Universitat Oberta de Catalunya
uoc.edu

Autor: Pedro Moreno Valdés
Tutor: Jordi Guirao Muns
Profesor: Víctor Garcia Font

Máster en Ciberseguridad y Privacidad
Sistemas Blockchain

13/06/2022

Créditos/Copyright

Notaciones y Convenciones

Arial 12 – Texto Normal
Arial 20 y en negrita – Texto Para títulos.
Times New Roman 8 – Código fuente.

FICHA DEL TRABAJO FINAL

Título del trabajo:	*DAPP Tres En Raya*
Nombre del autor:	*Pedro Moreno Valdés*
Nombre del consultor/a:	*Jordi Guirao Muns*
Nombre del PRA:	*Víctor Garcia Font*
Fecha de entrega (mm/aaaa):	06/2022
Titulación:	*Máster en Ciberseguridad y Privacidad*
Área del Trabajo Final:	*Sistemas Blockchain*
Idioma del trabajo:	*Castellano*
Palabras clave	*Blockchain, cadena de bloques, transacciones, juego, online, Tres en raya, fichas, jugadores, DAPP*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

En este proyecto final de Máster se realizará una aplicación web responsive (DAPP) la cual permitirá a dos usuarios poder jugar de manera síncrona una partida al famoso juego del tres en raya.

El objetivo principal del proyecto es el de crear una interfaz web (DAPP) que se comunique en su *background* con un contrato inteligente desarrollado en Solidity para la famosa red de bloques Ethereum.

Por otro lado, se busca poder emparejar jugadores mediante un hash generado de forma aleatoria y en la que un jugador genera un hash y el otro jugador solo se encargará de copiar y pegar el hash pasado por el creador de la partida con la finalidad de poder emparejarlos y realizar un juego.

Por último, a diferencia de un juego de tres en raya convencional que se puede encontrar por internet es que este no depende de una arquitectura convencional cliente-servidor. La arquitectura de este juego funciona a través de una red descentralizada P2P en la red de Ethereum.

Abstract (in English, 250 words or less):

In this final Master's project, a responsive web application (DAPP) will be launched which will allow two users play between them several games in the famous tic-tac-toe game simultaneously.

The main goal of this project is to create a web interface (DAPP) that communicates in the background with an intelligent contract developed in Solidity for the famous Ethereum block network.

On the other hand, it seeks to be able to match players through a randomly generated hash and in which a player generates a hash and the second one will paste the hash provided by the first player (Owner of the game) with the purpose of being able to match themself and make a game.

To sum up, In contrast with whole conventional tic-tac-toe games that can be found on the internet, this one does not depend on a conventional client-server architecture. The architecture of this game works through a decentralized P2P network on the Ethereum network.

Índice

Figuras y tablas

Índice de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

La blockchain es la nueva revolución digital, es la tecnología que nos ofrece transparencia, seguridad, trazabilidad e inmutabilidad. Aunque actualmente estemos empezando a oír el concepto *blockchain* sí que es verdad que esta tecnología aún se encuentra en una fase prematura. Se precisa de un desarrollo más avanzado para posibilitar su adopción a todos los sectores.

Teniendo en cuenta todas sus características anteriormente mencionadas la blockchain aspira a ser la palanca disruptiva de los modelos de negocio tal y como los conocemos hoy día y a su vez de la sociedad. Todo esto se debe a su gran poder de transformación de procesos y operaciones tradicionales que comúnmente son aceptados.

En este trabajo se pretende desarrollar el famoso juego del tres en raya ejecutándose en una red descentralizada como es Ethereum aprovechando el potencial que esta red nos ofrece con su lenguaje de programación dedicado a los contratos inteligentes que actualmente conocemos como Solidity.

El juego está pensado para ser jugado desde un *frontend* (DAPP) desarrollado en tecnologías webs la cual permitirá a diferentes usuarios crear partidas y poder jugar de manera síncrona. Esto se consigue ya que desde una DAPP podemos disponer de un código *backend* (contrato en Solidity) ejecutándose en una red descentralizada punto a punto.

1.2 ¿Qué es Blockchain?

La blockchain se podría definir como una especie de base de datos pública que se actualiza y se comparte en una serie de ordenadores conectados en red. (Internet tal y como lo conocemos hoy en día).

Block: viene del hecho que los datos y el estado se almacenan en lotes secuenciales o “bloques”. Por ejemplo: Si enviamos ETH a otra persona, los datos de la transacción deben añadirse a un bloque para que se realice con éxito.

Chain: Se refiere al hecho de que cada bloque hace referencia criptográficamente a su antecesor. Los datos del bloque no se pueden cambiar sin cambiar todos los bloques sucesivos. Lo que requeriría el consenso de la red entera.

Por tanto, la cadena de bloques (blockchain) se va generando a través de los distintos participantes que componen una red Blockchain. Cualquier registro de una cadena puede ser consultado, pero no borrado ni modificado.

Con todas estas características comentadas, podríamos decir que la blockchain tiene las siguientes propiedades:

- **Inmutable:** Cuando se crea un dato en la cadena es irreversible.
- **Transparente**: Todos los bloques son visibles y pueden consultarse.
- **Confianza:** Gracias al estado de consenso distribuido de la red P2P que la forma y en la cual se crean nuevos bloques, se garantiza en todo momento la confianza en el proceso de creación de los mismos.

1.3 ¿Qué es Bitcoin?

Bitcoin es una moneda de intercambio digital nacida en el año 2008. Se diferencia de otras monedas por no estar regulada por ningún gobierno o banco emisor (descentralizada) y usar la revolucionaria tecnología llamada Blockchain para su infraestructura y seguridad.

Esta moneda fue mencionada por primera vez en un documento técnico publicado por alguien con el seudónimo de Satoshi Nakamoto, de quien nunca se conoció su nombre real o si era una o varias personas. Aunque Bitcoin se creara en el 2008, decimos que empezó en 2009 porque es cuando el mismo **Satoshi Nakamoto** minó el bloque inicial de la cadena, conocido como **Bloque Génesis**.

1.4 ¿Qué es Ethereum?

Ethereum es la blockchain programable del mundo. Y podríamos decir que es la innovación de Bitcoin, pero con grandes diferencias.

Se ha visto que Bitcoin permite utilizar dinero digital sin proveedor de pago o bancos al igual que Ethereum. La gran diferencia reside en que Ethereum es programable, así que también se puede utilizar para diferentes activos digitales, incluido Bitcoin.

Esto también significa que Ethereum es más que pagos. Es un mercado de servicios financieros, **juegos** y aplicaciones donde no pueden robar información o censurar.

Más técnicamente hablando y ahora es cuando se verá la relación con la blockchain podríamos decir que en Ethereum hay un computador único y canónico (llamado máquina virtual de Ethereum o EVM), Cualquiera que participe en la red de Ethereum mantiene una copia del estado de este ordenador. Adicionalmente, cualquier nodo puede emitir una petición para que este ordenador realice un cálculo arbitrario.

Cuando se transmite una solicitud de este tipo, los demás nodos de la red verifican, validan y ejecutan el cálculo. Esto causa un cambio de estado en la EVM, que se realiza y propaga a través de toda la red.

Las peticiones de cálculo se llaman **solicitudes de transacción**; el registro de todas las transacciones así como el estado actual de la EVM se almacena en la **blockchain** que, a su vez, almacenan y acuerdan todos los nodos.

Los mecanismos criptográficos garantizan que, una vez que las transacciones se verifican y se añaden a la blockchain, ya no se pueden manipular.

1.5 ¿Qué es una DAPP?

Sabiendo que es la blockchain y habiendo explicado las diferencias entre Bitcoin y Ethereum y viendo que el segundo tiene mejores características que el primero por el solo hecho de tener la funcionalidad de poder crear contratos inteligentes con el lenguaje de Solidity.

Podríamos definir la DAPP (Decentralized Applications) como un tipo de aplicación cuyo funcionamiento no depende de servidores centrales, sino que funciona en base a una red descentralizada.

Las DAPPS permiten que las personas puedan acceder a distintos servicios de forma segura. Estas aplicaciones pueden ser utilizadas en cualquier tipo de dispositivo personal, como *smarthphones*, ordenadores o incluso ser accesibles vía web.

Como se ha visto, Ethereum tiene la capacidad de crear lo que llamamos contratos inteligentes, o "*smart contracts*". (Todos los programas cargados y ejecutados en la red).

A nivel muy básico, se puede pensar en un contrato inteligente como una especie de máquina expendedora: un script que, cuando se solicita con ciertos parámetros, realiza algunas acciones o cálculos si se cumplen determinadas condiciones. Por ejemplo, clicar una casilla del cuadro del tres en raya para validar una tirada y registrarla en la cadena de bloques.

Cualquier persona con conocimientos puede crear un contrato inteligente y hacerlo público en la red, usando la blockchain como su capa de datos, a cambio de una tasa/comisión pagada a la red. A continuación, cualquier usuario puede solicitar el uso del contrato inteligente para ejecutar su código, de nuevo, a cambio de una comisión pagada a la red.

En resumen, mediante los contratos inteligentes los desarrolladores pueden construir e implementar arbitrariamente complejas aplicaciones y servicios orientados al usuario: mercados, instrumentos financieros, juegos, etc.

1.6 Objetivos del proyecto

El objetivo principal del proyecto es llevar a cabo la creación del famoso juego del 3 en raya en la red descentralizada de Ethereum.
El *backend* del proyecto que contiene toda la lógica del juego y contenidos esta desarrollada en el lenguaje de programación base que ofrece Ethereum 2.0 que es Solidity en una de las versiones más actuales de máquina virtual del mismo.

Así mismo, otro de los objetivos es desarrollar una DAPP (*frontend*) responsive para poder interactuar con el contrato inteligente en Solidity (backend) utilizando la librería web3.js y tecnologías web como: HTML / JS / CSS / Bootstrap.

Por último, hay que destacar que uno de los hitos de este proyecto es que dos jugadores puedan encontrarse y poder realizar partidas de forma síncrona.

1.6.1 Objetivos generales

- Desarrollar el *smart contract* en Solidity que contenga la lógica del juego Desplegar el contrato en local mediante *truffle* y ganache. Posteriormente probarlo en *testnet Kovan*.
- Crear la interfaz web en local utilizando *Webpack* con librería web3.js/ethjs.js y tecnologías como: HTML / JS / Bootstrap...
- Probar que dos jugadores pueden jugar entre ellos con algún ID generado de juego.
- Sitio *responsive*, para ser visto en dispositivos pequeños
- Temporizador entre tiradas contando el tiempo de ejecución del bloque y que el juego tenga en cuenta cuando pasar el turno al siguiente jugador en caso de que uno decida por no tirar.

1.6.2 Objetivos principales

Objetivos de la aplicación/producto/servicio:

- Crear una gran comunidad de usuarios que puedan divertirse jugando al tres en raya sin ánimo de lucro.

Objetivos para el cliente/usuario:

- Poder enlazar dos usuarios en una partida.

Objetivos personales del autor del TF:

- Poner en práctica todos los conocimientos aprendidos en el Master en la realización de un contrato inteligente y realizando una DAPP para interactuar con el mismo.
- Adquirir nuevos conocimientos con la realización del proyecto.
- Afrontar y dar solución a todos los posibles problemas que puedan surgir durante la ejecución del proyecto.

1.7 Enfoque y método seguido

La finalidad de este proyecto es el desarrollo de una aplicación web (DAPP) responsive que ofrezca compatibilidad con los diferentes dispositivos que podemos encontrar en nuestro día a día, como ordenadores de escritorio, dispositivos móviles, tabletas….

El funcionamiento de backend con toda la estructura del juego estará desarrollada en Solidity para la red descentralizada Ethereum y enlazada mediante el servicio *webpack* y la librería web3.js. Es por eso, que se ha optado a la utilización de tecnologías web para el desarrollo de este proyecto.

La estrategia ha sido desarrollar un nuevo producto, dado que en internet tenemos muchos juegos multijugador que dependen de un servidor central y el objetivo de este proyecto es crear el juego del tres en raya de manera descentralizada.

La metodología de trabajo escogida es en cascada ya que los objetivos, requisitos y fechas están perfectamente definidos desde el principio del proyecto. Dividiendo el proyecto en las siguientes fases: Requisitos, Diseño, Implementación, Verificación y Mantenimiento.

