FACULTAD 2

PLATAFORMA PARA JUEGOS MULTIJUGADOR POR TURNOS DESTINADA A DISPOSITIVOS MÓVILES ANDROID

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

AUTORES:

YENIEL CASTELLANOS DÍAZ

RONALD RODRIGUEZ MARTÍN

TUTOR:

ING. DAMIAN ILIZASTEGUI ARRIBA

LA HABANA, JUNIO DEL 2017

"AÑO 59 DE LA REVOLUCIÓN"

Pensamiento

"Nunca consideres el estudio como una obligación, sino como la oportunidad para penetrar el bello y maravilloso mundo del saber"

Albert Einstein

Declaración de autoría

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ___ días del mes de junio del año 2017.

_____ _____

Yeniel Castellanos Díaz (Autor) Ronald Rodriguez Martín (Autor)

_____

Ing. Damian Ilizastegui Arriba (Tutor)

Datos de contacto

Autor:

Yeniel Castellanos Díaz

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: ycastellanos@estudiantes.uci.cu

Autor:

Ronald Rodriguez Martín

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: rrmartin@estudiantes.uci.cu

Tutor:

Ing. Damian Ilizastegui Arriba

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: dilizastegui@uci.cu

Dedicatoria de:

Yeniel Castellanos Díaz

A mis abuelas Dalia y Emelia. Este fue el sueño de sus últimos 24 años de vida. Gracias por no dejarme solo ni un momento de mi vida, pensar en ustedes y poder dedicarles esto, es lo que me dio fuerzas para seguir adelante.

Dedicatoria de:

Ronald Rodriguez Martín

A mis padres Ronald y María Elena por no dejarme solo en ningún momento de mi vida y por su constante apoyo incondicional ya que sin ustedes no habría tenido la motivación necesaria para llegar tan lejos en la vida, les dedico esto y muchas gracias por todo.

Agradecimientos de:

Yeniel Castellanos Díaz

Quiero agradecer en primer lugar a mi mamá por su apoyo incondicional.

A mi papá que desde la distancia me supo ayudar y darme consejos que recordaré siempre, así como exigir el máximo de mí.

No podía pasar por alto la oportunidad de agradecerle a mis hermanos en especial a Yosiel que sin su apoyo tecnológico no podría haber salido adelante.

A mi tía Diana que es casi como si fuera mi madre, por siempre estar ahí a mi lado cada vez que lo he necesitado.

A mi abuelo Ernesto, mi ejemplo a seguir.

A Baby, mi novia, mi amiga, mi confidente, la persona que supo estar a mi lado en los momentos más difíciles de mi vida y no se rindió. La que soportó mi mal carácter debido al estrés que trae consigo hacer una tesis. Me ayudó muchísimo con mis problemas de concordancia a la hora de escribir la tesis y por muchas razones más que si las digo tendría que escribir otra tesis.

A mis amigos de la universidad, esos que me han ayudado en estos años, muchos están, otros ya se fueron, pero a todos quiero que les llegue mis agradecimientos. En especial a Alberto mi compañero de apartamento y amigo, a Yenlis, Yanira, Jossue, Bárbaro y a Ronald mi compañero de tesis.

A mi tutor Damian, que un poco más y lo vuelvo loco de tantas veces que lo molestaba.

A todos los profesores que tuve durante estos cinco años de carrera porque fueron los que me ayudaron a formarme profesionalmente, en especial a Madelín Haro, Lizandro, Osmar y Sergio Reyes, gracias por todo.

Agradecimientos de:

Ronald Rodriguez Martín

Quiero agradecer nuevamente a mis padres por servirme de práctica y por apoyarme en cada momento y cada paso de la realización de esta tesis.

A mi abuela María del Carmen que sin sus constantes críticas no hubiera sabido capaz de diferenciar lo que hacía mal o bien, gracias por abrirme los ojos cuando yo fui demasiado arrogante para no hacerlo.

A mi hermana Giselle por prestarme sus conocimientos y experiencias de la vida tanto como estudiante y como tesista.

A todos mis amigos del barrio que un poco más y no hay necesidad de repasar más la exposición pues cada vez que veía alguno me preguntaba de qué iba la tesis.

A mis amigos de la universidad, en especial a mi compañero de cuarto Bárbaro, que gracias a sus conocimientos pude seguir adelante tanto en mi tesis como en mi carrera.

A mi compañero de tesis Yeniel, que en todo el curso ya le salgo hasta en la sopa.

A mi tutor Damian por su apoyo incondicional tanto en el documento como en el desarrollo de la plataforma y por estar presente cuando más lo necesitábamos.

A todo el grupo de profesores que tuve durante mi vida como estudiante universitario, muchas gracias a todos

Resumen

Uno de los tipos de juegos para móviles más populares son aquellos en los que, en la misma partida, participan dos o más oponentes de forma simultánea; también conocidos como juegos multijugador. Actualmente el centro TLM no cuenta con un mecanismo mediante el cual diferentes clientes móviles puedan jugar simultáneamente independientemente de donde estén localizados físicamente. El presente trabajo presenta la investigación, diseño e implementación de una plataforma de juegos que permite a diferentes dispositivos móviles Android interactuar a través de juegos independientemente de donde estén localizados físicamente.

Palabras clave: juegos multijugador, juegos móviles, Android, plataforma de juegos.

Índice

Introducción

El juego es una actividad tan antigua como el hombre mismo, aunque su concepto, y su forma de practicarlo varía según la cultura de los pueblos. El ser humano lo realiza de forma innata, debido a una experiencia placentera como resultado de un compromiso en particular, es un estímulo valioso mediante el cual el individuo se vuelve más hábil, perspicaz, ligero, diestro, fuerte y sobre todo alegre, así lo definen Lacayo y Coello(1992), donde también consideran que los niños aprenden a crecer en una forma recreativa. (Montero y Alvarado 2001)

Con el uso de las Tecnologías de la Información y las Comunicaciones (TIC) se han desarrollado varias aplicaciones conocidas como videojuegos que tributan a la diversión y al aprendizaje de las personas

Los videojuegos son programas de ordenador que, conectados a una pantalla de computadora o de televisión, integran un sistema de video y audio. A través de ese sistema el usuario puede vivir experiencias y disfrutar de actividades que en la realidad no practicaría. Estos videojuegos comenzaron a extenderse de forma imparable a partir de la década de los ochenta, cuando hicieron un espacio en muchos hogares y generaron un mercado que ejerce una gran presión económica. (Abreu et al. 2011)

En la actualidad, el desarrollo de las tecnologías móviles ha permitido un incremento de la capacidad de cómputo en los dispositivos, surgiendo de esta forma, un nuevo medio para los videojuegos.

Según un estudio del centro de investigaciones de inteligencia de negocio para el mercado de los juegos Superdataresearch (Superdatasearch 2017b) en el año 2016: “Los consumidores de juegos gastaron 41 mil millodes de dólares en juegos móviles, conducidos por los títulos Pokémon GO y Crash Royale. El mercado de los juegos móviles ha empezado a madurar y actualmente se parece más al mercado tradicional, requiriendo una producción cada vez mayor de valor y marketing”. (Superdatasearch 2017a)

Uno de los tipos de videojuegos más populares son aquellos en los que, en una partida interactúan varios jugadores ya sea compitiendo o colaborando entre ellos. Para este tipo de videojuego se hace necesario diseñar un mecanismo de comunicación que permita la interacción entre los jugadores. Según Smed y Hakonen (Smed y Hakonen 2006) existen cuatro principales arquitecturas de comunicación para los juegos multijugador:

a) Nodo único (Single Node): La arquitectura Nodo único, tiene la implementación más simple. Como su nombre indica, consta de un solo nodo en el cuál los jugadores interactúan sin necesidad de red.
b) Punto a punto (Peer-to-Peer): Esta arquitectura contiene un conjunto de nodos conectados por una red. Estos nodos no usan ningún intermediario para su comunicación y cada uno de ellos puede transmitir mensajes a cualquier otro conectado a la red.
c) Cliente-Servidor (Client-Server): En esta arquitectura un nodo es promovido al rol de servidor. Todas las comunicaciones son manejadas por este nodo servidor, mientras que los otros nodos se mantienen con sus roles de cliente, comunicándose entre ellos a través del servidor.
d) Red de servidores (Server-Network): En esta arquitectura los servidores están interconectados entre sí. La comunicación puede pensarse como una red Punto a punto entre los servidores que controlan un conjunto de clientes. Cada cliente se conecta a un servidor local, este a su vez está conectado a otros servidores locales posibilitando la comunicación entre todos los clientes.

