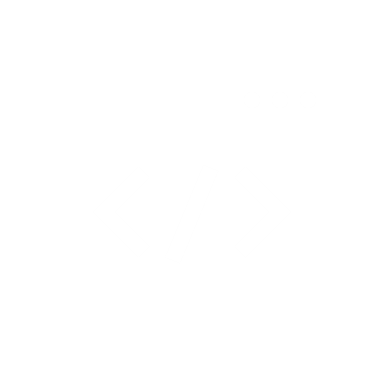
****

**Proyecto de Desarrollo**

**Directores**

Jordán Pascual Espada y Daniel Fernández Álvarez

**Director**

Jordán Pascual Espada y Daniel Fernández Álvarez

**Trabajo de Fin de Grado**

**Julio 2019**

*Videojuego Basado en Sustainable Sea*

**Autor**

David Ferreiro Fernández

**Autor**

José Manuel Redondo López

**Grado en Ingeniería Informática del Software**

Grado en Ingeniería Informática del Software

**Versión 1.76 (01/07/2019)**

**Modelo Métrica 3**

Copyright (C) 2019 **JOSÉ MANUEL REDONDO LÓPEZ**. [1]

*Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.*

*A copy of the license is included in the section entitled "GNU Free Documentation License".*

Agradecimientos

Esta sección está dedicada a aquellos que han hecho posible este proyecto:

* Los chicos de la facultad de Biología, por permitirme digitalizar su juego.
* Mis tutores, por aceptar el proyecto y ayudarme durante todas las etapas.
* Los sujetos de pruebas que me han dado su feedback durante las pruebas de usuario. Muchas gracias a Lucía, Aitor, Victor, Cristian, David, Daniel y Tatiana. Así como de nuevo a los chicos de Biología, por su feedback.
* Mis padres, por correr con los gastos indirectos del proyecto, tales como el recibo de la luz de casa.
* A José Manuel Redondo López, por la creación de la plantilla usada para esta documentación.

**Muchas gracias a todos.**

Índice de contenido

[Capítulo 1 Introducción 15](#_Toc45038968)

[INT 1: Contexto del Proyecto 16](#_Toc45038969)

[INT 2: Motivación del Sistema 16](#_Toc45038970)

[PSI 2.1: Especificación del Ámbito y Alcance 16](#_Toc45038971)

[PSI 2.3: Definición del Plan de Trabajo 19](#_Toc45038972)

[INT 3: Descripción del Juego y Reglas 20](#_Toc45038973)

[INT 4: Objetivo del Proyecto 20](#_Toc45038974)

[INT 5: Metodología de Trabajo 20](#_Toc45038975)

[Capítulo 2 Estudio de Viabilidad del Sistema 21](#_Toc45038976)

[EVS: Estudio y Valoración de Alternativas de Solución y Selección de Alternativa Final 22](#_Toc45038977)

[Unity 22](#_Toc45038978)

[Unreal Engine 24](#_Toc45038979)

[Cocos2d-JS 25](#_Toc45038980)

[Conclusión 26](#_Toc45038981)

[Capítulo 3 Análisis del Sistema de Información 29](#_Toc45038982)

[ASI 1: Definición del Sistema 30](#_Toc45038983)

[Determinación del Alcance del Sistema 30](#_Toc45038984)

[ASI 2: Establecimiento de Requisitos 31](#_Toc45038985)

[Obtención de los Requisitos del Sistema 31](#_Toc45038986)

[Identificación de Actores del Sistema 32](#_Toc45038987)

[Especificación de Casos de Uso 33](#_Toc45038988)

[ASI 3: Identificación de Subsistemas de Análisis 36](#_Toc45038989)

[Descripción de los Subsistemas 36](#_Toc45038990)

[Descripción de los Interfaces entre Subsistemas 36](#_Toc45038991)

[ASI 4: Análisis de los Casos de Uso 37](#_Toc45038992)

[Caso de Uso 1 39](#_Toc45038993)

[Caso de Uso 2 41](#_Toc45038994)

[ASI 5: Análisis de Clases 45](#_Toc45038995)

[Diagrama de Clases 45](#_Toc45038996)

[Descripción de las Clases 45](#_Toc45038997)

[ASI 8: Definición de Interfaces de Usuario 47](#_Toc45038998)

[Descripción de la Interfaz 47](#_Toc45038999)

[Definición del aspecto de la interfaz 48](#_Toc45039000)

[Descripción del Comportamiento de la Interfaz 48](#_Toc45039001)

[Diagrama de Navegabilidad 49](#_Toc45039002)

[ASI 10: Especificación del Plan de Pruebas 50](#_Toc45039003)

[Capítulo 4 Diseño del Sistema de Información 53](#_Toc45039004)

[DSI 3: Diseño de Casos de Uso Reales 54](#_Toc45039005)

[Caso de Uso 1.1 55](#_Toc45039006)

[Caso de Uso 1.2 55](#_Toc45039007)

[DSI 4: Diseño de Clases 57](#_Toc45039008)

[Diagrama de Clases 57](#_Toc45039009)

[DSI 5: Diseño de la Arquitectura de Módulos del Sistema 58](#_Toc45039010)

[Diagramas de Paquetes 58](#_Toc45039011)

[Diagramas de Componentes 58](#_Toc45039012)

[DSI 10: Especificación Técnica del Plan de Pruebas 59](#_Toc45039013)

[Pruebas Unitarias 59](#_Toc45039014)

[Pruebas de Aceptación 60](#_Toc45039015)

[Pruebas de Usabilidad y Accesibilidad 61](#_Toc45039016)

[Pruebas de Rendimiento 62](#_Toc45039017)

[Capítulo 5 Construcción del Sistema de Información 63](#_Toc45039018)

[CSI 1: Preparación del Entorno de Generación y Construcción 64](#_Toc45039019)

[Lenguajes de programación 64](#_Toc45039020)

[Herramientas y programas usados para el desarrollo 64](#_Toc45039021)

[CSI 2: Generación del Código de los Componentes y Procedimientos 66](#_Toc45039022)

[CSI 3: Ejecución de las Pruebas Unitarias 67](#_Toc45039023)

[CSI 4: Ejecución de las Pruebas de Aceptación 69](#_Toc45039024)

[CSI 5: Ejecución de las Pruebas de Usuario 70](#_Toc45039025)

[CSI 6: Ejecución y Conclusiones de las Pruebas de Rendimiento 72](#_Toc45039026)

[Especificaciones del sistema de pruebas 72](#_Toc45039027)

[Resultados 72](#_Toc45039028)

[Conclusiones 73](#_Toc45039029)

[CSI 7: Elaboración de los Manuales de Usuario 73](#_Toc45039030)

[Manual de Usuario 74](#_Toc45039031)

[Manual del Programador 74](#_Toc45039032)

[Capítulo 6 Apéndices 75](#_Toc45039033)

[Presupuesto 76](#_Toc45039034)

[Problemas Encontrados y Decisiones tomadas Durante el Desarrollo 77](#_Toc45039035)

[Generación dinámica del tablero 77](#_Toc45039036)

[Tamaño variable del mapa 77](#_Toc45039037)

[Dados con físicas vs generador de números aleatorios 77](#_Toc45039038)

[Mostrar/Ocultar vs Crear/Destruir el dado 78](#_Toc45039039)

[Cartas que afectan a casillas adyacentes 78](#_Toc45039040)

[Animación de los barcos 78](#_Toc45039041)

[I18N 78](#_Toc45039042)

[Conclusiones 80](#_Toc45039043)

[Ampliaciones 81](#_Toc45039044)

[Requisitos para futuras iteraciones 81](#_Toc45039045)

[Referencias Bibliográficas 82](#_Toc45039046)

[Contenido entregado en los anexos 85](#_Toc45039047)

[Contenidos 85](#_Toc45039048)

[GNU Free Documentation License 88](#_Toc45039049)

[ADDENDUM: How to use this License for your documents 95](#_Toc45039050)

Índice de Figuras

[Ejemplo de caso de uso 1 33](#_Toc45038955)

[Ejemplo de caso de uso 2 34](#_Toc45038956)

[Descripción de las actividades de un escenario con un diagrama de robustez (I) 40](#_Toc45038957)

[Descripción de las actividades de un escenario con un diagrama de robustez (II) 41](#_Toc45038958)

[Diagrama de clases de ejemplo 45](#_Toc45038959)

[Boceto de una interfaz 48](#_Toc45038960)

Índice de Tablas

[Catalogo definitivo de requisitos 31](#_Toc45038961)

[Primera ejecución de las pruebas 67](#_Toc45038962)

[Excepción generada en el test BoatInitialization 67](#_Toc45038963)

[Ejecución de las pruebas tras la corrección 68](#_Toc45038964)

[Checklist de pruebas de Usabilidad 71](#_Toc45038965)

[Estructura general del fichero anexo entregado 86](#_Toc45038966)

[Estructura de la carpeta “desarrollo” del fichero anexo entregado 87](#_Toc45038967)

# Introducción

Fase de Planificación

PSI

## INT 1: Contexto del Proyecto

Este proyecto nace a partir de un proyecto de innovación de la facultad de Biología, cuyo objetivo es aumentar el conocimiento acerca de la sostenibilidad marina del público general mediante gamificación, es decir, aplicar mecánicas de juegos o, en este caso, crear un juego completo a actividades de otro tipo para aumentar el interés, en este caso la biología marina y la importancia de la pesca sostenible y el cuidado del ecosistema marino.

El proyecto consiste en realizar una serie de preguntas con varias respuestas acerca de los temas mencionados en el párrafo anterior a un grupo de sujetos, tras ello, jugar una partida al juego de tablero y, finalmente, tras acabar la partida, contestar a las mismas preguntas y comprobar si los sujetos acertaban más preguntas que previo haber jugado el juego, es decir, si los sujetos **han aprendido algo jugando al juego.** Puede encontrar el cuestionario utilizado como anexo del proyecto PLACEHOLDER.

El estudio demostró que existía una mejora en las respuestas tras haber jugado, por no mencionar que los jugadores disfrutaban de la experiencia del juego, sobre todo compitiendo los unos con los otros.

El juego se probó en sujetos de distintos rangos de edad: adolescentes, universitarios, adultos, etc. Los niños demostraban mayor entusiasmo jugando al juego, pero no aprendían tanto como otros sujetos, en general el rango de edad que más éxito demostró en el experimento fue de **18-22 años** aproximadamente. Estos sujetos disfrutaban compitiendo con sus compañeros en el juego de mesa y tenían capacidad mental más que suficiente para aprender algo, no solamente disfrutar con el juego.

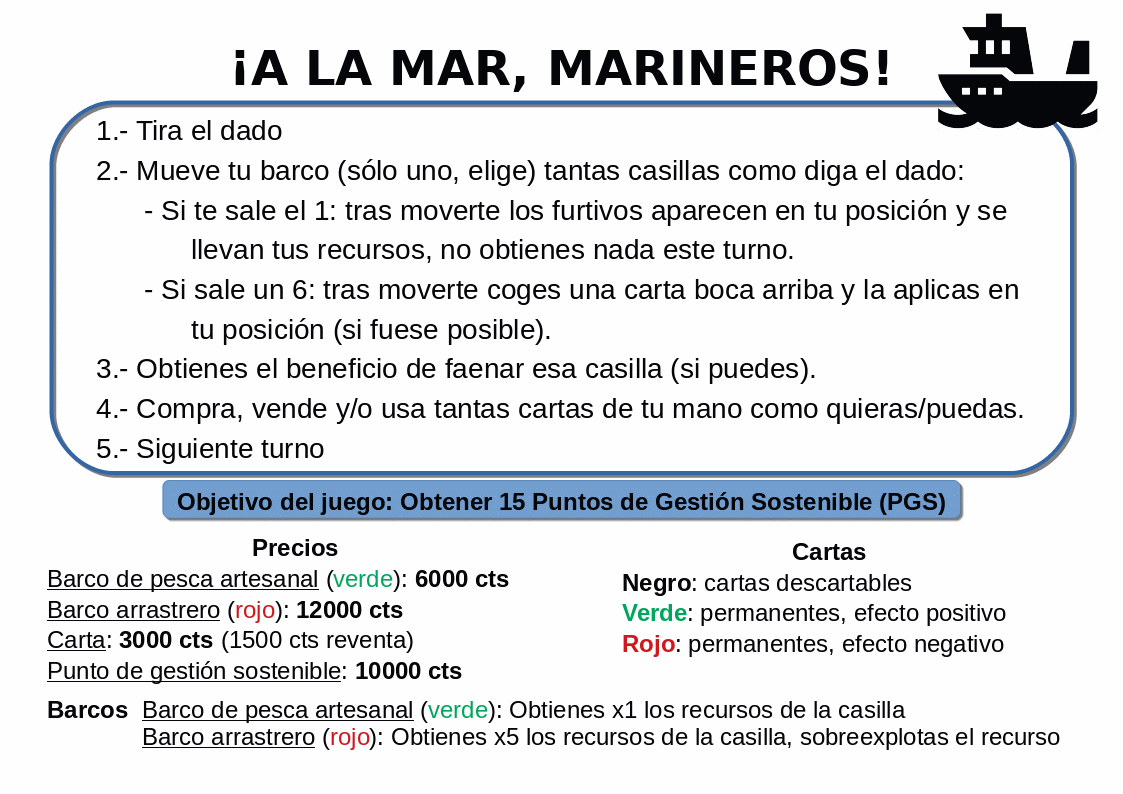
## INT 2: Motivación del Sistema

El juego ha demostrado ser una herramienta útil a la hora de aumentar el interés de los sujetos del proyecto, sin embargo, no era práctico, dado que era necesario transportar todas las piezas del juego y montarlo en el lugar del estudio, además tras cada partida era necesario restablecer el tablero a su estado original, dado que el estado del tablero varía según va avanzando la partida, como describiremos posteriormente en PLACEHOLDER.