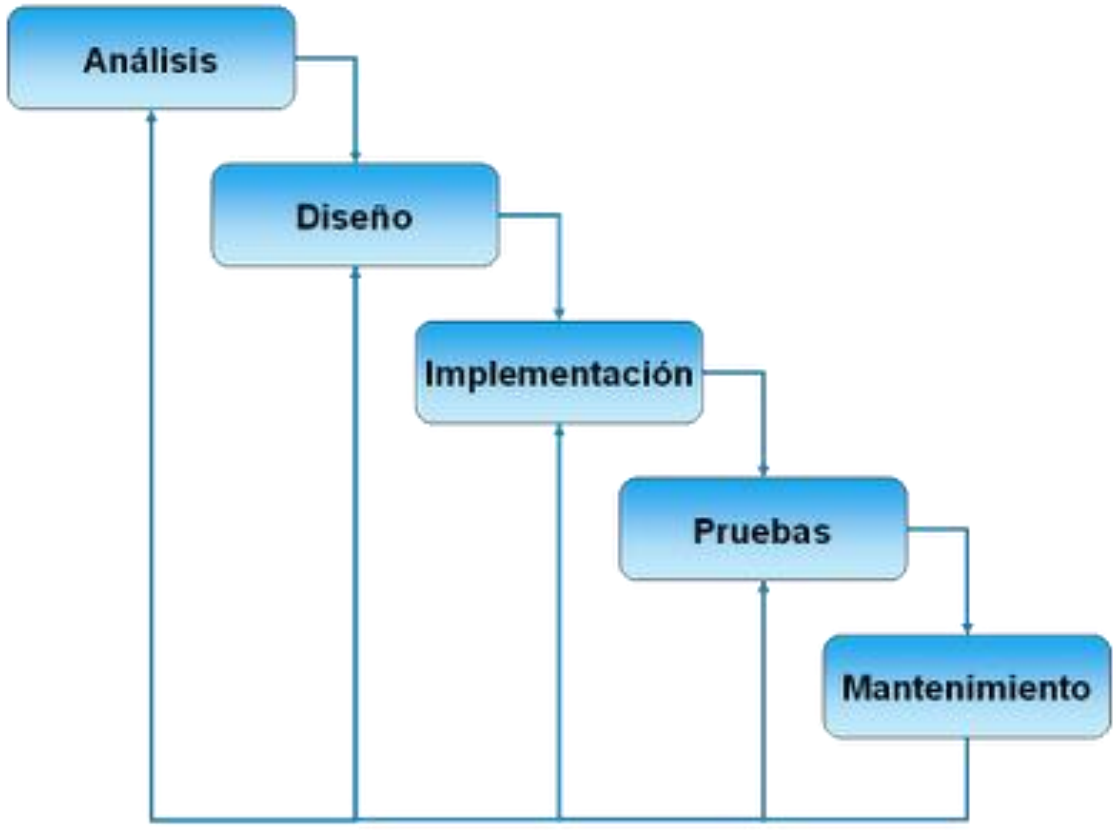


Figura 1: Metodología en cascada.

1.8 Planificación

En el punto de planificación se describen los materiales y recursos que se utilizarán para el desarrollo del proyecto, las tareas en las que se dividirá y se marcaran los tiempos en un diagrama Gantt de planificación temporal sobre cada una de las tareas y subtareas.

Tareas:

Nombre de tarea	Duración	Comienzo	Fin
Proyecto Tres En Raya	118,04 días	mi. 16/02/22 0:00	sá. 24/06/23 11:01
Plan de trabajo	**13,96 días**	**mi. 16/02/22 0:00**	**ma. 01/03/22 23:12**
Elección del tema	1 día	mi. 16/02/22 0:00	ju. 17/02/22 0:01
Búsqueda de proyectos similares	2,33 días	ju. 17/02/22 0:01	sá. 19/02/22 8:02
Definir las herramientas necesarias	2,96 días	sá. 19/02/22 8:02	ma. 22/02/22 7:04
Enfoque y método	1 día	ma. 22/02/22 7:04	mi. 23/02/22 7:05
Elaboración de la memoria PAC 1	6,29 días	mi. 23/02/22 7:05	ma. 01/03/22 14:06
Definición de los requisitos	**27,95 días**	**mi. 02/03/22 0:00**	**ma. 29/03/22 23:15**
Definición detallada de requisitos	2,62 días	ma. 01/03/22 14:06	vi. 04/03/22 5:07
Definición de tareas y planificación	4,62 días	sá. 05/03/22 5:07	mi. 09/03/22 20:09
Definición de objetivos	3,62 días	mi. 09/03/22 20:09	do. 13/03/22 11:06
Definición de alcance de proyecto	3,96 días	mi. 16/03/22 21:03	do. 20/03/22 20:10
Elaboración de la memoria PAC 2	8,62 días	do. 20/03/22 20:04	ma. 29/03/22 11:07
Diseño	**26,98 días**	**mi. 30/03/22 0:00**	**ma. 26/04/22 0:02**
Definición perfiles de usuario	3,95 días	ma. 29/03/22 11:07	sá. 02/04/22 10:02
Diseño conceptual	4,62 días	ma. 05/04/22 10:11	do. 10/04/22 1:10
Prototipo	2,96 días	lu. 11/04/22 1:12	ju. 14/04/22 0:18
Casos de uso	4,62 días	vi. 15/04/22 0:13	ma. 19/04/22 15:09
Descripción de la arquitectura	2,62 días	mi. 20/04/22 15:13	sá. 23/04/22 6:09
Elaboración memoria PAC 3	1,96 días	do. 24/04/22 6:15	ma. 26/04/22 5:16
Implementación	**34,61 días**	**mi. 27/04/22 5:16**	**ma. 31/05/22 20:23**
Instalación de todo el software necesario	2,63 días	ma. 26/04/22 5:16	ju. 28/04/22 20:25
Desarrollo de la DAPP	**27,13 días**	**sá. 30/04/22 20:23**	**vi. 27/05/22 23:59**
Back-End	14,62 días	sá. 30/04/22 20:23	do. 15/05/22 11:31
Front-End	9,63 días	do. 15/05/22 11:31	mi. 25/05/22 2:44
Memoria Final	5 días	ju. 26/05/22 0:00	ma. 31/05/22 0:02
Documentación Memoria Final (PAC4)	**2,96 días**	**mi. 25/05/22 2:44**	**sá. 28/05/22 1:51**
Revisión de la documentación	0,96 días	lu. 30/05/22 1:42	ma. 31/05/22 0:47
Entregable del Producto	1 día	ma. 31/05/22 0:45	mi. 01/06/22 0:46
Vídeo de presentación	**6 días**	**mi. 01/06/22 8:45**	**ma. 07/06/22 8:51**
Elaboración de diapositivas	2 días	mi. 01/06/22 8:45	vi. 03/06/22 8:47
Video + Entregable	3,29 días	sá. 04/06/22 8:00	ma. 07/06/22 15:00
Defensa Proyecto	**11,74 días**	**lu. 13/06/22 6:05**	**vi. 24/06/22 23:59**

Figura 2: Diagrama de GANTT.

Diagrama de Gantt en el **ANEXO 1.** Se han dividido las tareas de planificación partiendo de la metodología en cascada. En cada fase del desarrollo del proyecto se han creado subpuntos dentro de las EDI con la especificación en detalle de la acción/acciones a realizar.

Tal como se puede ver, en cada tarea se muestra la fecha de inicio y fin para cada una. En esta estimación aparece el número total de horas aproximadas con decimales que se dedicaran por fase. Por otro lado, en los títulos

principales (“marcados en negrita”) se marca el período inicial total y fecha final de todo el conjunto.

Seguramente haya dificultades e imprevistos a la hora de desarrollar el proyecto. Sobre todo, en las fases de diseño e implementación. En tal caso, los días se verían modificados acortando tiempo de otras tareas.

1.9 Entregables del Proyecto

Una vez definida la planificación, se pueden extraer los siguientes entregables en cada una de las etapas:

Planificación:

La propia planificación se compone de los siguientes entregables:

- Concepto de la aplicación
- Planificación
- Descripción de la aplicación
- Análisis de otras aplicaciones similares
- Costes aproximados
- Tecnologías que se van a utilizar.

Diseño:

- En la etapa de diseño se obtendrá la memoria del diseño de todos los aspectos que componen la aplicación:
- Usuarios que utilizarán la aplicación
- Casos de Uso
- Navegación de la app

Implementación:

- Código fuente de toda la DAPP:
- Back-End
- Front-End

Final de Proyecto:

Se obtendrán los siguientes entregables:

- Código fuente
- Publicación del contenido en test network (Kovan)
- Vídeo de presentación del proyecto.
- Defensa del Proyecto

1.10 Análisis de mercado

Un análisis de mercado sirve de base para la toma de decisiones de compra y venta. Aunque también sirve para evaluar el estado del mercado actual o de posibles nuevos mercados.

Actualmente los videojuegos están a la orden del día y aunque podemos seguir viendo muchos videojuegos online no nos haríamos la idea de cuantos de ellos están empezando a funcionar con la Blockchain.
En un pasado remoto jugábamos a videojuegos de recreativas introduciendo monedas, después vimos al avance de los juegos free top lay y actualmente estamos en la era del play to earn o jugar para ganar.

Los juegos play to earn suelen ser juegos basados en blockchain, es decir, ganar una serie de ítems que son únicos y que se pueden vender, comprar e intercambiar como sería el caso de los NFT (Tokens no fungibles).

Según datos de DappRadar 900.00 usuarios juegan a diario a juegos que están posicionados en la blockchain. Al utilizar la tecnología NFT estos juegos son capaces de generar 2.320$ millones en ventas durante los pasados meses de julio, agosto y setiembre de 2021.

A continuación se expondrán algunos juegos más jugados que utilizan la blockchain y con sus correspondientes usuarios diarios que los frecuentan.

Juego:	Usuarios diarios:	Foto:
Splinterlands:	427.800	
Alien Worlds	278.390	
Axie Infinity	90.000	

Farmers World	51.390	
CryptoMines	50.480	

Figura 3: Tabla de juego en blockchain con usuarios

Según datos de Bloomberg, Pelham Smuthers y GamingScan.com El mercado mundial de los juegos tiene actualmente un valor de $180.000 millones y es la forma de entretenimiento de más rápido crecimiento a nivel mundial. A continuación se puede observar el gráfico de evolución que lo demuestra:

Figura 4: Tabla gráfica del valor del videojuego

Ingresos del mercado mundial de juegos. Fuente: Bloomberg, Pelham Smithers, GamingScan.com

Por tanto, La industria de los videojuegos no para de crecer y en el futuro, muy probablemente, este crecimiento irá de la mano de la tecnología blockchain.

Es por eso que en este proyecto optamos por realizar un pequeño juego para la blockchain con posibilidades de poder expandir su estrategia a futuro.

El tres en raya al no ser un producto comercializado no tiene una competencia directa, pero sí que el objetivo es destacar de todo lo que pueda haber ahora mismo por la red y de la misma temática. Es por eso, que se analizará este apartado de manera superficial por tal de ver todos sus puntos fuertes/débiles, así como, su target, perfiles de usuarios y otros derivados.

1.10.1 Público objetivo

El público objetivo del tres en raya es muy amplio y variado, que podría comprender todas las edades. Teniendo en cuenta quien utilizará la aplicación, el target principal del juego son personas que tienen nociones sobre la blockchain o hace algún tipo de transacciones financieras o; simplemente, curiosos.

Resumiendo, el **público objetivo** sería:

- **Target:** Cualquier persona / Entendido de la *blockchain* / curiosos e inversores.
- **Rango de edad:** > 6
- **Sexo:** Neutro
- **Conocimientos:** Personas que sepan utilizar un ordenador o un dispositivo móvil/tableta.
- **Geografía:** Tanto ámbito nacional como internacional

1.10.2 Público potencial

El público potencial es uno de los aspectos fundamentales que se tiene en cuenta siempre en toda entidad económica. Es aquí donde radica la oportunidad de crecimiento y desarrollo.

En este caso, el juego del Tres en Raya al no ser un producto comercial en el cual no habrá ningún tipo de retribución económica, se hace hincapié en este apartado (muy superficialmente) sobre su principal público potencial que se ha considerado y del cual podrían beneficiarse algunas personas o instituciones.

Como **público potencial** nos podemos encontrar varios supuestos, pero estos son los que se proponen:

- **Personas con conocimientos blockchain**: Colectivo que domina superficialmente la tecnología, ya sea porque han invertido en algún tipo de criptomoneda/contratos inteligentes o están aprendiendo a interactuar con esta tecnología.
- **Amantes del *gaming*:** Este colectivo es muy amplio y repartido geográficamente. Muchos *gamers* pueden acceder al juego y realizar partidas tantas veces como quieran.

1.11 Estado del arte:

Existen numerosos juegos multijugador que sean de la temática del Tres en Raya pero si observamos el funcionamiento de estos, la gran mayoría son juegos centralizados y siempre dependen de un servidor para procesar y devolver las peticiones.

Si buscamos por juegos del Tres en Raya que se juegue de forma descentralizada, no hay mucha variedad pero aun así se pueden encontrar algunos que ya están subidos a plataformas de producción.

El siguiente juego funciona a través de la blockchain utilizando la red Ethereum a través de HIVE.

https://tic-tac-toe.mahdiyari.info/

A continuación veremos algunas ventajas de usar un sistema descentralizado frente un sistema centralizado:

Característica:	Centralizado	Blockchain
Administración de la información	Existe un administrador de la información	La información se encuentra descentralizada.
Sistema de Seguridad	El administrador debe implementar un sistema de seguridad con la finalidad de proteger la información.	Existe un sistema criptográfico, el cual puede variar a través de los mecanismos de claves públicas y privadas
Transparencia	El administrador establece los mecanismos por medio de los cuales los participantes acceden a la totalidad de la información dentro de protocolos establecidos para tal fin.	Los Participantes del sistema tienen la posibilidad de acceder a la información y verificarla a través de la cadena de bloques.
Costos	Se materializan costos por razón de la infraestructura tecnológica y en materia de la ciberseguridad que requiere el administrador central en el manejo de la información.	Existe una reducción de costos, ya que el manejo de la información es emplazado por códigos algorítmicos, los cuales, a través de nodos, procesan y verifican la información de forma independiente de cada transacción.
Alterabilidad de la información	Depende de los sistemas tecnológicos de ciberseguridad con que cuenta el administrador, los cuales no son inmunes a ataques cibernéticos.	Al existir una descentralización de la información, la cual está organizada en bloques por medio de procesos algorítmicos, la manipulación y alteración de dicha información es difícil de realizar.

