitir el acceso al juego. Pese a la robustez de esta tecnología y a su largo porvenir, en la actualidad solo dispositivos con sistemas operativos *Windows* soportan dicha tecnología. Por este motivo se decidió optar por una tecnología que fuese accesible por un mayor número posible de personas. El segundo protocolo estudiado fue *telnet*, tecnología que utilizan actualmente la mayoría de MUDs para permitir la conexión a ellos. Sin embargo, pese a que casi todas las plataformas permiten el uso de este protocolo, se trata de una tecnología muy antigua con fallos de seguridad y restricciones de uso considerables. Tras

esto se estudio la posibilidad de utilizar el lenguaje HTML como medio de conexión accesible por todos. Sin embargo, este lenguaje no permite de forma correcta la creación de clientes que se ajusten a las necesidades de este tipo de juegos. Como método alternativo se pensó en la creación de clientes basados en Plug-in u objetos ActiveX, pero estas tecnologías dieron resultados de rendimiento pobres y, sobre todo, eran completamente inaccesibles para el colectivo con discapacidad visual. Finalmente, investigando sobre el estado actual de la tecnología *AJAX*, se descubrió que el nuevo lenguaje *HTML5* permite en sus nuevas características desarrollar un cliente puramente web capaz de conectarse al juego.

Una vez descubiertos los diferentes medios posibles de conexión, sus ventajas e inconvenientes, se decidió por crear un sistema que permitiese la conexión al juego desde cualquiera de los protocolos que se analizaron con mejores resultados. Los protocolos elegidos fueron:

- Telnet
- Windows Comunication Fundation
- HTML5

El objetivo final es que existan multitud de plataformas que permitan acceder al juego, desde móviles hasta ordenadores de sobremesa con cualquier sistema operativo. Sin olvidar el objetivo específico de que todos los usuarios sean tratados de forma similar ante la lógica de juego.

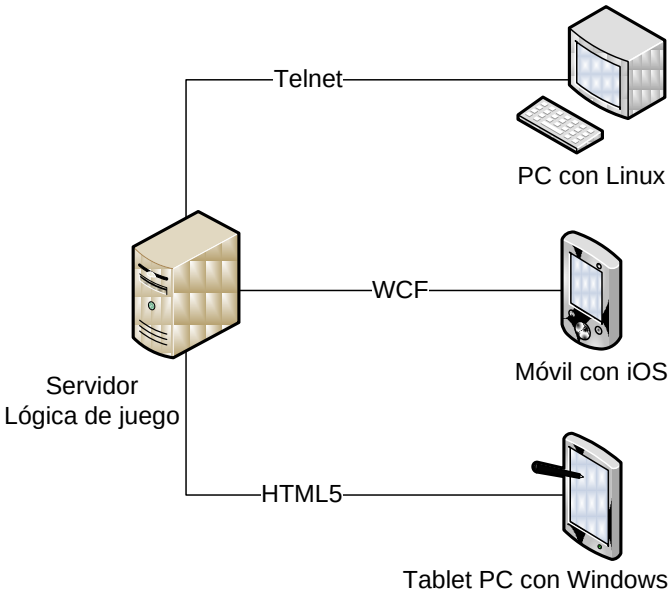


Ilustración 27: Diagrama multiterminal

A lo largo de este capítulo se analizan por orden cada una de las tecnologías implementadas, para finalmente presentar un esquema general de la arquitectura desarrollada.

5.1 Telnet

Telnet es el nombre de un protocolo de red que sirve para acceder mediante una red a otra máquina para manejarla remotamente como si estuviéramos sentados delante de ella. También es el nombre del programa informático que implementa el cliente. Para que la conexión funcione, como en todos los servicios de Internet, la máquina a la que se acceda debe tener un programa especial que reciba y gestione las conexiones. El puerto que se utiliza generalmente es el 23. Telnet sólo sirve para acceder en modo terminal, es decir, sin gráficos, pero fue una herramienta muy útil para arreglar fallos a distancia, sin necesidad de estar físicamente en el mismo sitio que la máquina que los tenía. También se usaba para consultar datos a distancia, como datos personales en máquinas accesibles por red, información bibliográfica, etc.

En el caso del juego, este protocolo es perfecto para permitir a los jugadores interactuar con el juego. El requisito fundamental es programar en el lado del servidor el protocolo que soporte estas conexiones. De cara a los usuarios existen multitud de aplicaciones cliente que permiten la conexión, aun que se ha decidido implementar una aplicación propia para el proyecto.

5.1.1 Servidor Telnet

El servidor telnet es el subprograma del sistema que permite la conexión de los clientes mediante el protocolo telnet. Su función de cara al sistema general es tratar las conexiones entrantes y encapsularlas en una interfaz que permita a la lógica del juego interactuar con dicha conexión independientemente de que se esté realizando bajo el protocolo telnet. Para ello el servidor consta de dos módulos fundamentales.

Por un lado, la clase *ServidorTelnet* es la encargada de abrir los puertos TCP 23, 80, 81 y 3000. Por cada puerto nombrado se inicia un nuevo hilo que gestionada cada conexión. En cada hilo el servidor espera la nueva conexión de un usuario. Cuando el evento de llegada de un nuevo usuario se dispara, el servidor telnet instancia una nueva clase *ConexiónTelnet* que es enviada al servidor general y el hilo vuelve a quedar dormido a la espera de una nueva conexión.

Para permitir el envió y recepción de mensajes por parte de los usuarios que se conectan al juego ha sido necesario la implementación específica de la clase abstracta *IConexion* para el protocolo telnet. Antes de comenzar a explicar el funcionamiento de la clase implementada, es importante resaltar cuales son los métodos y propiedades de la clase abstracta nombrada. Uno de las propiedades más representativas de dicha clase es el concepto de *Columnas*. Esta propiedad especifica el número de caracteres máximo que puede leer un usuario en una sola línea de texto. Este dato es fundamental para que la lógica de juego pueda presentarle la información de forma correctamente formateada. Por otro lado, los métodos *EnviarMensaje* y *RecibirMensaje* son los encargados de gestionar el flujo de mensajes, tanto de salida como del entrada, del usuario. El siguiente diagrama UML representa la clase IConexion:

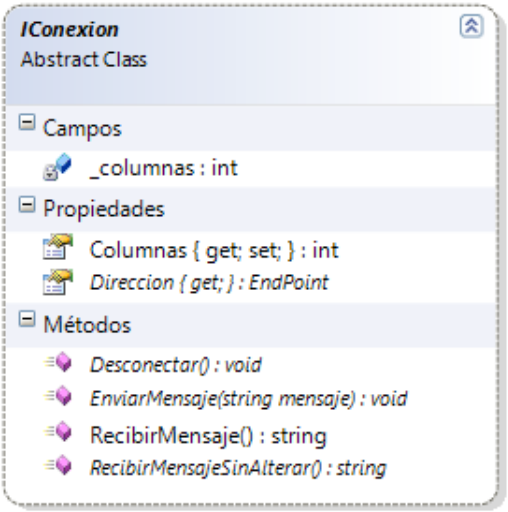


Ilustración 28: Diagrama de clases IConexion

Para permitir a la lógica de juego interactuar con los usuarios que se conectan mediante el protocolo telnet se ha creado una clase que hereda de IConexion. Esta clase implementa el protocolo telnet en su totalidad y se sirve de la clase *TpcClient* (incluida en la propia *API* de *.NET Framework 4.0*) para enviar y recibir información en forma de bytes. Para la codificación y decodificación de los mensajes de texto a formato byte se ha utilizado la codificación *ISO-8859-1*.

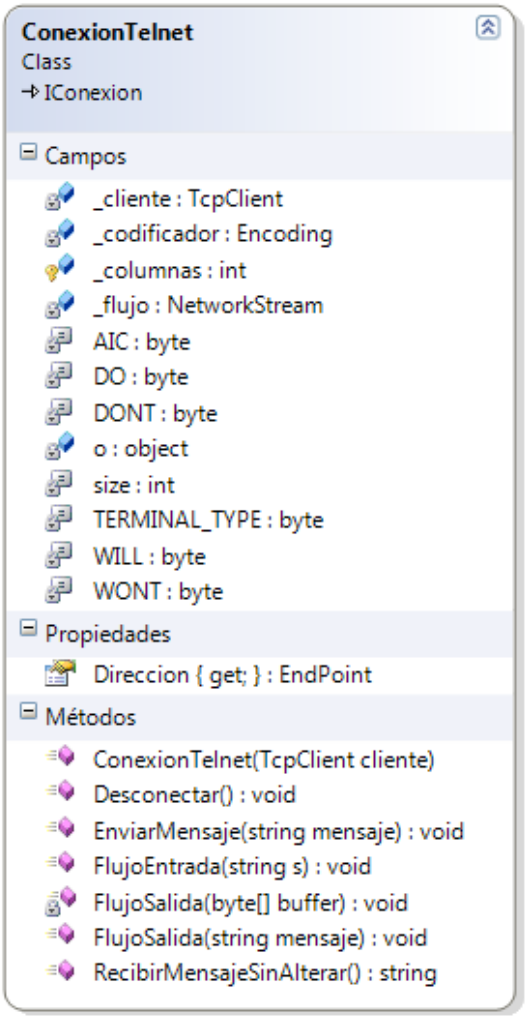


Ilustración 29: Diagrama de clases ConexionTelnet

5.1.2 Cliente Telnet

La gran longevidad de este protocolo ha permitido que se desarrollen multitud de clientes telnet para plataformas dispares. Además, el gran uso del protocolo por los MUDs a favorecido el nacimiento de aplicaciones exclusivamente preparadas para disfrutar de este tipo de juegos a través de este protocolo.

En primer lugar existen diferentes aplicaciones genéricas para realizar conexiones telnet. Cualquiera de ellas puede ser usada para conectar al juego. El listado siguiente muestra alguna de las aplicaciones más utilizadas por diferentes plataformas:

- PuTTy
- AbsoluteTelnet
- Host Explorer
- RUMBA
- Line Mode Browser

- NCSA Telnet
- TeraTerm
- Passport
- SecureCRT
- TeSSH
- ZOC Terminal
- SyncTERM
- PCMan
- PowerTerm InterConnect

Por otro lado, la masiva utilización de este protocolo por parte de los MUDs ha favorecido el surgimiento de aplicaciones especializadas en este tipo de juegos. Permiten utilizar el protocolo telnet para conectarse al juego, pero además incluyen gran cantidad de características que mejoran la experiencia de usuario. Según *The MUD Connector* (Cowan, 2008), existen multitud de clientes telnet especializados en conexiones a MUDs, denominados *MUD Clients*:

- AL MUD Client
- BIM Telnet Client
- Clio MUD2 Client
- Fire Client
- GGMUD
- KMud
- Lyntin
- MMUCL
- MUSHClient
- MudMaster
- Nettern
- Pueblo/UE
- SClient
- SMud 1.0
- Savitar
- SimpleMU
- Stick in the MUD
- The Bean Java MUD Client
- The Spod!
- XpertMUD
- TinTin++
- Trebuchet Muck Client
- WinMUD
- zMud
- cMud
- tkMOO
- ymusk

Además de todo ello se ha querido realizar una implementación propia de una aplicación telnet con las características propias de un cliente especializado en el juego de MUDs. Su descripción bien valdría la escritura de un documento completo, pero no será incluida en este documento para facilitar su lectura. A continuación se muestra una captura de pantalla del cliente desarrollado.