Teniendo esto en mente surge una necesidad, encontrar una forma de facilitar el proceso de jugar una partida en cualquier lugar y/o momento en el que se requiera. Para esto, una de las primeras ideas que se nos viene a la mente es la de un crear un **videojuego replicando el juego de tablero.**

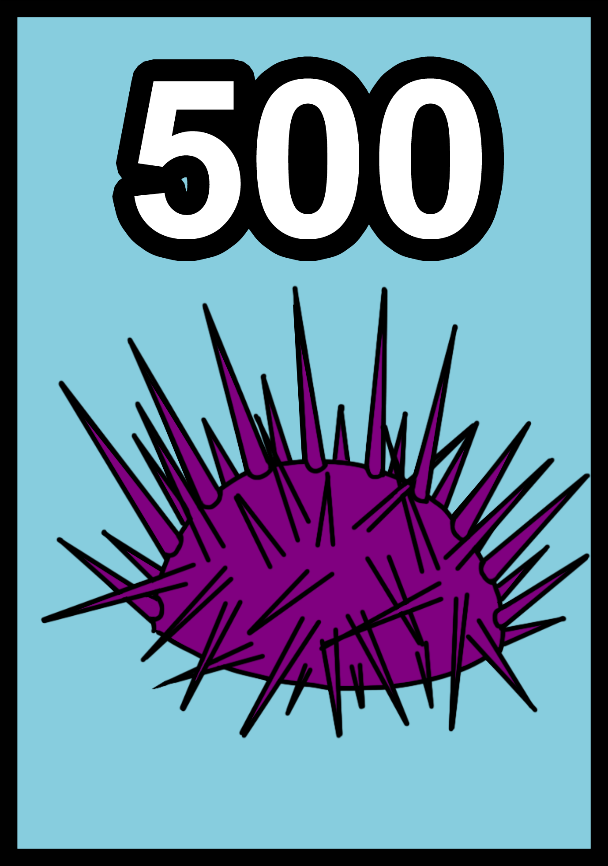
* Comenzar una nueva partida en un videojuego es tan simple como hacer un par de clics.
* Transportar un videojuego es mucho más simple que transportar un juego de mesa, ya que solamente necesitas un dispositivo adecuado capaz de ejecutarlo, **como un portátil, una Tablet o incluso un teléfono móvil.**
* Sin embargo, algo que debemos tener en cuenta es que los videojuegos suelen tener mayor aceptación entre el público más joven, mientras que el público más adulto suele preferir los juegos físicos. Acorde a lo descrito en el PLACEHOLDER, el público objetivo de este proyecto es gente en general joven, por lo que no deberíamos encontrarnos con rechazo si les presentamos un videojuego

## INT 3: Descripción del Juego y Reglas

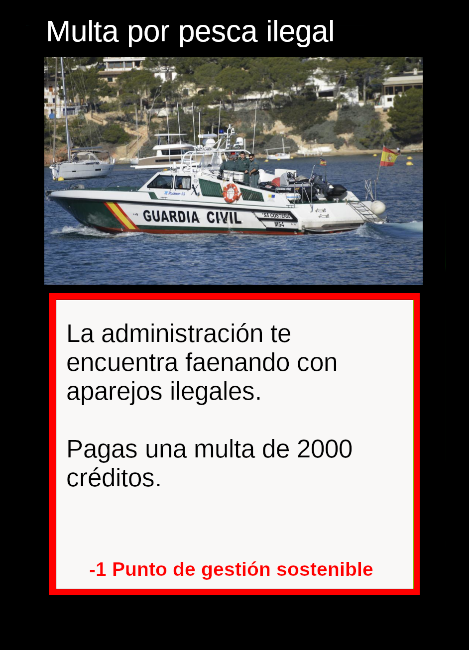


**Cheatsheet presentada a los jugadores del juego de mesa.**

Sustainable Sea (¡A la mar, marineros! En español) es un juego para 2 a 4 jugadores semejante al Monopoly clásico, tenemos un tablero de casillas con progresión lineal (no hay elecciones a la hora de avanzar, una casilla solo puede llevar a la siguiente). La condición de victoria es llegar a **15 puntos de gestión sostenible** (posteriormente en el documento quedan abreviados como **PGS**).



**Ejemplo de casilla**

**Ejemplos de cartas. En orden una carta descartable (negra), permanente negativa (roja) y permanente positiva (verde).**

Los jugadores comienzan la partida con un barco cada uno, sin créditos (posteriormente en el documento quedan abreviados como **CTS**) y sin PGS. Para conseguir CTS deberán faenar en las casillas del tablero, las cuales tienen un coste asociado. Por otro lado, para conseguir PGS deberán realizar acciones consideradas sostenibles mediante el juego de cartas o comprar un PGS por 10.000 CTS.

Durante su turno, los jugadores pueden hacer las siguientes acciones:

* Tirar el dado, mover uno de sus barcos, tantas casillas como sea el resultado de la tirada y obtener los recursos marcados en la casilla.
* Comprar cartas, por 3000 CTS cada una.
* Vender cartas, recibiendo 1500 CTS por cada una.
* Jugar tantas cartas como se quiera.
* Comprar PGS, por 10.000 CTS cada uno.

Si la tirada resulta en un 6, el jugador robará una carta y se jugará inmediatamente sobre sí mismo o sobre la casilla destino del barco que se haya movido, dependiendo de la naturaleza de la carta.

Si la tirada resulta en un 1, aparecerán **furtivos** en la casilla destino del barco que se haya movido, una casilla ocupada por furtivos **no da recursos a los jugadores**.

Si un barco atraviesa la casilla de salida, recibe un cobro de 2000 CTS y le da la oportunidad al jugador de comprar un nuevo barco:

* Un barco artesanal por 6000 CTS. Los barcos artesanales son los barcos estándar del juego, no tienen ninguna bonificación especial.
* Un barco de arrastre por 12.000 CTS. Los barcos de arrastre obtienen **cinco veces el número de recursos de las casillas en las que faenan**, sin embargo, si un barco de arrastre faena sobre una casilla, **esta queda sobreexplotada**. Al igual que si la casilla estuviera ocupada por furtivos, las casillas sobreexplotadas **no dan recursos a los jugadores.**
* No existe ninguna limitación al número de barcos que un jugador pueda tener.

## INT 4: Objetivo del Proyecto

El objetivo del proyecto consiste en entregar un videojuego basado en Sustainable Sea que permita jugar una partida completa, tal y como si se estuviera jugando con el juego físico, mejorando así todos los procesos actuales necesarios para el proyecto, tanto el transporte del juego, como el montaje del tablero, como la preparación para cada partida.

## INT 5: Metodología de Trabajo

Para este proyecto se utiliza una metodología ágil basada en **sprints** cortos de 2 semanas, durante los cuales se ampliará el producto mediante **features**, que serán extraídas de los requisitos al inicio de cada sprint.

### MT 1: Planificación y Retrospectiva del sprint

Al inicio de un sprint, se realizará una reunión, ya sea presencial o por videoconferencia, entre desarrollador y tutor, en la que se decidirá el alcance del sprint y la fecha de entrega (normalmente el mismo día dos semanas más tarde). Durante esta reunión se revisará lo realizado en el sprint anterior mediante una prueba en vivo y se discutirán las decisiones que hayan sido tomadas durante el desarrollo, así como aclarar dudas que hayan podido surgir. Estas reuniones no tendrán una duración superior a una hora y tras ellas se redactará un acta que recoja lo más destacable, resumido de la forma más simple posible. Estas actas están disponibles como anexo en PLACEHOLDER.

### MT 2: Extracción de las features

Las features se extraerán de los requisitos en función de su prioridad, incrementando el valor y la funcionalidad del producto hasta llegar al producto mínimo viable (a posterior PMV). En nuestro caso el PMV será jugar una partida completa de Sustainable Sea en un dispositivo, tal y como se ha definido en el apartado anterior PLACEHOLDER.

### MT 3: Definición de una feature

Una feature estará compuesta por un título simple que defina su objetivo principal y una serie de criterios de aceptación que el producto deberá cumplir para dar esa feature como satisfecha al finalizar el sprint. Cuando dichos criterios de aceptación se cumplan, el código se subirá a la rama master del repositorio y se grabará un video demostrativo de la nueva funcionalidad, tal y como se define en el plan de pruebas del proyecto PLACEHOLDER. Se adjuntará el video al issue correspondiente a la feature y se cerrará, añadiendo comentarios si se considera oportuno.



**Ejemplo de una feature como issue en el repositorio**

### MT 4: Rama master del proyecto

Los desarrollos serán implementados y probados en la rama DEVELOP, una vez validado que se cumplen los criterios de aceptación, el código se subirá a la rama master del repositorio.

### MT 5: Definición del Plan de Trabajo

Deben tenerse en cuenta las siguientes consideraciones:

1. Este proyecto es realizado por un estudiante, por lo que la planificación del comienzo de este (de Octubre a Enero) es mucho más laxa que la del resto (Febrero a Julio).
2. Los requisitos por realizar durante un sprint son decididos durante la reunión retrospectiva, por lo que esta planificación será viva y cambiante a lo largo del proyecto.
3. Durante un sprint, se pueden realizar otros trabajos de menor prioridad si se completa la planificación. Un ejemplo sería la creación del menú principal o la animación de los barcos.

La planificación completa del proyecto figura como anexo del mismo, tal y como se indica en PLACEHOLDER.

El proyecto tiene fecha de comienzo el día 1 de Octubre de 2019 y como fecha de fin el día 29 de Junio de 2020, día en el que se realiza la demo con los clientes, tal y como se define en PLACEHOLDER. El calendario de trabajo está definido en 7 horas a la semana, lo que nos resulta en un total de 448 horas para el proyecto.

# Estudio de Viabilidad del Sistema

Fase de Desarrollo

EVS

## EVS: Estudio y Valoración de Alternativas de Solución y Selección de Alternativa Final

En esta sección se describirán las diferentes alternativas viables en cuanto al framework base a utilizar para desarrollar el juego, así como de sus ventajas, sus inconvenientes y otros factores que pudieran afectar a la ejecución del proyecto. Finalmente, se realizará una comparación objetiva de las diferentes alternativas y se llegará a una conclusión sobre el sistema utilizado.

### Unity

#### Descripción del sistema



**Logo de Unity**

Unity es un motor gráfico multiplataforma propiedad de la empresa homónima que tiene a su espalda más de una década y soporte para más de 25 plataformas. Es uno de los motores gráficos más usados en la actualidad, especialmente entre desarrolladores independientes y pequeñas empresas, debido a su modelo de pago por suscripción, que permite su uso gratuito mientras que el juego no supere los 100.000 $ en beneficios anuales.

Unity permite crear juegos tanto en 2D como en 3D utilizando como lenguaje C# y poniendo a disposición de los desarrolladores potentes herramientas desde el mismo editor.

#### Ventajas

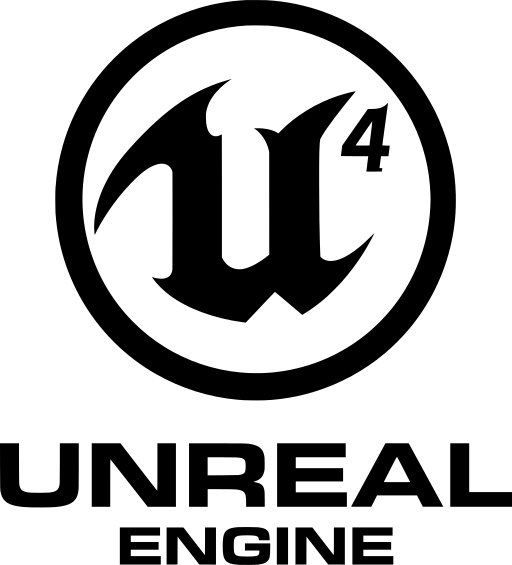
* Documentación extensa y bien definida, complementada también con un foro muy activo de la comunidad de desarrolladores y una serie de tutoriales interactivos. Esto supone una facilidad extra a la hora de empezar con Unity desde cero.
* Ligero. Unity suele estar relacionado con juegos ligeros, sobre todo móviles, esto es un punto a favor dado el objetivo del proyecto, ser utilizado en estudios de investigación (PLACEHOLDER).
* Utiliza C#. Uno de los lenguajes orientados a objetos más populares, esto significa que el desarrollo no sea especialmente complejo para el desarrollador y además permite que gente no relacionada con Unity pueda entender el código.
* Editor muy potente e intuitivo. El editor de Unity permite numerosas virguerías, como arrastrar recursos para referenciarlos en los ficheros de código.
* Múltiples herramientas integradas en el editor. El editor de Unity incluye varias herramientas para complementar el desarrollo, como el Test Runner que se utilizará durante las pruebas unitarias (PLACEHOLDER) o el Unity Profiler, para mediciones de rendimiento del código del juego. Por otro lado, también tiene integración nativa con Visual Studio, lo que permite un desarrollo y una depuración del juego muy ágiles.
* Multiplataforma. Unity permite cambiar la plataforma objetivo con una simple opción a la hora de compilar el juego.

#### Inconvenientes

* Aunque el juego sea ligero, el proyecto se expande rápidamente debido a la gran cantidad de recursos que se crean (materiales, texturas, prefabricados, modelos, etc).
* El rendimiento puede llegar a suponer un problema si no se cuida el código con optimizaciones.

### Unreal Engine

#### Descripción del sistema



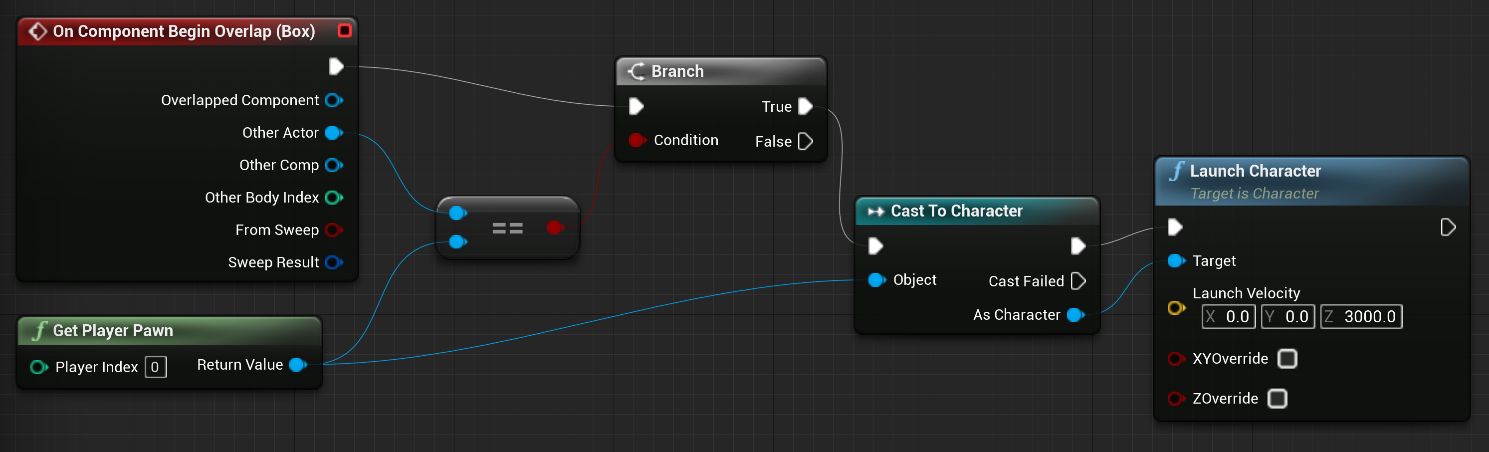
**Logo de Unreal Engine 4**

Unreal Engine es, junto a Unity, de los motores gráficos más usados actualmente. Al igual que Unity es un motor gráfico multiplataforma que sorporta juegos tanto en 2D como en 3D, sin embargo, Unreal es superior en cuanto al 3D gracias a su superior renderizado. Unreal Engine es propiedad de Epic Games y también presenta un modelo de suscripción, en este caso permite su uso gratuito hasta superar el millón de dólares, a partir de cuyo momento comienza a cobrar un 5% de los ingresos, salvo que se acceda a publicar el juego en la plataforma virtual de Epic, la Epic Games Store.

Unreal Engine permite crear videojuegos (y otras aplicaciones) mediante un lenguaje propio basado en grafos de acciones llamado Blueprints o mediante programación tradicional en C++.

#### Ventajas

* Multiplataforma. Al igual que Unity, Unreal permite cambiar la plataforma objetivo de forma simple.
* El editor de Unreal, aunque más complejo, ofrece muchas más posibilidades.
* Blueprints. Unreal tiene una herramienta conocida como Blueprints, la cual permite programar el juego mediante grafos de acciones. Esto es especialmente útil para funcionalidades simples o si no se conoce C++, que es el lenguaje usado por el motor.
* Estudiado en Software de Entretenimiento y Videojuegos. El desarrollador ya está familiarizado y tiene experiencia con este motor gráfico.



