INTERFAZ WEB PARA UN JUEGO DE TABLERO ONLINE

MEMORIA DEL PROYECTO

Grado en Ingeniería Informática

TRABAJO DE FIN DE GRADO

Septiembre, 2023

Autor:

Helena Fajín Riveiro

Tutor:

Rodrigo Santamaría Vicente

Certificado del tutor

D. Rodrigo Santamaría Vicente, profesor del Departamento de Informática y Automática de la Universidad de Salamanca.

CERTIFICA:

Que el trabajo titulado "Interfaz web para un juego de tablero online", que se presenta, ha sido realizado por Helena Fajín Riveiro, con DNI 45905953-T y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado en Ingeniería Informática en esta Universidad.

Salamanca, 7 de septiembre de 2023

Fdo.:

Firmado por SANTAMARIA VICENTE RODRIGO - ***7930** el día 07/09/2023 con un certificado emitido por

Resumen

Este Trabajo de Fin de Grado consiste en el desarrollo de una interfaz web para un juego de tablero online llamado The Wall is Down, creado por el tutor Rodrigo Santamaría Vicente en base a uno ya existente, Twilight Struggle.

En esta memoria se recoge el proceso de análisis, diseño y desarrollo de la interfaz, la cual ha sido implementada en JavaScript usando la librería React. Para su funcionamiento es necesario realizar peticiones HTTP que permitan mostrar la información a los jugadores sobre de las acciones llevadas a cabo en el juego. Consume los servicios RESTful de una infraestructura SOA elaborada por Javier Vidal Ruano para su Trabajo de Fin de Master, que representaría la parte del servidor y motor del juego.

El objetivo de este proyecto es la digitalización del juego The Wall is Down ofreciendo a los usuarios la posibilidad de jugar con una interfaz online.

Palabras clave: juego de mesa, online, interfaz, React, JavaScript, peticiones HTTP, infraestructura SOA, servicios RESTful.

Abstract

This Bachelor's Degree Final project consists in the development of a web interface for an online board game called The Wall is Down, created by the tutor Rodrigo Santamaría Vicente and based on an existing one, Twilight Struggle.

In this memory it is collected the process of the analisys, design and development of the interface, which has been implemented in JavaScript using React library. In order to work the use of HTTP requests is necessary because it allows the interface to show information about the actions that the players perform in the game. The interface consumes RESTful services from a SOA infrastructure created by Javier Vidal Ruano for his Master's Degree Final project, which represents the server part.

The purpose of this project is the digitization of the game The Wall is Down giving the users the possibility of playing with an online interface.

Key words: board game, online, interface, React, JavaScript, HTTP requests, SOA infrastructure, RESTful services.

Índice general

Índice de figuras

1 Introducción

Los juegos de mesa son una de las formas de entretenimiento más antiguas de la humanidad. Desde los dados, el ajedrez, el dominó, hasta el famoso Monopoly han entretenido a muchas personas, además de promover el desarrollo de la memoria, la concentración y la imaginación. Y es que a pesar de que la creación de videojuegos está en auge y avanza a una velocidad vertiginosa, los juegos de mesa online no se quedan atrás. El Parchís online en España llegó a alcanzar hasta tres millones de descargas durante el confinamiento y mundialmente supera ya los cien millones. Estos datos demuestran la gran demanda de este tipo de sistemas y la necesidad de su desarrollo.

La interfaz se basa en el juego The Wall is Down, el cuál es una modificación de Twilight Struggle ideada por el tutor Rodrigo Santamaría Vicente, con el propósito de hacerlo multimodal y de que reflejara las condiciones políticas, económicas y militares actuales.

Se implementó con el lenguaje de JavaScript usando la librería de React y para poder mostrar la información del juego a los usuarios, se usaron peticiones HTTP a los servicios RESTful de una API SOA creada por Javier Vidal Ruano como Trabajo de Fin de Master con el título "Arquitectura de servicios RESTful para un juego de tablero online" [4].

1.1. Reglas básicas del juego

The World is Down es un juego de estrategia en el que pueden participar hasta cuatro jugadores que representan las cuatro potencias mundiales: Europa, Estados Unidos, Rusia y China. Está ambientado en la caída del muro de Berlín, que marca el inicio de la post-guerra fría. El nombre del juego hace referencia a este suceso histórico. Para poder ganar es necesario conseguir dominar y controlar el mayor número posible de países durante las dos épocas en las que se divide el juego, pre-9/11 y post-9/11, así como conseguir la mayor puntución de VP.

Cada ronda se compone de la fase de cabecera y de la fase de acción.
En la fase de cabecera, cada jugador escoge una carta que se usará para calcular el orden de los turnos para la fase de acción. Se comprueban todos los números de las cartas de cabecera y se ordenan de mayor a menor. Si se tiene una carta de puntuación el valor de ésta es de 0 en el orden. A continuación, según el orden obtenido, cada jugador juega su carta. Esta última se resuelve por su texto.
La fase de acción se repite dos veces en cada ronda. Primero se escoge la carta y se decide si es jugada por influencia o por desestabilización, siempre y cuando no sea una carta de puntuación. Si se decide jugar la carta por influencia se añadirá influencia en un país que tenga igual o menor estabilidad como puntos tenga la carta. Si sobran puntos se puede ganar influencia en otro país. En cambio, si se decide jugar por desestabilización, se realiza el siguiente procedimiento:
Se tira un dado de seis y se calcula:

$$resultado\_dado + puntos\_carta - estabilidad\_pais\_a\_desestabilizar \times 2 = Xpuntos$$

Si el resultado es positivo, se puede eliminar hasta X puntos de influencia en un país. Si el país es conflictivo, se pierde un punto VP. Cuando se juega una carta por operación que pertenece al bloque contrario (Este/Oeste) el texto de la carta también se resuelve.

Las cartas de puntuación se deben jugar antes de que termine la ronda. Se usan para obtener puntos VP según la presencia, dominación y control que se tiene en una región. Se puede encontrar más información en el siguiente enlace de Google Drive (**link**)[3].

2 | Objetivos

En este apartado se identifican tanto los objetivos personales como los objetivos del sistema que serán necesarios conseguir para alcanzar un correcto desarrollo del proyecto. El objetivo principal consiste en digitalizar el juego The Wall is Down y que los usuarios puedan jugar usando la interfaz de forma online.

2.1. Objetivos funcionales

Los siguientes objetivos son los que se quieren alcanzar en el proyecto:

- **Gestión de usuarios**: El sistema deberá permitir a los usuarios tanto entrar como salir del juego. El usuario no tendrá que identificarse para ello, pues será el sistema quien le otorgue un token único.
- **Gestión de partidas**: Podrán crear partidas siendo los anfitriones, unirse a partidas ya creadas por otro usuario y consultarlas mediante un listado que permite ver la información detallada y/o eliminarlas.
- **Jugar partida**: Dentro de una partida el usuario debe poder escoger sus cartas tanto en la fase de cabecera como en la fase de acción. Además, puede escoger la operación a realizar a la hora de jugar la carta: si es una carta de cabecera, se jugará por texto; si es de puntuacion, se resolverá y se guardará la puntuación; y si no se dan estos casos, el usuario podrá escoger entre jugarla por desestabilización o por influencia.

- **Información tablero**: La información del tablero, al igual que la influencia de cada jugador en los países o la puntuación podrán ser visualizadas por los usuarios en los momentos en los que no estén jugando las cartas de su turno correspondiente.

2.2. Objetivos no funcionales

En cuanto a los objetivos no funcionales, se han identificado los siguientes:

- **Usabilidad**: El sistema deberá ser intuitivo y cómodo en la interacción con el usuario.
- **Portabilidad**: El sistema deberá poder ser ejecutado en distintos sistemas operativos.
- **Privacidad**: El sistema deberá garantizar el anonimato de los usuarios.
- **Concurrencia**: El sistema deberá garantizar que pueda ser utilizado por varios usuarios a la vez sin que afecte al correcto funcionamiento.

2.3. Objetivos personales

En cuanto a los objetivos personales, el primero es desarrollar el Trabajo de Fin de Grado de la mejor forma posible siendo ésta la primera vez trabajando en un proyecto de una dimensión mayor y de forma individiual.

Respecto al crecimiento profesional, se ha tenido la oportunidad de aprender JavaScript, el cual es uno de los lenguajes de programación más usado para el desarrollo web y su conocimiento abre más oportunidades dentro del ámbito laboral.
También se ha profundizado en el uso de herramientas como el entorno de NodeJS con React, que en conjunto forman el framework más usado por desarrolladores web.

Puede que el objetivo más complejo de alcanzar sea conseguir compaginar el desarrollo del proyecto con un trabajo de prácticas a media jornada. La carga laboral diaria resulta notable, por lo que es imprescindible una buena gestión de las tareas y disciplina para afrontar el proyecto.

3 | Conceptos teóricos

Una interfaz web es una estructura formada por elementos gráficos, componentes visuales, interactivos y de diseño para el usuario, que representan información y que hace posible el acceso a los contenidos del sitio web. Su objetivo principal es proporcionar un entorno visual que permita la comunicación de un programa con el usuario.

Una aplicación web se puede dividir en dos partes: el backend y el frontend, ambas de igual importancia. A pesar de que este proyecto se engloba en la parte del frontend, también se proporcionará una definición de los elementos del backend que han sido necesarios.

El frontend engloba todo el código necesario para obtener la interfaz de un sitio web, haciendo posible que el usuario pueda interactuar directamente. Para una buena experiencia de usuario se prioriza la elección de los componentes de diseño como tipografía, colores o estilos, así como también la estructura o el contenido web.

El backend es el código responsable de que la página o aplicación web funcione de forma correcta, como el procesamiento de solicitudes y datos que el frontend necesita comunicar, o garantizar la seguridad y privacidad del sitio. Normalmente está formado por una API y una base de datos.

API es la abreviatura de Application Programming Interface (interfaz de programación de aplicaciones). Consiste en un programa que expone un conjunto de reglas y protocolos que permiten que otro se pueda comunicar con éste. Para poder usarlas sólo es necesario cumplir con las especificaciones establecidas para la comunica-

ción.

La arquitectura orientada a los servicios (SOA) es un estilo arquitectónico diseñado para el desarrollo de aplicaciones en base a servicios disponibles. Se caracteriza por su flexibilidad y su versatilidad, lo que hace posible que los servicios sean consumidos por clientes.

REST (REpresentational State Transfer) es un estilo de arquitectura software que indica cómo se debe realizar el intercambio y manejo de datos a través de servicios web. Su protocolo se basa en la arquitectura Cliente-Servidor y en el protocolo HTTP. Para que un servicio se le pueda llamar RESTful es necesario que su URI sea del tipo http://foo.com/resource, que use datos en formato estándar como JSON, que sólo use peticiones GET, POST, PUT y DELETE como peticiones HTTP, y que se pueda acceder a través de un navegador.