Figura 5: Sistema Centralizado vs Descentralizado.

Fuente: Elaborada sobre la base de Preukschat (2017).

2. Diseño Conceptual

2.1 Definición de la aplicación

Se desarrolla una aplicación web *responsive (DAPP)* que permite la integración con cualquier tipo de resolución de pantalla y de sistema operativo. Además se desarrolla un *smartcontract* que contiene toda la lógica del juego.

La DAPP permite Crear Partidas, Encontrar partidas existentes y jugar partidas. Todo ello es llevado de manera descentralizada gracias a un contrato inteligente desarrollado en Solidity y publicado en local mediante Ganache y en *testnet Kovan Network*.

Existen tres perfiles en la aplicación. Jugador 1 (Creador de la partida), Jugador 2 (Jugador que se une a la partida), Espectadores (Pueden ver la partida pero sin interactuar con el juego).

2.2 Diseño Centrado en el usuario

La mayoría de los usuarios, esperan que la *interface* de las ***app's*** sean amigables y fáciles de utilizar. Por tanto, es importante realizar un análisis previo del diseño a realizar con el fin de cumplir este requerimiento.

El diseño centrado en el usuario es un punto muy importante, ya que son estos quienes realmente utilizarán la aplicación. Se debe hacer hincapié en introducir al usuario en la fase de diseño, ya que es este el actor principal sobre la aplicación. Con el fin de poder consensuar la interacción usuario-aplicación.

Es por eso, que en los apartados: 1.6.1 Objetivos generales y 1.6.2 Objetivos principales del proyecto se ha hecho un análisis sobre el **público objetivo** y **público potencial** para los que se espera que vaya a ir destinada la aplicación.

2.3 Casos de uso

A continuación se incluye el diagrama de casos de uso de la aplicación, distinguiendo las funcionalidades del rol de usuario que la aplicación contempla.

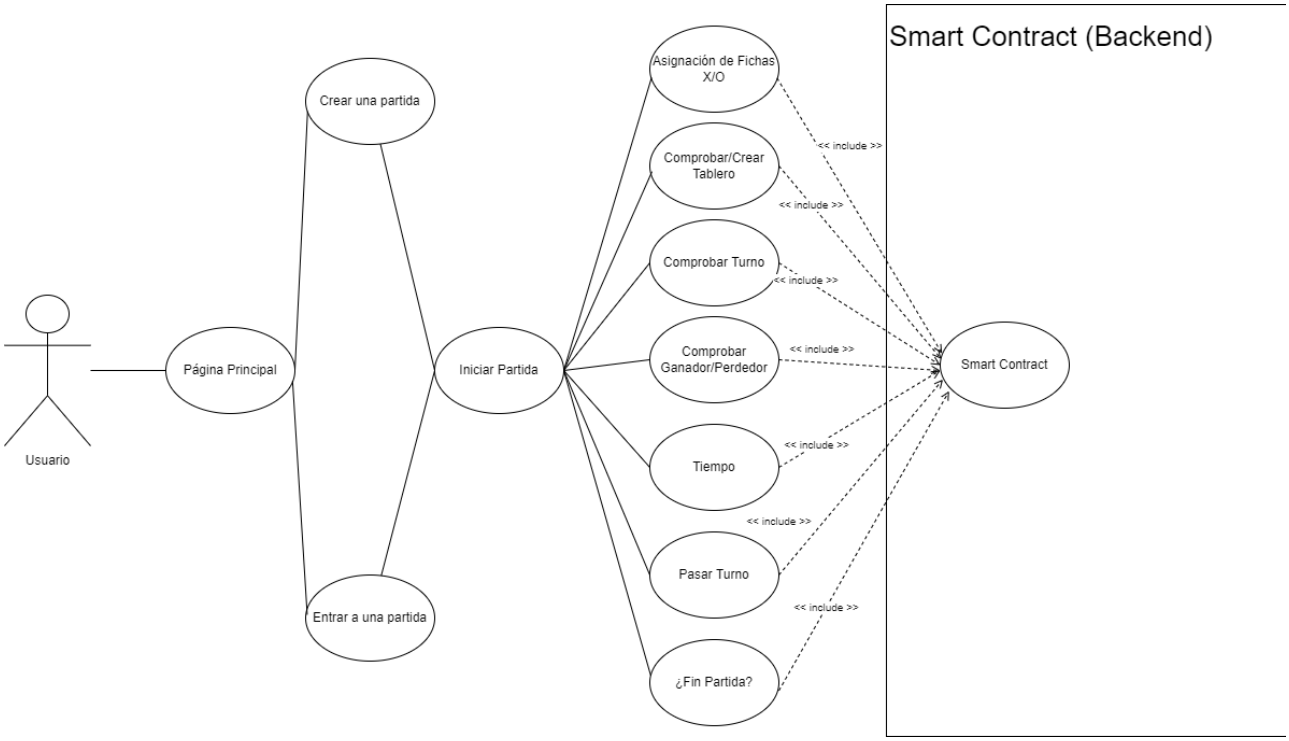


Figura 6: Diagrama Caso de uso

- **Definiciones:**

CU_01: Página Principal	
Nombre:	Página Principal
Actores:	Usuario con wallet metamask
Objetivos:	Visualizar el contenido de página para poder proseguir con los demás CU.
Precondición:	1) Entrar a la página web. 2) Tener metamask con una cuenta en activo
Postcondición:	El usuario podrá visualizar el contenido de la D'APP además de la pantalla de bienvenida, mostrando su wallet seleccionada en metamask (usuario) y además podrá crear partidas o Entrar en partida existente.
Escenario básico:	- El usuario entra en la D'APP - En caso de tener Wallet compatible con Metamask podrá crear y/o entrar en partida.

Figura 7: Caso de uso 01 – Página Principal

CU_02: Crear Partida	
Nombre:	Crear Partida
Actores:	Usuario con Wallet en Metamask
Objetivos:	Poder crear una partida a través de la generación de un contrato inteligente.
Precondición:	1) Entrar a la página web. 2) Tener metamask con una cuenta en activo
Postcondición:	El usuario al clicar en “Crear Partida” generará un hash de contrato que será el identificador de esa partida y estará dispuesto a realizar la primera tirada .
Escenario básico:	- El usuario entra en la D’APP - En caso de tener Wallet compatible con Metamask podrá generar un hash de partida - Empezar el juego

Figura 8: Caso de uso 02 – Crear Partida

CU_03: Entrar en Partida	
Nombre:	Entrar en Partida
Actores:	Usuario con Wallet en Metamask
Objetivos:	Poder acceder a una partida que previamente un usuario haya creado a través de la generación de un contrato inteligente.
Precondición:	1) Entrar a la página web. 2) Tener metamask con una cuenta en activo 3) Introducir hash del contrato pasado por el jugador 1.
Postcondición:	El usuario al clicar en “Entrar Partida” generará un hash de contrato que será el identificador de esa partida y estará dispuesto a realizar la primera tirada .
Escenario básico:	- El usuario entra en la D’APP - En caso de tener Wallet compatible con Metamask

	podrá introducir un hash de partida. - Empezar el juego

Figura 9: Caso de uso 03 – Entrar en Partida

CU_04: Asignación de Fichas	
Nombre:	Asignación de Fichas
Actores:	Usuario con Wallet en Metamask
Objetivos:	Asignar la ficha del jugador según sea el creador de la partida o el usuario que se une.
Precondición:	1) Entrar a la página web. 2) Tener metamask con una cuenta en activo 3) Introducir hash del contrato pasado por el jugador 1. O haberse unido en partida como jugador 2.
Postcondición:	Si ya hay una partida en juego, el jugador 1 (creador de la partida) siempre se le asignará “X”. Al jugador 2 se le asignará “O”.
Escenario básico:	- El usuario entra en la D’APP - El usuario crea o se une a partida - Empezar el juego - Jugador 1: X, Jugador 2: O

Figura 10: Caso de uso 04 – Asignación de fichas

CU_05: Crear Tablero/Comprobar Tablero	
Nombre:	Crear Tablero / Comprobar Tablero
Actores:	Usuario con Wallet en Metamask
Objetivos:	Poder generar/actualizar el tablero de la partida que previamente un usuario haya creado a través de la generación de un contrato inteligente.
Precondición:	1) Entrar a la página web. 2) Tener metamask con una cuenta en activo 3) Crear una partida / contrato

Postcondición:	El usuario al entrar en partida verá un tablero vacío el cual se va rellenando conforme a los jugadores les llegue su turno.
Escenario básico:	- El usuario entra en la D'APP - En caso de tener Wallet compatible con Metamask crea o entra en partida. - Empezar el juego / Realizar tirada.

Figura 11: Caso de uso 05 – Crear tablero / Comprobar tablero

CU_06: Comprobar turno	
Nombre:	Comprobar Turno
Actores:	Usuario con Wallet en Metamask
Objetivos:	Poder saber quién es el jugador con privilegios de tirada.
Precondición:	1) Entrar a la página web. 2) Tener metamask con una cuenta en activo 3) Crear una partida / contrato
Postcondición:	El usuario al entrar en partida verá reflejado si es su turno y la ficha correspondiente a ese turno.
Escenario básico:	- El usuario entra en la D'APP - En caso de tener Wallet compatible con Metamask crea o entra en partida. - Empezar el juego / Realizar tirada.

Figura 12: Caso de uso 06 – Comprobar turno

CU_07: Comprobar Ganador / Perdedor	
Nombre:	Comprobar Ganador/Perdedor
Actores:	Usuario con Wallet en Metamask
Objetivos:	Poder saber el estado de la partida y ver si existe un ganador.
Precondición:	1) Entrar a la página web. 2) Tener metamask con una cuenta en activo

	3) Crear una partida / contrato 4) Realizar jugadas
Postcondición:	El usuario estará informado en todo momento de si hay ganador o no.
Escenario básico:	- El usuario entra en la D'APP - En caso de tener Wallet compatible con Metamask crea o entra en partida. - Empezar el juego / Realizar tirada.

Figura 13: Caso de uso 07 – Comprobar Ganador / Perdedor

CU_08: Tiempo	
Nombre:	Tiempo
Actores:	Usuario con Wallet en Metamask
Objetivos:	Poder saber el tiempo entre turnos
Precondición:	1) Entrar a la página web. 2) Tener metamask con una cuenta en activo 3) Crear una partida / contrato 4) Realizar jugadas
Postcondición:	El usuario estará informado en todo momento del tiempo de tirada del que dispone.
Escenario básico:	- El usuario entra en la D'APP - En caso de tener Wallet compatible con Metamask crea o entra en partida. - Empezar el juego / Realizar tirada. - Tiempo entre tirada > 120 segundos.

Figura 14: Caso de uso 08 – Tiempo

CU_09: Pasar Turno	
Nombre:	Pasar Turno
Actores:	Usuario con Wallet en Metamask

Objetivos:	Poder pasar turno al jugador contrario
Precondición:	1) Entrar a la página web. 2) Tener metamask con una cuenta en activo 3) Crear una partida / contrato 4) Realizar jugadas 5) Un turno deben ser > 120 s
Postcondición:	El usuario tendrá a disposición un botón de acción para poder pasar turno en caso de que el jugador contrario no realice una tirada dentro del tiempo estipulado.
Escenario básico:	- El usuario entra en la D'APP - En caso de tener Wallet compatible con Metamask crea o entra en partida. - Empezar el juego / Realizar tirada. - Tiempo entre tirada > 120 segundos.

Figura 15: Caso de uso 09 – Pasar Turno

CU_10: Fin Partida	
Nombre:	Fin Partida
Actores:	Usuario con Wallet en Metamask
Objetivos:	Poder cerrar una partida con un ganador si se cumplen las condiciones
Precondición:	1) Entrar a la página web. 2) Tener metamask con una cuenta en activo 3) Crear una partida / contrato 4) Realizar jugadas 5) Obtener un ganador
Postcondición:	El usuario no podrá realizar más jugadas en caso de que haya un vencedor.
Escenario básico:	- El usuario entra en la D'APP - En caso de tener Wallet compatible con Metamask

	crea o entra en partida. - Empezar el juego / Realizar tirada. - Ganar/Perder el juego.

