PROYECTO FINAL DE INGENIERÍA

DESARROLLO DE UN JUEGO PARA DISPOSITIVOS MÓVILES BASADO EN LA GEOLOCALIZACIÓN

Iljin, Ignacio – LU 128.388

Jungblut, Nicolás – LU 130.562

Ingeniería en Informática

Tutor:
Godio, Claudio, UADE

Marzo 9, 2017

UNIVERSIDAD ARGENTINA DE LA EMPRESA

FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

RESUMEN

El presente trabajo consiste en el análisis, diseño y desarrollo de un videojuego que utilice geolocalización para definir las reglas de jugabilidad y las formas de interacción de los usuarios.

Se desarrolló un diseño conceptual en base a una idea inicial en donde dos jugadores luchan entre sí dentro de una representación virtual de su entorno real, utilizando sus posiciones reales para generar la misma. Dicho concepto fue expandido luego para agregar funcionalidad adicional que complejice el diseño y lo haga más atractivo, desafiante y divertido para los jugadores que participen de la partida.

Durante un análisis previo se buscó seleccionar la herramienta que mejor se adapte a las necesidades y requerimientos del proyecto, como también a la falta de experiencia. Se analizaron diferentes alternativas que probaron ser de lo más variadas, demostrando que la tecnología se encuentra en un nivel de madurez que brinda una amplia oferta de posibilidades para diferentes niveles de conocimiento y requerimientos.

Para el desarrollo de la aplicación cliente se definió utilizar la arquitectura MVC (Modelo - Vista - Controlador) la cual fue desarrollada utilizando el Framework Titanium Appcelerator, y como lenguaje de programación se optó por el uso de JavaScript. En cuanto a la aplicación servidor, se utilizó una arquitectura de N-Capas desarrollada con el Framework Microsoft .NET utilizando como lenguaje de programación C#. Para ambos desarrollos se implementó SOAP para la interacción entre ellos.

El proyecto sirvió como medio de aprendizaje en el desarrollo tanto de videojuegos como de aplicaciones para plataformas mobiles, dos campos en los cuales no se contaba con experiencia ni profesional ni académica previa.

Se encontró que existe una profunda diferencia entre el desarrollo de sistemas tradicionales y de videojuegos. Dicha diferencia está basada en que mientras que en un sistema tradicional la guía principal del diseño y el desarrollo del sistema son los objetivos de negocio, los cuales están completamente objetivados y son definibles con precisión, en un videojuego la guía principal es el “factor diversión”, el cual es imposible de definir con precisión, además de tratarse de algo sumamente subjetivo.

ABSTRACT

The present work consists of the analysis, design and development of a videogame that uses geolocalization to define gameplay and interaction between players.

A conceptual design was developed, based on an initial idea where players would face each other in a virtual representation of the real world, using their real coordinates to generate that representation. Later on, that concept was expanded to add new functionalities and mechanics to make the game design more complex, challenging and fun to play.

An initial analysis was made in order to find the best tool that could adapt to the requirements and necessities of the project, including the lack of technical experience. Different alternatives, that proved to be varied in their characteristics, were taken into consideration, which proved that the mobile technology has been in such a level of maturity that there is a wide offer of alternatives for different levels of knowledge and requirements.

In order to develop the client application, an MVC architecture was defined, which was implemented using the Titanium Appcelerator Framework. JavaScript was chosen as the programming language. Regarding the server application, an N-Layer architecture was used, and it was developed using Microsoft .NET Framework, choosing C# as a programming language. For both sides, SOAP Services were used in order to develop the interaction between them.

This project was used as a means to learn to develop mobile applications and videogames, both fields in which no previous experience, neither academic nor professional, was accounted for.

It was found that there is a great difference between developing traditional systems and videogames. Such difference is based on the fact that, whereas traditional systems are based on business objectives, which are clear, neutral and can be precisely defined, a videogame is based on the “fun factor”, which is subjective and can’t be defined with accuracy.

Contenido

INTRODUCCION

Objetivos

Principal: Desarrollar un videojuego multijugador online para dispositivos móviles Android que utilice geolocalización como parte integral de sus mecánicas principales, en donde la interacción entre los usuarios/jugadores se vea influenciada por la ubicación física real de los mismos.

Secundario: Conocer las diferentes alternativas para desarrollar aplicaciones para dispositivos móviles Android, reconociendo fortalezas y debilidades de cada una, eligiendo la más óptima para el desarrollo planeado. Aprender a utilizar dicha plataforma y luego realizar un informe en donde se den a conocer las expectativas iniciales y el resultado final de la experiencia.

Contenido

El primer capítulo versa sobre las generalidades de la temática, presentando como se llegó al estado actual del mercado de los videojuegos, cuál es el estado actual del mercado de los videojuegos y cuáles son los pormenores del diseño de videojuegos.

El segundo capítulo explica la metodología de desarrollo elegida, el análisis de competencias y las herramientas seleccionadas para la elaboración del proyecto.

El tercer capítulo presenta los diferentes niveles de diseño de la aplicación, desde su funcionalidad, arquitectura llegando finalmente al diseño técnico detallado de bajo nivel.

El cuarto capítulo detalla la experiencia vivida durante el desarrollo del proyecto y los problemas que fueron surgiendo.

El quinto capítulo menciona brevemente las pruebas realizadas durante el desarrollo y cuáles fueron las problemáticas encontradas durante la ejecución de las mismas.

El sexto y último capítulo describe las observaciones y conclusiones surgidas del proyecto.

CAPITULO I: Generalidades

1.1 Definición del Problema

La popularización de los dispositivos móviles trajo aparejado no solo un profundo cambio en la forma en la que las personas se comunican e interactúan entre sí, sino que también un cambio en las formas en las que las personas pasan sus momentos de ocio. Los videojuegos, que tuvieron su gran gestación durante los años 1970 y 1980, para explotar durante la década de los '90 y popularizarse más allá del nicho al que apuntaban originalmente a partir del año 2000, ingresaron repentinamente en la vida de todas las personas. Los teléfonos inteligentes, dispositivos que se volvieron indispensables en la vida diaria, le permitieron el acceso a todo el mundo a una consola de videojuegos portátil: ahora, sean o no fanáticos de los videojuegos, todos tienen la posibilidad de jugar en cualquier momento y en cualquier lugar.

Este avance tecnológico provocó también un cambio profundo en el paradigma de los videojuegos: a pesar de que aún persiste la idea que la mayoría de los jugadores son menores de edad, en 2015 la edad promedio de los jugadores se calculó en 35 años. También, mientras el 26% de los jugadores son menores de 18 años, el 27% son mayores de 50. Además, el 31% de los jugadores prefieren los juegos sociales y el 35% tiene como medio preferido un *Smartphone*[1].

Está claro que este significativo avance en el uso de los videojuegos como medio de diversión no hubiese sido posible sin la expansión del mercado de los dispositivos móviles: la posibilidad de jugar sin la necesidad de comprar un dispositivo dedicado de forma exclusiva atrajo al mundo de los videojuegos a nuevos consumidores, los llamados "jugadores casuales", quienes en otro momento solamente jugaban desde sus computadoras de escritorio por períodos esporádicos y muy acotados de tiempo.

[1] Cifras obtenidas a partir de una encuesta sobre una base de 4.000 hogares en los Estados Unidos de América, realizada por Entertainment Software Association, organización compuesta por las empresas más grandes del mercado.

Hoy en día tanto los jugadores casuales como los duros[2] disfrutan de los juegos en celulares, y el dinero que los mismos invierten en videojuegos crece año a año: según Newzoo, uno de los proveedores de estadísticas de videojuegos más importantes, las ganancias de los mercados de los videojuegos de dispositivos móviles no han parado de crecer, pasando de los 24,5 mil millones de dólares en 2014 a los 30 mil millones de dólares en 2015, estimando que alcanzarían los 44,2 mil millones de dólares en 2018.

Además de crecer como mercado, las ventas de videojuegos para dispositivos móviles también han crecido en lo que respecta su participación dentro de las ganancias referidas a videojuegos en general (es decir, tomando también en cuenta los videojuegos de consolas hogareñas, consolas portátiles y computadoras), pasando de un 29% de participación en 2014 a un 33% en 2015, y estimando una participación del 39% en 2018.

Teniendo en cuenta esas estadísticas y predicciones es completamente seguro afirmar que el mercado de los videojuegos para dispositivos móviles se encuentra en continua expansión por lo que siempre tendrá las puertas abiertas a oportunidades de negocio. Con la presente Tesis, “Desarrollo de un videojuego que utilice geolocalización”, se propone aprovechar ese crecimiento para ofrecer un producto que si bien no resulta revolucionario, sí utiliza características exitosas conocidas combinadas de una forma que hasta el día de hoy no se ha visto en el mercado.

[2] Derivado del término anglosajón “hardcore gamer”, que hace referencia a quienes invierten mucho tiempo y dinero en videojuegos, además de interesarse en los mismos más allá del juego en sí.

CAPITULO II: Análisis

2.1 Análisis de Competencia

Se presenta a continuación un análisis de la competencia, del cual se obtuvo un listado de videojuegos con características de base similares. Por cada uno de ellos se detalla su temática, la empresa desarrolladora, la base de usuarios y el modelo de negocio que aplica.

Luego del mismo se presenta un cuadro que lista las similitudes y diferencias con cada uno de los juegos encontrados, seguido de una conclusión basada en el mismo en la cual se marcan cuáles son los posibles competidores más importantes, incluyendo un análisis FODA.

2.1.1 Análisis de Competidores

Tourality

http://www.tourality.com/

El juego enfrenta a los usuarios, tanto de forma solitaria como en equipos, con el objetivo de alcanzar ciertos puntos predefinidos en el mapa. Los jugadores tienen que desplazarse en el mundo real a dichos puntos en el menor tiempo posible. Los lugares a alcanzar pueden tener diferentes propiedades que son desconocidas hasta llegar al mismo: dar puntos al jugador, robarle puntos al jugador, recuperar los puntos robados, etc. El juego también cuenta con ítems que se compran con puntos y pueden ser utilizados a beneficio del jugador o para perjudicar a sus rivales. El juego es persistente y todos los jugadores están compitiendo entre sí de forma simultánea.

- Modelo de negocio: Freemium
- Desarrollada por: Creative Workline
- Cantidad de descargas estimadas: de 10000 a 50000

GPS Tycoon

https://play.google.com/store/apps/details?id=com.gpstycoon

El juego se basa en la compra y venta de tierras. Los jugadores compran, de forma virtual, parcelas de tierra en el mapa, las cuales a su vez generan dinero para poder comprar más parcelas, o proteger las propias de las compras de otros jugadores. Para comprar una parcela es necesario encontrarse físicamente en la misma. El juego es persistente y todos los jugadores del mundo compiten entre sí de forma simultánea.

- Modelo de negocio: Free to Play
- Desarrollada por: Desarrollador Particular, no una empresa
- Cantidad de descargas estimada: de 1000 a 5000

GeoCaching

https://www.geocaching.com/play

En este juego los jugadores de todo el mundo buscan "cofres de tesoros" en el mundo real, utilizando una serie de pistas. Una vez encontrado, el jugador puede tomar algún objeto de adentro del mismo, anotar su nombre dentro de una hoja (si la misma está presente dentro del cofre) y dejar algún regalo para el próximo jugador (al tomar un objeto, tiene que dejar otro de igual o mayor valor). Finalmente, los jugadores relatan su experiencia y la suben al servidor del juego.

El juego cuenta con una versión Premium paga, la cual da acceso a los listados de mejores cofres, un mejor sistema de pistas, mejor sistema de mapas y otras mejoras en la interfaz de usuario.

- Modelo de negocio: Freemium
- Desarrollada por: Groundspeak
- Cantidad de descargas estimadas: de 1.000.000 a 5.000.000

Ingress

https://play.google.com/store/apps/details?id=com.nianticproject.ingress

En este juego cada jugador elige un bando que se encuentra en permanente conflicto con el bando rival. Los jugadores tienen que acercarse en la vida real a puntos clave estipulados en el mapa, basados en diferentes puntos de interés como pueden ser monumentos, museos para hackearlos o defenderlos, utilizando "puntos de acción" (un recurso finito propio de cada jugador, que se regenera con el tiempo y las acciones exitosas). Los jugadores del mismo equipo pueden colaborar entre sí para la conquista o defensa de los puntos de interés. El juego continúa de forma permanente, sin ningún fin específico más allá de asegurar la dominación de la mayor parte del mapa posible por parte del equipo al que uno pertenece.

- Modelo de negocio: Publicidad
- Desarrollada por: Google NianticLabs
- Cantidad de descargas estimadas: de 5.000.000 a 10.000.000

Pokemon Go

http://www.pokemongo.com

Este juego está basado en la popular y sumamente exitosa franquicia Pokemon, que ya posee en su haber una gran cantidad de videojuegos, series de televisión, merchandising y películas. El objetivo es moverse por el mapa en la vida real para poder llegar a diferentes ubicaciones, generadas al azar, en donde se encuentran los Pokemons para poder atraparlos. También es posible acercarse a otros puntos, predefinidos con anterioridad y fijos de forma permanente, para poder utilizar a los Pokemons capturados para combatir y conquistar dichos puntos.

- Modelo de negocio: Free to play
- Desarrollada por: Google NianticLabs
- Cantidad de descargas estimadas: más de 50.000.000 a nivel mundial

Parallel Kingdom

http://www.parallelkingdom.com/

Este juego crea un mundo virtual por sobre el mundo real. Los jugadores controlan a un personaje con habilidades especiales y las unidades que este puede fabricar. Los mismos pueden conquistar los lugares en donde físicamente se encuentran, o desplazar a su personaje hacia donde físicamente se encuentran sus amigos, sin necesidad de que ellos viajen a esa ubicación. Los diferentes terrenos del juego que se pueden conquistar pueden generar recursos para los jugadores que los poseen.

Los jugadores pueden enfrentar a sus personajes entre sí, como también a personajes controlados por el ordenador (denominados como NPC, *non player characters*). El juego continúa de forma permanente, sin ningún fin específico más allá de asegurar la dominación de la mayor parte del mapa y de recursos posibles.

- Modelo de negocio: Free to Play
- Desarrollado por: PerBlue
- Cantidad de descargas estimadas: de 1.000.000 a 5.000.000

2.1.2 Similitudes y diferencias

TABLA I: Comparativa entre Competidores

Juego	Similitudes	Diferencias
Tourality	Conquista de puntos en el mapa.	Es necesario moverse para alcanzar los objetivos. No es de combate.
GPS Tycoon	El mapa de juego depende de la ubicación física del jugador. Manejo de recursos.	Juego persistente (es decir, no hay partidas con principio y fin). Es necesario moverse para alcanzar los objetivos. No es de combate.

GeoCaching	Ninguna.	No es competitivo. Es necesario moverse para alcanzar los objetivos. No es de combate.
Ingress	Puntos de interés del juego coinciden con puntos de interés del mundo real. Se requiere la conquista de puntos de interés para obtener recursos. Manejo de recursos.	Juego persistente. Es necesario moverse para alcanzar los objetivos. Es en tiempo real. Es en equipos. No se fabrican unidades.
Pokemon Go	Puntos de interés del juego coinciden con puntos de interés del mundo real.	Juego persistente. Es necesario moverse para alcanzar los objetivos. Es en tiempo real. No es de estrategia.
Parallel Kingdom	Combate entre personajes y ejércitos. Es posible fabricar unidades. Manejo de recursos.	Juego persistente. Es necesario moverse para alcanzar los objetivos. Es en tiempo real. Es posible mover al personaje a lugares en donde el jugador no se encuentra físicamente.

2.1.3 Conclusiones

Como puede verse, un tópico recurrente en los juegos existentes que utilizan geolocalización es el movimiento de las personas a puntos específicos para realizar alguna acción dentro del juego. La gran mayoría propone mundos persistentes – es decir, una gran y continua partida que nunca termina, en donde el objetivo es acumular la mayor cantidad de puntos posibles, completando una serie de objetivos que se reciclan con el tiempo – y en muchos casos, partidas en equipos.

Por concepto, el juego que más se parece a la idea planteada en este proyecto es el Parallel Kingdom, ya que el mismo permite crear unidades, conquistar lugares en el mapa del

juego que se corresponden a lugares del mundo real y atacar a otros jugadores. Sin embargo, el hecho de que sea necesario desplazarse en el mundo real y que cada jugador este representado por un personaje con habilidades especiales lo convierte en un juego completamente diferente. Sumado a eso, Parallel Kingdom se trata de un juego persistente, por lo cual no está basado en partidas.

Ingress se trata de uno de los juegos con geolocalización más populares del mercado. Desarrollado por Google, el mismo cuenta con una interfaz estéticamente atractiva, una premisa que empuja a un juego colaborativo entre los jugadores del mismo equipo y a la competencia con los rivales. Al tratarse de un juego muy popular, con una amplia base de jugadores y patrocinado por una empresa de un alto perfil y alcance, aún al tratarse de un estilo de juego diferente se convierte en la principal amenaza.

Pokemon Go es el juego con geolocalización más popular del mundo, a pesar de ser el más reciente. El nivel de publicidad y popularidad que el mismo ha alcanzado está fuertemente atado a la marca Pokemon, que más allá del mercado de los videojuegos es una marca con un peso específico casi tan grande como Coca-Cola, por poner un ejemplo, con 400 licencias vendidas y un mercado de un valor de 47 mil millones de dólares.[3] Sin embargo, por las características del juego y la forma de jugar, se trata de un juego que entra en una categoría completamente diferente.

Tourality, Geo Caching y GPS Tycoon son tres juegos con conceptos completamente diferentes al planteado en el proyecto. Los primeros dos se basan en recorrer lugares para alcanzar objetivos, en el caso de Geo Caching objetivos en el mundo real con recompensas físicas, lo cual requiere por parte de los usuarios cierta planificación para poder participar de una sesión de juego, reservándose un importante espacio de tiempo para poder desplazarse de un lado a otro. GPS Tycoon se trata de un juego de administración, con ciertas similitudes al conocido juego de mesa Monopoly, por lo que si bien los jugadores pueden jugar en cualquier momento, se trata de un estilo de juego completamente diferente al planteado en la tesis.

[3] Cifras oficiales de The Pokemon Company, la compañía propietaria de los derechos comerciales de la marca Pokemon: https://www.pokemon.co.jp/corporate/en/data/

TABLA II: FODA

FORTALEZAS	OPORTUNIDADES
Alta experiencia en desarrollos de aplicaciones cliente-servidor.	No existe actualmente un juego en el mercado que combine estrategia y geolocalización. El mercado de jugadores en dispositivos móviles se encuentra en continua expansión. A pesar de que Ingress se encuentra hace mucho tiempo en el mercado, aun no salió de la fase beta. A pesar de que Pokemon Go es una marca fuerte, se trata de un tipo de videojuegos completamente diferente.
DEBILIDADES	**AMENAZAS**
No se cuenta con una marca fuerte como la de algunos competidores. Falta de experiencia en desarrollos para plataformas *móviles*. Falta de experiencia en desarrollo de videojuegos.	Existen muchos juegos en el mercado con una base importante de jugadores. Algunos de los juegos existentes están patrocinados o desarrollados por empresas de gran calibre. Competir con Pokemon Go resulta imposible por la fuerza de la marca.

2.2 Plan de Proyecto

En esta sección se describen las metodologías adoptadas para la administración del proyecto y el desarrollo del ciclo de vida de la aplicación. Además se detallan los riesgos a tener en cuenta para el desarrollo de la aplicación y los planes de respuesta ante cada uno de ellos. Por último se muestra el plan de desarrollo del proyecto con la descripción de los pasos realizados.

2.2.1 Metodologías de desarrollo

La metodología de desarrollo de software en ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Para el desarrollo de la solución se optó por la utilización de Metodologías Ágiles (AUP) las cuales se enfocan en la gente y en los resultados, además de brindar mecanismos para la gestión de cambios que implican un menor esfuerzo.

Las metodologías tradicionales funcionan muy bien en proyectos donde el problema es conocido y la solución del mismo está bien definida. En este entorno es fácil analizar, diseñar y ejecutar una solución, pero en el contexto del proyecto que se está desarrollando los requerimientos son cambiantes ya que se parte de un objetivo que puede sufrir modificaciones durante el desarrollo del mismo debido a que se deben realizar análisis sobre las herramientas a utilizar y sobre todas las posibles soluciones para lograr el objetivo. En el entorno planteado son más eficaces las metodologías ágiles ya que éstas incorporan mecanismos de gestión del cambio.

Todas las Metodologías Ágiles cumplen con el Manifiesto Ágil que no es más que una serie de principios agrupados en 4 valores:

- **Los individuos y su interacción**, por encima de los procesos y herramientas.
- **El software que funciona**, frente a la documentación exhaustiva.
- **La colaboración con el cliente**, por encima de la negociación contractual.
- **La respuesta al cambio**, por encima del seguimiento de un plan.

Durante el proceso de desarrollo se siguieron las pautas de la Metodología Proceso Unificado Ágil o Agile Unified Process (AUP). Dicha metodología es un una versión simplificada del proceso Unificado de Rational (RUP) y describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.

El AUP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas (Test Driven Development - TDD), Modelado Ágil, Gestión de Cambios Ágil, y Refactorización de Base de Datos.

El proceso unificado (Unified Process o UP) es un marco de desarrollo de software iterativo e incremental. A menudo es considerado como un proceso altamente ceremonioso porque especifica muchas actividades y artefactos involucrados en el desarrollo de un

proyecto de software. Dado que es un marco de procesos, puede ser adaptado y la más conocida es RUP (Rational Unified Process) de IBM.

AUP se preocupa especialmente de la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. Para ello, se crean y mantienen listas identificando los riesgos desde etapas iníciales del proyecto. Especialmente relevante en este sentido es el desarrollo de prototipos ejecutables durante la elaboración del producto, donde se demuestre la validez de la arquitectura para los requisitos claves del producto y que determinan los riesgos técnicos.

El proceso AUP establece un Modelo más simple que el planteado en RUP por lo que reúne en una única disciplina las disciplinas de Modelado de Negocio, Requerimientos y Análisis y Diseño. El resto de las disciplinas (Implementación, Test, Despliegue, Gestión de Configuración, Gestión de Proyecto y Entorno) coinciden con las restantes de RUP.

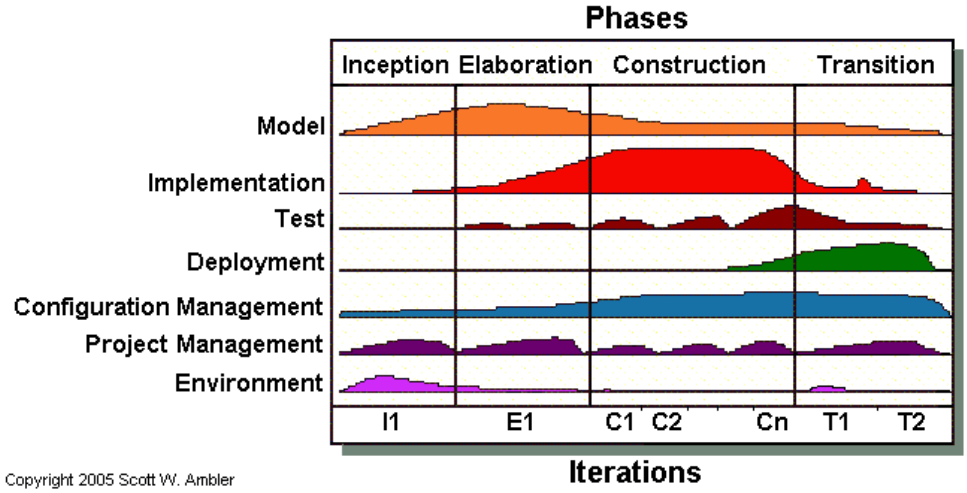

Figura 1 - The Agile Unified Process (AUP)

Fuente: The Agile Unified Process <http://www.ambysoft.com/unifiedprocess/agileUP.html>

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva:

- ***Inception*** (Iniciación): El objetivo es identificar el alcance inicial del proyecto, definir una o varias arquitecturas potenciales para el sistema y obtener la aceptación por parte de los stakeholders.

- ***Elaboration*** (Elaboración): El objetivo es identificar y validar la arquitectura del sistema.
- ***Construction*** (Construcción): El objetivo consiste en construir software desde un punto de vista incremental basado en las prioridades de los stakeholders.
- ***Transition*** (Transición): El objetivo es validar y desplegar el sistema en el entorno de producción. Previamente se realizan las pruebas necesarias para su validación en el entorno de preproducción.

A lo largo de las cuatro fases se desarrollan actividades relativas a siete disciplinas de manera iterativa:

1. ***Model*** (Modelo): El objetivo de esta disciplina es entender el negocio de la organización, el problema de dominio que se aborda desde el proyecto, y determinar una solución viable para resolver el problema de dominio.
2. ***Implementation*** (Implementación): El objetivo de esta disciplina es transformar su modelo (s) en código ejecutable y realizar pruebas básicas, en particular pruebas unitarias.
3. ***Test*** (Pruebas): El objetivo de esta disciplina consiste en realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, y verificar que se cumplan los requerimientos.
4. ***Deployment*** (Despliegue): El objetivo de esta disciplina es planear la entrega del sistema y ejecutar el plan para hacer que el sistema quede disponible para los usuarios finales.
5. ***Configuration Management*** (Gestión de configuración): El objetivo de esta disciplina es la gestión de acceso a los artefactos del proyecto. Esto incluye, además de la traza de versiones de los artefactos, el control de cambios y la gestión de los mismos.
6. ***Project Management*** (Gestión de proyecto): El objetivo de esta disciplina es dirigir las actividades que se llevan a cabo en el proyecto. Esto incluye, la gestión de riesgos, la dirección de personas (la asignación de tareas,

seguimiento de progresos, etc.), y coordinación con las personas y sistemas que se encuentran fuera del alcance del proyecto.

7. ***Environment*** (Entorno): El objetivo de esta disciplina es apoyar el resto de los esfuerzos para asegurar que los procesos, métodos y herramientas estarán disponibles para el equipo cuando los necesiten.

Los equipos de AUP suelen ofrecer versiones de desarrollo de la aplicación al final de cada iteración. Una versión de desarrollo de una aplicación es algo que podría potencialmente ser liberado en el ambiente de producción si se coloca basándose en los controles de calidad (QA) y pruebas de funcionamiento realizadas en el ambiente de pre-producción. La primera versión de desarrollo de la aplicación que se libera en el ambiente de producción a menudo toma más tiempo en comparación a las versiones posteriores.

Un enfoque temprano sobre los problemas de implementación no solo permite evitar problemas sino que también permite tomar ventaja utilizando la experiencia adquirida durante la etapa de desarrollo. Por ejemplo, cuando se implementa un software en el ambiente de testeo se deben tomar notas de lo que funciona y lo que no; dichas notas luego pueden servir para la elaboración del plan de implementación que se llevará a cabo en el ambiente de producción.

Principios de la AUP

La AUP es ágil, porque está basada en los siguientes principios:

1. ***El personal sabe lo que está haciendo***. La gente no va a leer la documentación detallada del proceso, pero va a querer alguna orientación de alto nivel y/o formación de vez en cuando.
2. ***Simplicidad***. Todo se describe concisamente utilizando algunas páginas, y no miles de ellas.
3. ***Agilidad***. AUP se ajuste a los valores y principios de Agile Alliance[4].
4. ***Centrarse en actividades de alto valor***. La atención se centra en las actividades que son esenciales para el desarrollo del proyecto, y no en todas las actividades que suceden durante el proyecto.

[4] Agile Alliance es una organización sin fines de lucro con membresía global, comprometida con el avance de los principios y prácticas de desarrollo ágil <www.agilealliance.org>

5. ***Independencia de las Herramientas***. Se puede utilizar cualquier conjunto de herramientas que se desee con AUP. Lo aconsejable es utilizar las herramientas que se adecúan de mejor manera al trabajo, a menudo suelen ser herramientas simples o incluso herramientas de código abierto.

2.2.2 Riesgos y planes de respuesta

Según el *Project Management Institute* riesgo es*: "un evento o condición incierta que, si ocurre, tiene un efecto positivo o negativo en al menos uno de los objetivos del proyecto, tales como tiempo, costo, alcance o calidad"*. El mismo califica los riesgos de la siguiente forma:

- Riesgos técnicos, de calidad y/o rendimiento: presente durante las actividades de diseño y desarrollo.
- Riesgos en la gerencia de proyectos: presentes en la gestión y manejo de los procesos de gestión del proyecto.
- Riesgos organizacionales: provenientes de la organización, tales como procesos adoptados, reglamentaciones internas, personal disponible, etc.
- Riesgos externos: externos al entorno de la organización, tales como legislación, contexto socio-económico, etc.

Los riesgos se miden luego en dos dimensiones diferentes:

- Probabilidad de ocurrencia: cuán viable es que dicho riesgo ocurra.
- Impacto: cuantificación arbitraria de cómo impacta en el proyecto, ya sea en términos económicos, tiempos, prestigio, legales, etc.