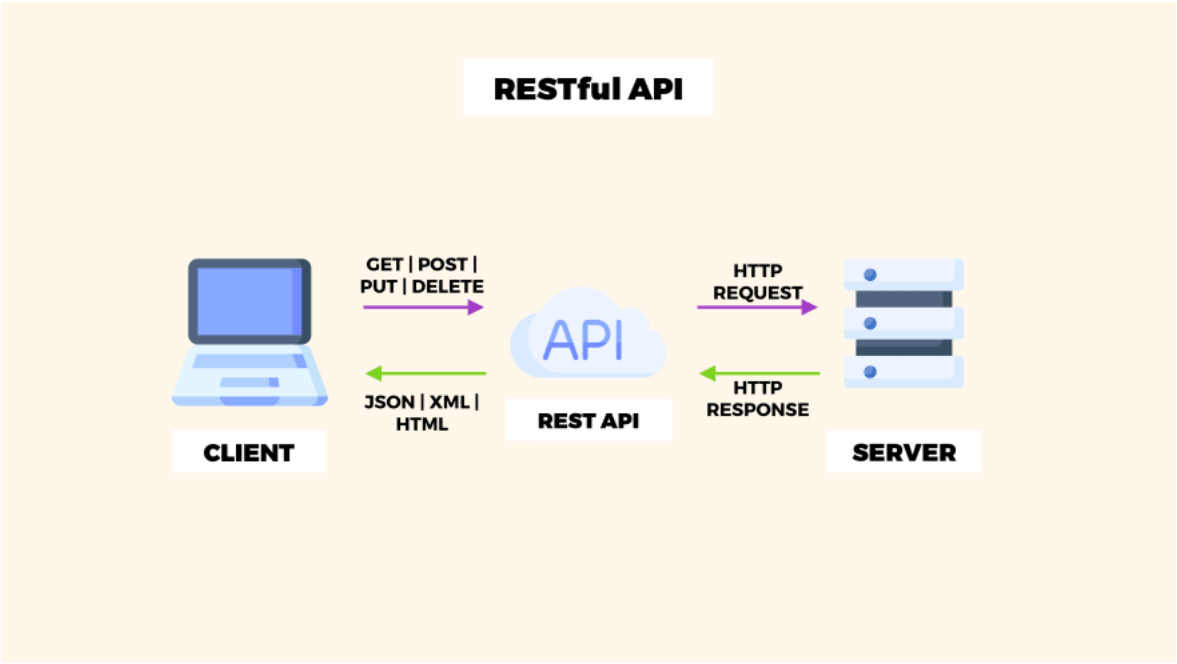


Figura 1: Arquitectura de una RESTful API

4 | Introducción a la interfaz

En este apartado se realizará una pequeña introducción a la interfaz elaborada para el juego The Wall is Down. Para empezar, en la figura [2] se puede ver la pantalla de inicio del juego:

Figura 2: Página de inicio

Una vez que se clica en el botón de Sign In se accede a la pantalla del menú principal, en donde se muestran las distintas opciones del usuario: crear una partida, ver las partidas o unirse a una. Además, también se dispone de una barra de cabecera con un botón para volver al menú principal y otro para salir del juego.

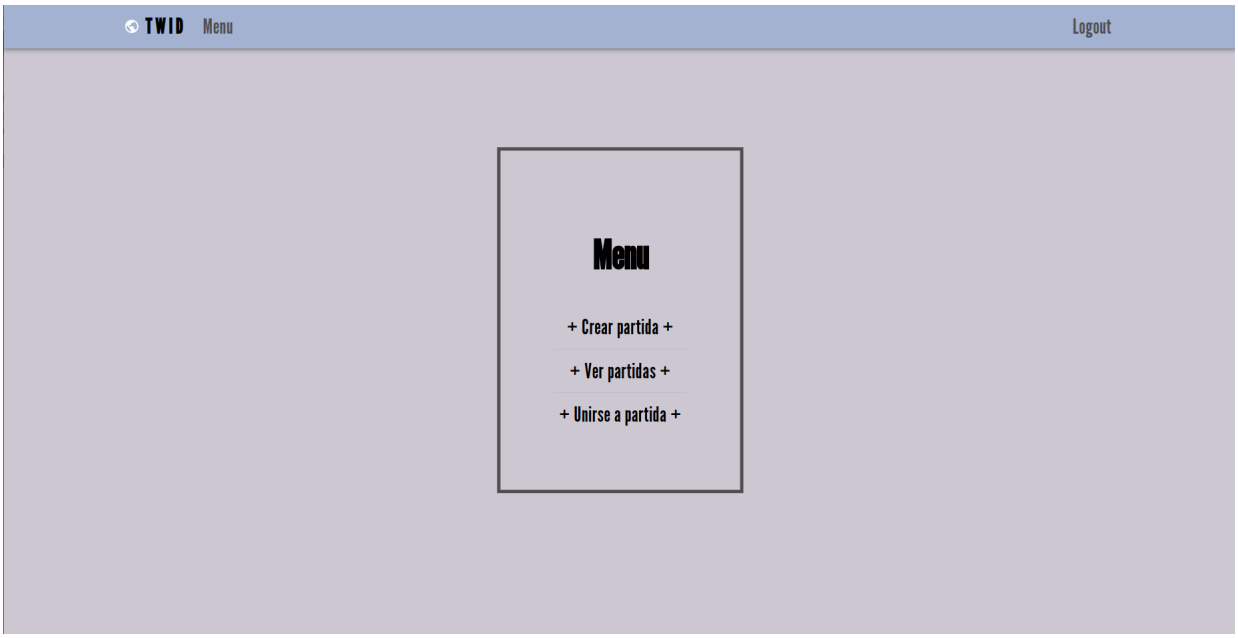


Figura 3: Página principal

Dentro de la partida, lo primero que se debe escoger es la carta de cabecera para poder elaborar el orden de jugada de la ronda, como se puede ver en la figura [4]. Después se procede a jugar la carta de cabecera según el orden establecido. A continuación, cada jugador escoge y juega sus cartas en la fase de acción, como se muestra en la figura [5].

Figura 4: Página tablero con cartas

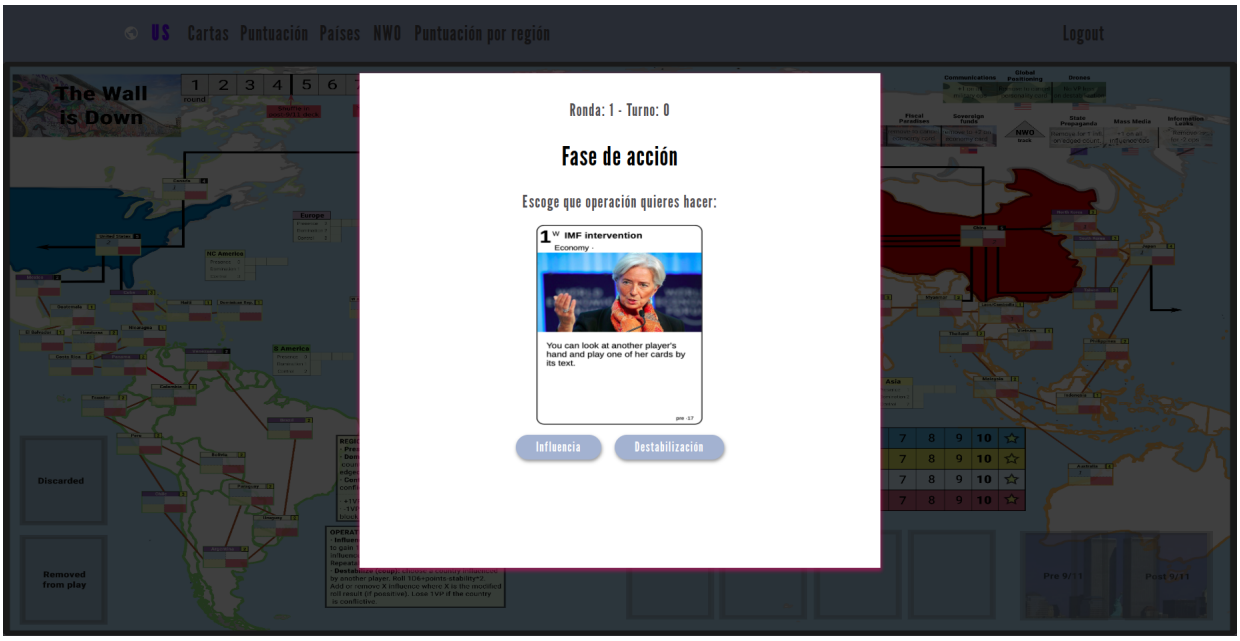


Figura 5: Página tablero jugando carta

Cuando no se está jugando el turno es posible visualizar los datos del tablero a través de la barra de navegación superior. Las opciones son: ver las cartas del jugador, la puntuación, el estado de influencia de los países, el estado de New World Order y la puntuación de cada región. También se puede salir del juego usando el botón de logout.

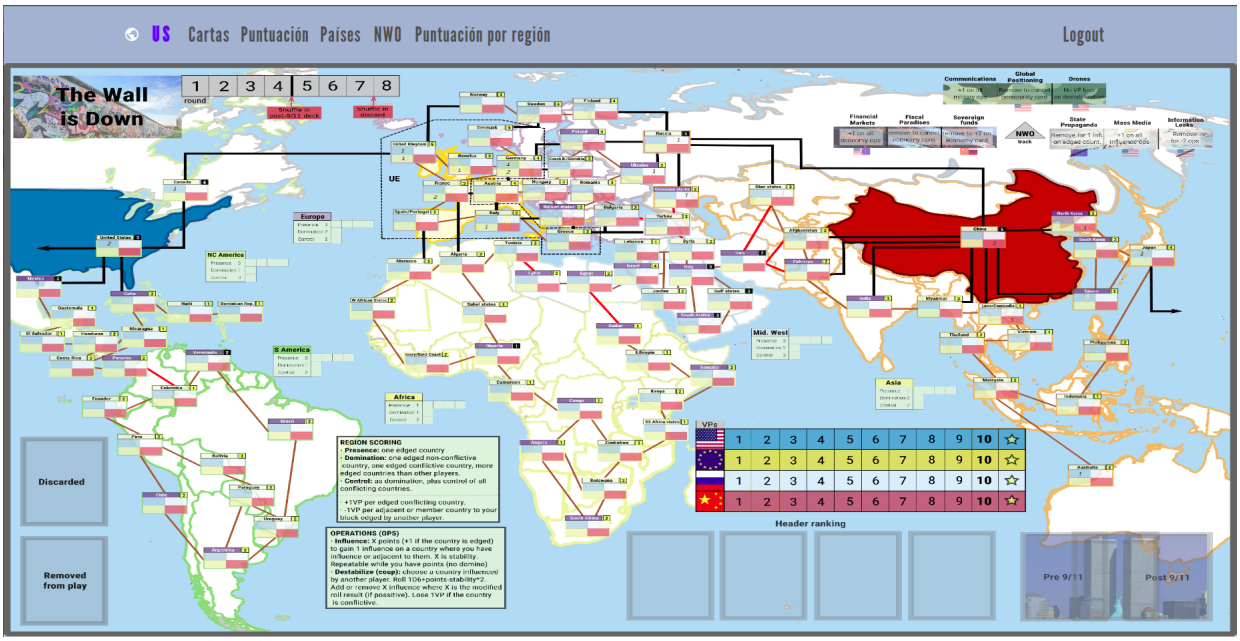


Figura 6: Página tablero

5 | Técnicas y herramientas

A continuación, se detallan las técnicas y las herramientas usadas durante el desarrollo del proyecto.

5.1. Herramientas de desarrollo

5.1.1. Visual Studio Code

Es el editor de código fuente usado para desarrollar la interfaz. Fue creado por la empresa de Microsoft y actualmente es de los más usados por los desarrolladores porque, aparte de ser gratis y de código abierto, soporta una gran cantidad de lenguajes. Es rápido, ligero, eficiente y posee numerosas herramientas integradas, así como extensiones que contribuyen a la mejora del flujo de trabajo.

5.1.2. NodeJS

NodeJS es un entorno en tiempo de ejecución de JavaScript diseñado para el desarrollo de aplicaciones web. Gracias a éste, es posible compilar y ejecutar el código del proyecto.

5.1.3. Git

Git es el sistema de control de versiones más usado mundialmente. Fue creado por Linus Torvalds en 2005 para conseguir tener un registro de los cambios efectuados en los proyectos de equipo, y mantener el control en caso de la aparición de conflictos. Una de las funcionalidades más importante que tiene es la posibilidad de crear ramas o branches de modo que cuando se desee hacer un cambio en el código, éste se haga de forma aislada y pueda ser comprobado antes de hacer el cambio efectivo en la rama principal. En caso contrario, siempre será posible volver a una versión del código estable.

