# INTERFAZ WEB PARA UN JUEGO DE TABLERO ONLINE

## Anexo IV: Documentación técnica de programación

Grado en Ingeniería Informática

TRABAJO DE FIN DE GRADO


Septiembre, 2023

**Autor:**

Helena Fajín Riveiro

**Tutor:**

Rodrigo Santamaría Vicente

# Índice general

**1. Introducción** **1**

**2. Estructura del sistema** **2**

2.1. Servidor . . . . . 2

2.2. Cliente . . . . . 2

**3. Interfaz** **4**

3.1. Páginas web . . . . . 4

3.2. Componentes . . . . . 5

3.3. Librería MUI . . . . . 6

3.4. Enrutador . . . . . 6

3.5. BroadcastChannel API . . . . . 6

3.6. Llamadas a la API . . . . . 7

3.7. Functiones getStatus y PlayCard . . . . . 7

**4. Puesta en marcha** **9**

4.1. Puesta en marcha de la API . . . . . 9

4.2. Puesta en marcha de la interfaz . . . . . 9

**5. Bibliografía** **11**

# Índice de figuras

1. Estructura sistema . . . . . 3

2. Código fuente . . . . . 4

3. Puesta en marcha API . . . . . 9

# 1 Introducción

En este anexo se recoge y explica la esctructura e implementación del proyecto, con el propósito de transmitir los conocimientos para su futuro mantenimiento a programadores o administradores, o también si se da el caso de necesidad de modificación o extensión.

Primero se comentará la estructura del sistema mencionando la parte del cliente y la del servidor, y en el siguiente apartado la de la implementación de la interfaz.

# 2 Estructura del sistema

## 2.1. Servidor

Este proyecto depende para su funcionamiento de una infraestructura SOA creada por Javier Vidal Ruano, la cuál elaboró como Trabajo de Fin de Master con título Arquitectura de servicios RESTful para un juego de tablero online. El código fuente se puede encontrar en la carpeta ./TWID-SOA.

## 2.2. Cliente

El código relacionado con el cliente se encuentra en la carpeta ./twid, donde a su vez los elementos más destacables se ubican en la carpeta ./twid/src, su estructura se puede ver a continuación:

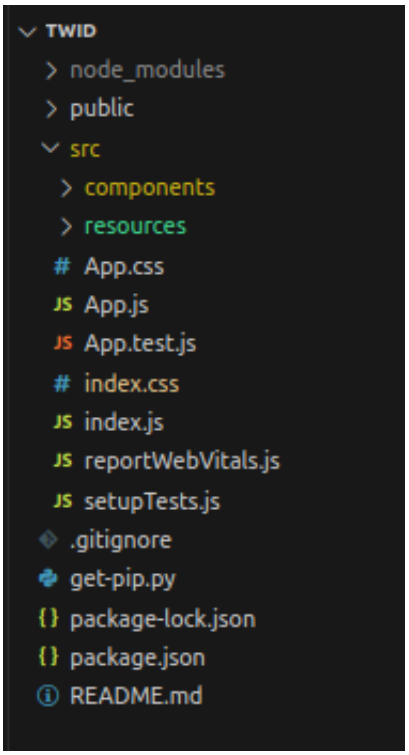


Figura 1: Estructura sistema

En la imagen [1] se puede ver el directorio src desplegado. El código fuente más importante se encuetra en las carpetas de components y resources.

- Directorio components: Recoge todos los componentes creados en JSX para formar los elementos de la interfaz.
- Directorio resources: aloja dos funciones que precisan los componentes para poder mostrar la información por pantalla. Ambas contienen llamadas a la API SOA.

# 3 Interfaz

A continuación se explicará la interfaz, la cuál ha sido implementada con JavaScript y desarrollada con la librería de React.

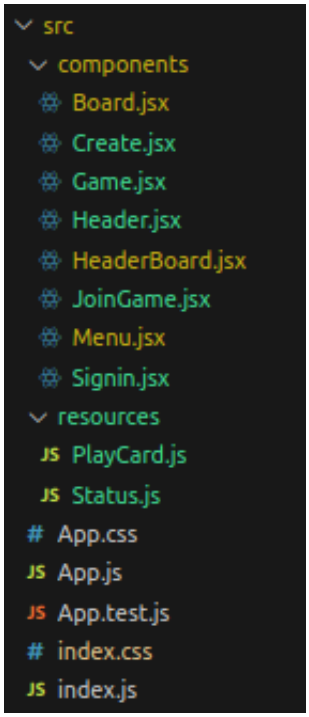


Figura 2: Código fuente

## 3.1. Páginas web

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

- Signin.jsx : muestra la página de inicio de la interfaz en donde los usuarios pueden loguearse. En ella se realiza una llamada a la API que devuelve un token único para poder identificar al usuario.
- Menu.jsx: representa 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, donde los usuarios deben escoger su jugador. Se realiza una llamada a la API que devuelve el identificador de la partida. Este se debe compartir con los jugadores que deseen unirse.
- Game.jsx: muestra las partidas del usuario, tanto las que ha creado como las que se ha unido.
- JoinGame.jsx: es la página que permite a los usuarios unirse a partidas ya creadas, para ello es necesario tener el identificador de la partida e introducirlo, seleccionar un jugador y darle al botón de unirse para realizar una petición a la API.
- Board.jsx: es la página más extensa, ya que es donde se lleva a cabo la partida. En ella se usan modales para jugar las cartas o ver la información del tablero.