**Ejemplo de Blueprint. Imagen sacada de** [**la documentación de Unreal Engine 4**](https://docs.unrealengine.com/en-US/Engine/Blueprints/QuickStart/index.html)

#### Inconvenientes

* Requiere mayor potencia del equipo de desarrollo que Unity.
* Los proyectos de Unreal crecen muchísimo más deprisa que los proyectos de Unity.
* La comunidad de Unreal es mucho más pequeña que la de Unity. Este motor es sobretodo usado en grandes compañías, no tanto por desarrolladores independientes, por lo que hay menos información publicada en la red.

### Cocos2d-JS

#### Descripción del sistema



**Logo de Cocos2D-JS**

Cocos2D es un framework open-source para el desarrollo de juegos de navegador en 2D, disponible en distintos lenguajes, como Objective-C, C++ o Javascript. En este caso estamos analizando la versión de Javascript, utilizada durante la asignatura de Software de Entretenimiento y Videojuegos.

#### Ventajas

* Open-source.
* Javascript. El código es únicamente Javascript, de modo que puede ser fácilmente revisado por personas que no estén familiarizadas con Cocos.
* Pensado para juegos de navegador. Esto supone dos cosas: los juegos no ocupan gran espacio y son multiplataforma nativos (siempre que el dispositivo soporte HTML).
* Estudiado en Software de Entretenimiento y Videojuegos. El desarrollador ya está familiarizado y tiene experiencia con este framework.

#### Inconvenientes

* Limitado a juegos en 2D.
* Muy poca comunidad .
* Documentación incompleta (o incluso inexistente en algunos casos).

### Conclusión

Tras estudiar las distintas alternativas previamente expuestas y sopesar las ventajas y inconvenientes de cada una se llegan a las siguientes conclusiones:

* Utilizar Unreal Engine para un juego de este alcance se considera demasiado, no necesitamos semejante potencia por parte del motor y solamente estaríamos dificultando la capacidad del juego de ser ejecutado en equipos de menor calibre, lo que podría suponer problemas durante los estudios de investigación de los clientes.
* Cocos2D nos ofrece una muy tentadora posibilidad de hacer un juego de navegador y permitir así que los clientes lo puedan alojar en un servidor simple e incluirlo en una página web, eliminando así la necesidad de equipos potentes, podrían utilizar cualquier Tablet del mercado para realizar sus estudios allá donde vayan. Sin embargo, la limitación de solamente poder hacer juegos en 2D es demasiado grande, especialmente para un juego de tablero donde creemos es importante darle buena percepción al jugador, por no mencionar los incontables problemas que se encontrarían durante el desarrollo ante la falta de documentación.
* Unity, aunque el desarrollador no tenga experiencia alguna, es aclamado por la comunidad por su facilidad de uso y gran potencia y parece el motor idóneo para realizar un juego con las características definidas. Teniendo esto en cuenta, así como sus inconvenientes, **se elige Unity como base para realizar el proyecto.**

# Análisis del Sistema de Información

Fase de Desarrollo

ASI

## ASI 1: Definición del Sistema

### Determinación del Alcance del Sistema

El alcance del sistema será definido como el **mínimo indispensable para jugar una partida completa de Sustainable Sea,** por lo que deberán implementarse todas las reglas del juego y las funcionalidades básicas de un juego de tablero.

Para esta primera iteración del proyecto nos ceñiremos únicamente a un sistema multijugador de un único dispositivo, es decir, varios jugadores en un único sistema, siendo los jugadores los que roten los controles. En iteraciones posteriores del proyecto puede ser interesante implementar un multijugador en línea, para lo que sería necesario establecer una aplicación que actúe como **servidor o alojamiento de las partidas** y otra aplicación que serían las que **se conectarían al servidor y enviarían las decisiones de los jugadores.**

En cuanto al aspecto gráfico del sistema, **tendrá prioridad baja,** se implementarán cambios para mejorar aspectos visuales del juego, pero lo principal será completar la funcionalidad.

## ASI 2: Establecimiento de Requisitos

### Obtención de los Requisitos del Sistema

En esta sección se especifican los requisitos del sistema que serán implementados durante esta iteración del proyecto. Los requisitos que han quedado fuera de la planificación, así como nuevos requisitos creados como fruto de las pruebas del sistema, pueden ser encontrados en PLACEHOLDER.

#### Requisitos del sistema de partidas

|  |  |
| --- | --- |
| Código | Descripción del Requisito |
| SIS1 | El sistema permitirá jugar una partida a Sustainable Sea. |
| SIS1.1 | El sistema permitirá crear una nueva partida de Sustainable Sea. |
| SIS1.1.1 | El creador podrá definir el número de jugadores. |
| SIS1.1.1.1 | El mínimo de jugadores será **2.** |
| SIS1.1.1.2 | El máximo de jugadores será **4.** |
| SIS1.1.2 | Al iniciar una partida: |
| SIS1.1.2.1 | El sistema cargará el tablero. |
| SIS1.1.2.1.1 | Las casillas de recursos serán ordenadas aleatoriamente. |
| SIS1.1.2.1.2 | El número de casillas de cada recurso será siempre el mismo. |
| SIS1.1.2.1.3 | El precio de cada casilla será siempre el mismo. |
| SIS1.1.2.1.4 | Los recursos, aunque ordenados aleatoriamenteestarán separados por **profundidad.** |
| SIS1.1.2.1.5 | Para facilitar una futura modificación, las casillas estarán definidas en un fichero de texto de la forma indicada en **Anexo** |
| SIS1.1.2.2 | El sistema asignará a cada jugador un barco pesquero artesanal. |
| SIS1.1.2.3 | Los jugadores empezarán con **0** CTS. |
| SIS1.1.2.4 | Los jugadores empezarán con **0** PGS. |
| SIS1.1.2.5 | Los jugadores empezarán **sin** cartas. |
| SIS1.1.3 | Durante cada turno: |
| SIS1.1.3.1 | El sistema mostrará un HUD. |
| SIS1.1.3.1.1 | El HUD mostrará para todos los jugadores: |
| SIS1.1.3.1.1.1 | El nombre. |
| SIS1.1.3.1.1.2 | El color. |
| SIS1.1.3.1.1.3 | El retrato. |
| SIS1.1.3.1.1.4 | Sus puntos de gestión sostenible. |
| SIS1.1.3.1.1.5 | Sus créditos. |
| SIS1.1.3.1.2 | El HUD mostrará a cada jugador: |
| SIS1.1.3.1.2.1 | Las cartas que posee. |
| SIS1.1.3.2 | El sistema permitirá lanzar un dado al jugador cuyo turno toque. |
| SIS1.1.3.2.1 | El jugador elegirá uno de sus barcos para moverlo tantas casillas como indique la tirada del dado. |
| SIS1.1.3.2.2 | Si sale un 6, el jugador robará una carta y la resolverá en la casilla destino si fuera posible. |
| SIS1.1.3.2.2.1 | En caso de que no fuera posible, el jugador se quedará con la carta. |
| SIS1.1.3.2.3 | Si sale un 1, los furtivos se moverán a la casilla destino. |
| SIS1.1.3.2.4 | El jugador obtendrá recursos de la casilla destino. |
| SIS1.1.3.2.4.1 | Si usa un barco artesanal, el jugador obtendrá recursos por valor del precio de la casilla. |
| SIS1.1.3.2.4.2 | Si usa un barco de arrastre, el jugador obtendrá recursos por valor del quíntuple del precio de la casilla. |
| SIS1.1.3.2.4.2.1 | Tras obtener recursos con un barco de arrastre, la casilla quedará marcada como sobreexplotada. |
| SIS1.1.3.2.4.2.1.1 | Las casillas sobreexplotadas no generan recursos. |
| sis1.1.3.2.4.3 | Si los furtivos están en esa casilla, el jugador no obtendrá recursos |
| sis1.1.3.3 | Previo al lanzamiento del dado, el jugador podrá realizar cualquiera de las siguientes acciones, tantas veces como quiera: |
| sis1.1.3.3.1 | Comprar un punto de gestión sostenible por **10000** créditos. |
| sis1.1.3.3.1.1 | Si el jugador compra un punto de gestión sostenible, el sistema: |
| sis1.1.3.3.1.1.1 | Restará el coste del punto de gestión sostenible a los créditos del jugador. |
| sis1.1.3.3.1.1.2 | Añadirá uno al contador de puntos de gestión sostenible del jugador en el HUD. |
| sis1.1.3.3.2 | Comprar una carta por **3000** créditos. |
| sis1.1.3.3.2.1 | Si el jugador compra una carta, el sistema: |
| sis1.1.3.3.2.1.1 | Restará el precio de la carta a los créditos del jugador. |
| sis1.1.3.3.2.1.2 | Añadirá una carta del mazo a la mano del jugador. |
| sis1.1.3.3.3 | Vender una carta de la mano para conseguir **1500** créditos. |
| sis1.1.3.3.3.1 | Si el jugador vende una carta, el sistema: |
| sis1.1.3.3.3.1.1 | Sumará el precio de la carta a los créditos del jugador. |
| sis1.1.3.3.3.1.2 | Eliminará la carta de la mano del jugador. |
| sis1.1.3.3.4 | Usar una carta de la mano. |
| sis1.1.3.3.4.1 | Si el jugador usa una carta, el sistema: |
| sis1.1.3.3.4.1.1 | Eliminará la carta de la mano del jugador. |
| sis1.1.3.3.4.1.2 | Aplicará el efecto de la carta jugada, dependiendo de su tipo. |
| sis1.1.3.4 | Tras la resolución del turno, el jugador podrá comprar un barco. |
| sis1.1.3.4.1 | El jugador solamente podrá comprar un barco si ha pasado por la casilla de salida (o ha caído en ella). |
| sis1.1.3.4.1.1 | El jugador podrá comprar un barco artesanal por **6000** créditos. |
| sis1.1.3.4.1.2 | El jugador podrá comprar un barco de arrastre por **12000** créditos. |
| sis1.1.3.4.1.3 | Si el jugador compra un barco, el sistema: |
| sis1.1.3.4.1.3.1 | Restará el coste del barco a los créditos del jugador. |
| sis1.1.3.4.1.3.2 | Añadirá un nuevo barco del tipo comprado a la flota del jugador. |
| sis1.1.3.4.1.3.3 | Generará un nuevo barco del tipo comprado en la casilla de salida. |
| sis1.1.4 | Cuando un jugador consiga **15 PGS**: |
| sis1.1.4.1 | El sistema terminará la partida. |
| sis1.1.4.1.1 | El sistema mostrará una pantalla de fin de partida, la cual mostrará: |
| sis1.1.4.1.1.1 | El nombre del jugador ganador. |
| sis1.1.4.1.1.2 | El color del jugador. |
| sis1.1.4.2 | El sistema permitirá volver al menú principal. |
|  |  |

#### Requisitos de cartas

|  |  |
| --- | --- |
| Código | Descripción del Requisito |
| CARD1 | Las cartas pueden ser de dos tipos: |
| CARD1.1 | Descartables (negro). |
| CARD1.1.1 | Las cartas descartables pueden ser jugadas y su efecto es resuelto inmediatamente. |
| CARD1.1.2 | Una vez resuelto el efecto, la carta es colocada en la pila de descartes. |
| CARD1.1.2.1 | Cuando se acaben las cartas del mazo: |
| CARD1.1.2.1.1 | Se barajará la pila de descartes |
| CARD1.1.2.1.2 | La pila de descartes pasará a ser el nuevo mazo |
| CARD1.2 | Permanentes. |
| CARD1.2.1 | Las cartas permanentes se colocan sobre una casilla. |
|  |  |
|  |  |
| CARD1.2.1.1 | El efecto de la carta se resuelve cada vez que se obtienen recursos de la casilla afectada. |
| CARD1.2.1.2 | El efecto de la carta durará: |
| CARD1.2.1.2.1 | Hasta el final de la partida. |
| CARD1.2.1.2.2 | Hasta que sea sustituida por una nueva carta. |
| CARD1.2.3 | Las cartas permanentes pueden ser, a su vez, de dos tipos: |
| CARD1.2.3.1 | Positivo (verde). |
| CARD1.2.3.1.1 | El efecto de la carta es positivo sobre la casilla. P.E. otorgar el doble de recursos. |
| CARD1.2.3.2 | Negativo (rojo). |
| CARD1.2.3.2.1 | El efecto de la carta es negativo sobre la casilla. P.E. otorgar la mitad de recursos o no otorgar recursos en absoluto. |
| CARD1.2.4 | Las cartas permanentes pueden afectar a las casillas adyacentes |
| CARD1.2.4.1 | El modificador para las casillas adyacentes puede ser diferente que el utilizado en la casilla en la que se juega la carta. |
|  |  |

#### Requisitos del menú principal

|  |  |
| --- | --- |
| Código | Descripción del Requisito |
| MENU1 | El sistema mostrará un menú principal al iniciar el juego. |
| MENU1.1 | El menú principal mostrará las siguientes opciones. |
| MENU1.1.1 | Nueva partida. |
| MENU1.1.1.1 | Al elegir esta opción, el sistema se comportará como se define en SIS1.1 |
| MENU1.1.2 | Salir del juego. |
| MENU1.1.2.1 | Al elegir esta opción, se cerrará el juego. |
| MENU1.1.3 | Varios idiomas: |
| MENU1.1.3.1 | Español. |
| MENU1.1.3.2 | Inglés. |
| MENU1.1.3.3 | Al elegir uno de los idiomas: |
| MENU1.1.3.3.1 | Se traducirán al nuevo idioma los recursos del menú principal. |
| MENU1.1.3.3.2 | Todos los nuevos recursos que se carguen en el juego estarán en este idioma. |
|  |  |

#### Requisitos no funcionales

|  |  |
| --- | --- |
| Código | Descripción del Requisito |
| RNF1 | El sistema mostrará los textos en el idioma elegido por el jugador en el menú principal. Tal y como se define en MENU1.1.3.3.2 |
| RNF1.1 | Para una mayor facilidad, los textos estarán externalizados en ficheros de texto, uno por cada idioma soportado. |
| RNF2 | Se utilizará Git como programa de control de versiones. |
| RNF2.1 | El repositorio estará alojado en GitHub. |
| RNF3 | Los recursos del proyecto serán gratuitos, de libre uso o de creación propia. |
| RNF3.1 | En caso de que el autor requiera reconocimiento, se añadirá su nombre y el enlace al recurso en el apartado de créditos. |
|  |  |

#### Anexo

|  |  |
| --- | --- |
| Código | Descripción del Requisito |
| ANEX1 | **Fichero de casillas**  El fichero contendrá la información de las casillas de la siguiente forma:  NOMBRE;RATIO;RECURSOS  P.E. COS\_ALGA;0.5;1600 |
|  |  |

### Especificación de Casos de Uso

Esencialmente nuestra aplicación tiene un único actor, que sería **el jugador** y tendríamos un único caso de uso, que sería **jugar una partida.** Este caso de uso se dividiría en varios sub-casos, por ejemplo crear una partida, mover un barco o jugar una carta.

