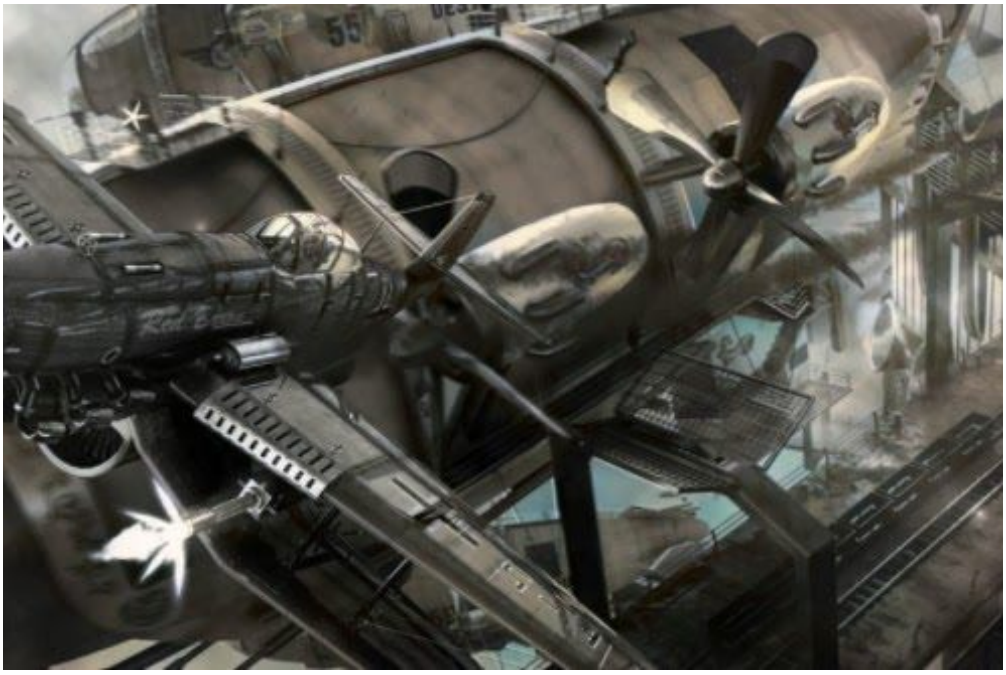


UNIVERSIDAD DE LA EMPRESA

# PROYECTO I

## Licenciatura en Informática

## Febrero - Marzo 2021


**Tutor:**
Ing. Adinolfi, Emiliano
**Integrantes:**

| | | |
|---|---|---|
| Bertoldi, | Fernando | 6.025.535-8 |
| Garciarena, | Sebastian | 5.205.441-7 |
| Segovia, | Joaquín | 4.739.544-4 |
| Serrentino, | Sebastian | 4.463.556-8 |
| Silva, | Ramiro | 4.652.148-4 |

# Indice:

# Introduccion

Se presenta en el presente documento, la resolución del Proyecto de Tercero de la Universidad de la Empresa. Que tiene como fin, converger en la resolución de un juego, todos los conocimientos adquiridos hasta el momento en la carrera. Gestionando la construcción [desarrollo] de una pieza de software completa y funcional, siguiendo las mejores prácticas de la Ingeniería de software.

# Objetivos

Se requiere la construcción de un juego, ambientado en hechos reales. Teniendo como actores, a los alumnos (implementando y documentando el proceso) y como contraparte el cuerpo docente de la UDE (corrigiendo la solución propuesta).

En concreto este documento tiene como principal propósito, ser el registro de las actividades/decisiones llevadas a cabo por el grupo a lo largo del proyecto. Desde el análisis inicial hasta la puesta en producción, aplicando los conocimientos adquiridos hasta el momento en la carrera.

# Descripción del problema

El juego deberá representar combates aéreos entre aviones. Se dispuso tomar como referencia aquellos combates que se dieron durante la primera guerra mundial, básicamente sobre el año 1916, entre el ejército Francés – Aleman, en el frente occidental. Cuyo objetivo será alcanzar la supremacía en el aire, cubriendo recursos y tropas en tierra. En este marco se plantea la realización del juego con los requerimientos así como toda la documentación pertinente.

# Especificación de Requerimientos

## Proposito

El objetivo es detallar y describir el comportamiento del juego, a partir de los requerimientos planteados por el cliente.

## Requerimientos Funcionales

**Requerimiento funcional 1: Ingreso al juego:**

El juego tendrá una pantalla de inicio, en donde se desplegará una instancia de login. En caso de ya estar registrado, el usuario deberá ingresar su usuario y contraseña. De lo contrario, tendrá la posibilidad de registrarse.

En el caso que deba registrarse, el usuario será dirigido a una nueva pantalla donde deberá ingresar su nombre de usuario (único para cada usuario), su contraseña elegida y confirmación de contraseña.

Una vez realizado el ingreso, de un usuario existente, se mostrará el Menú Principal.

**Requerimiento funcional 2: Menú Principal**

Opciones { Crear Partida, Unirse a Partida, Cargar Partida(requiere login), Salir }

El sistema contará con un menú principal que mostrará las siguientes opciones: Crear Partida, Unirse a Partida, Cargar Partida(requiere login), Salir.

**Requerimiento funcional 3: Crear partida**

Mostrar código para que se una otro jugador.

Cuando se seleccione “Crear Partida” se desplegará en pantalla un identificador de partida, de modo que el usuario podrá compartirlo con otro usuario y así jugar una partida. Solo este usuario será quien pueda guardar la partida sin terminar.

**Requerimiento funcional 4: Unirse a partida**

Campo para insertar código y botón para unirse a otra partida.

La sección de “Unirse a Partida” mostrará un listado con las partidas que estén creadas al momento. Al seleccionar la partida, el usuario podrá unirse a la partida quedando a la espera de jugador contrincante.

**Requerimiento funcional 5: Pausar Partida**

Botón de pausa para desplegar el menú dentro del juego (por ejemplo para guardar).

En cualquier momento de la partida, cualquiera de los jugadores puede pausar. Solo aquel que pauso el juego, puede Abandonar/Reanudar/Guardar (si fue quien creó la partida). El otro jugador solo puede Abandonar o esperar.

**Requerimiento funcional 6: Guardar partida**

Botón de pausa-->Guardar la partida, el usuario quien creó la partida puede guardar.

Es una foto completa de todo el tablero. Luego la redirección al panel.

Durante el desarrollo del juego, aquel usuario que creó la partida, podrá guardar la partida en el momento que lo desee, persistiendo toda la partida. Una vez guardada la partida, se redireccionará al panel.

**Requerimiento funcional 8: Cargar partida**

Require login. Carga a partir de un listado de tus partidas guardadas.

El juego contará con una opción para cargar partidas guardadas que permitirá seleccionar de un listado de las partidas previamente guardadas por el usuario. Cada línea del listado dirá el nombre de la partida, ld de la partida y tendrá un botón para unirse

**Requerimiento funcional 9: Distribución de vistas y elementos del tablero en el Juego**

El juego contará con tableros de información en la pantalla, cada jugador contará con información del estado de sus elementos de juego y un mecanismo para una vista lateral.

**Requerimiento funcional 10: Aviones (x4) manejados por jugador (se manipula de a 1 avión)**
Todo lo que implica el manejo del avión, diseño, elecciones y desarrollo del objeto.

El juego, se dará entre 2 equipos,en donde cada uno tendrá en primera instancia 1 jugador. En futuras versiones el sistema prevé que sean conformados por más jugadores.