Dado que esta tesis no se desarrolla en un ámbito organizacional, los últimos dos tipos de riegos (Organizacionales y Externos) no han sido tomados en cuenta para el análisis.

Se presenta a continuación una Matriz de Probabilidad e Impacto, listando los riegos considerados:

Listado de riesgos técnicos, de calidad y/o rendimiento:

1. La arquitectura propuesta no sirve para acompañar los requerimientos técnicos.
2. La curva de aprendizaje de las tecnologías es muy pronunciada.
3. Desconocimiento en nuevas herramientas genera atrasos en el desarrollo.
4. Ausencia de buenas prácticas.
5. Diseño muy complejo o ininteligible.
6. La infraestructura disponible no tiene el rendimiento necesario para el desarrollo.
7. El análisis o el diseño no cumplen con todas las funcionalidades.

Listado de riesgos en la gerencia de proyectos:

8. Estimación errónea en la duración de algunas actividades.
9. Cambios en los requerimientos del proyecto durante su desarrollo.
10. No se monitorean los riesgos correctamente.

TABLA III: MATRIZ P.I.

Probabilidad\Impacto	**Insignificante**	**Menor**	**Moderado**	**Mayor**	**Catastrófico**
Casi cierto					
Probable		3		8	
Posible	2			5-10	
Poco probable		4		1	6-7
Remoto					9

Se plantea luego la necesidad de adoptar diferentes estrategias para poder atacar los riesgos encontrados y de esa forma reducir su impacto y/o probabilidad de ocurrencia. Para lograr eso, existen cuatro estrategias posibles:

- Aceptar: dado que es imposible eliminar dicha amenaza del proyecto, la misma es aceptada y tenida en cuenta durante el análisis, estimación y desarrollo. También es posible que su bajo impacto o posibilidad de ocurrencia no justifique un plan de acción contra el mismo.

- Evitar: implica modificar el plan de proyecto para eliminar la amenaza, ya sea cambiando el cronograma, las estimaciones, los requerimientos, la tecnología a utilizar o, en casos extremos, cancelando el proyecto.
- Mitigar: implica reducir la probabilidad de ocurrencia o el nivel del impacto del mismo en el proyecto. Una correcta mitigación suele ser menos costosa que reparar el daño causado por un riesgo luego de que el mismo ocurra. Puede implicar cambios en los procesos a adoptar, efectuar más pruebas, cambios en la tecnología, etc.
- Transferir: consiste en trasladar el impacto, o parte del mismo, y la gestión del riesgo a un tercero. Puede implicar la contratación de un proveedor externo para realizar tareas específicas.

En base a la anterior evaluación, se plantean las siguientes estrategias para cada uno de los riesgos encontrados:

TABLA IV: Plan de respuesta

Riesgo	Estrategia	Acciones
1- La arquitectura propuesta no sirve para acompañar los requerimientos técnicos	Mitigar	Invertir más tiempo en la etapa de diseño para obtener la arquitectura que mejor se adapte a las necesidades técnicas y capacidades del equipo. Consultar con un experto para validar la arquitectura diseñada.
2- La curva de aprendizaje de las tecnologías es muy pronunciada	Aceptar	La adopción de nuevas tecnologías siempre implica someterse a la curva de aprendizaje de las mismas.
3- Desconocimiento en nuevas herramientas genera atrasos en el desarrollo	Mitigar	Si bien es imposible eliminar por completo este riesgo, es posible reducir su probabilidad de ocurrencia seleccionando herramientas con buena documentación y con un uso similar a herramientas ya conocidas.
4- Ausencia de buenas prácticas	Aceptar	Se desestima este riesgo dado que los miembros del equipo tienen experiencia suficiente y adoptaran buenas prácticas del comienzo al fin del proyecto.

5- Diseño muy complejo o ininteligible	Evitar	Se dedicará un mayor tiempo a la etapa de diseño y se acudirá a un experto para la revisión del mismo.
6- La infraestructura disponible no tiene el rendimiento necesario para el desarrollo	Mitigar	Durante la etapa de análisis de herramientas y la de diseño se tomaran los recaudos para que los mismos se adapten a la infraestructura disponible por el equipo.
7- El análisis o el diseño no cumplen con todas las funcionalidades	Evitar	Invertir suficiente tiempo en la etapa de análisis y diseño para obtener una documentación exhaustiva que abarque todas las funcionalidades. Consultar a un experto para la revisión de la documentación.
8- Estimación errónea en la duración de algunas actividades	Mitigar	Utilizar metodologías de estimación acordes que permitan reducir la probabilidad de ocurrencia de este riesgo.
9- Cambios en los requerimientos del proyecto durante su desarrollo	Aceptar	Se desestima este riesgo dado que, si bien su impacto es alto, su probabilidad de ocurrencia es cercana a nula, ya que los requerimientos del proyecto están bajo el control absoluto del equipo.
10- No se monitorean los riesgos correctamente	Evitar	Verificar que durante el desarrollo del proyecto no se presenten los riesgos para los que se haya adoptado las estrategias Mitigar y Evitar.

2.2.3 Plan de desarrollo

A continuación se detallan las tareas definidas en el Plan de desarrollo con una breve descripción de las mismas:

Iniciación

- *Diseño del juego*: Redacción de Game Design Document (GDD). Documento en el cual se describe en detalle el diseño del juego, él mismo incluye las funcionalidades básicas como avanzadas para lograr los objetivos respetando las reglas definidas.

- *Análisis de Competidores*: Búsqueda y análisis de aplicaciones de entretenimiento que utilicen la geolocalización para su desarrollo.
- *Análisis de Riesgos*: Detección y análisis de los riesgos que pueden surgir en el desarrollo de una aplicación mediante la utilización de nuevas tecnologías. Además incluye la definición de las acciones a realizar ante los riesgos detectados.
- *Identificación de arquitecturas*
    - Investigación de factibilidad técnica: selección de los Frameworks más utilizados en el desarrollo de aplicaciones móviles para la plataforma Android y comparación entre los mismos. La comparación incluye los requisitos de instalación, lenguajes de programación utilizados, características del entorno de desarrollo, licenciamiento, documentación, etc. Con dicha comparación se pretende determinar el Framework a utilizar para el desarrollo de la aplicación.
    - Investigación técnica: investigación sobre cómo implementar las mecánicas básicas del juego detalladas en el documento Game Desing Document.
- *Identificación de casos de uso*: Detección de los actores e interacciones que pueden tener con la aplicación.

Elaboración

- *Elaboración de casos de uso*: Elaboración de los Diagramas que representan a los actores y sus interacciones con la aplicación
- *Definición de arquitectura*: elaboración del Diagrama de Arquitectura en el cual se representan los módulos/componentes que conforman la arquitectura de la aplicación. Especificación de los módulos/componentes principales, repositorios de datos, zonas de red y relación entre los módulos/componentes.
- *Definición de estándares*
    - Redacción de High Concept: documento que resume el concepto básico y los puntos claves del juego, permitiendo a cualquier persona

tener una idea rápida sobre que se trata el juego y que se puede esperar del mismo.

- Diseño de mecánicas avanzadas del juego: Definición de las mecánicas avanzadas del juego para finalizar el documento Game Design Document.

- *Identificación de clases y entidades*: Armado de Diagrama de Clases y Diagrama de Entidad Relación.

Construcción

- *Generación del ambiente de desarrollo (Cliente y Servidor)*: configuración del hardware e instalación del software necesario para el desarrollo del juego. Requisitos obtenidos del documento de Investigación de factibilidad técnica.
- *Desarrollo del módulo Servidor y módulo Cliente*: desarrollo del juego junto con el armado de toda la documentación necesaria para el mismo, como diagramas de clases, diagramas de secuencia, etc.
- *Pruebas unitarias*: ejecución de Pruebas Unitarias sobre los módulos desarrollados en el Cliente y en el Servidor.
- *Playtesting y feedback*: Testeo de la aplicación desarrollada.
- *Iteración para mejoras en base al feedback*: corrección de errores e implementación de mejoras detectadas en la etapa de testeo.
- *Playtesting, feedback y ajustes menores de balanceo*: Nueva etapa de testeo posterior a la etapa de corrección de errores e implementación de mejoras para realizar ajustes menores sobre el juego.

Transición

- *Implantación*: Publicación de la aplicación móvil y del Servidor.
- *Cierre de Proyecto*: Ajustes finales en la documentación referida al proyecto.

Desarrollo de Videojuego que utilice Geolocalización	**224 días**
Iniciación	**40 días**
Diseño del juego (Redacción de GDD)	20 días
Análisis de Competidores	5 días
Análisis de Riesgos	8 días
Identificación de arquitecturas	3 días
Identificación de casos de uso	4 días
Elaboración	**22 días**
Elaboración de casos de uso	5 días
Definición de arquitectura	8 días
Definición de estándares	1 días
Identificación de clases y entidades	8 días
Construcción	**150 días**
Iteración 1	55 días
Generación de ambiente de desarrollo Servidor	5 días
Generación de ambiente de desarrollo Cliente	10 días
Implementación módulos Módulo Central e Iniciador de Partidas en Servidor	15 días
Ejecución pruebas unitarias de Servidor	2 días
Desarrollo Cliente	20 días
Ejecución pruebas unitarias de Cliente	2 días
Diagramas de secuencia	1 días
Iteración 2	23 días
Implementación módulos Búsqueda de Partidas en Servidor	10 días
Ejecución pruebas unitarias de Servidor	1 días
Desarrollo Cliente	10 días
Ejecución pruebas unitarias de Cliente	1 días
Diagramas de secuencia	1 días
Iteración 3	23 días
Implementación módulos Control de Conexión y Controlador de Partidas	10 días
Ejecución pruebas unitarias de Servidor	1 días
Desarrollo Cliente	10 días
Ejecución pruebas unitarias de Cliente	1 días
Diagramas de secuencia	1 días
Iteración 4	21 días
Implementación capa de persistencia en Servidor	4 días
Ejecución pruebas unitarias de Servidor	1 días
Desarrollo Cliente	10 días
Ejecución pruebas unitarias de Cliente	1 días
Ejecución pruebas integrales	5 días
Iteración 5	13 días
Desarrollo correcciones en Servidor	5 días
Desarrollo correcciones en Cliente	5 días
Ejecución pruebas integrales	3 días
Iteración 6	10 días
Desarrollo correcciones en Servidor	2 días
Desarrollo correcciones en Cliente	2 días
Ejecución pruebas integrales	1 días
Playstesting y feedback	5 días
Iteración 7	5 días
Desarrollo correcciones en Servidor	2 días
Desarrollo correcciones en Cliente	2 días
Actualización de Documentación	1 días
Transición	**12 días**
Implantación	10 días
Cierre de Proyecto	2 días

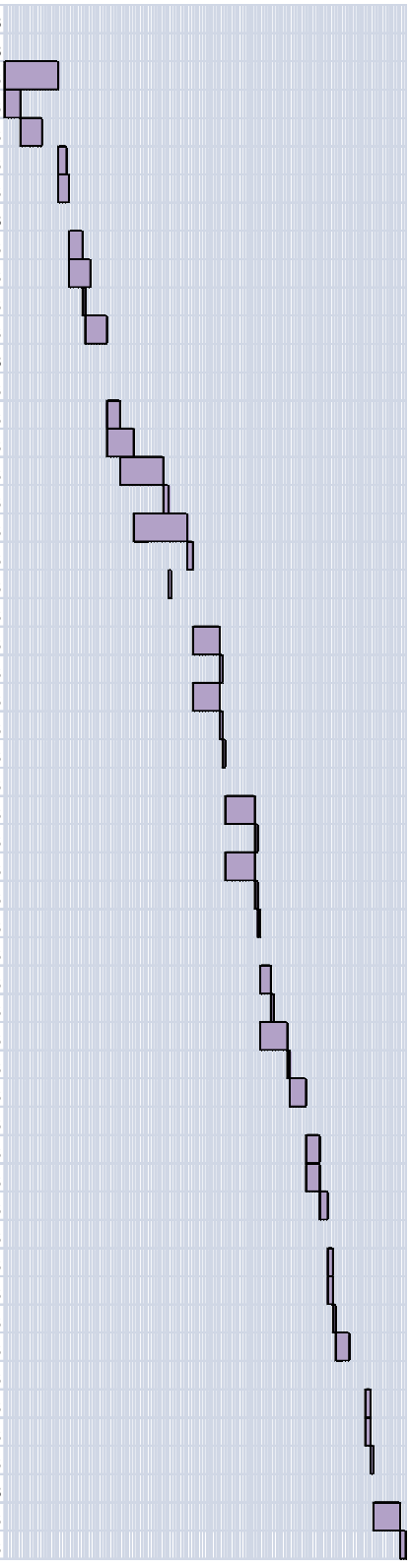

Figura 2 - Plan de Proyecto - Diagrama de Gantt

2.3 Herramientas de desarrollo

2.3.1 Análisis de alternativas

Antes de comenzar el desarrollo de una aplicación, se debe determinar la tecnología con la cual se va a trabajar. Para ello, se tienen que investigar las distintas herramientas disponibles y compararlas entre sí para seleccionar la que mejor se adapta a las necesidades y limitaciones del proyecto.

A continuación, se describen distintas herramientas contempladas para el desarrollo y se muestran cuadros comparativos de las características comunes que comparten entre sí.

Un Framework es un marco de trabajo que ofrece componentes como librerías, pero además provee plantillas o estructuras que definen el funcionamiento de las aplicaciones. La utilización de un Framework en el desarrollo de una aplicación implica un cierto coste inicial de aprendizaje, aunque a largo plazo es probable que facilite tanto el desarrollo como el mantenimiento de la misma. Por estos motivos, es importante elegir bien el Framework con el cual se va a trabajar ya que debe brindar las funcionalidades deseadas y no debe ser difícil de aprender y de interpretar.

Según el reporte "*State of Mobile App Developers 2016*" emitido por ***INMOBI*** en 2016, la plataforma con la mayor cantidad de desarrolladores de aplicaciones móviles es Android, seguida por Apple iOS.

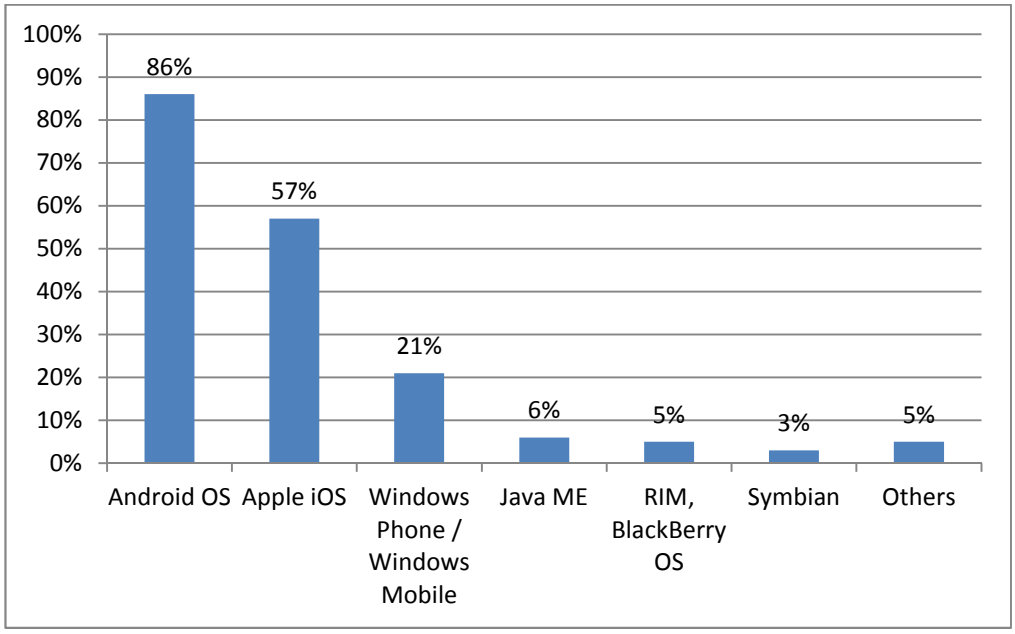

Figura 3 - Plataformas elegidas para el desarrollo de aplicaciones en 2016.

Fuente: *State of Mobile App Developers 2016*

Estos datos ayudan a confirmar la decisión de desarrollar la aplicación para la plataforma Android. El reporte mencionado además muestra los lenguajes más utilizados por los desarrolladores, algunos de los cuales permiten el desarrollo de aplicaciones para múltiple plataformas con el mínimo esfuerzo de adaptación, tal como HTML5 y JavaScript, ya que son interpretados por todos los navegadores móviles. La adecuación del código de la aplicación para las distintas plataformas está relacionada con el uso de funcionalidades propias de dicha plataforma, las cuales son mínimas.

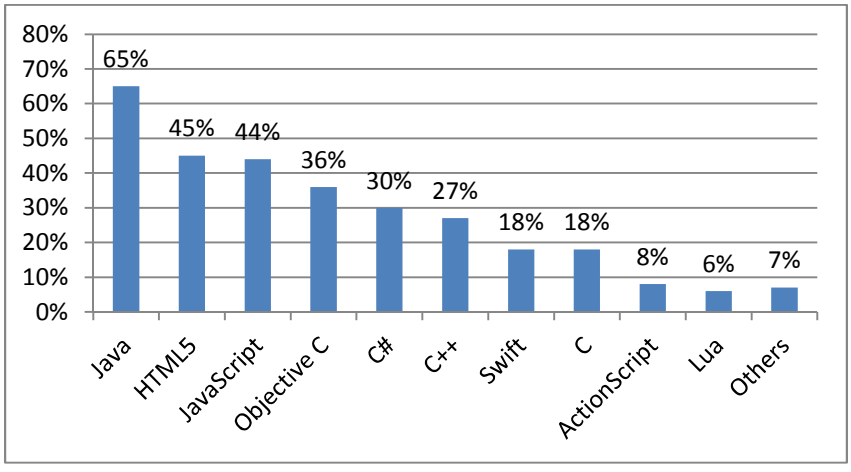

Figura 4- Lenguajes más utilizados.

Fuente: State of Mobile App Developers 2016

En base a los lenguajes mencionados, con mayor porcentaje de usuarios, se realizan varias comparaciones entre diferentes Frameworks basado en dichos lenguajes, para determinar cuál se adecua mejor a las necesidades del proyecto.

Instalación de los Frameworks

A la hora de instalar un Framework, es importante conocer qué se necesita previamente para poder utilizarlo al igual que la plataforma bajo la cual trabaja. Otro punto importante es la complejidad para su instalación, ya que puede demorar el comienzo del proyecto por retrasos en el armado del ambiente de desarrollo.

TABLA V: Requerimientos necesarios para la instalación de los Framework

Requisitos para la Instalación	Titanium Mobile	Android SDK	ADT Bundle Android	JQuery Mobile	PhoneGap	Sencha Touch	Dojo Mobile
Java JDK	X	X	X	X	X	X	X
Java JRE					X	Opcional	
Git	X						
Node.js	X				X		X
Registración	X						
Firefox para JavaScript debugging	X						
IDE		X		X	X	Eclipse. Opcional	X
Android Development Tools (ADT)		Si se utiliza Eclipse		X	X		X
Native Development Kit (NDK)		Opcional					
Dispositivo Virtual Android (AVD)		Opcional		Opcional			
Eclipse JDT Plugin		X					
Android SDK				X	X	Opcional	X
Librería PhoneGap				X			
Librería Cordova				Opcional	X		

Web Service						X	
Sencha CMD						X	
Ruby						X	
Rhino							X

En todos los casos, la instalación de los requisitos previos se debe hacer individualmente excepto si se utiliza Titanium Mobile, este Framework provee un instalador que se encarga de descargar e instalar cada uno de los componentes necesarios.

En cuanto a la plataforma de instalación no se diferencian los Frameworks.

Tabla VI: Plataformas donde se pueden instalar los Frameworks.

Plataforma de Instalación	Titanium Mobile	Android SDK	ADT Bundle Android	JQuery Mobile	PhoneGap	Sencha Touch	Dojo Mobile
Windows	X	X	X		X	X	
MAC OS	X	X	X		X	X	
Linux		X	X		X	X	
Independiente SO				X			X

En relación a la forma de instalación de cada uno de los Framework, se observan diferencias y en algunos casos la complejidad es mayor.

Tabla VII: Métodos de Instalación de los Frameworks

Instalación Framework	Titanium Mobile	Android SDK	ADT Bundle Android	JQuery Mobile	PhoneGap	Sencha Touch	Dojo Mobile
Wizard	X	X					
Descomprimir archivo			X			En el Web Server	
Línea de comando					X		
Copiar librerías				En el proyecto			En el proyecto
Documentación Oficial sobre instalación	X	X	X		X	X	X

Los Frameworks que presentan mayor complejidad para su instalación son PhoneGap, Sencha Touch y Dojo Mobile debido a la cantidad y complejidad de instalación de

sus requisitos previos, y por otro lado, JQuery Mobile por no contar en la documentación oficial con una guía de instalación.

Por lo tanto, los Frameworks con menor complejidad para instalar son Titanium Mobile, Android SDK y ADT Bundle Android.

Plataformas móviles soportadas

Muchos de los Frameworks tienen la característica de permitir construir un único código que es soportado por múltiples plataformas móviles, de manera que el desarrollador puede obtener diversas extensiones de la aplicación construida y cargarlas al mercado de los diferentes sistemas operativos compatibles, sin necesidad de realizar modificaciones en el código.

Tabla VIII: Plataformas móviles soportadas por los Frameworks

Plataformas Móviles Soportadas	Titanium Mobile	Android SDK	ADT Bundle Android	JQuery Mobile	PhoneGap	Sencha Touch	Dojo Mobile
Android	X	X	X	X	X	X	X
Apple iOS	X			X	X	X	X
Blackberry	X			X	X	X	X
Windows	X			X	X	X	X
Mobile Web	X			X			
Otros Mobile OS				X	X	X	

Android SDK y ADT Bundle Android no permiten el desarrollo de aplicaciones que soportan múltiples plataformas, con lo cual el resto de los Frameworks permiten migrar el código desarrollado para una plataforma a otra con una mínima adecuación del mismo.

Algunos de los Frameworks soportan más plataformas pero la mayoría soporta las principales plataformas móviles, Android y Apple iOS.

Lenguaje de Programación

El lenguaje de programación es un punto importante a la hora de elegir con qué Framework se va a trabajar. Dependiendo de la alternativa seleccionada, se utilizarán diferentes códigos y comandos que permitirán desarrollar, personalizar, y manejar las funciones de la aplicación. Por lo general, la elección se encuentra influenciada por el

conocimiento previo que tiene el desarrollador, ya que si el mismo domina un lenguaje de programación, puede elegir el Framework que trabaja con dicho lenguaje.

Tabla IX: Lenguajes de Programación utilizados por los Frameworks.

Lenguaje de Programación	Titanium Mobile	Android SDK	ADT Bundle Android	JQuery Mobile	PhoneGap	Sencha Touch	Dojo Mobile
Java		X	X				
HTML5	X			X	X	X	X
HTML					X		
JavaScript	X			X	X	X	X
CSS				X	X	X	X
JSON	X						
Ruby	X						
Python	X						
PHP Scripist	X						
Nativo (C/C++)	X	X	X		X		

X Lenguaje principal utilizado por el Framework

Como se observa en la tabla, Titanium Mobile es el Framework que soporta la mayor cantidad de lenguajes de programación. Además, tiene la particularidad que el Framework a la hora de compilar la aplicación convierte el código JavaScript en código nativo, con lo cual se independiza a la aplicación de los Browsers de los dispositivos móviles. La ventaja de la utilización de Lenguaje Nativo es la velocidad de la aplicación y mayor eficiencia en la utilización de los recursos del dispositivo.

Entorno de Desarrollo

Un entorno de desarrollo, llamado también IDE (sigla en inglés de Integrated Development Environment), es un software compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

Un IDE es un entorno de programación que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Este software facilita el trabajo del desarrollador proveyendo un marco de trabajo amigable para la mayoría de los lenguajes de programación.

Algunos Frameworks cuentan con su propio IDE para el desarrollo de las aplicaciones, y otros necesitan de un Entorno de Desarrollo facilitado por un tercero.

Tabla X: Propiedades de los Entornos de Desarrollo.
Fuente: Estudio comparativo de alternativas y Frameworks de programación, para el desarrollo de aplicaciones móviles en entorno Android

Entorno de Desarrollo	Titanium Mobile	Android SDK	ADT Bundle Android	JQuery Mobile	PhoneGap	Sencha Touch	Dojo Mobile
IDE Propio	SI	NO	Eclipse	NO	NO	NO	NO
Facilidad de diseñar la interfaz de usuario sin código	SI	SI	SI	Limitada	Limitada	Limitada	SI
Ayuda en la escritura del código (autocompletar comandos, alternativas, disponibles, etc.)	SI	SI	SI	Limitada	Limitada	Limitada	Limitada

De los Framework para múltiples plataformas, Titanium Mobile provee la mayor cantidad de facilidades para el desarrollo de aplicaciones. El IDE provisto por Titanium Mobile está basado en Eclipse, con lo cual si el desarrollador está acostumbrado a utilizar este último IDE no va a encontrar dificultad en el uso del IDE de Titanium Mobile.

PhoneGap, Sencha Touch y Dojo Mobile permiten el desarrollo de aplicaciones mediante IDEs de terceros, pero requieren que la creación del proyecto y la compilación de la aplicación se realicen mediante línea de comando utilizando las herramientas provistas por estos Frameworks.

Facilidad de desarrollo

Cada Framework dispone de una página web donde explica cómo funciona el software y la manera de utilizarlo para programar las aplicaciones. Por otra parte, solo algunos de ellos cuentan con soporte técnico especializado, tutoriales, ejemplos, foros, y muchas otras opciones que ayudan al programador a realizar la aplicación de manera más fácil.

Tabla XI: Propiedades y facilidades de los distintos Frameworks.
Fuente: Estudio comparativo de alternativas y Frameworks de programación, para el desarrollo de aplicaciones móviles en entorno Android

Nivel de Desarrollo	Titanium Mobile	Android SDK	ADT Bundle Android	JQuery Mobile	PhoneGap	Sencha Touch	Dojo Mobile
Documentos y Tutoriales oficiales	SI	SI	SI	Limitada	SI	SI	SI
Soporte Técnico oficial	SI	SI	SI	NO	NO	SI	NO
Acceso a casos de estudio, ejemplos y código fuente de aplicaciones	SI	SI	SI	Limitado	SI	SI	SI
Acceso a recursos y herramientas del teléfono (cámara, sensores, propiedades, etc.)	SI	SI	SI	Limitado	SI	Limitada	Limitado
Foros y Ayudas oficiales	SI	SI	SI	SI	SI	SI	SI

Todos los Frameworks ofrecen documentación sobre el uso del mismo, sin embargo, solo Android SDK y ADT Bundle Android ponen a disposición de los programadores la opción de soporte técnico de forma gratuita. Por otro lado, las diferentes alternativas cuentan con una comunidad activa de programadores dispuestos a ayudar y brindar soporte a aquellos que lo necesiten mediante el uso de foros y discusiones. Para acceder a accesorios y funcionalidades del teléfono, jQuery Mobile y Dojo Mobile deben hacer uso de las librerías de PhoneGap, mientras que Sencha Touch cuenta con una lista limitada de herramientas a las que pueden acceder.

Android SDK, ADT Bundle Android y Titanium Mobile representan los Frameworks con mejor documentación y ayuda al desarrollador para programar aplicaciones. Del mismo

modo son los que cuentan con el API más completo para acceder a elementos y accesorios del teléfono, así como el manejo de eventos del sistema.

Licencia de distribución

La licencia de distribución establece legalmente para qué puede ser utilizado el producto y de qué manera se lo puede distribuir. Además, determina si puede ser utilizado para aplicaciones comerciales o únicamente para aplicaciones que se distribuyan bajo la misma licencia.

Lo más importante de la licencia es si se debe pagar por el uso del producto o se lo puede utilizar de forma gratuita respetando las restricciones impuestas por la empresa distribuidora.

Tabla XII: Licencias de Distribución de los Frameworks.

<table>
<tr><th>Licencia</th><th>Titanium Mobile</th><th colspan="2">Android SDK</th><th>ADT Bundle Android</th><th>JQuery Mobile</th><th>PhoneGap</th><th>Sencha Touch</th><th>Dojo Mobile</th></tr>
<tr><td rowspan="2"></td><td rowspan="2">Open Source (Apache 2.0 license)</td><td colspan="3">Android Open Source Project</td><td rowspan="2">MIT license</td><td rowspan="2">Apache Software Foundation (ASF) under the name Apache Cordova</td><td>GNU GPL license v3</td><td rowspan="2">BSD and AFL license</td></tr>
<tr><td colspan="3">Apache Software License, Version 2.0</td><td>Comercial</td></tr>
</table>

Todos los Framework analizados cuentan con licencias gratuitas que pueden ser utilizadas en aplicaciones comerciales y no comerciales.