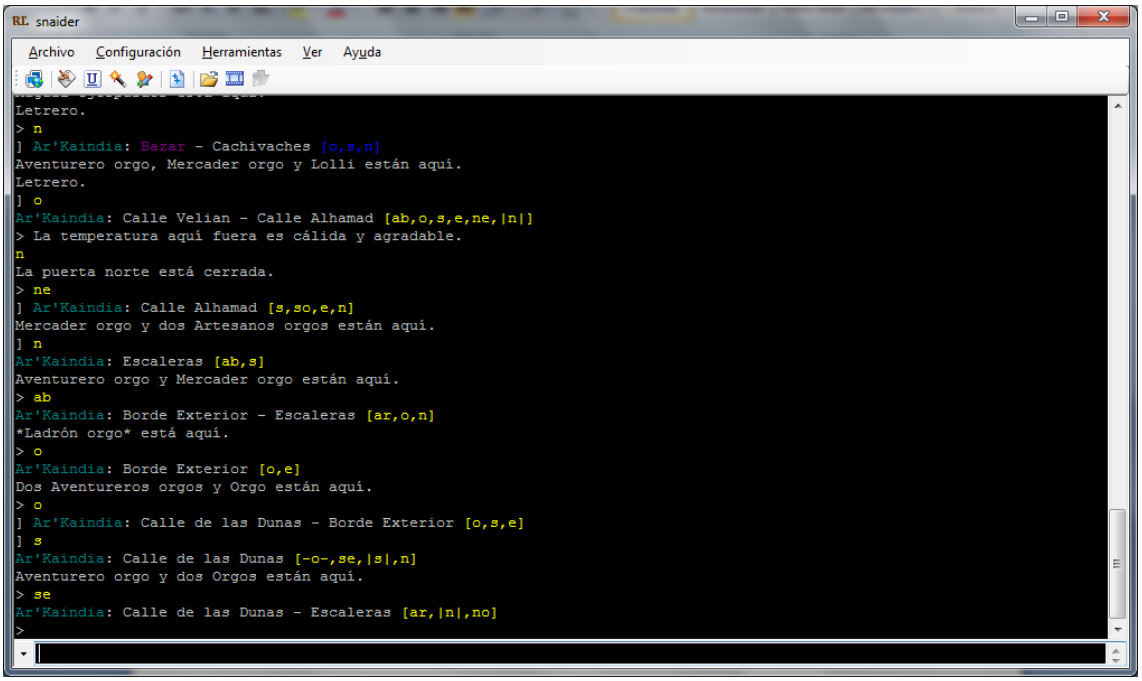


Ilustración 30: Aplicación desarrolada para el protocolo Telnet

5.2 WCF

WCF es el acrónimo inglés de *Windows Comunication Fundation*, tecnología creada por *Microsoft* para unificar las tecnologías informáticas distribuidas de la compañía. Nadie podría explicar mejor que ellos el propio funcionamiento de esta tecnología; por ello a continuación se muestra la definición que ellos ofrecen:

> ***La aceptación global de servicios Web que incluye los protocolos estándar para la comunicación de aplicación a aplicación, ha cambiado el desarrollo de software. Por ejemplo, las funciones que proporcionan los servicios Web ahora incluyen seguridad, coordinación de transacciones distribuidas y una comunicación fiable. Las ventajas de los cambios en servicios Web se deberían reflejar en las herramientas y tecnologías que los programadores utilizan. Windows Communication Foundation (WCF) está diseñado para ofrecer un enfoque manejable a la informática distribuida,***

interoperabilidad ancha y asistencia directa para la orientación sobre el servicio.

WCF simplifica el desarrollo de aplicaciones conectadas a través de un nuevo modelo de programación orientado a servicios. WCF admite muchos estilos de desarrollo de aplicaciones distribuidas proporcionando una arquitectura superpuesta. En su base, la arquitectura de canal de WCF proporciona primitivos asíncronos de paso de aprobación de mensajes sin tipo. Generados sobre esta base están las funciones de protocolos para un intercambio de datos de transacción seguro y fiable, así como una amplia variedad de opciones de codificación y transporte.

El modelo de programación tipificada (llamado modelo de servicio) está diseñado para facilitar el desarrollo de aplicaciones distribuidas y proporcionar a los desarrolladores pericia en servicios Web ASP.NET, comunicación remota .NET Framework y Enterprise Services, así como a aquellos que llegan a WCF con cierta experiencia en desarrollo. El modelo de servicio presenta una asignación sencilla de conceptos de servicios Web para aquellos de Common Language Runtime (CLR) .NET Framework, incluyendo la asignación ampliable y flexible de mensajes para la implementación de servicios en lenguajes como Visual C# o Visual Basic. Incluye funciones de serialización que habilitan el acoplamiento separado y el control de versiones y proporciona integración e interoperabilidad con sistemas distribuidos .NET Framework existentes, como Message Queue Server (MSMQ), COM+, servicios Web ASP.NET, Mejoras de servicios Web (WSE) y varias funciones más.

Para recabar mayor información sobre esta tecnología se recomienda la lectura de su información conceptual en la web corporativa de la compañía (Microsoft Corporation).

El objetivo es conseguir un medio de conexión que sea soportado por la gran mayoría de plataformas que utilicen sistemas operativos Windows.

Dentro de los servicios WCF existen dos alternativas de servicios: unidireccionales y dúplex. Dado el funcionamiento del juego, en el que el servidor envía información a los usuarios en cualquier momento, es necesario utilizar un contrato *dúplex*. Un contrato de servicios dúplex es un modelo de intercambio de mensajes en el que ambos extremos pueden enviar mensajes al otro de manera independiente. Un servicio dúplex, por tanto, puede enviar mensajes de vuelta al extremo del cliente, proporcionando un comportamiento parecido a los eventos. La comunicación dúplex se produce cuando un cliente se conecta a un servicio y proporciona al servicio un canal en el que el servicio puede devolver los mensajes al cliente.

5.2.1 Servidor WCF

El servidor WCF es, en realidad, un servicio web hospedado bajo las directrices de WCF. Su funcionamiento es sencillo, pues su carácter heterogéneo simplifica enormemente, tanto el trabajo de diseño como la implementación final.

Lo primero que ha de tenerse en cuenta es que el concepto de servicio dúplex descrito anteriormente. Para establecer este tipo de servicios en el lado del servidor hace falta establecer dos interfaces de comunicación. Con este objetivo se crea *IServidorWCF* y *IClienteCallback*.

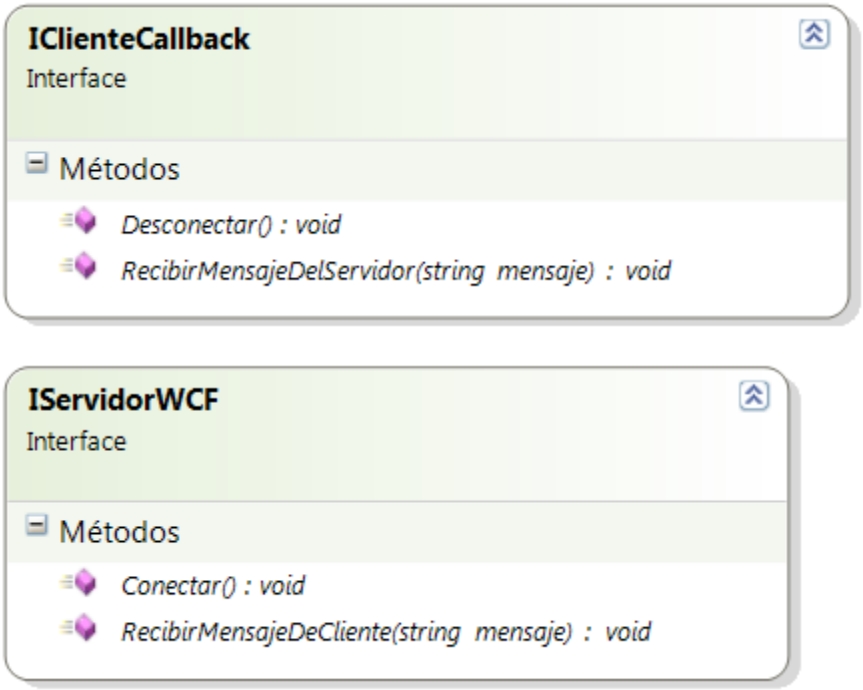


Ilustración 31: Diagrama de clases Interfaces WCF

La interfaz servidora es la que permite a los clientes establecer la conexión y enviar mensajes en forma de cadena de caracteres. La interfaz del cliente, por su parte, ofrece al servidor la posibilidad de enviar mensajes a los clientes y ordenarles la desconexión con el servidor.

La clase *ServidorWCF* cumple la interfaz descrita anteriormente.

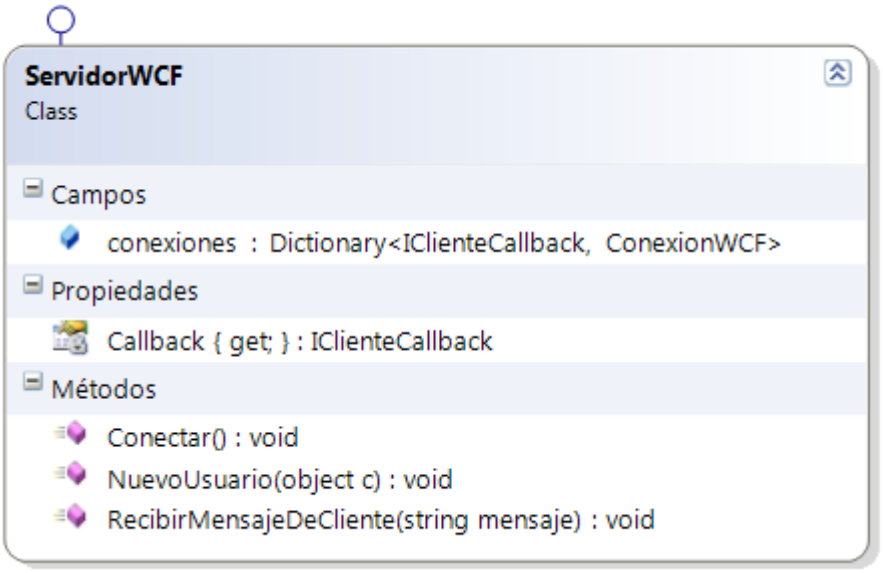


Ilustración 32: Diagrama de clases ServidorWCF

Como puede verse, la recepción de mensajes de los usuarios no queda relevada a una instancia específica para cada cliente, sino que es el propio servidor WCF el que recibe todos los mensajes de todos los clientes WCF conectados a él. Gracias al diccionario *conexiones*, el servidor puede entonces encauzar el flujo de información hacia la permitente instancia *IConexion* implementada para esta tecnología.

Una vez más, para que el servidor general pueda tratar por igual a los jugadores que utilicen la tecnología WCF, ha sido necesario crear una clase específica que herede de la interfaz IConexion. Esta clase se ha denominado *ConexionWCF*.

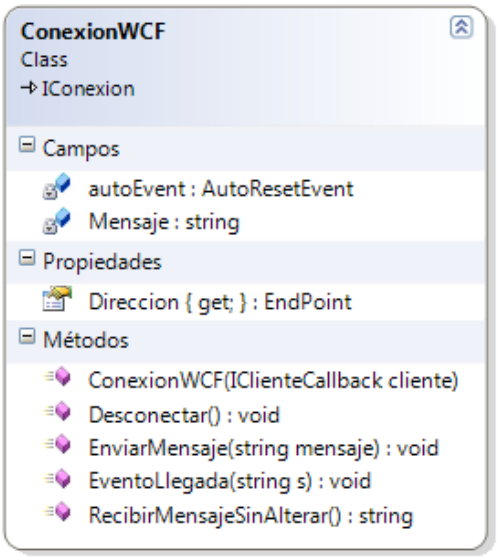


Ilustración 33: Diagrama de clases ConexionWCF

Su funcionamiento es realmente sencillo pues no utiliza métodos de codificación y decodificación. La propia capa de presentación interna de WCF realiza ese trabajo. La única complicación encontrada ha sido tratar de emular una ejecución secuencial mediante esta tecnología, orientada a eventos. El objeto es que un solo hilo controle la recepción de mensajes por parte de los clientes. Este hilo debe dormirse a la espera de recibir un nuevo mensaje, tal y como se realiza en el servidor telnet. Sin embargo, WCF no permite parar el hilo de ejecución implícitamente. Como solución a este problema se ha utilizado un semáforo, que se desactiva a la hora de esperar un mensaje, y es activado desde el servidor WCF cuando este recibe un mensaje de dicho cliente.

5.2.2 Cliente WCF

Para la prueba del servicio dúplex creado se ha implementado una pequeña aplicación de escritorio utilizando la tecnología WCF y *WPF* (Windows Presentation Fundation). Una vez más, si descripción puede suponer un nuevo capítulo para el documento, y por ello se ha decidido no incluir nada en esta memoria.