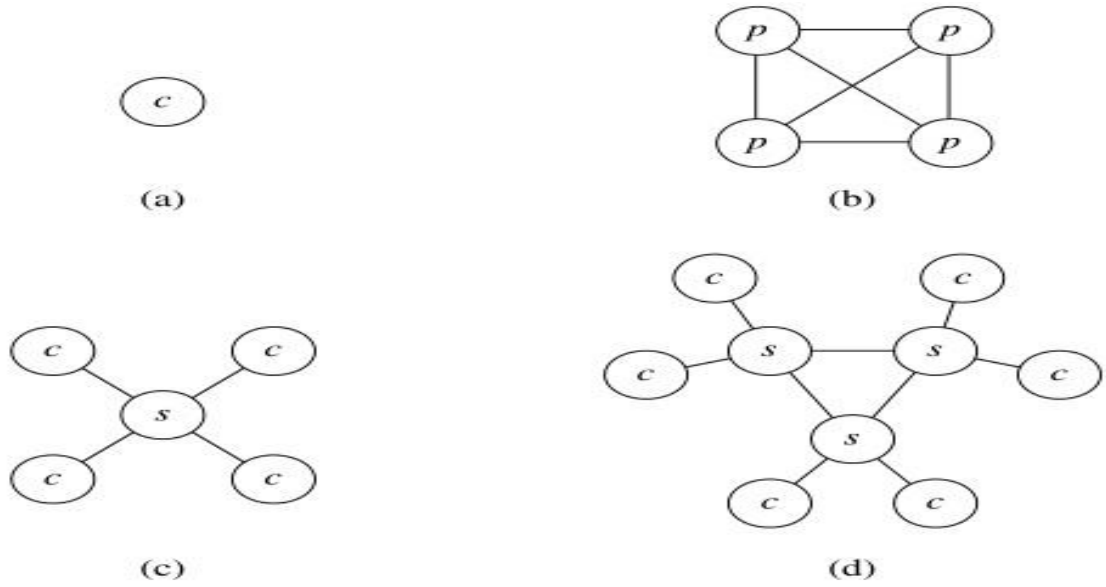

Figura 1: Esquemas de la arquitectura de comunicación (a) Nodo único, (b) Punto a punto, (c) Cliente – servidor, (d) Red de servidores.

Para los juegos móviles el uso de la arquitectura Nodo único es muy poco usado. Una de los mecanismos más usado es dividir la pantalla del nodo creando un espacio en ella asignado a cada jugador, en el caso de los móviles esto es imposible debido al pequeño tamaño de las pantallas. La forma usual de implementar esta arquitectura en el mundo móvil es en juegos por turnos dónde los jugadores deben pasarse el dispositivo entre sí según a quien le corresponda el turno, en este escenario el juego

pierde dinamismo, no acapara la atención del jugador totalmente y se torna confuso en caso de que intervengan más de dos jugadores.

En el caso de la arquitectura Punto a punto para los móviles se utilizan las tecnologías de comunicación Wi-Fi y Bluetooth. Estas tecnologías tienen un rango de acción pequeño, aproximadamente 10 y 5 metros respectivamente, lo que obliga a los jugadores a estar físicamente cercanos. En el caso de los juegos que requieran un servidor central uno de los dispositivos deberá asumir este rol y esto sobrecarga el dispositivo al tener que ejecutar la lógica de los clientes así como la lógica del servidor. La mayoría de los dispositivos móviles no poseen una capacidad de procesamiento que les permita desarrollar estas tareas de forma simultánea, sin que el rendimiento se vea afectado. Este nivel de procesamiento puede drenar de forma más rápida la carga de la batería en los móviles, limitando su vida útil.

El uso de la arquitectura Red de servidores, sería el mas idóneo para los juegos multijugador, pues ofrece una gran cantidad de conexiones concurrentes y alta escalabilidad. La desventaja principal de esta arquitectura es su alto costo ya que se necesita comprar y mantener un conjunto de servidores. Esta arquitectura es usualmente reservada a grandes empresas con suficientes recursos.

La Universidad de las Ciencias Informáticas (UCI) forma parte de la estrategia de informatización del país. El centro TLM que pertenece a la Facultad 2 de la UCI cuenta con el departamento Desarrollo de Aplicaciones donde una de sus líneas es el desarrollo de juegos y aplicaciones para celulares. En este centro se han desarrollado varios juegos multijugador para dispositivos móviles Android. Todos los juegos multijugador desarrollados en este centro utilizan la arquitectura punto a punto, por lo que tienen las desventajas y limitaciones mencionadas anteriormente. En aras de ampliar la cartera de productos a comercializar en este departamento y resolver los problemas de comunicación de los juegos que se desarrollan se hace necesaria la creación de un mecanismo mediante el cual diferentes clientes móviles puedan jugar de forma simultánea independientemente de donde estén localizados físicamente.

A partir de lo planteado anteriormente se define como **problema de la investigación**: ¿Cómo lograr que diferentes clientes móviles puedan interactuar a través de juegos independientemente de donde estén localizados físicamente?

Se obtiene del problema definido el **objeto de estudio**: Plataforma de juegos para dispositivos móviles.

Con aras de dar cumplimiento al problema planteado se entiende como **objetivo general**: La creación de una plataforma de juegos multijugador por turnos que provea

un servidor de juegos, así como un conjunto de librerías de comunicación para Android y un protocolo de comunicación entre el servidor y los móviles.

Se describe como **campo de acción**: Plataforma de juegos multijugador por turnos para Android.

Para dar cumplimiento al objetivo general se precisaron las siguientes **tareas de investigación**:

- Estudio de los antecedentes de los productos con las funcionalidades de la plataforma a desarrollar.
- Selección de las diferentes herramientas y metodologías que faciliten el desarrollo.
- Análisis de la solución propuesta para guiar el proceso de desarrollo de la misma.
- Diseño de la solución propuesta para guiar el proceso de desarrollo de la misma.
- Implementación de las funcionalidades necesarias para dar solución a la problemática planteada.
- Ejecución de pruebas a la solución desarrollada para comprobar su correcto funcionamiento.

Métodos Científicos

En calidad de **Métodos Teóricos** se utilizaron los métodos:

Histórico-lógico

Este método es un procedimiento por el cual se analizan diversos hechos o fenómenos de manera secuencial, aludiendo a la aparición y desarrollo de los factores presentes. Se utiliza con el objetivo de definir las tendencias actuales para el trabajo con interfaces de usuario. (Villafuerte [sin fecha])

Análisis y Síntesis

Este método será utilizado en el estudio y análisis de bibliografías relacionadas con el argumento para ocupar algunas posiciones teóricas vinculadas al objetivo de la investigación.

Modelación

Se utiliza para la modelación de diagramas, representar el proceso de desarrollo y propiciar un mejor entendimiento de la solución a implementar.

Los **Métodos Empíricos** empleados fueron:

Observación

Este método será utilizado para poder percibir los cambios y características que se presenten durante el proceso de desarrollo y así definir cuáles son las principales deficiencias desde el punto de vista visual de la aplicación.

Estructura de la investigación

Capítulo 1: Fundamentación Teórica: En este capítulo se hará un análisis haciendo referencia a los elementos teóricos que fundamentan la propuesta y el desarrollo de la plataforma de juego. De esta forma se complementa un estudio de los antecedentes donde se establecen los conceptos fundamentales a seguir. Además, se realiza una caracterización de los principales servidores de juegos multijugador en el mercado actual. Luego se examinan las posibles metodologías a usar, y finalmente, se definen las herramientas y tecnologías que serán utilizadas en el desarrollo de dicha plataforma.

Capítulo 2: Propuesta de Solución. Exploración y planificación: En este capítulo se estará realizando una valoración de las primeras dos fases del ciclo de vida de la metodología XP, exploración y planificación. Se establecen las funcionalidades del sistema, las historias de usuarios que servirán para un mejor entendimiento y conocimiento del software. Estas historias de usuarios son escritas por el cliente representando en las principales necesidades del sistema. Además, se plantearán los requisitos no funcionales del sistema definidos por el cliente.

Capítulo 3: Diseño e Implementación: En este capítulo se realiza una valoración de la fase de Diseño e Implementación de la metodología XP abordadas anteriormente, se describen las tarjetas CRC (Contenido, Responsabilidad y Colaboración) para un mejor entendimiento del sistema. Además, se exponen las tareas de programación o ingeniería generadas por cada historia de usuario.

Capítulo 4: Protocolo de comunicación: En este capítulo se describe el protocolo de comunicación y el flujo de información entre el cliente y el servidor de la plataforma a desarrollar en este proyecto.

Capítulo 1: Fundamentación Teórica

1.1 Conceptos fundamentales

Juegos online

Son aquellos videojuegos en los que se juega mediante una conexión a la red de Internet, bien mediante una PC o mediante una videoconsola que permita dicho acceso. Aunque algunos se pueden jugar en solitario, su atractivo principal reside en la posibilidad de compartir escenario con un gran número de personas gracias a una conexión a Internet.

Servidor de juego multijugador

Son plataformas diseñadas para alojar videojuegos multijugador en línea donde una de sus funciones es controlar el tráfico de información tanto del cliente como del servidor, se les da soporte a estos juegos, además de manejar las cuentas de usuario y las estadísticas por juego.