Algunos de los Frameworks se distribuyen bajo varias licencias debido a que tienen que respetar las licencias de las herramientas que utilizan para su funcionamiento.

2.3.2 Selección de herramientas

Servidor

La elección de la plataforma sobre la cual desarrollar el servidor fue simple. Las necesidades del proyecto son las siguientes:

- Comunicación mediante servicios SOAP.
- Conexión a base de datos.

- Manejo de hilos.

Teniendo en cuenta las mismas, se decidió utilizar el Framework .NET como plataforma de desarrollo para el servidor, y MS-SQL como plataforma para la base de datos. El único factor determinante en esta decisión fue la amplia experiencia profesional con la que cuentas ambos tesistas sobre dichas tecnologías, en comparación con cualquiera de las alternativas.

Cliente

Con las comparaciones realizadas se determina que los Frameworks que proveen la mayor cantidad de funcionalidades y facilidades para el desarrollo de aplicaciones son Android SDK, ADT Bundle Android y Titanium Mobile. Además, estos Frameworks ofrecen la mayor cantidad de documentación y son los que presentan la menor complejidad para su instalación.

Un punto importante es que estos Framework trabajan con el código nativo de los equipos móviles, con lo que se obtiene una mejor administración de los distintos dispositivos.

En relación al reporte mencionado al principio, la mayoría de los programadores desarrollan aplicaciones para las plataformas Android y Apple iOS. Esto se debe porque la mayoría de los equipos móviles en el mercado utilizan estas plataformas.

Analizando los puntos mencionados en los párrafos anteriores, se decidió realizar la aplicación utilizando el Framework Titanium Mobile de Appcelerator que permite en un futuro migrar la misma a otras plataformas con el mínimo esfuerzo de adecuación.

CAPITULO III: Diseño

La tesis versa sobre el desarrollo un videojuego de estrategia por turnos que utiliza geolocalización combinada con un servicio de mapas para enfrentar a dos jugadores entre sí en un escenario basado en el mapa del lugar en el que se encuentran físicamente.

Los jugadores, que se encuentran ejecutando una aplicación en su dispositivo móvil y a su vez se encuentran en ubicaciones diferentes, se enfrentan entre sí conectados a través de internet, intercambiando por este medio las acciones que realizaron en la aplicación ejecutándose en su dispositivo.

En su conjunto, el juego deberá poder realizar lo siguiente:

1. La implementación de las reglas del juego, descriptas en el Documento de Diseño de Juego (GDD).
2. Búsqueda de partidas entre jugadores, de acuerdo a las limitaciones geográficas descriptas en el Documento de Diseño de Juego (GDD).
3. Comunicación entre los dos jugadores que se encuentran participando de una partida.
4. Guardado de datos sobre el historial de partidas jugadas, ganadas y perdidas por cada jugador.
5. Autenticación de usuarios con el sistema

3.1 Diseño del juego

3.1.1 High Concept

Definición

El documento High Concept es definido por Andrew Rollings y Ernst Adams como un documento corto de entre dos y cuatro páginas en las que se plasman de forma breve y concisa el concepto general de un proyecto para poder dar a conocer la propuesta inicial.

Adams va más allá y en “Fundamentals of Game Design” detalla cuáles son algunos de los puntos básicos que un buen High Concept debería incluir:

- Declaración: una descripción corta sobre de qué trata del juego.

- Rol del jugador: describir que rol cumple el jugador en el juego. ¿Está pretendiendo ser algo o alguien? ¿Participa en más de un rol? ¿Hay un avatar que lo represente?
- Gameplay: hay que hacer una breve y concisa descripción de cómo se juega al juego y que modos de juego tendrá el mismo (un solo jugador, multijugador competitivo o cooperativo, etc.).
- Género: a lo largo de los años ha sido posible desarrollar etiquetas que describen a los juegos en base a sus mecánicas. Hay que informar cuales describen mejor al juego, y en caso que ninguna lo haga describir porque.
- Plataforma: describe en que sistemas va a jugarse: consolas, celulares, computadoras, realidad virtual.
- Audiencia: informar a quienes se apunta como jugadores del juego: hombres, mujeres, edades, etc.
- Estética: hay que describir cómo se verá el mundo del juego y el modo en el que lo experimentará el jugador.
- Flujo: una simple descripción del flujo de juego, desde el principio de la partida hasta el final. En caso de que corresponda, se hará una breve mención a la historia del juego.
- Marketing: dado que un juego es un producto que tiene que ser vendido, hay que describir cuáles son los potenciales competidores del juego, qué es lo que hace al juego destacarse por sobre el resto, oportunidades de mercado, marketing y merchandising (en caso de que sea posible), modelo de monetización.

Aplicación

Por su extensión y estructura el High Concept se incluyó dentro del ANEXO IV.

3.1.2 Game Design Document

Definición

Según Ernst Adams, el proceso de escribir un documento transforma una idea en un plan. Aun cuando nadie lo lea, una idea escrita simboliza una decisión tomada y una conclusión alcanzada. Si una funcionalidad de un juego no está descripta por escrito, existe una alta probabilidad de que haya sido pasado por alto y que alguien tenga que inventarla

sobre la marcha o, lo que es aún peor, que cada integrante del equipo tenga una opinión diferente de como tendría que funcionar.

El Game Design Document, de ahora en más GDD para abreviar, es un documento que recaba todos los detalles relativos al diseño del juego, ya sean jugabilidad, historia, personajes, posibilidades, entorno, ambientación, controles, etc. describiendo cada uno de estos al máximo de detalle. El GDD es un documento no técnico que ha de servirle de guía a todos aquellos involucrados en el desarrollo del videojuego (sean desarrolladores, artistas, sonidistas, actores de voces y todo otro rol que pueda participar dependiendo de la envergadura del proyecto).

Se puede equiparar al GDD a un documento de Especificación de Requisitos de Software (ERS), aunque mucho más laxo y menos estructurado, dado que no existe ninguna definición formal ni estándar sobre su estructura, dado lo difícil que puede resultar trasladar las mecánicas de un juego a las secciones definidas por un ERS, aunque si existe cierto consenso general al cual se llegó a partir de diversos casos exitosos de la industria. Damion Schubert, un reconocido diseñador con años de experiencia en títulos que fueron éxitos comerciales como también aclamados por la crítica especializada, mencionó en una lectura algunos indicadores de un mal documento de diseño:

- Interconexión entre muchos sistemas, haciendo que sea muy difícil rastrear el origen de cada idea expresada.
- Exceso de diseño, incluyendo muchas funcionalidades y características que jamás llegaran a la versión final del juego.
- Documentos que no están pensados de forma iterativa. El proceso de encontrar mecánicas que sean divertidas requiere de iteraciones, y el documento que describe dichas mecánicas también tiene que serlo. No es posible que desde su primera versión un documento sea perfecto.
- Documentos que no se mantiene actualizados durante el ciclo de desarrollo.

Según Schubert hay ocho puntos básicos a tener en cuenta a la hora de diseñar un GDD:

- Enfocado: el GDD será leído por los productores, por los diseñadores, por los artistas y por todos aquellos que participen en el proceso de la creación del videojuego, por lo que tiene que poder hablar en un lenguaje común que todos puedan entender. Sin embargo, hay que tener en cuenta que la audiencia más importante son los programadores, ya que ellos serán los responsables de que el videojuego cobre vida.
- Corto: un documento de poca longitud es más fácil de leer, escribir y mantener.
- Priorizado: el documento tiene que marcar cuáles son los puntos más importantes, tanto a nivel funcionalidad como a nivel estético.
- Ilustrado: siguiendo el dicho que "una imagen vale más que mil palabras", hay que recurrir siempre que sea posible a imágenes para poder describir las ideas, ya sean garabatos, maquetas, diagramas de colores o imágenes de otros videojuegos que puedan explicar la idea en cuestión.
- No decirle a otros cómo hacer su trabajo: un GDD tiene que explicar "qué hacer" pero no "cómo hacerlo".
- Legible: un buen GDD es cómodo para leer. Tiene suficiente espacio en blanco, letra de formato agradable, buen uso de negritas, cursivas y subrayado para remarcar ideas y correcta identificación de encabezados y títulos.
- Sin redundancia: si el documento necesita información que ya fue descripta anteriormente o está descripta en otro documento, hace referencia a la misma en lugar de repetirla. Esto ayudará a mantener una longitud baja y a evitar errores de contradicción.
- Sin ambigüedades: un GDD tiene que estar escrito en lenguaje declarativo. No tiene que contener expresiones como "tal vez", "probablemente" o "puede que".

Aplicación

Por su extensión y estructura el GDD se incluyó dentro del ANEXO V.

3.2 Diseño de alto nivel

3.2.1 Arquitectura de la solución

Para poder llevar a cabo los objetivos, el juego será diseñado bajo la arquitectura de cliente-servidor, en donde cada jugador contará con una aplicación cliente propia y un solo servidor central actuará como intermediario entre todos los clientes permitiéndole a los mismos jugar partidas, encargándose no solo de interconectar a los dos jugadores que van a jugar la partida, sino también haciendo de intermediario en las mismas.

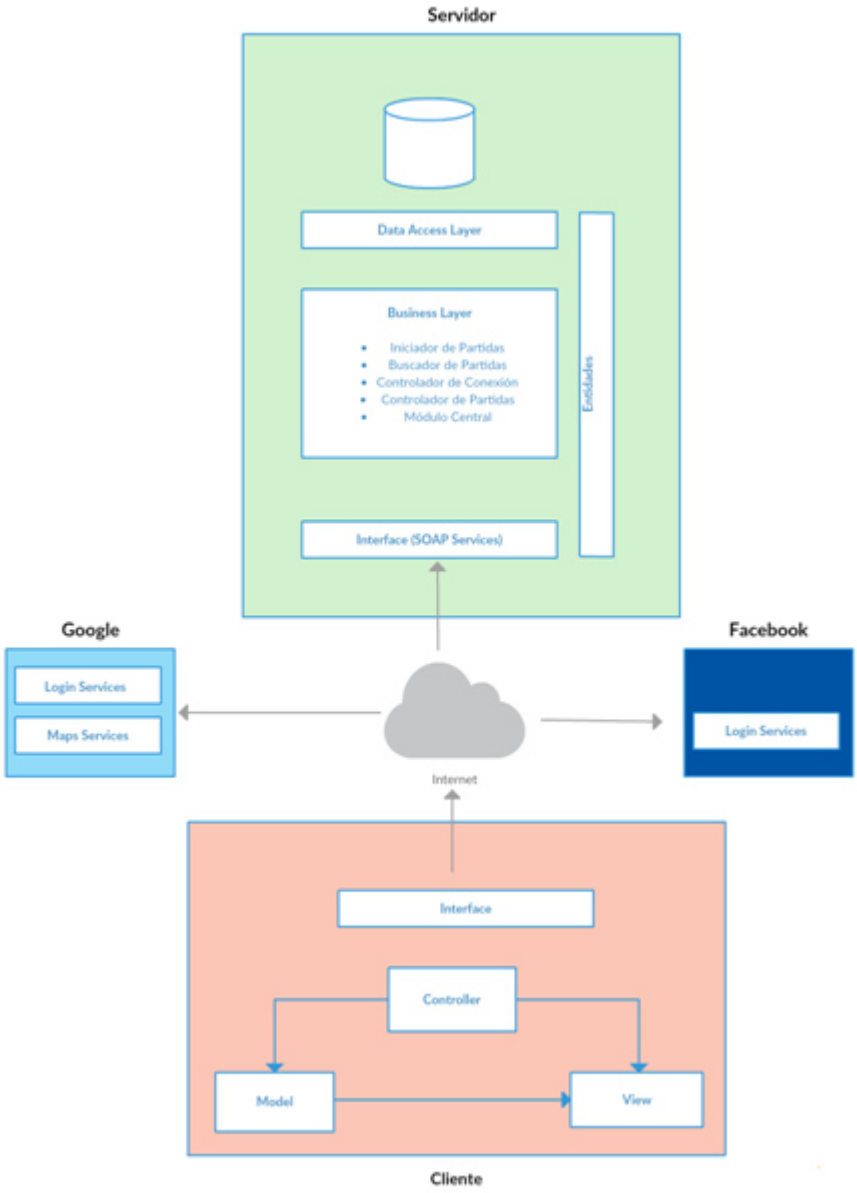

Figura 5 – Diagrama de Arquitectura

3.2.2 Servidor

Módulos de Negocio

Se listan a continuación los diferentes módulos de negocio con los que cuenta la aplicación servidor.

- **Iniciador de Partidas:** contiene la lógica a partir de la cual, teniendo dos jugadores listos, se prepara la partida para los mismos, consultando los puntos de interés cercanos a cada uno y recopilando los que quedarán finalmente en el mapa de la partida (según lo definido en el GDD).
- **Búsqueda de Partidas**: el objetivo de este módulo es el de estar verificando constantemente el listado de jugadores que se encuentran buscando una partida y cuando se encuentra a dos que cumplan las condiciones de inicio de partida (definidas en el GDD) llama al módulo Iniciador de Partidas para dar comienzo a la misma.
- **Control de conexión:** este módulo se encarga de controlar que ninguno de los dos jugadores haya abandonado la partida en curso, y finalizar la partida comunicándole al otro jugador en caso afirmativo.
- **Controlador de Partidas**: este módulo se encarga de que cada jugador reciba el listado de acciones realizadas por su rival, permitiendo que la partida se lleve a cabo.
- **Modulo Central**: el objetivo de este módulo es centralizar la comunicación entre los jugadores y los diferentes módulos del servidor, exponiendo los métodos que expone la capa de interfaz mediante Servicios SOAP.

Interfaces

Se listarán a continuación las interfaces expuestas por el servidor.

- **Buscar Partida:** método llamado por los usuarios para comenzar la búsqueda de una partida.
- **Controlar Búsqueda:** método llamado periódicamente por los usuarios para controlar el estado de la búsqueda de una partida. Es necesario que primero se haya llamado a Buscar Partida.
- **Cancelar Búsqueda:** método llamado por los usuarios para cancelar la búsqueda de una partida. Es necesario que primero se haya llamado a Buscar Partida.

- **Postear Acciones:** método llamado por los jugadores cada vez que realizan una acción, para poder informarle al jugador rival sobre las mismas.
- **Solicitar Acciones:** método llamado por los jugadores periódicamente para poder informarse sobre las acciones realizadas por el jugador rival.
- **Login:** método llamado por los usuarios una vez que se autenticaron con Facebook o Google, informándole al servidor su información y recuperando los datos (según lo especificado en el GDD).
- **Buscar Historia:** método llamado por los usuarios para recuperar el historial completo de partidas jugadas.

Servicios Externos

La aplicación consume los siguientes servicios externos:

- **Servicio de Mapas de Google:** permite recabar los mapas en base a la posición del usuario, calcular las rutas hacia los puntos seleccionados en diferentes modalidades (de a píe o en automóvil, según lo especificado en el GDD) y las ubicaciones de los puntos de interés cercanos a los usuarios (según especificado en el GDD).
- **Servicio de Autenticación de Facebook:** permite recabar los datos básicos de la persona (nombre de usuario en Facebook y Nombre completo son los datos requeridos por nuestra aplicación).
- **Servicio de Autenticación de Google:** permite recabar los datos básicos de la persona (nombre de usuario en Google y Nombre completo son los datos requeridos por nuestra aplicación).

3.2.3 Cliente

Entidades

Se listarán a continuación las Entidades que conforman al Cliente

- **Unit**: Clase abstracta de la cual heredan los atributos y métodos las clases Tank, Troop y Motorcycle.
- **Troop**: clase que representa los atributos y acciones de las Unidades del Tipo Tropa.

- **Motorcycl**e: clase que representa los atributos y acciones de las Unidades del Tipo Moto.
- **Tank**: clase que representa los atributos y acciones de las Unidades del Tipo Tanque.
- **Player**: clase utilizada para crear la instancia del Jugador y del Enemigo.
- **CommandCenter**: clase que representa los atributos y acciones de los Centros de Comando.
- **PointOfConquest**: clase que representa los atributos de los Puntos de Conquista.
- **Card**: clase que representa los atributos y acciones de las Cartas de Acción.
- **Message**: clase que representa los atributos de los Mensajes entre el Cliente y el Servidor.

Controladores

Se listarán a continuación los Controladores utilizado por el Cliente:

- **Controller**: clase encargada de administrar todas las acciones realizadas por el Usuario sobre la aplicación Cliente.
- **ServerController**: clase encargada de consultar al Servidor, a través de una interface SOAP, sobre las acciones realizadas por el Contrincante.
- **ServerInformer**: clase encargada de informar al Servidor, a través de una interface SOAP, las acciones realizadas por el Jugador.

Módulos Auxiliares

- **Geo**: clase que contiene los métodos para el cálculo de distancia entre coordenadas.
- **GoogleAuth**: clase que contiene los métodos necesarios para validar las cuentas de Google+.

Vistas

En general cada instancia de una Entidad cuenta con 2 vistas, una se utiliza para visualizar el estado de la instancia y la otra vista se utiliza para representar a las instancias en el Mapa.

A continuación se listan las vistas:

- **CardDetails**: vista utilizada para mostrar el detalle de las Cartas de Acción pertenecientes al Usuario y poder seleccionarlas para su uso.
- **CommandCenterDetails**: vista utilizada para mostrar el estado y acciones de los Centros de Comando.
- **CommandCenterView:** vista que representa una instancia de la Entidad Centro de Comando en el Mapa.
- **MapDetails:** vista utilizada para mostrar las acciones referidas a la partida, por ejemplo Fin de Turno.
- **MapView:** vista utilizada para mostrar el mapa y posicionar al Jugador, Enemigo, Tropas, Motos, Tanques, Centros de Comando y Puntos de Conquista.
- **UnitDetails:** vista utilizada para mostrar el estado y acciones de una instancia de la Entidad Tropa, Moto o Tanque. Se utiliza la misma vista porque tienen los mismos atributos y acciones las Entidades.
- **MotorcycleView:** vista que representa una instancia de la Entidad Moto sobre el Mapa.
- **TankView:** vista que representa una instancia de la Entidad Tanque sobre el Mapa.
- **TroopView:** vista que representa una instancia de la Entidad Tropa sobre el Mapa.
- **PlayerDetails:** vista utilizada para mostrar el estado y acciones de una instancia de la Entidad Player. En el caso que la instancia represente al Enemigo, no se muestran las acciones.
- **PlayerView:** vista que representa una instancia de la Entidad Player sobre el Mapa; se utiliza tanto para el Jugador como para el Enemigo.

Mensajes

Se listaran a continuación los mensajes enviados al Servidor por parte del Cliente utilizando la interfaz SOAP.

- **End Turn**: le indica al servidor que el Usuario dio por finalizado su Turno.
- **Win Match**: le indica al servidor que el Usuario ganó la Partida.
- **Cancel Search Game**: le indica al Servidor que cancele la búsqueda de un Contrincante para iniciar una Partida.

- **Notify Creation**: le indica al Servidor que se creó una Tropa, Moto, Tanque o Centro de Comando.
- **Notify Movement:** le indica al Servidor el desplazamiento que realizó una Tropa, Moto o Tanque.
- **Notify Unit Attack:** le indica al Servidor que se realizó un ataque sobre una de las Tropas, Motos o Tanques del Enemigo.
- **Notify Command Center Attack**: le indica al Servidor que se realizó un ataque sobre uno de los Centros de Comando del Enemigo.
- **Notify Conquest**: le indica al Servidor que se conquistó un Punto de Conquista.

3.3 Casos de Uso

Un caso de uso es un documento narrativo que describe el comportamiento de un sistema desde el punto de vista de un actor. El actor es una entidad externa del sistema que participa en el caso de uso, suelen ser seres humanos o cualquier tipo de sistema.

3.3.1 Inicio de Sesión del Jugador

TABLA XIII: Caso de uso Inicio de Sesión del Jugador

Flujo Normal	Actor	Sistema
	1) El Jugador abre la aplicación en el dispositivo móvil.	
		2) El Sistema le solicita al Jugador iniciar sesión con su cuenta de Facebook/Google.
	3) El Jugador ingresa su usuario y contraseña de Facebook/Google.	
		4) El Sistema valida ante Facebook/Google el usuario y contraseña.
		5) El Sistema le envía al Servidor Central el ID y USERNAME devuelto por Facebook/Google.

		6) El Servidor Central verifica si existe el usuario y si no lo crea. Devuelve la confirmación del inicio de sesión. 7) El Sistema visualiza el Menú Principal y solicita que se seleccione una de las siguientes opciones: ***Buscar Partida***, ***Nueva Ubicación o Salir***.
	8) El Jugador selecciona la Opción ***Buscar Partida***.	Ver Caso de Uso Buscar Partida
Flujo Alternativo	**Actor**	**Servidor Central**
Opción ***Nueva Ubicación***	8) El Jugador selecciona la Opción ***Nueva Ubicación***. 11) El Jugador selecciona la nueva ubicación que desea utilizar y la confirma.	9) El Sistema visualiza el mapa centrado en la ubicación actual de Jugador 10) El Sistema le solicita al Jugador que seleccione un punto en el mapa para ser utilizado como la ubicación del Jugador. 12) El Sistema persiste la nueva ubicación del Jugador para ser utilizada en la Búsqueda de la Partida.

		13) El Sistema visualiza el Menú Principal y solicita que se seleccione una de las siguientes opciones: ***Buscar Partida***, ***Nueva Ubicación*** o ***Salir***.
Opción ***Salir***.	8) El Jugador selecciona la Opción ***Salir***.	9) El Sistema cierra la aplicación.

3.3.2 Buscar Partida

TABLA XIV: Caso de uso Buscar Partida

Flujo Normal	Actor	Servidor Central
	1) El Jugador selecciona la Opción ***Buscar Partida***.	2) El Sistema le informa al Servidor Central, el cual guarda la información del Jugador (ID y posición) en un Listado de Jugadores Buscando Partidas (LJBP) 3) El Servidor Central recorre periódicamente el LJBP buscando un par de jugadores que se encuentren dentro de los radios de distancia mínimos y máximos. 4) Cuando el Servidor Central encuentra un par de jugadores dentro de los radios de distancia mínimos y máximos, crea una partida y la introduce dentro del

	5) Periódicamente el Jugador verifica el estado de la Búsqueda.	Listado de Partidas Pendientes de Confirmación (LPPC). 6) El Sistema le avisa al Servidor Central. Si el Servidor Central encuentra el ID del jugador dentro del LPPC, le devuelve el ID de la Partida al jugador y la da por iniciada sin esperar la confirmación del otro jugador. 7) El Servidor Central pasa la partida del LPPC al Listado de Partidas en Curso (LPC). 8) El Sistema visualiza el mapa junto la posición de los Jugadores y Puntos de Conquista. Inicia el flujo del juego, permitiéndole al Jugador realizar sus acciones correspondientes.
Flujo Alternativo	**Actor**	**Servidor Central**
Cancelación de Búsqueda	5) El Jugador selecciona la Opción ***Cancelar Búsqueda***	6) El Sistema le avisa al Servidor Central, el que busca al Jugador en el LJBP y lo quita del mismo.
Cancelación de Búsqueda (cuando se encuentra una Partida)	5) El Jugador selecciona la Opción ***Cancelar Búsqueda***	6) El Sistema le avisa al Servidor Central, el cual busca al Jugador en

		el LJBP y no lo encuentra, lo cual significa que ya se encontró una Partida para el mismo. Luego, le avisa al Sistema que no puede Cancelar la búsqueda y le devuelve la Partida encontrada, siguiendo con el Paso 6) del Flujo Normal

3.3.3 Creación de Partida

TABLA XV: Caso de uso Creación de Partida

Flujo Normal	Sistema	Servidor Central
	1) A partir de la verificación de la búsqueda de partida del jugador, el Servidor Central encuentra el ID del jugador dentro del LPPC y da por iniciada la partida sin esperar la confirmación del otro jugador.	2) El Servidor Central obtiene los Puntos de Interés (PI) de ambos jugadores, consultando el servicio de Google Maps. 3) El Servidor Central calcula la cantidad máxima (N) de Puntos de Conquista (PC) en base a la distancia entre ambos jugadores. 4) El Servidor Central busca los PI comunes a ambos jugadores y los agrega a una lista de PC. 5) El Servidor Central le asigna un valor a cada PI restantes, basado en

		la distancia entre cada punto y ambos jugadores y ordena la lista en forma ascendente, agregándolos a la lista de PC 6) El Servidor Central devuelve la Partida inicializada, con el listado de PC, los ID de los jugadores que participan, el jugador a quien le corresponde el primer Turno y el ID propio de la partida.
	7) El Sistema recibe la Partida e inicia el flujo del juego, permitiéndoles a cada Jugador realizar sus acciones correspondientes.	

3.3.4 Acciones del Jugador

TABLA XVI: Caso de uso Acciones del Jugador

Flujo Normal	Actor	Sistema
	1) El Jugador "A" selecciona el icono de ***Jugador*** en el Mapa.	
		2) El Sistema muestra las acciones que puede realizar este tipo de objeto: ***Crear Unidad***, ***Activar Tarjeta*** y ***Finalizar Turno***.
	3) El Jugador "A" selecciona la opción ***Crear Unidad***.	
		4) El Sistema verifica que el Jugador "A" tenga la cantidad de ***Puntos Monetarios*** necesarios para crear una nueva Unidad.

		5) El Sistema crea un nuevo objeto del tipo Unidad y lo ubica en la lista de Unidades a crear de Jugador, ya que cada tipo de unidad requiere una cantidad de Turnos para poder ser utilizada y que aparezca en el mapa. 6) El Sistema le descuenta al Jugador "A" la cantidad de ***Puntos Monetarios*** que costó crear la Unidad.
Flujo Alternativo	**Actor**	**Sistema**
El Jugador no cuenta con la cantidad necesaria de ***Puntos Monetarios***.		5) El Sistema le informa al Jugador que los ***Puntos Monetarios*** no alcanzan para realizar la operación seleccionada.
Opción ***Activar Tarjeta.***	3) El Jugador "A" selecciona la opción ***Activar Tarjeta.*** 6) El Jugador "A" selecciona una Tarjeta.	4) El Sistema verifica si está cargada la lista de Tarjetas. Si no lo está, obtiene la lista desde el Servidor Central. 5) El Sistema visualiza la lista de Tarjetas disponibles, son aquellas que no están activas. 7) El Sistema marca como activa la Tarjeta seleccionada e informa al Servidor Central la Tarjeta que se

		activó. 8) El Sistema cierra la ventana con el listado de las Tarjetas.
El Jugador "A" cancela la acción ***Activar Tarjeta***.	7) El Jugador "A" cancela la selección de una Tarjeta	8) El Sistema cierra la ventana con el listado de las Tarjetas.
Opción ***Finalizar Turno***.	3) El Jugador "A" selecciona la opción ***Finalizar Turno***.	4) El Sistema le informa al Servidor Central que el Jugador "A" finalizó la cantidad de acciones en el Turno actual. 5) El Servidor Central le informa al Jugador "B" que es su Turno para realizar acciones.

3.3.5 Iniciar Turno

TABLA XVII: Caso de uso Iniciar Turno

Flujo Normal	Actor	Sistema
		1) El Servidor Central recibe el mensaje de ***Finalizar Turno*** por parte del Jugador “B” y le indica al Sistema que es el Turno del Jugador “A” 2) El Sistema restablece la cantidad de ***Puntos de Acción*** y ***Puntos Monetarios*** del Jugador “A”

		3) El Sistema recorre la lista de todas las Unidades que se están construyendo e incrementa en uno el tiempo de construcción de cada una. 4) El Sistema posiciona en el mapa aquellas Unidades que completaron su tiempo de construcción. 5) El Sistema habilita las acciones que puede realizar el Jugador sobre sus objetos y le notifica al Jugador "A" que es su tuno para jugar.
	6) El Jugador "A" comienza a realizar las acciones sobre sus objetos.	

3.3.6 Acciones de Unidad

TABLA XVIII: Caso de uso Acciones de la Unidad

Flujo Normal	Actor	Sistema
	1) El Jugador "A" selecciona una ***Unidad*** en el Mapa.	
		2) El Sistema muestra el estado de la Unidad y las acciones que puede realizar este tipo de objeto: ***Atacar y Avanzar***.
	3) El Jugador "A" mantiene presionado sobre el icono de la Unidad y lo arrastra en el mapa hasta donde desea ubicarlo.	
		4) El Sistema almacena las coordenadas donde el Jugador "A" desea trasladar a la Unidad.