Figura 16: Caso de uso 10 – Fin Partida

2.4 Requisitos funcionales

Requisitos funcionales	
RF1	La aplicación permite a los usuarios **crear** partidas
RF2	La aplicación permite **Unirse** a partidas
RF3	La aplicación permite **generar hash de partidas** para los usuarios que crean partidas.
RF4	La aplicación permite **insertar hash de partida** para los usuarios que se unen a partidas..
RF5	La aplicación permite **crear un tablero de juego** por partida.
RF6	La aplicación permite **realizar jugadas sobre el tablero.**
RF7	La aplicación permite **la gestión de turnos.**
RF8	La aplicación permite **la gestión de tiempo entre turnos.**
RF9	La aplicación permite **comprobar si la partida sigue en curso o ha acabado.**
RF10	La aplicación permite **realizar comprobaciones sobre la cuenta activa** configurada en metamask.
Requisitos no funcionales	
RNF1	La aplicación debe ser compatible con todos los navegadores.
RNF2	La aplicación debe ser *responsive* a los diferentes tipos de resoluciones.
RNF3	La aplicación debe ser fácil e intuitiva.
RNF4	La aplicación debe ser fácil de desarrollar y mantener

Requisitos Software para el proyecto	
RS1	S.O Base **Linux Ubuntu 20.04.3 LTS** para poder instalar todo el aplicativo de software necesario.
RS2	**Ganache** como software que nos proporciona una red de pruebas en local.
RS3	**NodeJS con Webpack** como empaquetador de módulos.
RS4	**Remix IDE Ethereum** como entorno de desarrollo principal del contrato inteligente y que provee la EVM para la blockchain. (Ethereum Virtual Machine).
RS5	**Metamask** como plugin en el navegador o móvil que actúa como wallet. También permite utilizar monederos en redes testnet como Kovan.
Requisitos Hardware para el proyecto	
RH1	Ordenador Desktop/Laptop para instalar app's básicas para el desarrollo en local

Figura 17: Requisitos funcionales – RF

3. Diseño y Arquitectura

En este apartado se explicarán los tipos de datos presentes en nuestra aplicación, así como entrando en detalle en el diseño de algunas funcionalidades del juego y se detallarán los esquemas de navegación.

3.1 Diseño de E-R

Aunque el juego en si no tenga presente una base de datos como tal, podríamos expresarlo en un diagrama Entidad-Relación de la siguiente manera:

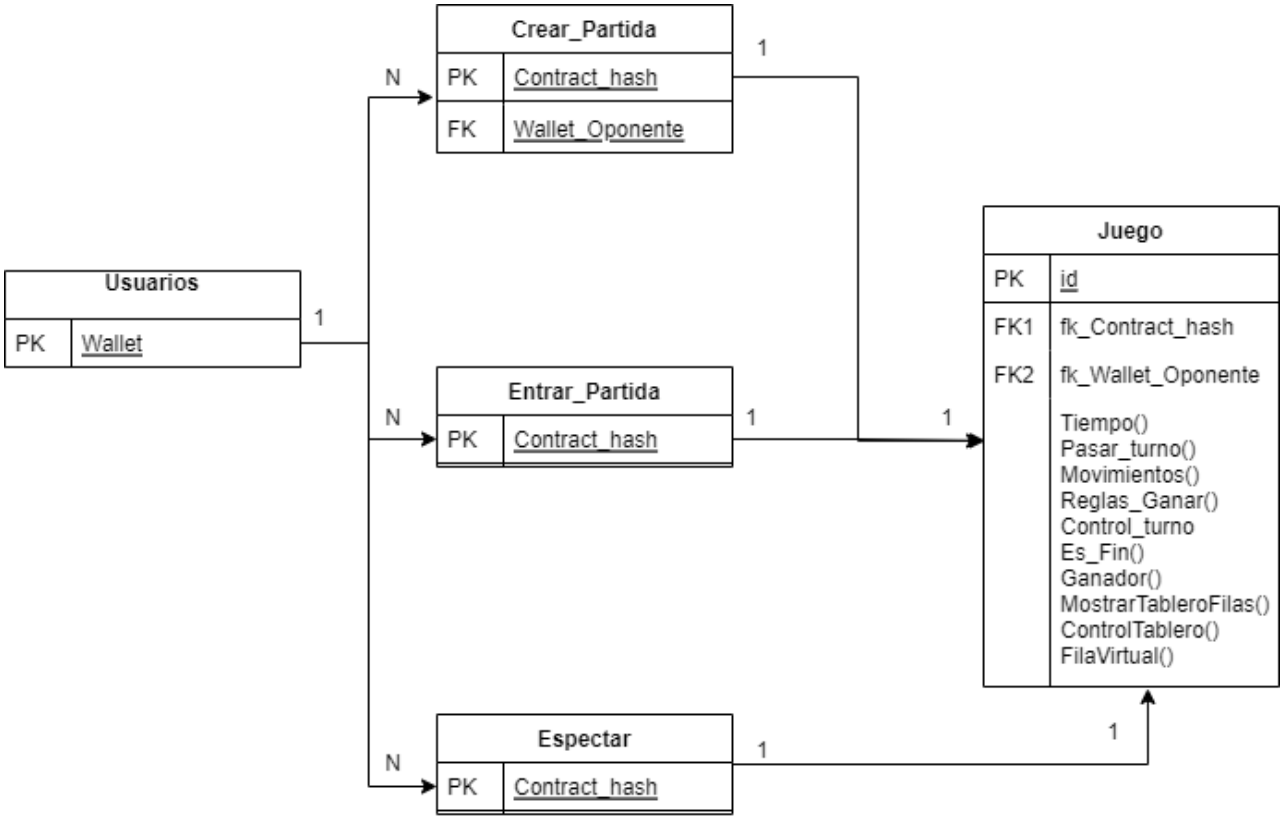


Figura 18: Diseño E-R

Un usuario que entra a la página principal del juego podrá crear partidas, unirse a una partida creada o espectar las partidas.

Al haber un hash por juego, cada partida generada por un usuario se considera única y nadie puede entrar o espectarla a no ser que tenga dicho hash.

3.2 Diseño de la Navegación

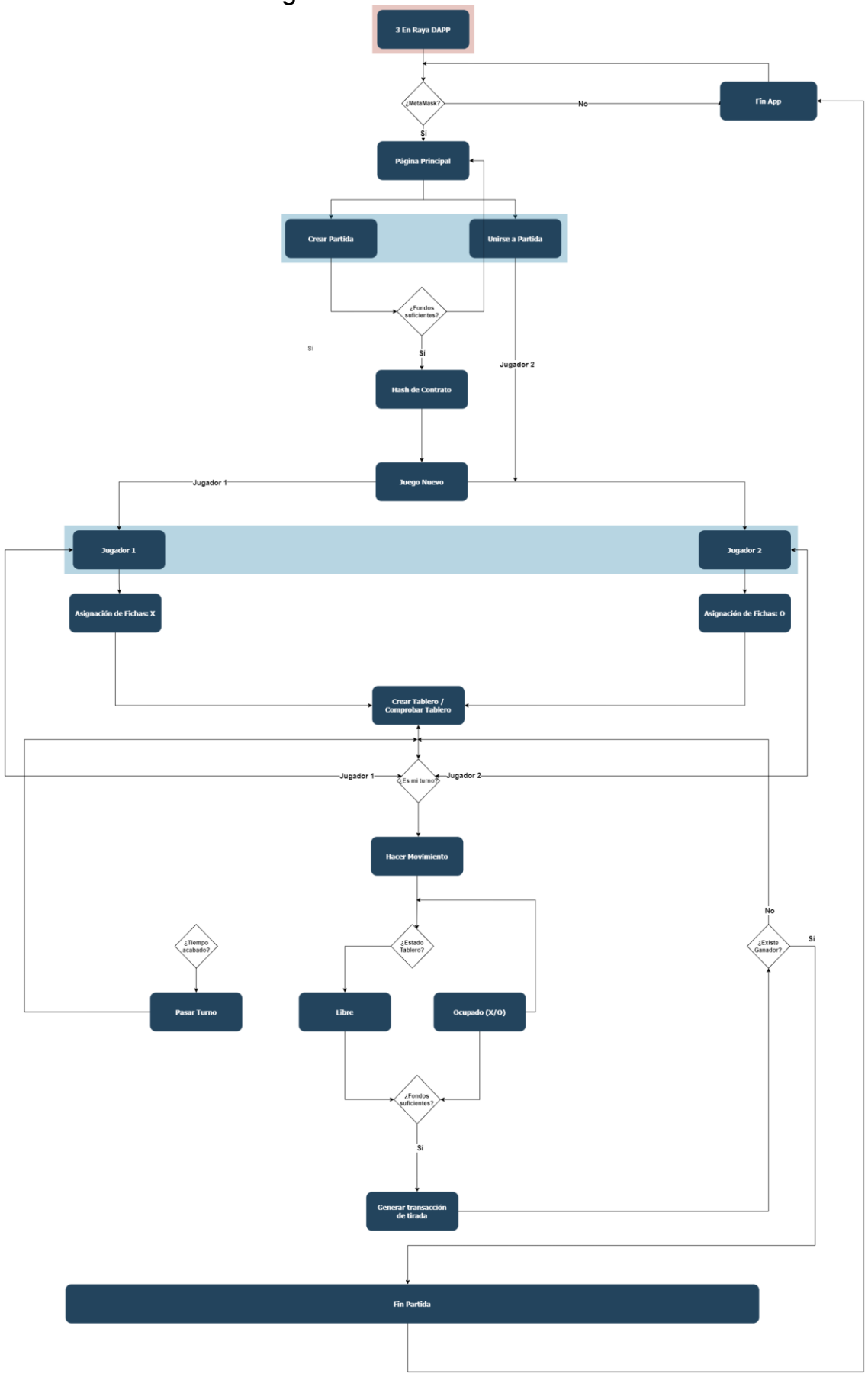


Figura 19: Diseño de Navegación

3.2.1 Definición de Pantallas:

- **Pantalla Principal:**
  Como pantalla principal tenemos la bienvenida al juego, identificando al jugar por su Wallet y dándole la bienvenida. Esta pantalla será del tipo *scroll* hacía la derecha para moverse entre los diferentes apartados de la DAPP.

Pantalla Bienvenida	
• ***Crear Partida***	• **Unirse a Partida**
• ***Hash devuelto al crear partida***	• ***Campo para indicar el hash de partida.***

Pantalla de Juego			
• ***Indicador ID de Partida (hash)***	• *Indicador de cuenta actual del jugador*	• *Indicador de Turno*	• ***Indicador de tiempo restante entre turnos***
• ***Indicador que dice si la partida sigue o ha finalizado***	• *Indicador de la figura o turno que toca hacer movimiento*	• *Tablero de juego*	

Pantalla de Reinicio
• *Botón reinicio de la D'APP al finalizar partida*

Figura 20: Definición de Pantallas

3.3 Diseño gráfico e interfaces

En este apartado se describirán toda la composición respecto al diseño gráfico de interfaces, describiendo: Fuentes tipográficas utilizadas y paleta de colores.

3.3.1 Diagrama de Colores

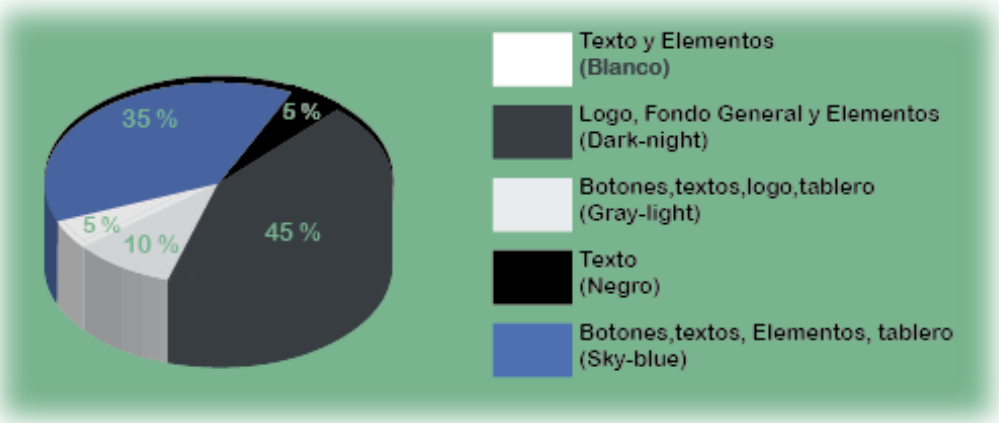
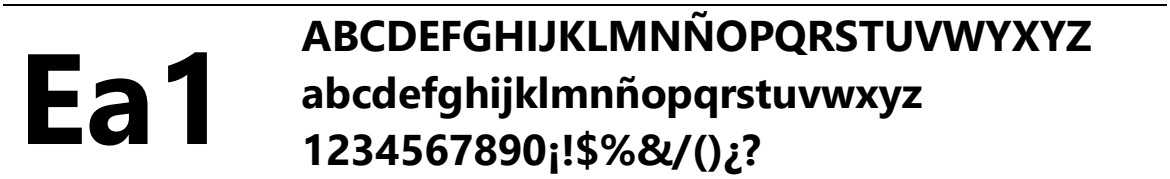


Figura 21: Diagrama de composición de colores

3.3.2 Fuentes Tipográficas

Fuente principal del logotipo (Logotipo, títulos y cuerpo):
Segoe UI



Fuente para el cuerpo de texto (Elementos del menú):

Times New Roman

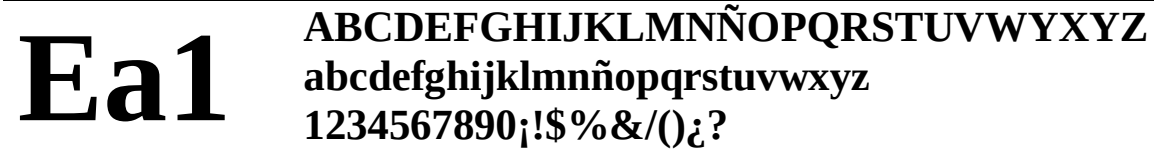


Figura 22: Muestras tipográficas

3.4 Lenguajes de programación utilizados

El desarrollo de la DAPP se llevará a cabo en **formato web**, ya que se quiere evitar incompatibilidades entre sistemas operativos de diferentes marcas del mercado. La forma más fácil de llegar a todos los públicos es a través de un navegador web, ya que hoy día cualquier dispositivo incorpora uno propio. Es por eso, que en este apartado trataremos los diferentes escogidos tanto para el Front-End como para el Back-End, justificando su uso.