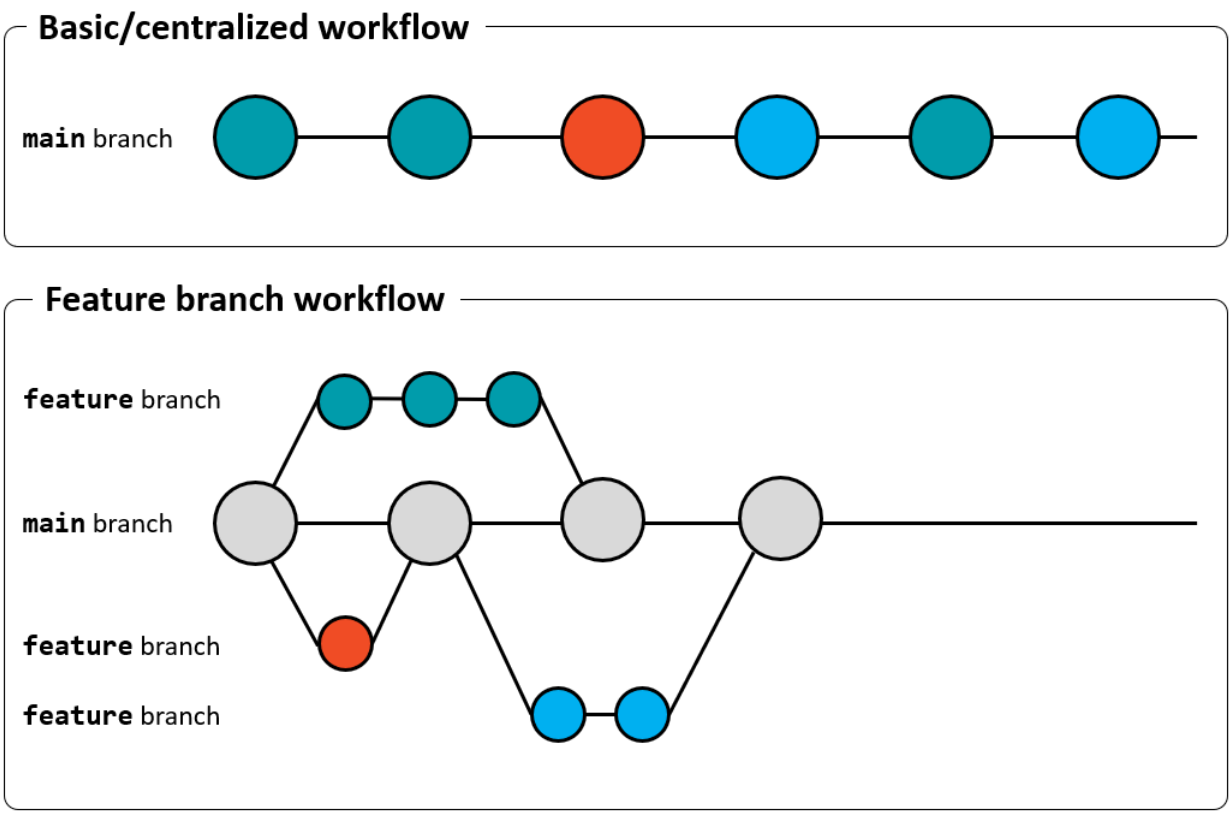


Figura 7: Representación de la funcionalidad de las ramas en Git

5.1.4. Github

Como su nombre indica, la web utiliza el sistema de control de versiones Git. Es un sistema donde los desarrolladores pueden subir su código en un repositorio en la nu-

be y tener la capacidad de administrarlo de la misma forma que se lleva a cabo en Git.

5.2. Herramientas CASE

5.2.1. Visual Paradigm

Visual Paradigm es una herramienta UML CASE que ayuda a desarrollar programas informáticos. Ofrece herramientas para capturar requisitos, planificar software, realizar el modelado de clases y el modelado de datos.

5.3. Herramientas de diseño

5.3.1. Figma

Figma es una plataforma de edición gráfica vectorial y de diseño de interfaces web. Su fama ha aumentado en el transcurso de los últimos años entre los diseñadores UI y empresas debido a que es una herramienta colaborativa y online, lo que la hace perfecta para ambientes de desarrollo en equipo. Con esta herramienta ha sido posible la elaboración y diseño del prototipo de la interfaz.

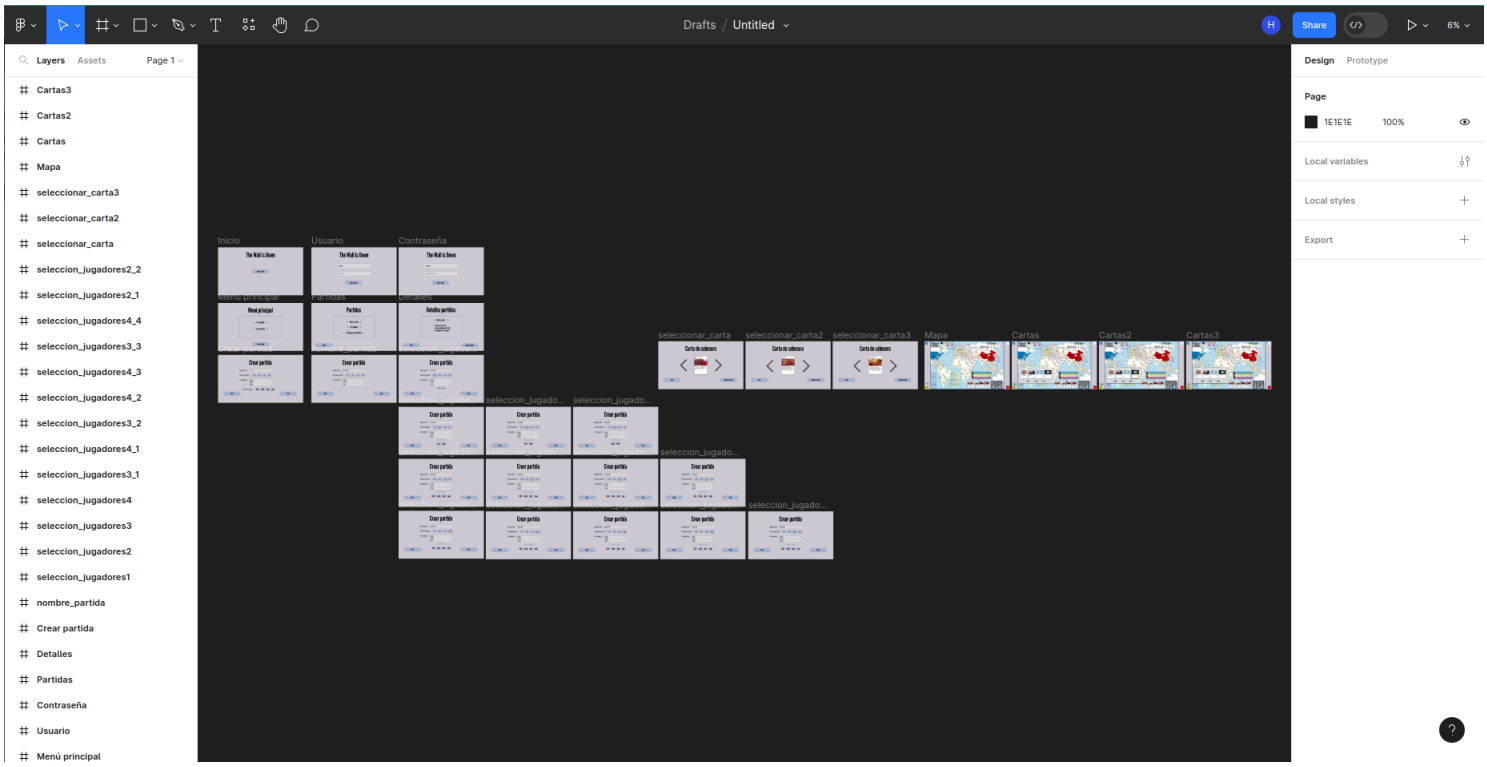


Figura 8: Herramienta Figma

5.4. Lenguajes de programación

5.4.1. JavaScript

JavaScript es el lenguaje de programación elegido para desarrollar todo el proyecto. Permite implementar funciones complejas en páginas web con contenido interactivo y dinámico. Hoy en día es un lenguaje que puede ser usado tanto para el lado del cliente como para el del servidor gracias a la existencia de NodeJS. Su gran popularidad se debe a su facilidad de aprendizaje, la velocidad de ejecución, múltiples opciones de efectos visuales, versatilidad a la hora de elaborar páginas web dinámicas y permitir trabajar en modo fullstack.

5.4.2. JSX

JSX es una extensión de la sintaxis de JavaScript que une la sintaxis de los lenguajes tipo HTML/XML para definir el árbol de elementos de nuestros componentes de React. Al compilarse se transforma a un objeto JavaScript que se mapeará a un elemento del DOM.

5.4.3. CSS

CSS es un lenguaje que proporciona y maneja el diseño en las páginas web. Sus siglas significan hojas de estilo en cascada (Cascading Style Sheets), lo cual significa que existe la posibilidad de que se tengan varias hojas con propiedades heredadas. Funciona aportándole diseño al contenido HTML para mejorar la experiencia de usuario mediante la creación de reglas. Se especifican los grupos de estilos y se indica a qué elementos de la página web se deben aplicar.

5.5. Librerías

5.5.1. ReactJS

ReactJS fue creada por Facebook al igual que el lenguaje JSX, es una de las librerías más populares de JavaScript para el desarrollo de aplicaciones móviles y web. Esta librería está orientada a componentes. Pese a que no está enteramente orientada a objetos ni es completamente funcional, combina elementos de ambos paradigmas de programación.

Esta librería se ha empleado por su funcionalidad para la creación de componentes reutilizables. Gracias a esto es posible usar un mismo elemento en varias partes del sitio web sin necesidad de volver a escribir todo el código, consiguiendo poco a poco una interfaz de usuario más compleja.

También el uso de estados (useState Hook) ha sido de gran ayuda, empleándose para controlar los cambios en la interfaz.

El funcionamiento de React consiste en crear una representación en memoria del DOM gracias al código JSX, llamado Virtual DOM, cuyas mejores características son que es ligero y sencillo de crear. Si la renderización de un componente se modifica, React compara los elementos que han cambiado y actualiza solo esas partes en el DOM real.

5.5.2. MUI

Como librería para incorporar el diseño se escogió MUI, que proporciona una gran cantidad de componentes básicos y avanzados para desarrollar la interfaz en React con mayor facilidad. Fue creada en código abierto por Google y está basada en Material Design.

6 | Aspectos relevantes del desarrollo

6.1. Metodologías empleadas

Durante el proceso de desarrollo del proyecto se han usado varias metodologías.

6.1.1. SCRUM

Las metodologías ágiles son de las más usadas dentro del entorno de equipos de desarrollo software porque cumplen con la idea de que en un proyecto surgen imprevistos y la necesidad de cambios. Consisten en un modelo de trabajo con un sistema iterativo que cumple con unas entregas cíclicas. SCRUM toma esta idea y funciona dividiendo el proyecto en tareas que se proponen completar con cada iteración llamada sprint. Al terminar cada fase de sprint se realiza una revisión, de esta forma se consigue abordar de manera más rápida y eficiente el problema: En la imagen [9] se puede visualizar las fases del proceso de SCRUM.