		5) El Sistema calcula la ruta desde la posición original hasta la nueva posición definida por el Jugador "A" utilizando los servicios de Google Maps. 6) El Sistema vuelve a posicionar la Unidad en su ubicación original y visualiza en el mapa la ruta a seguir por la misma.
	7) El Jugador "A" selecciona la opción ***Avanzar***	
		8) El Sistema verifica si el Jugador cuenta con la cantidad necesaria de ***Puntos de Acción*** para poder hacer avanzar a la Unidad. 9) El Sistema le descuenta al Jugador "A" la cantidad de ***Puntos de Acción*** necesarios para realizar la acción. 10) El Sistema desplaza la Unidad una distancia X sobre la ruta. Cada Unidad tiene la posibilidad de desplazarse una distancia X por cada vez que se realiza la acción Avanzar. 11) El Sistema le informa al servidor Central del desplazamiento de la Unidad.

		12) El Servidor Central le avisa al Sistema del Jugador "B" que la Unidad Y del Jugador "A" se desplazó a una nueva posición para que refresque el mapa.
Flujo Alternativo	**Actor**	**Sistema**
El Jugador no cuenta con la cantidad necesaria de ***Puntos de Acción***.		9) El Sistema le informa al Jugador que los ***Puntos de Acción*** no alcanzan para realizar la operación seleccionada.
Opción ***Atacar.***	3) El Jugador "A" selecciona la opción ***Atacar.*** 5) El Jugador "A" selecciona el objetivo a atacar.	4) El Sistema solicita al Jugador "A" que seleccione el objetivo a atacar. 6) El Sistema verifica que el objetivo seleccionado se encuentre a la distancia adecuada para realizar el Ataque. 7) El Sistema verifica si el objetivo pertenece al Jugador contrincante o es un Punto de Conquista. 8) El objetivo pertenece al jugador contrincante, entonces el Sistema verifica que el Jugador "A" tenga la

		cantidad de ***Puntos de Acción*** necesarios para realizar un Ataque. 9) El Sistema le descuenta al Jugador "A" la cantidad de ***Puntos de Acción*** necesarios para realizar la acción. 10) El Sistema le descuenta una cantidad X de puntos de vida al objetivo atacado y una cantidad Y de puntos de vida a la Unidad que realizó la acción en forma de contraataque. 11) El Sistema le informa de la acción al Servidor Central para que éste le informe al Jugador "B". 12) El Sistema verifica si el objetivo atacado llegó a 0 puntos de vida.
El objetivo seleccionado para ser atacado no se encuentra a la distancia adecuada		7) El Sistema le informa al Jugador "A" que el objetivo a atacar no se encuentra dentro del radio permitido para ser atacado por su Unidad.
El objetivo seleccionado para ser atacado es un ***Punto de Conquista***.		8) El Sistema crea un nuevo ***Centro de Comando*** en la ubicación del Punto de Conquista y lo agrega a la lista de Centros de Comandos del Jugador "A".

El Jugador no cuenta con la cantidad necesaria de ***Puntos de Acción***.		9) El Sistema le informa al Jugador que los ***Puntos de Acción*** no alcanzan para realizar la operación seleccionada.
El objetivo atacado llegó a 0 puntos de vida		12) El Sistema verifica que tipo de objetivo es: ***Unidad***, ***Centro de Comando*** o ***Jugador***. 13) El objetivo es una ***Unidad***, entonces el Sistema elimina la Unidad del mapa y le envía un mensaje al Servidor Central informando que se tiene que eliminar la Unidad. 14) El Servidor Central le informa al Jugador "B" que fue derrotada su Unidad.
El objetivo es un ***Centro de Comando***.		12) El objetivo es un ***Centro de Comando***, entonces el Centro de Comando pasa a ser del Jugador "A". El Sistema agrega el Centro de Comando a la lista de Centros de Comandos del Jugador "A". 13) El Sistema le informa al Servidor Central que el Jugador "A" conquistó el Centro de Comando. 14) El Servidor Central le informa

		al Jugador "B" que el Centro de Comando fue conquistado por el Jugador "A" y lo elimina de la lista de Centros de Comandos del Jugador "B".
El objetivo es el ***Comando Central*** del Jugador contrincante.		12) El objetivo es el ***Comando Central*** del Jugador contrincante. Entonces el Jugador "A" ganó la partida. 13) El Sistema le informa al Servidor Central que el Jugador "A" ganó la partida. 14) El Sistema guarda en la base de datos la fecha y hora de la partida, y quien fue el jugador vencedor y el jugador derrotado. 15) El Sistema calcula los puntos de experiencia ganados por el Jugador vencedor y le suma ese puntaje.

3.3.7 Acciones de Objeto Centro de Comando

TABLA XIX: Caso de Uso Acciones de los Centros de Comando

Flujo Normal	Actor	Sistema
	1) El Jugador "A" selecciona el icono de ***Centro de Comando*** en el Mapa.	2) El Sistema muestra las acciones que puede realizar este tipo de objeto: ***Crear Unidad***.

	3) El Jugador "A" selecciona la opción ***Crear Unidad***.	4) El Sistema verifica que el Jugador "A" tenga la cantidad de ***Puntos Monetarios*** necesarios para crear una nueva Unidad. 5) El Sistema crea un nuevo objeto del tipo Unidad y lo ubica en la lista de Unidades a crear del Centro de Comando, ya que cada tipo de Unidad requiere una cantidad de Turnos para poder ser utilizada y que aparezca en el mapa. 6) El Sistema le descuenta al Jugador "A" la cantidad de ***Puntos Monetarios*** que costó crear la Unidad.
Flujo Alternativo	**Actor**	**Sistema**
El Jugador no cuenta con la cantidad necesaria de ***Puntos Monetarios***.		5) El Sistema le informa al Jugador que los ***Puntos Monetarios*** no alcanzan para realizar la operación seleccionada.

3.3.8 Detectar Desconexión del Jugador en Turno

TABLA XX: Caso de uso Detectar desconexión del Jugador en Turno

Flujo Normal	Sistema	Servidor Central
	1) El Sistema del Jugador en Turno envía un mensaje correspondiente a alguna acción del Jugador en Turno.	

		2) El Sistema le avisa al Servidor Central, que guarde la Hora en la que se recibió el último mensaje para el Jugador en Turno. 3) El Servidor Central recorre periódicamente el Listado de Partidas en Curso (LPC) verificando cuando se recibió el último mensaje del jugador en Turno. 4) Si el tiempo transcurrido desde que se recibió el último mensaje es mayor al estipulado para controlar y el flag de desconexión no se encuentra activo, se carga un mensaje de PING para que el jugador en Turno responda y se activa un flag de desconexión.
	5) El Sistema del Jugador en Turno recibe el mensaje de PING por parte del Servidor Central y lo responde.	6) El Servidor Central recibe la respuesta al mensaje de PING. Luego, se desactiva el flag de desconexión y se actualiza la Hora del último mensaje recibido para el Jugador en Turno.
Flujo Alternativo	**Sistema**	**Servidor Central**
El tiempo transcurrido es menor al de control		4) El tiempo transcurrido desde que se recibió el último mensaje es

		menor al estipulado para controlar, luego el Servidor Central no realiza ninguna acción.
El Flag de desconexión se encuentra activo	6) El Sistema del Jugador Rival verifica de forma periódica si hay nuevas Acciones del Rival. 8) El Sistema del Jugador Rival recibe el Listado de Acciones del oponente solicitado y detecta que entre los mensajes recibidos se encuentra el de Finalización de Partida por Desconexión del Oponente. Luego, el Sistema	4) Si el tiempo transcurrido desde que se recibió el último mensaje es mayor al estipulado para controlar y el flag de desconexión se encuentra activo, se asume que el Jugador se desconectó. 5) El Servidor Central da por finalizada la partida. Se carga un mensaje para avisarle al Jugador Rival que la partida fue finalizada por desconexión del oponente y se activa un flag de Finalización. 7) El Servidor Central recibe el pedido de Devolver las Nuevas Acciones. Al devolver las mismas, detecta que está activo el flag de Finalización, por lo que da por finalizada la Partida, sin penalizar ni recompensar a ningún jugador.

	entiende que la Partida en Curso fue finalizada.	

3.3.9 Búsqueda de Historial

TABLA XXI: Caso de uso Búsqueda de Historial

Flujo Normal	Actor	Sistema
	1) El usuario selecciona la opción ***Historial de Partidas*** en el menú	2) El sistema consulta la base de datos, buscando todas las partidas en las que haya estado involucrado el jugador, y le muestra el listado.

3.4 Diseño de bajo nivel

3.4.1 Servidor

Tal como se vio en el diagrama de arquitectura, el servidor está compuesto por 4 capas:

1. Interface: capa que expone servicios SOAP para ser consumidos por la aplicación cliente.
2. Negocio: capa que maneja las reglas de negocio de la aplicación servidor, comunicándose con los servicios externos, respondiendo a los llamados a la interface y utilizando la misma para comunicarse con los clientes.
3. Entidades: capa que es trasversal a toda la aplicación que contiene las entidades y mensajes utilizados tanto por el cliente como por el servidor.
4. Base de Datos: capa que se encarga de la persistencia y recuperación de los datos asociados a los usuarios.

Capa Interface

La capa de interface es la más simple de todas y solo expone las interfaces descriptas en el diagrama de arquitectura. No se muestra ningún diagrama ya que resultaría redundante.

Capa Negocio

La capa de negocio es la más compleja de todas. Es la encargada de realizar todas las tareas del servidor, implementando los módulos descriptos por el diagrama de arquitectura.

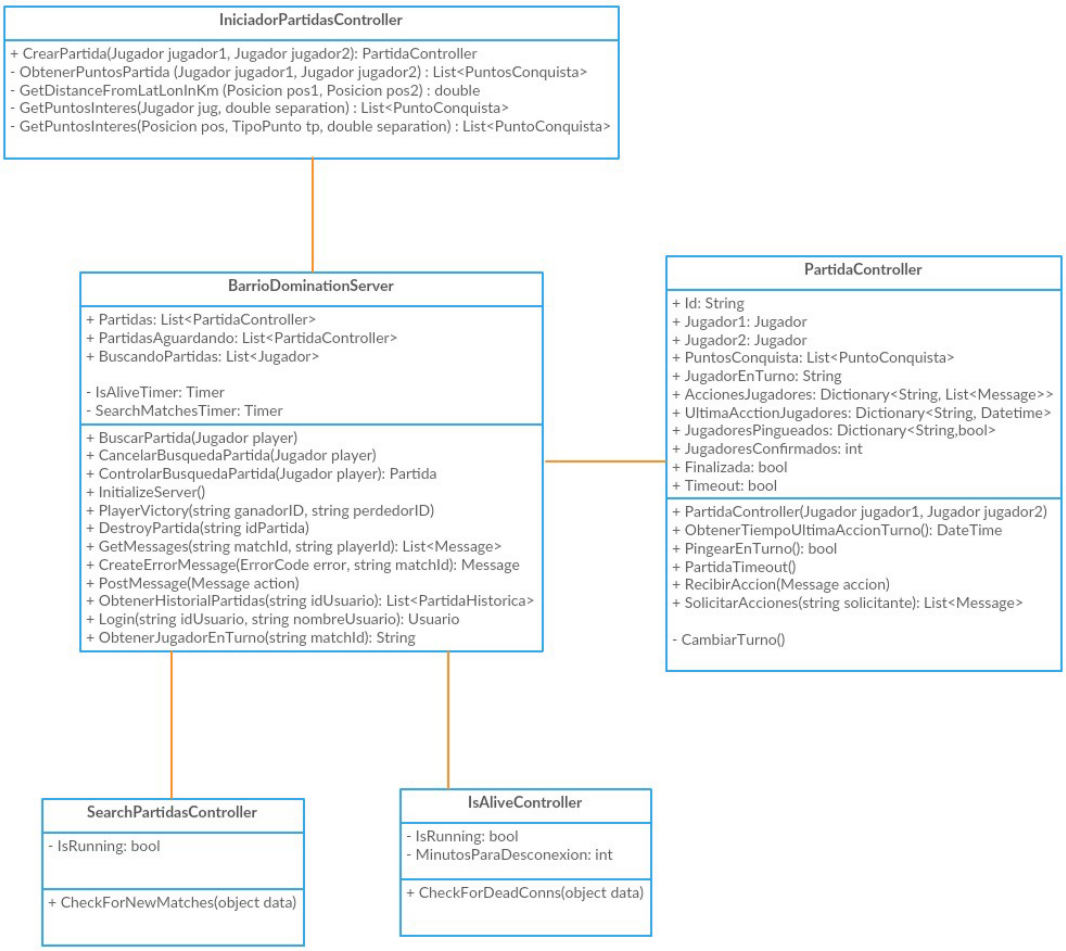

Figura 6 – Diagrama de Clases del Servidor, capa negocio

Capa Entidades:

La capa entidades está compuesta por dos grandes grupos:

1. **Business**: las entidades de este grupo son más bien de negocio, no siendo utilizadas durante el transcurso de una partida, sino para tener información específica de los usuarios. Estas son las únicas entidades que requieren ser persistidas.
2. **Model**: las entidades de este grupo son las utilizadas durante una partida. Entre las mismas se incluyen tanto las entidades que responden al modelo definido en el GDD como también a los mensajes formado por los clientes, los cuales el servidor se encarga de transmitir.

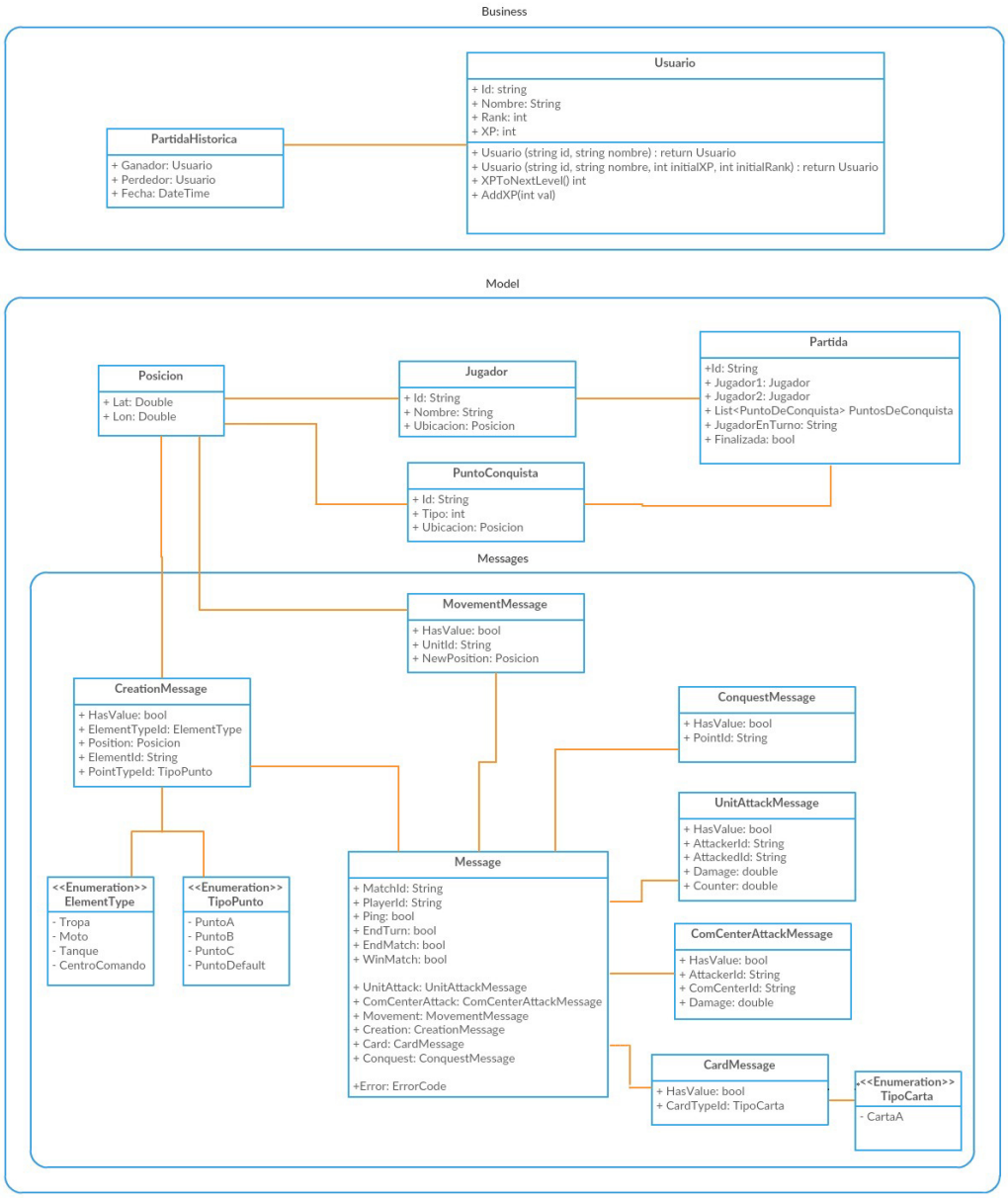

Figura 7 – Diagrama de Clases del Servidor, capa entidades

Capa Base de Datos:

La capa de base de datos provee los mecanismos de persistencia y recuperación de datos. No toda la información precisa ser persistida. Las entidades que serán persistidas son aquellas que pertenecen al grupo BUSINESS de la capa de Entidades. El diseño de esta capa resulta redundante, por lo que no será diagramado.

3.4.2 Cliente

Tal como se vio en el diagrama de arquitectura, el cliente utiliza la arquitectura MVC (modelo-vista-controlador). Primero se visualiza como se relacionan las clases entre sí y luego el detalle de cada una de las mismas.

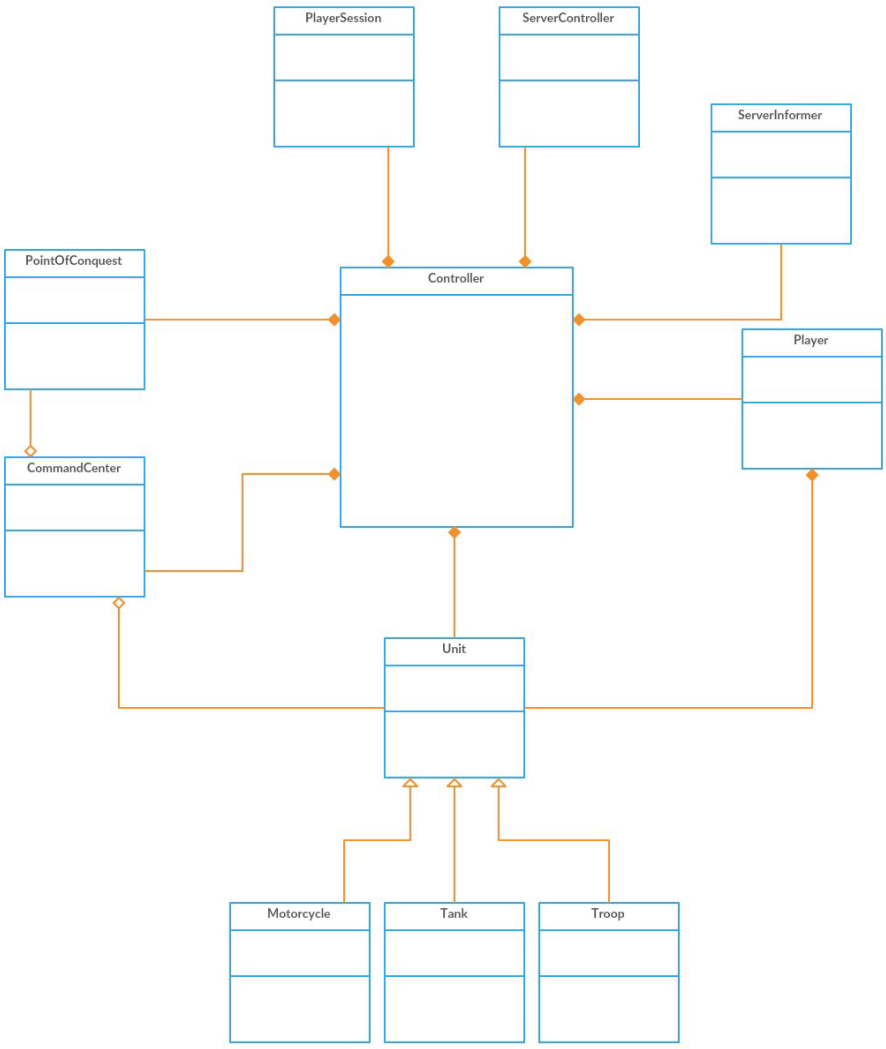

Figura 8 – Esquema del Diagrama de Clases

Figura 9 – Clase Controller

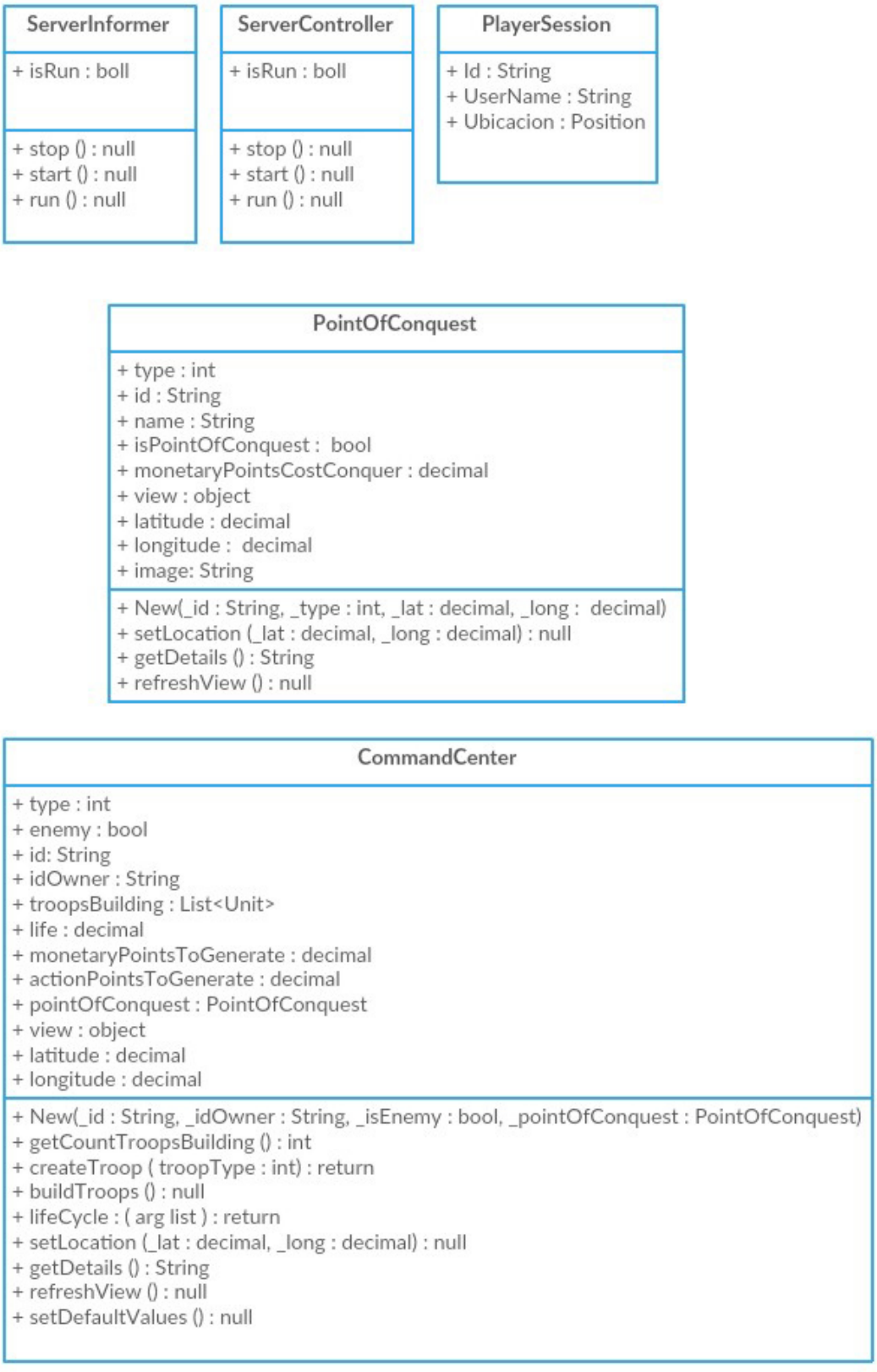

Figura 10 – Clases ServerController, PlayerSession, PointOfConquest y CommandCenter

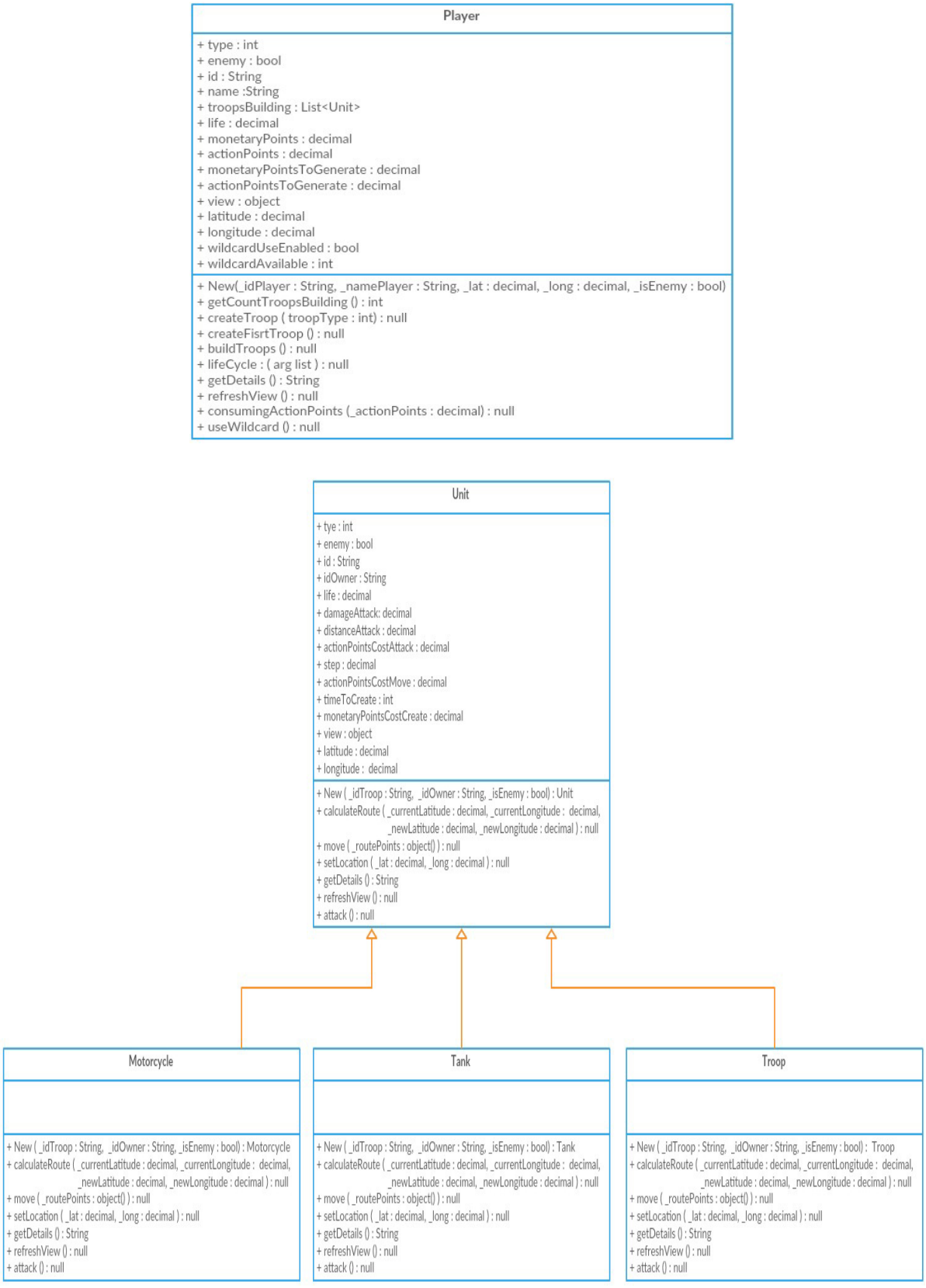

Figura 11 – Clases Player, Unit, Motorcycle, Tank y Troop

Clases para el manejo de mensajes entre el Cliente y el Servidor

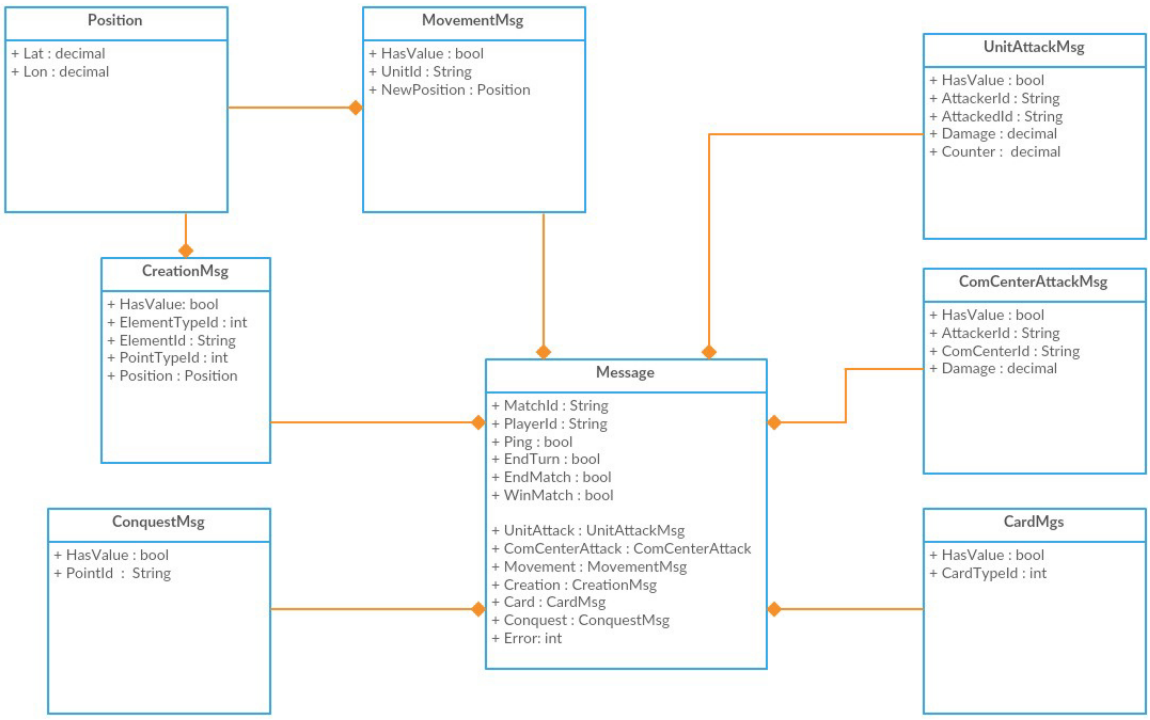

Figura 12 – Diagrama de Clases utilizadas para el manejo de la mensajería entre el Cliente y el Servidor

3.4.3 Diagramas de Secuencia

Se presentaran a continuación los diagramas de secuencia en donde se muestran las interacciones de los diferentes componentes del sistema para llevar a cabo las funcionalidades descriptas por los casos de uso.