3.4.1 Front-End

Para el desarrollo principal **Front-End** de la página se escoge **HTML5**, debido a su capacidad de mostrar contenido dinámico al usuario, su fácil manejo y su gran flexibilidad. También habría que hacer mención de que la principal ventaja de este es que es soportado por todos los dispositivos sin tener instalado ninguna utilidad adicional. Así que, **HTML5** sería ideal para llevar el juego del 3 en raya a todos los rincones del mundo. Por otro lado, HTML5 dispone de unos grandes aliados y que sin estos el lenguaje no tendría sentido. **CSS3 y JavaScript**, para incorporar estructura y funcionalidad al lenguaje que, además, es capaz de aportar contenido semántico a sus elementos.

CSS3 incorpora mejoras centradas, sobre todo, en el ámbito visual. Es la parte que se encarga de dar forma y color a todo el contenido de la página, además de poder aportar la funcionalidad de crear sitios *responsive como ya se ha mencionado anteriormente en el proyecto.*

Como complemento al CSS3, se utiliza el **Framework Bootstrap 4** que ayuda a desarrollar interfaces de gran usabilidad y totalmente *responsive*.

Como característica principal de **Bootstrap 4** es que es *software* libre y fácil de utilizar. Es un gran complemento que puede ser integrado con HTML5 de manera que permite personalizar todos sus elementos (como las Barras de Navegación, Formularios, Tablas, Botones, *Glyphicons*, etc.).

Por último y no podría ser menos, se utilizará como lenguaje de programación en el lado del cliente: **JavaScript.** Las siguientes ventajas, responden el porqué de su elección:

- Es un lenguaje muy sencillo.
- Es veloz, por lo tanto tiende a ejecutar funciones inmediatamente.
- Cuenta con múltiples opciones de efectos visuales.
- Es soportado por los navegadores más populares y es compatible con los dispositivos más modernos. Es muy versátil, puesto que es muy útil para desarrollar páginas dinámicas y aplicaciones web.
- Es una buena solución para poner en práctica la validación de datos en un formulario.

- Es multiplataforma, puede ser ejecutado de manera híbrida en cualquier sistema operativo móvil.
- Es el único lenguaje que permite trabajar modo *FullStack* en cualquier tipo de desarrollo de programación.
- Además de todo lo anterior, se utiliza en combinación con **JQuery**
- Fácil integración de la librería web3 y el contrato inteligente en la blockchain.

JQuery:

- Sintaxis liviana, teniendo en cuenta que la sencillez y poca extensión de código.
- Funciona independientemente del navegador sobre el que se visualiza el *site*. Al servir para todos, el desarrollador tiene menos trabajo, menos código, menos espacio, menos problemas.
- Existen cientos de *plugin*. Adaptables al tipo de proyecto que se esté desarrollando.
- Es fácil de aprender. Los resultados que nos ofrece con poco esfuerzo son de los más relevantes.
- Su grado de penetración entre los desarrolladores es amplio, existiendo una gran comunidad, con lo cual, el soporte, la documentación y los recursos se nos presentan amplios también.

Figura 23: Logotipo de HTML5,CSS,JS,Jquery

3.4.2 Back-End

Para el desarrollo principal Back-End se escoge Solidity, ya que es el principal lenguaje orientado a objetos de Ethereum y a través del cual nos permite generar contratos inteligentes. Además permite la fácil integración mediante librerías del frontend entre la DAPP y el contrato inteligente.

Se escoge el lenguaje de programación Solidity por las siguientes razones:

- Su sintaxis es similar a la de JavaScript y está enfocado específicamente a la Ethereum Virtual Machine (EVM).

- Solidity es capaz de hacer posible las aplicaciones descentralizadas (DAPPS) de la red Ethereum (ETH). Esto brinda a los desarrolladores una manera más fácil de realizar complejas aplicaciones distribuidas sacando, de este modo, el máximo provecho a la Ethereum Virtual Machine (EVM).

Figura 24: Logotipo de Ethereum y Solidity

4. Implementación

En este apartado se detallan las diferentes herramientas, *Plugins* y Apis utilizadas en el desarrollo del proyecto.

4.1 Herramientas Utilizadas

4.1.1 Brackets

Brackets es un editor de código fuente moderno y de código abierto, enfocado principalmente para el desarrollo web.

Brackets ha sido la herramienta principal de desarrollo de este proyecto, ya que ha ofrecido compatibilidad con los principales lenguajes de programación utilizados en este proyecto, como son: HTML5,JS y CSS3.

4.1.2 Webpack

Webpack es una herramienta configurable que nos ayudará a realizar algunas tareas básicas en el desarrollo Frontend en tareas automatizadas y preparar nuestra aplicación web para producción.

Lo podríamos definir como un empaquetador de módulos, que permite generar un archivo único con todos aquellos módulos que necesita nuestra aplicación para funcionar. Por poner un ejemplo, permite incluir todos los archivos

javascript .js en un único archivo, incluso se pueden incluir hasta archivos de estilos .css

Además incluye gran cantidad de *plugins* que facilitan mucho las tareas de trabajo y minimizan el tiempo de ejecución de manera espectacular. Uno de los plugins a destacar es la función de *autoreload* por cada cambio guardado que hacemos en nuestros proyectos.

4.1.3 JSfiddle

Es una herramienta *Online* muy útil, sencilla de utilizar y a la vez fantástica por su forma de poder hacer pruebas con nuestro código HTML,CSS y JS de forma fácil y cómoda. Además, es totalmente compatible con diferentes *frameworks* (Mootools, jQuery, Prototype, YUI, Dojo, Extjs, etc.)

4.1.4 IDE REMIX

Remix IDE es una aplicación web y de escritorio de código abierto que fomenta un ciclo de desarrollo rápido y tiene un amplio conjunto de complementos con GUI intuitivas. Remix se utiliza para todo desarrollo de contratos inteligentes en el backend.

4.1.5 Logomaster.ai

Es una aplicación que genera varios diseños de logotipos según preferencias y estilos seleccionados previamente por el usuario. Tras elegir el diseño que queremos, permite la descarga de las imágenes y una breve guía sobre su implantación en tu sitio web.

4.2 Requisitos de Instalación

La DAPP 3 en raya está implantada en un paquete de módulos llamado: webpack para Linux y que depende de nodejs para su ejecución. Además incorpora un sistema de servidor Web para desarrolladores llamado webpack dev server y que puede ser utilizado como servidor HTTP para servir ficheros HTML,CSS,JS… mientras se desarrolla en él.

Se escoge **webpack** debido a su fácil implantación, ya que no hay que instalar las aplicaciones una por una. Con este paquete, la instalación ya incluye todo el *software* necesario y funcionalidades básicas para poder empezar a desarrollar un proyecto web. Además, es una aplicación modular, la cual permite integrar con contratos inteligentes realizados en Solidity perfectamente mediante la librería web3.js.

Por último, cabe destacar su **licencia MIT** (totalmente gratuita y sin costes), además de ofrecer compatibilidad con gran variedad de sistemas operativos.

Figura 25: Logotipo de Webpack, nodeJS y web3JS

También se utiliza el software Ganache que proporciona una red de pruebas local bastante sencilla e intuitiva y que para instalarlo habrá que descargárselo de la web oficial de los productos que forman parte de la *truffle suite*.
https://truffleframework.com/ganache

Figura 26: Logotipo de Ganache

4.3 Entorno de desarrollo

El entorno de desarrollo ha sido el siguiente:

4.3.1 Entorno de sobremesa

Procesador:	I7-6700K 4.00Ghz
Ram:	16 GB
S.O:	Windows 10 Home
Disco Duro:	1 TB SSD
Uso:	Mayoritariamente es donde se ha desarrollado el proyecto y donde tiene todas las aplicaciones posteriormente comentadas en la fase de implementación.

Figura 27: Entorno Desarrollo - Sobremesa

4.4 Entorno de Pruebas Móviles

Como entorno de pruebas, además de los dos dispositivos anteriores mencionados, se han hecho pruebas a nivel de dispositivo móvil, utilizando **iPhone XS** y **Samsung S10e**.

- **iPhone XS:**

Procesador:	Apple A12 Bionic
Pantalla:	5.8 Pulgadas (OLED)
Resolución:	2,436 x1,125 pixeles
Ram:	4 GB
S.O:	IOS 12
Disco Duro:	64 GB SSD
Uso:	Se utiliza para realizar pruebas de compatibilidad de vista desde el navegador Safari de Apple.

Figura 28: Entorno de Pruebas – iPhone XS

- **SAMSUNG S10e:**

Procesador:	Snapdragon 2,7 GHz Octa-Core
Pantalla:	5.8 Pulgadas (AMOLED)
Resolución:	2280 x 1080 pixeles
Ram:	6 GB
S.O:	Android
Disco Duro:	128 GB SSD
Uso:	Se utiliza para realizar pruebas de compatibilidad de vista desde el navegador Chrome de Android.

Figura 29: Entorno de Pruebas – Samsung S10e

4.5 Librerías Utilizadas

Librería	Descripción
Font-Awesome	Librería que permite añadir iconos de diseño procedentes del paquete Font Awesome
Jquery	Librería como complemento al JavaScript que permite agregar interactividad y efectos visuales.
Web3js y ethjs	Colección de librerías que permiten interactuar con un nodo de Ethereum local o remoto mediante HTTP, IPC o WebSocket.

Figura 30: Librerías – Tabla Librerías

4.6 Frameworks Utilizados

Librería	Descripción
Bootstrap v4	Es el frameworks CSS utilizado en el diseño gráfico de la aplicación

Figura 31: Frameworks – Tabla Frameworks

4.7 Lenguajes y técnicas utilizadas

Librería	Descripción
Solidity	Lenguaje de programación del Back-End, utilizado en la lógica que controla el juego.
HTML5 + CSS3 + JS	Lenguaje mediante el cual definimos toda la estructura de la vista en el Front-End y comunicación mediante librería web3 y el backend de la DAPP.

Figura 32: Lenguajes y técnicas – Tabla Lenguajes

4.8 Ejemplos de Implementación

4.8.1 Back-End

- Función tiempo:

La función **empezar()** recoge el *timestamp* en el que se lanza la transacción

```
function empezar() public{
  _Empezar = block.timestamp;
}
```

Figura 33: Implementación – empezar()

La función **finalizar()** recoge el tiempo total entre tiradas.

```
function finalizar() public{
  _Terminar = 120+_Empezar;
}
```

Figura 34: Implementación – finalizar()

La función **Tiempo()** es la que hace las operaciones con los tiempos obtenidos y entrega los resultados en la DAPP posteriormente, en el if se

diferencian dos partes, la resta para obtener el tiempo que queda actualmente y la segunda parte que cuando el resultado sea 0 pare de contar devolviendo un 0 como resultado.

```
function Tiempo() public view returns (uint){
    uint finish = 0;
    uint now = block.timestamp;
   if (_Terminar > now) {
        uint pendingTime = _Terminar - now;
        return pendingTime;
    }else{
        return finish;
    }
}
```

Figura 35: Implementación – Tiempo()

La función **constructor()** recibe por parámetro la dirección de la wallet del usuario oponente pasada al contrato con el fin de inicializar un juego.

```
constructor (address _jugador2)
  {
     jugador1 = msg.sender; //Jugador1 siempre será el que crea el contrato O inicia la partida
     jugador2 = _jugador2; // Pasamos por parámetro la dirección del Jugador 2.
          ….
   }
```

Figura 36: Implementación – Constructor()

La función **Movimientos()** recibe por parámetro posición donde ha marcado el usuario tirar mediante variables X e Y.

x,y|x,y|x,y
0,0|1,0|2,0
0,1|1,1|2,1
0,2|1,2|2,2

Figura 37: Implementación – Movimientos()

Además, se tienen en cuenta las siguientes condiciones:

```
// Comprobar que los jugadores están en juego
require (msg.sender == jugador1 || msg.sender == jugador2);

//Comprobar que el juego no ha acabado
require(!EsFin(), "El juego ha terminado");

//Comprobar que el jugador puede mover
require (msg.sender == Control_turno(),"No es turno del jugador");

//Comprobar que el campo seleccionado este vacío
require (Tablero[x][y] == Estado_Tablero.Libre);
```

La función **Reglas_Ganar()** tiene definida todas las condiciones que se pueden dar para que un jugador gane la partida. Las casuísticas están declaradas por columnas, filas y diagonales.