A continuación, se muestran ejemplos de algunos de estos sub-casos, así como una ligera descripción de cada uno de ellos.

|  |
| --- |
| **Nombre del Caso de Uso** |
| Crear una partida |
| **Descripción** | |
| El jugador, desde el menú principal, iniciará una nueva partida, seleccionando el número de jugadores que desee. Una vez finalizado el proceso de carga, el jugador verá el tablero generado, con la interfaz gráfica correspondiente al número de jugadores seleccionado y el turno del primer jugador listo para jugar. | |

|  |
| --- |
| **Nombre del Caso de Uso** |
| Mover un barco |
| **Descripción** | |
| El jugador tirará el dado, una vez obtenido el resultado seleccionará uno de los barcos que esté en su posesión y este se moverá tantas casillas como resultado haya salido del dado. Una vez llegue a la casilla de destino, el jugador recibirá tanto dinero como recursos haya en la casilla de destino. | |

|  |
| --- |
| **Nombre del Caso de Uso** |
| Comprar una carta |
| **Descripción** | |
| El jugador, si tuviera 3000 CTS o más podrá indicar al sistema que quiere comprar una carta. El sistema añadirá a la mano del jugador una carta del mazo y le restará los 3000 CTS correspondientes a la compra de la carta. | |

|  |
| --- |
| **Nombre del Caso de Uso** |
| Vender una carta |
| **Descripción** | |
| El jugador, si tuviera una o más cartas en su inventario podrá indicar al sistema que quiere vender una o más cartas. Tras el jugador elegir las cartas que desea vender, el sistema eliminará dichas cartas de la mano del jugador, añadiéndolas al mazo de descartes. El sistema añadirá a los créditos del jugador 1500 CTS por cada carta vendida. | |

|  |
| --- |
| **Nombre del Caso de Uso** |
| Jugar una carta |
| **Descripción** | |
| El jugador, si tuviera al menos una carta, puede indicar al sistema que quiere jugarla. El jugador deberá indicar la casilla o el jugador (dependiendo del tipo de carta) en el que desea jugarlo.  El sistema:   1. Eliminará dicha carta de la mano del jugador, añadiéndola al mazo de descartes. 2. Aplicará el efecto de la carta contra la casilla o el jugador objetivo. | |

|  |
| --- |
| **Nombre del Caso de Uso** |
| Comprar un barco artesanal |
| **Descripción** | |
| El jugador, si tiene 6000 CTS o más y ha atravesado la casilla de salida durante su turno, podrá indicar al final de este que quiere comprar un barco artesanal.  El sistema:   1. Creará un nuevo barco artesanal en la casilla de salida 2. Asignará el nuevo barco al jugador 3. Restará 6000 CTS al dinero del jugador 4. Terminará el turno del jugador | |

|  |
| --- |
| **Nombre del Caso de Uso** |
| Comprar un barco de arrastre |
| **Descripción** | |
| El jugador, si tiene 12000 CTS o más y ha atravesado la casilla de salida durante su turno, podrá indicar al final de este que quiere comprar un barco de arrastre.  El sistema:   1. Creará un nuevo barco de arrastre en la casilla de salida 2. Asignará el nuevo barco al jugador 3. Restará 12000 CTS al dinero del jugador   Terminará el turno del jugador | |

|  |
| --- |
| **Nombre del Caso de Uso** |
| Comprar un PGS |
| **Descripción** | |
| El jugador, si tiene 10000 CTS o más, podrá indicar al sistema que quiere comprar un PGS. El sistema restará 10000 CTS al dinero del jugador y le añadirá un PGS a su cantidad de PGS. | |

## ASI 3: Análisis de los Casos de Uso

A continuación, se analizan con mayor detalle los casos de uso propuestos en el apartado anterior, explicando los pasos necesarios para la realización de cada uno de ellos, así como los escenarios alternativos si los hubiera.

### Crear una partida

|  |  |
| --- | --- |
| Crear UNA PARTIDA | |
| Precondiciones | El usuario debe estar en el menú principal |
| Poscondiciones | Debe haberse creado una nueva partida, generado un nuevo tablero y haberse inicializado los jugadores, así como sus barcos. |
| Actores | Iniciado por el jugador y terminado por el sistema |
| Descripción | El jugador:  Accederá a la pantalla de nueva partida  Seleccionará el número de jugadores para la partida  El sistema cargará una nueva partida |
| Variaciones (escenarios secundarios) | **-** |
| Excepciones | **-** |
| Notas | - |

### Mover un barco

|  |  |
| --- | --- |
| Mover un barco | |
| Precondiciones | 1. El jugador está en una partida 2. El jugador tiene el turno actual |
| Poscondiciones | El barco del jugador se ha desplazado N casillas |
| Actores | Iniciado por un el jugador y finalizado por el sistema. |
| Descripción | El sistema muestra la pantalla de turno del jugador.  El jugador selecciona tirar el dado.  El sistema lanza el dado.  El sistema devuelve el resultado de la tirada (N).  El usuario elige uno de sus barcos.  El barco se mueve N casillas. |
| Variaciones (escenarios secundarios) | **Escenario Alternativo 1**: El jugador elige un barco de otro jugador  El sistema no mueve el barco |
| Excepciones | **-** |
| Notas | - |

### Comprar una carta

|  |  |
| --- | --- |
| Comprar una carta | |
| Precondiciones | 1. El jugador está en una partida 2. El jugador tiene el turno actual |
| Poscondiciones | El jugador recibe una nueva carta |
| Actores | Iniciado por un el jugador y finalizado por el sistema. |
| Descripción | El sistema muestra la pantalla de turno del jugador.  El jugador selecciona comprar una carta.  El sistema asigna a la mano del jugador una carta del mazo.  El sistema resta al jugador los CTS correspondientes a comprar una carta |
| Variaciones (escenarios secundarios) | **Escenario Alternativo 1**: El jugador no tiene suficiente dinero  El sistema no permite la compra.  Volver al paso 1 del escenario principal. |
| Excepciones | **-** |
| Notas | - |

### Vender una carta

|  |  |
| --- | --- |
| Vender una carta | |
| Precondiciones | 1. El jugador está en una partida 2. El jugador tiene el turno actual |
| Poscondiciones | El jugador gana dinero a cambio de sus cartas |
| Actores | Iniciado por un el jugador y finalizado por el sistema. |
| Descripción | El sistema muestra la pantalla de turno del jugador.  El jugador selecciona vender cartas.  El jugador selecciona las cartas que quiere vender.  El sistema suma al jugador los CTS correspondientes a vender cada carta.  El sistema elimina las cartas de la mano del jugador.  El sistema añade las cartas al mazo de descartes. |
| Variaciones (escenarios secundarios) | **Escenario Alternativo 1**: El jugador no tiene cartas.  El sistema no permite vender cartas.  Volver al paso 1 del escenario principal. |
| Excepciones | **-** |
| Notas | - |

### Jugar una carta

|  |  |
| --- | --- |
| Jugar una carta | |
| Precondiciones | 1. El jugador está en una partida 2. El jugador tiene el turno actual |
| Poscondiciones | El jugador pierde una carta, la casilla/jugador objetivo sufre un efecto |
| Actores | Iniciado por un el jugador y finalizado por el sistema. |
| Descripción | El sistema muestra la pantalla de turno del jugador.  El jugador selecciona jugar carta.  El jugador selecciona la carta que desea jugar.  El jugador selecciona la casilla o el jugador contra el que jugar la carta.  El sistema elimina la carta de la mano del jugador.  El sistema aplica el efecto al jugador o casilla objetivo. |
| Variaciones (escenarios secundarios) | **Escenario Alternativo 1**: El jugador no tiene cartas.  El sistema no permite jugar cartas.  Volver al paso 1 del escenario principal.  **Escenario Alternativo 2**: El jugador juega la carta contra sí mismo.  El sistema no permite jugar la carta contra sí mismo.  Volver al paso 4 del escenario principal.  **Escenario Alternativo 3**: El objetivo no es seleccionable.  El sistema no permite jugar la carta.  Volver al paso 4 del escenario principal. |
| Excepciones | **-** |
| Notas | - |

### Comprar un Barco Artesanal

|  |  |
| --- | --- |
| Comprar un Barco Artesanal | |
| Precondiciones | 1. El jugador está en una partida 2. Ha terminado el turno del jugador 3. Un barco del jugador ha atravesado la casilla de salida |
| Poscondiciones | El jugador tiene un barco artesanal más |
| Actores | Iniciado por un el jugador y finalizado por el sistema. |
| Descripción | El sistema muestra la pantalla de compra de barcos.  El jugador selecciona comprar barco artesanal.  El sistema resta al jugador los CTS correspondientes a comprar un barco artesanal.  El sistema genera un nuevo barco artesanal en la casilla de salida.  El sistema asigna el nuevo barco al jugador.  El sistema termina el turno del jugador. |
| Variaciones (escenarios secundarios) | **Escenario Alternativo 1**: El jugador no tiene dinero suficiente.  El sistema no permite la compra.  Avanzar al último paso del escenario principal. |
| Excepciones | **-** |
| Notas | - |

### Comprar un Barco de Arrastre

|  |  |
| --- | --- |
| Comprar un Barco de Arrastre | |
| Precondiciones | 1. El jugador está en una partida 2. El jugador tiene el turno actual |
| Poscondiciones | El jugador tiene un barco de arrastre más |
| Actores | Iniciado por un el jugador y finalizado por el sistema. |
| Descripción | El sistema muestra la pantalla de compra de barcos.  El jugador selecciona comprar barco artesanal.  El sistema resta al jugador los CTS correspondientes a comprar un barco artesanal.  El sistema genera un nuevo barco artesanal en la casilla de salida.  El sistema asigna el nuevo barco al jugador.  El sistema termina el turno del jugador. |
| Variaciones (escenarios secundarios) | **Escenario Alternativo 1**: El jugador no tiene dinero suficiente.  El sistema no permite la compra.  Avanzar al último paso del escenario principal. |
| Excepciones | **-** |
| Notas | - |

### Comprar un PGS

|  |  |
| --- | --- |
| Comprar un PGS | |
| Precondiciones | 1. El jugador está en una partida 2. El jugador tiene el turno actual |
| Poscondiciones | El jugador obtiene un PGS |
| Actores | Iniciado por un el jugador y finalizado por el sistema. |
| Descripción | El sistema muestra la pantalla de turno del jugador.  El jugador selecciona comprar PGS.  El sistema resta al jugador los CTS correspondientes a comprar un PGS.  El sistema añade un PGS al jugador. |
| Variaciones (escenarios secundarios) | **Escenario Alternativo 1**: El jugador no tiene dinero.  El sistema no permite comprar PGS.  Volver al paso 1 del escenario principal. |
| Excepciones | **-** |
| Notas | - |

## ASI 4: Análisis de Clases

En esta sección identificaremos a partir de los requisitos y los casos de uso un conjunto de clases inicial para el sistema, que nos servirán de base para un diseño más elaborado en las posteriores fases del documento.

### Diagrama de Clases

Este sería el diagrama resultado del análisis, tenemos un conjunto reducido de clases fuertemente relacionadas entre sí.

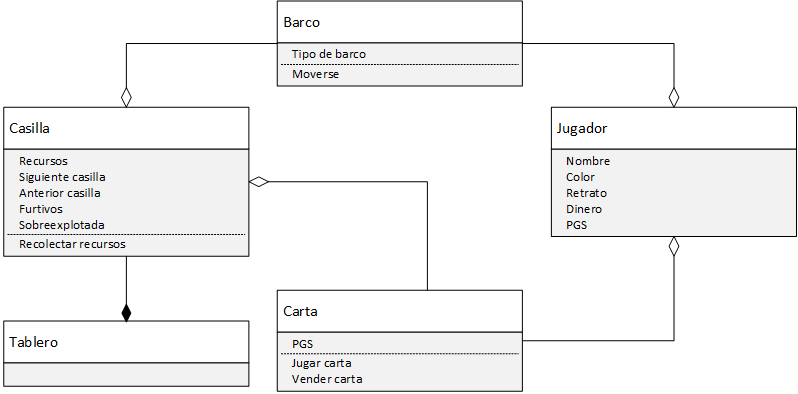


Diagrama de clases (fase análisis)

### Descripción de las Clases

A continuación, daremos una leve descripción de las clases que hemos identificado, así como explicaremos sus responsabilidades y funciones principales.

#### Tablero

|  |
| --- |
| **Nombre del Caso de Uso** |
| Crear una partida |
| **Descripción** | |
| Tablero será la clase encargada de generar nuestro tablero de juego, así como de crear los primeros barcos e inicializar los atributos de los jugadores (dinero, PGS). | |
| **Responsabilidades** | |
| * Generación del tablero * Inicialización de componentes del juego (barcos, jugadores, casillas) | |
| **Atributos Propuestos** | |
| **-** | |
| **Métodos Propuestos** | |
| **-** | |

***Casilla***

|  |
| --- |
| **Nombre del Caso de Uso** |
| Crear una partida, Mover un barco, Jugar una carta, Comprar un barco (arrastrero o artesanal) |
| **Descripción** | |
| Casilla será la clase que formará nuestro tablero de juego, los barcos deberán recorrerlas para avanzar. Cada casilla tendrá la información de su casilla siguiente, formando una cadena. Tendrá además la información de los barcos que hay actualmente en ella. | |
| **Responsabilidades** | |
| * Recursos * Posición de los barcos * Enlazar las unas con las otras para formar el tablero * Objetivo para las cartas | |
| **Atributos Propuestos** | |
| **Recursos**: La cantidad de recursos que esta casilla da por defecto  **Siguiente casilla**: Enlace con la siguiente casilla del tablero, necesario para que avancen los barcos. También necesario para las cartas que afectan a casillas adyacentes.  **Anterior casilla**: Enlace con la casilla anterior del tablero, necesario para las cartas que afectan a casillas adyacentes.  **Furtivos**: Indica si hay furtivos en esta casilla, las casillas con furtivos no dan recursos.  **Sobreexplotada**: Indica si la casilla está sobreexplotada, las casillas sobreexplotadas no dan recursos. | |
| **Métodos Propuestos** | |
| **Recolectar recursos:**Es conveniente crear un método que nos devuelva directamente los recursos obtenidos de la casilla, dado que hay tantas variables sobre esto (si tiene una carta, si tiene furtivos, si esta sobreexplotada, etc.) | |

***Jugador***

|  |
| --- |
| **Nombre del Caso de Uso** |
| Comprar una carta, Vender una carta, Jugar una carta, Comprar un barco (de arrastre o artesanal), Comprar un PGS |
| **Descripción** | |
| Jugador será una clase en la que principalmente se guardará información sobre cada jugador, no se pretende que tenga mucha funcionalidad por sí solo. Se guardará además una referencia a sus barcos y a sus cartas. | |
| **Responsabilidades** | |
| Información del jugador | |
| **Atributos Propuestos** | |
| **Nombre**: Atributo que guarda el nombre del jugador  **Color**: Color del jugador, utilizado para pintar los barcos de su propiedad  **Dinero**: Dinero del jugador  **PGS**: PGS del jugador  **Retrato**: Retrato del jugador que se mostrará en las diferentes pantallas del juego. | |
| **Métodos Propuestos** | |
| - | |

***Barco***

|  |
| --- |
| **Nombre del Caso de Uso** |
| Mover un barco, Comprar un Barco (de arrastre o artesanal) |
| **Descripción** | |
| Barco será la clase que represente a las fichas de los jugadores. Cada jugador comenzará con un único barco, pero potencialmente puede tener un número ilimitado de ellos. Los barcos pueden ser de dos tipos, de arrastre o artesanales. La clase Barco tendrá la información de la casilla en la que está situado actualmente. | |
| **Responsabilidades** | |
| Representación del jugador en el tablero.  Recolección de recursos en las casillas. | |
| **Atributos Propuestos** | |
| **Tipo de barco**: El tipo de barco es importante a la hora de recolectar recursos, dado que los barcos de arrastre reciben el quíntuple de recursos, pero sobreexplotan la casilla. | |
| **Métodos Propuestos** | |
| **Moverse**: Moverse un número de casillas determinado desde la casilla actual. | |

***Carta***

|  |
| --- |
| **Nombre del Caso de Uso** |
| Comprar una carta, Vender una carta, Jugar una carta |
| **Descripción** | |
| La clase carta servirá para representar un efecto, cuyo objetivo puede ser tanto un jugador como una casilla. Las cartas que afecten a una casilla tendrán una referencia a la carta en concreto que haya sido jugada. Además de las manos de los jugadores, existirán en juego el mazo de cartas y el mazo de descartes. | |
| **Responsabilidades** | |
| Provocar efectos en casillas o jugadores | |
| **Atributos Propuestos** | |
| **PGS**: Cantidad de PGS que recibe o pierde el jugador que juegue la carta o, en el caso de cartas que cuyo objetivo sea un jugador, el jugador objetivo. | |
| **Métodos Propuestos** | |
| **Jugar carta**: Aplicar efecto de la carta a una casilla o jugador.  **Vender carta**: Descartar la carta de la mano del jugador a cambio de dinero. | |

## ASI 5: Definición de Interfaces de Usuario

En esta sección y las siguientes tiene prioridad los contenidos de la asignatura **“Comunicación Persona Máquina”** de segundo curso. No obstante, en caso de que **se esté diseñando una aplicación móvil**, para realizar la misma de forma lo más correcta posible se deberán tener en cuenta **los principios SOLID** impartidos en la asignatura “**Software para Dispositivos Móviles**”: <https://github.com/solid/solid-spec>. Nótese que el uso de estos principios afecta a más secciones que solo la definición de interfaces de usuario.