Cada jugador tendrá 4 aviones manejados, los cuales comandara de a uno. Estos últimos tendrán cualidades como por ej. velocidad, altura, vida, armas,bombas, etc.

**Requerimiento funcional 11: Desplazamiento de aviones.**

Los aviones serán manejados con las teclas (W,S) para moverse en el eje vertical y (A,D) para el eje horizontal. Para cambiar de altura, tenemos 1,2 y 3.

**Requerimiento funcional 12: Desplazamiento de artilleros.**

Los **artilleros** tendrán movimiento/disparo automático, la ubicación será aleatoria, no se podrá elegir. Podrán moverse en la zona del campo entre la base y la línea de frontera, en dirección horizontal, vertical..

No podrán superar las 6 unidades por equipo.

No podrán tener movimiento si:

- Es vista por un avión aliado o enemigo.
- Está disparando al enemigo.
- En otro caso, puede moverse o no.

Solo podrán disparar al avión activó enemigo.

**Requerimiento funcional 13: Disparo de bala**

Diseñar e implementar tanto los disparos de los aviones como el de la torre/artilleros.

Cada uno de los aviones manejados por los usuarios disparará balas que tendrán una trayectoria en dirección y sentido del movimiento con el que viene. Para disparar se utilizará la barra espaciadora.

- Los artilleros/torre disparan automáticamente.
- Los aviones disparan manualmente (presionando la barra espaciadora).
- Las municiones son ilimitadas.

**Requerimiento funcional 14.: Disparo de bomba**

Diseñar e implementar arrojar bomba de aviones

Cada uno de los aviones manejados por los usuarios podrá disparar bombas. Estas tendrán una trayectoria lineal, y perpendicular al plano de altura del avión. A su vez tendrá velocidad constante.

Las municiones son limitadas, siempre y cuando la estación de bombas de la base no haya sido destruida. Deberá retornar a la base para recargar la munición.

El usuario disparará la bomba solo si está en altura 3 y con la tecla B.

**Requerimiento funcional 15: Impacto de bala**

Diseñar e implementar tanto los impactos de los disparos de los aviones como de la torre y las artillerías.

En el caso de los aviones, un impacto de bala, provoca un 2% de daño en la vida.

En el caso de los artilleros, un impacto de bala, provoca un 10% de daño en la vida.

En el caso de los elementos de la base, un impacto de bala, provoca:

- En la torre un 3% de daño en el blindaje.
- En la estación de combustible un 1% de daño en el blindaje.
- En la estación de bombas 6% de daño en el blindaje.

**Requerimiento funcional 16: Impacto de bomba**

Diseñar e implementar el impacto de las bombas de aviones, sobre los elementos del campo enemigo.

Las bombas podrán impactar únicamente en: elementos de la base y artilleros. En cualquier caso, provoca un daño del 100% de vida.

**Requerimiento funcional 17: Ganar partida:**

Condiciones de victoria de una partida

La partida es ganada por un equipo cuando se produce alguno de los siguientes casos mencionados a continuación.

- Se destruyeron todas los aviones enemigos/aliados.
- Se destruyó todos los elementos de la base enemiga/aliada.

Se mostrará un mensaje informando que Equipo alcanzó la victoria y se mostrará la opción para volver al menú principal.

**Requerimiento funcional 18: Vista aérea del juego:**

El juego contará de una vista principal superior desde la cual se observará el avión en todo momento y otros elementos dependiendo de la posición del avión. El campo aliado siempre será visible.

**Requerimiento funcional 19: Vista lateral de los aviones:**
Vista de elementos en perspectiva lateral

La vista lateral de los aviones se mostrará al presionar la tecla para activar dicha vista y estará sincronizada con la vista superior del campo aliado.

**Requerimiento funcional 20: Destruir avión:**

Cuando esto sucede el avión desaparece del mapa y el jugador no lo puede seguir manejando.

Un avión es destruido, cuando:

- Luego de impactar balas disparadas por el enemigo (avión, torre, artillero), la vida del avión llega a cero.
- Si se queda sin combustible.
- Si es autodestruido con la tecla pertinente.

**Requerimiento funcional 21: Destruir elemento de base o artillero:**

Cuando esto sucede el elemento desaparece del mapa y el jugador no puede seguir contando con este.

Un elemento de la base o artillero es destruido cuando:

- Luego de impactar balas o bombas disparadas por el enemigo (avión).

Si se destruye la estación de combustible, no podrá recargar con más combustible a sus aviones.

Si se destruye la estación de bombas, no podrá recargar con más bombas a sus aviones.

## Requerimientos no funcionales

**Requerimiento NO funcional 1: Uso de websockets (WS).**

La comunicación entre el jugador y el servidor web, durante la partida, será establecida mediante el uso de websockets.

El Servidor WS  hará de HUB, de modo que los jugadores (Clientes WS) se envían entre sí eventos de cambios, y el servidor encamina las conversaciones entre los pares de sesiones que se van a ir formando (Jug1-Jug2). La coordinacion y sincronizacion de los movimientos de cada jugador, queda del lado del jugador (orientado a eventos).

**Requerimiento NO funcional 2: Aplicación con Arq. Lógica en 3 capas.**

Se realizará una separación en 3 capas: lógica, gráfica y persistencia, permitiendo modificar cualquiera sin afectar a las demás. La capa lógica y la de persistencia se desarrollarán en Java/MySQL mientras que la gráfica en Phaser (Javascript).

**Requerimiento NO funcional 3: La aplicación debe correr en navegadores específicos.**

El desarrollo del sistema se realizará contemplando diferentes navegadores web. Asegurándose el correcto funcionamiento en Chrome y Firefox.

**Requerimiento NO funcional 4: Debe establecer conexiones cliente-servidor.**

El juego debe poder desplegarse en un servidor local. Una vez desplegado, diferentes PC se deben poder conectar a través del navegador (clientes) y ejecutar el juego.

**Requerimiento NO funcional 5: Persistencia de datos.**

El juego debe ser capaz de almacenar los datos de las partidas guardadas y usuarios registrados en una base de datos.

**Requerimiento No funcional 6: Cantidad de jugadores.**

Dentro del juego, las partidas deben tener un máximo de dos jugadores concurrentes.

# Estudio de Factibilidad

## Propósito

Este estudio *management-oriented*, tiene como objetivo analizar la factibilidad de llevar a cabo este proyecto en base a diferentes factores como por ej. técnicos, económicos, legales y humanos que detallaremos a continuación.

Durante el estudio de factibilidad se analiza por qué y para qué se considera desarrollar el sistema.

## Factibilidad Técnica

Determina si se cuenta o no, de los conocimientos y habilidades en el manejo de métodos, procedimientos y funciones requeridas para el desarrollo e implantación del proyecto. Además indica si se dispone del equipo y herramientas para llevarlo a cabo, y de no ser así, si existe la posibilidad de generarlos o crearlos en el tiempo requerido por el proyecto..

La metodología/tecnología seleccionada por el equipo, es experimentada. Phaser es un framework de JavaScript específico para videojuegos. También es preciso mencionar que Phaser contempla los requerimientos y además, brinda un marco de documentación y foros, favorable para el desarrollo con dicha tecnología.

Para verificar la factibilidad técnica se realizaron pruebas de concepto, las cuales tuvieron resultados exitosos, determinando que el proyecto es realizable en tiempo y forma.