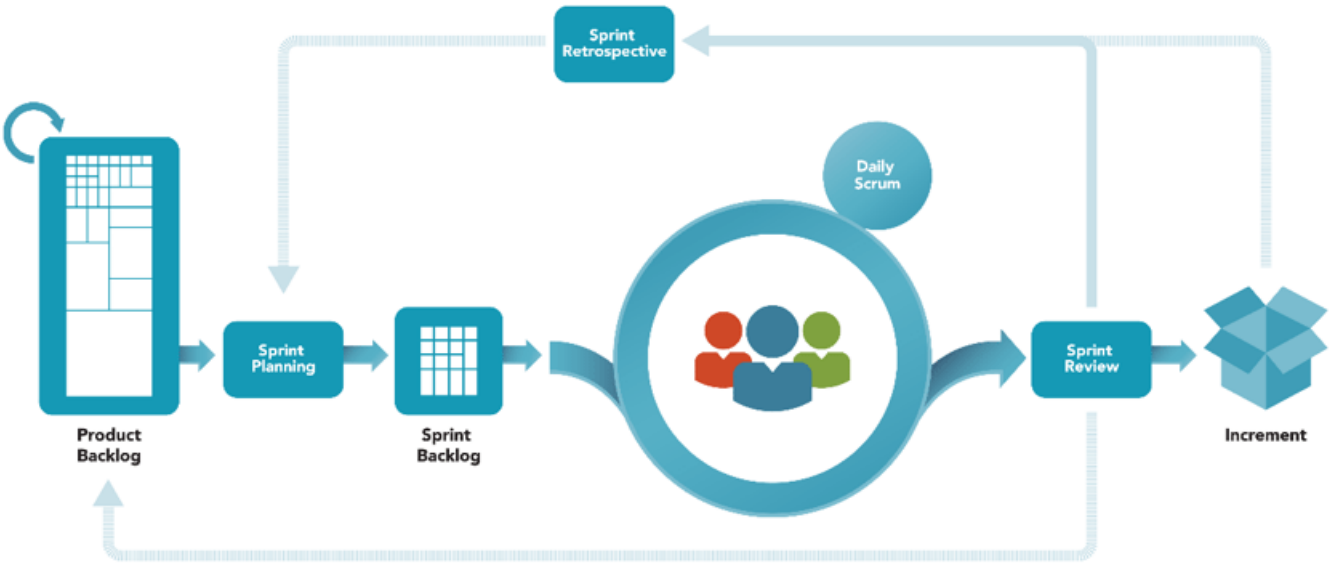


Figura 9: Metodología ágil SCRUM

Product Backlog

Ésta es la primera fase del proceso donde es necesario obtener una lista de tareas ordenada según su prioridad. Para conseguir esta lista primero fue necesario conocer los requisitos del sitema. Estos últimos se obtuvieron gracias a la elicitación de requisitos descrita en el anexo II "Desarrollo de requisitos del software", del que se hablará más adelante , y también de la creación de historias de usuario, que ayudan a una mejor comprensión durante la planificación del sprint. Disponer de las reglas del juego elaboradas por el tutor resultó de gran ayuda para poder entender las necesidades de los usuarios y crear las historias.

En las siguientes figuras [10], [11] y [12] se pueden ver las tareas obtenidas y ordenadas según su prioridad. El criterio de organización ha sido seleccionar cuáles de ellas eran más necesarias. Por ejemplo, fue necesario crear primero la página de inicio de sesión con la llamada a la API que identifica al usuario, antes que la creación de una partida porque el usuario debe estar identificado por la API.

Organización para resolución de tareas		
1º	**TA1**	Crear una página de inicio de sesión mediante un botón.
2º	**TA2**	Crear una llamada a la API SOA que registre un usuario y devuelva un token.
3º	**TA7**	Crear en el menú de la página principal un botón que permita crear una partida.
4º	**TA8**	Crear una interfaz para crear la partida.
5º	**TA9**	Crear una llamada a la API SOA que cree una partida al usuario y devuelva un identificador.
6º	**TA10**	Crear en el menú de la página principal un botón que permita unirme a una partida.
7º	**TA11**	Crear una interfaz de unión de la partida.
8	**TA12**	Crear una llamada a la API SOA que una al usuario a una partida mediante un identificador introducido.
9º	**TA3**	Crear un botón accesible en varias páginas para cerrar sesión.
10º	**TA4**	Crear una llamada a la API SOA que elimine el usuario y sus datos.
11º	**TA5**	Crear en el menú de la página principal un botón que permita ver las partidas.
12º	**TA6**	Crear una llamada a la API SOA que pida las partidas del usuario.
13º	**TA17**	Crear en la interfaz de creación de partidas un botón o list box para seleccionar un jugador.
14º	**TA18**	Crear en la interfaz de creación de partidas un botón para empezar la partida.
15º	**TA19**	Crear una llamada a la API para empezar la partida.
16º	**TA20**	Crear un sistema de comunicación entre jugadores para avisar de que la partida ha empezado.

Figura 10: Tabla tareas ordenadas por prioridad 1

17º	**TA21**	Crear una interfaz o página de juego para la partida.
18º	**TA22**	Crear una sección para visualizar cartas.
19º	**TA23**	Crear una opción de seleccionar una carta.
20º	**TA24**	Crear una petición a la API SOA para seleccionar la carta de cabecera.
21º	**TA25**	Crear una sección de jugar una carta, p.ej. con un modal.
22º	**TA26**	Crear un sistema de comunicación o avisos entre jugadores para jugar las cartas de cabecera de forma organizada.
23º	**TA27**	Crear una petición a la API SOA para jugar la carta de cabecera por texto o puntuación.
24º	**TA28**	Crear un botón para jugar la carta por influencia.
25º	**TA29**	Crear una petición a la API SOA para jugar la carta por influencia, si no es posible, jugarla por new world order.
26º	**TA30**	Crear un botón para jugar la carta por desestabilización.
27º	**TA31**	Crear una peticion a la API SOA para jugar la carta por desestabilización, si no es posible, jugarla por new world order.
28º	**TA32**	Crear un botón para jugar la carta por puntuación.
29º	**TA33**	Crear una peticion a la API SOA para jugar la carta por puntuación.
30º	**TA34**	Crear un botón que abra un modal para ver mis cartas.
31º	**TA35**	Crear una peticion a la API SOA para ver mis cartas restantes.
32º	**TA36**	Crear un botón que abra un modal para ver la puntuación.
33º	**TA37**	Crear una peticion a la API SOA para ver la puntuación.
34º	**TA38**	Crear un botón que abra un modal para ver el estado de influencia de los países.
35º	**TA39**	Crear una petición a la API SOA para ver el estado de influencia de los países.
36º	**TA45**	Crear sistema para calcular el ganador de la partida.
37º	**TA40**	Crear un botón que abra un modal para ver el estado del new world order.
38º	**TA41**	Crear una peticion a la API SOA para ver el estado del new world order.

Figura 11: Tabla tareas ordenadas por prioridad 2

39	**TA13**	Crear en la página de ver partidas un botón para eliminar.
40º	**TA14**	Crear una llamada a la API SOA que elimine la partida seleccionada.
41º	**TA15**	Crear en la página de ver partidas un botón para ver el detalle de una partida.
42º	**TA16**	Crear una llamada a la API SOA que devuelva los detalles de la partida.
43º	**TA42**	Crear un botón que abra un modal para ver la puntuación de las regiones.
44º	**TA43**	Crear una petición a la API SOA para ver la puntuación de las regiones.
45º	**TA44**	Crear un modal que aparezca cuando finalice la partida con un botón para volver a la página princial.

Figura 12: Tabla tareas ordenadas por prioridad 3

Sprint Planning

En esta fase es donde se planifica los sprints. Para ello es necesario relizar los puntos de historias. Se trata de una medida que permite estimar el esfuerzo que se debe hacer para terminar un trabajo, asignando un valor según la complejidad, el volumen de trabajo o la incertidumbre. Particularmente, la estimación resultó una fase complicada por la falta de conocimiento ya que es la primera vez que se aborda la elaboración de un trabajo con un lenguaje y entorno de desarrollo en un principio desconocido y de tales dimensiones. A continuación, es necesario saber aproximadamente cuantos sprints son necesarios realizar para obtener el producto y la estimacion en tiempo. Se estableció una duración de sprint de 2 semanas, pero como se desconocía la capacidad de cuántos puntos de historia se podían completar se tomó por referencia la cantiadad de puntos elaborados en el primer sprint. El resultado fue de 6 puntos de historias y teniendo 49 puntos en total se calcularon:

$$\frac{49PH}{6PH/sprint} \approx 8sprints$$

La estimación de sprints necesarios para completar el proyecto son 8, que en tiempo equivale a:

$$8sprints \times 2semanas = 16semanas = \textbf{4 meses}$$

En la figura [13] a continuación se pueden ver las historias de usuario con su respectiva puntuación:

Historias de usuario		PH
US-001	**Como** un usuario no logueado **quiero** poder iniciar sesión **para** que el sistema pueda otorgarme una identificación.	1
US-002	**Como** un usuario logueado **quiero** poder cerrar sesión **para** que el sistema olvide mi información.	1
US-003	**Como** un usuario logueado **quiero** poder ver mis partidas **para** saber qué partidas he creado o me he unido.	2
US-004	**Como** usuario logueado **quiero** poder crear partidas **para** compartirlas con otro usuario y poder jugarlas posteriormente.	3
US-005	**Como** usuario logueado **quiero** poder unirme a partidas **para** poder jugarlas posteriormente.	2
US-006	**Como** usuario logueado **quiero** poder eliminar mis partidas **para** eliminar las partidas que no me van a hacer falta.	2
US-007	**Como** usuario logueado **quiero** poder ver el detalle de mis partidas **para** poder visualizar su información.	1
US-008	**Como** usuario logueado **quiero** poder seleccionar jugador **para** poder tener un jugador a la hora de jugar la partida.	1
US-009	**Como** usuario logueado **quiero** poder empezar la partida creada **para** indicarle al resto que ya es posible jugar.	2
US-010	**Como** jugador **quiero** poder establecer la carta de cabecera **para** indicarle al sistema mi carta con la que se calculará el orden de jugada.	3
US-011	**Como** jugador **quiero** poder jugar mi carta de cabecera **para** resolver el texto de mi carta.	3
US-012	**Como** jugador **quiero** poder jugar una carta de influencia **para** añadir influencia en un país.	5
US-013	**Como** jugador **quiero** poder jugar mi carta de desestabilización **para** eliminar influencia en un país.	5
US-014	**Como** jugador **quiero** poder jugar mi carta de puntuación **para** conseguir mi puntuación en la región de la carta.	3
US-015	**Como** jugador **quiero** poder ver mis cartas restantes **para** poder ver la información en ellas y calcular mi próxima jugada.	2
US-016	**Como** jugador **quiero** poder ver la puntuación **para** visualizar que otro jugador está ganando o perdiendo.	2
US-017	**Como** jugador **quiero** poder ver el estado de influencia de los países **para** poder saber que carta debo escoger a continuación.	3
US-018	**Como** jugador **quiero** poder ver el estado del New World Order **para** visualizar la información.	3
US-019	**Como** jugador **quiero** poder ver la tabla de puntuación de cada región **para** poder calcular quien tiene ventaja.	2
US-020	**Como** jugador **quiero** poder ver el ganador de la partida **para** saber quien ha ganado.	4
Total		49

Figura 13: Puntos de historia

Sprint Backlog

En esta fase se identifican las tareas que se deben completar en el sprint. Es primordial tener en cuenta la duración y dificultad de las tareas para no cometer el fallo de sobrecargarlo y poder llegar a completarlo correctamente. En las siguientes tablas [14] y [15] se puede ver el reparto de tareas elaborado mediante diagramas de Gantt. La primera columna representa el número del sprint y la segunda, la semana.