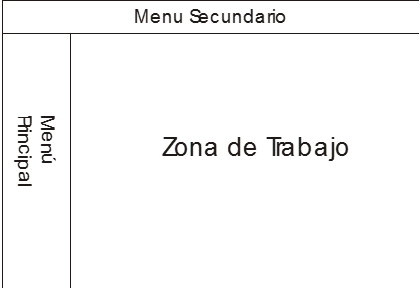
A la hora de diseñar un interfaz de usuario, debemos cumplir con las normas de comunicación persona-máquina existentes, procurando que el interfaz sea usable, permita manejar el programa de manera eficiente y que no sea propenso a provocar errores en los usuarios. Esto debe hacerse así porque los diseños obedecen al resultado de hacer un diseñado centrado en el usuario, que simplemente nos lleva a simular en la pantalla el trabajo que realiza sobre una mesa o, en general, su entorno de trabajo existente hasta el momento. Un enlace que puede ser de ayuda a la hora de tomar determinadas decisiones a la hora de construir el interface de la aplicación es el siguiente: <http://www.ambysoft.com/essays/userInterfaceDesign.html>.

### Descripción de la Interfaz

En esta sección debemos crear la especificación de las interfaces entre el usuario y el sistema a construir, incluyendo todos los diferentes tipos de pantallas que van a existir, los cuadros de diálogo o los informes que le proporcionarán al usuario.

En este apartado también es importante identificar posibles grupos de usuarios para así aplicar las pantallas a dichos grupos, así como detallar otros aspectos, como lo que vamos a incluir en las pantallas para cumplir con normas de accesibilidad y usabilidad.

Para los distintos tipos de pantallas, una opción es empezar por un esquema que muestre la disposición de estas, que permita identificar donde irá cada elemento y las diferentes zonas de trabajo. Se muestra un ejemplo con este dibujo, aunque lo más normal es usar un programa de wireframing como Balsamiq Mockups (<https://balsamiq.com/wireframes/>) o similar para hacer este tipo de esquemas de pantalla:



Boceto de una interfaz

### Definición del aspecto de la interfaz

**Otra posible opción es diseñar ya las pantallas definitivas sin funcionalidad**, solo para ver como quedarán en el producto final (es decir, crear un prototipo), lo que tiene la ventaja de poder enseñarle al cliente el aspecto de la aplicación desde un primer momento sin necesidad de mostrar los esquemas previos (esa sección podría omitirse en este caso).

Esta sección debe mostrar ya la interfaz definitiva de la aplicación (que evidentemente deberá ser una evolución del diseño mostrado en el apartado anterior) y las diferentes partes de las que consta, suponiendo que ya ha sido examinado, validado y aceptado por el cliente. Aunque luego durante el resto de la construcción del software se hagan retoques a la interfaz que se reflejen aquí si así fuese aconsejable, conviene tener claro el aspecto aproximadamente final del producto antes de construir su funcionalidad concreta, para lograr una mejor comprensión de las tareas a realizar y que el cliente sepa desde un primer momento lo que va a obtener, su aspecto y cómo se comporta (subsecciones siguientes).

Como es más que probable que todas las pantallas tengan elementos de interfaz comunes, esta es la sección donde se va a hablar de cada uno de esos elementos, su propósito y su función (barra de menús, barras de estado, etc.).

### Descripción del Comportamiento de la Interfaz

En este apartado debemos especificar cosas como los convenios que vamos a crear para validar la entrada de datos de la aplicación, los mensajes de error que mostraremos y el tipo de ayuda que vamos a proporcionar al usuario.

### Diagrama de Navegabilidad

En esta sección incluiremos un diagrama que muestre la navegación que habrá entre las pantallas del programa y su relación con las computaciones que tienen lugar en las mismas. Debemos mostrar solo las transiciones entre pantallas y no el contenido de cada pantalla en sí, ya que este diagrama tiene simbología especial para todos sus elementos. Podemos encontrar más información en:

* [www.agilemodeling.com/artifacts/uiFlowDiagram.htm](http://www.agilemodeling.com/artifacts/uiFlowDiagram.htm).

# Diseño del Sistema de Información

Fase de Desarrollo

DSI

Si se está usando como base/guía de contenidos las plantillas ARC42 de la asignatura “**Arquitectura del Software**”, la sección de diseño debería contemplar los siguientes elementos de estas plantillas por lo menos en alguna de sus subsecciones:

* Solution Strategy
  + Technological decisions and reaching the quality goals
  + Design Patterns
  + Organizational decisions
* Runtime View
* Deployment View
* Cross-cutting Concepts
* Design Decisions

## DSI 3: Diseño de Casos de Uso Reales

En esta sección y las siguientes tiene prioridad los contenidos de las asignaturas **“Arquitectura del Software”, “Ingeniería del Proceso Software”** y “**Diseño del Software**” de tercer curso. Además, en caso de que sea un proyecto con una parte web, deben aplicarse también los contenidos que se estimen oportuno de las asignaturas “**Software y Estándares para la Web**” y “**Sistemas Distribuidos e Internet**” de tercer curso.

Esta sección se usará, entre otras cosas, para evolucionar y detallar los diagramas de robustez que hemos desarrollado en el análisis usando diagramas de interacción y de estados. La estructura a seguir es la de incluir el diagrama en sí (dibujo) y luego hacer una lista explicando cada uno de los pasos existentes en dicho diagrama que lo requieran. Los diagramas deben incluir nombres de clases, métodos y parámetros “reales”, con la intención de que puedan trasladarse directamente a la implementación del sistema (el objetivo de un buen diseño). Por este motivo no debemos escatimar en detalles a la hora de desarrollarlos, ya que se supone que de estos diagramas podremos extraer directamente la implementación de la aplicación.

Cuando se menciona que los diagramas del diseño son una evolución de los del análisis significa que podemos aprovechar el trabajo ya realizado para su creación, adaptando la información en los diagramas correspondientes del análisis para desarrollar los nuevos diagramas (se debe lograr coherencia entre las entidades desarrolladas en ambas fases).

En cuanto al número de diagramas a contemplar, al igual que se dijo anteriormente no es necesario incluir en el diseño todos y cada uno de los posibles diagramas, sino que podremos ahorrar extender aquellos en los que:

* Las operaciones sean muy simples (en cuyo caso se puede optar por no decir nada al respecto o colocar simplemente un texto explicativo).
* La operación sea muy similar o idéntica a una que ya está desarrollada en esta sección. En este caso lo mejor es poner en el diagrama original a que otros casos se aplica o representa y no repetirlo.

### Caso de Uso 1.1

#### Diagramas de Interacción (Comunicación y Secuencia)

Diagramas de secuencia de diseño correspondiente al escenario principal del caso de uso y a aquellos alternativos que merezcan ser desarrollados (estableciendo un paralelismo con la nomenclatura usada en el análisis). Podemos encontrar más información en:  <http://pst.web.cern.ch/PST/HandBookWorkBook/Handbook/SoftwareEngineering/UCDOM_interaction.html>

En el caso de que creamos necesario mostrar el comportamiento de varios objetos que colaboran para la consecución de un determinado fin común, debemos crear un diagrama de comunicación de objetos que represente este comportamiento. Podemos encontrar más información acerca de los mismos en esta dirección: <http://www.agilemodeling.com/artifacts/communicationDiagram.htm>

#### Diagramas de Estados de las Clases

En caso necesario, si es conveniente dar más detalles de cómo se comporta un determinado escenario, podemos incluir diagramas de estados de las clases involucradas en el mismo para indicar las distintas fases por las que una clase pasará durante el mismo. Podemos encontrar más información sobre estos diagramas en:  <http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm>

#### Diagramas de Actividades

Si hay alguna operación o funcionalidad dentro del sistema que merezca la pena destacar se documentará mediante un diagrama de actividades. Puede haber tantos como consideréis necesarios. Más información acerca de estos diagramas se puede encontrar en estos enlaces:

* [http://dn.codegear.com/article/31863#activity-diagrams](http://dn.codegear.com/article/31863)
* <http://www.agilemodeling.com/artifacts/activityDiagram.htm>

### Caso de Uso 1.2

Misma estructura anterior, etc.

## DSI 4: Diseño de Clases

En esta sección representaremos diagramas que muestren los paquetes (y la relación existente entre ellos) y las clases que formarán parte de la implementación final del sistema (incluyendo también las relaciones existentes entre ellas), explicando todo aquel dato acerca de la utilidad de los mismos y justificando todo aquello que consideremos necesario.

* Debemos extraer la información de métodos y atributos a incluir en cada clase (así como las relaciones existentes entre ellas) de los diagramas que hemos desarrollado en el punto anterior.
* También debemos mostrar las clases adicionales que pueden ser necesarias en función de lo que hemos desarrollado anteriormente.
* Debemos mostrar todas las asociaciones y agregaciones (en general, relaciones) que necesitemos, en función de dichos diagramas.
* El conjunto de atributos de las clases debemos crearlo según los métodos, relaciones y en general de las necesidades de cada clase del sistema.
* La jerarquía de clases debemos pensarla de acuerdo a las necesidades de cada subsistema y aplicación.
* Debemos aclarar todos aquellos aspectos que no queden claros en el diagrama, usando notas.
* En general, debemos tener en cuenta que el diseño es una evolución del análisis, por lo que las clases que incluyamos en el diseño deben corresponderse razonablemente con todo el trabajo hecho en esa parte del documento.

### Diagrama de Clases

Es necesario mostrar por lo menos un diagrama global de clases. Si el diagrama de clases global fuese muy grande, podemos dividir el diagrama en varios diagramas más pequeños según la estructura de la aplicación, mostrando el principal abreviado tal y como se explica en los diagramas de paquetes. Otra opción es hacer que la hoja que tenga el diagrama se imprima en otro formato que nos proporcione más espacio (en horizontal, A3…).

## DSI 5: Diseño de la Arquitectura de Módulos del Sistema

### Diagramas de Paquetes

Aunque el concepto de paquete se puede aplicar a varios elementos del modelo, lo más típico en un TFG es agrupar clases con ellos. Los paquetes agruparán clases que estén relacionadas con una determinada funcionalidad y este diagrama debe mostrar también las relaciones existentes entre dichos paquetes (por ejemplo, dos paquetes estarán relacionados si algunas de sus clases se usan entre ellas o se envían mensajes). En esencia se trata de mostrar la organización lógica de la aplicación (en contraposición con la organización física, que aparecerá en los diagramas posteriores), presentando como se agrupan las clases de dicha aplicación en paquetes y la relación existente entre los mismos. Más información sobre estos diagramas se puede encontrar en:

* <http://www.agilemodeling.com/artifacts/packageDiagram.htm>
* <http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_packagediagram.html>

Los diagramas de paquetes son especialmente útiles cuando el diagrama de clases de todo el sistema es demasiado grande para visualizarse correctamente. Podemos entonces hacer un diagrama de paquetes (cada paquete contendrá N clases del sistema) y luego detallar individualmente cada uno de ellos donde corresponda.

#### Paquete 1

Descripción de los paquetes del diagrama (que tipo de elementos contiene, para qué sirven los mismos…).

#### Paquete 2

### Diagramas de Componentes

Los diagramas de componentes muestran los diferentes componentes de un sistema y sus dependencias. Un componente representa un módulo de código físico. Muchas veces se suele identificar un componente con un paquete, pero no siempre es así ya que los componentes representan el empaquetado físico del código. Por tanto, una misma clase puede estar presente en varios componentes, pero definida en un único paquete.

Es también una práctica común el incluir estos diagramas dentro de los de despliegue (que veremos a continuación) para que la distribución física de las distintas partes de la aplicación (y los componentes que las forman) en las distintas máquinas disponibles quede más clara.

Para tener más información sobre este tema, se pueden consultar los siguientes enlaces:

* <http://www.agilemodeling.com/artifacts/componentDiagram.htm>
* <http://www.visual-paradigm.com/VPGallery/diagrams/Component.html>
* <http://www.ibm.com/developerworks/rational/library/dec04/bell/>

 Si se integra con el diagrama siguiente (el de despliegue), en las descripciones que hagamos de los elementos debemos describir también los componentes. En caso contrario, describiremos los componentes en esta sección.

## DSI 10: Especificación Técnica del Plan de Pruebas

Para este proyecto se definirán y realizarán los siguientes tipos de pruebas:

* Pruebas unitarias
* Pruebas de Aceptación de las Features
* Pruebas de Usuario (Accesibilidad + Usabilidad)
* Pruebas de rendimiento

### Pruebas Unitarias

El objetivo de las pruebas unitarias del proyecto será verificar que las clases clave del proyecto definidas previamente en PLACEHOLDER (Jugador, Barco, Casilla) así como verificar que las funcionalidades clave del juego no dejan de funcionar al introducir nuevas features (por ejemplo, el lanzamiento del dado).

Las pruebas unitarias serán ejecutadas previa entrega de cada feature y deben ser todas superadas antes de subir el código al repositorio del proyecto. En el caso de que algún test falle, se buscará una explicación utilizando el depurador de Visual Studio, que permite acoplarse al proceso del juego y así depurar la aplicación y se realizarán las modificaciones necesarias al nuevo código de forma que se cumpla la feature y se superen las pruebas unitarias.

Además de ejecutar las pruebas, con cada nueva funcionalidad de la aplicación deberán incluirse nuevas pruebas para dichas funcionalidades, si se considerase necesario.

Para las pruebas unitarias se utilizará la herramienta **Test Runner,** la herramienta de pruebas integrada en el editor de Unity, la cual está basada en NUnit, el framework de pruebas más popular para .NET.