## Factibilidad Económica

Se refiere a que se dispone del capital en efectivo o de los créditos para el financiamiento, necesarios para invertir en el desarrollo del proyecto. Todos los recursos utilizados (Notebook, Internet, etc.) para el desarrollo del proyecto pertenecen a los integrantes del equipo. Tanto el software, las tecnologías y los repositorios de información son de uso libre y/o con licencias adquiridas, no siendo necesario gasto económico alguno.

El tiempo invertido por los integrantes del equipo, será el único costo asociado al proyecto .

## Factibilidad Legal

El desarrollo del proyecto no deberá infringir la norma o ley establecida a nivel local, municipal, estatal, etc. Es una evaluación que demuestra que el negocio puede ponerse en marcha y mantenerse, mostrando evidencias de que se ha planeado cuidadosamente.

Para el desarrollo del proyecto se utilizará software open source, de uso público y/o gratuito. Además el código será creado completamente por los integrantes del equipo. Para la representación de cualquier ítem (contenido multimedia, imágenes, gifs, entre otros) se utilizarán aquellos creados por el equipo.

No se almacenarán datos personales de los usuarios, respetando lo establecido en la ley 18.331 del 11 de agosto de 2008.

## Factibilidad Humana

Se refiere a que debe existir el personal capacitado requerido para llevar a cabo el proyecto y así mismo, deben existir usuarios finales dispuestos a emplear los productos o servicios generados por el proyecto o sistema desarrollado.

El equipo cuenta con experiencias de éxito previas en trabajos similares. En cada oportunidad cumplió con los objetivos, los entregables y en los tiempos estipulados. Los integrantes del equipo cuentan con diferentes habilidades y capacidades, las cuales se complementan o amalgaman adecuadamente.

Ya que, en años anteriores, proyectos de similares características a éste, fueran realizados con éxito por alumnos con perfiles similares a quienes conforman nuestro equipo de trabajo, se asume que es factible su realización.

## Conclusiones

Se concluye que es viable la realización del proyecto en el tiempo estipulado, con la tecnología seleccionada y con los recursos disponibles.

# Análisis de Riesgos

## Propósito

La finalidad de esta etapa es proporcionar un conjunto de principios y prácticas formales que, correctamente aplicadas, colaboren a que el proyecto sea exitoso. A través de la realización de un análisis estructurado para el manejo de diferentes amenazas que puedan afectar negativamente su desarrollo. Por otro lado, se dispone una estrategia proactiva para mitigar las amenazas halladas y continuar con el correcto proceso del proyecto.

La comunicación entre cliente y el equipo, ante  por ej. la generación de un entregable, es una forma excelente de tomar contacto con los posibles riesgos que pueden aparecer en el desarrollo.

## Identificación de riesgos

Se lleva a cabo la identificación de riesgos potenciales. Para lo cual, se realizó una evaluación y un plan de contingencia Realizando foco en los principales riesgos detectados. Para dicha evaluación se toman en cuenta dos factores: la probabilidad de que ocurra un riesgo puntual, y como segundo factor, el impacto potencial del mismo sobre el proyecto.

Para la **probabilidad** de ocurrencia del riesgo se utiliza la siguiente escala:

- Excepcional (1 - 20%)
- Improbable (21 - 40%)
- Posible (41 - 60%)
- Probable (61 - 80%)
- Muy probable (81 - 99%)

Para la escala de **impacto** del riesgo se utiliza la siguiente escala:

- Insignificante (1): no provoca ninguna alteración en lo planificado
- Muy bajo impacto (2): provoca un cambio prácticamente transparente para toda la planificación
- Tolerable (3): provoca un cambio con bajo costo y bajo perjuicio para el tiempo estipulado de otras tareas.
- Crítico (4): provoca el no cumplimiento del alcance del proyecto, o un desvío significativo del 50%.
- Catastrófico (5): provoca la cancelación del proyecto.

Para poder obtener la clasificación para cada riesgo, se multiplica el centro del rango de la probabilidad de ocurrencia por el impacto del mismo.

| Riesgo | Porcentaje de probabilidad | Indicador de impacto | Ponderacion |
|---|---|---|---|
| No cumplir con los requerimientos mínimos | 30 | 5 | 150 |
| No cumplir con calidad esperada en Producto | 40 | 3 | 120 |
| No cumplir con calidad esperada en Documentos | 40 | 3 | 120 |
| Pérdida de código/documentos por uso indebido de repositorios | 40 | 5 | 200 |
| Mala elección de la tecnología | 50 | 4 | 200 |
| Problemas de instalación en demo | 70 | 3 | 210 |
| Plazos estipulados insuficientes | 50 | 5 | 250 |
| Surgimientos de nuevos requerimientos | 10 | 4 | 40 |
| Más de la mitad de integrantes abandona el proyecto | 10 | 5 | 50 |
| Disconcordancia entre los integrantes del proyecto | 10 | 3 | 30 |
| Enfermedad de un integrante | 10 | 2 | 20 |
| Abandono de un integrante | 10 | 2 | 20 |
| Pérdida de equipos de trabajo | 10 | 2 | 20 |
| Curva de aprendizaje mal calculada | 30 | 3 | 90 |

## Evaluación de riesgos

Se evalúan los riesgos de acuerdo a su ponderación y se aparta una línea de criticidad asumiendo su riesgo. Los riesgos por debajo de 100 puntos de ponderación se consideran riesgos aceptables, mientras que los mayores a 100 se consideran riesgos críticos para los cuales se realiza un plan de contingencia y mitigación.

Es preciso mencionar que en diferentes etapas de una organización, proyecto o vida de un producto, la cantidad de riesgos puede modificarse, así como la probabilidad e impacto de cada uno. Por este motivo, se define que en cada iteración, los riesgos se deben volver a revisar y analizar los indicadores para poder tomar decisiones correctas y ejecutar las acciones necesarias.

## Contingencia de riesgos

De la evaluación de riesgos propuesta se toman como riesgos críticos los cuales tengan puntos de ponderación mayor a 100 para los cuales se plantea un plan de contingencia y mitigación.

**Riesgo crítico 1:** Pérdida de código/documentos por uso indebido de repositorios.

- Plan de mitigación:
  - El encargado de SCM deberá validar las actualizaciones al repositorio.
- Plan de contingencia:
  - Recuperar el último respaldo realizado anteriormente.

**Riesgo crítico 2:** Plazos estipulados insuficientes.

- Plan de mitigación:
  - Se realizarán revisiones del progreso del proyecto y se ajustará al cronograma del proyecto.
  - En caso de ser necesario y apoyándonos en que el equipo es multidisciplinario los integrantes que necesiten apoyo lo van a tener de sus compañeros previo a comunicarlo al gerente del proyecto.
  - Aumento de dedicación o recorte de alcance.
- Plan de contingencia:
  - Utilizar herramientas de planificación y administración.

**Riesgo crítico 3:** No cumplir con los requerimientos mínimos.

- Plan de mitigación:
  - Los integrantes pedirán días de licencia en sus correspondientes trabajos y se re plantean las tareas restantes para alcanzar los objetivos mínimos.
  - Se revisará el avance del producto verificando si cumple con los atributos de calidad, los requerimientos de calidad solicitados.
- Plan de contingencia:
  - Se realiza un estudio de los requerimientos sin cumplir y se re-planifica el cronograma del proyecto.

**Riesgo crítico 4:** Mala elección de la tecnología.

- Plan de mitigación:
  - Revisar las tecnologías utilizadas en proyectos de años anteriores que se adecue a los requerimientos de forma exitosa. Investigar nuevas.
  - Validación de la tecnología con el “referente de tecnologías” y con el tutor.