A continuación se incluye una captura de pantalla de la aplicación desarrollada:

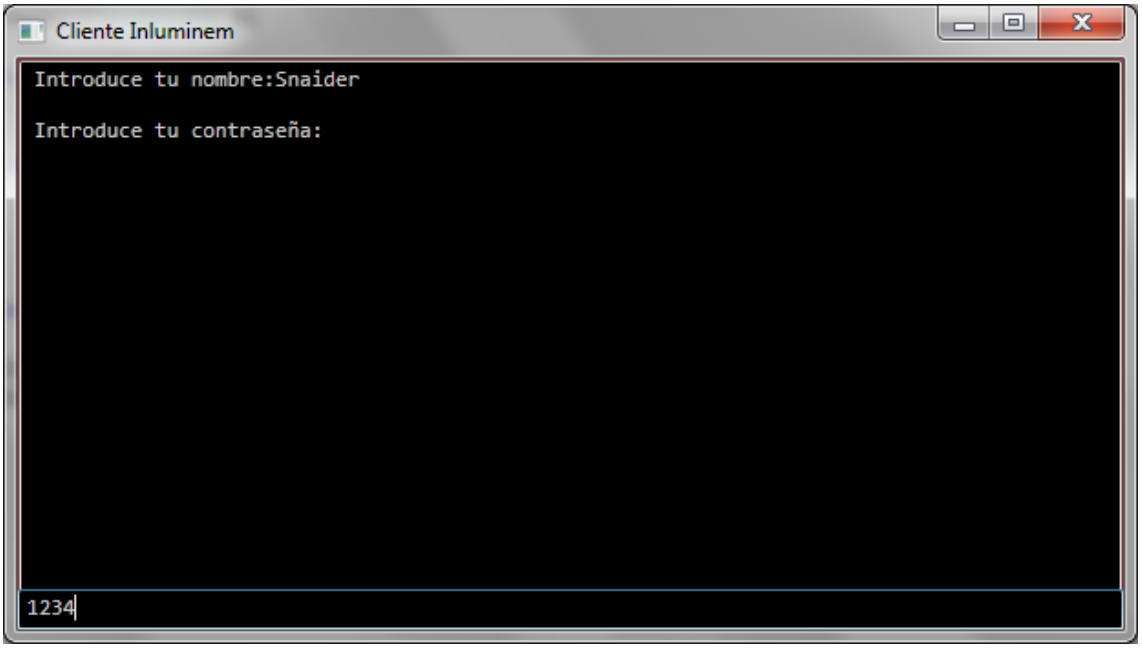


Ilustración 34: Aplicación desarrollada para WCF

5.3 HTML5

Uno de los principales motivos por los que se comenzó a estudiar el lenguaje HTML como posible medio de conexión fue por su carácter global. Actualmente cualquier persona con acceso a Internet dispone de un navegador capaz de interpretar código HTML. Es lógico pensar por tanto, que si desarrollásemos una aplicación basada puramente en código HTML sería accesible por cualquier persona y por tanto estaríamos cumpliendo los objetivos de la filosofía *diseño para todos*.

Sin embargo, la versión actual de HTML no permite la creación de páginas capaces de ofrecer un medio de conexión al juego. No es así con la nueva versión de HTML, el llamado HTML5, y en concreto su nueva API *WebSockets*. Todavía es una tecnología muy nueva, y el apoyo está empezando a ser implementado en la mayoría de los principales navegadores y servidores web. Sin embargo, no cuentan con él todavía, y la mayoría de los clientes de la aplicación web no podrá usarla. Sin embargo, su futuro es muy esperanzador, y conviene hablar de WebSockets y como esta tecnología pretende cambiar el futuro de las webs.

La World Wide Web e Internet comenzó como un mecanismo de entrega de contenido estático, dando un paso atrás en comparación con las aplicaciones de escritorio tradicionales. En las primeras aplicaciones de Internet era necesario solicitar explícitamente cada pieza de información, y el servidor enviaba sólo los datos solicitados.

Entonces vino lo que hoy conocemos como "desarrollo de aplicaciones Web 2.0": HTML dinámico, uso intensivo de JavaScript, AJAX, y varios plugins (Flash y Microsoft Silverlight). Estas aplicaciones eran dinámicas y flexibles, y trajeron a la mayor parte de los usuarios una rica experiencia interactiva disfrutando de aplicaciones de escritorio cliente-servidor en la web. Sin embargo, debido a la arquitectura de solicitud y de respuesta en que estas aplicaciones se basan, las últimas aplicaciones dinámicas de Internet todavía no pueden obtener datos en tiempo real como hacían aplicaciones de escritorio cliente-servidor hace más de una década atrás.

La razón es que estas aplicaciones todavía tienen que iniciar la comunicación desde el navegador web para obtener datos desde el servidor, es decir, la parte cliente de la aplicación que se ejecuta en el navegador no tiene manera de saber si el servidor tiene algunos datos para él.

Por lo tanto, los ingenieros del software comenzaron a trabajar en la solución a este problema. El cliente necesita ser consciente de las peticiones que el servidor puede tener para él. Pero, ¿cómo lograr esto? Bueno, si la comunicación sólo puede ser iniciada por el cliente, entonces lo vamos a hacer de esa manera. Vamos a hacer que el cliente siempre pregunte al servidor si hay algunos datos preparados para él.

Mediante el uso de esta técnica, la aplicación web consulta el servidor de forma regular, sobre la base de un contador de tiempo, usando una simple petición HTTP cada vez que se agota el tiempo. Sin embargo, el coste de las peticiones frecuentes es muy alto - tanto en términos de ancho de banda de red, así como la infraestructura de servidores necesarios para apoyar un gran número de consultas.

Después de un corto tiempo, esta técnica se vio totalmente inutilizable debido a la necesidad de grandes recursos. Luego vinieron las técnicas de programación Comet – long polling y streamin.

En long polling, también conocido como consulta asíncrona, el navegador envía una petición al servidor y el servidor mantiene la solicitud de apertura un período de tiempo determinado. Si se recibe una notificación en ese plazo, se envía al cliente una respuesta que contiene el mensaje. Si la notificación no se recibe dentro del plazo establecido, el servidor envía una respuesta de terminar la solicitud de consulta y el navegador vuelve a enviar la solicitud al servidor.

Cuando se utiliza streaming, el navegador envía una solicitud completa, pero el servidor envía y mantiene una respuesta abierta que se actualiza continuamente. Se envía una solicitud y se mantiene abierta de forma indefinida (o por un período de tiempo) y la respuesta se actualiza cada vez que un mensaje esté listo para ser enviado, pero el servidor no da señales de completar la respuesta, lo que mantiene abierta la conexión para enviar mensajes futuros.

Pero si se piensa bien, estas técnicas son sólo trucos, trucos utilizados para simular una tecnología que no existe: el envío de eventos por el servidor. En realidad, si el servidor pudiese iniciar la comunicación, ninguno de estos trucos feos serían necesarios.

Se ha tenido que esperar hasta que naciese HTML5 para resolver este problema. La sección de Comunicación de la especificación HTML5 introduce el envío de eventos del servidor web y WebSockets. Estas nuevas características permiten un nuevo paradigma de programación de aplicaciones web que va a revolucionar la manera de desarrollar y desplegar aplicaciones web.

El envío de eventos por parte del servidor no es del todo la solución, sino una formalización de las técnicas de programación Comet. La especificación HTML5 sobre el envío de eventos introduce un nuevo elemento DOM llamada EventSource.

Pero el elemento sobre el que tiene que caer toda nuestra atención es a la especificación de HTML5 sobre WebSockets: la interfaz WebSockets define un canal de comunicación full-duplex que se expone a través de una interfaz de JavaScript en los navegadores compatibles con HTML5.

Cuando una nueva conexión WebSockets se ha establecido, el navegador abre una conexión HTTP al servidor y luego negocia con el servidor para mejorar la conexión a un canal de conexión WebSockets dedicado y persistente. Este proceso se ajusta automáticamente a través de un túnel con el servidor, resolviendo numerosos problemas de las diferentes técnicas de programación Comet. Una vez establecido, WebSockets es un canal full dúplex entre el cliente y el servidor.

Sin embargo, esta tecnología reciente aun no ha sido implementada por muchos navegadores. Por otro lado, algunos navegadores se inician con la característica de WebSockets desactivada de forma predeterminada, con motivo de unos problemas de seguridad encontrados en el protocolo que aun no han sido resueltos. La siguiente tabla (Deveria, 2011) muestra el estado actual de los navegadores respecto a su implementación de WebSocket:

Tabla 6: Soporte de WebSockets en los principales navegadores

Versiones	IE	Firefox	Safari	Chrome	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
Dos versiones atrás	7.0	3.6	3.2	10.0	11.0	3.2		10.0	2.1

Versiones	IE	Firefox	Safari	Chrome	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser	
Version anterior	8.0	4.0	4.0	11.0	11.1	4.0-4.1		11.0	2.2	
Actual	9.0	5.0	5.0	12.0	11.5	4.2-4.3	5.0-6.0	11.1	2.3	3.0
Futuro cercano		6.0	5.1	13.0	12.0					
Futuro lejano	10.0	7.0		14.0	12.1					

Soportado, no soportado, parcialmente soportado y soporte desconocido.

Además de la enorme ventaja que supone tener un canal de comunicación full dúplex gracias a HTML5, existe otra gran ventaja que merece la pena describir. Esta nueva versión de HTML es completamente compatible con la última versión del IOS de Apple a través de su navegador web Safari, lo cual permitirá que los dispositivos iPhone/iPad de Apple puedan beneficiarse de las nuevas posibilidades que ofrece HTML5; sobre todo en multimedia y videojuegos, llenando así el vacío que existe actualmente en estos dispositivos al no soportar Flash. Sin embargo, éstas son solo algunas de las novedades de las que pueden sacar provecho estos dispositivos para ofrecer nuevas mejoras de cara a sus usuarios. A continuación comentaremos algunas de estas novedades:

- Disponibilidad de una base de datos para almacenamiento DOM (‘Document Object Model‘), la cual permite almacenar información de forma segura en el lado del cliente para ser consultada posteriormente por la aplicación. Esto agiliza la rapidez de las consultas al estar los datos almacenados localmente en el cliente .A partir de la versión 2.1 del nuevo iPhone, estos dispositivos ya tienen disponible una base de datos para este tipo de almacenamiento.
- Existencia de una Caché de aplicaciones. Esta es una opción muy interesante, pues cuando guardamos una aplicación web, creando un icono en el Home Screen (pantalla principal donde se ubican los iconos de nuestras aplicaciones) de nuestro iPhone, si esta utiliza dicha caché, en ella se almacenarán datos que permiten una ejecución más rápida de la aplicación incluso sin estar conectados a internet (similar a *Google Gears* (Pimentel, 2007)). Para que la aplicación se guarde en caché tendremos que especificarlo mediante el tipo MIME correspondiente. Sin embargo, esta característica será exclusiva de la versión Webkit de Safari Mobile, y no aparecerá en ninguna otra versión. Aun que esta característica en no supone una gran aportación al proyecto, gracias a esta opción las aplicaciones web pueden resultar más prácticas y poseer una mayor disponibilidad para los usuarios, pues éstos no necesitan disponer de conexión a internet para hacer uso de las aplicaciones.
- Al ser HTML5 una versión orientada al desarrollo de sitios web y aplicaciones modernas, posee etiquetas para uso de canvas 2D, 3D, audio y vídeo, lo que permite que los navegadores reproduzcan contenidos multimedia de forma nativa sin necesidad de plugins como Adobe Flash Player. Algunos sitios como YouTube ya barajan la posibilidad de migrar a esta nueva tecnología ya que les reportaría una mayor compatibilidad entre las diferentes plataformas (PC, iPhone, Android, Mac,…) facilitando el acceso de un número mayor de usuarios a sus contenidos. En un futuro

esta característica permitiría mejorar la experiencia del juego mediante la incorporación de contenidos multimedia dentro o fuera del juego.

5.3.1 Servidor HTML5

Una de las tareas más complejas del proyecto ha supuesto la incorporación de un servidor WebSockets. La API del lenguaje de servidor utilizado, ASP .NET en su versión 4.0, no incorpora ningún tipo de implementación referida a los WebSockets.