API

Una API es una Interfaz de Programación de Aplicaciones (Application Programming Interface) es decir, un conjunto de funciones que facilitan el intercambio de mensajes o datos entre dos aplicaciones. Permite que dos aplicaciones que trabajan al mismo tiempo, como podría ser un procesador de texto y una hoja de cálculo, se comuniquen e intercambien datos.

En Internet, una API permite que un sitio web brinde determinado servicio a otro, a través de llamadas a funciones documentadas y publicadas, facilitando de esta manera el “mash-up” o mezcla de servicios.

Por ejemplo, hoy es posible que desde un blog personal se puedan publicar noticias del sitio de un famoso periódico, mezcladas con fotos que ya están alojadas en un sitio de fotografías, a través de llamadas a la API de estos dos servicios.

Una API detalla solamente la forma de llamar a cada función y la tarea que esta desempeña, sin importar cómo se lleva a cabo esta tarea.

Las API abren distintos tipos de diálogos con el proveedor para obtener o actualizar información en el mismo, entre ellos:

- Acceso a bases de datos
- Comunicación cliente/servidor
- Comunicación peer-to-peer

- Comunicación en tiempo real
- Event-driven (orientada a eventos)
- Store and forward
- Procesamiento de transacciones

Una API puede combinar recuperación de errores, traducción de datos, seguridad, manejo de colas y nomenclatura con una interfaz fácil de asimilar, que comprende acciones y comandos simples, pero con muchas opciones.

Para invocar una API, el programa debe llamar a una función tipo “send”, especificando parámetros para el nombre de destino, indicadores de datos y opciones de confirmación. (Alejandra 2007)

Websocket

Es un protocolo que permite una comunicación en ambos sentidos entre un cliente que ejecuta un código de forma no confiable en un ambiente controlado, a un host remoto que ha optado por comunicaciones de ese código.

El objetivo de esta tecnología es proveer un mecanismo para aplicaciones que carecen de comunicación en ambos sentidos con servidores que usan múltiples conexiones abiertas HTTP. Este protocolo está diseñado para reemplazar las tecnologías de comunicación bidireccionales existentes que utilizan HTTP como capa de transporte para beneficiarse de la infraestructura existente (proxies, filtrado, autenticación).

El protocolo WebSocket intenta abordar los objetivos de las tecnologías HTTP bidireccionales existentes en el contexto de la infraestructura HTTP. Está diseñado para trabajar a través de los puertos HTTP 80 y 443, así como para soportar proxies e intermediarios HTTP, aunque esto implique cierta complejidad específica del entorno actual.

Sin embargo, el diseño no limita al WebSocket con HTTP, y en futuras implementaciones podrían utilizar un apretón de manos (handshake) más simple sobre un puerto dedicado sin necesidad de reinventar todo el protocolo. Este último punto es importante porque los patrones de tráfico de la mensajería interactiva no coinciden estrechamente con el tráfico HTTP estándar y pueden inducir cargas inusuales en algunos componentes.

Una vez que el cliente y el servidor han enviado sus apretones de manos, y si este fue exitoso, entonces la parte de transferencia de datos se inicia. Luego, en un canal de

comunicación bidireccional donde cada lado puede, independientemente del otro, enviar datos a voluntad.

Después de un apretón de manos exitoso, los clientes y los servidores transfieren datos de ida y vuelta en unidades conceptuales referidas en esta especificación como "mensajes". En el cable, un mensaje está compuesto de una o más tramas.

El mensaje del WebSocket no corresponde necesariamente a un enmarcado de capa de red particular, ya que un mensaje fragmentado puede ser fusionado o dividido por un intermediario. Un marco tiene un tipo asociado.

Cada trama perteneciente al mismo mensaje conteniendo el mismo tipo de datos. En términos generales, existen tipos de datos textuales, datos binarios y cuadros de control (que no están destinados a transportar datos para la aplicación, sino para señalización del nivel de protocolo, así como para indicar que la conexión debe ser cerrada). (Fette 2011)

1.2 Servidores de juegos multijugador en el mercado actual

A continuación, se estudian las plataformas y tecnologías ya existentes en el mercado que se ajustan a las necesidades del sistema de juegos multijugador por turnos para dispositivos Android.

SmartFox Server

SmartFox Server es un servidor desarrollado en Java que permite la implementación de servidores de videojuegos mediante el uso de extensiones. Estas permiten gestionar la información que se enviará entre el servidor y los clientes.

Además de estar desarrollado en Java, se despliega sobre un servidor Tomcat para poder acceder vía web al apartado de administración.

La conexión entre el servidor y el cliente se realiza mediante HTTP intercambiándose mensajes JSON que es un formato ligero para la comunicación entre ambos.

Este servidor ha sido desarrollado para poder interactuar con diferentes plataformas que lo hace muy versátil. El uso de este servidor simplifica el proceso de desarrollo por lo que es cómodo de cara al programador. (SmartFoxServer 2017)

Características

- Multiplataforma: al estar desarrollado sobre Java puede correr sobre cualquier Sistema Operativo.

- Configuración visual y administración: se realiza a través de una herramienta administrativa que proporciona funciones avanzadas de estadísticas en tiempo real, gestión de Baneo, entre otras.

Ventajas

- Relativamente sencillo y rápido de integrar.
- Paneles de administración fáciles de utilizar.
- Gran cantidad de información sobre el estado del servidor en tiempo real.
- La compañía actualiza activamente el servidor.
- Resuelve eficientemente los aspectos relacionados con las gestiones de zonas, salas, presencia de usuarios, etc.
- Proporciona un sistema de autenticación y gestión de usuarios extensible.

Desventajas

- No almacena mensajes para usuarios desconectados (requeriría programar extensiones).
- Tiene costo económico.
- No se puede acceder al código fuente ni modificar aspectos no contemplados en los mecanismos de extensión.
- Conexiones con poca estabilidad en dispositivos Android.
- Al final requiere un esfuerzo de programación en relación con las extensiones.

SmartFoxServer puede ser un servidor funcional porque solucionaría la problemática antes planteada. Sin embargo, la plataforma presenta dos importantes aspectos que hace que sea poco propicia para usarla como base en la construcción de la plataforma a desarrollar: tiene costo económico y presenta conexiones poco estables en dispositivos Android.

ES5 ElectroServer

ElectroServer ES5 es la nueva plataforma de juegos desarrollada por Electrotank que está diseñada para soportar juegos multijugador desarrollados en Flash, Unity3D, iOS y Android. Cuenta con una nueva API la cual permite múltiples conexiones de usuarios desde diversas plataformas simultáneamente, además de gran escalabilidad y rendimiento sobre todo para los proyectos multiplataforma.

Ventajas

- Soporte para juegos desarrollados en Flash, Unity3D, iPhone / iPad y Android.
- Licencias de usuario simultáneas ilimitadas.
- Rendimiento sólido de roca a escala.
- Tecnología anti-hacking.
- Soporte UDP para altas tasas de paquetes.
- Protocolo binario de segunda generación.
- Seguridad mejorada y de múltiples niveles.
- Elegante interfaz de administración de ElectroServer.

Desventajas

Hay tres planes para elegir de acuerdo con la cantidad de usuarios concurrentes (UC) ya que tiene una licencia de pago. (Electrotank 2017)

Precio de licencia (por cantidad de usuarios concurrentes)

- ElectroServer5 - 1-25 UC - Gratis
- ElectroServer5 - 26-1.000 UC - $ 999
- ElectroServer5 - Ilimitados UC - $ 4,999

ElectroServer ES5 es una plataforma que responde a las necesidades del negocio y funcionaría perfectamente con todas las plataformas móviles, pero cuando se conectan más de 25 usuarios simultáneamente a la plataforma hay que pagar una licencia que puede ascender hasta los $ 4,999 por lo cual se decidió no utilizarlo en este proyecto.

Google Play Game Services

Es un sistema creado por Google que facilita y agiliza el desarrollo de juegos proporcionando una API con el que manejar ciertas funcionalidades muy frecuentes en el desarrollo de este tipo de aplicaciones como son:

- Logros.
- Rankings.
- Almacenamiento de datos en la nube.
- Comunicación multijugador en tiempo real.

Google proporciona esta API mediante librerías para Android, iOS y Web. Todas las funcionalidades están disponibles para las tres plataformas excepto el módulo para multijugador en tiempo real, que solo está disponible para Android.

El sistema es gratuito siempre y cuando no se superen los 50 millones de peticiones a la API diaria, límite a partir del cual habría que negociar con Google el coste (del cual no se han encontrado precios orientativos, aunque tomando como referencias otras APIs similares de Google los precios una vez superados los límites gratuitos de cortesía suelen ser muy elevados).