## 3.2. 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, para lo que es necesario hacer una llamada a la API.
- HeaderBoard.jsx: Se creó otra barra para la página del tablero para poder ver su información durante la partida, como ver las cartas o la puntuación. También

tiene un botón como el componente anterior que hace una llamada a la API para hacer logout del juego.

## 3.3. Librería MUI

La librería MUI resultó de gran ayuda durante la implementacion, pues disponer de tantos componentes ya creados y diseñados donde personificarlos es realmente fácil, ahorró mucho tiempo. Además en la página de MUI se dispone de una documentación amplia y buena para entender el funcionamiento y su implementación en el código. Muchos de los componentes que se han usado han sido los modales, botones, barras de navegación, menús, etc.

## 3.4. Enrutador

Para poder pasar de una página a otra fue necesario usar la librería React Router. Esta 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

## 3.5. BroadcastChannel API

La interfaz está preparada para usarse de forma local, pero para poder probarla es necesario jugar o emular una partida. Para que fuera posible, 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 menos al emisor. Cuando este 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.

## 3.6. Llamadas a la API

Para realizar todas las llamadas a la API se uso el objeto XMLHttpRequest de JavaScript. Para mostrar un ejemplo, la siguiente peticion POST para hacer login en el juego:

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

## 3.7. Functiones getStatus y PlayCard

Estas dos funciones se encuentran en los directorios ./twid/src/resources/PlayCard.js y ./twid/src/resources/getStatus.js Ambas estan basadas en el código elaborado por el tutor Rodrigo Santamaría Vicente, que desarrolló con el proposito de hacer pruebas en la isfraestructura SOA. Este se puede encontrar en los directorios ./TWID-SOA/frontend/utils.py y ./TWID-SOA/frontend/test.py

#### PlayCard

Esta función es la que hace posible jugar las cartas. Es responsable de llamar a las distintas funciones que forman los datos que necesita la API, como por ejemplo defineTargetsText() o defineTargetsDestabilization() y 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 esta tiene una P, o en la fase de cabecera jugar la carta por texto.

**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.

# 4 | Puesta en marcha

## 4.1. Puesta en marcha de la API

Para iniciar la API de forma local, es necesario tener docker-compose instalado, y ejecutar la siguiente orden en un terminal en el directorio ./TWID:

- sudo docker-compose -f docker-compose.yml up –build –remove-orphans –force-recreate

Una vez activo se debe de ver de la siguiente forma:

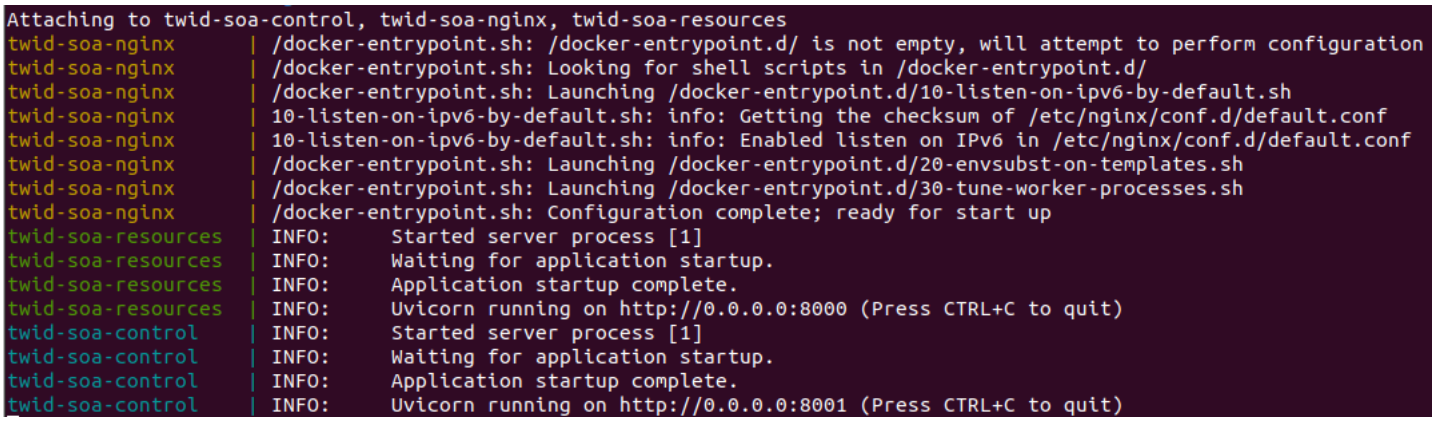


Figura 3: Puesta en marcha API

## 4.2. Puesta en marcha de la interfaz

Para poner en marcha la interfaz es necesario tener instalado node 18.17.1 y npm 9.6.7. Después dentro del directorio ./twid se deberán ejecutar las órdenes :

- npm install: para descargar las dependencias necesarias de node_modules.
- npm start: para lanzar la interfaz

Una vez hecho esto, dentro del navegador de Google Chrome se podrá acceder a http://localhost:3000/ y se visualizará la interfaz.

# 5 | Bibliografía


1. MUI, MUI CORE, [En línea]. Available: **link**. [Último acceso: septiembre 2022]
2. React, React Documentation, [En línea]. Available:**link**. [Último acceso: septiembre 2022]
3. MDN Web Docs, Broadcast Channel API, [En línea]. Available: **link**. [Último acceso: septiembre 2022]
4. React Router, Router Components, [En línea]. Available: **link**. [Último acceso: septiembre 2022]
5. W3schools, XML HttpRequest, [En línea]. Available: **link**. [Último acceso: septiembre 2022]
6. StackOverflow. Sitio web de apoyo Stack OverFlow, [En línea]. Available: **link**. [Último acceso: septiembre 2022]