Las pruebas se implementarán en un único fichero .cs y serán de dos tipos:

1. Pruebas de las clases principales del juego: el objetivo de estas pruebas será comprobar que las clases fundamentales del juego funcionan correctamente y su inicialización al crear una nueva partida es adecuada. Se realizarán pruebas unitarias para las clases:
   1. Player
   2. Boat
   3. Tile
   4. Todas las clases que implementan ICard (es decir, las cartas)
2. Pruebas de funcionalidad: el objetivo de estas pruebas es asegurar que funcionalidades clave del juego funcionan correctamente, tales como la tirada del dado o la capacidad de un barco de moverse de casilla. Las funcionalidades consideradas fundamentales para una partida son las siguientes:
   1. Tirar el dado
   2. Mover un barco (solamente si te pertenece)
   3. Cambiar de turno
   4. Recibir una carta al sacar un 6
   5. Mover el barco de furtivos
   6. El barco de furtivos se mueve al sacar un 1
   7. Comprar cartas
   8. Comprar barcos
   9. Sobreexplotación de casillas

Se utilizará además el método **Setup** para que previa ejecución de cada uno de los tests, se cargue una nueva partida con dos jugadores. Esto supone una comprobación implícita del sistema de generación del tablero.

### Pruebas de Aceptación

El objetivo de las pruebas de aceptación será asegurar que lo implementado durante un sprint cumpla con los criterios de aceptación establecidos para las features planificadas para dicho sprint. Estas pruebas serán manuales y se realizarán todos los sprints del proyecto.

Las pruebas de aceptación tendrán dos partes:

1. Video demostrativo: Al completar una feature, se grabará un video de la nueva funcionalidad que demuestre que el estado actual del producto cumple los criterios de aceptación definidos en la feature al inicio del sprint.
2. Demostración en vivo: Además del video, se realizará una demostración en vivo al tutor del proyecto durante la reunión retrospectiva del sprint, explicando la implementación realizada y sus posibles decisiones tomadas. El tutor del proyecto validará la feature si cumple con lo esperado.

Además de estas pruebas cada sprint, se realizará una prueba en vivo con los clientes previa entrega del proyecto, aportando feedback sobre correcciones o posibles mejoras para implementar durante futuras iteraciones, pidiéndoles además que rellenen el formulario definido en PLACEHOLDER

### Pruebas de Usabilidad y Accesibilidad

Para las pruebas de usabilidad y accesibilidad se elegirán usuarios de diferentes edades y conocimientos tecnológicos y se les permitirá probar la aplicación jugando entre ellos en grupos de tamaño variable según la disponibilidad de los sujetos. La edad de los sujetos estará comprendida en el rango de entre 10 y 35 años, poniendo especial interés en el rango de entre los 18 y los 22 años, dado que fue el que obtenía mejores resultados en las pruebas realizadas con el juego de mesa físico.

Las pruebas de usabilidad y accesibilidad seguirán el siguiente procedimiento:

1. Prueba de la aplicación, jugar una partida con varios sujetos, guiándoles lo menos posible, de forma que tengamos el feedback más puro posible, especialmente sobre la usabilidad de la aplicación.
2. Rellenar un formulario de preguntas rápidas, este formulario nos dará una visión general sobre el perfil del usuario, así como su opinión general del juego.
3. Recolectar sugerencias y posibles mejoras de los usuarios.

El cuestionario diseñado para las pruebas es el siguiente:

Todas las preguntas se responden en una escala de 1 a 5, donde 1 es muy en desacuerdo y 5 es muy de acuerdo.

1. Tengo experiencia con tecnología en general y me considero hábil tratando con ella
2. Tengo experiencia jugando a videojuegos y me considero hábil en ellos
3. Prefiero un videojuego antes que un juego de mesa físico

Estas tres preguntas iniciales nos darán el perfil del usuario con el que estamos tratando.

1. Los controles me han resultado intuitivos
2. No he tenido problemas para saber lo que puedo/tengo que hacer

Estas dos preguntas nos permitirán refinar los controles del juego y aumentar el feedback que se le da al jugador en los puntos más problemáticos.

1. La estética del juego me ha parecido agradable

Si bien no es tan importante como la funcionalidad, un juego bonito tiene mayor aceptación entre los jugadores.

1. Pienso que el juego es muy largo o se me ha hecho monótono

Si el jugador tiene la sensación de que una partida es muy larga puede llegar a arruinar su experiencia, para evitar esto habría que acelerar el ritmo del juego, ya sea físicamente cambiando las animaciones o realizando modificaciones/equilibrados a distintas reglas. P.E. bajar los precios, cambiar el número de cartas de un tipo…

1. En general mi experiencia ha sido buena
2. Recomendaría este juego a mis amigos

Estas últimas preguntas son de un tono más general, para conocer la opinión del usuario sobre el juego en sí, como un todo, porque puede ocurrir que el juego tenga alguna parte peor que las demás, por ejemplo, unos malos controles, y sin embargo, no afectar a la experiencia general del usuario. Por tanto, es importante comparar estas últimas preguntas con las anteriores y ver en qué medida afectan los componentes concretos a la totalidad del juego.

### Pruebas de Rendimiento

Debido al alcance establecido para esta iteración del proyecto, las pruebas de rendimiento tienen poco peso, dado que, con el estado esperado de la aplicación en el momento de la entrega, se ejecutará únicamente una partida por equipo, con un máximo de 4 jugadores por partida, lo cual no supondrá un consumo de recursos significante.

Para esta iteración se realizarán mediciones del uso de recursos en un sistema ejecutando una partida del juego (CPU, memoria, consumo de GPU). A partir de estas mediciones es sacarán conclusiones y se buscarán mejoras que permitan optimizar el juego si fuera necesario.

En iteraciones posteriores del proyecto, en las que el juego soporte partidas multijugador desde diferentes dispositivos, lo ideal sería realizar pruebas de rendimiento sobre la máquina servidor que aloje las diferentes partidas, probando los límites soportados. Por ejemplo, el número máximo de partidas que se pueden alojar en el servidor sin dañar la experiencia del jugador. En un juego de este tipo, que no requiere de tiempos de reacción excesivamente rápidos, podemos suponer que el límite sería una latencia de hasta 150 ms.

Para realizar estas pruebas utilizaremos la herramienta **MSI Afterburner,** ya que nos ofrece toda la información que necesitaremos (uso de CPU, RAM, GPU y fotogramas por segundo actuales). Para un examen más exhaustivo del rendimiento de la aplicación, se podría utilizar el **Unity Profiler,** la herramienta propia de Unity para realizar pruebas de rendimiento.

# Construcción del Sistema de Información

Fase de Desarrollo

CSI

## CSI 1: Preparación del Entorno de Generación y Construcción

En esta sección se describen una serie de aspectos relativos a la implementación del software desarrollado.

### Lenguajes de programación

Como se menciona anteriormente en PLACEHOLDER, para el proyecto se va a utilizar el motor gráfico Unity, por lo que el único lenguaje de programación a utilizar será C#.

En el caso de los ficheros utilizados para lectura/escritura de la información, se utilizarán ficheros de texto .txt con formato variable dependiendo de la finalidad deseada, dichos formatos son los siguientes:

1. Conteo de tipos de casillas. El formato será NombreCasilla;RatioAparición;Recursos, donde RatioAparición es el número de casillas de un tipo sobre un total de 1, es decir, si una casilla tiene una ratio de 0.2, en un tablero de 10 casillas aparecerá dos veces. El origen de esto es el requisito PLACEHOLDER.
2. Conteo de tipos de cartas, así como sus nombres y efectos. El formato será NombreCarta;PGS;NumeroCartas, donde PGS es la cantidad de PGS que se le da al jugador que usa la carta (pueden ser positivos, negativos o cero) y NumeroCartas es la cantidad de cartas de ese tipo que habrá en la baraja.
3. Traducciones. El formato será CLAVE=TRADUCCIÓN.

### Herramientas y programas usados para el desarrollo

Para el desarrollo de este proyecto se han utilizado las siguientes herramientas (se incluye enlace a la página web de la herramienta):

1. [Unity 2019.2.11f1](https://store.unity.com). Desarrollo principal del juego, gestión de recursos y de las escenas, así como el uso de herramientas del editor durante otras fases del desarrollo.
2. [Microsoft Visual Studio Enterprise 2019 16.4.4](https://visualstudio.microsoft.com/es/). Utilizado para la creación y modificación de los ficheros de código C# del proyecto, así como para gestionar el control de versiones.
3. [GitHub](https://github.com/rimorD). Utilizado para control de versiones del proyecto.
4. [Notepad++ v7.8.5](https://notepad-plus-plus.org). Utilizado para creación y modificación de los ficheros de texto del proyecto.

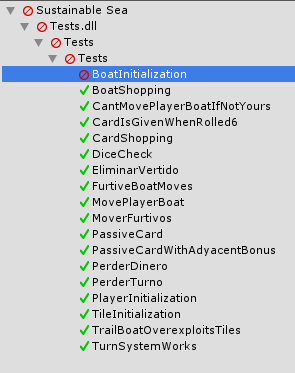
## CSI 2: Generación del Código de los Componentes y Procedimientos

En esta sección debemos describir **las clases más relevantes** de nuestra aplicación de manera que se detallen todos sus atributos y métodos, así como su descripción y función exacta, tal y como aparecen en el código de la aplicación. Para hacerlo lo mejor posible debemos elaborar esta descripción una vez cerrado el desarrollo de la aplicación en sí, para evitar problemas de versiones por revisiones al código que surjan durante el desarrollo. Para ello se recomienda emplear una herramienta que vuelque las clases y sus comentarios a un formato que podamos usar en este documento y que contemple todos los aspectos vistos antes (es decir, que el resultado sea similar a la tabla mostrada más abajo). Herramientas como Javadoc o similares son adecuadas para este fin.

Debe tenerse en cuenta que aquí solo se incluirán las clases más importantes, para el resto pueden referenciarse los anexos o repositorios tipo GitHub que se hayan usado para el desarrollo…

## CSI 3: Ejecución de las Pruebas Unitarias

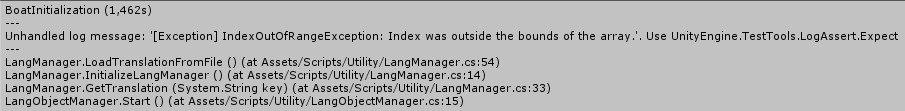
Para este documento se utilizará únicamente la última ejecución de las pruebas unitarias, es decir, con el juego listo para la entrega de la primera versión y previas las pruebas de aceptación, tanto del sprint como de los clientes.



Primera ejecución de las pruebas

Tras ejecutar las pruebas por primera vez, vemos que superan todos los tests excepto **BoatInitialization** que se encarga de probar los atributos y métodos de la clase Boat, así como su correcta inicialización.

Vemos que durante la ejecución del test ha ocurrido un IndexOutOfRangeException.



Excepción generada en el test BoatInitialization

Utilizando el depurador de Visual Studio, colocamos breakpoints en las líneas anteriores al error y comprobamos el estado de las variables implicadas. El error ocurría debido a que el fichero con las traducciones estaba incorrectamente formado al tener una línea en blanco, de forma que daba error al procesarla. Se corrige el fichero de traducciones y se vuelven a ejecutar las pruebas.



Ejecución de las pruebas tras la corrección

Tras esta ligera corrección, todas las pruebas son superadas y podemos continuar a las pruebas de aceptación.

## CSI 4: Ejecución de las Pruebas de Aceptación

En esta sección hablaremos únicamente de las pruebas de aceptación realizadas por los clientes, los videos demostrativos de cada feature se añaden al proyecto como PLACEHOLDER y las revisiones del tutor en la reunión de revisión del sprint se pueden encontrar en las actas de las reuniones en PLACEHOLDER.

**Pruebas de Aceptación realizadas por los clientes**

Utilizando Microsoft Teams realizamos una conferencia para realizar una prueba en directo con los clientes. A dicha reunión acuden el desarrollador del proyecto (David Ferreiro), uno de los tutores (Daniel Fernández) y Marina Parrondo y Fernando Rayón por parte de cliente.

Se realiza una partida del juego de dos jugadores, uno para cada cliente y se van probando una a una las funcionalidades. Recibimos el siguiente feedback:

* En Linux no va bien, los botones y menús no se traducen, en un principio se cree que puede ser porque no se tenga instalada la fuente Arial, que es la usada por el juego. Esto puede corregirse incluyendo la fuente en el proyecto.
* En el juego de mesa los PGS no pueden ser negativos. Se tuvo una discusión sobre esto, dado que si bien es cierto en el juego de mesa no era así, realmente tiene bastante sentido, esto queda a la espera de ver si se convierte en un requisito.
* La carta ciclogénesis ha sido implementada incorrectamente, se ha implementado como perder el turno, cuando lo que debería hacer es devolver a todos los barcos de una casilla a la casilla de salida.
* Es difícil saber de quién es el turno cuando estás en la vista de tablero, habría que marcar el jugador actual y sus barcos de alguna forma.
* Al sacar un 6 la carta que robas debe jugarse siempre que sea posible, no únicamente las cartas que se juegan sobre casillas.
* Las fichas azules son difíciles de ver con las casillas azules, especialmente en casillas de interior.
* La casilla inicial en el juego de tablero no da recursos. Se podría dejar como sobreexplotada en el videojuego para que se distinga. En un futuro podría añadirse más elementos visuales y hacer que parezca un puerto.
* Girar el tablero para que las casillas de costa estén hacia abajo.
* Extender la funcionalidad del zoom también a las cartas.
* Avisar de que se puede realizar zoom.
* Añadir alguna ayuda visual a la casilla en la que se encuentran los furtivos, esto es especialmente útil para las casillas de las esquinas, dado que comparten posición para los furtivos.
* En el caso de las cartas que afectan a casillas adyacentes, distinguir la carta original de la carta que se juega en las adyacentes como ayuda visual.
* Los barcos arrastreros no recolectan recursos siempre, deben preguntar primero al usuario si quieren hacerlo.
* Si se sobreexplotasen dos tercios del tablero, la partida se termina y ningún jugador gana.
* Añadir un menú de pausa que permita salir del juego y acceder a las opciones si las hubiera.
* Permitir el guardado y el cargado de partidas.
* El juego se ha traducido al inglés, sin embargo, las cartas están únicamente en español. Los clientes nos proporcionarán las cartas en inglés y deberemos añadirlas.

Estas sugerencias se han tenido en cuenta, junto al feedback de los jugadores de las pruebas de usuario para los requisitos de futuras iteraciones del proyecto, como se ha mencionado en PLACEHOLDER.

## CSI 5: Ejecución de las Pruebas de Usuario

Se han realizado las pruebas de usuario definidas en PLACEHOLDER con 7 personas, divididas en dos grupos de dos personas y un grupo de tres personas. La media de edad de los participantes es de **22 años,** siendo el menor de todos de **10 años** y el mayor de todos de **35 años.**

**Resultados de los Cuestionarios**

A continuación, se muestran las preguntas del formulario previamente definido en PLACEHOLDER, junto a la media de las respuestas que los usuarios han dado:

|  |  |
| --- | --- |
| **Pregunta** | **Media de las Respuestas** |
| **Tengo experiencia con tecnología en general y me considero hábil tratando con ella** | **5** |
| **Tengo experiencia jugando a videojuegos y me considero hábil en ellos** | **4.5** |
| **Prefiero un videojuego antes que un juego de mesa físico** | **3.3** |
| **Los controles me han resultado intuitivos** | **3.6** |
| **No he tenido problemas para saber lo que puedo/tengo que hacer** | **3** |
| **La estetica del juego me ha parecido agradable** | **4** |
| **Pienso que el juego es muy largo o se me ha hecho monótono** | **3.4** |
| **En general mi experiencia ha sido buena** | **4.1** |
| **Recomendaría este juego a mis amigos** | **3.6** |

Checklist de pruebas de Usabilidad

Como podemos ver nuestro mayor problema se encuentra en que no le damos suficiente información al usuario, sin embargo, en general el juego provoca una experiencia agradable a los usuarios.