Proporciona una API para crear de una forma sencilla una partida en la que varios usuarios jueguen de forma simultánea. La librería de Google gestiona automáticamente todas las conexiones de red y se encarga de sus aspectos a bajo nivel. (Google Play Game Services 2017)

Ventajas

- Muy sencillo y rápido de integrar.
- El sistema escala de forma automática.
- Gratuito si el juego no se masifica.
- Desarrollado por Google.
- Adoptado por numerosas aplicaciones.

Desventajas

- Personalización inexistente.
- No permite el acceso a los datos de otros usuarios.
- Solo cliente: No permite el acceso y modificación desde el servidor.
- No es extensible.

Estas desventajas junto con la nula personalización, la necesidad de que los usuarios accedan al servidor con su cuenta de Google de manera obligatoria además de su extremada rigidez y la falta de extensibilidad indican que el sistema está pensado para simplificar el desarrollo de juegos relativamente sencillos cuyos requisitos se ajusten a lo que la plataforma ofrece y por tanto, aunque ha ayudado en gran medida en la recolección de los requisitos que la plataforma final debería reunir, no cumple con todos los del sistema a desarrollar.

Resultados de los antecedentes

Ninguna alternativa considerada ha compensado las necesidades planteadas. En este caso se hizo necesario establecer una solución totalmente personalizada que se ajuste de forma flexible a los requisitos de los juegos que se desarrollaron sobre el sistema.

1.3 Metodologías de Desarrollo

Se hace uso de la metodología ágil: Extreme Programming (XP).

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. (Torres y López 2003)

Se optó por usar la metodología XP debido a las siguientes características del proyecto:

- El cliente tiene una comunicación sistemática y personal con los programadores, lo que potencializa las relaciones interpersonales siguiendo la idea de tener un producto exitoso y a corto plazo.
- El equipo de desarrollo se compone por dos integrantes. Para dar cumplimiento a cada una de sus iteraciones se definen historias de usuario.
- Al ser un proyecto pequeño y con un corto tiempo de entrega, su proceso de desarrollo se orienta más hacia el desarrollo que la documentación.

1.4 Tecnologías y herramientas

Entorno de desarrollo integrado (IDE): Netbeans v8.1

Netbeans es un entorno de desarrollo gratuito y de código abierto que permite el uso de un amplio rango de tecnologías tanto para escritorio, aplicaciones Web, como para dispositivos móviles. Da soporte a las siguientes tecnologías: Java, PHP, Groovy, C/C++, HTML5. Además, puede instalarse en varios sistemas operativos: Windows, Linux, Mac OS. La Plataforma NetBeans es un framework de Java en el que puede basar grandes aplicaciones de escritorio. NetBeans IDE es uno de los cientos de aplicaciones basadas en la plataforma NetBeans.

La plataforma NetBeans contiene una API que simplifica el manejo de ventanas, acciones, archivos y muchas otras cosas típicas de las aplicaciones. Cada característica distinta de una aplicación de la plataforma NetBeans puede proporcionarse mediante un módulo distinto que es comparable a un complemento.

Un módulo de NetBeans es un grupo de clases Java que proporcionan una aplicación con una característica específica. También puede crear nuevos módulos para NetBeans IDE. Por ejemplo, puede escribir módulos que hacen que sus tecnologías de vanguardia favoritas estén disponibles para los usuarios de NetBeans IDE. Alternativamente, puede crear un módulo para proporcionar una función de editor adicional. (Netbeans 2017)

Apache Tomcat v8.0.27

Tomcat es un contenedor web con soporte de servlets y JSPs. Es desarrollado y actualizado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria. Puede funcionar como servidor web por sí mismo. Debido a que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java. (Apache Tomcat 2017)

Lenguaje de programación: Java 8

Java es un lenguaje de programación de propósito general. En la actualidad su uso está muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Es creado por la compañía Sun Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras.

El principal motivo para la utilización de java como lenguaje de programación es que Android tiene a Java como lenguaje base. Además, es un lenguaje multiplataforma, lo que permite su ejecución en diferentes sistemas operativos. Está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. (Rodríguez [sin fecha])

Plataforma de desarrollo

Java Enterprise Edition (JEE): el objetivo de la plataforma JEE es proporcionar a los desarrolladores con un potente conjunto de APIs, reducir el tiempo de desarrollo y la complejidad y mejorar el rendimiento de la aplicación.

La plataforma JEE utiliza un modelo de programación simplificado. Los descriptores de despliegue XML son opcionales. En su lugar, un desarrollador puede simplemente introducir la información como una anotación directamente en un archivo fuente de Java y el servidor JEE configurará el componente de despliegue y el tiempo de ejecución. Con anotaciones, la especificación de la información se coloca directamente

en el código del programa que lo afecta. En la plataforma JEE, la inyección de dependencia se puede aplicar a todos los recursos que un componente necesita, ocultando la creación y búsqueda de recursos. La inyección de dependencia permite que el contenedor JEE inserte automáticamente referencias a otros componentes o recursos requeridos utilizando anotaciones. (Jendrock 2009)

Sistema gestor de base de datos: PostgreSQL v9.4

PostgreSQL es uno de los Sistemas de Gestión de Bases de Datos (SGBD) más avanzados a nivel mundial. Sus características técnicas lo hacen uno de los más potentes y robustos del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. (PostgreSQL 2017)

Conclusiones parciales

Durante el desarrollo de este capítulo se hizo una caracterización de los servidores de juegos multijugador que más se aproximan a la plataforma propuesta. Luego se seleccionó la metodología a usar, culminando en una caracterización de las herramientas y tecnologías que se emplearán en el proyecto.

Capítulo 2: Propuesta de Solución. Exploración y planificación

2.1 Propuesta de Solución

La propuesta de solución está basada en la arquitectura cliente-servidor. El cliente es un dispositivo móvil (dígase celular y/o Tablet) el cual contiene una aplicación del sistema operativo Android desarrollada en la universidad que, podrá conectarse directamente al servidor vía Wi-Fi. El servidor es un ordenador que al utilizar el servidor Apache Tomcat a través de los Websockets puede realizar transacciones Full-Dúplex, permitiendo así realizar una conexión dinámica y directa entre la aplicación y el servidor.

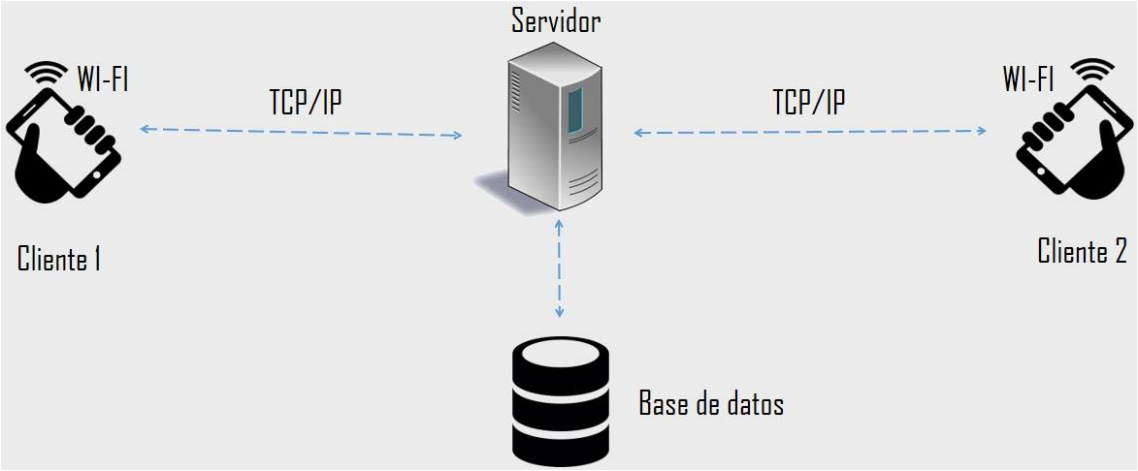

Figura 2: Propuesta de solución

2.2 Características del sistema

Una característica del sistema no es más que una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo. También se puede decir que es una capacidad que debe estar presente en un sistema para satisfacer un contrato, estándar, especificación u otro documento formal. Las características del sistema pueden dividirse en funcionales y no funcionales. (Gimson 2012)

2.2.1 Características no funcionales del sistema

Las características no funcionales del sistema tienen que ver con las que de una u otra forma puedan limitar el sistema, donde se aplican en su totalidad. Para satisfacer al cliente y lograr una buena calidad se listaron las siguientes propiedades y cualidades:

2.2.1.1 Requisitos no funcionales de la aplicación Servidor

Seguridad

- La plataforma solo proporcionará acceso a clientes autenticados, mediante un usuario y contraseña.
- La plataforma prohibirá el acceso al usuario que realice más de 5 intentos fallidos de autenticación.
- La plataforma validará la información que sea introducida por el usuario para garantizar la integridad del sistema.
- La plataforma le dará un acceso restringido a cada usuario, limitándolo solo a las funciones permitidas.

Restricciones de diseño

- El sistema será implementado haciendo uso de la plataforma JEE.