En resumen, la implementación del servidor ha consistido en establecer un *Socket* en el puerto 8181, gestionar cada nueva conexión mediante la creación de un nuevo hilo y realizar la negociación especifica del protocolo descrita en el RFC (Fette, 2011). El siguiente diagrama UML muestra la clase implementada:

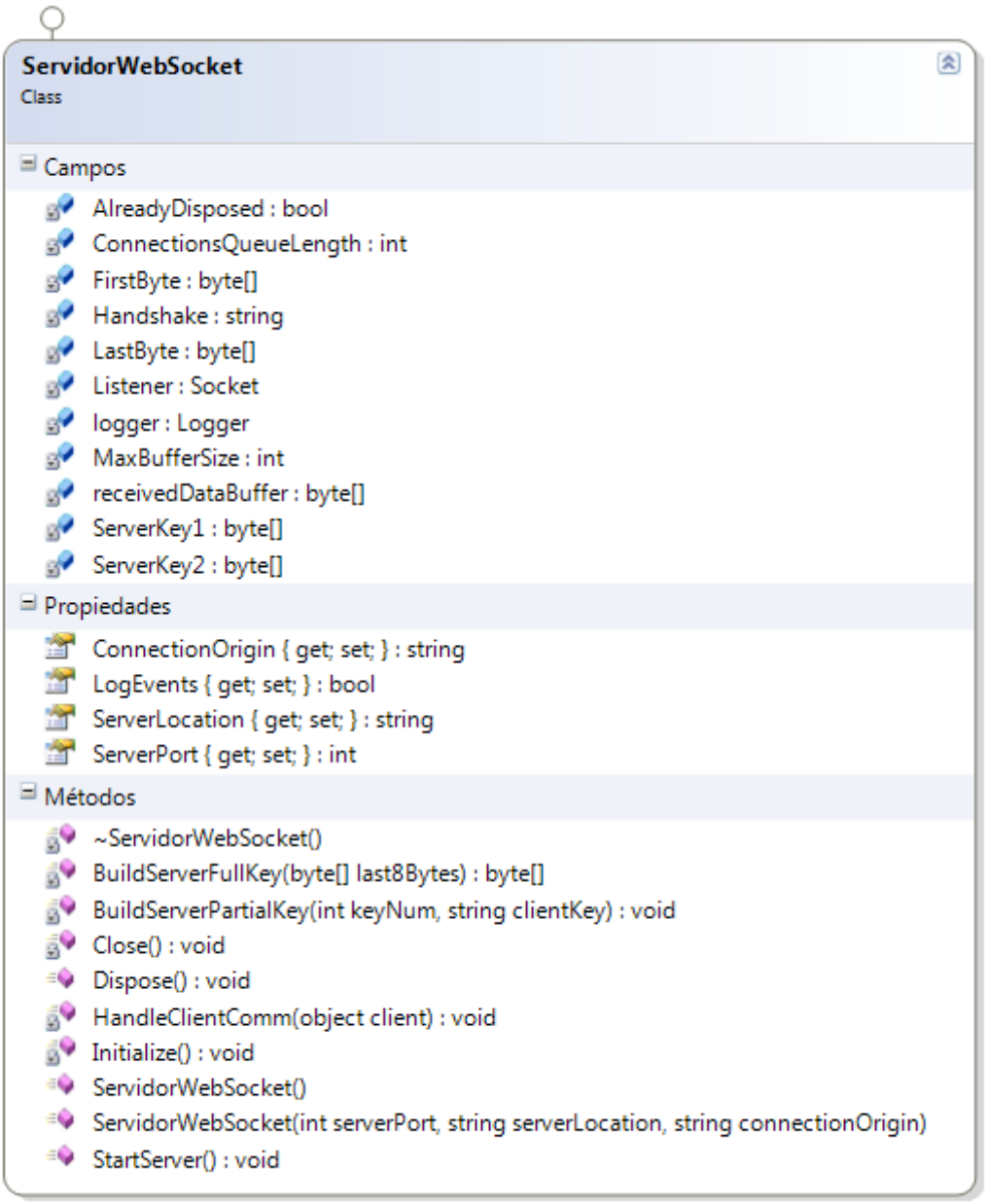


Ilustración 35: Diagrama de clases ServidorWebSocket

Una vez más ha sido necesario crear una clase que cumpla la interfaz IConexion vista anteriormente con el objetivo de permitir que los usuarios conectados mediante WebSockets puedan ser tratados de igual forma por la lógica del juego. Por suerte, la implementación de esta clase no ha sido tan compleja como el servidor. La utilización de la API de *Socket* permite, sin muchas modificaciones con respecto a lo habitual en el uso de esta librería, enviar y recibir mensajes por parte de los clientes. El siguiente diagrama muestra la clase implementada:

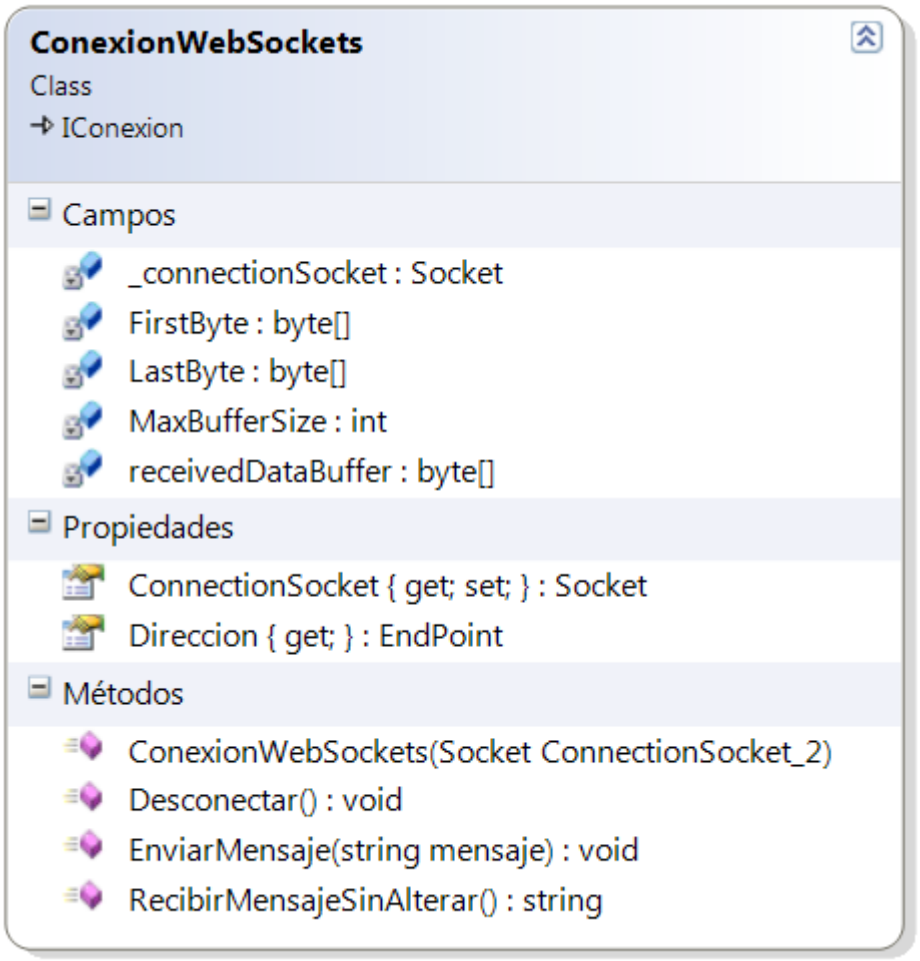


Ilustración 36: Diagrama de clases ConexionWebSockets

5.3.2 Cliente HTML5

A diferencia de los otros medios de conexión al juego, en los que las aplicaciones son programadas exclusivamente para realizar un tipo de conexión específica (véase las aplicaciones WCF), en este caso los clientes que utilizan este medio de conexión son meros intérpretes que deben adquirir el código específico que permita la realización de la conexión. Por este motivo, en este apartado no se hace referencia a los navegadores web como clientes HTML5, sino que se entiende por cliente al código enviado a los navegadores para que estos puedan realizar la conexión.

En primer lugar hay que destacar que la tecnología utilizada es HTML5, y por tanto el lenguaje de programación utilizado es *JavaScript*. La versión HTML5 incorpora una nueva API, descrita en el apartado anterior, denominada WebSockets. Como también se ha dicho, esta API aun no es soportada por todos los navegadores. Por ello, el primer objetivo del programa JavaScript es comprobar si el navegador web es compatible con esta API. Para ello se intenta instanciar una variable de tipo *WebSocket*. Si esto produce una excepción entonces el navegador no es

compatible con esta API. El siguiente fragmento de código implementa este método de comprobación:

```
    var WebSocketsExist = true;
    try {
        var dummy = new WebSocket("ws://finance.example.com");
    } catch (ex) {
        WebSocketsExist = false;
    }

    if (WebSocketsExist) {
        Log("Tu explorador soporta Web Sockets.", "OK");
        Conectar();
    } else {
        Log("Lo sentimos, tu explorador no soporta Web Sockets.", "ERROR");
    }
```

Código 5: Comprobación de soporte de WebSockets

Una vez comprobada la compatibilidad con WebSockets, el siguiente paso es establecer la conexión con el servidor. Ya que el servidor no dispone de una dirección IP permanente, es necesario modificar el código JavaScript para adaptarse a la configuración actual del servidor. ASP .NET, al igual que PHP, permite realizar cambios en la estructura de una página web antes de ser entregada al navegador. En este caso, se ha utilizado el método *Page_Load* para realizar estos cambios. El siguiente código muestra la inyección de código JavaScript a la página, en el que se instancia la variable *WebSocket* utilizando los parámetros apropiados a la configuración del servidor:

```
protected void Page_Load(object sender, EventArgs e)
    {

        Type cstype = this.GetType();

        ClientScript.RegisterStartupScript(cstype, "onload",
            "<script type=\"text/javascript\">" +
            "function Conectar() {" +
            "Log(\"Conectando con el servidor ...\", \"OK\");" +
            "try {" +
            "   ws = new WebSocket(\"ws://" + Servidor.IPLocal.ToString() +
":8181/Inluminem\");" +
            "  SocketCreated = true;" +
            "} catch (ex) {" +
            "   Log(ex, \"ERROR\");" +
            "   return;" +
            "}" +
            "ws.onopen = WSonOpen;" +
            "ws.onmessage = WSonMessage;" +
            "ws.onclose = WSonClose;" +
            "ws.onerror = WSonError;" +
            "};" +
```

```
            "Iniciar();"+
            " </script>");


        }
```

Código 6: Inserción de JavaScript específico

Finalmente, los métodos principales de gestión de la conexión son *WSonMessage* y *SendDataClicked*. El método WSonMessage se desencadena cuando el servidor envía un mensaje al usuario. La forma en la que se visualiza la información enviada por el servidor es modificando dinámicamente al árbol DOM, incorporando nuevo elementos. Además, es necesario incorporar las técnicas JavaScript descritas en el apartado Interacciones que facilitan la accesibilidad para permitir que los usuarios con discapacidad visual puedan utilizar lectores de pantalla. El siguiente cuadro de código corresponde al evento descrito y al método utilizado para visualizar la información en pantalla:

```
function WSonMessage(event) {
    Log(event.data);
};


function Log(Text, MessageType) {

    if (MessageType == "OK") Text = "<span style='color: green;'>" + Text +
"</span>" + "<br/>";
    if (MessageType == "ERROR") Text = "<span style='color: red;'>" + Text +
"</span>" + "<br/>";

    var objAnchor = document.createElement('a');
    objAnchor.setAttribute('href', '#target');
    objAnchor.setAttribute('onkeydown', 'TeclaPulsadaEnlace(event);');
    objAnchor.innerHTML = Text;

    var LogContainer = document.getElementById("log");
    LogContainer.appendChild(objAnchor);

    LogContainer.scrollTop = LogContainer.scrollHeight;
    objAnchor.focus();

};
```

Código 7: Menajo de mensajes de entrada

El método SendDataClicked es desencadenado cuando el usuario imprime la tecla *enter*. Este evento envía al servidor el texto que se ha introducido en el cuadro de texto disponible en la interfaz de usuario. Es importante destacar que la página muestra también la información que el usuario envía al servidor, pero esta no es visualizada de igual forma que la recibida por el servidor, con el objetivo de evitar así que los lectores de pantalla lean sólo la información realmente relevante. El siguiente fragmento de código corresponde al método nombrado:

```
function SendDataClicked() {
    if (document.getElementById("DataToSend").value != "") {
        ws.send(document.getElementById("DataToSend").value);
        Text = "<span style='color: brown;'>>" +
document.getElementById("DataToSend").value + "</span>";
        document.getElementById("log").innerHTML =
document.getElementById("log").innerHTML + Text + "<br/>";
        document.getElementById("DataToSend").value = "";
        var LogContainer = document.getElementById("log");
        LogContainer.scrollTop = LogContainer.scrollHeight;
    }
};
```