- Plan de contingencia:
    - Investigar en profundidad las tecnologías seleccionadas buscando la forma de adecuarla a los requerimientos de forma exitosa.

**Riesgo crítico 5:** Problemas de instalación en demo

- Plan de mitigación:
    - Realizar pruebas sólidas en varios equipos de despliegue de webapp consolidada. Tener otra máquina virtual para instalar webapp desde 0.
- Plan de contingencia:
    - Utilizar los elementos ya instalados como por ejemplo, abrir proyecto en eclipse y ejecutar el juego desde VisualStudio los proyecto funcionando en varios equipos.

**Riesgo crítico 6:** No cumplir con la calidad esperada - Producto.

- Plan de mitigación:
    - Asignar más recursos en pos de poder llevar a cabo cabalmente los procedimientos parte del plan de calidad definido. Verificando si se cumple con los atributos de calidad requeridos.
- Plan de contingencia:
    - Se revisan los requerimientos de calidad planteados, realizándose un seguimiento de los mismos. El encargado de documentarlo es el encargado SQA.

**Riesgo crítico 7:** No cumplir con la calidad esperada - Documentación.

- Plan de mitigación:
    - El encargado de SCM deberá utilizar guías para asegurar la calidad de los documentos, y el encargado de SQA deberá revisarlos para su validación.
- Plan de contingencia:
    - Revisar la documentación para identificar las fallas, dejando su correspondiente registro.

## Seguimiento de iteraciones:

Se evaluó en cada iteración y se identificó que no existieron cambios significativos con respecto a los indicadores de cada riesgo.

Como se mencionó anteriormente, en etapas tempranas era más probable por ejemplo, la renuncia de un integrante y a medida que el proyecto avanzaba, esta probabilidad fue decrementando. Debido a que los cambios eran insignificantes, los primeros indicadores, cálculos y planificaciones, se mantuvieron a lo largo de todo el proyecto. Más adelante en este documento, se muestra la gestión respecto a cada iteración (véase Gestión de Riesgos).

# Plan de Comunicación

## Propósito

Se elaboró un plan de comunicación donde se establece cómo, cuándo y de qué forma se va a realizar la comunicación entre los integrantes del grupo y el PM con el tutor.

## Plan

| Requisitos de la comunicación | Información a comunicar | Canales de comunicación | Actores | Frecuencia |
|---|---|---|---|---|
| Estado del Proyecto | Estado de proyecto<br>Dudas<br>Temas pendientes | Google Drive, Discord, Whatsapp, otras. | PM → Equipo | Diaria-Semanal |
| Reuniones internas de Equipo | Discutir aspectos técnicos, mostrar avances, realizar pruebas. | Google Drive, Discord, Whatsapp | Equipo → Equipo | Diaria |
| Reuniones de Tutorías | Se revisa el acta previa y el avance del grupo para con los pendientes. Se plantean dudas. Se fijan metas para la próxima. | Zoom | Equipo → Tutor | 2 veces por Semana |
| Modificación en Calendario de Reuniones | De existir alguna modificación en el día/hora de la reunión | Email | PM → Equipo | Eventual |
| Integracion | En cada marge que el SCM | Email/GitHub | SCM → Equipo | Eventual |
| Commit | En cada commit realizado por desarrollador | Email/GitHub | Desarrollador → Equipo | Eventual |
| Actas de Reunion | Se envía el acta de reunión con el tutor para el seguimiento del avance del proyecto | Email | Gerente→ Tutor, Equipo | 2 veces por Semana |
| Entregables Tecnicos | Especificación de requerimientos<br>Estudio de Factibilidad<br>Plan SQA | Google Drive<br>Email | Equipo | Diaria-Semanal |
| Otros Entregables | Avance técnico de proyecto (movimiento de aviones, disparos, etc) | Google Drive<br>GitHub | Equipo | Diaria-Semanal |

# Modelo y proceso de desarrollo

## Elección

El modelo de proceso de desarrollo elegido para llevar a cabo la implementación del proyecto es el ***Iterativo Incremental***. El cliente puede obtener resultados importantes, utilizables desde la primera iteración. Dado que cada iteración debe dar como resultado requerimientos funcionales terminados, se minimiza así el número de errores que se producen en el desarrollo y se aumenta la calidad.

El mismo permite conocer el progreso real del proyecto desde las primeras iteraciones y extrapolar si su finalización es viable en la fecha prevista. El cliente puede decidir re-priorizar los requisitos del proyecto, añadir nuevos equipos, demorarlo,cancelarlo, etc.

Facilita ver la evolución del proyecto. Cada iteración posibilita gestionar de manera natural los cambios. El análisis de los aciertos y los fallos, permite la adopción de medidas para mejorar los resultados en la siguiente.

Al terminar cada iteración se realizará un testeo exhaustivo de los requerimientos desarrollados, con el objetivo de asegurar la calidad del producto y avanzar sobre lo seguro.

Los riesgos pueden mitigarse desde el inicio, ya que el equipo debe gestionar los problemas que pueden aparecer en cada iteracion.Al hacer patentes estos riesgos, es posible iniciar su mitigación de forma temprana.

Además esta metodología permite gestionar la complejidad del proyecto, ya que en cada iteración se priorizan los requisitos, trabajando con aquellos que aportan más valor en ese momento. Contribuyendo a cumplir con el trabajo de forma ordenada y en tiempo.

Se detalla en la siguiente sección: “Iteraciones” el contenido de cada una de estas.

Para organizar las tareas, utilizamos la herramienta de SpreadSheet de Google Drive, que permite organizar las tareas a cumplir. Permitiendo al equipo tener una visual (real-time) del status del avance del proyecto como la asignación de los recursos y la clasificación de las tareas.

## Etapa de definición

Antes de definir/implementar las iteraciones, es necesario realizar principalmente algunas definiciones. Estas se alcanzan luego de que se relevan y especifican los requerimientos, se realiza el estudio de factibilidad, análisis de riesgos, asignación de roles, definición de casos de uso, diagramas de clases y de implementación.

La etapa de definición se extendió por una semana.

## Iteraciones

Se realizarán 3 iteraciones, en cada una de las cuales se implementarán los requerimientos que se detallan a continuación.

| Iteración 1 | Iteración 2 | Iteración 3: |
|---|---|---|
| 1. Movimiento Manual + Físicas de Aviones.<br>2. Movimiento Automático + Físicas de Torre.<br>3. Movimiento Automático + Físicas de Artilleros.<br>4. Vista aérea del campo.<br>5. Disparo de bala/bomba.<br>6. Impacto de bala/bomba. | 1. Tableros de info.<br>2. Ingreso al juego.<br>3. Menu Principal.<br>4. Creación de partida.<br>5. Unirse a partida.<br>6. Pausar partida.<br>7. Ganar partida. | 1. Vista lateral del campo.<br>2. Abandonar partida.<br>3. Guardar partida.<br>4. Cargar partida.<br>5. Menú opciones (en partida). |

## Fase final

En esta etapa se realiza la revisión global de la documentación alcanzada y se conforma el entregable (código y documentación) para ser enviado por el medio correspondiente (mail).

# Planificación

## Proposito

La finalidad de esta etapa es la de realizar una planificación inicial del desarrollo de software en cuestión, ajustándose a un cronograma tentativo. Esto comúnmente representa de manera gráfica o virtual la forma en la que se desenvolverá el proceso de dicho software. Teniéndose en cuenta los riesgos o eventualidades que pueden surgir, para luego ir refinando dicha planificación con el verdadero desarrollo del mismo.