La función **EsFin()** Es la encargada de comprobar si la partida ha acabado. Existen dos formas de decir que una partida ha acabado:

- La partida acaba si hay ganador sin espacios de tablero libre

o;

- Si los movimientos llegan a ocupar las 8 posiciones o equivalente a tablero lleno.

La función **Ganador()** Es la encargada de devolver un ganador una vez sabemos que existe un ganador.

Tres valores para devolver:

- 1 – Jugador (2) con fichas O es el ganador
- 2- Jugador (1) con fichas X es el ganador.
- 0 – No existe ganador.

La función **MostrarTablero()** Es la encargada de devolver el paquete de string con el resultado del tablero tras haberlo comprobado mediante función **FilaVirtual()** y **ControlTablero()**.

La función **ControlTablero()** Es la encargada de retornar si una casilla está ocupada por una ficha de jugador o sigue libre.

La función **FilaVirtual()** Es la encargada de devolver el paquete de string con el resultado de las filas para posteriormente montarlas en **MostrarTablero()** y que será llamado desde la DAPP.

La función **Dibujar_Figura_Jugador()** Es la encargada de decidir los turnos del jugador que le toca tirar actualmente. Se controla de la siguiente manera:

Si el turno es del tipo par, tira el jugador “X” si es impar, tira el jugador “O”.

```
function Dibujar_Figura_Jugador() public view returns (Estado_Tablero)
 {
    if(movimientos % 2 == 0)
    {
       return Estado_Tablero.X;
    }else
    {
       return Estado_Tablero.O;
    }
 }
```

Figura 38: Implementación – Dibujar_Figura_Jugador()

La función **Pasar_turno()** Es la encargada de pasar el token de turno o de tirada al jugador contrario una vez se cumplen una serie de condiciones y además se vuelven a resetear los tiempos.

4.8.2 Front-End

4.8.2.1 Font-Awsome

En la DAPP se incluye la librería Font Awesome Free Edition que para cargarla se tendrá que hacer referencia a la librería mediante el siguiente link desde el header de la aplicación (dependiendo de la versión que se utilice).

```
<link href="//maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet">
```

Figura 39: Font Awesome – Header

Para poder hacer implementaciones de sus iconografías, se pueden implementar en campos , Por ejemplo, con el siguiente código mostraremos el icono de un reloj:

Código:	Resultado:
	

Figura 40: Font Awesome – Implementación

4.8.2.2 Bytecode

Para poder generar contratos desde la propia DAPP utilizamos el *bytecode* que nos da la propia compilación del contrato inteligente que vendría a ser la representación hexadecimal del contrato. Esta la incluimos dentro de la variable *byteCode* del *front* y que incluiremos en la generación del contrato:

Código:	Resultado:
meta = eth.contract(abi, byteCode, { from: accounts[0], gas: '3000000' });	

Figura 41: Bytecode

4.8.2.3 ABI Code

Para Ethereum y el EVM, un contrato es sólo un programa que se ejecuta en secuencia de bytes. Sólo el lenguaje de alto nivel como Solidity define cómo se llega desde el punto de entrada del programa hasta el punto de entrada de una función concreta. Cuando una aplicación externa u otro contrato inteligente quiere interactuar con la blockchain, necesita tener algún conocimiento de la interfaz de un contrato inteligente, como una forma de identificar un método y sus parámetros. Esto es facilitado por la Interfaz Binaria de Aplicación de Ethereum (ABI).

El ABI es similar a la Interfaz de Programa de Aplicación (API), que es esencialmente una representación de la interfaz del código en forma legible para el ser humano o en un lenguaje de alto nivel. En el EVM el código compilado que se almacena como datos binarios y las interfaces legibles por humanos desaparecen y las interacciones de los contratos inteligentes tienen que traducirse a un formato binario que pueda ser interpretado por el EVM. La ABI define los métodos y estructuras que se pueden utilizar de forma sencilla para interactuar con ese contrato binario (al igual que la API), sólo que a un nivel inferior. La ABI indica a la persona que llama la información necesaria

(firmas de funciones y declaraciones de variables) para codificarla de manera que sea entendida por la máquina virtual que llama al código de bytes (contrato). Este proceso se llama codificación ABI.

Código:

```
[
	{
		"inputs": [],
		"name": "empezar",
		"outputs": [],
		"stateMutability": "nonpayable",
		"type": "function"
	},
	{
		"inputs": [],
		"name": "finalizar",
		"outputs": [],
		"stateMutability": "nonpayable",
		"type": "function"
	},
	....
```

Figura 42: ABI Code

5. Demostración

5.1 Prototipos

Se han elaborado algunos *Wireframes* esenciales de prototipado para tener una idea sobre el diseño general del juego. A continuación, se pueden ver los *Wireframes* de la pantalla principal y de la pantalla de juego para un usuario que se haya unido a una partida tanto en versión de escritorio como móvil.

5.2 Desktops Wireframes

Página del menú principal:

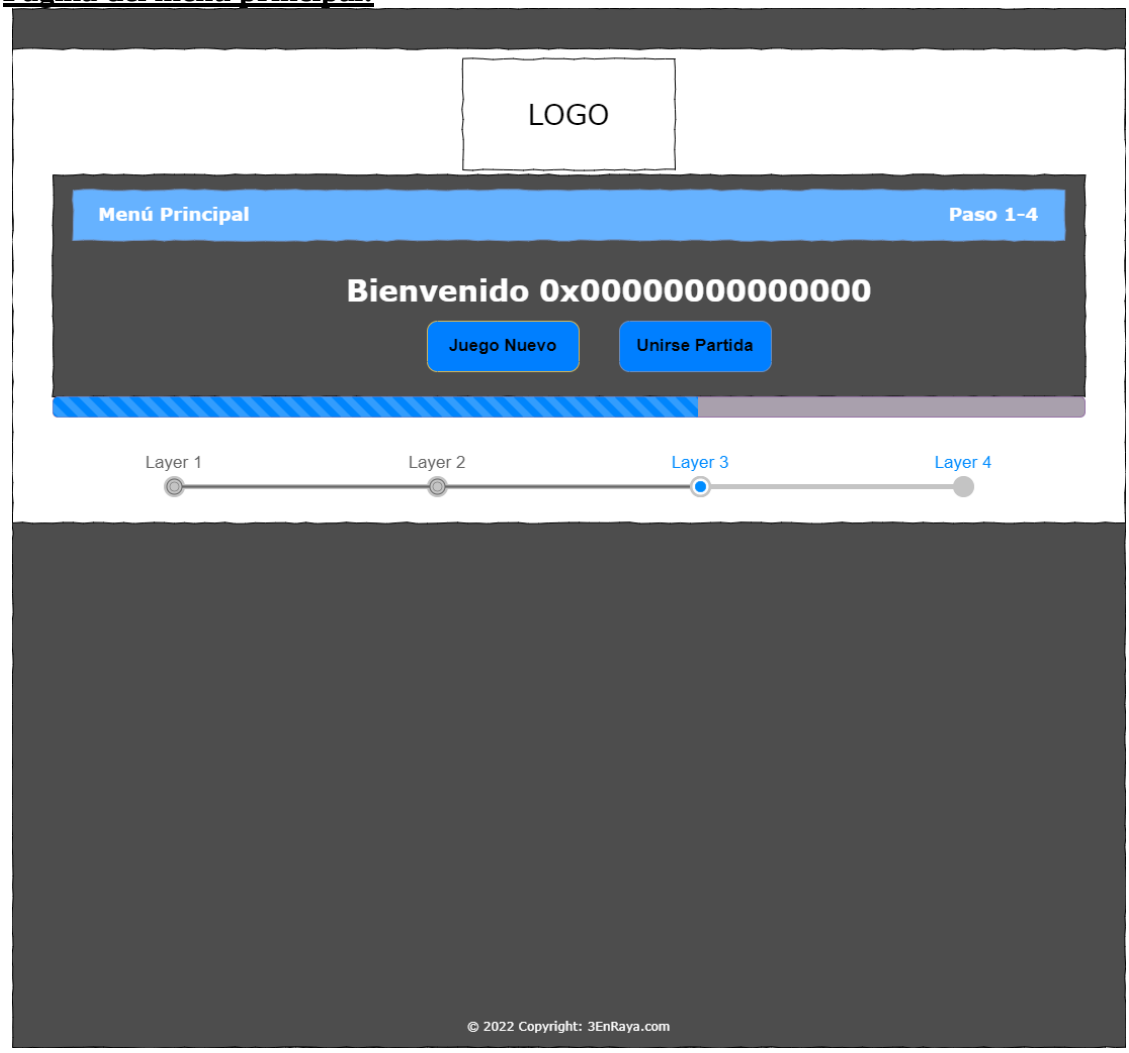


Figura 43: Prototipo – Página Principal (Desktop)

Página de crear partida:

Figura 44: Prototipo – Crear Partida (Desktop)

Página de unirse a partida:

Figura 45: Prototipo – Unirse Partida (Desktop)

Página de juego:

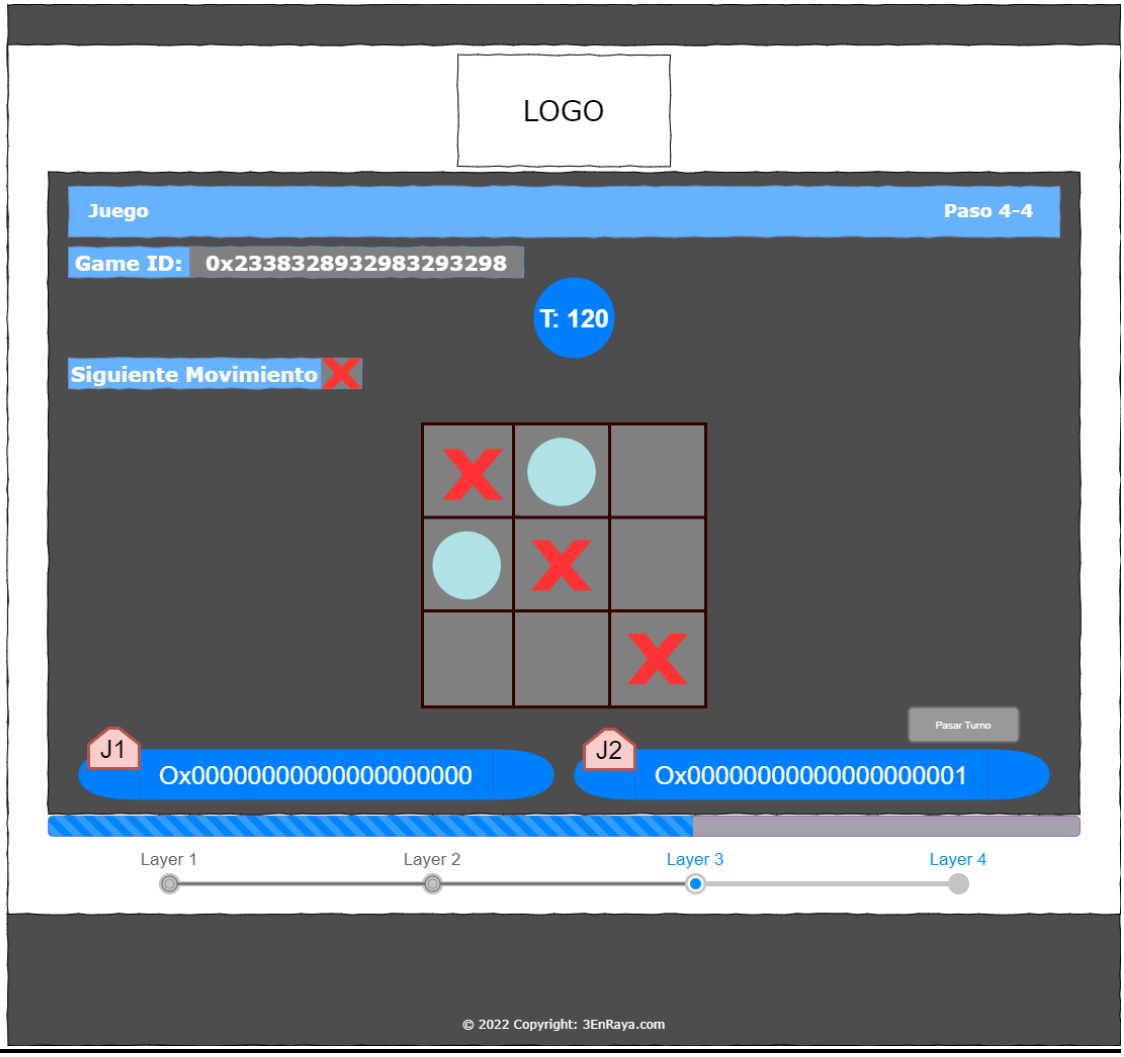


Figura 46: Prototipo – Página de juego (Desktop)

5.3 Mobile Wireframes

Página del menú principal:

Figura 47: Prototipo – Página Principal (Mobile)

Página de crear partida:

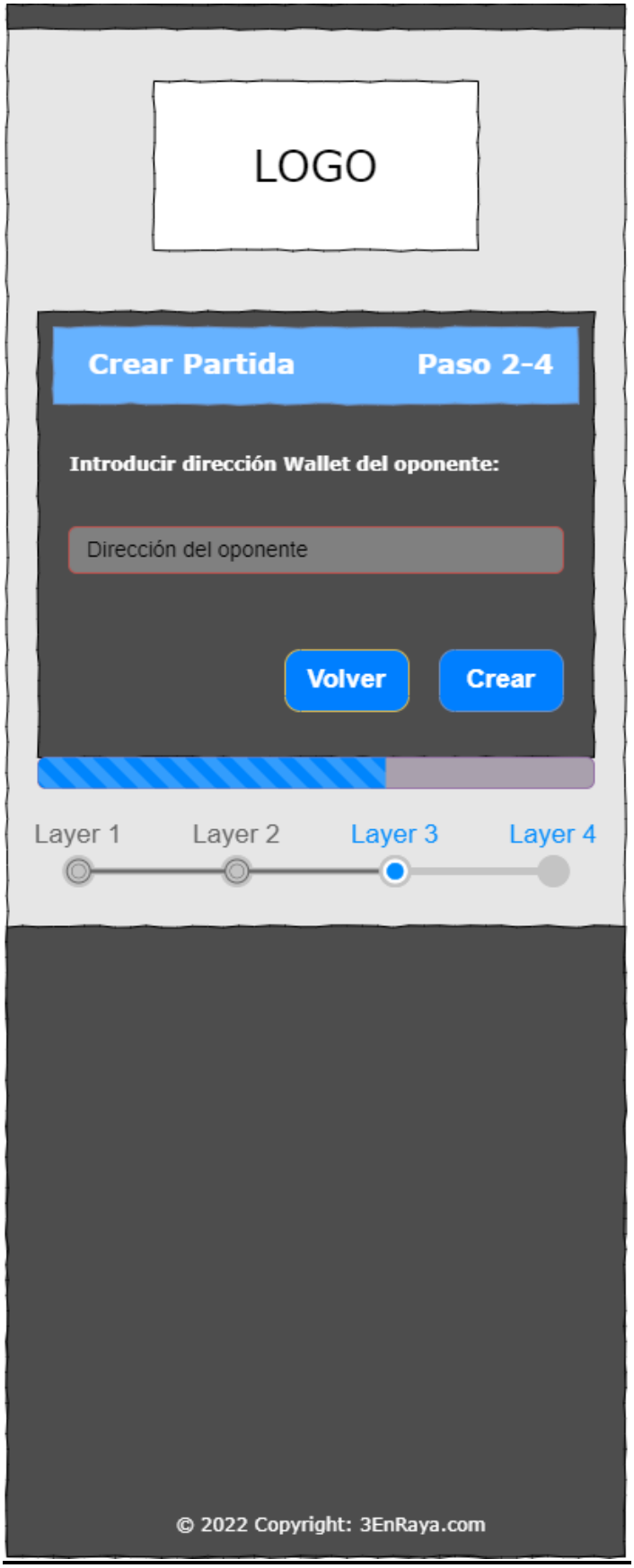


Figura 48: Prototipo – Crear Partida (Mobile)

Página de unirse a partida:

Figura 49: Prototipo – Unirse Partida (Mobile)

Página de juego:

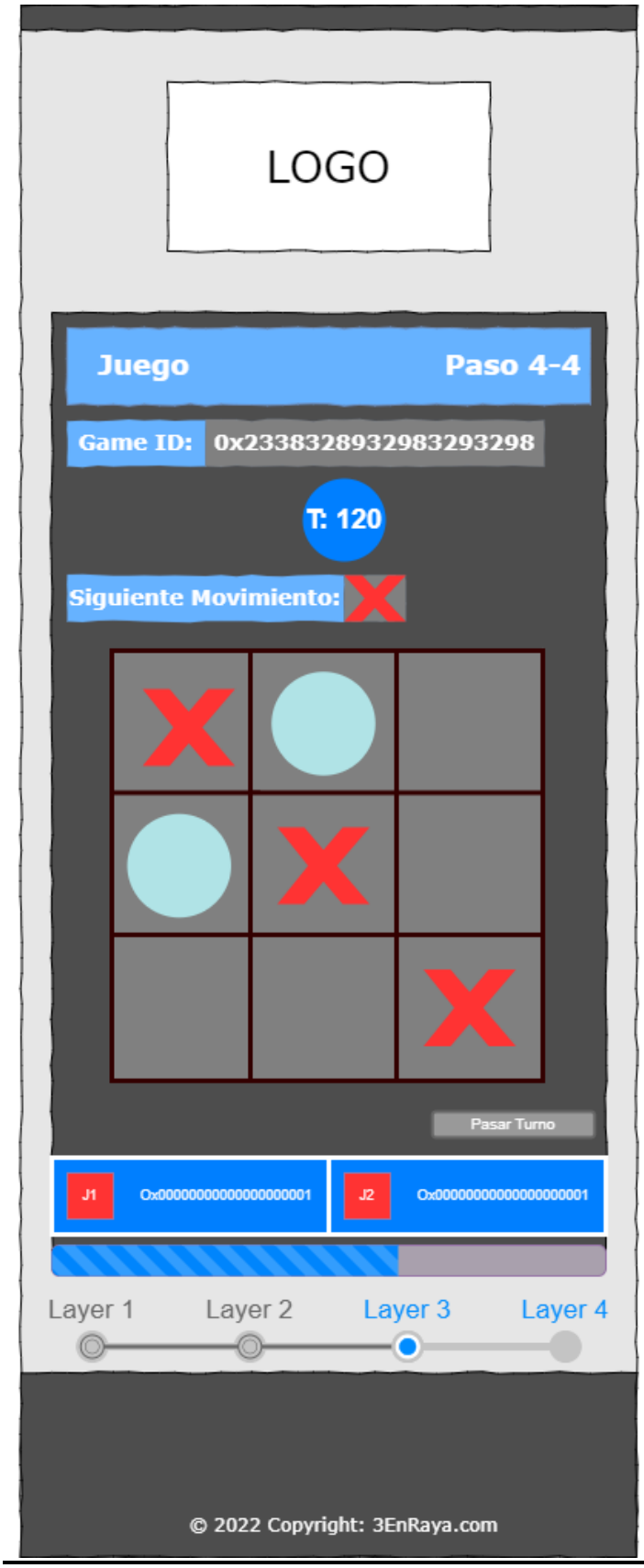


Figura 50: Prototipo – Página de juego (Mobile)

5.4 Desktop Hi-Fi

Página del menú principal:

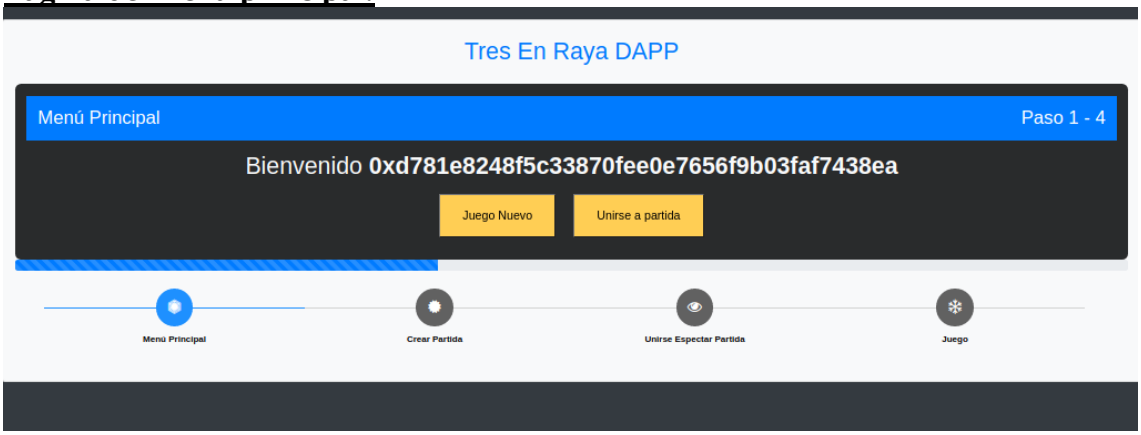


Figura 51: Prototipo Hi-Fi – Página Principal (Desktop)

Página de crear partida:

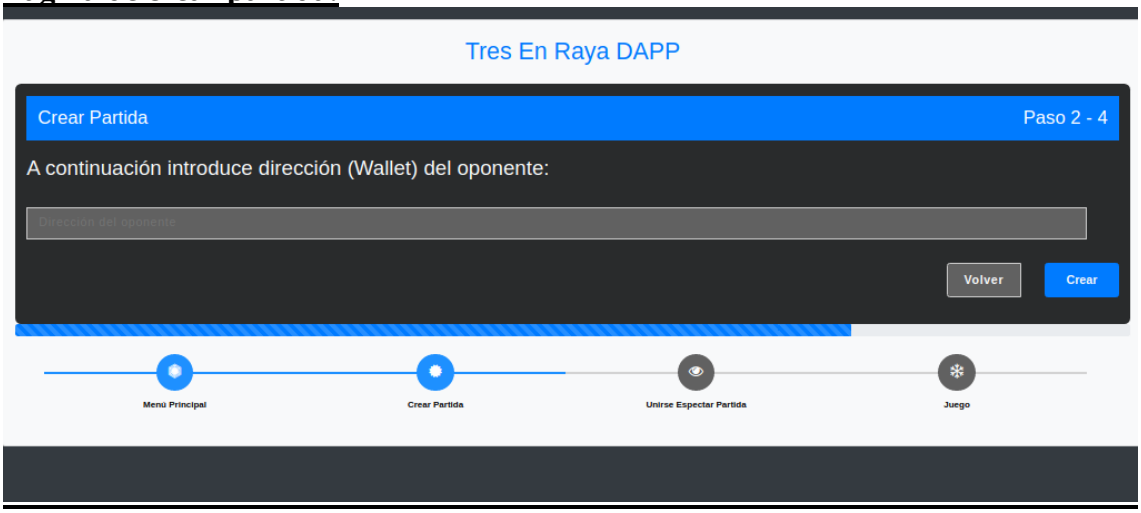


Figura 52: Prototipo Hi-Fi – Crear Partida (Desktop)

Página de unirse a partida:

Figura 53: Prototipo Hi-Fi – Unirse Partida (Desktop)

Página de juego:

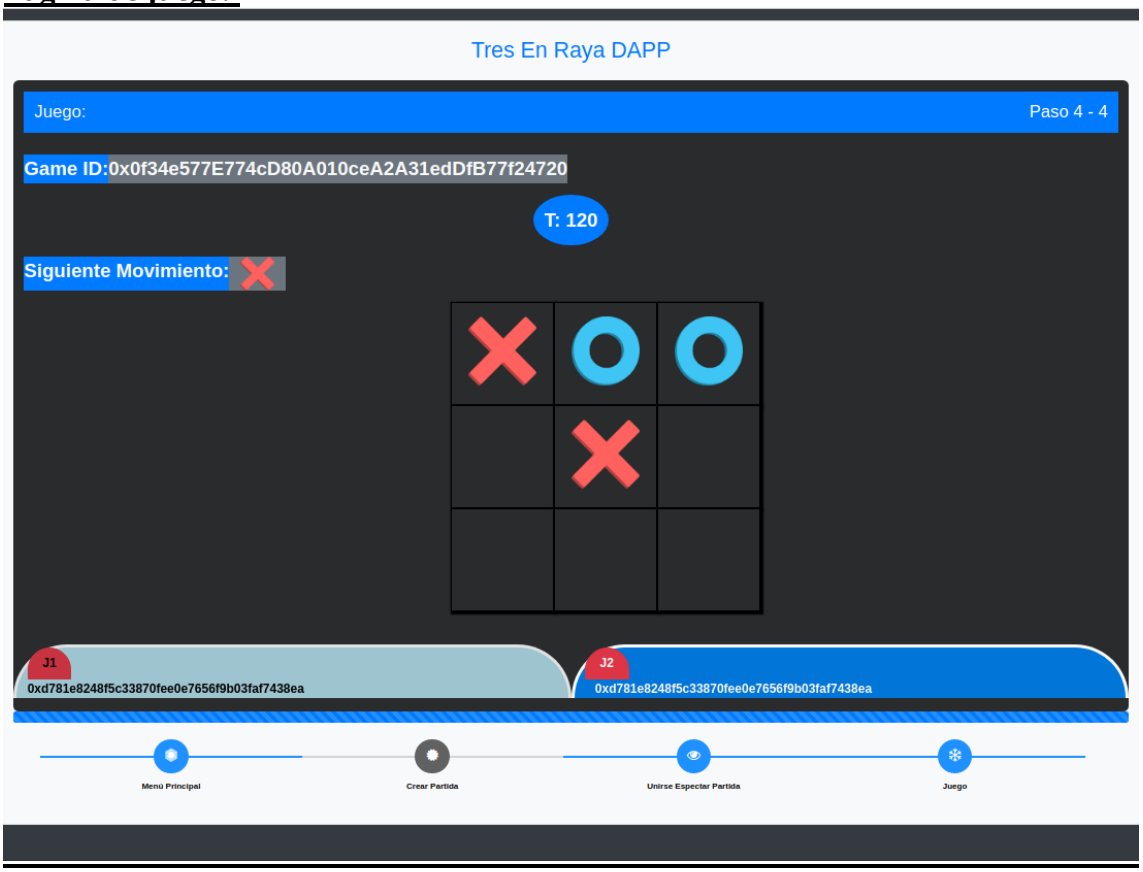


Figura 54: Prototipo Hi-Fi – Página de Juego (Desktop)