2.2.1.2 Requisitos no funcionales de la aplicación Cliente

Usabilidad

- La plataforma podrá ser usada por cualquier usuario que posea un conocimiento básico en el uso del móvil con sistema operativo Android.
- La plataforma será desplegada usando el idioma español.

Seguridad

- Los usuarios deberán autenticarse en aras de poder acceder al sistema.

Requisitos de hardware

- Los móviles deben tener todos los requerimientos de hardware necesarios para poder establecer una conexión con el servidor.

Requisitos de software

- El teléfono móvil debe contar con el sistema operativo Android en su versión 4.4 como mínimo.

2.2.2 Funcionalidades del sistema

Después de haber conocido los objetivos que se quieren lograr con el sistema, así como los requisitos no funcionales del mismo, se puede proseguir con el análisis de las funcionalidades con las que debe contar el software para darle respuesta a esos objetivos.

En relación con lo antes planteado el software debe ser capaz de:

- Autenticar usuario.
- Adicionar usuario.
- Gestionar partida.
- Controlar la lógica de juego.
- Control de estadísticas.
- Mensajería instantánea.

2.3 Fase de exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología. (Penadés y Penadés 2013)

2.3.1 Historias de usuarios

Las historias de usuario son requerimientos ya que expresan el problema que el sistema o producto software debe resolver. Las Historias de Usuario son un enfoque de requerimientos ágiles que se focalizan en establecer conversaciones acerca de las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad, usualmente un usuario. Poseen las siguientes características: una descripción escrita que será utilizada para planificar y posteriormente disgregar los detalles con el dueño del producto, las conversaciones propiamente dichas con el dueño del producto y las pruebas que han de determinar si las historias están finalizadas o no. (Izaurralde 2013)

2.3.2 Relación de las historias de usuario

Como parte del proceso de trabajo dentro de la fase de exploración se identificaron las siguientes Historias de Usuarios:

Historia de Usuario	
Número: 1	**Usuario:** Cliente
Nombre de la Historia: Autenticar usuario	

Prioridad en negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: El sistema permitirá la autenticación mediante el uso de usuario y contraseña, los cuales se verificarán en la base de datos y de ser correctos se generará un Token único, el cual se enviará al usuario y será el que usará para identificarse mientras esté conectado.	
Observaciones	

Tabla 1: HU Autenticar usuario.

Historia de Usuario	
Número: 2	**Usuario:** Cliente
Nombre de la Historia: Adicionar usuario	
Prioridad en negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: El sistema permitirá adicionar un usuario nuevo que se quiera conectar al sistema. Para ello el usuario debe enviar mediante la aplicación su nombre de usuario, contraseña y correo electrónico.	
Observaciones	

Tabla 2: HU Adicionar usuario.

Historia de Usuario	
Número: 3	**Usuario:** Cliente
Nombre de la Historia: Lógica de juego.	
Prioridad en negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos estimados: 3	**Iteración asignada:** 2
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez	

Martín.
Descripción: El sistema controlará las jugadas de cada jugador, respetando los turnos de juego y controlando el estado de la partida. Al final de la partida deberá dar a conocer el ganador y el perdedor.
Observaciones

Tabla 3: HU Lógica de juego

Historia de Usuario	
Número: 4	**Usuario:** Cliente
Nombre de la Historia: Gestionar partida	
Prioridad en negocio: Alta	**Riesgo de desarrollo:** Media
Puntos estimados: 2	**Iteración asignada:** 3
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: El sistema brindará al usuario las opciones de adicionar, unirse o eliminar partidas dependiendo de sus necesidades.	
Observaciones	

Tabla 4: Gestionar Partida.

Historia de Usuario	
Número: 5	**Usuario:** Cliente
Nombre de la Historia: Mensajería instantánea.	
Prioridad en negocio: Media	**Riesgo de desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 3
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: El sistema proveerá un pequeño servidor de mensajería instantánea para asegurar la comunicación interna de los jugadores.	

Observaciones

Tabla 5: HU Mensajería instantánea.

Historia de Usuario	
Número: 6	**Usuario:** Cliente
Nombre de la Historia: Control de estadísticas.	
Prioridad en negocio: Baja	**Riesgo de desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 3
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: El sistema controlará las estadísticas de cada jugador, actualizando sus registros en la base de datos.	
Observaciones	

Tabla 6: HU Mensajería instantánea.

2.4 Fase de Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. (Google 2013)

2.4.1 Estimación de esfuerzos

Para lograr un desarrollo eficiente y satisfactorio, se realizó una estimación de esfuerzos para cada una de las Historias de Usuarios identificadas en el proceso de planificación y se llegó a los resultados que se muestran a continuación:

Historias de Usuario	Puntos de estimación
Adicionar Usuario	1 semana
Autenticar usuario	1 semana
Lógica de juego	2 semanas
Gestionar partida	1 semana
Mensajería instantánea	1 semana
Control de estadísticas	1 semana

Tabla 7: Estimación de esfuerzos.

2.4.2 Plan de Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura. Sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas y las tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores. (Google 2013)

Después de haber definido las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, se tomó la decisión de realizar el sistema en iteraciones, las cuales se detallan a continuación:

Iteración # 1

La primera iteración tiene como objetivo la implementación de las HU #1 y #2 con prioridades para el cliente de Alta. Se disponen de 2 semanas para implementar las tareas. Se obtiene como resultado una primera versión del sistema propuesto la cual

será mostrada al cliente para desarrollar la retroalimentación con el equipo de desarrollo, para luego pasar a la siguiente iteración.

Iteración # 2

En esta iteración se darán cumplimiento a la HU #3 las cuales complementarían a la ya hecha. Se cuenta con 3 semanas para llevar a cabo la implementación de esta iteración. Al finalizar se obtendrá una segunda versión del sistema propuesto y se le hará llegar al cliente la iteración anterior junto con la presente para la aprobación o cambios pertinentes con el cliente.

Iteración # 3

En esta iteración se darán cumplimiento a las HU #4, 5 y 6 las cuales culminarían el proceso de entregas al cliente para completar la solución final del sistema la cual pasará a la fase de pruebas.

2.4.3 Plan de duración de las iteraciones

Para lograr una mayor organización del trabajo se crea un plan de duración de las iteraciones que tiene como objetivo mostrar la duración de cada iteración, así como el orden en que serán implementadas las historias de usuarios en cada una de ellas.

No. Iteraciones	Historias de Usuario	Duración total de Iteraciones
Iteración #1	Adicionar Usuario	2 semanas
	Autenticar usuario	
Iteración # 2	Lógica de juego	3 semanas
Iteración # 3	Gestionar partida	3 semana
	Mensajería instantánea.	
	Control de estadísticas.	

Tabla 8: Plan de duración de las Iteraciones.

2.4.4 Plan de Entrega

A continuación, se muestra el plan de entregas desarrollado para dar solución al problema planteado. Al desarrollar el mismo se tuvo en cuenta los puntos de estimación para obtener un resultado final.

No. Iteración	Duración	Fecha Inicio	Fecha Final
Iteración # 1	3 semanas	3-4-2017	14-4-2017
Iteración # 2	3 semanas	17-4-2017	5-5-2017
Iteración # 3	3 semanas	8-5-2017	26-5-2017

Tabla 9: Plan de entrega.

Conclusiones parciales

En este capítulo se realizó la propuesta de solución a implementar. Para esto se definieron los requisitos no funcionales y las HU definidas por el cliente. Luego se pasó a realizar la estimación de esfuerzo para cada uno de las HU, aterrizando finalmente en la construcción del Plan de duración de las iteraciones y el Plan de entregas.

Capítulo 3: Diseño e Implementación

3.1 Arquitectura de la solución propuesta

La metodología XP plantea que el desarrollo de una aplicación o software debe realizarse de forma iterativa y obtener al término de cada iteración un producto funcional que debe ser probado. Además, debe ser mostrado al cliente y al emitir una opinión se pueda obtener una mejor visión de lo que desea. Esta metodología está dirigida a potenciar las relaciones interpersonales como premisa para lograr el mejor resultado posible en el desarrollo del software, proponiendo un trabajo en equipo y buscando la preparación de los desarrolladores para lograr un buen ambiente de trabajo.

En las aplicaciones desarrolladas bajo las pautas de XP no se requiere la representación del sistema mediante diagramas de clases utilizando notación UML, al utilizarse otras técnicas como las tarjetas CRC. Sin embargo, se puede hacer uso de los diagramas cuando estos aporten mejoras en la comunicación entre los desarrolladores y su enfoque en la información importante.