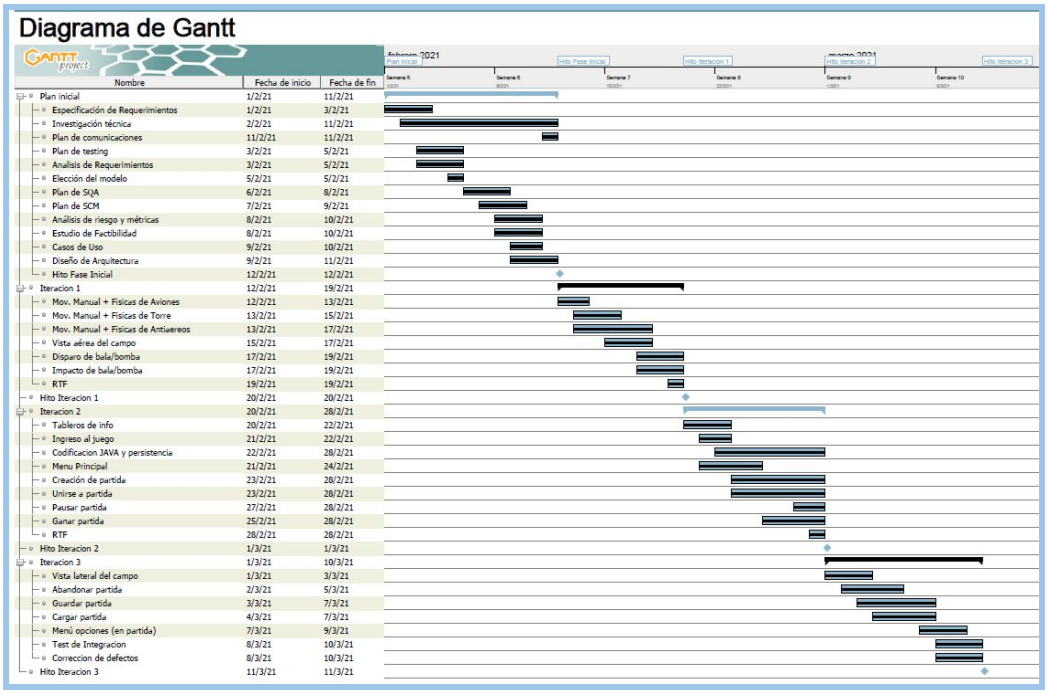
En esta instancia también se define un cronograma de rotación de roles, para los integrantes del proyecto con el objetivo de amalgamar los avances individuales en grupales, y vice versa.

## Diagrama de Gannt

Ver “06 - Diagrama Gannt Incial.pdf” en carpeta virtual entregable.

Ver “07 - Diagrama Gannt Final.pdf” en carpeta virtual entregable.

Las diferencias que puedan surgir entre el Gant inicial y el final, son producto de ocurrencias de riesgos críticos y/o actividades de refactoring que pudieran haber surgido. La gestión de esos cambios se encuentra en el plan de Gestión de Riesgos. Para una mejor visualización, acceder a los links superiores o en la carpeta de diagramas.




# Tecnologías utilizadas y Arquitecturas:

Se describen a continuación las tecnologías utilizadas para llevar a cabo el proyecto.

| Tecnologías | Descripción |
|---|---|
| Eclipse 2018-12 | IDE utilizado para el desarrollo del sistema |

| | |
|---|---|
| Versionado= GitHUB | Utilizamos GitHUB para el respaldo y versionado del código. |
| Phaser 3 (JavaScript) | Framework utilizado para el desarrollo del juego |
| JavaScript | Lenguaje de programación. |
| WEBSOCKET | Técnica de desarrollo web para crear aplicaciones |
| JSON | JavaScript Object Notation, formato de texto ligero para el intercambio de datos |
| HTML 5 | Lenguaje utilizado para realizar páginas web |
| Tomcat Apache | Contenedor de servlets que permite hostear el servidor web. |
| MYSQL | Sistema de administración de Base de datos relacionales. |
| CSS3 | Lenguaje que se utiliza para dar diseño a páginas web |

| ***Tools*** | ***Descripción*** |
|---|---|
| Gant Proyect | Herramientas de Gestión de proyectos. |
| draw.io | Herramienta para crear diagramas de clase, clases de uso, etc. |
| GitHUB | Como repositorio de código, respaldo y versionado. |
| Gimp | Editor de imagenes. |
| Google Docs | Documentos de Texto, planillado, presentación. |

## Patrones de diseño

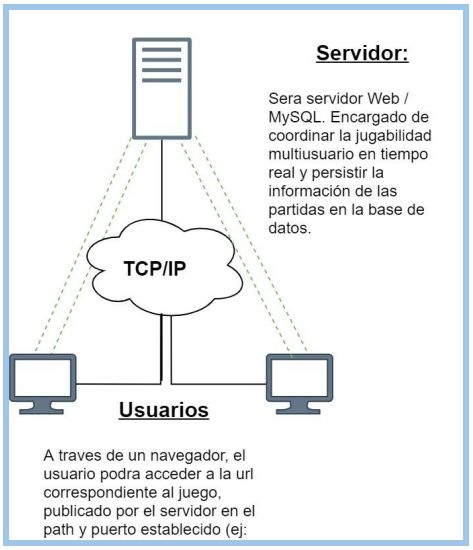
Se utilizaron los siguiente Patrones de diseño para lograr una correcta separación en tres capas:

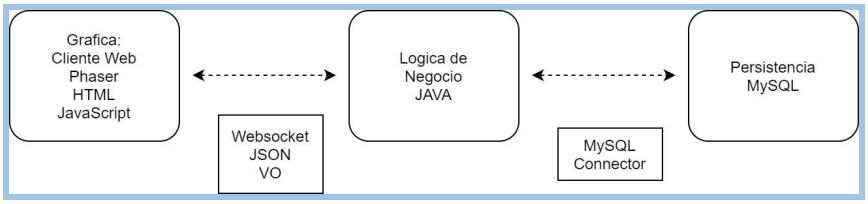
- FACADE: Sirve como único punto de acceso a la Lógica del sistema.
- VALUE OBJECT (VO): Resuelve la transferencia de información entre la capa gráfica y la lógica.
- DATA ACCESS OBJECT (DAO): Establece un límite entre la capa lógica y la persistencia. Logra desacoplar la lógica del medio de persistencia utilizado.
- SINGLETON: es un patrón de diseño que permite restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Utilizado en la interfaz de la fachada a publicar.