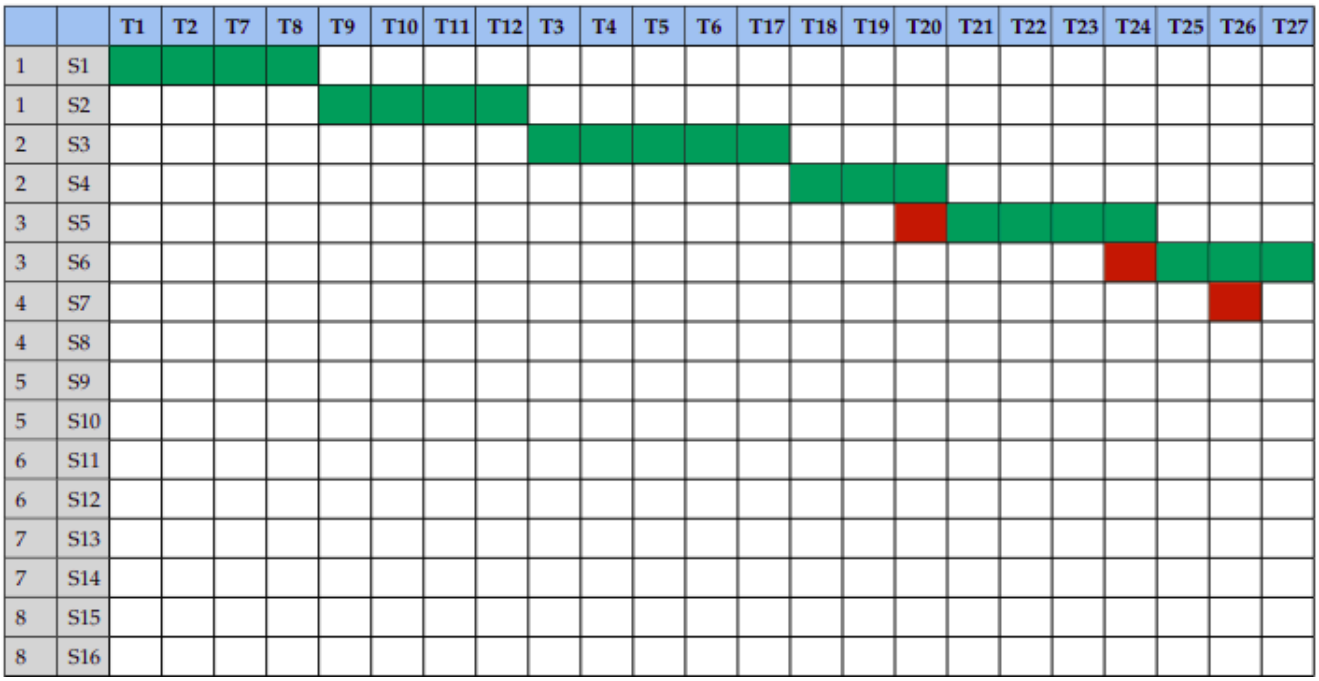


Figura 14: Diagrama de Gantt 1

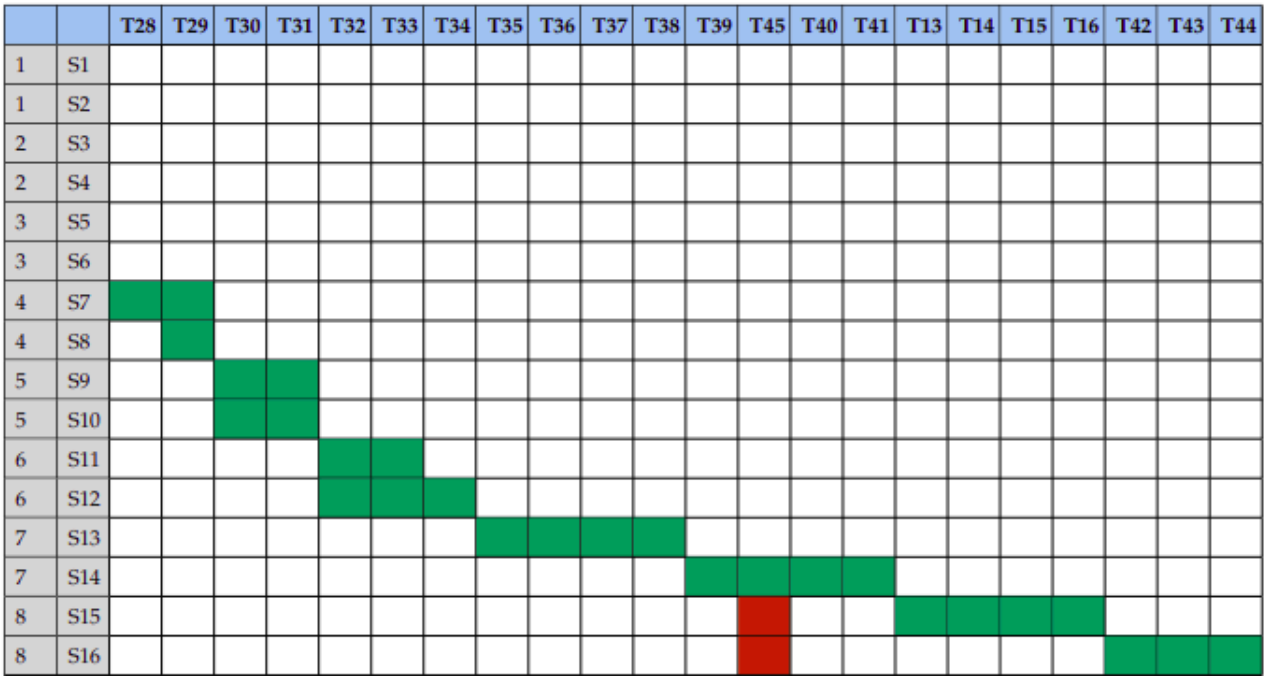


Figura 15: Diagrama de Gantt 2

Daily Sprint

En un equipo de desarrollo que toma como modelo de trabajo la metodología de SCRUM, los daily sprints son pequeñas reuniones diarias donde se inspecciona el

progreso para conocer si el avance del sprint es positivo. En este caso, al tratarse de un proyecto individual, no se vio la necesidad de llevar a cabo esta fase.

Sprint Review

Esta etapa supone el final del sprint: es el momento donde se revisa y se determina si los objetivos o tareas establecidas se han alcanzado. En las imágenes [14] y [15], las celdas marcadas en rojo indican las tareas no alcanzadas según la planificación inicial, marcada en verde. Esto indica que la estimación de la dificultad en la tarea no era la adecuada o que la carga de trabajo del sprint no estuvo bien repartida.

6.1.2. Proceso Unificado

Para este proyecto se usaron algunas técnicas del Proceso Unificado. Es un marco de trabajo iterativo e incremental compuesto de cuatro fases: Inicio, Elaboración, Construcción y Transición. Es un proceso conducido por casos de uso, centrado en la arquitectura y basado en componentes. La razón por la que se usaron técnicas del Proceso Unificado fue la necesidad de una captación de requisitos, que constituye la primera fase de este proceso.
Primero se definió el modelo de casos de uso, que ayuda a representar la funcionalidad y comportamiento del sistema. El anexo II "Especificaciones de los requisitos del software" es la documentación del modelo de casos de uso donde se capturaron los objetivos, los actores, los requisitos funcionales, no funcionales y de información. A continuación están las capturas del modelo de casos de uso:

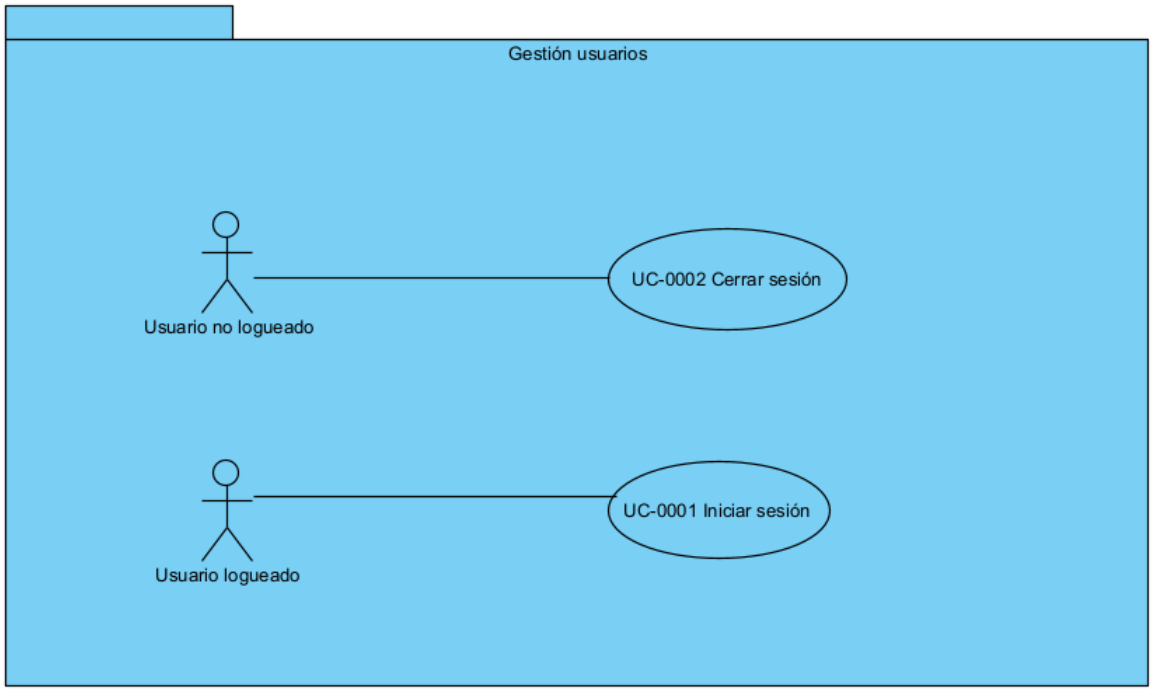


Figura 16: Casos de uso paquete gestión de usuarios

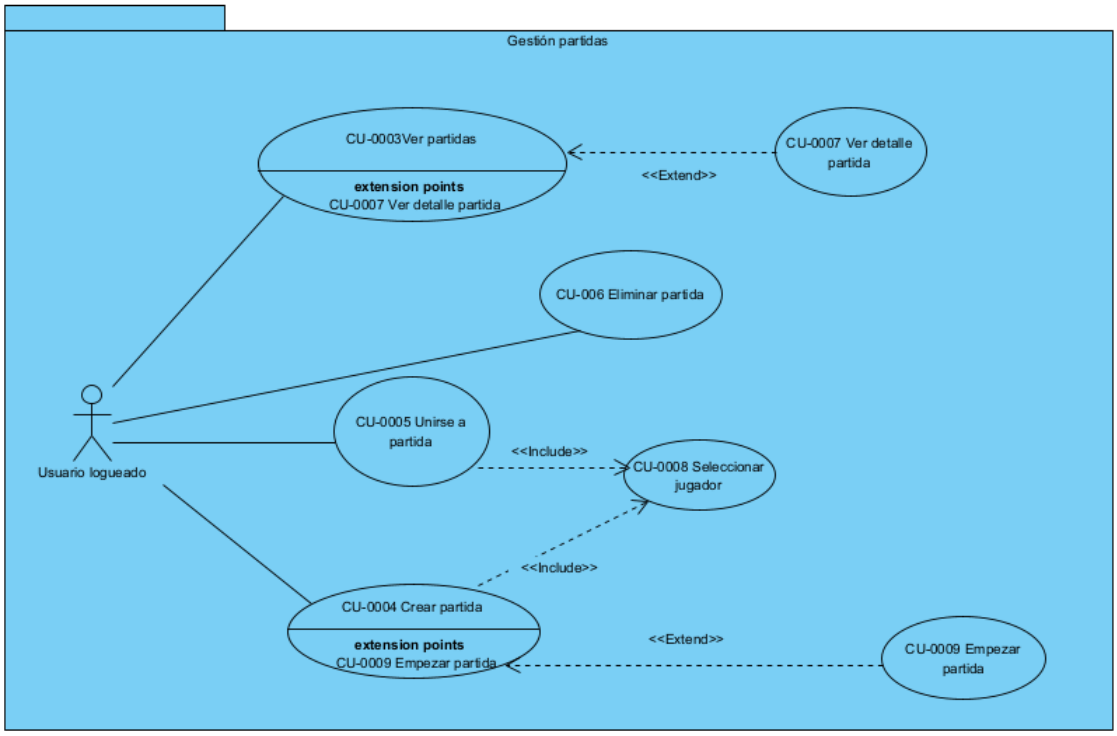


Figura 17: Casos de uso paquete gestión de partidas

Figura 18: Casos de uso paquete gestión de tablero

A pesar de que una de las mayores utilidades del modelo de dominio es ser una forma de "inspiración" para el diseño de los objetos software, React está más bien orientado a componentes. El motivo principal para llevar a cabo su elaboración es que ayuda a definir los requisitos de datos estáticos de un sistema y muestra las clases conceptuales en el dominio del problema.