3.1.1 Patrón arquitectónico: N Capas

Se propone la utilización del patrón arquitectónico N Capas que tiene como objetivo principal separar los diferentes aspectos del desarrollo, tales como las cuestiones lógicas del negocio y los mecanismos de almacenamiento. Los componentes de cada capa se comunican con los componentes de otras capas a través de interfaces bien conocidas, cada nivel agrega las responsabilidades y abstracciones del nivel inferior. (de la Torre y otros 2010)

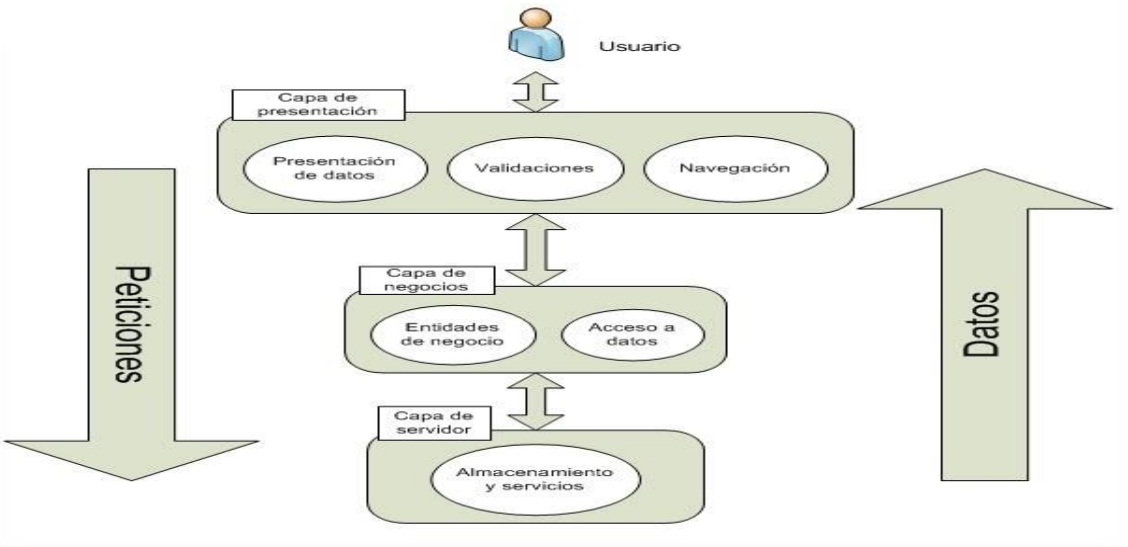

Figura 3: Patrón Arquitectónico N Capas

Principios claves

- Separa de forma clara la funcionalidad de cada capa.
- Cada capa contiene la funcionalidad relacionada solo con las tareas de esa capa.
- Las capas inferiores no tienen dependencia de las capas superiores.

Beneficios

- Aislamiento: se pueden realizar actualizaciones en el inferior de las capas sin que esto afecte el resto del sistema.
- Abstracción: los cambios se realizan a alto nivel y se puede incrementar o reducir el nivel de abstracción que se usa en cada capa del modelo.

3.1.2 Patrones de Diseño

La calidad de diseño de la interacción de los objetos y la asignación de responsabilidades presentan gran variación. Las decisiones poco acertadas dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender. Una implementación hábil se funda en los principios cardinales que rigen un buen diseño orientado a objetos. (Larman 1999)

Con aras de alcanzar un diseño con la calidad requerida se usan los Patrones Generales de Software para Asignar Responsabilidades (GRASP), así como los patrones Pandilla de los Cuatro (GOF).

Patrones GRASP

GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades). El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se quiere diseñar eficazmente el software orientado a objetos. (Larman 1999)

A continuación, se muestran los patrones GRASP utilizados:

- **Experto:** Es un patrón que se utiliza más que cualquier otro al asignar responsabilidades. Este patrón se evidencia en la mayoría de las clases ya que estas cuentan con la información necesaria para llevar a cabo las tareas correspondientes a cada una.
- **Creador:** Las clases que tienen la responsabilidad de crear objetos contienen toda la información necesaria para construir los mismos.

- **Bajo Acoplamiento:** Las clases están relacionadas con el mínimo de estas posibles que puedan relacionarse.
- **Alta cohesión:** Se aplica en la mayoría las clases del diseño, ya que en cada una solo se implementan las funcionalidades que le corresponden.

Patrones GOF

- Los patrones GoF se descubren como una forma indispensable de enfrentarse a la programación, a partir de entonces estos patrones son conocidos como los patrones de la pandilla de los cuatro (GoF, gang of four). (Gamma y Helm [sin fecha])
- **Singleton (instancia única):** garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Restringe la instanciación de una clase o valor de un tipo a un solo objeto. Un ejemplo de este patrón se encuentra en la clase GameManager.
- **Factory Method (método de fabricación):** centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística, es decir, la diversidad de casos particulares que se pueden prever para elegir el subtipo a crear. Parte del principio de que las subclases determinan la clase a implementar.

3.1.3 Tarjetas CRC

Las tarjetas CRC son una metodología para el diseño de software orientadas a objetos y creadas por Kent Beck y Ward Cunningham.

Cada tarjeta contiene el nombre de la clase que representa, además existe una descripción de las responsabilidades (métodos) asociadas con la clase, así como una lista de otras clases relacionadas mediante el envío de mensajes. El sistema CRC se usa principalmente como herramienta didáctica y como metodología para estudiar la conducta de los diseñadores orientados a objetos. Las tarjetas CRC son también un recordatorio y ayudan a los programadores experimentados y principiantes a comunicarse entre sí acerca de la modelación del entorno con objetos. (Rivera 2012)

El factor decisivo en la utilización de esta técnica para diseñar la aplicación que se desea desarrollar fue las características que esta brinda en cuanto a facilidad de su uso y entendimiento. En la tabla #12 se describe la Tarjeta CRC correspondiente a la clase ReadJson, la descripción las tarjetas restantes se podrán encontrar en los anexos.

Tarjeta CRC	
Clase: ReadJSON	
Responsabilidades: Parse	**Colaboraciones:** ConnecToDB LoginMsg joinGameMsg LogOutMsg disconnecGameMsg

Tabla 10: ReadJSON

3.2 Desarrollo de iteraciones

En la fase de planificación de la metodología XP se detallaron cada una de las HU correspondientes a cada iteración según la selección del cliente. Durante el desarrollo de las iteraciones se realiza una revisión del plan de iteraciones debido a que el mismo puede o no ser modificado. Como parte de este plan se crean tareas para ayudar a organizar la implementación exitosa de la HU.

Estas tareas en la que se descomponen las HU son las llamadas tareas de ingeniería, asignadas a una persona que pertenece al equipo de desarrollo y es quien responde por la implementación asignada. Estas tareas son escritas por los programadores y no por el cliente, dado que son para uso estricto de los primeros.

La planificación que se llevó a cabo para el desarrollo del sistema está compuesta por dos iteraciones pues permite que al final de la última iteración se logre un producto con todas las restricciones y características deseadas por el cliente. A continuación, se detallan cada una de las tareas de la ingeniería por iteraciones.

3.2.1 Iteración 1

Esta iteración tiene como objetivo darle cumplimiento a las HU #1 y HU #2

Historias de Usuario	Tiempo de implementación (semanas)	
	Estimación	Real
Adicionar Usuario	1	1
Autenticar usuario	1	1

Tabla 11: HU abordadas en la primera iteración

Mediante las siguientes tablas se evidencian las tareas de programación o ingeniería en las que fue desglosada la Historia de Usuario #1 y #2 para un mejor funcionamiento de la aplicación.

Tareas de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 1
Nombre de la tarea: Adicionar Usuario	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 3-4-2017	**Fecha fin:** 7-4-2017
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: Se implementarán las funcionalidades necesarias para que el usuario pueda después de mandar sus datos de nombre de usuario, contraseña y correo electrónico, se agreguen a la base de datos y pueda entrar al sistema correctamente.	

Tabla 12: Tarea de ingeniería #1 de la HU #1.

Tareas de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 2
Nombre de la tarea: Autenticar usuario	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1
Fecha inicio: 10-4-2017	**Fecha fin:** 14-4-2017
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: Se implementarán las funcionalidades necesarias para que el usuario se le autorice, después de mandar sus datos de nombre de usuario y contraseña, pueda verificar en la base de datos si están correctos y si el servidor le genera un Token, el cual, se le enviará al cliente. Este usará dicho Token para identificarse con el servidor mientras tenga su sesión abierta.	

Tabla 13: Tarea de ingeniería #1 de la HU #2

3.2.2 Iteración 2

Esta iteración tiene como finalidad desarrollar la HU #3.

Historias de Usuario	Tiempo de implementación

	(semanas)	
	Estimación	Real
Lógica de juego	3	3

Tabla 14: HU abordada en la segunda iteración.

En estas tablas se exponen las tareas de programación o ingeniería en las que fue desglosada la Historia de Usuario #3 para lograr un funcionamiento correcto de la aplicación.

Tareas de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 3
Nombre de la tarea: Lógica de juego	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 3
Fecha inicio: 17-4-2017	**Fecha fin:** 5-5-2017
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: Se implementarán las funcionalidades necesarias para llevar el control de las jugadas hechas por el usuario. Un mismo usuario no puede hacer 2 jugadas consecutivas.	