Inicio de Sesión

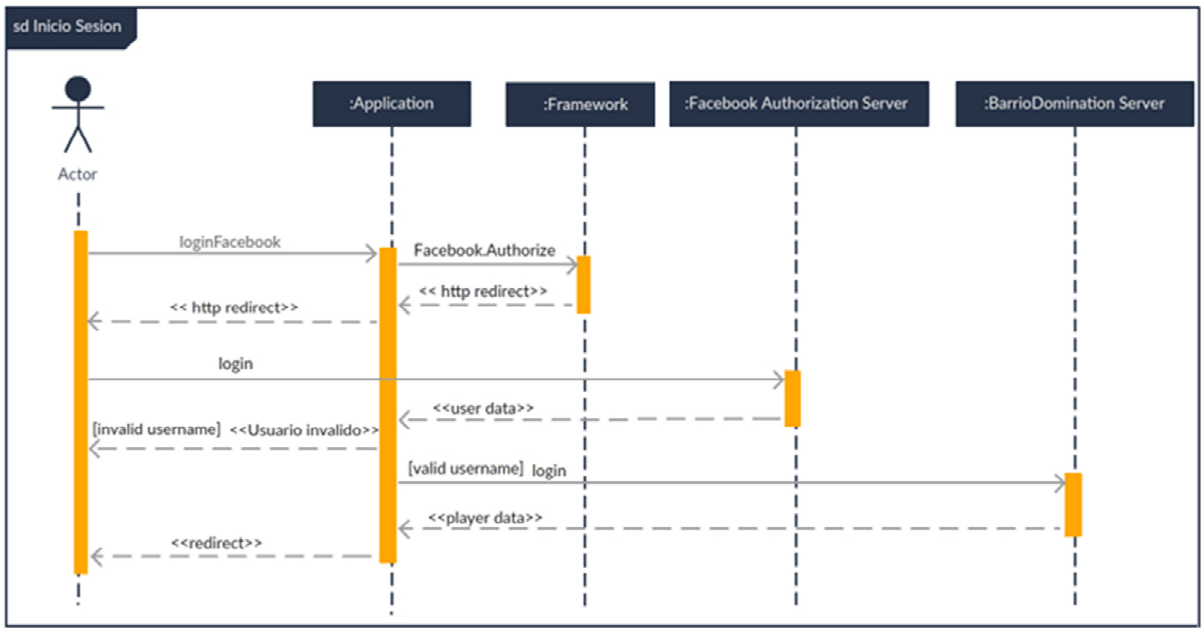

Figura 13 – Diagrama de Secuencia del Inicio de Sesión

Buscar Partida

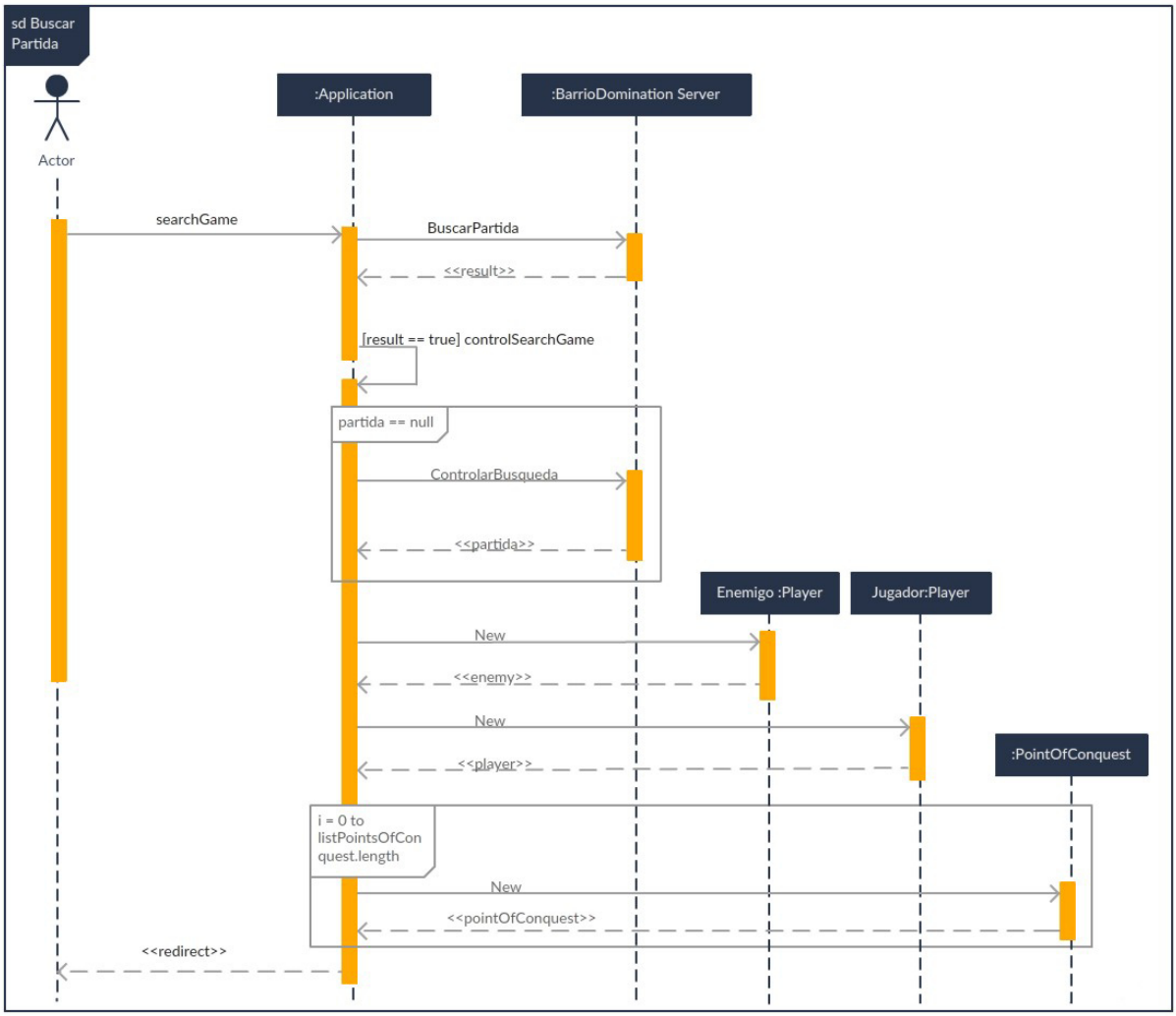

Figura 14 – Diagrama de Secuencia Buscar Partida

Crear Partida

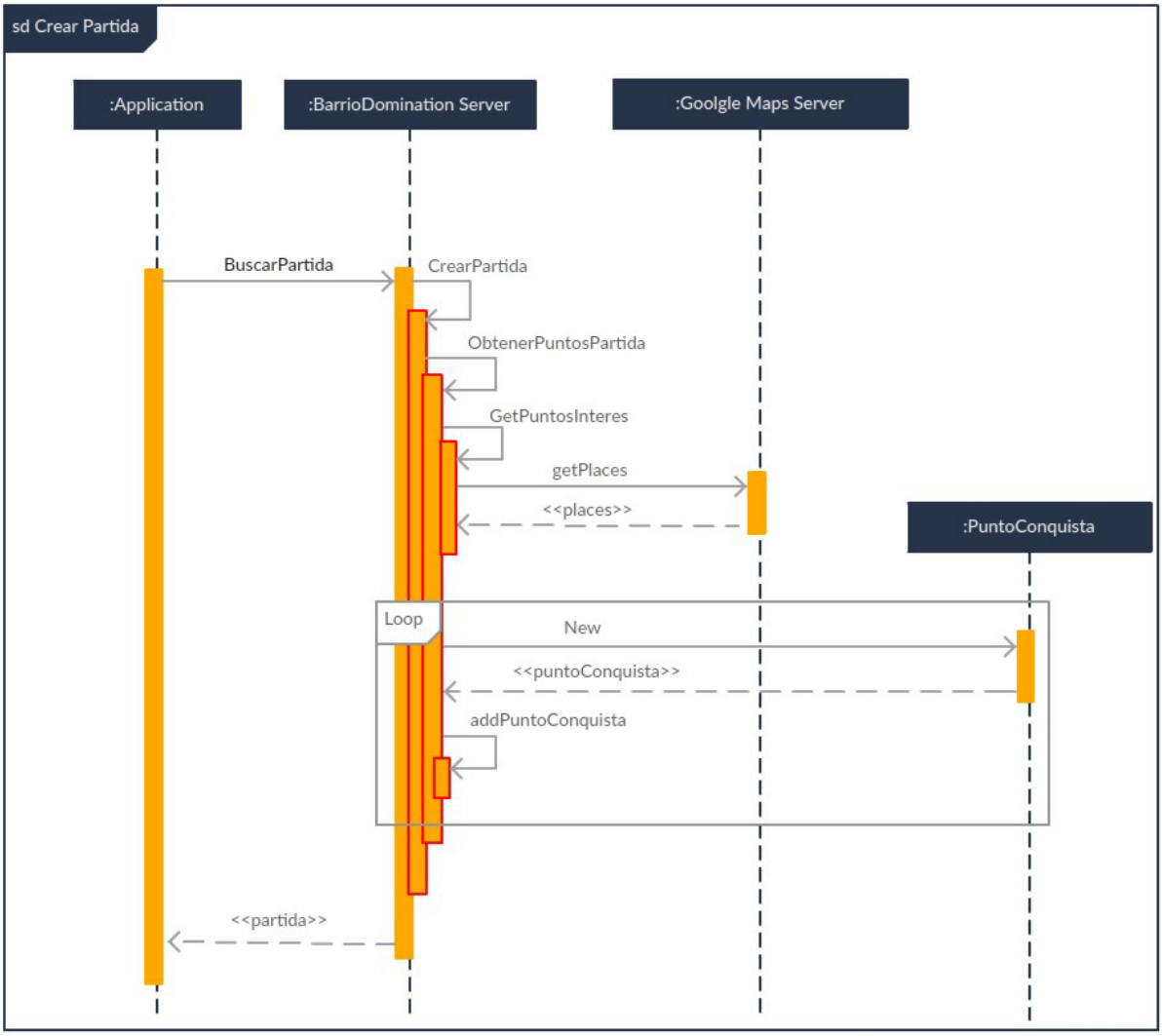

Figura 15 - Diagrama de Secuencia Crear Partida

Crear Unidad

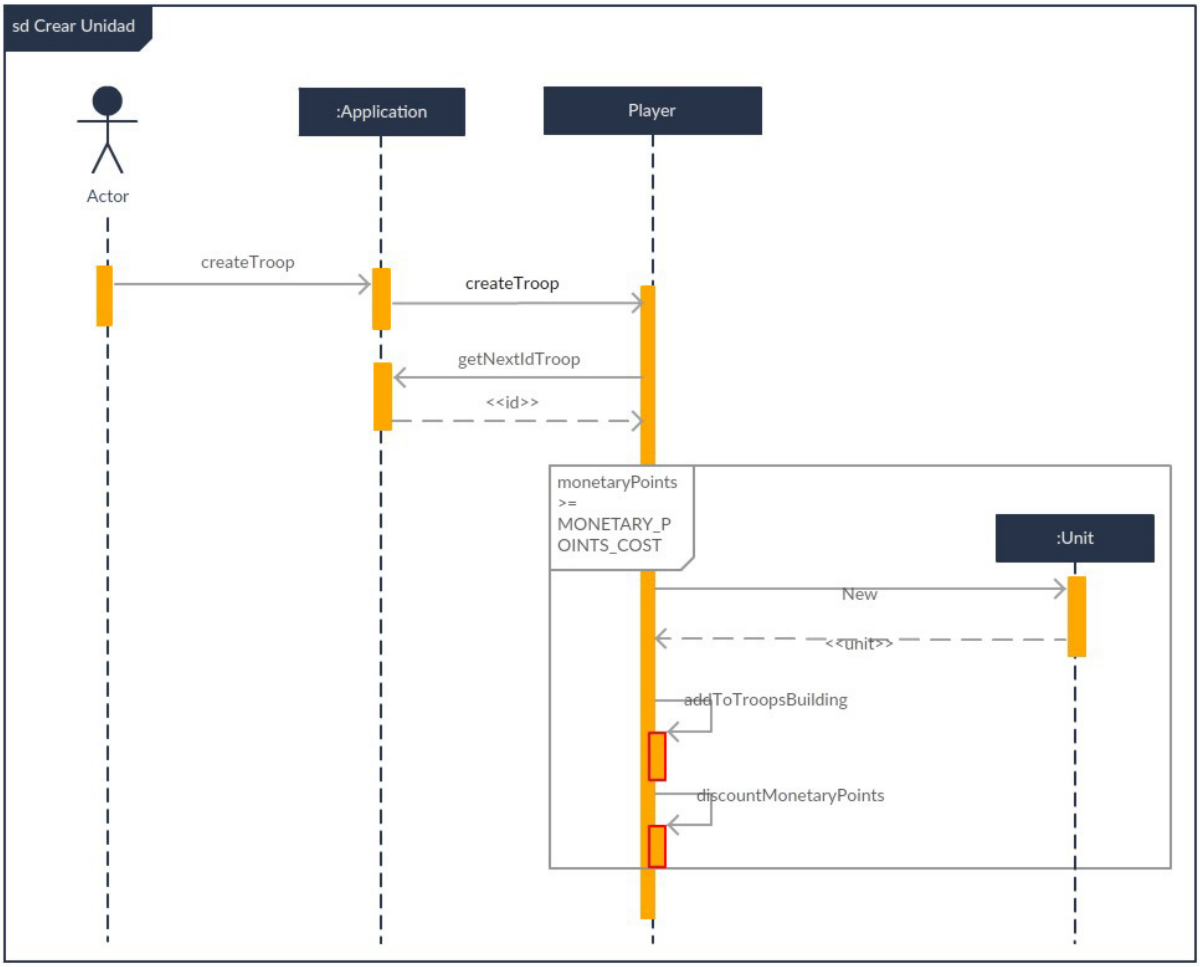

Figura 16 – Diagrama de Secuencia Crear Unidad

Activar Tarjeta

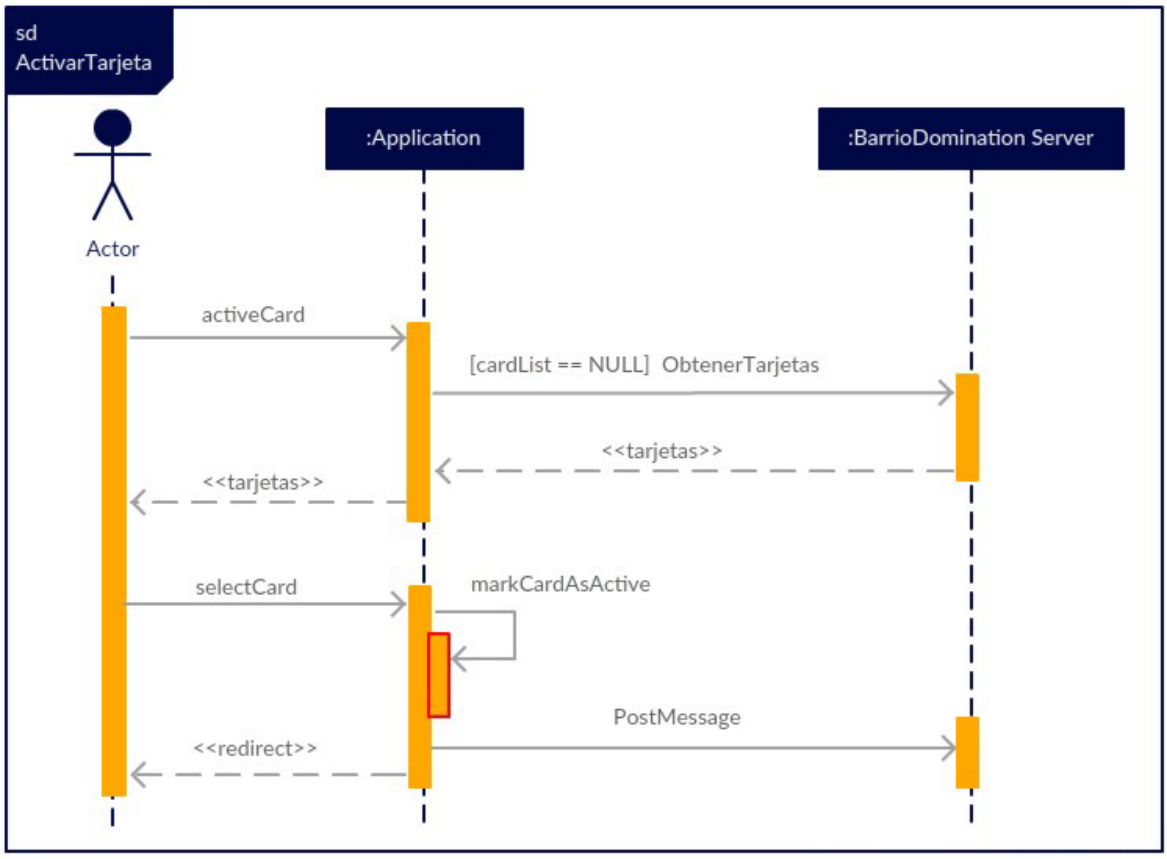

Figura 17 – Diagrama de Secuencia Activar Tarjeta

Cambio de Turno

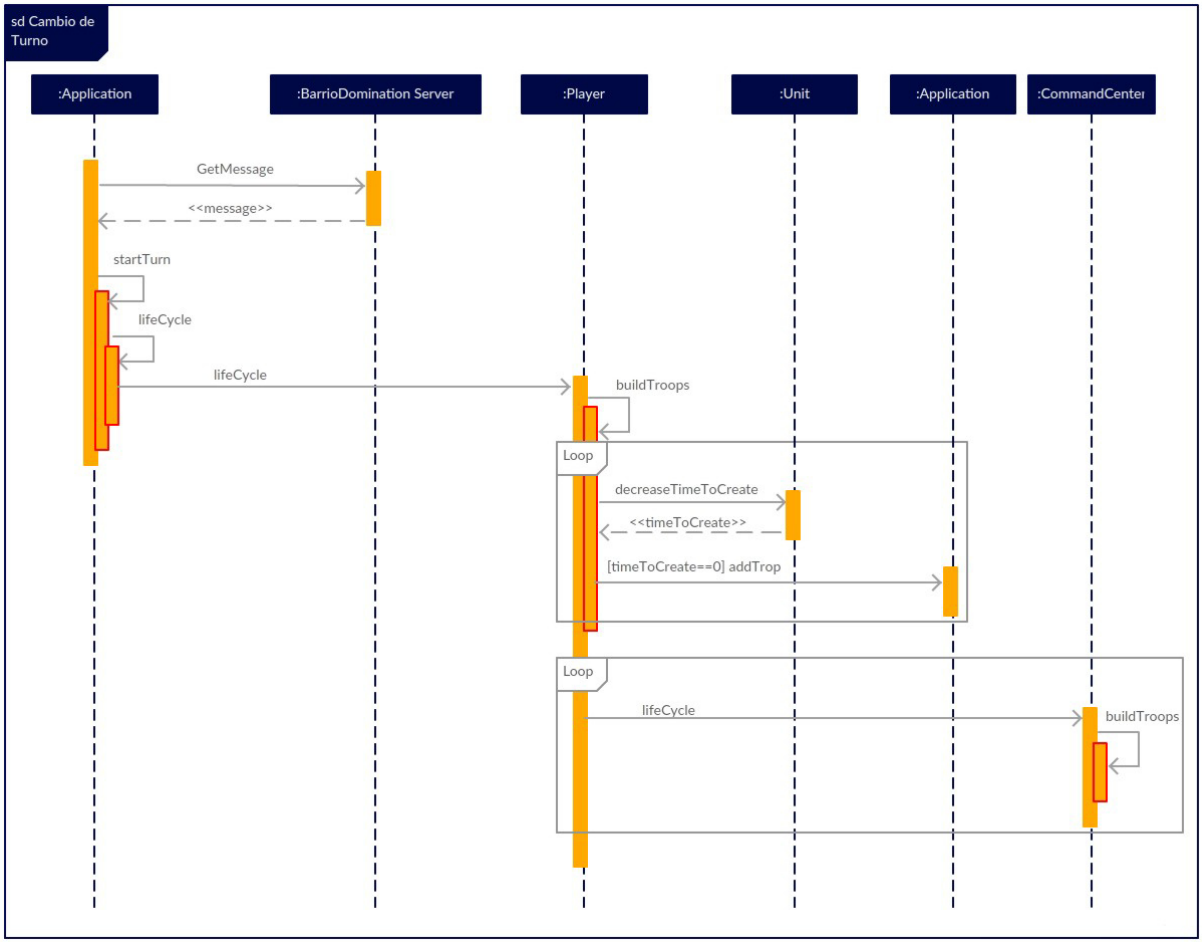

Figura 18 - Diagrama de Secuencia Cambio de Turno

Mover Unidad

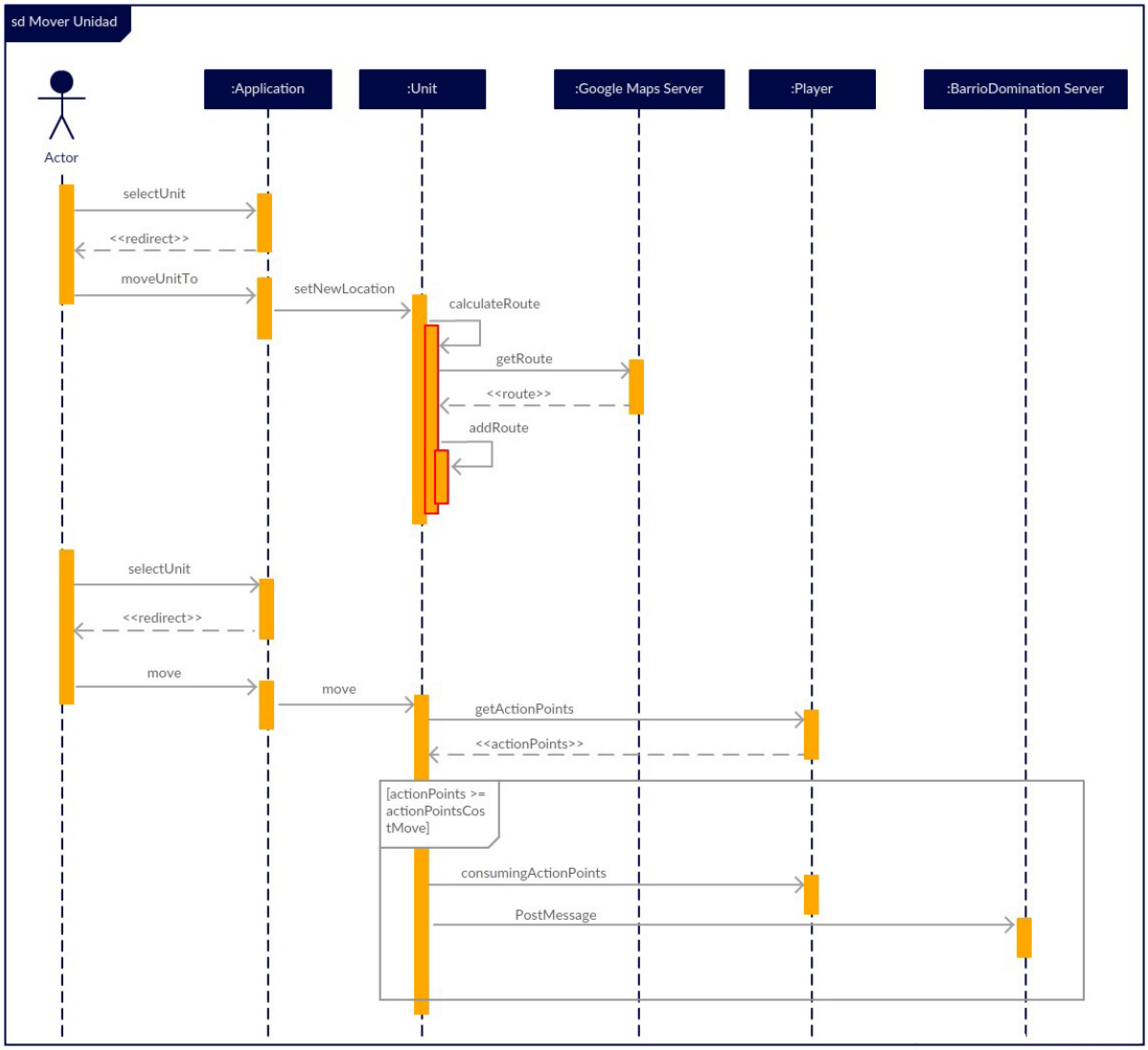

Figura 19 – Diagrama de Secuencia Mover Unidad

Ataque objetivo Enemigo

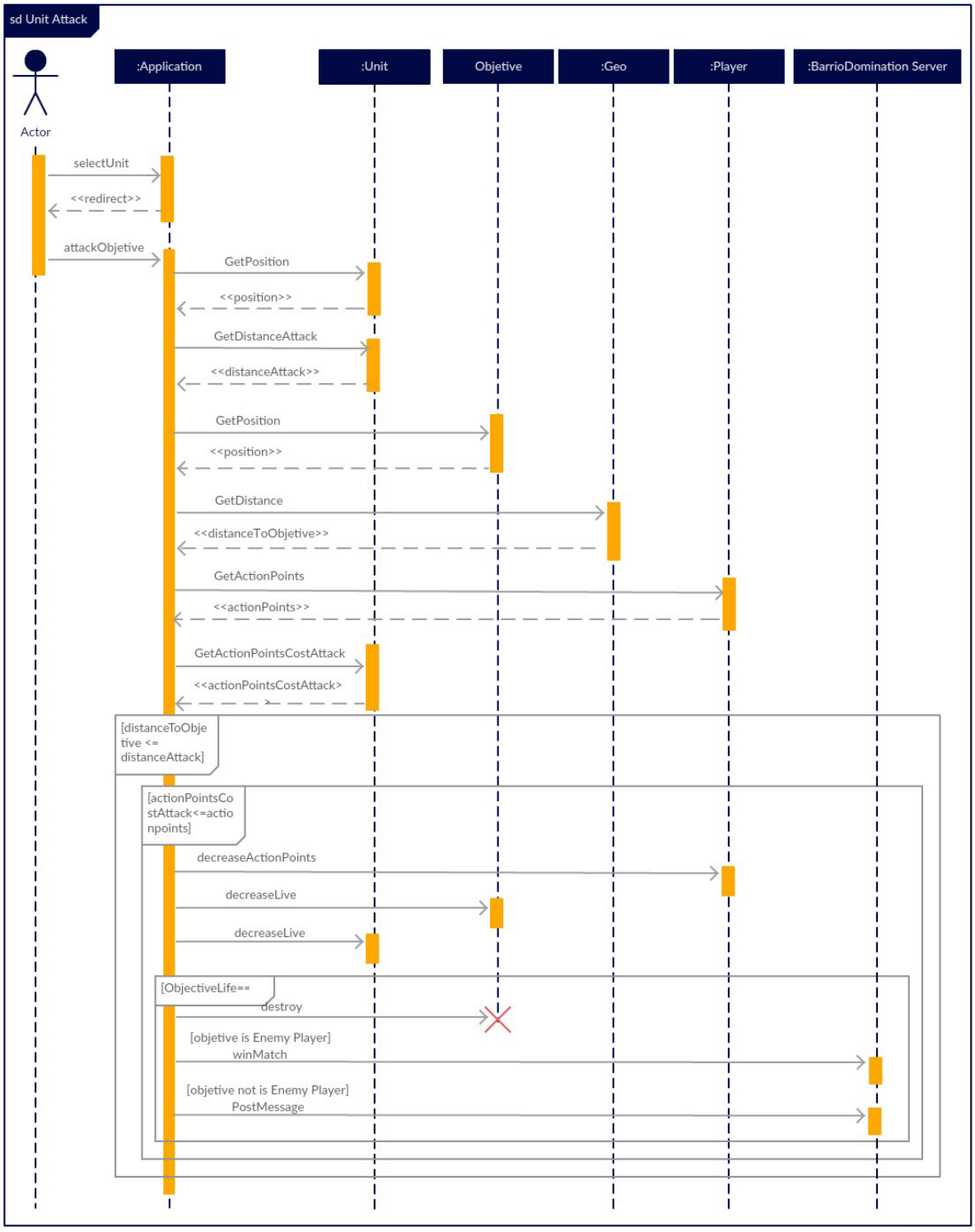

Figura 20 – Diagrama de Secuencia Ataque objetivo Enemigo

CAPITULO IV: Desarrollo

4.1 Desarrollo del servidor

La aplicación que se utiliza del lado del servidor se encuentra desarrollada bajo el lenguaje de programación C#, el cual es interpretado y compilado por la plataforma .NET de Microsoft. A su vez, utiliza una base de datos MS-SQL.