**Sugerencias de los usuarios**

Los usuarios que han probado el juego han sugerido los siguientes cambios/posibles mejoras:

* En el caso de las cartas que afectan a casillas adyacentes, distinguir la carta de las casillas adyacentes de la carta original. Esta sugerencia fue mencionada también por los clientes.
* Añadir más instrucciones al uso de cartas, no está claro cómo se juega una carta. Esta sugerencia se ha repetido en varios jugadores.
* Incluir un tutorial o explicación rápida de cómo jugar al inicio.
* No utilizar fichas azules, dado que todo el tablero es azul prácticamente. Esta sugerencia fue mencionada también por los clientes.
* En la vista del tablero, actualizar la puntuación/dinero de los jugadores en tiempo real.
* Indicar que los furtivos son una característica, es confuso que haya un barco en mitad del tablero porque sí y no se mencione en ningún momento.
* El juego no tiene música ni efectos de sonido.
* Las animaciones son muy “forzadas”.
* Los turnos siempre siguen el mismo orden, incluso entre diferentes partidas, podría estar bien que se aleatorizara el orden de los turnos al inicio de la partida.

Estas sugerencias se han tenido en cuenta, junto al feedback de los clientes para los requisitos de futuras iteraciones del proyecto, como se ha mencionado en PLACEHOLDER.

## CSI 6: Ejecución y Conclusiones de las Pruebas de Rendimiento

### Especificaciones del sistema de pruebas

CPU: Intel(R) Core(TM) i5-6600 CPU @ 3.30GHz

RAM: 16 GB DDR4 2133 MHz

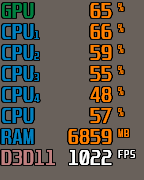
GPU: GeForce GTX 1660 SUPER

SO: Windows 10 Pro 64 bits

### Resultados

Se ejecuta el juego con los gráficos a la mayor calidad permitida por Unity, con resolución de 1920 x 1080 y se crea una nueva partida con 4 jugadores.

En la esquina superior izquierda de la pantalla podemos ver la información proporcionada por el MSI Afterburner.



**Información proporcionada por MSI Afterburner**



**Información proporcionada por Windows Task Manager**

Podemos ver como el juego está consumiendo entre un 60-70%, tanto de la GPU como de la CPU. La RAM en uso mostrada por el Afterburner es la RAM total ocupada en el equipo, no únicamente por el juego, en el administrador de tareas de Windows podemos ver que la cantidad de memoria ocupada por el juego es insignificante (97 MB).

Sin embargo, el juego está consumiendo una cantidad inaceptable de CPU y GPU. Podemos ver como el juego está corriendo a **1022 fotogramas por segundo.** Esta cantidad es innecesariamente alta, dado que la mayoría de las pantallas muestran como máximo unos 60 FPS, como es el caso del monitor del sistema de pruebas. Podemos suponer entonces que el juego está solicitando tantos recursos para lograr la mayor ratio de FPS posible.

### Conclusiones

Tras analizar los resultados de la prueba y realizar una ligera investigación sobre medidas de optimización para juegos en Unity, se proponen las siguientes optimizaciones para la siguiente iteración del proyecto:

1. Limitar los FPS del juego. Para esto se utiliza **Application.targetFrameRate**. Si no está definido, Unity por defecto trata de conseguir el máximo de fps posibles para la plataforma, esto supone un consumo innecesario de recursos para un juego de estas características, tal y como hemos observado durante las pruebas. Normalmente los juegos suelen tener un objetivo de unos 60 FPS, pero en nuestro caso no importa mucho, dado que las animaciones no son tan complejas. Con 30 FPS los jugadores no notarían diferencia alguna y se lograría un mejor rendimiento en equipos de menor calibre.
2. Marcar objetos de la escena que no se muevan como estáticos, como las casillas. Las casillas no han sido marcadas como objetos estáticos de Unity, lo que quiere decir que se está gastando procesamiento en cálculos de movimientos de objetos que no van a moverse nunca.
3. Crear un único material que tenga todas las texturas necesarias del juego. Es posible crear un único material con una única textura que abarque todas las texturas necesarias por el juego y luego mapear la posición de la textura que queremos en cada objeto, de esta forma se alivia la carga de materiales, que es uno de los procesos más costosos y se reduce el tamaño del juego.
4. Examinar el código con el Unity Profiler. Utilizando el Profiler podemos observar qué piezas del código consumen más tiempo y tratar de redefinirlas con un algoritmo mejor.

Estas modificaciones se incluyen como requisitos para siguientes iteraciones junto con el feedback de los clientes y de las pruebas de usuario, tal y como se indica en PLACEHOLDER.

## CSI 7: Elaboración de los Manuales de Usuario

Esta aplicación no requiere instalación alguna y para su ejecución simplemente será necesario correr su ejecutable, por lo que no se definirán ni manual de instalación ni manual de arranque.

A continuación, se definen los manuales de usuario y del programador.

### Manual de Usuario

El manual de usuario es algo muy importante debido a que es el documento que servirá a los usuarios de nuestro sistema para saber cómo funciona cada una de las partes de nuestra aplicación. Debemos pues describir cómo funcionan todas las opciones de la misma, que parámetros tiene, que cosas debemos hacer para que todas las operaciones funcionen correctamente y cualquier otro aspecto que consideremos oportuno para explicar el funcionamiento del sistema.

No debemos escatimar detalles en este manual ya que es la herramienta para que los usuarios comprendan nuestro sistema. También debemos hacer el mayor uso posible de capturas de pantalla para mejorar nuestras explicaciones.

### Manual del Programador

En este manual debemos describir cualquier aspecto que pueda ayudar a otros programadores a ampliar, modificar o entender aspectos de la construcción de nuestra aplicación. Debemos por tanto hacer una descripción general de los distintos aspectos involucrados en la construcción del sistema que puedan ser más difíciles de entender y también describir los procedimientos necesarios para hacer ciertas ampliaciones que hayamos contemplado en el diseño del sistema (añadir nuevas entidades, nuevos atributos a entidades existentes, nuevos servicios que usen a los ya desarrollados, modificaciones en la interfaz, etc.).

# Apéndices

## Presupuesto

Aquí se especificará el presupuesto del desarrollo del software tal y como se ha enseñado a realizar en la asignatura “**Dirección y Planificación de Proyectos Informáticos**” de cuarto curso.

## Problemas Encontrados y Decisiones tomadas Durante el Desarrollo

En esta sección se enumeran los diferentes problemas y/o decisiones que han sido tomadas a lo largo de la fase de desarrollo. Esta sección ha sido acomodada a la documentación a partir de la página de la Wiki del proyecto homónima.

### Generación dinámica del tablero

Aumenta la jugabilidad, dado que cada partida el tablero será diferente y, además, permite añadir nuevos tipos de casillas de una manera simple, habría que crear un nuevo prefabricado para dicha casilla y modificar los conteos del fichero de texto para que comience a generarla. Si se hubiera creado un tablero estático, para introducir un nuevo tipo de casilla, habría que modificar el tablero manualmente, desplazando las casillas existentes, eliminándolas, creando otras nuevas, etc...

### Tamaño variable del mapa

Aunque el tamaño del tablero será de 40 casillas, existe la posibilidad de crear tableros más grandes, sin embargo, el tamaño deberá ser múltiplo de 40 para que encajen los conteos de los tipos de casillas (80, 120...)

### Dados con físicas vs generador de números aleatorios

Ambas formas de "lanzar dados" tienen sus pros y sus contras, los dados con físicas son muy visuales, dado que directamente se crea un modelo de dado y se lanza al tablero, lo cual es atractivo y realista, pero, por otro lado, es complicado obtener el resultado y puede suponer un descenso del rendimiento, sobre todo si se utilizan varios dados.

El generador de aleatorios es una estrategia comúnmente usada, dado que su efecto en el rendimiento es menor y es simple de implementar, su defecto es que para que sea tan atractivo como los dados con físicas hay que dedicar un gran esfuerzo. Ciertas compañías de juegos móviles utilizan animaciones predefinidas, en las cuales se sabe en qué cara va a terminar el dado, y modifican la textura del dado para que muestre la cara con el número que ha sido generado aleatoriamente.

En nuestro caso existirá un único dado, por lo que se ha decidido implementar la solución de dados con físicas, dada su mayor simplicidad a la hora de crear atractivo. Para obtener el resultado del dado, se realizará un cálculo del producto de los ejes X e Y del dado, es decir, un producto de dos vectores, con el resultado de este producto y su signo, sabremos qué cara es el resultado.

### Mostrar/Ocultar vs Crear/Destruir el dado

Mientras no se está tirando el dado, tenerlo en el tablero solamente creará problemas (dificultades visuales, colisiones inesperadas, etc.) por lo que se considera necesario encargarse del dado mientras no se está usando. Esto puede hacerse de dos formas, moverlo a la posición de origen de la tirada, ocultarlo y mostrarlo cuando se vaya a tirar o destruirlo e instanciar un nuevo dado cuando se tire.

Crear/destruir es simple de implementar, sin embargo, puede causar problemas de memoria dado que estaríamos condicionados por el recolector de basura de Unity, el cual no tenemos certeza de cuando se ejecutará.

Por otro lado, mostrar/ocultar no supone ninguna desventaja aparente, dado que únicamente tendremos un dado, por lo que tenerlo cargado todo el tiempo no debería influir demasiado en el rendimiento, menos aun deshabilitando su renderer (haciéndolo así invisible) y no debería provocar colisiones siempre y cuando la posición de origen de la tirada esté en un lugar adecuado.

### Cartas que afectan a casillas adyacentes

En un principio se planteaba tener una única carta en la casilla en la que se hubiera jugado y que las casillas adyacentes, a la hora de recolectar recursos, comprobaran si están bajo la influencia de una carta. Debido a un bug desconocido por el cual a la hora de la comprobación se consideraba la carta como nula, finalmente se ha implementado de la siguiente forma: Al jugar una carta que afecta a las casillas adyacentes, se crean nuevas cartas para dichas casillas, las cuales tienen el efecto correspondiente. Estas cartas no dan PGS, tienen la misma imagen que la carta original y se dibujarán bajo la casilla como si se hubieran jugado en ella.

### Animación de los barcos

En lugar de sacar los barcos fuera de la fila de casillas como se tenía pensado en los documentos de diseño iniciales, se ha decidido elevarlos en el aire, dado que, si los arrastrásemos fuera de la casilla, seguiríamos teniendo el problema de que atravesaría otros barcos que pudiera haber en la casilla.

### I18N

El idioma por defecto del juego es el español, dado que es el idioma materno de tanto los clientes, como los desarrolladores. El juego se traduce además al inglés, dado que aumenta mucho el valor del producto final respecto al esfuerzo necesario para la implementación. Teniendo español e inglés disponibles, el juego está disponible para la mayor parte del mercado. La implementación realizada permite que se añadan nuevos idiomas al juego en el futuro de forma simple:

1. Incluir un nuevo fichero strings.LANGCODE.txt con las traducciones del juego
2. Incluir el código del nuevo idioma aceptado en la clave ACCEPTED\_LANGUAGES de la clase Definitions
3. Incluir en el menú principal del juego el botón correspondiente al nuevo idioma

## Conclusiones

Conclusiones del sistema: Qué hemos elaborado, si los resultados están dentro de lo esperado, si hemos cumplido las expectativas, justificación de haber escogido las mejores opciones para cada uno de los aspectos del sistema, etc.

## Ampliaciones

Como resultado de la ejecución de las pruebas de aceptación con los clientes PLACEHOLDER, las pruebas de usuario PLACEHOLDER y las pruebas de rendimiento PLACEHOLDER, se crean los siguientes requisitos que serán considerados para siguientes iteraciones del proyecto, junto a los requisitos que se han dejado fuera del alcance en PLACEHOLDER:

### Requisitos para futuras iteraciones

Los siguientes requisitos han quedado fuera de la planificación y serán implementados en futuras iteraciones del proyecto.

#### Requisitos no incluidos en la planificación

##### Requisitos del Sistema de partidas

|  |  |
| --- | --- |
| Código | Descripción del Requisito |
| AUD1 | El sistema incorporará un gestor de audio. |
| AUD1.1 | El gestor de audio incluirá música de ambiente |
| AUD1.1.1 | En el menú principal. |
| AUD1.1.2 | En la creación de una partida. |
| AUD1.1.3 | Durante la partida. |
| AUD1.2 | El gestor de audio incluirá efectos de sonido durante ciertas acciones. |
|  |  |
|  |  |
|  |  |

* + 1. El sistema permitirá definir reglas de la partida al creador.
       1. El creador podrá definir la condición de fin de partida entre las siguientes:
          1. Número determinado de rondas.

El número de rondas mínimo será **10.**

El número máximo de rondas será **50.**

* + - * 1. Hasta que se terminen las cartas.
        2. Número determinado de puntos de gestión sostenible.

El número de puntos mínimo será **15.**

El número de puntos máximo será **30.**

* + - * 1. Cuando la condición establecida se cumpla, el sistema terminará la partida como se define en **1.2.3**
        2. El sistema solamente permitirá elegir una condición de fin de partida.

Si el creador elige una condición de fin de partida teniendo otra previamente seleccionada, el sistema se quedará con la nueva.

* + - 1. El jugador podrá añadir jugadores a la partida.
         1. Jugadores controlados por la máquina.
         2. Jugadores humanos (único dispositivo).
         3. El sistema asignará a cada jugador, según su origen (Humano o máquina):

Un nombre.

En caso de jugadores humanos, este nombre podrá ser modificado.

Este nombre será mostrado durante la partida en el HUD.

Un avatar.

En caso de jugadores humanos, se podrá elegir entre una serie de avatares.

Este avatar será mostrado durante la partida en el HUD.

Un color.

En caso de jugadores humanos, el color podrá ser modificado.

Este color se utilizará durante la partida.

El sistema mostrará el color para cada jugador en el HUD.

El sistema coloreará los barcos de cada jugador con su correspondiente color.

* + - * 1. El creador no podrá añadir más jugadores que el número de jugadores definido en la partida **1.1.1.2**
      1. El creador podrá eliminar jugadores que previamente haya añadido a la partida.
      2. El sistema requerirá para iniciar la partida que:
         1. La condición de fin de partida haya sido definida.
         2. El número de jugadores haya sido definido.
         3. El número de jugadores coincida con el definido.
         4. Al menos **uno** de los jugadores sea humano.
      3. Cada jugador tirará un dado.
         1. El resultado de las tiradas, de mayor a menor, indicará el orden de los turnos de los jugadores en cada ronda.

Número de cartas en su mano.

Un indicador en caso de que sea su turno.

Los barcos de cada tipo que posee.

En caso de que el jugador solo tenga un barco, no será necesario que lo elija.

Los puntos de gestión sostenible de cada jugador.

La tabla de puntuación estará ordenada por los puntos de gestión sostenible de cada jugador de mayor a menor.

Los créditos de cada jugador.

El número de barcos de cada tipo de cada jugador.

El número de cartas totales obtenidas por cada jugador.

El número de casillas sobreexplotadas por cada jugador.