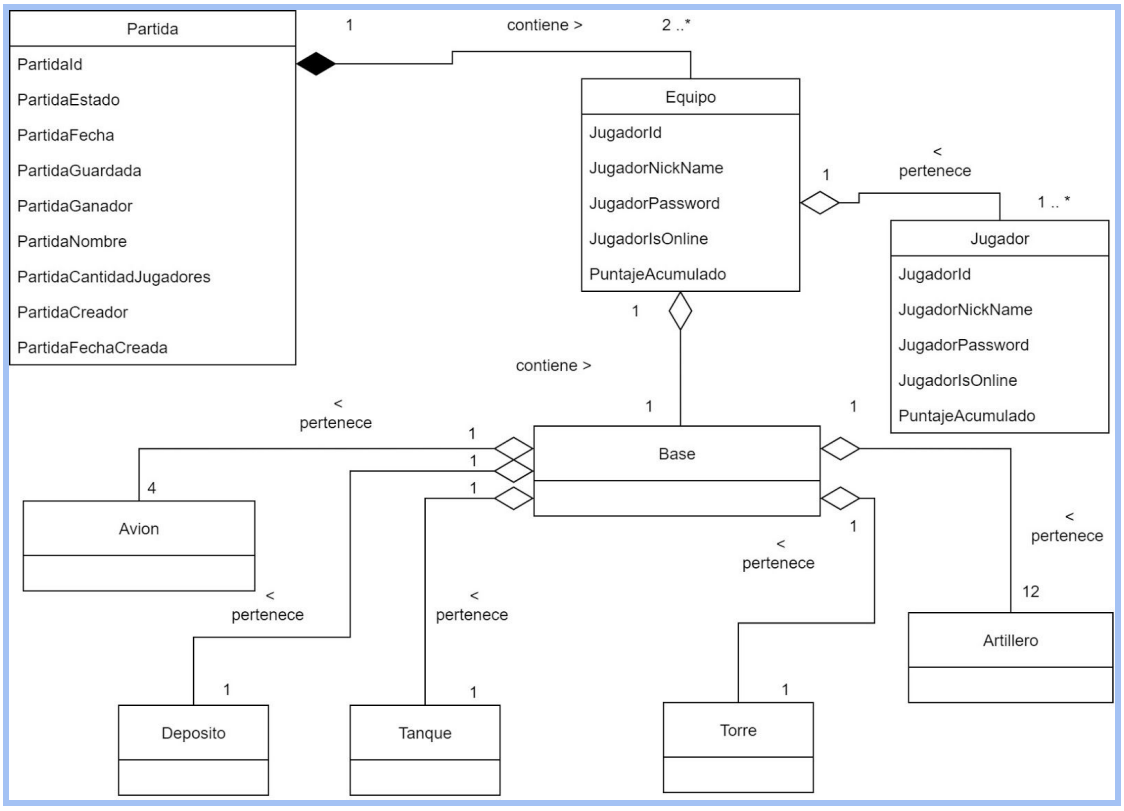
Estos patrones permiten la implementación en 3 capas. Estos brindan **mantenibilidad,** permitiendo realizar modificaciones una vez liberado. Y **portabilidad** al poder ejecutarse en distintas plataformas o ambientes (hardware, software, contexto) a bajo costo.

# Diagramas de Arquitecturas y otros

Se muestran a continuación de forma ilustrativa los diagramas de Arquitectura Lógica, Física y Conceptual (en este orden).




**Lista de diagramas adjuntos en la carpeta virtual como parte de la entrega final del Proyecto.**

- **01 - Diagrama Arquitectura Física.pdf**
- **02 - Diagrama Arquitectura Logica.pdf**
- **03 - Diagrama UML Conceptual.pdf**
- **04 - Diagrama UML Implementación.pdf**
- **05 - Diagrama MER.pdf**
- **08 - Diagramas-Casos de Uso.pdf**

Los diagramas adjuntos, podrán visualizarse tanto en drawIO, de forma online; como en la carpeta de “Diagramas” en versión PDF vectorial para su óptima visualización.

# Plan de SQA

## Propósito

El objetivo de esta sección es definir el Plan de aseguramiento de la calidad del proyecto **“Saviors of the Somme”**.
Se detalla la organización, tareas y responsabilidades del Equipo SQA, que permitirán validar la calidad del software en base a requerimientos, estándares y procesos establecidos. Se utilizan guías, herramientas, técnicas y metodologías para llevar a cabo las actividades y reportes.
Los estándares que se utilizan en este Plan de Aseguramiento de la Calidad.

- IEEE-Std-730-1998, IEEE Standard for Software Quality Assurance Plans, June 1998.
- IEEE-Std-730.1-1995, IEEE Guide for Software Quality Assurance Planning, December 1995.

## Documentos de referencia

Se hace referencia a los siguientes documentos:

- Revisión técnica.
- Documento de factibilidad.
- Análisis de riesgos.
- Especificación de requerimientos.
- Plan de gestión.
- Modelo de desarrollo.
- Plan de pruebas.
- Estándar de codificación.
- Plan de SCM

## Gestión y organización

Se detallan las tareas, responsabilidades y organización que se llevarán a cabo para asegurar la calidad del software. Se describen los roles que tendrá cada integrante del proyecto y las responsabilidades para cada uno de ellos.
El responsable de SCM es quien se encargará de controlar y unificar los cambios que se realizan sobre las versiones de código. Es el encargado de informar a los desarrolladores del equipo del estado de las versiones y sobre cual trabajar en cada caso.
Para la toma de decisiones sobre cualquier cambio pueda afectar el proyecto se realizará una reunión (presencial o virtual) entre todo el equipo, para la toma de decisión conjunta.

### Roles y Responsabilidades

En la siguiente tabla se muestra la distribución de roles y responsabilidades:

Primera etapa: comprendida en las fechas 02/02/2021 a 24/02/2021.

Segunda etapa: comprendida en las fechas 25/02/2021 a 16/03/2021.

| | ***Nombre*** | | |
|---|---|---|---|
| ***Rol*** | ***Primera Etapa*** | ***Segunda Etapa*** | ***Documentacion*** |
| Gerente de proyecto | Sebastian Serrentino | --- | Doc. Factibilidad; Análisis de Riesgos; Plan de gestión; Modelo de desarrollo |
| Arquitecto de software | Joaquin Segovia | Ramiro Silvia | Doc. Factibilidad; Modelo de desarrollo |
| Responsable de SQA | Sebastian Garciarena | Fernando Bertoldi | Doc. Factibilidad; Análisis de Riesgos; Plan de SQA; Estandar de codificacion; Plan de Pruebas |
| Responsable de SCM | Ramiro Silvia | Joaquin Segovia | Plan de SCM |
| Analista funcional | Fernando Bertoldi | Ramiro Silvia | Esp. Requerimientos; Análisis de Riesgos; Modelo de desarrollo |
| DBA e Infra | Sebastian Serrentino | Fernando Bertoldi | Plan de Instalación |
| Diseño Interactivo y AudioVisual | Joaquin Segovia | Sebastian Serrentino | WireFrames; MockUps |
| Desarrollador/es | Todo el equipo | | Micro Documentación de funcionalidades |
| Tester/s | Todo el equipo | | Caso de prueba |
| Documentador | Todo el equipo | | - |

## Documentacion:

Se detalla la documentación requerida para asegurar la calidad del proyecto:

- Requerimientos de software
  - Véase Especificación de Requerimientos.
- Tecnologías utilizadas y Arquitectura.
- Plan de SCM
- Plan de pruebas
  - Véase Anexo - Plan de Pruebas.
- Manuales
  - Véase Anexo - Manuales de Usuario.
  - Véase Anexo - Manuales de Instalación.
- Estándares
  - Véase Anexo - Estándares

Estos se desarrollan en el presente documento, en las secciones del mismo nombre.

# Gestión de Riesgos

En la etapa de análisis del proyecto se identificaron los riesgos potenciales, se realizó una evaluación y un plan de contingencia para los principales riesgos detectados.

Véase Análisis de Riesgos.

Durante el transcurso del proyecto se va realizando seguimiento de los riesgos identificados y de ocurrir, se les aplicará el plan de contingencia correspondiente. Los riesgos críticos identificados en la etapa de análisis fueron los siguientes:

- RC1: Pérdida de código/documentos por uso indebido de repositorios.
- RC2: Plazos estipulados insuficientes.
- RC3: No cumplir con los requerimientos mínimos.
- RC4: Mala elección de la tecnología.
- RC5: Problemas de instalación en demo
- RC6: No cumplir con la calidad esperada - Producto.
- RC7: No cumplir con la calidad esperada - Documentación.