4.1.1 Experiencia del desarrollo

En la sección *Herramientas de Desarrollo* del *Capítulo II* se explicó que la elección de las herramientas utilizadas se debió a la experiencia profesional con la que cuentan los tesistas en dichas tecnologías. Teniendo eso en cuenta, se descartó por completo cualquier riesgo introducido por una curva de aprendizaje.

Para la utilización de la plataforma .NET de Microsoft se utilizó el Framework 4.0 y el Visual Studio 2010 con licencia Premium, que fue obtenida gracias al convenio entre la UADE y Microsoft. Para la base de datos, se decidió utilizar la versión SQL Express que Microsoft provee de forma gratuita a través de su página web.

Durante el desarrollo del sistema en su conjunto encontramos que es más eficiente realizar la invocación de los servicios de Mapas de Google para obtener los puntos de interés desde el servidor al momento de inicializar las partidas, en lugar de hacerlo desde el cliente. Esto provocó un ligero rediseño del caso de uso que requirió una modificación de código, la cual resultó bastante simple gracias a la facilidad de uso y la amplia documentación de los servicios de Google.

Podemos mencionar como otro hecho a destacar la necesidad de implementar una función para el cálculo de distancia entre dos pares de coordenadas, basada en la Fórmula de Haversine, la cual está descripta en el Anexo VI.

4.1.2 Problemas encontrados

Durante el desarrollo del servidor no se encontraron problemas especialmente complejos, lo cual era la razón principal de la elección de las herramientas utilizadas, y resulta acorde al análisis FODA realizado en el *CAPITULO II: Análisis*.

Como hecho destacable podemos mencionar que se encontró una incompatibilidad entre dos parámetros de los servicios de Mapas de Google: los filtros RADIUS (utilizado para especificar el radio de búsqueda, en metros, alrededor del punto de ubicación sobre el cual se realiza la búsqueda) y RANKBY (utilizado para solicitar que los puntos encontrados vengan ordenados bajo un criterio predeterminado) son excluyentes. Esta incompatibilidad, la cual no se encontraba documentada al momento de realizar el desarrollo, se encontró en base a las pruebas realizadas y a comentarios en diferentes foros.

4.2 Desarrollo del cliente

La aplicación que se utiliza del lado del cliente se encuentra desarrollada bajo el lenguaje de programación JavaScript, el cual es interpretado de forma nativa por la mayoría de los Sistemas Operativos. Esto permite que la aplicación sea multiplataforma con solo leves modificaciones.

Utiliza los mapas y servicios de Google Maps para la visualización y posicionamiento de los objetos del juego y para el cálculo de rutas entre distintos puntos del mapa. En relación a la obtención de distancias y dirección de desplazamiento utiliza ecuaciones matemáticas de navegación marítima y aérea (Anexo VI). Dichas ecuaciones fueron implementadas en una clase auxiliar del proyecto con el propósito de agilizar el cálculo e independizar la aplicación de un servicio perteneciente a terceros, ya que el Framework no cuenta con un módulo para el cálculo de distancias y el servicio ofrecido por Google incrementa el tiempo de respuesta de la aplicación en determinadas acciones del jugador.

La comunicación con los servicios de Google Maps y la Aplicación Servidor se realiza a través de Web Services de forma Asincrónica. En ocasiones esto permite que la aplicación continúe realizando procesos hasta que obtiene la respuesta del Servicio.

Para la autenticación de los usuarios se utiliza un módulo propietario del Framework que permite la validación de las cuentas ante los servidores de Facebook y se desarrolló un módulo para la validación de las cuentas de Google. Con la utilización de estos módulos la Aplicación Servidor no requiere administrar las contraseñas de los usuarios, únicamente necesita almacenar los nombres y los ID.

4.2.1 Experiencia del desarrollo

En la sección *Herramientas de Desarrollo* del *Capítulo II* se compararon las distintas alternativas para el desarrollo de la aplicación y se justificó la elección del Framework *Titanium Mobile* de *Appcelerator*.

Para la utilización del Framework *Titanium Mobile* se debe crear un usuario en la página del proveedor, el cual permite descargar el Paquete de Desarrollo. Dicho usuario es necesario para poder abrir el Entorno de Desarrollo Integrado (IDE), ya que cada vez que se abre el mismo se debe iniciar sesión contra los servidores de *Appcelerator*.

La documentación del Framework se encuentra organizada de tal manera que cada capítulo representa un módulo provisto por el Framework. En cada sección se describen las propiedades y métodos de cada componente, junto con ejemplos se su utilización. Además, se especifica en que Sistema Operativo y versión del mismo se puede utilizar la propiedad o método. Esto último es muy útil porque permite decidir con cautela las propiedades y métodos a utilizar en el caso de desarrollar una aplicación multiplataforma. Al comienzo de cada texto se indica si el componente descripto se encuentra vigente o si fue reemplazado por otro. En el caso se haber sido reemplazado proporciona un link a la documentación del nuevo componente.

El Paquete de Desarrollo provisto por *Appcelerator* utiliza como Entorno de Desarrollo Integrado una variante del IDE Elipse de Eclipse Foundation, con lo cual permite la instalación de plugins desarrollados por terceros para facilitar la programación y el control de versionados. A diferencia del IDE Eclipse original, utiliza como lenguaje principal de programación JavaScript y HTML y un lenguaje propietario denominado Alloy, mezcla de JavaScript, HTML y XML. La utilización de JavaScript y HTML permite que las aplicaciones

puedan ser desarrolladas para cualquier sistema operativo y dispositivo realizando cambios menores en los métodos o propiedades utilizados.

Con respecto a la Geolocalización, el Framework cuenta con un módulo para el manejo de los mapas de Google (en adelante denominado Maps) y un módulo para obtener la ubicación del dispositivo (en adelante denominado Geolocation). Éste último módulo utiliza el GPS del dispositivo o la conexión a Internet para el cálculo de la posición geográfica. El GPS como origen de la información referida a la ubicación, brinda una mayor precisión con respecto a la conexión a Internet, pero consume más batería. Estos 2 factores deben ser tenidos en cuenta a la hora de definir cuál va a ser el proveedor de la ubicación. El módulo Maps permite visualizar los mapas provistos por el servicio de Google Maps, obtener las coordenadas de puntos en el mapa, crear rutas entre distintos puntos y crear anotaciones que representan puntos de interés. Además, la vista del mapa y sus componentes cuentan con eventos que responden a acciones realizadas por el usuario, como por ejemplo desplazar una anotación de un punto a otro o hacer clic sobre el mapa y obtener las coordenadas del punto.

Las anotaciones que permite crear el Framework en los mapas cuentan con acciones reducidas, pero al poder realizarse herencia de las clases propietarias se logra obtener una nueva clase con mayor funcionalidad y nuevas propiedades que se adaptan a los requerimientos de la aplicación que se está desarrollando.

El módulo Maps no cuenta con funcionalidad para cálculo de distancias y obtención de rutas entre distintos puntos. Para el cálculo de distancias se pueden implementar las ecuaciones matemáticas utilizadas en la navegación marítima y aérea, las cuales son de público conocimiento y en varios sitios ya se encuentran codificadas en distintos lenguajes de programación. También se puede hacer uso de los servicios de Google Maps, pero dichos servicios cuentan con una cantidad limitada de consultas en su modo gratuito y son asincrónicos, lo cual conlleva a un mayor tiempo en la obtención del dato. Las rutas entre distintos puntos se deben obtener mediante el servicio de Google Maps, el cual posibilita definir el modo de desplazamiento de un punto al otro, esto determina si van a ser tenidas en cuenta las direcciones de desplazamiento de las calles para el cálculo de la ruta. Este servicio también es asincrónico, por ende, se tiene un *delay* entre la consulta de la ruta y la obtención de los datos. Los servicios de Google Maps son accedidos a través de un Web Service, y

como se comentó anteriormente son asincrónicos, debido a esto, se tiene que definir en el objeto utilizado para la conexión una función que se va a ejecutar cuando se reciba una respuesta (callback), ya sea por el procesamiento correcto del pedido o por un error producido.

El Framework también cuenta con un módulo para Facebook. Con este módulo se puede iniciar sesión en Facebook desde la aplicación que se está desarrollando y también realizar publicaciones. Para poder utilizar las cuentas de Google en la autenticación de los usuarios fue necesario el desarrollo de un módulo que realice las consultas necesarias a los servicios de Google ya que el Framework no cuenta con un módulo propio para poder realizar acciones de forma directa contra Google.

Tanto para utilizar los servicios de Google como los de Facebook es necesario crear una cuenta y registrar los datos de la aplicación que va a hacer uso de los mismos. En ambos casos, el proveedor proporciona un ID y una clave que deben cargarse en el archivo de configuración de la aplicación. Sin estos datos el proveedor deniega el acceso a sus servicios ya que en cada llamado al servicio, tanto de Facebook como de Google, es obligatorio informar el ID y clave para la autenticación de la aplicación.

4.2.2 Problemas encontrados

El principal problema que tiene el Paquete de Desarrollo proporcionado por *Appcelerator* es que requiere una conexión a Internet activa siempre. Como se mencionó anteriormente, el IDE (Entorno de Desarrollo Integrado) requiere iniciar sesión cuando se lo abre, con lo cual no se puede continuar con el desarrollo de la aplicación si no se cuenta con una conexión de Internet. Si el IDE no puede validar el usuario y la contraseña se cierra de forma automática.

El hecho de utilizar JavaScript como lenguaje principal facilita el desarrollo de aplicaciones multiplataforma. Pero presenta una desventaja al ser un lenguaje de programación que no requiere la declaración previa de las variables. Cuando el código es muy complejo resulta difícil encontrar una variable que no fue declara o detectar una variable que se encuentra mal escrita. El IDE cuenta con herramientas para detectar errores de sintaxis en el código pero no puede detectar variables que no se encuentran declaradas. Este tipo de error

se manifiesta en tiempo de ejecución y puede llevar unas horas en ser detectado en la codificación.

Cuando se desarrollan aplicaciones para dispositivos móviles el IDE permite utilizar emuladores para ejecutar la aplicación en un entorno de testeo. Sin embargo, dependiendo de los módulos del Framework utilizados se puede o no ejecutar la aplicación en los emuladores. En el caso de incorporar al proyecto el módulo de los Mapas, la ejecución de la aplicación se debe realizar sobre un dispositivo real. Esto conlleva a que no se pueda realizar una ejecución paso a paso de la aplicación para controlar el estado de las variables en cada línea de código. Con lo cual, detectar un error en la asignación de un valor a una variable o la ejecución de una instrucción se torna un poco más complejo ya que se debe controlar el estado de las variables mediante otros medios, como por ejemplo, guardar en un archivo el estado de las variables que se desean controlar.

CAPITULO V: Pruebas

5.1 Pruebas Realizadas

En cada fase de desarrollo de la aplicación se realizaron pruebas unitarias e integrales, ya que los componentes proporcionados por el Framework Titanium Mobile requerían mayor funcionalidad y se debía verificar el correcto funcionamiento de las llamadas al Servicio Web de Google Maps.

Las pruebas de los Servicios Web de Google Maps fueron llevadas a cabo en distintos dispositivos físicos con Sistema Operativo Android. El Framework Titanium Mobile establece que las funcionalidades de Geolocalización tienen que ser testeadas en equipos físicos debido a que los emuladores no soportan el módulo.

Las primeras pruebas fueron llevadas a cabo en un ambiente aislado sin la integración con la aplicación Servidor. Dichas pruebas incluían la ubicación de los componentes en el mapa, el cálculo y visualización de las rutas de desplazamiento de las unidades, el desplazamiento de las unidades siguiendo las rutas asignadas y las acciones de cada uno de los componentes.

Una vez verificado el correcto funcionamiento de los componentes del Juego en el ambiente aislado se comenzaron con las pruebas integrales contra la aplicación Servidor. En estas pruebas todavía no había interacción entre dos aplicaciones del tipo Cliente, solamente se realizó el testeo sobre una aplicación Cliente y la otra aplicación se encontraba simulada a través de una página Web. En esta etapa se probó el intercambio de mensajes entre la aplicación Cliente y Servidor, ya que el Cliente debe informar todas sus acciones al Servidor y además consultar al mismo las acciones realizadas por el contrincante. También se verificó el funcionamiento del Servidor para la asignación de partidas y la obtención de los puntos de interés a ser conquistados por los jugadores.

Por último, se realizaron las pruebas entre dos aplicaciones del tipo Cliente interactuado con la aplicación Servidor. El objetivo de esta prueba final de integración era verificar el correcto funcionamiento de la aplicación Cliente y de la aplicación Servidor ante acciones realizadas por personas y no estipuladas con anterioridad. Ya que en las pruebas anteriores las acciones se realizaban para probar una funcionalidad puntual del sistema. Al ser

realizadas las acciones por personas, las mismas son aleatorias y no siguen un patrón, con lo cual se pudieron detectar errores que no habían sido contemplados en las pruebas anteriores.

En cada una de las pruebas de integración se verificaba el correcto funcionamiento de los componentes del Juego y los llamados a la API de Google Maps.

5.2 Conclusión

El IDE proporcionado por el Paquete de Desarrollo de Appcelerator no permite realizar un debug del código fuente cuando la aplicación se ejecuta en un equipo físico, con lo cual se deben realizar pruebas del tipo caja negra. Esto dificulta la verificación del código ya que no se puede realizar una ejecución paso a paso.

Cada ejecución de la aplicación implica la compilación de todo el proyecto, establecer la conexión contra el dispositivo físico o emulado y la instalación de la aplicación en el dispositivo. Estos pasos pueden dar error si el dispositivo demora en responder a las peticiones del IDE. Con lo cual, el verificar una corrección simple conlleva demasiado tiempo.

La realización de pruebas unitarias permitió agilizar la verificación del código fuente ya que la utilización del módulo de Geolocalización proporcionado por el Framework obligaba a realizar pruebas del tipo Caja Negra por no poderse realizar una ejecución paso a paso en equipos físicos. Las pruebas unitarias que no requerían del módulo de Geolocalización se realizaron de forma aislada en dispositivos emulados, lo cual permitió efectuar pruebas del tipo Caja Blanca.

Las pruebas de la aplicación demoraron más tiempo del previsto debido a los pasos realizados por el IDE para la compilación y ejecución de aplicaciones. Estos pasos pueden arrojar errores aleatorios durante la compilación por problemas del IDE o bien por una demora en la respuesta de los dispositivos físicos o emulados. Por estos motivos, en ocasiones, las pruebas eran imposibles de realizarse.

CAPITULO VI: Conclusiones

5.1 Observaciones

Este proyecto fue pensado con el objetivo de desarrollar un producto novedoso vinculado a tecnologías y metodologías para las cuales no se contaba con experiencia previa.

Se generó un diseño inicial a partir de una idea, plasmándose en un high concept, luego de lo cual se realizó un análisis de productos similares disponibles actualmente en el mercado con la finalidad de que el producto final desarrollado sea diferente a los previamente existentes.

Una vez consolidada la idea se comenzó a expandir y a detallar la misma, agregando profundidad y complejidad, generando así el documento de diseño del juego - llamado “game design document” -, un tipo de documentación completamente diferente a la que se suele utilizar en sistemas, dada su amplia audiencia, lo cual deriva en una mezcla de lenguaje técnico y lenguaje informal, y su alta variabilidad, dado que sufre muchos cambios en el tiempo, producto de que las definiciones se van ajustando y modificando a medida que las ideas son puestas en práctica.

La utilización de prácticas ágiles para el desarrollo se vio sustentada por la variabilidad del diseño: a diferencia de un sistema tradicional, donde los requerimientos están vinculados a objetivos de negocio, los cuales no varían significativamente en el tiempo, en el desarrollo de videojuegos los mismos están vinculados a un factor completamente subjetivo como la “diversión”, por lo que los mismos requieren de ajustes constantes.

Se apostó por una plataforma móvil llamada Titanium Mobile Appcelerator dado que pareció ser la que ofrecía mejores prestaciones para las necesidades de desarrollo y una curva de aprendizaje más suave, combinada con una plataforma Web basada en Framework .NET para aprovechar la amplia experiencia profesional en la misma. En ninguno de los casos se requirió una inversión de capital para adquirir licencias, dado que por el lado de Titanium Mobile la empresa provee licencias gratuitas de código abierto, mientras que las licencias de .NET fueron provistas por el convenio académico entre UADE y Microsoft.

El trabajo fue pensado con fines académicos, no comerciales, por lo que no está desarrollado para obtener rédito económico. Sin embargo, su diseño sí permite ampliar el desarrollo para agregar funcionalidad con fines lucrativos en el futuro; temática que se encuentra tratada en el Anexo II.

5.2 Conclusiones

- Existen diversas herramientas para el desarrollo de aplicaciones para plataformas móviles que cubren una amplia variedad de requerimientos. Con este proyecto se obtuvo un análisis sobre las mismas, que puede ser utilizado para definir cuál es la mejor a utilizar en futuros proyectos.
- El Framework Titanium Mobile Appcelerator para el desarrollo de la aplicación cliente no cumplió con las expectativas. A medida que se fue desarrollando la aplicación fueron apareciendo problemas de performance, inconvenientes para el testeo en dispositivos reales y limitaciones en los módulos de los mapas y geolocalización. En ocasiones se tuvieron que desarrollar módulos propios para agregar funcionalidad que no era proporcionada por el Framework.
- La utilización de JavaScript (JS) como lenguaje de programación no fue tenido en cuenta como un posible problema a la hora del desarrollo de la aplicación. Por tratarse de un lenguaje no estructurado hay errores que no pueden ser detectados por el IDE, como por ejemplo, la utilización de una variable que no fue previamente declarada. Este tipo de errores producen pérdidas de horas en la revisión del código fuente.
- En cuanto a la documentación del Framework Titanium Mobile Appcelerator, en un principio la misma parecía completa, pero a medida que se requería profundizar en el funcionamiento de los módulos se encontró que la misma era incompleta en algunos casos y en otros no era lo suficientemente clara. Por tal motivo, se recurrió a consultar foros mantenidos por usuarios del Framework que compartían sus conocimientos y código fuente.

- La incorporación de buenas prácticas y de la metodología AUP en las etapas de desarrollo permitieron sobrellevar todos los inconvenientes surgidos con el Framework Titanium Mobile.
- El desarrollo de videojuegos, por contar con un carácter de base subjetivo como es el "factor diversión", es completamente diferente al desarrollo tradicional de sistemas, con documentación mucho más cambiante en el tiempo y un proceso mucho más dependiente de las iteraciones.
- El mercado de los videojuegos tiene una composición mucho más amplia de la que inicialmente uno podría imaginar, y aún queda mucho margen de crecimiento, en especial para el mercado latinoamericano.

BIBLIOGRAFIA

Android Developers [en línea]. [Consulta 13 junio 2014].
<http://developer.android.com>

API Documentation – Appcelerator Platform [en línea]. [Consulta 3 julio 2014].
<http://docs.appcelerator.com/platform/latest/#!/api>

Calcular distancia a partir de datos GPS [en línea]. [Consulta 10 enero 2016].
<http://www.deif.org/blog/calcular-distancia-a-partir-de-datos-gps/>

DOJO Mobile [en línea]. [Consulta 12 junio 2014].
<http://dojotoolkit.org>

Essential facts about the Computer and Video Games Industry [en línea]. Estados Unidos de América: Entertainment Software Association. Abril 2015. [consulta: 5 mayo 2016].
<http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf>

Facebook for Developers [en línea]. [consulta 8 mayo 2016].
<https://developers.facebook.com/>

Geographic Information Systems FAQ [en línea]. [consulta 10 enero 2016].
<www.faqs.org/faqs/geography/infosystems-faq/>

GIS: Cálculo de distancias sobre la Tierra [en línea]. [consulta 10 enero 2016].
<https://personal.franchu.net/2007/11/16/gis-calculo-de-distancias-sobre-la-tierra/>

GIS Map Info [en línea]. [consulta 10 enero 2016].
<www.igismap.com>

Google API - Setting up OAuth 2.0 [en línea]. [consulta 20 Agosto 2014].
<https://support.google.com/cloud/answer/6158849?hl=en>

MORINE Iskandar y RICARDO José. Estudio comparativo de alternativas y Frameworks de programación, para el desarrollo de aplicaciones móviles en entorno Android. Proyecto Final de Carrera (Ingeniería de Telecomunicaciones). Barcelona, España. Universidad Politécnica de Cataluña, Escuela Técnica Superior de Ingeniería en Telecomunicaciones. 2013. 125p.

Jquery Mobile [en línea]. [consulta 11 junio 2014]. <http://jquerymobile.com>

Movable Type Scripts [en línea]. [consulta 10 enero 2016]. <www.movable-type.co.uk>

Newzoo. Global Report: US and China Take Half of $113BN Games Market in 2018. *Newzoo* [en línea]. 18 Mayo 2015. [consulta: 11 julio 2016]. <https://newzoo.com/insights/articles/us-and-china-take-half-of-113bn-games-market-in-2018>

Phonegap [en línea]. [consulta 11 junio 2014]. <http://phonegap.com>

Sencha Touch [en línea]. [consulta 12 junio 2014]. <www.sencha.com>

SINNOT R.W. Virtues of the Haversine. *Sky and Telescope,* (68): 159, 1984.

Titanium Mobile [en línea]. [consulta 11 junio 2014]. <www.appcelerator.com>

State of Mobile App Developers 2016 [en línea]. San Francisco, California: INMOBI. 2016. [consulta 20 febrero 2017]. <http://www.inmobi.com/insights/download/whitepapers/state-of-mobile-app-developers-2016/>

VARGAS, Carlos. Administración del Riesgo en Proyectos Informáticos. San José, Costa Rica. Club de Investigación Tecnológica [en línea]. Noviembre 2007. [consulta: 6 mayo 2016]. <http://www.clubinvestigacioncr.com/docs/informe39.pdf>

HUNICKE Robin, LEBLANC Marc y ZUBEK Rober. MDA: A Formal Approach to Game Design and Game Research. Evanston, Estados Unidos de América. Northwestern University. [en línea]. [consulta 11 junio 2014]. <https://www.cs.northwestern.edu/~hunicke/MDA.pdf>

ROUSE, Richard III. Game Design: Theory & Practice. Wordware Publishing Inc., 2004. ISBN 978-1556229121.

SCHELL, Jesse. The Art of Game Design, A Book of Lenses. Amsterdam. Elsevier/Morgan Kaufmann, 2008. ISBN: 978-0123694966.

ROLLINGS Andrew y ADAMS Ernest. On Game Design. 2003, Prentice Hall Computer. ISBN: 978-1592730018.

ADAMS Ernest. Fundamentals of Game Design. 2013, Addison Wesley Longman INC. ISBN: 978-0321929679.

SCHUBERT, Damion. How To Write Great Design Documents. [video]. Game Developers Conference. San Francisco, Estados Unidos de América, 2007. [en línea]. <http://www.gdcvault.com/play/581/Writing-Great-Design>

ANEXOS

ANEXO I – Conceptos de Diseño de Videojuegos

En "The Art of Game Design, A Book of Lenses", Jesse Schell define el diseño de un videojuego como *el acto de decidir qué es lo que un juego tendría que ser* [Jesse Schell, 2008]. Está claro que no se trata de una decisión única sino que de cientos o miles. Muchas de ellas se hacen desde el comienzo, con lápiz y papel. Pero otras se hacen una vez que se ve el juego en acción, dado que nadie posee una imaginación perfecta y no siempre las ideas terminan sintiéndose como uno lo esperaba.

Schell explica que al diseñar un juego se está creando una experiencia, y son los jugadores quienes van a experimentarla. Pero el juego no es la experiencia en sí, sino que es el medio por el cual los jugadores la viven. Y como toda experiencia termina siendo subjetiva; no hay ninguna forma correcta o predeterminada para poder diseñar la herramienta ideal, en este caso el videojuego, para experimentarla.

Pero hay algo que sí se puede saber: qué es lo que un jugador quiere y qué es lo que un jugador espera. Sabiendo estas cosas, es posible ofrecerle la mejor herramienta que explote esos deseos y expectativas. Richard Rouse describe algunos de ellos en "Game Design: Theory & Practice".

Un jugador quiere:

- *Un desafío*: es uno de los factores de motivación más importantes en un juego. Lo importante de los desafíos es que al ser superados siempre se termina aprendiendo algo. Con los videojuegos sucede lo mismo, y aunque a veces la lección aprendida solo es aplicable dentro del ámbito del mismo juego, en contadas veces es trasladable a la vida real.
- *Socializar*: desde los juegos de mesa hasta los videojuegos, casi todos los juegos tienen un componente social, ya sea porque obliga a los jugadores a interactuar entre sí, o porque simplemente actúa como punto de interés común entre diferentes personas.

- *Una experiencia dinámica solitaria*: de la misma forma que hay jugadores que quieren socializar, hay otros que quieren divertirse sin que otras personas los molesten. Pero a diferencia de ver una película o leer un libro, un juego provee algo con lo cual se puede interactuar.
- *Presumir*: al igual que cualquier experiencia competitiva, los juegos también proveen un ámbito en donde las personas pueden alardear de sus habilidades. Ya sea un juego en donde varios jugadores se enfrenten directamente o uno en donde el jugador juegue solo, el hecho de superar a otra persona siempre es un factor muy motivante.
- *Una experiencia emocional*: ya sea un golpe de adrenalina, satisfacción, tristeza, felicidad. Al igual que al ver una pintura o escuchar un disco, cuando una persona interactúa con algo busca que esto lo afecte emocionalmente de alguna forma, ya sea positiva o negativamente.
- *Explorar*: ya sea de forma espacial, recorriendo escenarios nuevos, o intelectual, imaginando nuevas estrategias, la posibilidad de explorar dentro de un juego hace que su experiencia sea mucho más enriquecedora.
- *Fantasear*: al igual que una novela de ficción, los juegos suelen permitirle a los jugadores escapar de su realidad, de su día a día mundano, para sumergirse en un mundo completamente diferente y ser otra persona. Ya sea convertirse en un héroe bárbaro o participar de actividades socialmente no aceptables, la interactividad de los juegos se termina convirtiendo en una forma de catarsis.
- *Interactuar*: todo lo descripto anteriormente se ve habilitado por la posibilidad de interacción. Si una persona no quisiera interactuar, probablemente se limitaría a leer un libro o ver una película, pero el deseo de interactuar es lo que hace la diferencia.

Un jugador espera:

- *Un mundo consistente*: de la misma forma que en el mundo real una acción similar conlleva resultados similares, los jugadores esperan que en el mundo de ficción creado la noción de predictibilidad se mantenga. Luego, un jugador tiene que poder prever, o por lo menos intuir, en todo momento cuál será el resultado de sus acciones.
- *Entender los límites*: dado que se trata de un mundo regido por un número finito de reglas, los juegos, ya sean de mesa o videojuegos, tienen límites claros sobre qué se

puede y qué no se puede hacer en ellos. Al igual que la predictibilidad, los jugadores esperan consistencia en las reglas y que las mismas no cambien inesperadamente.