Código 8: Envio de mensajes

Finalmente, se incorpora una captura de pantalla de la web implementada vista desde el navegador *Google Chrome (12.0.742.112)*:

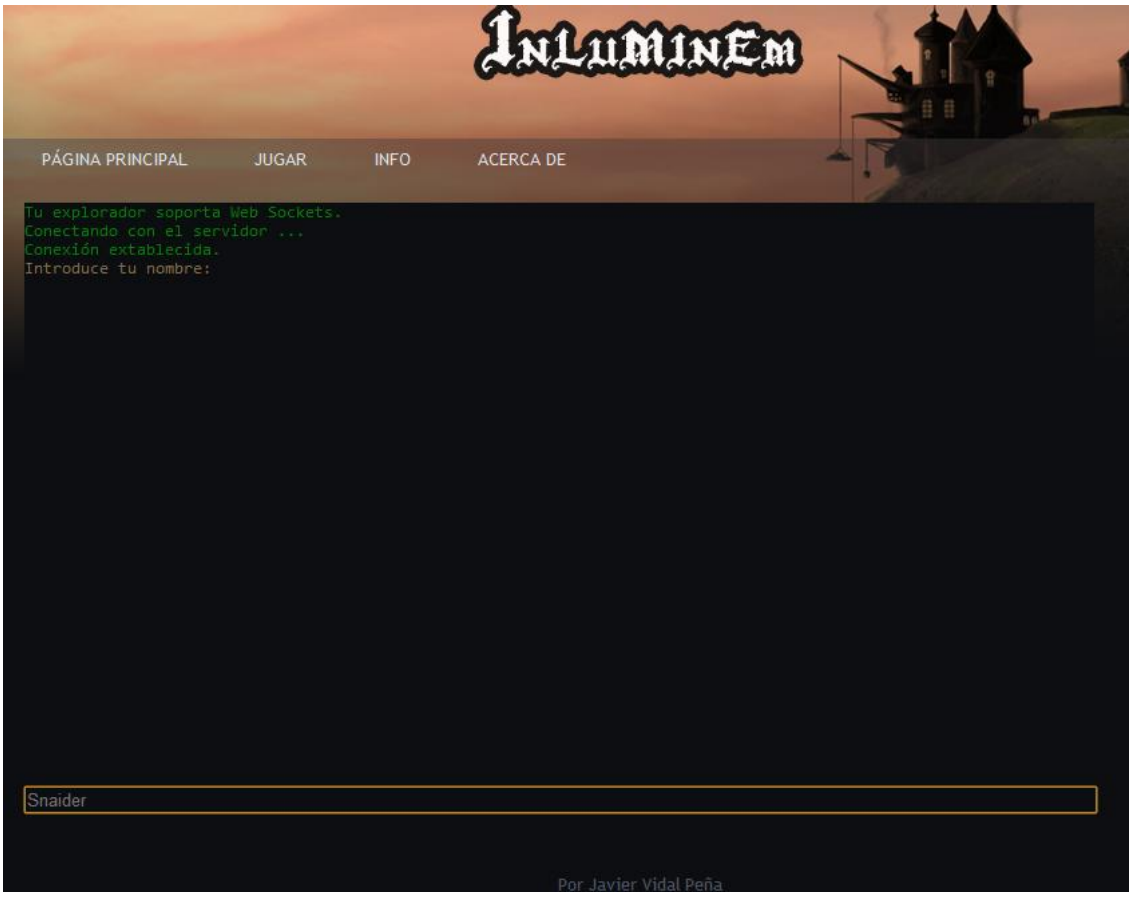


Ilustración 37: Interfaz web creada

5.4 Esquema general

Como se ha visto anteriormente, uno de los objetivos principales del sistema es permitir su conexión desde diferentes medios. De cara a la lógica del juego, el método de conexión elegido por el usuario es desconocido, y por tanto los jugadores son tratados de igual forma independientemente de su forma de conectar. Para hacer esto posible la implementación se ha basado en un modelo basado en capas. En este apartado se analiza la estructura de este modelo, las razones de su diseño y finalmente se muestra el diagrama de despliegue que describe la arquitectura utilizada.

En primer lugar es importante presentar el modelo general que describe el modelo de capas implementado y su funcionalidad. En él se puede visualizar el núcleo de juego, como una entidad independiente del modelo de conexión al juego. En un lugar intermedio se encuentra la capa definida como *Conectividad*, que está formada por las interfaces que posibilitan el objetivo descrito anteriormente. Y finalmente se presentan las tres tecnologías de conexión implementadas, junto con los posibles clientes potenciales que pueden hacer uso de dichos medios de conexión a través de Internet.

Ilustración 38: Modelo multiterminal creado

Sin embargo, el diagrama anterior no refleja el modelo de accesibilidad descrito en el capitulo Modelo de accesibilidad, ni tampoco el funcionamiento de las capas. En primer lugar, la visualización del juego es diferente por cada jugador, y esta depende principalmente de dos características:

- El nivel de accesibilidad: como se vio en el segundo capítulo del documento, no todos los usuarios pueden percibir el juego de la misma forma. Por este motivo el juego ha de mostrar la información textual adaptada a las capacidades de cada usuario. Por ello es necesario crear una capa de accesibilidad que permita transformar la información enviada por el servidor antes de ser presentada.
- El tipo de terminal usado: los mensajes enviados por el servidor permiten un cierto nivel de formato mediante la incorporación de metadatos. Sin embargo, no todos los terminales de conexión son capaces de interpretar los mismos metadatos, y por tanto la información enviada por el juego debe ser una vez más transformada para que su visualización se adapta al dispositivo del usuario.

Con estas dos premisas se unifica el modelo por capas visto en este capítulo junto con el modelo descrito en el capitulo Uso equiparable, creando la capa *Accesibilidad* y *Presentación*.

Por un lado, la primera capa permite la transformación del texto enviado por el juego en función de tres niveles de accesibilidad: ninguna, daltonismo y ceguera total. Mientras que la accesibilidad nula no transforma el texto enviado, los otros dos sí realizan cambios en los datos. El modelo de accesibilidad *daltonismo* trata los colores de forma que estos puedan ser diferenciados por el tipo de daltonismo. La accesibilidad denominada *ceguera total*, elimina por completo el formato del texto y elimina por completo cualquier tipo de dibujo o representación visual realizada con caracteres (un mapa ASCII, por ejemplo).

Por otro, la capa de presentación permite transformar los metadatos del lenguaje abstracto creado en metadatos que dan formato al texto en función del tipo de conexión realizada. Por un lado, los usuarios que utilicen el cliente HTML5 para conectar, lógicamente recibirán los mensajes utilizando el lenguaje de marcado HTML. Por otro, los usuarios que utilicen WCF recibirán el formato del texto mediante *Rich Text Format* (formato de texto enriquecido a menudo abreviado como RTF), formato de archivo informático que es capaz de ser leído por la mayoría de procesadores de textos. Finalmente, los usuarios del protocolo telnet recibirán el formato del texto mediante los denominados *códigos de escape ANSI*.

El siguiente diagrama muestra el flujo de entrada y salida de información desde el servidor. Como puede observarse, la información enviada por el servidor atraviesa las dos capas descritas anteriormente. Sin embargo, el flujo de entrada es sencillo, no atraviesa ninguna capa, y el texto enviado por los usuarios no es proceso por ningún módulo, es entregado directamente al servidor; cumpliendo así las pautas de equilibrio y no discriminación.

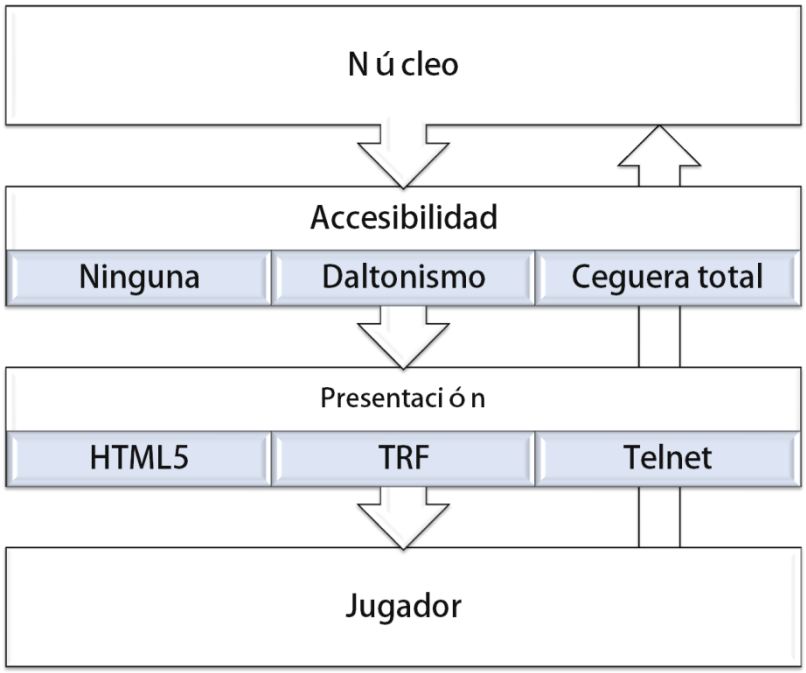


Ilustración 39: Capas finalmente implementadas

La pregunta que queda por resolver es la forma o arquitectura utilizada para hacer posible la interacción entre todos los subsistemas descritos en este documento. Para explicarlo se expone a continuación un diagrama de despliegue. En el diagrama se visualiza el sistema completo englobado dentro del marco ASP. NET. Esto es debido a que todo el sistema se expone bajo un mismo sitio web utilizando la aplicación *Internet Information Services* (IIS). En un principio, el sistema comienza a funcionar cuando una página del juego es visitada. En ese momento, el hilo del servidor crea diferentes hilos de ejecución que inicializan los servidores y el núcleo de juego. De esta manera el servidor HTTP queda liberado y listo para recibir nuevas peticiones HTTP. La base de datos se encuentra alojada junto al sitio web, de forma que la conexión se realiza mediante un acceso relativo al sitio. Por último destacar que, al formar todo parte de un mismo dominio de aplicación, las entidades instanciadas mediante la lógica del juego (o núcleo) puedan ser analizadas en tiempo real mediante las páginas web de contenido. Estas a su vez se sirven de la base de datos para extraer información persistentes que no tiene porqué encontrarse cargada en memoria en esos momentos.

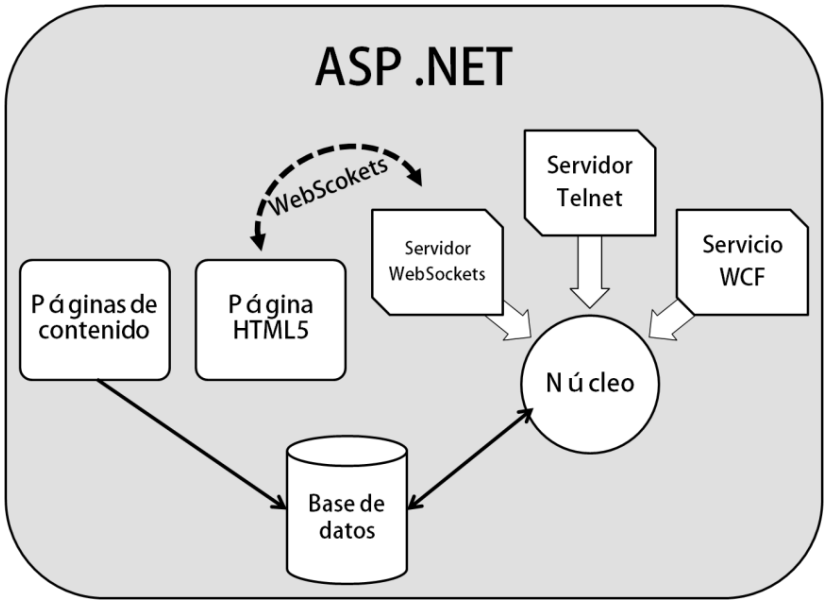


Ilustración 40: Diagrama de despliegue

6 Conclusiones

Finalmente se ha conseguido crear un juego de texto en red capaz de mejorar las relaciones sociales entre personas con discapacidad visual y usuarios sin discapacidad. Los jugadores pueden ahora compartir un mismo mundo y escribir entre todos una misma historia, independientemente de sus características o preferencias.