##### Requisitos de audio

|  |  |
| --- | --- |
| Código | Descripción del Requisito |
| AUD1 | El sistema incorporará un gestor de audio. |
| AUD1.1 | El gestor de audio incluirá música de ambiente |
| AUD1.1.1 | En el menú principal. |
| AUD1.1.2 | En la creación de una partida. |
| AUD1.1.3 | Durante la partida. |
| AUD1.2 | El gestor de audio incluirá efectos de sonido durante ciertas acciones. |
|  |  |

##### Requisitos multijugador

|  |  |
| --- | --- |
| Código | Descripción del Requisito |
| MP1 | Se mostrará una lista con las partidas multijugador disponibles. |
| MP1.1 | Se mostrará la siguiente información de las partidas: |
| MP1.1.1 | Nombre del creador de la partida. |
| MP1.1.2 | Número actual de jugadores. |
| MP1.1.3 | Número máximo de jugadores. |
| MP1.1.4 | Si está protegida por contraseña. |
| MP1.2 | El creador podrá definir si su partida acepta jugadores externos. |
| MP1.2.1 | Un jugador podrá unirse a una partida que acepte jugadores externos. |
| MP1.2.2 | El creador podrá definir una contraseña para su partida. |
| MP1.2.2.1 | La contraseña será de tipo texto. |
| MP1.2.2.1.1 | La contraseña no tiene longitud mínima. |
| MP1.2.2.1.2 | La longitud máxima será de 16 caracteres. |
| MP1.2.2.2 | Los jugadores que quieran unirse deberán introducir la contraseña. |
| MP1.2.2.2.1 | En caso de que la contraseña coincida con la establecida por el creador, se unirá a la partida. |
| MP1.2.2.2.2 | * En caso contrario:   + El sistema mostrará un mensaje indicando que la contraseña introducida no es correcta.   + El jugador no se unirá a la partida. |
|  |  |

#### Nuevos requisitos surgidos de las pruebas

La siguiente lista de requisitos ha surgido a partir de las pruebas realizadas sobre el sistema final obtenido en esta iteración del proyecto y se incluirán en la planificación de futuras iteraciones junto los requisitos del apartado anterior.

## Referencias Bibliográficas

La siguiente guía explica como incluir referencias bibliográficas correctamente en Word y es la que se recomienda usar (y se ha usado en este documento): <https://support.office.com/es-es/article/agregar-citas-en-un-documento-de-word-ab9322bb-a8d3-47f4-80c8-63c06779f127>

La referencia a esta plantilla en ResearchGate (la primera de todas) **debe dejarse en el documento tal cual aparece a continuación**:

|  |  |
| --- | --- |
| [1] | J. M. Redondo, «Documentos-modelo para Trabajos de Fin de Grado/Master de la Escuela de Informática de Oviedo,» 17 6 2019. [En línea]. Available: https://www.researchgate.net/publication/327882831\_Plantilla\_de\_Proyectos\_de\_Fin\_de\_Carrera\_de\_la\_Escuela\_de\_Informatica\_de\_Oviedo. |
| [2] | J. M. Redondo y F. Ortín, «A SaaS framework for credit risk analysis services,» *IEEE Latin America Transactions,* vol. 15, nº 3, pp. 474-481, 2017. |
| [3] | J. M. Redondo y L. Varela, «Filesync and Era Literaria. Realistic Open Source Webs To Develop Web Security Skills,» *Journal of Web Engineering,* vol. 17, nº 5, pp. 1-22, 2018. |
| [4] | O. Zaleski, M. Navarro, S. Díaz, J. M. Redondo y M. Labrador, «Clinical Gait Assessment Comparison: Smartphone-based versus Inertial Measurements Units,» de *IEEE Southeast Con*, 2018. |
| [5] | A. M. Fernández, B. M. González, J. M. Redondo y A. J. Sánchez, «Tirsus IV: Navegación Multidimensional en Aplicaciones Hipermedia sobre Acontecimientos Históricos,» de *Interacción 2000*, Granada, 2000. |
| [6] | A. J. Sánchez, A. M. Fernández, B. M. González y J. M. Redondo, «Tirsus II, Aplicación de Hipermedia para la Enseñanza de la Historia,» de *Interacción 2000*, Granada, 2000. |
| [7] | J. M. Redondo, A. J. Sánchez, A. M. Fernández y B. M. González, «Cinemedia Astur: Herramienta Generadora de Títulos Hipermedia de Interfaz Flexible,» de *Interacción 2000*, Granada, 2000. |
| [8] | I. Lagartos, J. M. Redondo y F. Ortín, «Efficient Runtime Metaprogramming Services for Java,» *Journal of Systems and Software,* vol. 153, pp. 220-237, 2019. |
| [9] | J. M. Redondo, «FanCines: Understand C Sharp concepts the easy way,» 1 5 2019. [En línea]. Available: https://www.researchgate.net/publication/333104441\_FanCines\_Understand\_C\_Sharp\_concepts\_the\_easy\_way. |
| [10] | J. M. Redondo, «Admin-zines: Understand Infrastructure Administration concepts the easy way,» 8 2019. [En línea]. Available: https://www.researchgate.net/publication/335023411\_Admin-zines\_Understand\_Infrastructure\_Administration\_concepts\_the\_easy\_way. |
| [11] | F. Ortin, J. M. Redondo y J. Quiroga, «Design and evaluation of an alternative programming paradigms course,» *Telematics and Informatics,* vol. 34, nº 6, pp. 813-823, 2017. |
| [12] | F. Ortin, D. Zapico y J. M. Redondo, «Utilización de patrones de diseño en la asignatura de Procesadores de Lenguaje,» de *XV Jornadas de Enseñanza Universitaria de la Informática*, Barecelona, 2009. |
| [13] | J. M. Redondo and F. Ortin, "A comprehensive evaluation of common python implementations," *IEEE Software,* vol. 32, no. 4, pp. 76-84, 2014. |
| [14] | F. Ortin, M. García and J. M. Q. J. Redondo, "Combining static and dynamic typing to achieve multiple dispatch," *Information–An International Interdisciplinary Journal,* vol. 16, no. 12, pp. 8731-8750, 2013. |
| [15] | J. M. Redondo, «New Features of C Sharp 8 and beyond,» 1 2019. [En línea]. Available: https://www.researchgate.net/publication/330514620\_New\_Features\_of\_C\_Sharp\_8\_and\_beyond. |
| [16] | J. M. Redondo, «New Features of C Sharp 7,» 1 2019. [En línea]. Available: https://www.researchgate.net/publication/330358763\_New\_Features\_of\_C\_Sharp\_7. |
| [17] | J. M. Redondo, «New Features of C Sharp 5 and 6,» 1 2019. [En línea]. Available: https://www.researchgate.net/publication/330223681\_New\_Features\_of\_C\_Sharp\_5\_and\_6. |
| [18] | H. Montero, «No solo usabilidad: revista sobre personas, diseño y tecnología,» 30 3 2003. [En línea]. Available: http://www.nosolousabilidad.com/articulos/heuristica.htm. |
| [19] | J. M. Redondo, Introducción Práctica a la Administración Segura de Servidores Apache Bajo Linux, Oviedo, Asturias: Servicio de Publicaciones de la Universidad de Oviedo, 2019. |
| [20] | J. M. Redondo, «Improving Student Assessment of a Server Administration Course Promoting Flexibility and Competitiveness,» *IEEE Transactions on Education,* vol. 62, nº 1, pp. 1-8, 2019. |
| [21] | J. M. Redondo, "Are You Breathacking? Introduction to hacking with a practical example," 1 7 2019. [Online]. Available: https://www.researchgate.net/publication/334303691\_Are\_You\_Breath-hacking\_Introduccion\_en\_Espanol\_al\_mundo\_del\_hacking\_con\_un\_ejemplo\_practico. |
| [22] | J. M. Redondo and D. Cuesta, "Towards Improving Productivity in NMap Security Audits," *Journal of Web Engineering,* vol. 18, no. 7, pp. 1-38, 2019. |
| [23] | I. Llaneza, J. M. Redondo y L. Vinuesa, «Towards Lightweight Mobile Pentesting Tools to Quickly Assess Machine Security Levels,» *IEEE Latin American Transactions,* p. pp, 2019. |

## Contenido entregado en los anexos

### Contenidos

Aparte del documento principal del trabajo de fin de grado se sube al sistema un fichero comprimido adjunto con una serie de contenidos adicionales: diagramas adicionales, documentación tipo Javadoc, código fuente, ejecutable, …. Aunque lo subido depende del tipo de proyecto a desarrollar, en esta sección se dan unas guías generales de como subirlo correctamente.

La plataforma de la Universidad tiene un límite de tamaño del fichero subido que suele ser de unos 40Mb. Si el adjunto fuese más grande, se recomienda crear un fichero README.TXT en el fichero comprimido subido y usar la función de compartir un enlace a un fichero de Dropbox, el OneDrive de la universidad o similar para que quien esté interesado en descargar estos ficheros. Preferiblemente debería usarse el OneDrive de la universidad, al ser institucional. También es posible apuntar a un repositorio GitHub o similar

La estructura de directorios del fichero adjunto debe poder recoger todos los ficheros relacionados con el proyecto, clasificándolos por su propósito dentro del mismo. Los tipos más frecuentes son: ficheros fuente, ficheros de configuración, ficheros de documentación…

Se deben crear directorios para contener cada uno de los tipos de ficheros. Tener una estructura estandarizada de los directorios del proyecto es importante por varias razones:

* Ayuda a localizar la información del proyecto. Por ejemplo, los ficheros fuente siempre deben estar en la carpeta src.
* Ayuda a los desarrolladores a determinar dónde debe ir cada fichero.
* Permite crear scripts de construcción estandarizados.

|  |  |
| --- | --- |
| **Directorio** | **Contenido** |
| *./* | Contiene un fichero README.TXT explicando toda esta estructura. |
| *./<nombre\_proyecto>* | Contiene toda la estructura de directorios del proyecto para desarrollo. **Ver la tabla Recomendación de estructura de directorios de desarrollo**. <nombre\_proyecto> debe sustituirse por el nombre corto del proyecto. |
| ./instalacion | Ficheros utilizados para la instalación del proyecto o un documento explicando donde descargar el software de terceros que necesita para instalarlo, y las versiones correspondientes si fuese necesario. Es también deseable incluir un fichero con el procedimiento de instalación típico abreviado para hacer una instalación del trabajo por defecto lo más rápidamente posible. |
| ./documentacion | Contiene la documentación asociada al proyecto en PDF (este mismo documento y cualquier otro que se considere necesario). |
| ./explotacion | BD y documento explicando donde descargar el software de terceros que se necesita para ejecutar el trabajo (por ejemplo, el servidor web), y las versiones correspondientes si fuese necesario. También se deberían incluir ficheros de configuración si fuesen necesarios para que el trabajo funcione |

Estructura general del fichero anexo entregado

#### Recomendación de la Estructura de Directorios de “desarrollo”

Se muestra aquí el contenido del directorio de desarrollo de la tabla anterior, incluyendo todos los directorios que deben depender del mismo. Este ejemplo se ha elaborado suponiendo que se están usando ciertas tecnologías Java. En caso de no usarlas, buscaremos un equivalente existente (si lo hay) en la que estemos usando nosotros. Si la aplicación está destinada a dispositivos móviles, se debe adaptar el propósito de las carpetas al equivalente en las tecnologías usadas para desarrollar para ellos.

|  |  |
| --- | --- |
| **Directorio** | **Contenido** |
| ./ Directorio raíz de “desarrollo” | Contiene los ficheros de proyecto del IDE utilizado |
| ./build | Contiene el build.xml de ant (si lo usamos) o herramienta de propósito similar. Debemos situarnos dentro para poder invocarlo |
| ./conf | Contiene los diferentes ficheros de configuración del proyecto. Podría contener distintos subdirectorios, en función de la tecnología usada. En este ejemplo se muestra un ejemplo de un proyecto *Web* hecho con tecnologías *Java*:   * **web**: contiene los ficheros de configuración de la aplicación Web (por ejemplo: web.xml) * **ear**: contiene los ficheros de configuración de una aplicación empresarial (por ejemplo: application.xml) * **ejb**: contiene los descriptores de despliegue de los EJB |
| ./dist | Directorio donde se sitúan los ficheros para la distribución del proyecto. Por ejemplo: los ficheros .war o .ear |
| ./doc | Contiene los ficheros generados por herramientas de generación de documentación automática como *Javadoc* o similar |
| ./lib | Bibliotecas externas (.jar, *.dll*, …) necesarias para compilar y distribuir, de las que depende este proyecto |
| ./compile-lib | Bibliotecas externas (.jar, .dll, …) necesarias para compilar pero que no deseamos distribuir |
| ./src | Ficheros de código fuente |
| ./src/sql | Este directorio contiene los scripts de *SQL* que permiten construir y meter los datos iniciales en la base de datos del proyecto (si existe) |
| ./web | Este directorio contiene los ficheros (.JSP, .ASPX, .HTML, …) de la *Web* (si el proyecto incluyese una) |
| ./web/images | Contiene las imágenes utilizadas por los ficheros de la web del proyecto |
| ./classes | Directorio donde se guardan los ficheros compilados (como por ejemplo los .class) |
| ./test | Directorio base para todos los ficheros utilizados en la automatización del proceso de prueba |
| ./test/java | Contiene todas las pruebas unitarias utilizadas en el proceso de prueba automatizado |
| ./test/sql | Scripts *SQL* utilizados en la carga de datos de prueba |

Estructura de la carpeta “desarrollo” del fichero anexo entregado

## GNU Free Documentation License

*Version 1.3, 3 November 2008*

*Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <*[*https://fsf.org/*](https://fsf.org/)*>*

*Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.*

#### 0. PREAMBLE

*The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.*

*This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.*

*We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.*

#### 1. APPLICABILITY AND DEFINITIONS

*This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.*

*A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.*

*A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.*

*The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.*

*The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.*

*A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".*

*Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.*

*The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.*

*The "publisher" means any person or entity that distributes copies of the Document to the public.*

*A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.*

*The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.*

#### 2. VERBATIM COPYING

*You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.*

*You may also lend copies, under the same conditions stated above, and you may publicly display copies.*

#### 3. COPYING IN QUANTITY

*If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.*

*If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.*

*If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.*

*It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.*

#### 4. MODIFICATIONS

*You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:*

1. *Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.*
2. *List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.*
3. *State on the Title page the name of the publisher of the Modified Version, as the publisher.*
4. *Preserve all the copyright notices of the Document.*
5. *Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.*
6. *Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.*
7. *Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.*
8. *Include an unaltered copy of this License.*
9. *Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.*
10. *Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.*
11. *For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.*
12. *Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.*
13. *Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.*
14. *Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.*
15. *Preserve any Warranty Disclaimers.*

*If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.*

*You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.*

*You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.*

*The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.*

#### 5. COMBINING DOCUMENTS

*You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.*

*The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.*

*In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".*

#### 6. COLLECTIONS OF DOCUMENTS

*You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.*

*You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.*

#### 7. AGGREGATION WITH INDEPENDENT WORKS

*A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.*

*If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.*

#### 8. TRANSLATION

*Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.*

*If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.*

#### 9. TERMINATION

*You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.*

*However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.*

*Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.*

*Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.*

#### 10. FUTURE REVISIONS OF THIS LICENSE

*The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See* [*https://www.gnu.org/licenses/*](https://www.gnu.org/licenses/)*.*

*Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.*

#### 11. RELICENSING

*"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.*

*"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.*

*"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.*

*An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.*

*The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.*

### ADDENDUM: How to use this License for your documents

*To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:*

Copyright (C) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document

under the terms of the GNU Free Documentation License, Version 1.3

or any later version published by the Free Software Foundation;

with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU

Free Documentation License".

*If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:*

with the Invariant Sections being LIST THEIR TITLES, with the

Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

*If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.*

*If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.*