***Seguimiento en Iteración 1:***

Durante la iteración se identificó que estaba ocurriendo el RC4. Se aplica su correspondiente plan de contingencia y mitigación. Se busca información acerca de otra tecnología a utilizar. El equipo acude a las tutorías tecnológicas disponibles para este proyecto con el Ing. Pablo Masás. Se reemplaza la tecnología en cuestión por una nueva y se continúa con el proyecto.

Este riesgo no provocó cambio en la planificación. En las tutorías se detecta el no cumplimiento con la calidad esperada en los documentos (Faltantes de RTF y planillado de testing), ocurriendo el RC7. Se aplicó rápidamente el plan de mitigación. Se realizan RTF de documentación y se realizan las fichas de testing faltantes.

En este caso tampoco es necesario realizar re-planificación.

***Seguimiento en Iteración 2:***

Durante la iteración no se detectó que se concretara ningún riesgo crítico planteado.

***Seguimiento en Iteración 3:***

Durante la iteración se identificó que estaba ocurriendo el RC2. Se aplica su correspondiente plan de contingencia. Se decide re-planificar las tareas de la iteración, modificando el correspondiente diagrama de Gantt. En este caso, se habían planificado las tareas dejando un "colchón" de una semana ante la fecha de entrega del proyecto, por lo que fue sencilla su re-planificación, logrando que el producto final se entregará en la fecha estipulada.

# Plan de SCM

Este plan fue realizado siguiendo la norma de calidad IEEE std. 828-2012 (Estandar para planes de administración de la configuración de Software), llevando a cabo la adaptación pertinente en base los puntos de relevancia para nuestro proyecto.

## Propósito

En esta sección, se describen las actividades de gestión de la configuración de software que serán llevadas a cabo durante el desarrollo del proyecto. Se incluyen, la definición de líneas base, el control de los ítems de configuración, los mecanismos de liberación de versiones y actualización de ambientes.

## Alcance

El presente plan de gestión de la configuración de Software, abarca la identificación de elementos de configuración a gestionar, las líneas base, la gestión de cambios, la actualización de ambientes de desarrollo y testeo interno para las fases del proyecto.
Resolviendo la actualización simultánea, el código compartido o conjunto y el manejo de versiones, contenidas en el desarrollo del videojuego.

## Gestion SCM

En esta sección se describen las responsabilidades y responsables para la realización de las actividades de gestión de configuración dentro del proyecto.

## Organización

Este punto se define según el documento de organización de equipo de proyecto.
Se tiene en cuenta el rol de Responsable de SCM por parte del grupo de desarrolladores donde se definen las responsabilidades y su dependencia del gerente de proyecto.

## Responsabilidades

El Responsable de SCM debe proveer el entorno de configuración para el proyecto. Debe preocuparse porque todos los integrantes del equipo entiendan y puedan ejecutar las actividades de SCM que el Plan les asigna, así como asegurar que éstas sean llevadas a cabo. Seguir la línea base, controlando las versiones y cambios de ella, son tareas correspondientes a este rol del proyecto.
El encargado del SCM serán dos integrantes: F. Bertoldi y R. Silva (gerente y encargado respectivamente) y como comité a todo el equipo.
Para la toma de decisiones sobre cambios en los requerimientos, arquitectura, modo de implementación, herramienta a utilizar o cualquier cambio que pueda afectar el proyecto se realizará una reunión (presencial o virtual) entre todo el equipo donde se discutirá sobre el tema y se tomará una decisión conjunta entre todos los integrantes del equipo. Revisar cada petición de cambio para su aprobación, rechazo o postergación. Establecer las condiciones de realización del cambio.

## Actividades

Identifica todas las actividades y tareas que se requieren para el manejo de la configuración del sistema. Estas deben ser tanto actividades técnicas como de gestión de SCM, así como las actividades generales del proyecto que tengan implicancia sobre el manejo de configuración.

## Administración de cambios

Los desarrolladores trabajan sobre sus versiones locales e informan al resto del equipo cuando se realiza un commit al sistema de gestión de versiones (GitHub).
Luego el responsable de SCM realiza los controles que entiende necesarios produciendo un "merge" de las versiones. Manteniendo siempre una versión sin errores con la mayor actualización posible.
Para el manejo y control de cambios de código se crea un repositorio mediante la herramienta GitHub, a la cual tendrán acceso todos los desarrolladores.

En la misma se tienen las siguientes branches:

- **Dev Branch**: Donde trabajan los desarrolladores.
- **Máster**: Donde se mantiene la última versión unificada sin errores.

Para el caso de los documentos se utiliza una versión compartida de Drive donde se mantienen los documentos actualizados.
Se crea un directorio compartido en drive, PROYECTO Entregable (nombres adentro), donde se agregan:

- Proyecto - Documento : Para la "Carpeta General" del proyecto.
- Proyecto - Anexos: Donde se adjuntan los documentos que no van en la "Carpeta General".

Se crea además, una carpeta de:

- Actas: Donde se guardan las actas de las distintas reuniones con las tutorías.
- Diagramas: Donde se adjuntan todos los diagramas, tablas, imágenes ilustrativas, referentes a la solución final.
- Instaladores: Contenedor de fuentes y recursos para deploy.

Paralelamente al trabajo colaborativo en la nube, cada integrante del equipo, maneja su versión local, con el fin de mantener respaldos actualizados, como contingencia antes problemas con la plataforma utilizada.

# Metricas:

Durante el proyecto se irán tomando diferentes métricas. Las seleccionadas fueron: esfuerzo, de diseño y de mantenimiento. Las siguientes mediciones se harán y se utilizarán para determinar el costo y el calendario de la situación de las actividades a lo largo del proyecto:

## De Diseño:

Como métrica de diseño se utilizara la *CCM: (Complejidad Ciclomática Media),* la cual es un indicador útil que nos permite predecir la dificultad de prueba y el mantenimiento de un programa. Los siguientes valores fueron obtenidos mediante la herramienta JHAWK [*Valores mayores a 10 sugieren una complejidad importante*]

| Iteracion | CCM |
|---|---|
| 1 | 1,90 |
| 2 | 2,41 |
| 3 | 2,60 |

Como se puede apreciar hay un crecimiento considerable de CCM durante la iteración 1 y la iteración 2.

Esto se debe a que durante las dos primeras iteraciones se realizó el trabajo de "logica mas dura", luego en la iteración 3 se nota una estabilización del índice.

Si bien durante las 3 iteraciones hubo un crecimiento del CCM , el mismo se considera despreciable ya que no sobrepasan el valor de 10.

# Métrica de Mantenimiento:

Se utilizará el índice de madurez de SW recomendado por la IEEE, dado que señala la estabilidad del software en base a la cantidad de cambios.

- MH = Módulos hoy, en la versión actual.
- MC = Módulos cambiados de la anterior.
- MA = Módulos agregados desde la anterior.
- ME = Módulos eliminados de la anterior.
- IMS = (MH – MC – MA – ME) / MH.