Mientras que los juegos online masivos siguen acaparando un mercado cada vez mayor, la forma en la que se presentan estos videojuegos sigue siendo completamente inaccesible para el colectivo de usuarios con discapacidad visual. Los juegos basados en texto, como los MUDs, siguen siendo una de las pocas alternativas para estos usuarios. Los lectores de pantalla pueden identificar e interpretar el texto que es enviado por el servidor. Esta interpretación se representa al usuario mediante sintetizadores de texto a voz, iconos sonoros, o una salida braille. La creación paralela de un modelo de accesibilidad siguiendo la filosofía de *Diseño para todos* permite además que cualquier colectivo, con o sin discapacidad, pueda acceder de forma similar y disfrutar de una misma experiencia de juego.

En cuanto al desarrollo, los escasos recursos necesarios para su implementación favorece la creación de este tipo de juegos por parte de grupos pequeños, en algunos casos incluso altruistas. Además, la incorporación de nuevas metodologías de desarrollo adaptadas a las nuevas tecnologías aumentan las posibilidades de desarrollar un juego atractivo e interesante para todos los usuarios. El uso de un sistema de información geográfica para la generación de un mundo rico en contenidos es un ejemplo de ello.

Por otro lado, la creación de un juego en línea en el que todos los jugadores comparten un mismo mundo favorece la creación de comunidades. La incorporación de herramientas de comunicación, la posibilidad de crear grupos dentro del juego y la necesidad de colaboración por parte de los usuarios son herramientas que incrementan aun más las opciones para que un jugador establezca lazos sociales. En este sentido, el juego, al no diferenciar el medio de comunicación usado por el usuario ni su nivel de accesibilidad, no solo mantiene un nivel de juego equilibrado, sino que favorece la relación entre usuarios con diferentes necesidades y características.

El sistema de juego, basado en mensajes de texto, permite su uso desde multitud de plataformas. Proporcionar diferentes medios de conexión es por tanto un logro muy importante que debe cumplirse en este tipo de juegos. Para ello existen numerosas tecnologías que dan apoyo a más o menos plataformas. En este proyecto se han analizado diferentes tecnologías y se han definido los protocolos telnet, WCF y WebSockets como las alternativas más favorables para ser implementadas. Pero para mejorar la experiencia de juego de todos los usuarios es importante que la lógica del juego sea completamente independiente del tipo de conexión que se realice. Para ello se propone un modelo basado en 3 capas: lógica, accesibilidad y presentación.

Finalmente, el uso de la tecnología HTML5 como medio de conexión al juego incrementa, por un lado, el número de terminales diferentes con acceso al juego. Por otro lado, su carácter novedoso reporta al juego una larga vida útil. Numerosas son las plataformas están adoptando compatibilidad con este lenguaje y fomentan la creación de aplicaciones basadas en esta nueva especificación de HTML. Para muchos, este lenguaje representa el futuro de la Web y

de las aplicaciones de escritorio, pues no será necesario más que un único desarrollo para hacer funcionar la aplicación en cualquier dispositivo. Por tanto, el número de plataformas con acceso al juego se verá incrementado con el tiempo.

7 Trabajos futuros

Los trabajos futuros se centran en permitir el acceso a un mayor número posible de personas y mejorar su experiencia de juego. En cuanto a la primera alternativa, el acceso será mejorado incrementando el número de medios de conexión al juego y mejorando en cada uno de ellos la accesibilidad de personas con discapacidad. En cuanto a mejorar la experiencia de juego, la incorporación de nuevos contenidos, la mejora de las aplicaciones cliente y la incorporación de nuevas formas de juego son futuras líneas de desarrollo del proyecto.

En primer lugar, sería conveniente desarrollar nuevos medios de conexión al juego. Mientras que la tecnología HTML5 sigue sin ser implementada por la mayoría de navegadores web, existen alternativas multiplataforma como Java o Adobe Flash Player que ya se encuentran en casi todos los dispositivos y que permiten la conexión al juego. Por otro lado, con los recursos necesarios, podrían desarrollarse aplicaciones nativas exclusivas para determinados sistemas operativos y dispositivos, con el objetivo de aprovechar de forma eficaz los recursos de cada sistema y mejorar así la experiencia de uso.

En cuanto a la accesibilidad, aun queda mucho trabajo por hacer por parte de los desarrollares de tecnología para que las nuevas técnicas sean completamente accesibilidad. Desgraciadamente, la falta de normativas de accesibilidad, el incumplimiento de los estándares por parte de los desarrolladores y la escasa economía que este tema genera, empeora la adaptación accesible de los medios de conexión. Se propone, por tanto, un estudio mucho más detallado y exhaustivo de la accesibilidad en cada uno de los medios creados y futuros.

En lo referente a la incorporación de nuevos contenidos, un posible incremento de los contenidos del juego atraería el interés, tanto de jugadores nuevos como antiguos. Para ello, se propone la incorporación de nuevas capas del sistema de información geográfica, que permitan el aumento de parámetros y el consiguiente enriquecimiento del entorno de juego. Se propone, por ejemplo, una capa que represente posibles rutas para transportes públicos, como barcos o zepelines. En esta línea también se propone la creación de nuevos generadores, que atendiendo a los parámetros de las salas, generen nuevos y diferentes contenidos. Por otro lado, la creación de un sistema de logros que permita al jugador establecerse dentro de un ranking de jugadores incrementaría su interés por conseguir reputación entre la comunidad. Finalmente, la implementación de un sistema de creación de *aventuras* que permitan a los jugadores realizar misiones especiales fuera de su rutina habitual de juego.

El uso actual de los clientes de conexión se limita al envío, recepción y visualización de mensajes de texto. Tecnologías heterogéneas como WCF permiten la creación de canales de comunicación auxiliares entre cliente-servidor que favorecen el flujo de información paralelo al del juego, no necesariamente textual. Esto permite la incorporación de nuevas funcionalidades del lado del cliente. Imagínese jugar utilizando una aplicación que muestra el mapa del mundo, su ubicación actual sobre el mapa y la de todos sus amigos. O imagínese poder abrir ventanas de conversación independientes de la ventana de visualización general. Por otro lado, tecnologías como HTML5 permiten la incorporación de elementos multimedia en su propio lenguaje, lo que incrementaría enormemente las alternativas de visualización de la información, y por tanto la accesibilidad del juego.

Finalmente, el estudio de nuevas formas de juego basadas en otros medios de comunicación, que no se centren exclusivamente en la información textual, que permitan al usuario utilizar otras cualidades y características para interactuar con el mundo. Mejorar el uso de otros medios de comunicación, mediante la incorporación de nuevos protocolos y técnicas como la telefonía IP y el reconocimiento de voz.

8 Bibliografía

Buono., S. A. (2005). *C# and game programming : a beginner's guide.* Wellesley, Mass: A.K. Peters.

Cowan, A. (2 de 10 de 2008). *mudconnect.com: The Mud Connector - Mud and RPG game index, mudlist and reviews*. Obtenido de http://www.mudconnect.com/resources/Mud_Resources:Mud_Clients.html

Deveria, A. (7 de 6 de 2011). *When can I use... Support tables for HTML5, CSS3, etc*. Obtenido de http://caniuse.com/#feat=websockets

Faulkner, G. L. (25 de 5 de 2006). *Juicy Studio: Making Ajax Work with Screen Readers*. Obtenido de http://juicystudio.com/article/making-ajax-work-with-screen-readers.php

Fette, I. (13 de 6 de 2011). *draft-ietf-hybi-thewebsocketprotocol-09 - The WebSocket protocol*. Obtenido de http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-09

Fundacion CTIC. (s.f.). *TAW - Servicios de accesibilidad y movilidad web*. Obtenido de http://www.tawdis.net/

García de Sola, M. (2006). *Libro blanco del diseño para todos en la Universidad.* Madrid: Fundación ONCE, Instituto de Mayores y Servicios Sociales.

Geel, I. V. (13 de 8 de 2010). *MMOData.net: News*. Obtenido de http://www.mmodata.net/

Hernández Leal, O. (2008). *C# 3.0 y LINQ : [aprende a sacar todo el partido a la última versión de .NET].* [s.l] : Krasis Consulting.

Hillar, G. C. (2009). *C# 2008 and 2005 threaded programming : beginner's guide : exploit the power of multiple processors for faster, more responsive software.* Birmingham, Mumbai: Packt Publishing.

Makofske, D. B. (2004). *TCP/IP sockets in C# : practical guide for programmers.* Amsterdam ; Boston: Elsevier.

Mellado, J. (26 de 1 de 2011). *HTML5: Web Sockets | inmensia*. Obtenido de http://www.inmensia.com/blog/20110125/html5_web_sockets.html

Microsoft Corporation. (s.f.). *¿Qué es Windows Communication Foundation?* Obtenido de http://msdn.microsoft.com/es-es/library/ms731082(v=vs.90).aspx

NCOUDL. (16 de 3 de 2011). *UDL Examples and Resources | National Center On Universal Design for Learning*. Obtenido de http://www.udlcenter.org/implementation/examples

ONCE. (12 de 2 de 2009). *CIDAT. MEJORAS EN EL USO, PARA PERSONAS CIEGAS Y DEFICIENTES VISUALES, DEL IPOD NANO DE APPLE*. Obtenido de http://cidat.once.es/home.cfm?id=569

Penton, R. (2005). *Beginning c# game programming.* Boston, MA: Course Technology PTR.

Pimentel, V. (31 de 5 de 2007). *Google Gears, la apuesta de Google para acceder a sus aplicaciones desconectado*. Obtenido de http://www.genbeta.com/web/google-gears-la-apuesta-de-google-para-acceder-a-sus-aplicaciones-desconectado

Prieto, O. M. (22 de 6 de 2010). *HTML 5 C# Web Sockets server and ASP.NET client implementation | Undisciplined Bytes*. Obtenido de http://www.undisciplinedbytes.com/2010/06/html-5-c-web-sockets-server-and-asp-net-client-implementation/

Stephanidis, C. (2001). *User Interfaces for All.* Mahwah, EEUU: Lawrence Erlbaum Associates.

Technosite. (2006). *Accesibilidad - diseño para todos - Technosite*. Obtenido de http://www.technosite.es/disenoparatodos.asp