Tabla 15:Tarea de ingeniería #1 de la HU #3

3.2.3 Iteración 3

Esta iteración tiene como finalidad desarrollar las HU #4, 5 y 6.

Historias de Usuario	**Tiempo de implementación (semanas)**	
	Estimación	Real
Gestionar partida	1	1
Mensajería instantánea	1	1
Control de estadísticas.	1	1

Tabla 16:HU abordadas en la tercera iteración.

A partir de estas tablas se evidencian las tareas de programación o ingeniería en las que fue desglosada las Historias de Usuario #4, 5 y 6 en el desarrollo funcional de la aplicación.

Tareas de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 4
Nombre de la tarea: Crear partida	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2/1
Fecha inicio: 8-5-2017	**Fecha fin:** 9-5-2017
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: Se implementarán las funcionalidades necesarias para darle al usuario la posibilidad de crear una partida recibiendo como información el juego del cual se quiere crear la partida, nombre de la partida y el usuario que la crea.	

Tabla 17: Tarea de Ingeniería #1 de la HU #4.

Tareas de Ingeniería	
No. De la Tarea: 2	**No. De la HU:** 4
Nombre de la tarea: Unirse a una partida	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2/1
Fecha inicio: 10-5-2017	**Fecha fin:** 11-5-2017
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: Se implementarán las funcionalidades necesarias para darle al usuario la posibilidad de unirse a una partida recibiendo como información el juego al cual pertenece la partida, nombre de la partida y el usuario que se quiere unir.	

Tabla 18: Tarea de Ingeniería #2 de la HU #4.

Tareas de Ingeniería	
No. De la Tarea: 3	**No. De la HU:** 4
Nombre de la tarea: Salir de una partida	

Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.1/1
Fecha inicio: 12-5-2017	**Fecha fin:** 12-5-2017
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: Se implementarán las funcionalidades necesarias para darle al usuario la posibilidad de salir de su partida recibiendo como información el juego al cual pertenece la partida, nombre de la partida. Si todos los usuarios de una partida deciden salir de la misma el sistema borrará la partida de forma inmediata.	

Tabla 19: Tarea de Ingeniería #3 de la HU #4.

Tareas de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 5
Nombre de la tarea: Mensajería instantánea	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1/1
Fecha inicio: 15-5-2017	**Fecha fin:** 19-5-2017
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: Se implementarán las funcionalidades necesarias para darle al usuario la posibilidad de intercambiar mensajes de texto.	

Tabla 20: Tarea de Ingeniería #1 de la HU #5.

Tareas de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 6
Nombre de la tarea: Actualizar estadísticas	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5/1
Fecha inicio: 22-5-2017	**Fecha fin: 24-6-2017**
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	

Descripción: Se implementarán las funcionalidades necesarias para que el servidor actualice en la base de datos las estadísticas de los jugadores.

Tabla 21: Tarea de Ingeniería #1 de la HU #6.

Tareas de Ingeniería	
No. De la Tarea: 2	**No. De la HU:** 6
Nombre de la tarea: Mostrar estadísticas	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.5/1
Fecha inicio: 24-5-2017	**Fecha fin:** 26-6-2017
Programadores responsables: Yeniel Castellanos Díaz y Ronald Rodriguez Martín.	
Descripción: Se implementarán las funcionalidades necesarias para darle al usuario la información referente a sus estadísticas.	

Tabla 22: Tarea de Ingeniería #2 de la HU #6.

3.3 Pruebas Unitarias

Las pruebas unitarias son una actividad fundamental en la metodología XP, pues aseguran que un determinado módulo cumpla con un comportamiento esperado en forma aislada antes de ser integrado al sistema.

Estas pruebas brindan al programador una inmediata retroalimentación de cómo está realizando su trabajo, permitiéndole realizar cambios de forma segura respaldado por efectivos casos de prueba.

Las pruebas unitarias fueron empleadas durante todo el proceso de desarrollo del componente y en cada una de las funcionalidades implementadas. Se probó el código con casos de prueba creados por el equipo de desarrollo y se verificó si los resultados brindados eran los esperados para cada funcionalidad.

Esta imagen muestra los resultados de algunas de las pruebas realizadas.

Ilustración 1. Resultado de las pruebas unitarias realizadas a las funcionalidades de la clase Test

Conclusiones Parciales

En este capítulo se desarrolló una versión de la solución propuesta que corresponde con el 100% de la aplicación final. Se trabajó en la fase de construcción de la mitología XP, describiéndose las tarjetas CRC para alcanzar el mayor entendimiento posible del sistema. Se evidenció con detalles las dos iteraciones ejecutadas en la fase de construcción de la aplicación, así como sus tareas de programación e ingeniería.

Capítulo 4: Protocolo de comunicación y flujo de información

4.1 Protocolo de comunicación entre la aplicación y el servidor

Los mensajes que intercambiarán los clientes y el servidor se rigen por el formato de mensajes JSON. Cada mensaje tendrá su tipo y, además, un mensaje.

En estas tablas se describirán los tipos de mensajes que da soporte el servidor, así como los mensajes asociados a cada uno.

Tipo	Atributos			
Login	user	pass		app
LogOut	token			
newUser	user	pass	email	app
createPartida	idJuego	user		token
disconecGame	idJuego	user		token
joinGame	idPartida	user		token
jugada	user	jugada		token
error	error_msg	error		token
chat	chat_msg	chat		token
listPartidas	app			

Tabla 23: Tipos de mensajes que se envían del cliente al servidor y sus atributos asociados.

A continuación, se muestran los mensajes que se envían desde el servidor hacia el cliente.

Tipo	Mensajes
re_createPartida	“msg”

re_Login	“token”
re_newUser	“token”
re_LogOut	“msg”
re_joinPartida	“msg”
re_jugada	“user”
re_chat	“chat_msg”
re_error	“error_msg”
re_listPartidas	“partidas”

Tabla 24: Relación de los tipos de mensajes que envía el servidor al cliente con su contenido respectivamente.

4.2 Usuario

El protocolo asume que el sistema usa una autenticación basada en tokens. El mensaje de login contiene la información necesaria para la autenticación (usuario, password, aplicación) y devuelve un token generado aleatoriamente que debe ser usado por el cliente en todas sus futuras peticiones. En caso de que la autenticación no fuera exitosa se devuelve un mensaje de error.

La siguiente tabla muestra un ejemplo de los mensajes intercambiados durante la autenticación de un usuario:

Petición Cliente	**Respuesta del Servidor**

{
“tipo”:” Login”,
“msg”: {
“user”:” admin”,
“pass”:” admin”,
“app”: “TicTacToe”
}
}
	{
	“tipo”: “re_Login”,
	“msg”: {
	“Token”: “054789613329”
	}
	}

Tabla 25: Relación de los tipos de mensajes intercambiados durante el proceso de autenticación.

4.4 Partidas

El protocolo provee mensajes para el control de las partidas, soportando las siguientes operaciones:

- Lista de partidas
- Creación de una partida
- Unirse / abandonar partida

4.4.1 Listado de Partidas

El cliente selecciona esta opción, luego la aplicación envía al servidor un mensaje de tipo “listaPartidas” a lo que el servidor responderá al usuario con un mensaje de tipo “re_listaPartidas” donde enviará una lista de las partidas activas que estén esperando por otro jugador para empezar, así como también el usuario que creó la partida.

Petición Cliente	Respuesta del Servidor

{
“tipo”: “listaPartidas”,
“msg”: {
app: “TicTacToe”
}
}
	{
	“tipo”: “re_listaPartidas”,
	“msg”: {
	“partida”: {
	“id”: “qwe232”,
	“user”: “admin”
	}
	}
	}

Tabla 26: Ejemplo de mensajes entre el cliente y el servidor.

4.4.2 Creación de una Partida

El cliente envía un mensaje al servidor de tipo “createPartida”. El servidor al recibir este mensaje creará una partida del tipo de juego al cual corresponde la aplicación y responderá con un mensaje de tipo “re_createPartida”. A cada partida se le asigna, en su creación, un identificador que es único dentro del servidor. Este identificador se envía al cliente dentro del mensaje “re_createPartida”.

Petición Cliente	Respuesta del Servidor

{
“tipo”: “createPartida”,
“msg”: {
token: “054789613329”,
idGame: “TicTacToe”,
idUser: “Pepe”,
idPartida: “MyPartida”
}
}
	{
	“tipo”: “re_Partida,
	“msg”: {
	text: “Partida Creada”
	}
	}

Tabla 27: Ejemplo de mensajes entre el cliente y el servidor.

4.4.3 Unirse a una Partida

El cliente envía al servidor un mensaje de tipo “joinGame”, el servidor agrega el usuario a la lista de jugadores de dicha partida y da comienzo a la misma.