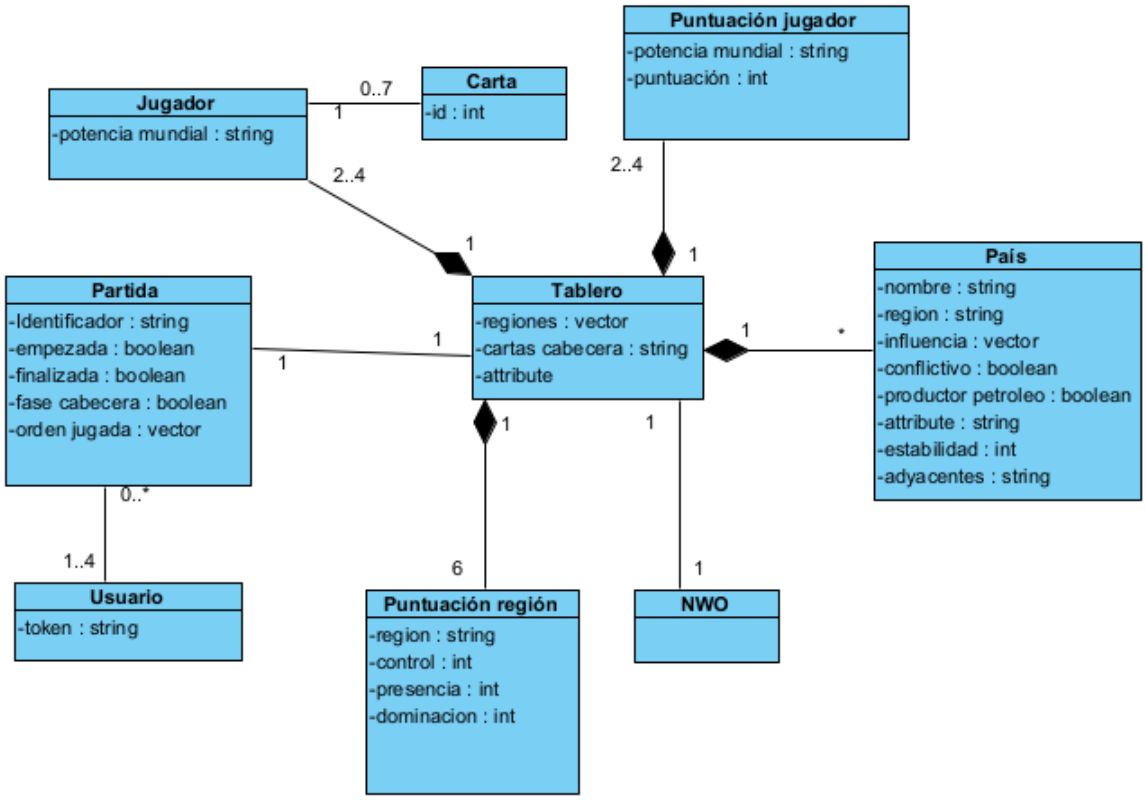


Figura 19: Modelo de dominio

También en el anexo III ”Análisis de requisitos” se elaboraron diagramas de secuencia para entender el proceso de intercambio de mensajes entre los componentes abstactos del sistema, la interacción de las entidades o la secuencia de pasos que suceden para que un actor del sistema cumpla su objetivo. A continuación se muestran algunos de ellos:

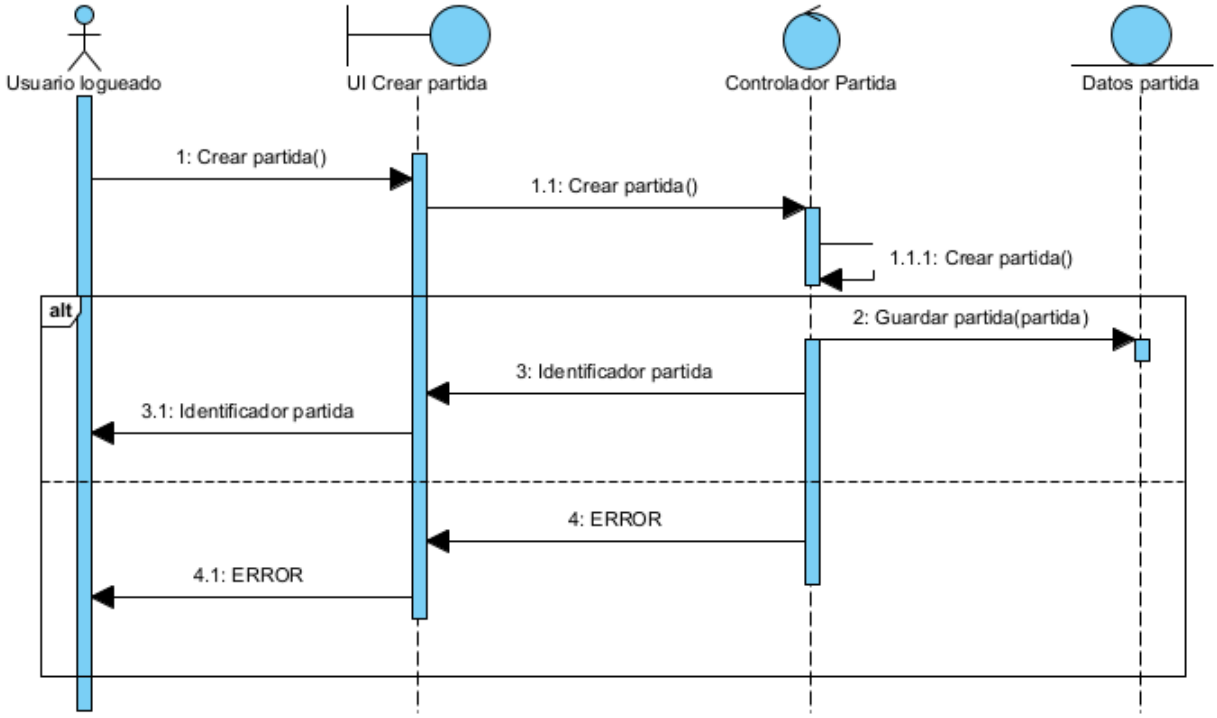


Figura 20: Diagrama de secuencia de crear partida

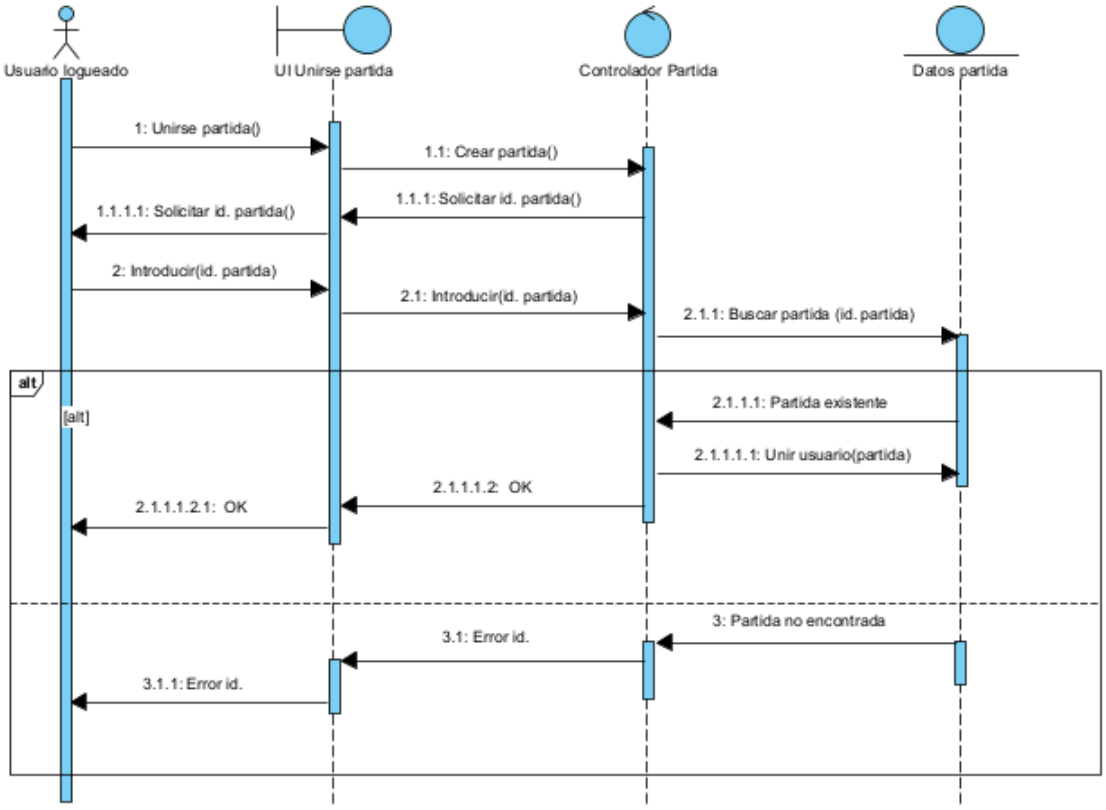


Figura 21: Diagrama de secuencia de unirse a la partida

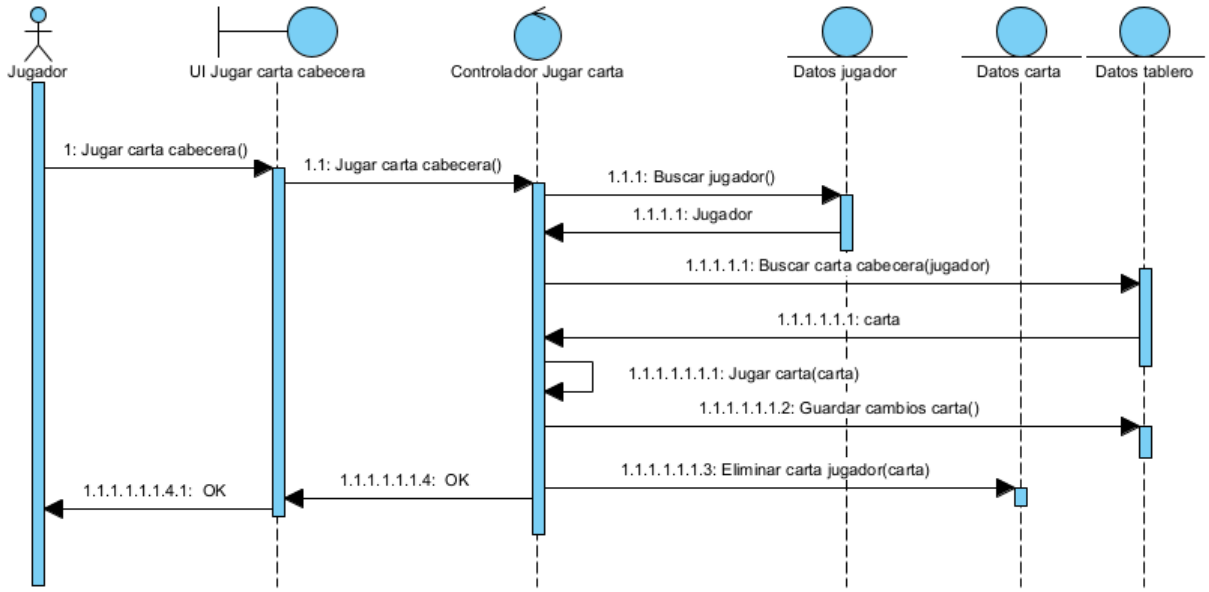


Figura 22: Diagrama de secuencia de jugar carta de cabecera

6.1.3. Diseño centrado en el usuario

El diseño centrado en el usuario es un proceso iterativo que aúna los objetivos en las necesidades de los usuarios para mejorar su experiencia. Los procesos necesarios para alcanzar estos objetivos se concentran en 3 fases:

- Fase de descubrimiento: Se busca entender el contexto de los usuarios que usan un sistema. Se define la audiencia del sistema y se analiza la existencia de competidores.
- Fase de conceptualización: Se unen y se trabajan las ideas obtenidas en la fase anterior, diseñándolas según las expectativas y motivaciones.
- Prototipado y pruebas de usuario: Se procede a crear el primer prototipo de la idea y mediante pruebas de usuario se modifica hasta poder obtener el primer prototipo funcional.