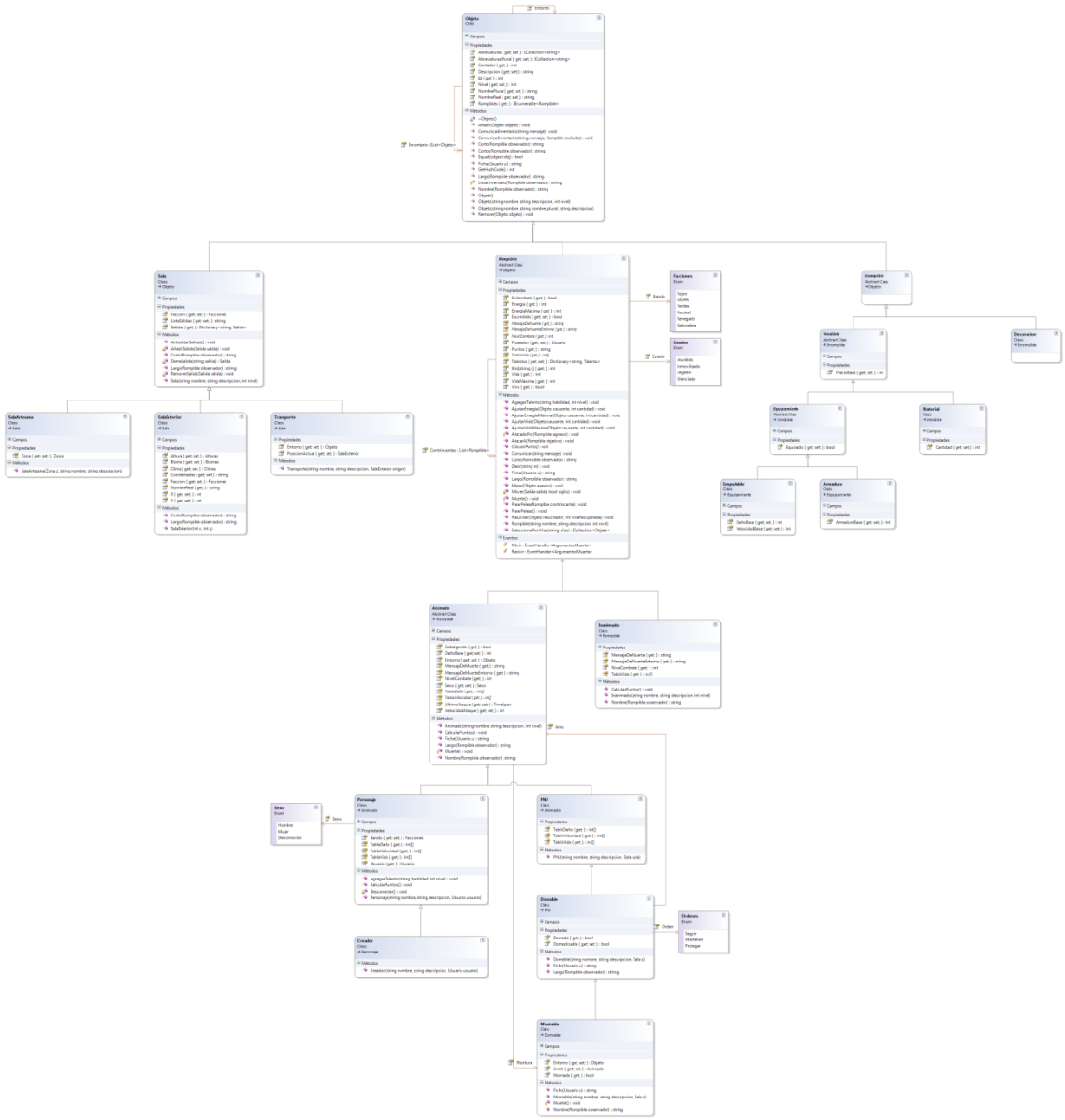
Tyflos. (30 de 5 de 2007). *Java y lectores de pantalla - Artículo de Weblogs de Discapnet*. Obtenido de http://programaraciegas.weblog.discapnet.es/articulo.aspx?idA=208

Tyflos. (23 de 11 de 2007). *Navegar por Internet con Jaws - Artículo de Weblogs de Discapnet*. Obtenido de http://programaraciegas.weblog.discapnet.es/articulo.aspx?idA=367

W3C. (23 de 10 de 2008). *Guía Breve de Accesibilidad Web*. Obtenido de http://www.w3c.es/divulgacion/guiasbreves/accesibilidad

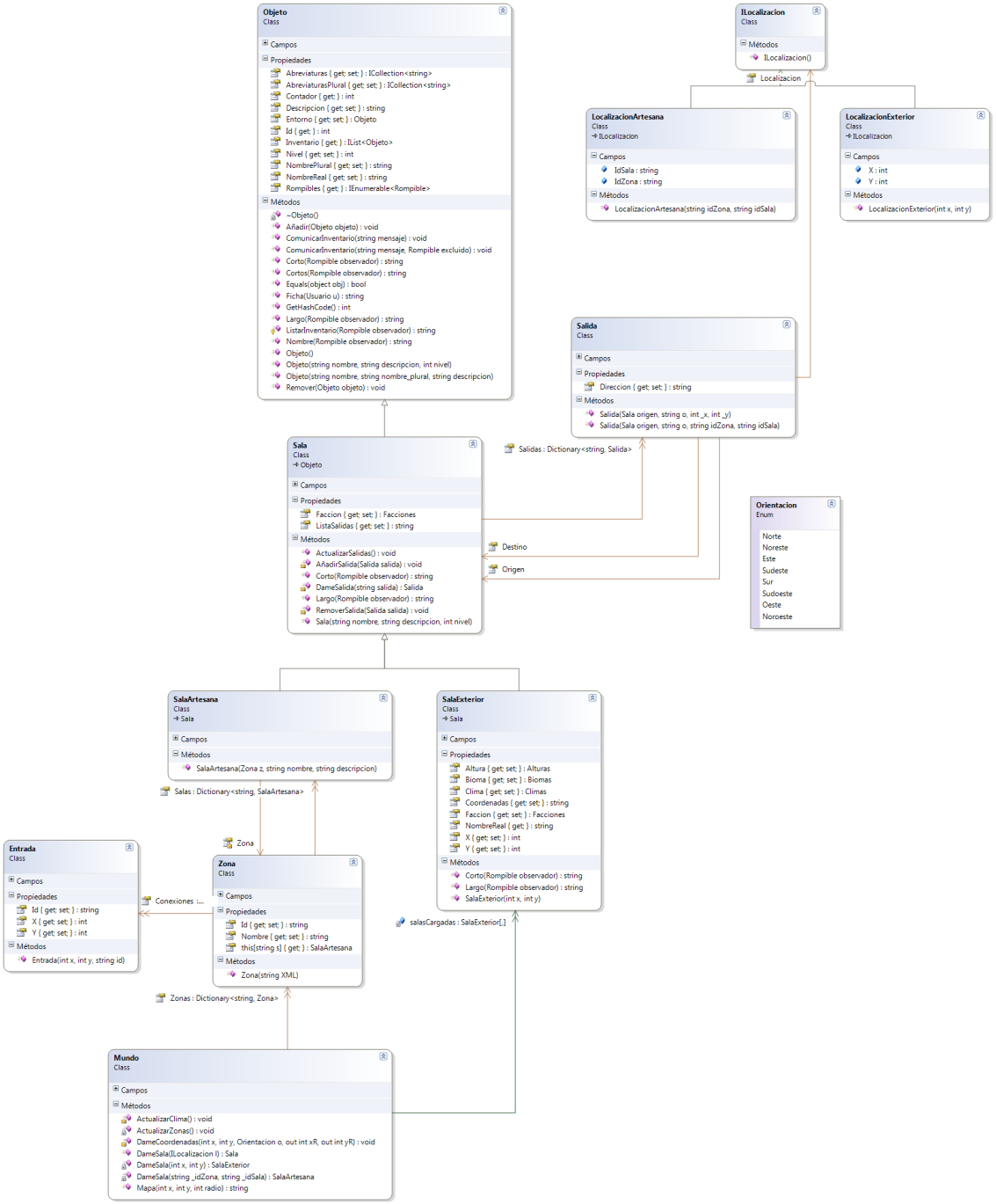
9 Anexos I: Diagrama de clases UML

9.1 Diagrama de entidades



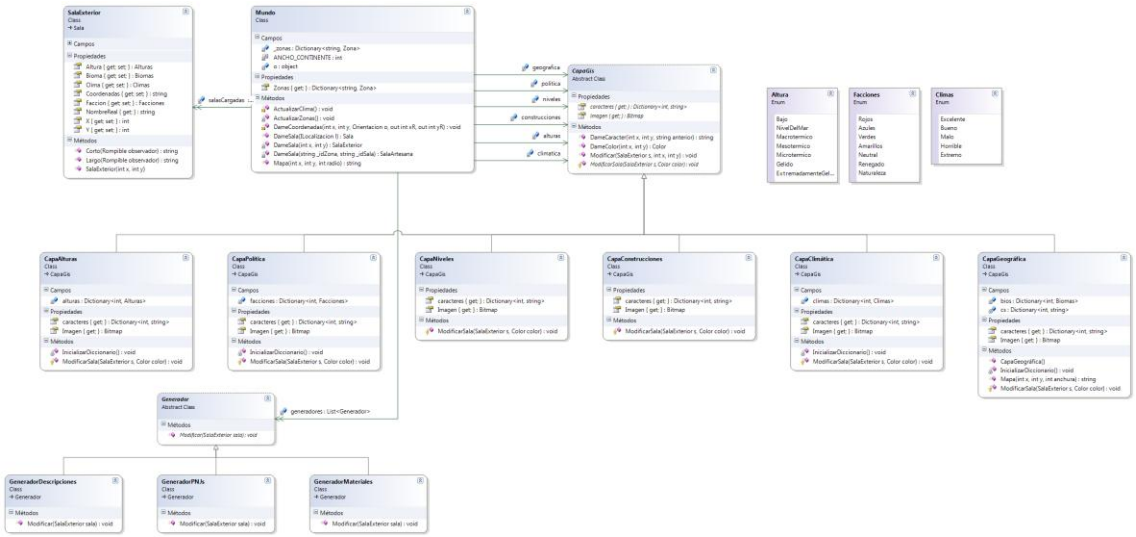
(Pinche para ampliar)

9.2 Diagrama del entorno



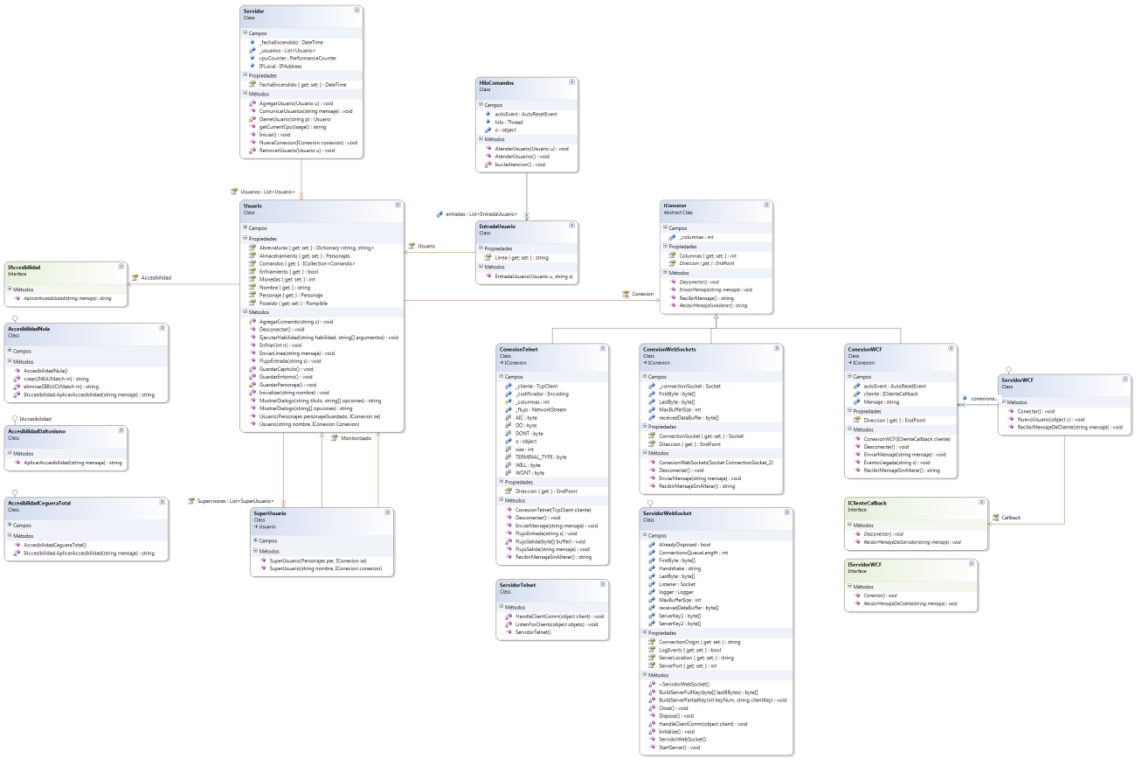
(Pinche para ampliar)