Si el índice tiende a 1, se considera que se está estabilizando el sistema.

| **Iteracion 1** | |
|---|---|
| **Variable** | **Valor** |
| MH | 8 |
| MC | 0 |
| MA | 0 |
| ME | 0 |
| ***X*** | ***Valor*** |
| ***IMS*** | ***1*** |

| **Iteracion 2** | |
|---|---|
| **Variable** | **Valor** |
| MH | 8 |
| MC | 2 |
| MA | 26 |
| ME | 0 |
| ***X*** | ***Valor*** |
| ***IMS*** | ***2*** |

| **Iteracion 3** | |
|---|---|
| **Variable** | **Valor** |
| MH | 34 |
| MC | 1 |
| MA | 4 |
| ME | 0 |
| ***X*** | ***Valor*** |
| ***IMS*** | ***0,85*** |

Se puede apreciar que en la iteración 2 respecto a la 1 existe un crecimiento del IMS, esto se debe a que en la iteración 2, se terminó de consolidar la “lógica dura”, mientras que en la iteración 3 se nota una estabilización ya que en dicha iteración se trabajó más bien en extras que no requirieron grandes implementaciones.

## Métricas de De Proceso:

Como insumo se cuenta con las horas dedicadas por cada uno de los integrantes del equipo, quienes realizan un registro diario de sus esfuerzos realizados, en cada iteración. Para estimar y medir estas horas, se dispuso una planilla en la cual cada integrante cargaba en cada iteración del proyecto las horas que dedicaba a cada actividad. Las actividades que identificamos fueron: Investigación, Análisis | Programación | Testing | Documentación | Gestión.

Se divide el proyecto en 4 fases: Plan inicial, iteración 1, iteración 2 e iteración 3.

Se planifica mediante diagrama de Gantt las diferentes tareas y tiempo que se dedicaran a cada fase o iteración.

Se toma un estimado de 980 horas de trabajo para el proyecto. Durante el desarrollo del mismo, se fue realizando el seguimiento de las horas de esfuerzo en cada iteración.

Luego de finalizada la iteración 3, se realiza la sumatoria de todas las horas trabajadas por actividad durante el proyecto.

El resultado fue el siguiente: Se trabajaron 1004 horas, lo cual hace una diferencia de 24 horas respecto a las horas estimadas. Se concluye que 24 horas en un proyecto de 6 semanas hace un desajuste menor a 4 horas por semana, en un equipo de 5 personas deriva en menos de una hora semanal de atraso por integrante, lo cual lo se considera un desajuste altamente aceptable. El porcentaje total de desajuste fue de 2,5 % respecto a las horas estimadas

Para ver las tablas por iteración, véase ANEXO - Métricas.

# Plan de Testing

## Proposito

Definir un proceso de testing que permita identificar fallas y verificar que se cumplan los requerimientos para asegurar la calidad final del producto.

## Descripción

Para la realización de esta planificación nos basamos en la norma IEEE 1008-1987 Standard for Software Unit Testing. Propósito. Esta Tiene como propósito definir el proceso de pruebas que permita identificar fallas y verificar que se cumplan los requerimientos para el aseguramiento de la calidad del producto.

## Composicion

Se realizarán pruebas de componente, una vez finalizada la prueba del componente específico se integrará con otro/s componentes cuya prueba haya finalizado.
Luego de finalizadas las pruebas e integración de componentes del subsistema (siendo un subsistema el backend y otro el frontend).
Se realizarán pruebas de integración de los subsistemas verificando la correcta comunicación de los mismos. Cumplido con lo anterior, en las pruebas de sistemas se procederá a probar al sistema como un todo.
En caso de falla de prueba y posterior corrección del “bug” se procederá con las pruebas de regresión, las cuales van a consistir en una vez corregido el error, probar el caso de prueba concreto y reprobar los casos previamente ejecutados, realizando test de humo sobre los requerimientos de la integración anterior para validar que no se haya roto nada.
Lo anterior aplica a cada iteración.

## Trazabilidad

Para cada prueba de SISTEMA realizada se deberá generar un formulario con formato de tabla(de aca en mas llamado caso de prueba que contenga la siguiente información: Número de prueba, fecha, téster, fase o iteración del proyecto, número de requerimiento, el resultado esperado, el resultado obtenido, las fallas que se detectaron y las acciones que se tomaron para la corrección de la falla.
Se considerará que un caso de prueba falla, si el resultado esperado difiere del obtenido en la ejecución.
El caso de prueba será la constancia de la realización de las pruebas realizadas y por consiguiente formará parte del entregable.

## Criterio de éxito de pruebas

Se considera exitoso el ciclo de pruebas si se cumplen dos condiciones:

1. Que las mismas contemplen al menos una prueba sobre cada requerimiento de la iteración.
2. Que las pruebas tengan un porcentaje de éxito del 70%.

En caso contrario, una vez que las mismas sean exitosas (luego de la corrección) se deben agregar nuevos casos de prueba intentando agregar casuísticas similares a las que fallaron inicialmente.

Véase Anexo (Plan de prueba) para mayor información.

# Conclusiones

Como conclusión se puede observar que a través de una Planificación de mejora continua, se garantiza la calidad de un producto de software. Entre las funciones básicas de la gestión del proyecto, nos resultó interesante la planificación de las tareas del mismo, su coordinación, asignación de tareas/roles, su organización/comunicación y la planificación de los esfuerzos. A su vez, queremos destacar la experiencia sobre tareas que involucran el monitoreo de los avances, control de cambios, planes de prueba, cálculo de métricas, gestión de riesgo y definición/coordinación de planes de trabajo.

Se podría decir que la mejora continua se consigue con la incorporación de ciertas pautas o tareas a nuestro proceso de desarrollo. Estas nos permitieron identificar ciertos puntos a mejorar, posteriormente realizando tareas correctivas en tiempo y forma, haciendo posible el replanteo y la definición de un proceso para prevenirlo en el futuro. Fue necesario que el proceso de mejora continua se aplique de manera cíclica, lo que nos permitió aprender y/o mejorar algo nuevo en cada ciclo, a través de la toma de decisión.

Creemos que fue una instancia muy buena para formarnos como profesionales habiendo desarrollado tanto competencias técnicas como habilidades blandas (llamadas “soft skills”).
Creemos que esta fue una instancia enriquecedora para los alumnos ya que se ponen en práctica todos los aspectos académicos aprendidos durante la carrera hasta el momento. Por otro lado, se aprenden nuevas tecnologías de desarrollo y se profundiza en nosotros la capacidad de investigar y resolución de problemas en tiempo de desarrollo.

# Trabajo a Futuro

Se detallan algunas funcionalidades futuras pensadas, en parte gracias al diseño y construcción del software.
Posibilidad de parametrizar tanto la cantidad de aviones como la cantidad de artilleros. Para esto se diseñaron objetos genéricos y herencias.

Posibilidad de agregar varios integrantes por cada equipo, donde podrían o no tener varias bases por cada equipo.
Posibilidad de agregar diferentes interacciones entre el menú y los usuarios, como por ejemplo, lista de partidas abiertas con salas para buscar jugador en la red
Indicadores futuros para poder generar puntajes, ranking y estadísticas de los jugadores en general y del uso del juego globalmente. Esto permitiría a futuro incluso generar un algoritmo de “matchmaking” para unir a jugadores del “mismo” nivel de competencia en el juego.

Creemos que se podrían agregar, muchas más animaciones y efectos que potenciarán el producto exponencialmente. Por ejemplo, elementos “soldaditos” que se distribuyan por el campo e interactúen con algún tipo de inteligencia.
Permitir jugadores anónimos sin necesidad de login.
Generar vista de moderador para futuras competencias, en las cuales, dicho moderador, podría modificar parámetros de la partida y visualizar en concreto ciertos elementos en otras perspectivas.
Se podría agregar algún componente en 3D con alguna tecnología que lo permita como UNITY.
Debido a las tecnologías utilizadas, se podría generar una versión para móviles a bajo costo.


UDE
UNIVERSIDAD DE LA EMPRESA