En el anexo V ”Diseño centrado en el usuario” se puede encontrar más información acerca del proceso para diseñar la interfaz.

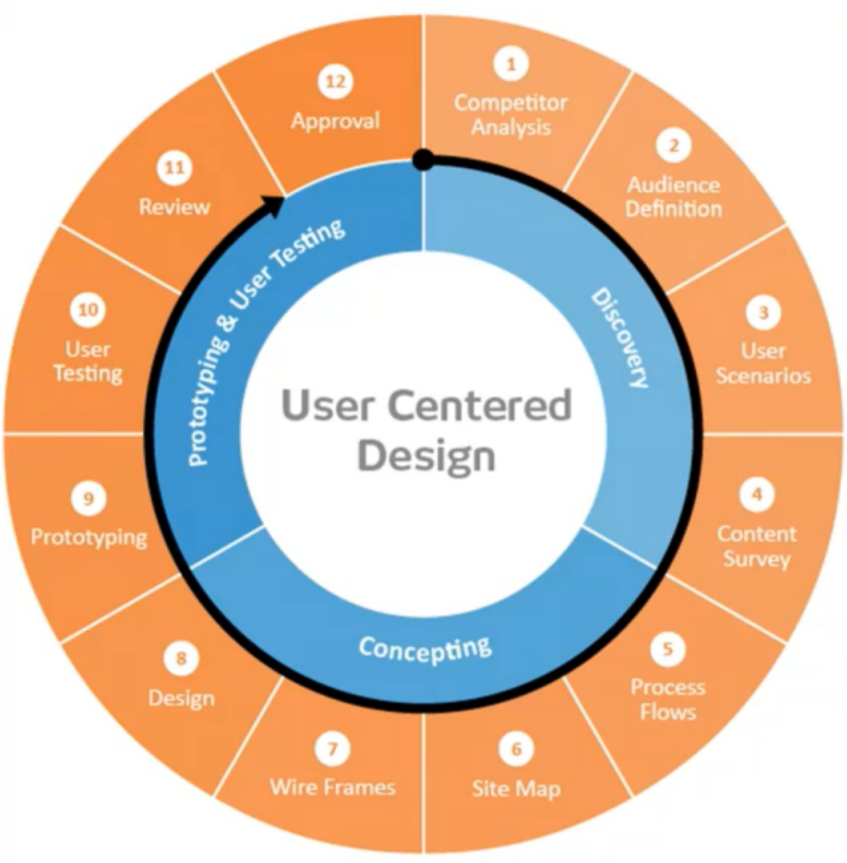


Figura 23: Diseño centrado en el usuario

Exploración

Realmente existe una gran cantidad de juegos de mesa digitalizados, incluso la app de Twilight Struggle se puede comprar a través de Google Store y App Store. Sin embargo, sigue siendo un juego que sólo permite dos jugadores y su ambientación puede resultar lejana para alguna audiencia. The Wall is Down cambia en que permite la participación de hasta 4 jugadores y se sitúa en una época con condiciones políticas, económicas y militares ambientada después de la guerra fría.

El juego, por lo tanto también la interfaz, está más enfocado a una audiencia adulta debido a la dificultad que presenta la comprensión de las reglas y la necesidad de ser capaz de formular tácticas y estrategias para poder jugar.

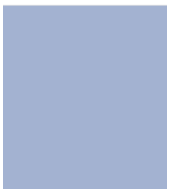
Diseño

Para la creación del diseño se intentó potenciar todo lo posible la calidad de la experiencia en la interacción persona-ordenador. Se seleccionó un tema que presentase una legibilidad de los textos clara, que fuese deferente hacia el usuario y que tuviese profundidad mediante el uso de capas visuales y de jerarquía.

La paleta de colores que se escogió fue la siguiente:

(a) Color fondo #CBC9D1

(b) Color tipografía #504f52



(c) Color componentes #A3B7D1

Figura 24: Paleta de colores

Esta selección se hizo así porque están presentes en el tablero del juego:

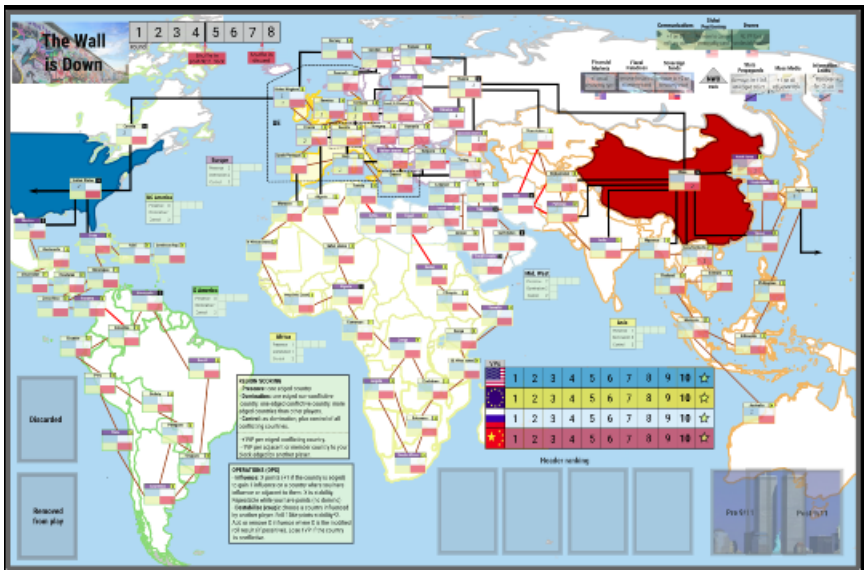


Figura 25: Tablero del juego

Se realizaron pruebas de los colores entre el fondo y la tipografía en la página de Adobe Color para comprobar el contraste establecido por WCAG sobre las relaciones mínimas de un texto legible con nivel AA. De esta forma se asegura la accesibilidad para daltónicos y que puedan experimentar la interfaz según lo previsto.

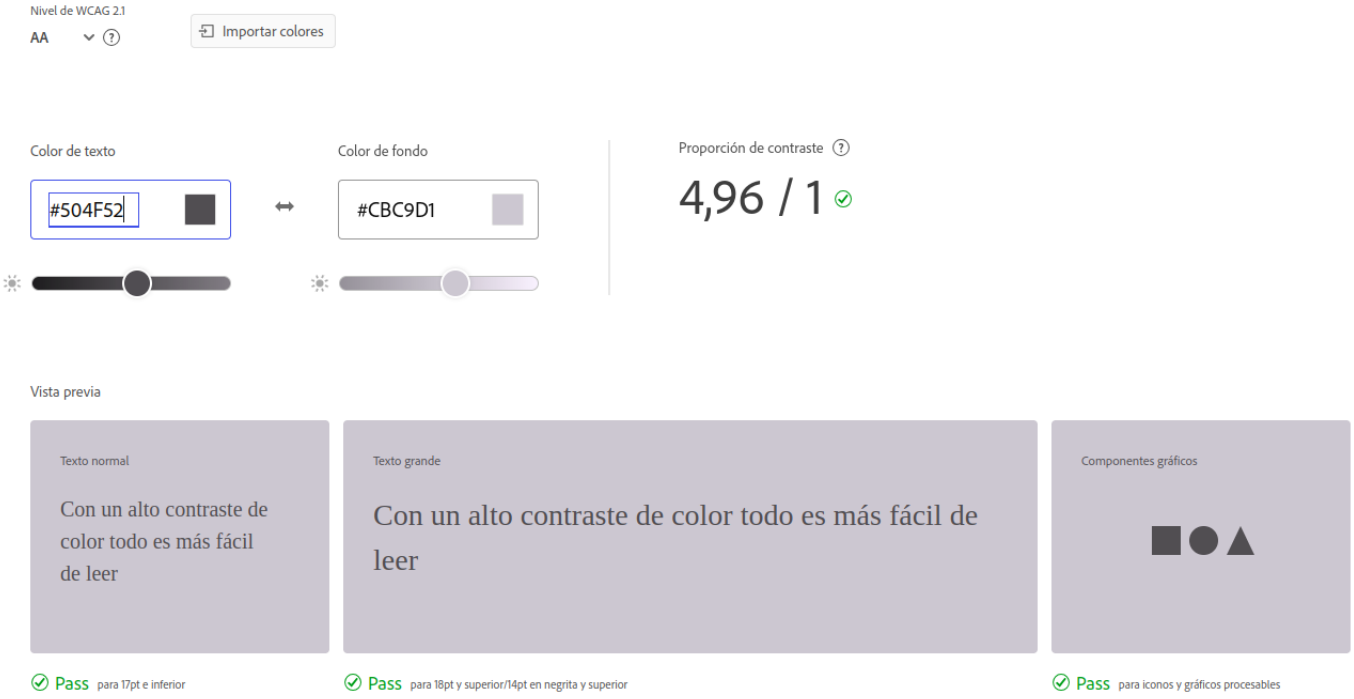


Figura 26: Validador contraste color tipografía

Como se puede observar en la imagen superior [26], se pasa la prueba de contraste con una proporción de 4,96/1.

En la siguiente [27] se ve que la paleta de colores es accesible para personas con daltonismo.

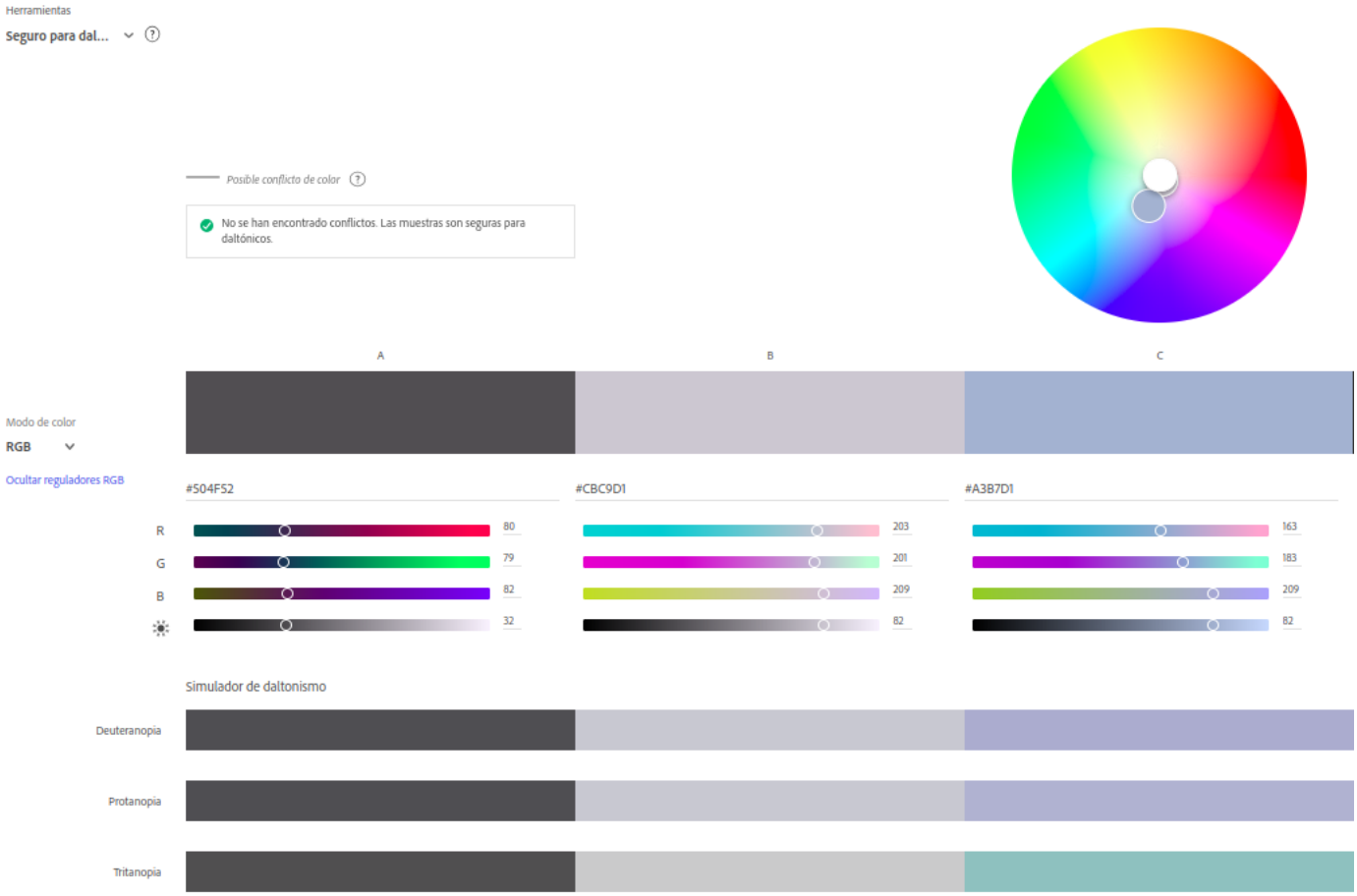


Figura 27: Validador accesibilidad daltonismo

Prototipado

El prototipo se llevó a cabo en la plataforma de Figma, para poder acceder y realizar una prueba se puede hacer a través del siguiente enlace: [link]. Dándole al botón de play en la parte superior derecha se abre una ventana que permite su navegación. Si es necesario, también se puede ajustar al tamaño de pantalla seleccionándolo en el menú desplegable de la parte superior derecha.