- *Que una solución razonable funcione*: una vez que se entendieron la reglas del juego, una vez que el jugador es capaz de intuir el resultado de sus acciones, el mismo espera que sea posible solucionar un problema o enigma que se le presente acatando las mismas. Si esto no ocurre, el jugador termina frustrado o confundido.
- *Dirección*: la diferencia entre un juego y la vida real es que en la vida real las personas se ponen sus propios objetivos y deciden como alcanzarlos. Si una persona está intentando escaparse de la vida real, quiere escapar de esa difícil decisión y espera no sólo que se le indique que es lo que tiene que hacer, sino cómo se espera que lo haga.
- *Inmersión*: si bien las personas que juegan entienden que no se trata de algo real, el cerebro humano termina haciendo olvidar este hecho. Al igual que con los libros o películas logran esa ilusión, comúnmente llamada "Suspensión de la incredulidad"[5] , los jugadores suelen esperar que dicha sensación se mantenga a lo largo de todo el juego.
- *Obstáculos*: como se dijo antes, un jugador quiere un desafío. Luego, espera que el juego le presente muchos obstáculos que no sean simples de sobrepasar. Pero dichos obstáculos tienen que ser claros: el jugador tiene que poder reconocer qué es lo que hizo mal y no sentir que alguien está haciendo trampa o impidiendo su progreso de forma arbitraria. Los jugadores tienen que culparse a sí mismos por su derrota y nunca culpar al juego.
- *Poca repetición*: una vez sorteado un obstáculo, descubierto un enigma o superado un desafío, un jugador suele preferir no tener que repetirlo, a menos que el mismo sea especialmente gratificante o entretenido, o a menos que la experiencia del juego esté construida en base al concepto de repetición.
- *No quedarse estancados*: pocas cosas resultan más frustrantes que llegar un punto en donde ya no sea posible avanzar. Estos puntos de estancamiento pueden alcanzarse ya sea por llegar a situaciones complejas que son muy difíciles de resolver o porque

[5] Suspensión de la incredulidad es una expresión que representa la voluntad de un sujeto para dejar de lado su sentido crítico, ignorando incoherencias o incompatibilidades de la obra de ficción en la que se encuentra inmerso (como por ejemplo la existencia del unicornio), permitiéndole adentrarse y disfrutar del mundo de ficción expuesto en la obra.

simplemente era necesario resolver un enigma en algún punto previo al cual no es posible volver.

- *Hacer en lugar de mirar*: el propósito de un juego es que sea interactivo. Si un jugador se pasa más tiempo mirando que interactuando, es decir jugando, entonces no se trata de un juego.
- *Un jugador no sabe lo que quiere, pero sabe cuándo no está presente*: a los jugadores les cuesta mucho poner en palabras que es lo que quieren en un videojuego. Lo más probable es que personas que disfrutan de un mismo género[6], si les preguntan, mencionen características dispares. Pero a pesar de no saber decir qué es lo que quieren, los jugadores se dan cuenta rápidamente cuando eso no está presente en el juego.

Teniendo en cuenta esto, es posible llegar a una conclusión: el diseño de juegos poco tiene que ver con el diseño de sistemas. Mientras que el diseño de un sistema involucra la creación de una herramienta que, a partir de sus funcionalidades, pueda solucionar una problemática determinada, un juego busca solucionar un problema indeterminado por tratarse de algo subjetivo. Luego, resulta difícil definir un estándar para simplificar el proceso de diseño.

Robin Hunicke, Marc LeBlanc y Robert Zubek propusieron un posible enfoque formal para acortar la brecha entre el diseño y el desarrollo de videojuegos, generando una metodología que permitiría clarificar y fortalecer el proceso iterativo de desarrollo, llamada MDA.

El acrónimo MDA viene de las palabras inglesas “Mechanics”, “Dynamics” y “Aesthetics” (Mecánicas, Dinámicas y Estéticas en castellano), entendiendo que las mismas son las contrapartes de los componentes de base de un videojuego: reglas, sistema y diversión.

[6] Es posible agrupar a los videojuegos por género según sus mecánicas básicas: estrategia, acción, aventura, simulación, etc. A su vez, muchos juegos también combinan elementos de diferentes géneros, dando lugar a nuevas categorías.

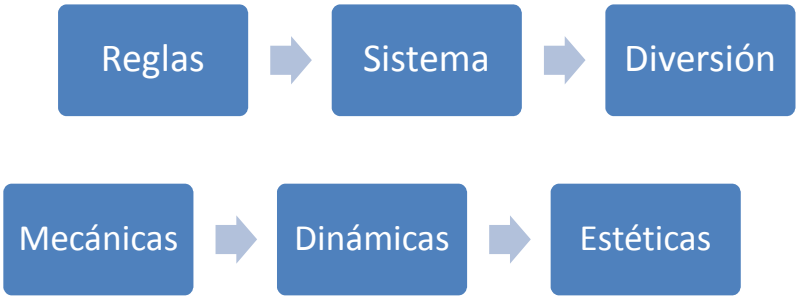

Figura I.1 – Metodología MDA

Se entiende lo siguiente por cada uno de estos componentes:

- *Mecánicas*: describen los componentes del juego.
- *Dinámicas*: describen como los componentes interactúan entre sí y como se ven afectados por las acciones del jugador.
- *Estéticas*: describen las respuestas emocionales esperadas en los jugadores cuando interactúan con el juego. En resumen: describen que es lo que hace que el juego sea divertido.

En los tres casos se busca que las definiciones sean lo más concretas posibles, no dejando lugar para dudas o ambigüedades.

La idea de MDA es partir desde las Estéticas, es decir las respuestas emocionales que se esperan del jugador, para crear Dinámicas que habiliten experimentar dichas emociones y terminar creando Mecánicas que permitan llevar a cabo dicha experiencia. Por ejemplo, en el juego del Truco los componentes serían:

- Estéticas: competencia, camaradería (cuando se juega en equipos).
- Dinámicas: valores de las cartas, como se suma el envido, el orden en el que se realizan las jugadas.
- Mecánicas: mezclado de cartas, distribución de cartas, cantidad de cartas que recibe cada jugador, el envido (si bien se lo menciona dentro de las dinámicas, aquí se hace referencia a la existencia del mismo, mientras que en las dinámicas se habla de cómo se combinan las cartas para la aparición del mismo).

Si bien esta metodología resulta sumamente útil a la hora de comenzar a idear el diseño de un juego, resulta evidente como los componentes subjetivos involucrados, las Estéticas, son difíciles de definir y no siempre resultan lo suficientemente descriptivas. Por otro lado, lo complejo del diseño resultante puede hacer que la línea que divide las Dinámicas de las Mecánicas no sea del todo clara.

A partir de este corto análisis se puede decir que no hay una metodología formal para diseñar videojuegos. Su gran carga subjetiva y emocional hace que muchos cataloguen al diseño de videojuegos como un "arte". Sin embargo, existe la posibilidad de guiarse por ciertos lineamientos y casos de éxito para sentar las bases de un buen diseño que facilite posteriormente la documentación y el desarrollo.

ANEXO II – Conceptos comerciales de los videojuegos para dispositivos móviles

Si bien la comercialización no forma parte del alcance planteado para la presente tesis, es importante marcar cuáles son las metodologías de comercialización existentes actualmente en el mercado para videojuegos para dispositivos móviles, para, de esta forma, dejar sentadas las bases para un posible planteo de comercialización del desarrollo.

Tradicionalmente, el software siempre fue visto como un producto con un precio fijo. Al comprar dicho producto, el cliente obtenía acceso irrestricto al uso del mismo. Si bien la comercialización de la industria del software evolucionó un poco con los años, nunca se alejó mucho de este concepto.

En el caso de los videojuegos para dispositivos móviles se produjo un quiebre importante en esa idiosincrasia, motivado principalmente por la explosión del público casual (el cual, vale la pena aclarar, existía desde antes, pero se vio potenciado con la popularización de los s*martphones*), es decir, aquellos jugadores que no pertenecen al núcleo tradicional de jugadores: personas con un interés limitado que solo buscan pasar unos pocos minutos de ocio. Dichas personas, al tener un interés limitado, también tienen reticencia a invertir grandes sumas de dinero.

Con eso en mente, el mercado evolucionó para proponer diferentes alternativas que permitan ofrecer juegos de alta calidad a precios muy bajos o de forma gratuita. Algunas de dichas alternativas son:

- *Free-to-Play*: este modelo de negocio permite que los usuarios descarguen y jueguen sus juegos de forma completamente gratuita, con la inclusión de una tienda virtual interna para comprar ítems dentro del juego utilizando dinero real. Dichos ítems pueden tratarse de mejoras cosméticas o de mejoras funcionales que permitan avanzar dentro del juego de forma más rápida. Se suele abreviar como F2P.
- *Freemium*: este modelo de negocio también permite la descarga gratuita del juego y también suele contar con una tienda virtual interna, pero a diferencia del modelo anterior, solo permite un acceso limitado al juego completo. Una vez alcanzado dicho límite, es necesario pagar para poder seguir jugando o avanzando en el juego. El término sale de combinar las palabras “Free” y “Premium”.

- *Publicidad*: este modelo de negocios permite la descarga gratuita del juego, pero el mismo incluye secciones con espacios publicitarios los cuales pueden ser vendidos y actualizados en línea. Por lo general se estila incluir una opción de compra para que aquellos jugadores que no deseen ver las publicidades puedan realizar un pago para deshacerse de las mismas de forma permanente. Cabe destacar que muchas veces este modelo suele combinarse con el de Free-to-Play.
- *Premium*: este es el modelo de negocios tradicional, en el cual los usuarios pagan una suma de dinero para poder descargar y acceder al juego. Cabe destacar que, para poder mantener precios en niveles bajos (por lo general entre 1 y 2 dólares), se estila combinar este modelo con el de Free-to-Play.

Es importante remarcar que no siempre la línea entre los diferentes modelos está bien definida y muchas veces se encontraran casos de juegos que aplican diferentes características de cada uno de ellos, creando un producto muy difícil de encuadrar en lo que a su estrategia de comercialización se refiere.

ANEXO III – Geolocalización en dispositivos móviles

La geolocalización es la capacidad para obtener la ubicación geográfica real de un dispositivo y si bien usualmente está relacionada con sistemas de posicionamiento, puede distinguirse de estos por un mayor énfasis en la determinación de una posición significativa (por ejemplo, una dirección de una calle) y no sólo por un conjunto de coordenadas geográficas.

Los dispositivos móviles modernos se caracterizan por contar con funciones de geolocalización que permiten obtener la posición del mismo utilizando tanto la red satelital GPS[7] como métodos alternativos, como ser TDOA[8], cuando no se encuentra disponible el acceso a la misma.

La ventaja con la que cuentan los dispositivos modernos es que los sistemas operativos más populares del mercado, es decir iOS y Android, ya cuentan con interfaces que permiten obtener los datos de ubicación. Sin embargo, hay ciertas dificultades a tener en cuenta a la hora de utilizar estos servicios:

- *Múltiples fuentes de ubicación*: ya sea por GPS o TDOA, es necesario elegir el método que más se adapte a las necesidades de precisión, velocidad de respuesta y eficiencia de batería.
- *Movimiento de usuario*: dado que la posición del usuario suele cambiar, hay que tener en cuenta el movimiento y recalcular nuevamente la posición de forma periódica, si es que así se requiere.
- *Precisión variable*: las posiciones obtenidas de diferentes fuentes no tienen el mismo nivel de precisión, por lo que puede ocurrir que el dato obtenido hace 10 segundos sea más preciso que un nuevo dato obtenido recientemente desde otra fuente o hasta de la misma.

[7] El sistema de posicionamiento global (GPS) es un sistema que permite determinar en toda la Tierra la posición de un objeto (una persona, un vehículo) con una precisión de hasta centímetros, mediante una red de 24 satélites en órbita con trayectorias sincronizadas para cubrir toda la superficie del planeta.

[8] El sistema Time Difference Of Arrival permite el cálculo de la posición de un dispositivo utilizando algoritmos que permiten calcular la posición en base al tiempo de respuesta de un número fijo de estaciones fijas, de las cuales se conoce su posición.

Estas dificultades van a implicar definiciones a la hora de decidir algunos de los puntos clave del flujo de obtención de la ubicación dentro de las aplicaciones. Los puntos clave sobre los cuales hay que decidir son:

- Cuando comenzar a obtener la ubicación: es recomendable hacerlo ni bien se inicia la aplicación o ni bien los usuarios activan la funcionalidad específica que requiere de dichos datos. Cuanto mayor sea el tiempo por el que se consulta se obtendrá una mejor precisión pero implicará un mayor consumo de batería.
- Cuando dejar de obtener la ubicación: esto es muy dependiente de las necesidades de negocio de la aplicación. Se recomienda dejar de hacer la consulta ni bien se cuenta con la información necesaria.
- Mantener el mejor estimado posible: si bien se puede asumir que la última ubicación obtenida es la más precisa, es posible que esto no sea así, por lo que habría que incluir criterios para poder mitigar estos posibles errores:
    - Controlar que la nueva ubicación sea significativamente más reciente que la anterior,
    - Controlar si la precisión informada por la nueva ubicación es mejor o peor que la anterior,
    - Controlar cual es la fuente de la nueva ubicación y determinar si la misma es más confiable que la anterior.

ANEXO IV – High Concept

High Concept

El mundo no es un lugar lo suficientemente grande. Vos, como "General Supremo", tenés que asegurarte la dominación absoluta de tu zona. La única forma de lograrlo es derrotando a todos los Generales cerca tuyo. Conquistá puntos clave, creá tus unidades de ataque y eliminá a tu contrincante antes que él haga lo mismo y pueda eliminarte a vos.

Características

- Vista 2D "top-down" (es decir, vista desde arriba). El juego se controla desplazándose por el mapa arrastrando un dedo (en el caso de jugarse en dispositivos con touch screen) o un cursor y seleccionando elementos con un toque (o click en el caso de jugarse con mouse).
- El juego enfrenta dos jugadores en una lucha por turnos en donde se deberán conquistar diferentes puntos, construyendo una base en los mismos, que proveerán a cada uno de los recursos necesarios para fabricar unidades de ataque, que serán utilizadas para poder eliminar al rival.
- Cada uno de los puntos conquistables poseerá diferentes características que le brindarán diversos beneficios a los jugadores, como ser: mayor generación de recursos, unidades más rápidas, fabricación de unidades más veloces, etc.
- Cada partida ganada otorgará puntos de experiencia que incrementarán el nivel del jugador y "dinero virtual" que puede ser utilizado para comprar "cartas de juego" entre cada partida.
- Cada jugador contará con "cartas de juego" que le permitirán obtener un bonus especial por única vez en un turno, a lo largo de todo una partida o de forma permanente. Las mismas se obtienen entre las partidas con el "dinero virtual obtenido".
- El terreno de juego estará formado por el mapa real de la ciudad en donde se encuentren ambos jugadores y los puntos conquistables se corresponderán a diversos puntos de interés que se encuentren en la cercanía de ambos.

Interface

El juego mostrará una interface simple que consta de:

- Mapa de juego, obtenido a partir de algún servicio de mapas online como ser Google Maps, Bing Maps, etc. (a determinar)
- Botones contextuales, que le permitirán al jugador realizar las diferentes acciones como ser: construir unidades, enviar unidades a atacar, actualizar una base, etc.
- Indicadores sobre los recursos disponibles y tiempo de juego
- Menú de búsqueda de partidas, para encontrar un jugador en las cercanías dispuesto a jugar una partida
- Tienda de compras, para utilizar el dinero virtual obtenido en el juego para comprar cartas de juego.

Motivación

Al tratarse de un juego online competitivo, el simple hecho de vencer a sus adversarios es lo que motiva al jugador a seguir jugando. Para ello resulta importante lograr viralizar el juego para que rápidamente crezca el número de jugadores, haciendo que sea más fácil encontrar un oponente contra el cual jugar y que sea más probable de jugar con una persona diferente cada vez que se juega una partida.

Género

Estrategia por turnos.

Plataforma

Inicialmente pensado para dispositivos móviles Android. Expandible a iPhone y potencialmente también PC, con posibilidad de permitir partidas “cross-platform”.

Audiencia

+13 años

ANEXO V – Game Design Document

Introducción

Concepto

El mundo no es un lugar lo suficientemente grande. Vos, como "General Supremo", tenés que asegurarte la dominación absoluta de tu zona. La única forma de lograrlo es derrotando a todos los Generales cerca tuyo. Conquistá puntos clave, creá tus unidades de ataque y eliminá a tu contrincante antes que él haga lo mismo y pueda eliminarte a vos.

Descripción

Barrio Domination es un juego de estrategia 2D para dispositivos móviles que utiliza geolocalización para que dos jugadores que se encuentran a una cierta distancia se enfrenten entre sí en un escenario definido por el mapa real de sus ubicaciones y diferentes puntos de interés (restaurantes, tiendas comerciales, edificios públicos, etc.).

Al comenzar cada partida se generará un escenario de combate en donde los diferentes puntos de interés cercanos a los jugadores se convertirán en **puntos de conquista**, cada uno con diferentes características (vinculadas al tipo de punto de interés que sean, como ser: restaurante, tienda de ropa o comisaria, por poner algunos ejemplos).

Ambos jugadores tienen que aniquilarse mutuamente, apoderándose de los diferentes **puntos de conquista** para aprovechar sus beneficios y lograr tener una ventaja por sobre su rival.

Experiencia

Barrio Domination es un juego multijugador competitivo. Teniendo en cuenta esto podemos definir los sentimientos que deseamos que el mismo despierte en los jugadores, de forma tal que luego podamos detallar claramente las dinámicas que puedan conducirlo a dicha experiencia y las mecánicas que soporten ese sistema.

Desafío

Los jugadores se estarán enfrentando entre sí. Si bien se ha avanzado mucho en el campo de la Inteligencia Artificial, la mente humana es aún el rival más desafiante que puede encontrar una persona. Luego, los jugadores van a sentirse continuamente desafiados, obligados a aprender, improvisar y pensar nuevas formas de vencer a sus rivales.

Camaradería

Aun tratándose de un juego competitivo, existe un grado importante de interacción humana. Al igual que en los grandes juegos de estrategia como el ajedrez, en donde los grandes rivales se respetan profundamente entre sí (o se tienen un rencor enorme), nuestros jugadores van a formar algún tipo de vínculo con sus rivales, en especial dado que por la cercanía geográfica es posible que se enfrenten habitualmente.

Fantasía

Ambos jugadores van a estar viendo el mundo real desde una nueva perspectiva fantástica. El hecho de que el terreno de juego este basado en el mundo real que rodea a ambos jugadores ayuda a sumergirlos en un mundo de ficción con el cual pueden sentir mayor afinidad por su conexión con la realidad.

Gameplay Básico

El juego tratará sobre dos jugadores luchando entre sí en un mapa que se corresponde al mapa real de su ubicación real, obtenida mediante geolocalización. Ambos jugadores tendrán que fabricar **unidades de ataque**, las cuales se irán **moviendo por las calles**. Para poder ganar la partida, cada jugador tiene que lograr que sus unidades se acerquen a la posición de su rival y le **inflijan daño hasta reducir a cero sus puntos de vida**.

Al iniciar la partida, se toma la posición física de cada jugador y se la establece como su posición inicial en el mapa. Luego, a lo largo de toda la partida, esa será la posición para dicho jugador, manteniéndose inmutable a lo largo de toda la partida.

A su vez, las unidades de ataque de cada jugador se pueden **enfrentar entre sí.** Cuando dos unidades enemigas se encuentran dentro de su rango de ataque, podrán atacarse.

La primera unidad cuyos **puntos de vida lleguen a cero** resultará eliminada. El daño recibido por una unidad se mantendrá por el resto de la partida.

Existen a su vez **puntos de conquista**. La finalidad de los mismos es la de **generar recursos** que sirvan para la estrategia de los jugadores y la de **fabricación de unidades**, para de esa forma poder fabricar múltiples unidades en forma simultánea.

Dependiendo del tipo de **punto de conquista**, el jugador que lo posea tendrá diferentes beneficios, como la fabricación de unidades de forma más rápida o una mayor generación de recursos de algún tipo. Los **puntos de conquista** pueden ser obtenidos por el primer jugador que lo alcance con una de sus unidades cuando se encuentran libres, convirtiéndolos en **centros de comando**. Cuando se encuentran ocupados, es posible reclamarlos ejerciendo una cantidad determinada de daño sobre el mismo.

La partida se desarrollara **por turnos**, a lo largo de los cuales cada jugador realizará todas las acciones que le sean posibles o que así desee según su estrategia. Cada acción a realizar por una unidad en una partida, ya sea moverse o atacar, requerirá de **puntos de acción**. La cantidad de puntos de acción necesarios para cada caso dependerán de la acción a realizar y la unidad en cuestión.

Para poder fabricar unidades, los jugadores requerirán **puntos monetarios**. Los mismos se utilizan como “moneda” a la hora de fabricar unidades.

El mapa de la partida estará formado por el mapa real de la ciudad en donde se encuentran los jugadores. A su vez, los **puntos de conquista** del mapa se generaran a partir de diferentes puntos de interés real (tiendas, oficinas gubernamentales, etc.) que se encuentren dentro del radio de los jugadores, siendo determinada la cantidad de los mismos por la distancia que separa a ambos jugadores.

Gameplay Extendido

Recursos

Los recursos son el elemento básico que les permiten a los jugadores poder realizar las diferentes acciones para lograr la victoria. Los mismos se obtienen a partir de los diferentes **puntos de conquista** que el jugador posea, por lo que a mayor cantidad de **puntos de conquista**, mayor será la entrada de recursos, pero al mismo tiempo será mayor la cantidad de lugares a defender.

El juego consta de dos recursos:

- **Puntos Monetarios** (PM): son "el dinero" del juego. Se utilizan para crear unidades.
- **Puntos de Acción** (PA): cada acción del juego, sea mover una unidad o hacer que ataque, requiere de una cantidad determinada de puntos de acción.

Tanto los PM como los PA solo son válidos a lo largo de una partida. Ambos jugadores comenzaran con la misma cantidad de PM y PA, y será su responsabilidad alcanzar los niveles óptimos de generación de los mismos que les sirvan para vencer a su rival. Los PM y PA no utilizados no se mantienen entre partidas, pero si se mantienen entre los turnos de una misma partida.

Unidades

Las unidades son utilizadas por los jugadores para atacarse mutuamente. Las mismas son la única forma que un jugador tiene de atacar al otro, conquistar puntos de conquista, atacar y destruir otras unidades y centros de comando.

Los jugadores pueden construir unidades utilizando PM desde su punto inicial como también desde los Centros de Comando, por lo que a mayor Centros de Comando poseídos, mayor cantidad de unidades podrán construirse de forma simultánea.

Características Comunes

- Vida
- Daño
- Distancia de ataque

- Velocidad (distancia que se puede desplazar en un movimiento)
- Costo de PA para moverse
- Costo de PA para atacar
- Costo de PM para fabricación
- Tiempo de fabricación (medido en turnos)

Movimiento

Las unidades se mantienen estáticas por defecto. Cuando el jugador desee que se muevan, tendrá que seleccionar la posición en el mapa a la que quiere que se dirijan. Luego, se utilizarán servicios de geolocalización para calcular la ruta a seguir de acuerdo a las trazas de las calles. Dicho cálculo no tiene costo alguno, ni de PM ni de PA.

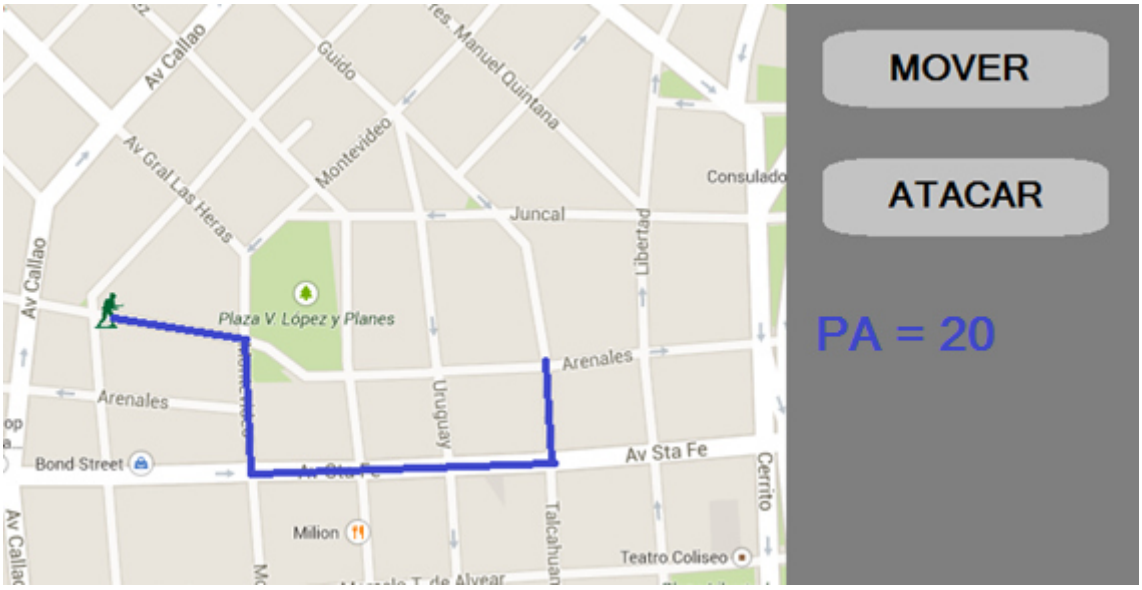

Figura V.1 – Cálculo de recorrido de unidades

Para moverse, el jugador tiene que explícitamente indicarle a cada unidad que se mueva. Una vez que se haya seleccionado un punto de destino y se haya calculado una ruta acorde, el jugador podrá moverse hacia allí. Cuando lo haga, esta unidad consumirá una cantidad determinada de PA y se moverá una distancia determinada, acorde a las características de la misma.

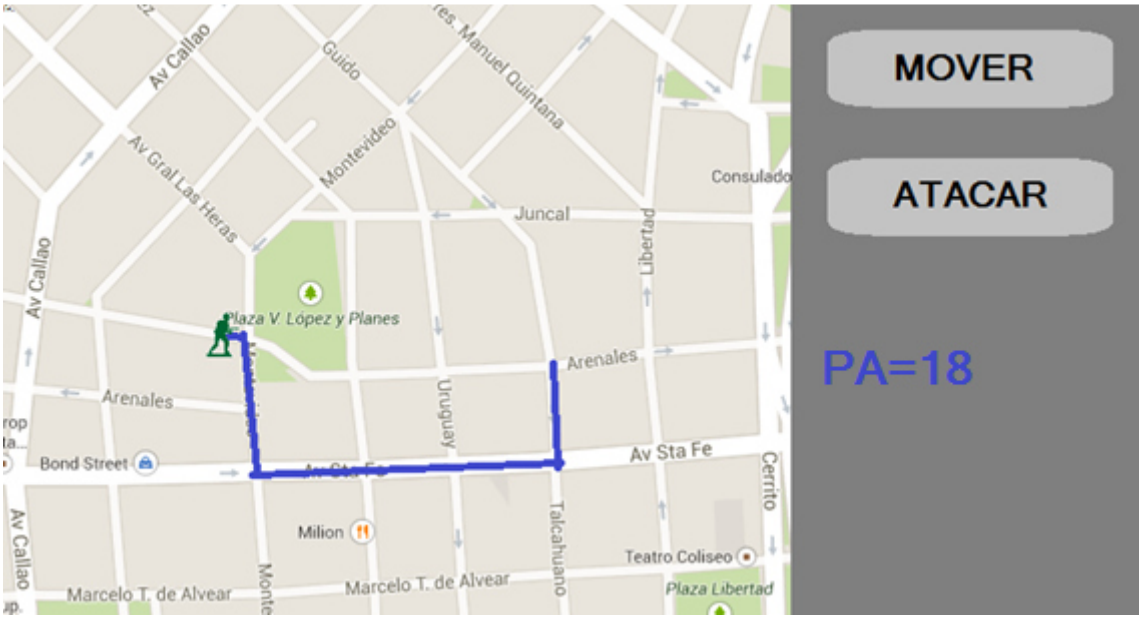

Figura V.2 – Movimiento de unidades

Ataque

Cuando una unidad enemiga se encuentra dentro de la distancia de ataque de una propia, el jugador puede elegir atacarla. Esto consumirá una cantidad determinada de PA. La distancia de ataque determina un radio: toda unidad enemiga que este dentro de la circunferencia determinada por dicho radio, podrá ser atacada.