9.3 Diagrama del mundo



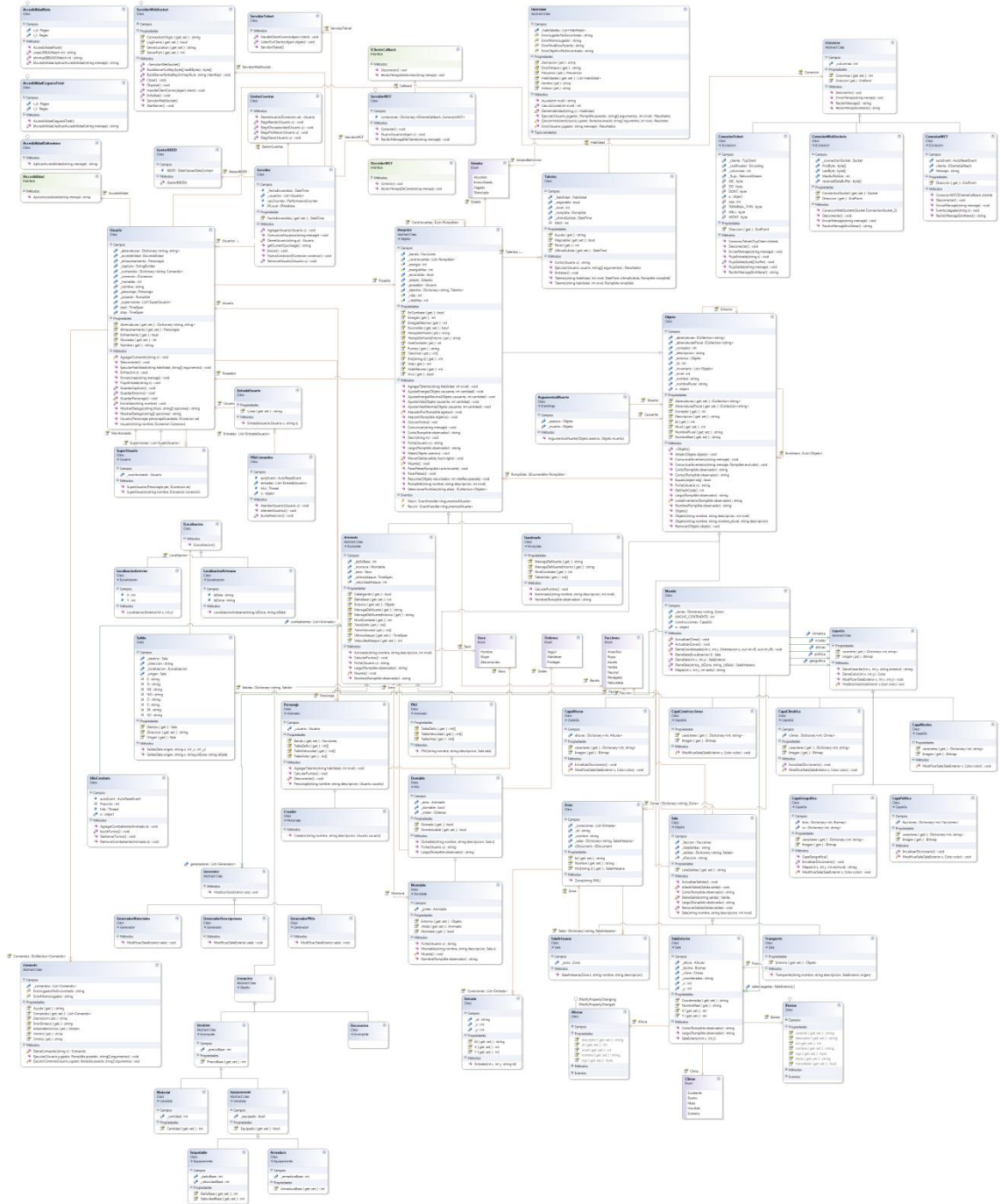
(Pinche para ampliar)

9.4 Diagrama de conectividad



(Pinche para ampliar)

9.5 Diagrama general



(Pinche para ampliar)

10 Anexos II: Diagrama entidad relación

(Pinche para ampliar)

11 Anexos III: Manual de juego

Hay que destacar que existen multitud alternativas para acceder al juego. Este manual pretende servir de guía para cualquier medio de conexión, y por tanto se ha redactado utilizando una visión holística.

En primer lugar, se trata de un juego en red basado en texto. El usuario interactúa con el sistema mediante mensajes de texto que son enviados por la red. Por otro lado, el servidor envía información al usuario mediante información textual. Los mensajes enviados por los usuarios serán denominados *acciones*, mientras que los mensajes enviados por el sistema se denominarán *mensajes*.

Una interfaz típica de juego estará compuesta por un cuadro de texto que permita escribir acciones, un área de texto no editable que visualice las acciones enviadas y los mensajes recibidos, y un botón que desencadene el envió de acciones al servidor. El siguiente diagrama muestra una interfaz genérica de juego:

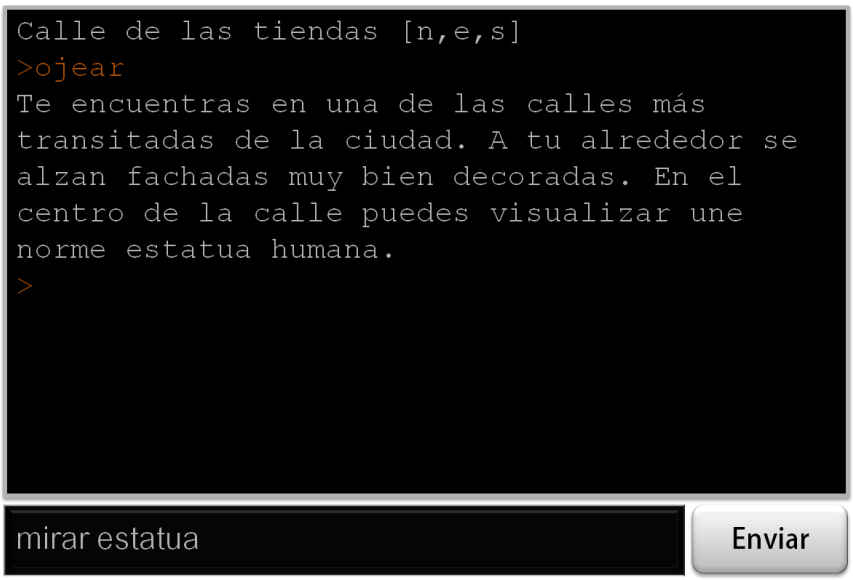


Ilustración 41: Interfaz genérica de juego

Al cuadro de texto sobre el que se escriben las acciones se le denomina *barra de acciones*, mientras que al área de texto que visualiza los mensajes se le denomina *historial*. La mayoría de clientes disponen de una barra de acciones y un historial independientes, pero algunas aplicaciones mezclan ambos elementos sobre un mismo área de texto. Actualmente existen tres tecnologías alternativas de conexión al juego: telnet, WCF y HTML5. Todas ellas disponen de clientes con esta interfaz.

Una vez conectados al juego recibimos nuestro primer menaje:

```
Introduce tu nombre:
```

Este mensaje nos indica que debemos introducir el nombre de nuestro personaje. Suponiendo que nuestro personaje no existe, lo introducimos y el sistema nos responde de la siguiente forma:

```
>Yahan
¡Vamos a crear tu personaje!
Sexo:
 a)Mujer    b)Hombre
```

El juego detecta que nuestro personaje no existe en la base de datos y el jugador comienza un proceso de creación de personajes. Durante este proceso el usuario especifica en cada paso diferentes características del personaje. En cada paso el jugador ha de seleccionar una de las opciones que se muestra en el juego. Para seleccionar una acción el jugador ha de enviar una acción compuesta por una sola letra.

```
>b
Profesión:
 a)Carpintero   b)Agricultor   c)Pastor       d)Picaro
>d
Bando:
 a)Rojos       b)Azules      c)Verdes      d)Amarillos
>a
```

Una vez establecidas las características del personaje, el juego nos pregunta si utilizamos algún lector de pantalla para jugar. Obviamente nuestra respuesta dependerá del nivel de accesibilidad con el que accedamos al juego.

```
¿Utilizas un lector de pantalla para jugar?
 a)Sí   b)No
>b
Introduce tu contraseña:
```

Una vez elegida una opción, el sistema nos pide que introduzcamos una contraseña. Esta contraseña será necesaria para poder volver a utilizar el mismo personaje la próxima vez que queramos jugar.

Tras introducir la contraseña, el juego comienza. Lo primero que podemos observar es un mensaje que nos envía el juego:

```
Bosque 706,200 Salidas: todas
Ardilla está aquí.
```

Este mensaje nos está indicando dónde estamos. Nos encontramos en una sala, un entorno destinado a jugadores y objetos. La sala se llama “Bosque”, y se encuentra en las coordenadas 706,200. El nombre de la sala ya nos da a entender el ambiente en el que nos encontramos. Si queremos saber más información sobre la sala podemos utilizar el comando mirar.

```
>mirar
Bosque 706,200 Salidas: todas.

     Te encuentras en un bosque perteneciente al territorio de los
Rojos. Te sitúas a una altura ligeramente elevada sobre el nivel del
mar, en una zona de nivel 0. El clima aquí es Excelente.


Ardilla está aquí.
```

Cada sala se conecta a otra mediante salidas, y un usuario puede moverse de una sala a otra utilizando esas salidas. Existen ocho tipos de salidas principalmente, una por cada punto cardinal. Para utilizar una salida, un jugador simplemente tiene que enviar una acción con el nombre de la salida. En este caso, la sala dispone de todas las salidas posibles; es decir, se encuentra conectada con ocho salas a su alrededor. Por ejemplo, podríamos movernos dos salas al este utilizando las acciones ‘e’ y ‘e’:

```
>e
Bosque 707,200 Salidas: todas
Ardilla está aquí.
>e
Bosque 708,200 Salidas: todas
Ardilla está aquí.
```

Antes se ha visto que, enviando la acción “mirar”, el juego envía un mensaje con la información sobre el entorno que nos rodea. También hemos escrito dos veces ‘e’ y el sistema ha respondido. Pero, ¿qué ocurre si escribimos algo extraño?

```
 >qwertyu
¿Cómo? Consulta tus 'comandos' y 'habilidades'.
```

El juego no lo reconoce. Esto es así porque el juego solo reconoce tres tipos diferentes de acciones:

- Salidas: como se ha visto antes, las salidas sirven para desplazarse por las salas. Las posibles salidas son: e,ne,n,no,o,so,s,se.
- Comandos: los comandos son acciones que permiten interactuar con el juego a diferentes niveles.
- Habilidades: son acciones que el jugador aprende durante el juego y que puede mejorar para obtener mejores resultados.

Ahora podemos utilizar la acción “comandos” para visualizar nuestros comandos disponibles:

```
>comandos
Comandos:

 mirar        ojear        decir        comandos     ayuda
quien
 salir        susurrar     chat         habilidades  mapa
ficha
 abreviaturas
```

Cualquiera de esas palabras puede ser interpretada por el juego. Para conocer cómo funcionan los comandos o habilidad podemos utilizar la acción “ayuda <comando>” y obtener información del sistema:

```
>ayuda mirar
mirar

      Obtiene la descripción extendida del elemento

Sintaxis: mirar
```

Además de los comandos existen habilidades. Las habilidades son formas de interactuar con el juego que se adquieren durante el juego. Cada habilidad tiene asociado un nivel. El nivel

representa la experiencia que el personaje tiene a esa habilidad; que como máximo podrá ser 100. Con el uso repetido de las habilidades el jugador puede incrementar el nivel de las habilidades. Podemos utilizar el comando “habilidades” para visualizar nuestras habilidades disponibles:

```
>habilidades
Habilidades:

 cabalgar   [                    ] (3,100)
 rastrear   [#                   ] (7,100)
 combatir   [#                   ] (8,100)
 esconderse [##                  ](12,100)
```

Una vez más podemos utilizar el comando “ayuda” para conocer más información sobre alguna habilidad:

```
>ayuda esconderse
esconderse


     Permite esconderse en el entorno y para no ser visto. Cualquier
movimiento hará que pierdas tu situación ventajosa.

Sintaxis: esconderse
Coste: 2
```

Para utilizar una habilidad simplemente debemos enviar una acción con su nombre:

```
>esconderse
Te escondes en tu entorno para que nadie te vea.
¡Has mejorado tu habilidad en esconderse!
```

Por último, para comunicarte con el resto de usuarios el jugador dispone de varios comandos. Uno de ellos es el comando “chat”, que permite enviar mensajes al resto de jugadores. De nuevo podemos consultar su funcionamiento con la acción “ayuda chat”:

```
>ayuda chat
chat
```

```
     Envia un mensaje a todos los usuarios

Sintaxis: chat <mensaje>
```

Bien, ahora podemos utilizar la acción “chat ¡Hola!” para enviar un mensaje de saludo a todos los usuarios del juego:

```
>chat ¡Hola!
[Chat] Yahan: ¡Hola!
```

El manual acaba aquí. A partir de este punto el usuario deberá explorar sus comandos y acciones para desenvolverse por el mundo. Se recomienda que, en caso de duda, se utilice alguno de los comandos de comunicación para pedir ayuda a alguno de los usuarios de la comunidad.