Petición Cliente	Respuesta del Servidor

{
“tipo”: “joinPartida”,
“msg”: {
token: “054789613329”,
idPartida: “MyPartida”,
idUser: “Pepe”
}
}
	{
	“tipo”: “re_joinPartida”,
	“msg”: {
	text: “Se ha unido correctamente”
	}
	}

Tabla 28: Ejemplo de mensajes entre el cliente y el servidor.

4.5 Juego

En este apartado es cuando la partida ya comenzó y los usuarios involucrados se envían las jugadas entre sí pasando primero por el servidor quien las analiza y lleva el control de la puntuación de la partida, del orden de juego, del tiempo y notifica quién gana y quién pierde.

Los mensajes de tipo jugada contienen el identificador del jugador que realizó la jugada, así como la posición X, Y de la jugada en la matriz del tablero.

Petición Cliente	Respuesta del Servidor

{
tipo: “jugada”,
msg: {
token: “054789613329”
idUser: “Pepe”,
jugada: {
x: “2”,
y: “2”
}
}
}
	{
	tipo: “re_jugada”,
	msg: {
	“x”: “2”,
	“y”: “2”
	}
	}

Tabla 29: Ejemplo de mensajes entre el cliente y el servidor.

Conclusiones Parciales

En este capítulo se describió el protocolo de comunicación y los mensajes que se envían entre el cliente y el servidor para asegurar la correcta comunicación entre ambos. También se vieron las acciones que realizan el servidor y el cliente al recibir los diferentes tipos de mensajes.

Conclusiones

Con el desarrollo de este trabajo se logró crear una plataforma que resuelve tanto el problema como el objetivo general planteado al comienzo de la investigación. De esta forma, durante el desarrollo de la investigación, se profundizó en el conocimiento de las herramientas utilizadas con el objetivo de conocer mejor sus funcionamientos y beneficios que brindan cada una de ellas. En aras de guiar todo el proceso se obtuvo un modelo a seguir a través de los artefactos brindados por la metodología XP, obteniéndose de esta forma una plataforma de juegos multijugador por turnos que provee un servidor de juegos, así como un conjunto de librerías de comunicación para Android y un protocolo de comunicación entre el servidor y los móviles.

Recomendaciones

Se recomienda continuar con el desarrollo de la plataforma desarrollada agregando nuevas funcionalidades como:

- Desarrollar una interfaz gráfica para la administración del servidor.
- Extender las funcionalidades del servidor para la implementación de juegos con control de tiempo.
- Extender las funcionalidades del servidor para la implementación de juegos en tiempo real.
- Implementar un sistema de motivación basado en el uso de logros.

Bibliografía

ABREU, C.C., VALDÉS, J.A., MEDINA, R.S., SOSA, D.P., QUINTANA, L.A. y ALAYÓN, J.L., 2011. Juegos de video y comportamiento en escolares de primaria y secundaria básica en Centro Habana, en el curso 2005-2006. *Revista Cubana de Higiene y Epidemiología* [en línea]. [Consulta: 9 noviembre 2016]. Disponible en: http://www.redalyc.org/articulo.oa?id=223221362003.

ALEJANDRA, 2007. API, Interface de Programación de Aplicaciones. [en línea]. [Consulta: 30 mayo 2017]. Disponible en: http://www.elwebmaster.com/referencia/api-interface-de-programacion-de-aplicaciones.

APACHE TOMCAT, 2017. Apache Tomcat® - Welcome! [en línea]. [Consulta: 4 mayo 2017]. Disponible en: http://tomcat.apache.org/.

DE LA TORRE, C.L. y OTROS, 2010. *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0*. S.l.: s.n.

ELECTROTANK, 2017. Electrotank. [en línea]. [Consulta: 4 mayo 2017]. Disponible en: http://www.electrotank.com/.

FETTE, I., 2011. The WebSocket Protocol. [en línea]. [Consulta: 10 marzo 2017]. Disponible en: https://tools.ietf.org/html/rfc6455.

GAMMA, E. y HELM, R., [sin fecha]. *Design Patterns - Elements of Reusable Software*. S.l.: s.n.

GIMSON, L., 2012. *Metodologías ágiles y desarrollo basado en conocimientos*. S.l.: s.n.

GOOGLE, 2013. Sities Google XP Metodologías. [en línea]. [Consulta: 4 mayo 2017]. Disponible en: https://sites.google.com/site/xpmetodologia/marcoteorico/funcionamiento.

GOOGLE PLAY GAME SERVICES, 2017. Play Games Services. *Google Developers* [en línea]. [Consulta: 4 mayo 2017]. Disponible en: https://developers.google.com/games/services/.

IZAURRALDE, M.P., 2013. *Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario*. 2013. S.l.: s.n.

JENDROCK, E., 2009. *The JavaEE6 Tutorial, Volume I Basic Concepts*. 2009. S.l.: s.n.

LARMAN, C., 1999. *UML Y PATRONES: Introducción al análisis y diseño orientado a objetos*. S.l.: s.n.

MONTERO, M.M. y ALVARADO, M. de los Á.M., 2001. El juego en los niños: enfoque teórico. *Educación* [en línea]. [Consulta: 9 noviembre 2016]. Disponible en: http://www.redalyc.org/articulo.oa?id=44025210.

NETBEANS, 2017. NetBeans Platform Learning Trail. [en línea]. [Consulta: 30 mayo 2017]. Disponible en: https://netbeans.org/kb/trails/platform.html.

PENADÉS, P.L. y PENADÉS, M.C., 2013. *Metodologías ágiles para el desarrollo de software: Extreme Programming (XP)*. 2013. S.l.: s.n.

POSTGRESQL, 2017. PostgreSQL: The world’s most advanced open source database. [en línea]. [Consulta: 4 mayo 2017]. Disponible en: https://www.postgresql.org/.

RIVERA, C., 2012. tarjetas crc. *prezi.com* [en línea]. [Consulta: 4 mayo 2017]. Disponible en: https://prezi.com/9hc9opeav71b/tarjetas-crc/.

RODRÍGUEZ, A., [sin fecha]. ¿Qué es Java? Concepto de programación orientada a objetos vs programación estructurada (CU00603B). [en línea]. [Consulta: 4 mayo 2017]. Disponible en: http://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=368:ique-es-java-concepto-de-programacion-orientada-a-objetos-vs-programacion-estructurada-cu00603b&catid=68&Itemid=188.

SMARTFOXSERVER, G.S.N.C.V.I.N., 8 12045 Fossano (CN) Italy Copyright (C), 2017. SmartFoxServer: massive multiplayer game server for Flash, Unity, HTML5, iOS and Android games, MMO, virtual worlds and communities. *SmartFoxServer* [en línea]. [Consulta: 4 mayo 2017]. Disponible en: http://www.smartfoxserver.com/.

SMED, J. y HAKONEN, H., 2006. *Algorithms and Networking for Computer Games*. S.l.: s.n.

SUPERDATASEARCH, 2017a. Market Brief Year in Review. [en línea]. [Consulta: 4 mayo 2017]. Disponible en: https://www.superdataresearch.com/.

SUPERDATASEARCH, 2017b. [Consulta: 4 mayo 2017]. Disponible en: https://www.superdataresearch.com/.

TORRES, P.L. y LÓPEZ, E.A.S., 2003. *Metodologías Ágiles en el Desarrollo de Software*. 2003. S.l.: s.n.

VILLAFUERTE, D.B.C., [sin fecha]. MÉTODO LÓGICO HISTÓRICO. [en línea]. [Consulta: 4 mayo 2017]. Disponible en: http://www.eumed.net/libros-gratis/2010e/816/METODO%20LOGICO%20HISTORICO.htm.

Anexos

Tarjeta CRC	
Clase: ReadJSON	
Responsabilidades: Parse	**Colaboraciones:** ConnecToDB LoginMsg joinGameMsg LogOutMsg disconnecGameMsg ChatWS

Tabla 30: ReadJSON.

Tarjeta CRC	
Clase: ChatWS	
Responsabilidades: handleChatMessage myError endChatChannel processNewUser processChatUpdate processSignoffRequest getCurrentUsername registrerUser updateUserList getUserList validateUsername broadcastUserListUpdate removeUser broadcastTranscriptUpdate addMessage	**Colaboraciones:**

Tabla 31:ChatWS

Tarjeta CRC	
Clase: LogOutMsg	
Responsabilidades: getToken	**Colaboraciones:** Msg

Tabla 32: LogOutMsg

Tarjeta CRC	
Clase: LoginMsg	
Responsabilidades: getTipo getUser getPass getApp	**Colaboraciones:** Msg

Tabla 33: LoginMsg

Tarjeta CRC	
Clase: disconnetGameMsg	
Responsabilidades: getToken getIdGame	**Colaboraciones:** Msg

Tabla 34: disconnetGameMsg.

Tarjeta CRC	
Clase: joinGameMsg	
Responsabilidades: getToken	**Colaboraciones:** Msg

getIdGame	

Tabla 35: joinGameMsg.