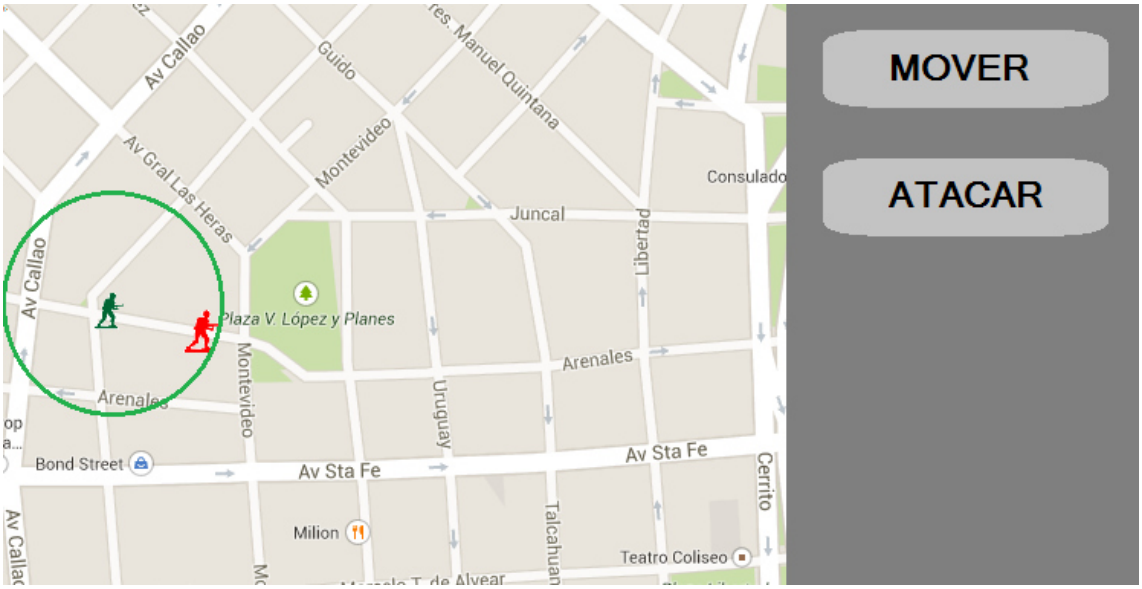

Figura V.3 – Radio de ataque de unidades

Al atacar una unidad enemiga, la misma tiene una chance del 33% (1 de cada 3 veces) para contra-atacar automáticamente, sin consumir ningún tipo de recurso del jugador rival durante esta acción.

Las formulas utilizada durante el ataque y contra-ataque, que determina el daño que se harán las unidades, son las siguientes:

$$\text{DAÑO ATAQUE} = \text{DAÑO}^{(1)} * \text{RAND}(0.9 ; 1.1)^{(3)} \quad \text{(V. 1)}$$

$$\text{DAÑO CONTRA-ATAQUE}^{(4)} = \text{DAÑO}^{(2)} * \text{RAND}(0.1 ; 0.2)^{(3)} \quad \text{(V. 2)}$$

(1) Hace referencia al valor del daño que puede realizar la unidad atacante
(2) Hace referencia al valor del daño que puede realizar la unidad contraatacante
(3) Función que devuelve un número de punto flotante al azar entre los dos indicados
(4) El Daño Contra-ataque solo se aplicará en caso de que corresponda según la chance de 1/3 mencionada anteriormente.

Sin embargo, se impone una restricción: el DAÑO CONTRA-ATAQUE no puede ser mayor que el 30% del DAÑO ATAQUE. En caso que eso suceda, el valor del DAÑO CONTRA-ATAQUE quedará recordado en dicho valor (es decir: 0.3 * DAÑO ATAQUE).

Luego, es posible que una unidad atacante con poca vida sea destruida al atacar una unidad rival, sin que la rival sea destruida en el proceso. Esto dependerá del daño que cada unidad realice, la vida que les quede disponible, y una cuota de azar.

A la hora de atacar Centros de Comandos rivales, la fórmula que aplica es la misma. La única diferencia es que los Centros de Comando no pueden contraatacar, por lo que la unidad atacante no sufrirá ningún daño. Lo mismo aplica para un Comando Central.

Más allá de la necesidad de contar con suficientes PA para poder atacar, no existe ninguna otra limitante para hacerlo. Luego, una unidad puede atacar repetidamente a otra, o a un centro de comando, tantas veces como recursos tenga disponible el jugador.

Unidad A: TROPA

Las unidades del tipo A, TROPAS, tienen la particularidad de ser la única unidad que puede moverse sin seguir el sentido de movimiento de las calles. Luego, al calcular el

recorrido para llegar de un punto a otro, le indicará al servicio correspondiente que se está moviendo “de a pie”.

Los valores de las características comunes para dicha unidad son:

- Vida : 10 puntos
- Daño : 2 puntos
- Distancia de ataque : 200 metros*
- Velocidad : 150 metros*
- Costo de PA para moverse: 1 puntos
- Costo de PA para atacar: 1 puntos
- Costo de PM para fabricación: 5 puntos
- Tiempo de fabricación: 1 turno

*NOTA: tanto éste como todos los otros valores similares que se presentaran de aquí en más están basadas en la escala en el mundo real, calculados a partir de las posiciones en el mapa.

Unidad B: MOTOS

Las unidades del tipo B, MOTOS, se mueven siguiendo el sentido de las calles. Luego, al calcular el recorrido para llegar de un punto a otro, le indicará al servicio correspondiente que se está moviendo “en vehículo”.

Los valores de las características comunes para dicha unidad son:

- Vida : 25 puntos
- Daño : 4 puntos
- Distancia de ataque : 200 metros
- Velocidad : 300 metros
- Costo de PA para moverse: 2 puntos
- Costo de PA para atacar: 1 puntos
- Costo de PM para fabricación: 15 puntos
- Tiempo de fabricación : 2 turnos

Unidad C: TANQUES

Las unidades del tipo C, TANQUES, se mueven siguiendo el sentido de las calles. Luego, al calcular el recorrido para llegar de un punto a otro, le indicará al servicio correspondiente que se está moviendo "en vehículo".

Los valores de las características comunes para dicha unidad son:

- Vida : 40 puntos
- Daño : 8 puntos
- Distancia de ataque : 400 metros
- Velocidad : 100 metros
- Costo de PA para moverse: 5 puntos
- Costo de PA para atacar: 2 puntos
- Costo de PM para fabricación: 25 puntos
- Tiempo de fabricación : 3 turnos

Escenario

El escenario del enfrentamiento entre los jugadores se generará al momento de iniciar la partida. El mismo estará formado por el mapa real de la región formada por la ubicación de ambos jugadores y las zonas que se encuentran entre ellos.

Existe una condición de base para la distancia entre ambos jugadores: para poder iniciar una partida, la misma tiene que ser mayor o igual a **2 km** y menor o igual a **30 km**.

A su vez, en el escenario existirán diversos **puntos de conquista**, que al ser capturados por un jugador le brindarán diferentes bonificaciones que dependerán del tipo de **punto de conquista** que se trate. La selección de dichos puntos se efectuará de la siguiente forma:

1. Se obtienen los diversos puntos de interés de diferentes categorías cercanos a cada uno de los jugadores.
2. Se buscan las coincidencias entre ambos listados para asignarles una mayor prioridad.
3. Se ordenan los puntos restantes en base a la cercanía promedio a ambos jugadores.

4. Se divide el listado por categoría.
5. Comienzan a tomarse de forma alternante puntos de interés de cada categoría y se crean a partir de los mismos **puntos de conquista**, hasta que se alcanza el número de **puntos de conquista** deseados.

El número de **puntos de conquista** dependerá de la distancia entre ambos jugadores: a mayor distancia, mayor será el número de **puntos de conquista**, siguiendo la siguiente fórmula:

- **Si D <= 5**

  N = 4

- **Si D < 20**

  N = 20 – [(5 / D) * 1.2 * (20 – D)] (V. 3)

- **Otros casos**

  N = 20

N: Cantidad de puntos de conquista

D: Distancia entre jugadores, medida en km

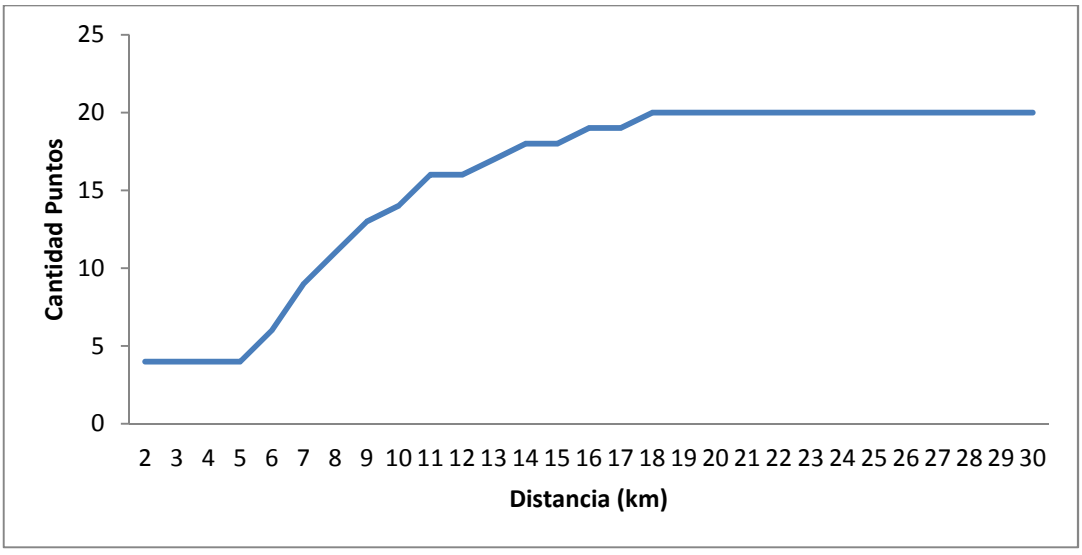

Figura V.4 – Relación de Puntos de Conquista según distancia

Puntos de Conquista/Centro de Comando

Los **puntos de conquista** se obtienen desde el servicio de mapas a partir de buscar puntos de interés para diferentes características (quiosco, restaurante, tienda de ropa, etc.).En la sección ESCENARIO se hizo una descripción de cómo funciona la elección de los puntos de conquista.

Cuando un jugador alcanza un **punto de conquista** con cualquiera de sus unidades, el mismo resulta ocupado y se convierte en un **centro de comando**, activándose sus beneficios (cabe destacar que los mismos empezarán a contabilizarse en el siguiente turno, en caso de que el jugador en cuestión aún lo esté ocupando) para el jugador que lo haya ocupado, además de pasar a estar disponible para que el jugador pueda utilizarlo para construir unidades (esto ocurre inmediatamente luego de haber sido ocupado).

Cuando un **centro de comando** es destruido, el usuario que lo destruyó pasa a ocuparlo. Se restablece al valor inicial la **vida** del mismo. Cabe destacar que al ocupar un **centro de comando** rival ocurre lo mismo que al momento de ocupar un **punto de conquista**: los beneficios se empezarán a percibir a partir del próximo turno, mientras que se podrá utilizar el mismo de inmediato para construir unidades.

A continuación se describirán las características comunes de todos los centros de comando y se detallaran las características propias de cada tipo de punto, indicando también a que categoría de punto de interés pertenecen los mismos (recordar que dichas categorías son las que se buscarán inicialmente para formar el listado de puntos de conquista de la partida).

Características comunes

La función principal de los centros de comando es la de actuar como:

- Generadores de recursos: cada **centro de comando** genera PA y PM. La cantidad generada de cada uno de estos recursos dependerá del tipo.
- Fábrica de unidades: cada **centro de comando** puede ser utilizado para fabricar unidades. Cada unidad tarda un tiempo determinado en fabricarse, medido en cantidad de turnos. Cada **centro de comando** puede generar **múltiples unidades**

simultáneamente. Todos los **centros de comando** fabrican unidades de la misma forma, sin presentar diferencias entre sí en este aspecto.

- Vida: cantidad de daño que puede recibir antes de ser destruido. Cuando un centro de comando es destruido, pasa a ser ocupado por el jugador que lo destruyó, volviendo la cantidad de puntos de vida a su valor inicial.

Punto A

Estos **puntos de conquista** se generan a partir de los puntos de interés que respondan a la categoría "**restaurante**".

Sus características son

- PA generados por turno: 5 puntos
- PM generados por turno: 5 puntos
- Vida: 30 puntos

Punto B

Estos **puntos de conquista** se generan a partir de los puntos de interés que respondan a la categoría "**banco**".

Sus características son

- PA generador por turno: 3 puntos
- PM generados por turno: 10 puntos
- Vida: 20 puntos

Punto C

Estos **puntos de conquista** se generan a partir de los puntos de interés que respondan a la categoría "**juguetería**".

Sus características son

- PA generador por turno: 10 puntos
- PM generados por turno: 3 puntos
- Vida: 20 puntos

Comando Central

El Comando Central de cada jugador estará ubicado en la posición en la que el jugador se encontraba al momento de iniciar la búsqueda de la partida, no moviéndose de la misma a pesar de que el jugador se esté moviendo en la vida real.

El Comando Central es el punto inicial de fabricación de unidades y generación de recursos y funciona igual que un Centro de Comando, con la diferencia que al ser destruido, el jugador rival gana la partida.

Sus características son

- PA generados por turno: 5 puntos
- PM generados por turno: 2 puntos
- Vida: 80 puntos

Carta Comodín

Cada jugador cuenta con tres cartas comodín para utilizar a lo largo de cada partida. Una vez consumidas, las mismas no estarán disponibles hasta que termine la partida y el jugador inicie una nueva. Las mismas sólo pueden ser utilizadas a razón de una por turno.

Al utilizar esta carta, todos los centros de comando que posea el jugador se adelantarán un turno, lo cual significa que:

- Los mismos volverán a generar los PA y PM que generan al inicio de un turno, los cuales estarán disponibles al instante para que el jugador utilice en el turno actual
- Las unidades que estén en construcción se adelantarán un turno, por lo que si a alguna unidad le faltaba un turno para completarse, la misma será creada, pudiendo ser utilizada por el jugador en el turno actual (en el caso de que le falten más turnos, se le resta uno y estará disponible una vez que hayan pasado los turnos restantes para ser completada).

Ranking y Experiencia

Cada jugador cuenta con un ranking determinado por las victorias del mismo. Un jugador ganará experiencia con cada partida ganada. Si la cantidad de experiencia acumulada

alcanza un valor determinado, el jugador avanzará un punto de ranking, volviéndose a cero la cantidad de experiencia. Esto continúa de forma indefinida.

La cantidad de experiencia necesaria para avanzar un punto de ranking dependerá exclusivamente del nivel del jugador según la siguiente fórmula:

$$\textbf{EXPERIENCIA NECESARIA} = \textbf{RANK}_{\textbf{propio}}^{1,2} \textbf{ x 10} \qquad \textbf{(V. 4)}$$

La cantidad de experiencia obtenida al ganar una partida dependerá del nivel de los dos jugadores que hayan participado en la misma, según la siguiente fórmula:

$$\textbf{EXPERIENCIA GANADA} = (\textbf{RANK}_{\textbf{rival}} / \textbf{RANK}_{\textbf{propio}}) \textbf{ x } \textbf{RANK}_{\textbf{propio}}^{1,2} \qquad \textbf{(V. 5)}$$

NOTA: todos los valores serán redondeados al número entero más cercano.

Si al sumar los puntos de experiencia el jugador avanza de ranking, el excedente de puntos de experiencia se suman al contador de experiencia del nuevo ranking, pudiendo de esta forma avanzarse varios ranking de una sola vez.

En caso de perder la partida el jugador no sumará ni restará puntos de experiencia, no viéndose afectado su ranking de ninguna forma.

Condiciones iniciales

Ambos jugadores comienzan la partida en igualdad de condiciones, siendo las mismas las siguientes:

- **PM** iniciales: 8
- **PA** iniciales: 5
- Una unidad **TROPA** ya fabricada, disponible para su uso, al lado del Comando Central de cada jugador.

Flujo del juego

A continuación se describirá el flujo de ejemplo de una partida desde que los jugadores ejecutan el juego hasta que uno de ellos resulta vencedor.

1. El JUGADOR 1 ejecuta el juego y elige la opción BUSCAR PARTIDA.

2. El JUGADOR 2 ejecuta el juego y elige la opción BUSCAR PARTIDA
3. El server determina que JUGADOR 1 y JUGADOR 2 se encuentran en el rango de distancias correcto para iniciar una partida y crea una partida con ambos jugadores (la creación de la partida implica la creación del escenario con los puntos de interés correspondientes).
4. El juego del JUGADOR 1 consulta periódicamente al server para saber si se ha encontrado una partida. Como resulta que se ha encontrado una partida, el mismo confirma que se ha unido a la misma y aguarda la confirmación por parte del JUGADOR 2
5. El juego del JUGADOR 2 consulta periódicamente al server para saber si ha encontrado una partida. Como resulta que se ha encontrado una partida, el mismo confirma que se ha unido a la misma y, dado que el JUGADOR 1 ya se ha unido, da comienzo a la partida.
6. El JUGADOR 1 tiene el primer turno. Crea unidades y las mueve de acuerdo a sus PM y PA disponibles. Una vez que no puede (por no contar con PA o PM suficientes) o no desea (aun cuando le resten PM o PA suficientes) realizar una nueva acción, da por finalizado su turno.
7. El JUGADOR 2 tiene el segundo turno. Crea unidades y las mueve de acuerdo a sus PM y PA disponibles. Una vez que no puede (por no contar con PA o PM suficientes) o no desea (aun cuando le resten PM o PA suficientes) realizar una nueva acción, da por finalizado su turno.
8. El JUGADOR 1 tiene el tercer turno. Crea unidades y las mueve de acuerdo a sus PM y PA disponibles. Una vez que no puede (por no contar con PA o PM suficientes) o no desea (aun cuando le resten PM o PA suficientes) realizar una nueva acción, da por finalizado su turno.
9. Se repite el ciclo, hasta que un jugador le quita todos los puntos de vida al Comando Central del jugador rival, momento en el cual dicho jugador resulta vencedor (por ejemplo, el JUGADOR 1).
10. Ambos jugadores son informados del resultado de la partida, indicándoles también la cantidad de puntos de experiencia ganados y el ranking actual al que el jugador pertenece.

ANEXO VI - Cálculo de Distancias entre puntos geográficos

MODELOS MATEMÁTICOS

Para poder calcular la distancia entre dos puntos, mediante un modelo matemático, es necesario plantear cómo se encuentran relacionados, ya que dicha relación va a determinar la complejidad y el tiempo del cálculo. La relación entre los puntos está vinculada a la forma con la cual se representa el modelo, por tal motivo, se debe definir un modelo que representa a la realidad lo mejor posible.

Se sabe que la Tierra no tiene una forma geométrica uniforme, con lo cual la misma puede ser representada como una esfera, un elipsoide (esfera achatada) o un geoide (forma casi esférica que evidencia un leve achatamiento en sus extremos).

Figura VI.1 – Forma de la Tierra

La forma que mejor representa a la Tierra es un geoide, pero por tratarse de una forma irregular conlleva a necesitar una mayor potencia y tiempo de cálculo para obtener las distancias entre 2 puntos.

En el caso de puntos muy próximos se puede considerar a la Tierra como un plano ya que la curvatura de la Tierra no influye en el cálculo de la distancia.

ESFERA

Es la representación más sencilla dada la simplicidad de la trigonometría esférica, pero se tiene que tener en cuenta que con dicho modelo se obtiene menor precisión. Para el cálculo de la distancia entre dos puntos situados en la superficie de una esfera se utiliza

la ***Fórmula de Haversine***, que es más precisa frente a otras alternativas como la ***Ley de los Cosenos*** o el ***Teorema de Pitágoras***.

ELIPSOIDE

Para un elipsoide existe un algoritmo iterativo conocido como ***Fórmulas de Vincenty*** que proporciona una precisión mejor que la obtenida por las fórmulas utilizadas para la esfera. El problema de este método es que al ser iterativo, puede provocar un bucle infinito en el cálculo para puntos casi antipodales (puntos situados en lados opuestos de la tierra).

Fórmulas

Teorema de Pitágoras

Si la distancia es menor a 20 km y la ubicación de los 2 puntos se encuentra expresada en Coordenadas Cartesianas por los pares (X1, Y1) y (X2, Y2), entonces se puede utilizar el modelo de Tierra plana, por la proximidad de los puntos, y aplicar el Teorema de Pitágoras para obtener la distancia entre los puntos mediante la siguiente ecuación:

$$d = \sqrt{(X2 - X1)^2 + (Y2 - Y1)^2} \qquad \text{(VI. 1)}$$

Donde ***d*** es la distancia entre los 2 puntos y se encuentra expresada en la misma unidad que las coordenadas.

La desventaja que presenta este cálculo es que los puntos tienen que estar expresados en Coordenadas Cartesianas, ya que si no se encuentran en dicha unidad se tiene un alto costo de cálculo para la conversión de Coordenadas Esféricas a Cartesianas para luego utilizar el modelo de Tierra plana.

Al utilizar esta fórmula se obtienen los siguientes errores:

- Menos de 30 metros para latitudes inferiores a 70 grados.
- Menos de 20 metros para latitudes inferiores a 50 grados
- Menos de 9 metros para latitudes inferiores a 30 grados.

Estos errores tienen en cuenta los efectos de convergencia de los meridianos y la curvatura de los paralelos.

Fórmula de Haversine – Calculo de la Distancia del Círculo Máximo

Si se considera a la Tierra como una esfera de radio R, y la ubicación de los 2 puntos se encuentra expresada en Coordenadas Esféricas (latitud, longitud) por los pares (lat1, long1) y (lat2, long2), entonces se puede utilizar la Formula de Haversine:

$$c = 2 * \arcsin(\sqrt{\sin^2(\frac{\phi_2 - \phi_1}{2}) + \cos(\phi_1) * \cos(\phi_2) * \sin^2(\frac{\lambda_2 - \lambda_1}{2})}) \quad \text{(VI. 2)}$$

$$d = R * c \quad \text{(VI. 3)}$$

Donde ϕ_1, ϕ_2 y λ_1, λ_2 se refiere a la latitud y longitud, expresadas ambas en radianes. Para convertir coordenadas decimales en radianes se debe multiplicar por $\frac{\pi}{180}$, y para convertir de radianes a coordenadas decimales se debe multiplicar por $\frac{180}{\pi}$.

El resultado intermedio ***c*** es la distancia sobre el gran cirulo[9] en radianes. La distancia sobre el gran círculo ***d*** estará en las mismas unidades que R.

Para proteger de posibles errores de redondeo que podrían afectar el cálculo computacional del arcoseno si los dos puntos son antipodales (ubicados en lados opuestos de la Tierra), se aplica la función mínimo al término que se encuentra dentro del arcoseno, como se muestra a continuación:

$$a = \sqrt{\sin^2(\frac{\phi_2 - \phi_1}{2}) + \cos(\phi_1) * \cos(\phi_2) * \sin^2(\frac{\lambda_2 - \lambda_1}{2})} \quad \text{(VI. 4)}$$

$$c = 2 * \arcsin(\min(1, a)) \quad \text{(VI. 5)}$$

$$d = R * c \quad \text{(VI. 6)}$$

[9] El **gran círculo**, denominado también **círculo mayor** o **círculo máximo**, es el círculo resultante de una sección realizada a una esfera mediante un plano que pase por su centro y la divida en dos hemisferios; la sección circular obtenida tiene el mismo diámetro que la esfera.
La distancia más corta entre dos puntos de la superficie de una esfera siempre es el arco de círculo máximo que los une.

Bajo estas condiciones, la fórmula de Haversine tiene condiciones adversas, pero el error que puede ser de hasta 2 Km está en el contexto de una distancia de cerca de 20.000 Km.

Ley de los Cosenos

$$a = \sin(lat_1) * \sin(lat_2) \quad \text{(VI. 7)}$$

$$b = \cos(lat_1) * \cos(lat_2) * \cos(long_2 - long_1) \quad \text{(VI. 8)}$$

$$c = \arccos(a + b) \quad \text{(VI. 9)}$$

$$d = R * c \quad \text{(VI. 10)}$$

Aunque esta fórmula es matemáticamente exacta, es poco fiable para pequeñas distancias porque el arcocoseno presenta un mal comportamiento cerca del origen. Esto se puede ver en la pequeña tabla que se muestra a continuación,

cos (5 grados) = 0.996194698

cos (1 grado) = 0.999847695

cos (1 minuto) = 0.9999999577

cos (1 segundo) = 0.9999999999882

cos (0.05 segundos) = 0.999999999999971

Realizando las operaciones con 7 cifras significativas no es posible diferenciar cosenos de distancias más pequeñas que un minuto de arco.

Calculo del ángulo o dirección entre 2 puntos geográficos

Se define ángulo o dirección entre 2 puntos geográfico como el ángulo formado entre la línea Norte-Sur de la Tierra (meridiano) que pasa por el punto de origen y la recta que conecta ambos puntos. Dicho ángulo se lo considera una unidad de dirección tomando como referencia el Polo Norte, y se lo calcula siguiendo el sentido de las agujas del reloj.

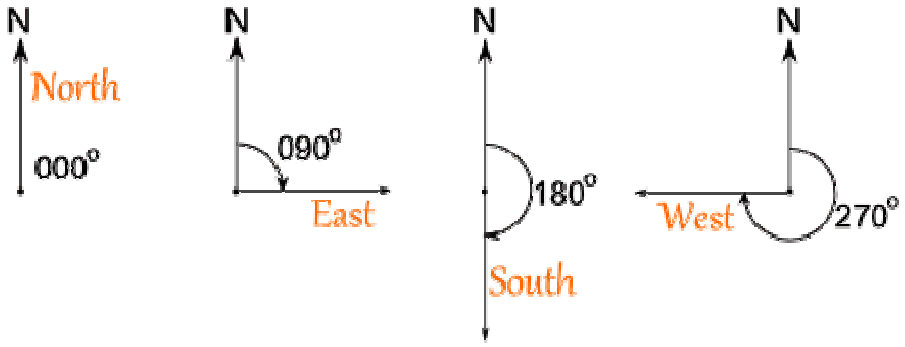

Figura VI.2 – Ángulos formados entre los puntos cardinales y el Polo Norte
Fuente: http://www.mathsisfun.com/measure/compass-north-south-east-west.html

$$\theta = \arctan^2\left(\sin(\Delta\lambda)\bullet\cos(\varphi_2);\cos(\varphi_1)\bullet\sin(\varphi_2)-\sin(\varphi_1)\bullet\cos(\varphi_2)\bullet\cos(\Delta\lambda)\right) \quad \text{(VI. 11)}$$

$(\varphi_1,\lambda_1) \equiv$ Coordenadas del punto de inicio, expresadas en radianes.

$(\varphi_2,\lambda_2) \equiv$ Coordenadas del punto final, expresadas en radianes.

$\Delta\lambda = \lambda_2 - \lambda_1 \equiv$ Diferencia entre las longitudes.

$\theta \equiv$ Ángulo o dirección entre los 2 puntos, expresado en radianes.

$\arctan^2$ retorna valores en el rango –n … +n (-180º … +180º), para normalizar el resultado y obtener valores en el rango 0º … 360º, se deben transformar los valores negativos (-) al rango 180º … 360º. Para ello primero se tiene que convertir a grados el valor de θ y después realizar el siguiente cálculo $(\theta+360)\%360$, donde % es la función módulo.

Calculo para obtener del punto de destino en función de un ángulo (dirección) y una distancia a partir de un punto inicial.

Utilizando como datos el punto inicial, el ángulo (dirección) inicial y una distancia se pueden obtener las coordenadas del punto final a lo largo del Círculo Máximo.

$$\varphi_2 = \arcsin\left(\sin(\varphi_1)\bullet\cos(\delta)+\cos(\varphi_1)\bullet\sin(\delta)\bullet\cos(\theta)\right) \quad \text{(VI. 12)}$$

$$\lambda_2 = \lambda_1 + \arctan^2\left(\sin(\theta)\bullet\sin(\delta)\bullet\cos(\varphi_1);\cos(\delta)-\sin(\varphi_1)\bullet\sin(\varphi_2)\right) \quad \text{(VI. 13)}$$

$(\varphi_1, \lambda_1) \equiv$ Coordenadas del punto de inicio, expresadas en radianes.

$d \equiv$ Distancia a recorrer

$R \equiv$ Radio Terrestre

$\delta = \frac{d}{R} \equiv$ Distancia angular, expresada en radianes.

$\theta \equiv$ Ángulo (dirección) inicial, expresado en radianes.

$(\varphi_2, \lambda_2) \equiv$ Coordenadas del punto final, expresadas en radianes.

Para que la longitud obtenida se encuentre expresada en el rango -180º…180º se la debe transformar utilizando la siguiente ecuación $((\lambda_2 + 540)\%360) - 180$, donde λ_2 debe estar expresado en grados.