6.2. Implementación

Como se ha mencionado anteriormente, para que la interfaz funcione es necesario el uso de la infraestructura SOA creada por Javier Vidal Ruano. El sistema tiene por lo

tanto una arquitectura cliente-servidor, donde el servidor y el motor sería la API SOA y la interfaz el cliente que consume los servicios RESTful expuestos.

Para que fuera posible usar la interfaz a través del navegador se necesitó añadir en el código main.py de la API del directorio ./TWID/control/ el módulo "fastapi.middleware.cors" para manejar Cross-Origin Resource Sharing (CORS), que proporciona middleware para habilitar o restringir el acceso a la API desde diferentes orígenes. Fue necesario debido a la política same- origin, que es un mecanismo que la mayoría de navegadores modernos tienen para restringir las interacciones entre documentos, scripts o archivos multimedia procedentes de un origen distinto al de la página web.

Para implementar la interfaz se ha usado la librería de React con JavaScript. La estructura del proyecto es la siguiente:

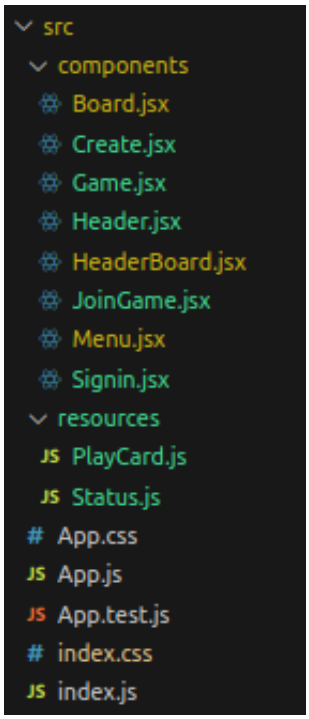


Figura 28: Estructura código fuente

Páginas web

Las páginas por las que se puede navegar en la interfaz están elaboradas mediante componentes JSX. Están recogidas en el directorio ./twid/src/components y se con-

forman por:

- Signin.jsx: Muestra la página de inicio de la interfaz en donde los usuarios pueden loguearse.
- Menu.jsx: Es la página principal del juego. Tiene un menú donde los usuarios pueden crear partidas nuevas, unirse a una creada por otro usuario y ver las partidas ya creadas.
- Create.jsx: Es la página que permite crear una partida nueva.
- Game.jsx: Muestra las partidas del usuario, tanto las que ha creado como a las que se ha unido. Existe la posibilidad de acceder a una de las partidas para ver los detalles o la opción de eliminarla
- JoinGame.jsx: Es la página que permite a los usuarios unirse a partidas ya creadas.
- Board.jsx: Es la página más extensa, ya que es donde se lleva a cabo la partida.

Componentes

Respecto a los componentes reutilizables, ubicados en el mismo directorio de componentes anterior, fue necesario crear dos:

- Header.jsx: Es la barra de navegación que permite volver a la página del menú y también hacer logout.
- HeaderBoard.jsx: Es la barra de navegación para la página del tablero que permite ver la información durante la partida, así como las cartas o la puntuación.

Enrutador

Para poder pasar de una página a otra fue necesario usar la librería React Router. Ésta hace posible definir rutas de navegación dentro de nuestro programa usando los componentes creados. El árbol de rutas está definido en ./twid/src/App.js

BroadcastChannel API

La interfaz está preparada para ser usada de forma local, pero para poder probarla es necesario jugar o emular una partida. Se necesita una forma de mandar avisos o eventos a los otros jugadores, para indicarles la llegada de su turno. Aquí es donde entra BroadcastChannel API.

Es una API sencilla que permite a los contextos de navegación comunicarse como ventanas, pestañas, frames o iframes. Funciona creando canales donde se pueden mandar mensajes para que lleguen a todos los oyentes excepto al emisor. Cuando éste se manda se lanza un evento que provoca una acción en los oyentes.

De esta forma, los jugadores pueden conocer la llegada de su turno cuando el anterior manda el mensaje por el canal.

Llamadas a la API

Para realizar todas las llamadas a la API se usó el objeto XMLHttpRequest de JavaScript. Éste es un ejemplo de la petición POST para hacer login en el juego:

- var guest = new XMLHttpRequest();
- guest.open('POST', 'https://localhost:443/auth/signin/guest',false);
- guest.send();

Función PlayCard

Esta función es la que hace posible jugar las cartas. Es responsable de llamar a las distintas funciones que elaboran los datos que necesita la API, como por ejemplo defineTargetsText() o defineTargetsDestabilization(); después realizar las peticiones de jugar la carta según la opción que se haya escogido en la interfaz: jugar la carta por influencia o por desestabilización, jugar la carta por puntuación si ésta tiene una P, o en la fase de cabecera jugar la carta por texto.

Función getStatus

Esta función es muy usada en distintas partes del código, ya que su objetivo es obtener el estado actual del tablero. Cuando se realiza una llamada, según las opciones introducidas, se pueden obtener las cartas restantes de los jugadores, las cartas de cabecera que se han seleccionado para la ronda, el número de ronda, la fase actual (fase de cabecera o fase de acción), el estado de influencia de los jugadores en todos los países, el estado del New World Order, etc. Para obtener toda esta información se hacen peticiones GET a la API.

En el anexo V "Documentación técnica de programación" se puede obtener más información acerca de la implementación de la interfaz y su puesta en marcha.

6.2.1. Dificultades durante la implementación

Al inicio de la implementación fue necesario investigar y dedicar tiempo al funcionamiento de la API SOA y de cómo realizar las llamadas. Entender cómo hacer peticiones mandando datos a través de las cabeceras supuso dedidar tiempo a mayores antes de poder empezar con algunas partes del diseño.

7 | Conclusiones

En este apartado se exponen las conclusiones acerca de la elaboración del Trabajo de Fin de Grado "Interfaz web para un juego de tablero online":

- Se ha digitalizado el juego The Wall is Down usando tecnologías conocidas y empleadas muy a menudo en la industria de hoy en día. Este tipo de proyectos permiten ser consciente de la importancia del desarrollo de las herramientas empleadas y pone en práctica las demandas de la industria del desarrollo web.
- Entender que en los proyectos de desarrollo software es imprescindible para su correcta elaboración la inclusión de metodologías que ayuden a planificar, gestionar y crear productos que satisfagan los requisitos de los clientes, las necesidades de los usuarios y faciliten el trabajo de los desarrolladores.
- Durante la elaboración del proyecto se ha tenido la oportunidad de aprender un lenguaje de programación que está al orden del día.
- La necesidad de tener un horario de trabajo al que atenerse y mantener la disciplina para cumplirlo es imprescindible para poder lograr los objetivos planteados.
- Finalmente, la interfaz online para el juego The Wall is Down ha sido un proyecto enriquecedor ya que ha hecho posible explorar una rama anteriormente desconocida, aprender conceptos del ámbito del frontend y entender la complejidad del diseño y desarrollo web.

8 | Líneas de trabajo futuras

Para finalizar se exponen posibles líneas de trabajo futuras del proyecto de cara a realizar mejoras en su funcionamiento.

- Añadir la funcionalidad de elegir país a influenciar o desestabilizar. En la interfaz cuando se desea jugar una carta de influencia o de desestabilización, el país donde se realiza la operación es seleccionado de forma arbitraria. La mejora consistiría en usar el tablero de forma interactiva donde el usuario pudiese escoger el país clicando, para añadir o eliminar influencia. Podría conseguirse a través del uso de librerías como D3.js
- Usar la interfaz para jugar desde máquinas distintas. Para llevar a cabo una partida actualmente, los jugadores deben realizarla desde la misma máquina y usando el mismo navegador; esto es así porque para compartir el inicio y fin de los turnos se mandan mensajes a través de los contextos de navegación. La mejora consistiría en crear un sistema donde el jugador pudiese avisar del fin de su turno para que el siguiente pudiese empezar a jugar, permitiendo a su vez que cada jugador lleve la partida en su propia máquina.

9 | Bibliografía

1. Graell, V. (2020, 7 abril). El Parchís, fenómeno de la cuarentena: casi tres millones de descargas de móvil en menos de un mes. ELMUNDO. [link]
2. ¿Qué es la arquitectura orientada a los servicios (SOA)? (s. f.). [link]
3. Rodrigo Santamaría Vicente. The Wall is Down. [link]
4. Javier Vidal Ruano. (2023, febrero). Arquitectura de servicios RESTful para un juego de tablero online. Trabajo de Fin de Máster. Universidad de Salamanca.
5. Mir, M. A. (2023, 2 abril). What are the advantages and disadvantages of using visual Studio code or Atom?. Medium. [link]
6. Next U. (2022, 20 julio). Ventajas y desventajas del Javascript, Next U. Blog, NextU LATAM. [link]
7. Zacharias, D. (2022, 23 septiembre). Why is JavaScript so popular? Code Power. [link]
8. ¿Qué es Figma? | La mejor herramienta de prototipado web. (s. f.). CEI: Escuela de Diseño y Marketing. [link]
9. A, D., & A, D. (2023). Qué es REAcT: definición, características y funcionamiento. Tutoriales Hostinger. [link]
10. Radigan, D. D. (s. f.). ¿Qué son los puntos de historia y cómo se estiman? Atlassian. [link]
11. Vige, W. (2022, 3 diciembre). Puntos de Historia: Guía para estimar las historias

de usuarios en Agile [2022] • Asana. [link]

12. Cohn, M. (s. f.). Sprint Backlog and the scrum sprint. Mountain Goat Software. [link]

13. App Store. (2016, 24 junio). Twilight Struggle. App Store. [link].

14. Twilight Struggle - aplicaciones en Google Play. (s. f.). [link]

15. Francisco José García Peñalvo, Alicia García Holgado, Andrea Vázquez Ingelmo. (2019/2020). Proceso Unificado.Departamento de Informática y Automática, Universidad de Salamanca. [link].

16. Francisco José García Peñalvo, Alicia García Holgado, Andrea Vázquez Ingelmo. (2019/2020). Modelo de dominio. Departamento de Informática y Automática, Universidad de Salamanca. [link]

17. Francisco José García Peñalvo, Alicia García Holgado, Andrea Vázquez Ingelmo. (2019/2020). Fundamentos de la vista de iteracción. Departamento de Informática y Automática, Universidad de Salamanca. [link]

18. Francisco José García Peñalvo, Alicia García Holgado, Andrea Vázquez Ingelmo. (2019/2020). Flujo de trabajo del proceso unificado. Departamento de Informática y Automática, Universidad de Salamanca. [link]