3) Ficha asignada:

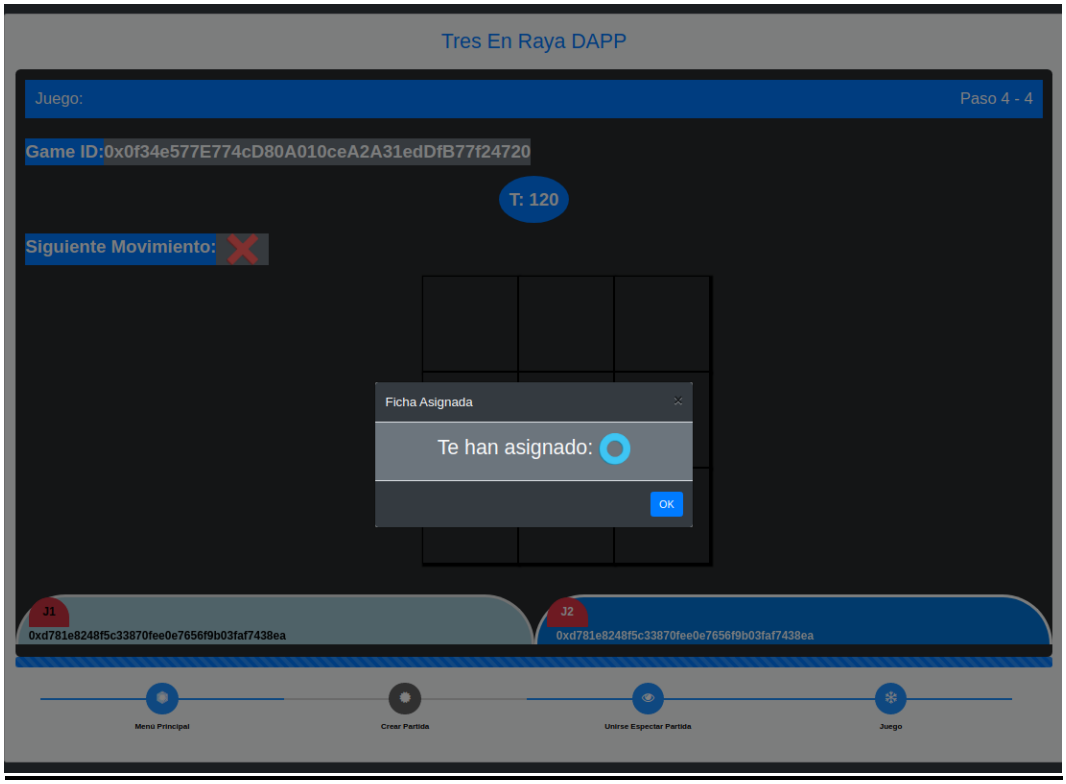


Figura 55: Prototipo Hi-Fi – Página de Ficha asignada en juego (Desktop)

4) Ganador / Crear Otra Partida:

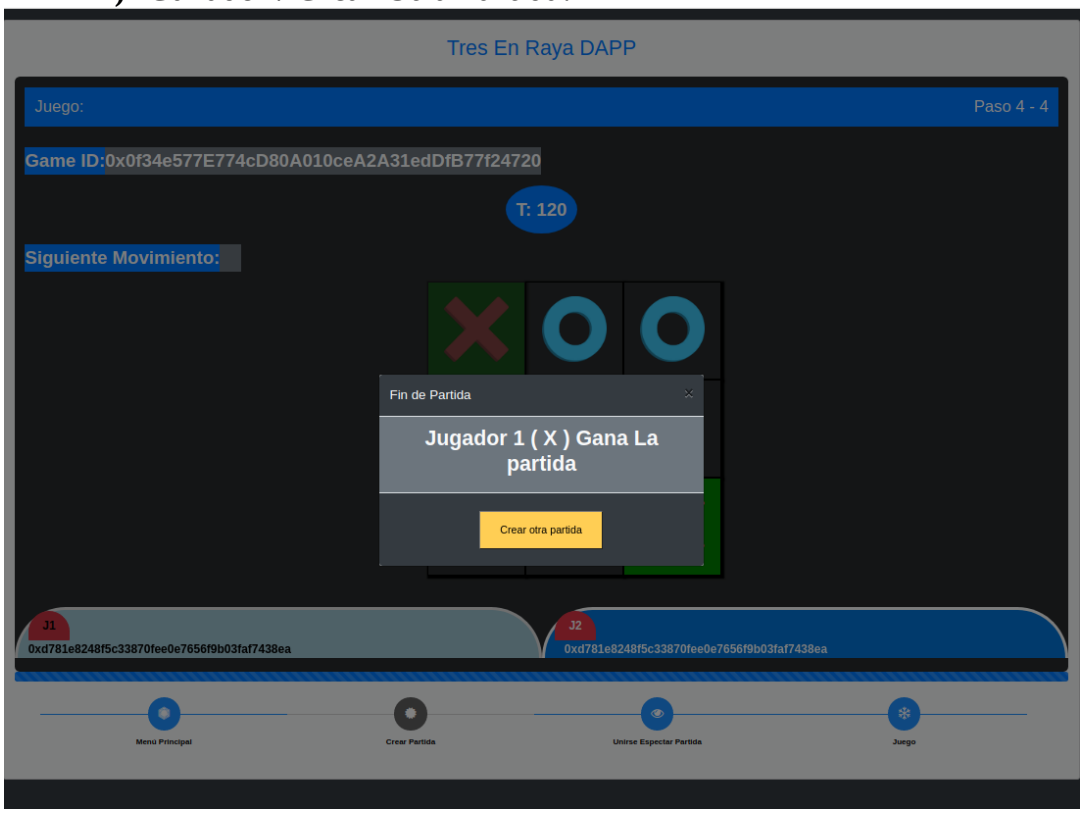


Figura 56: Prototipo Hi-Fi – Página Fin Partida en juego (Desktop)

5.5 Mobile Hi-Fi

Página del menú principal:

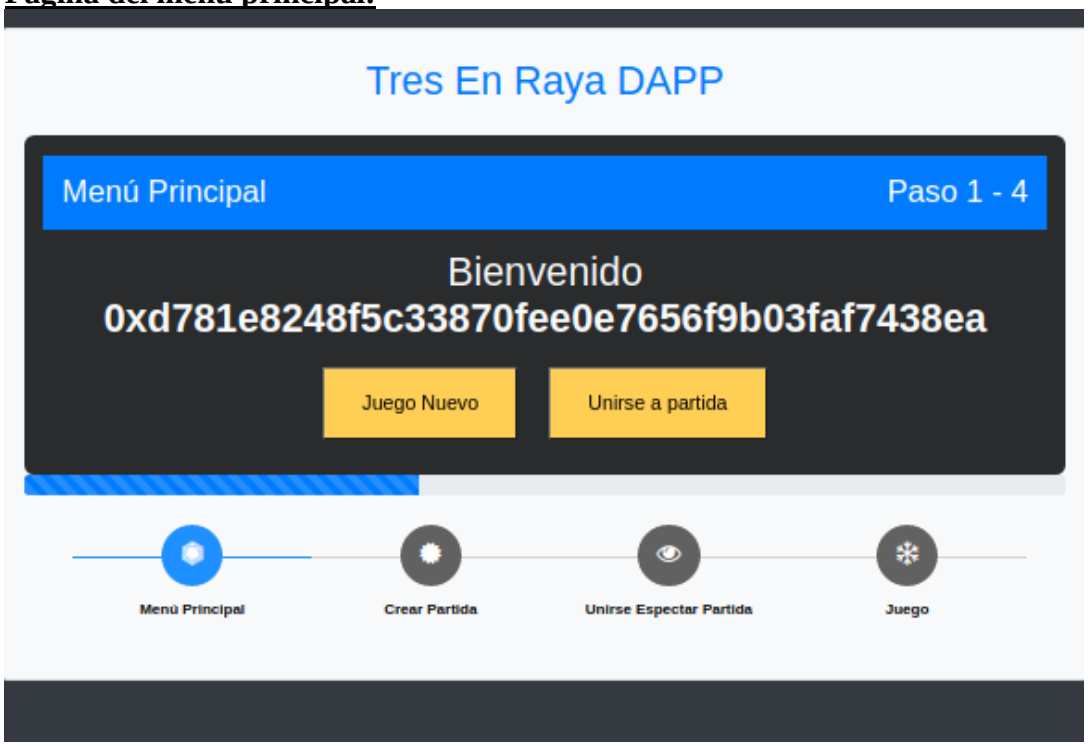


Figura 57: Prototipo Hi-Fi – Página Principal (Mobile)

Página de crear partida:

Figura 58: Prototipo Hi-Fi Crear Partida (Mobile)

Página de unirse a partida:

Figura 59: Prototipo Hi-Fi – Unirse a Partida (Mobile)

Página de juego:

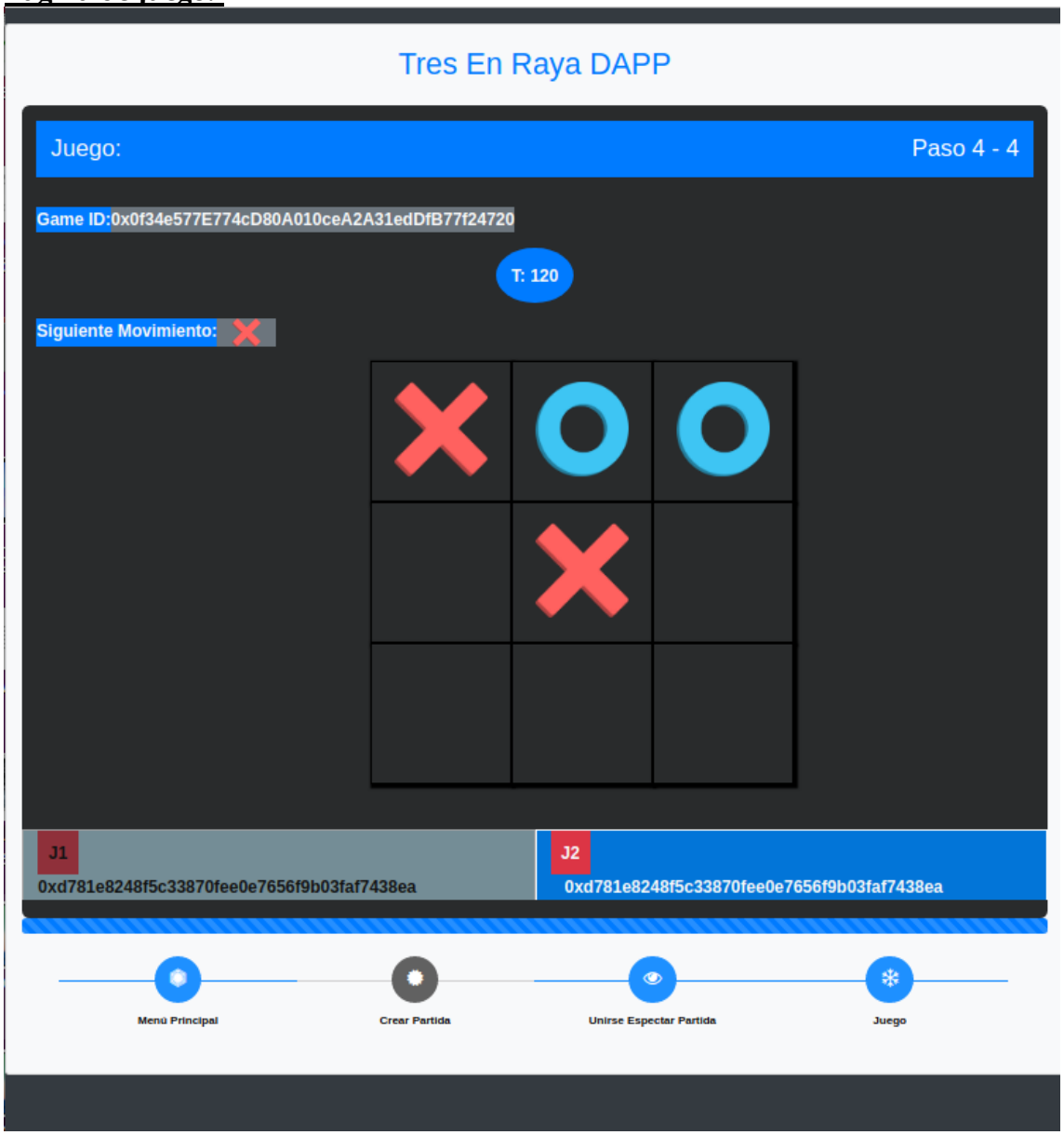


Figura 60: Prototipo Hi-Fi – Página de juego (Mobile)

4) Ficha asignada:

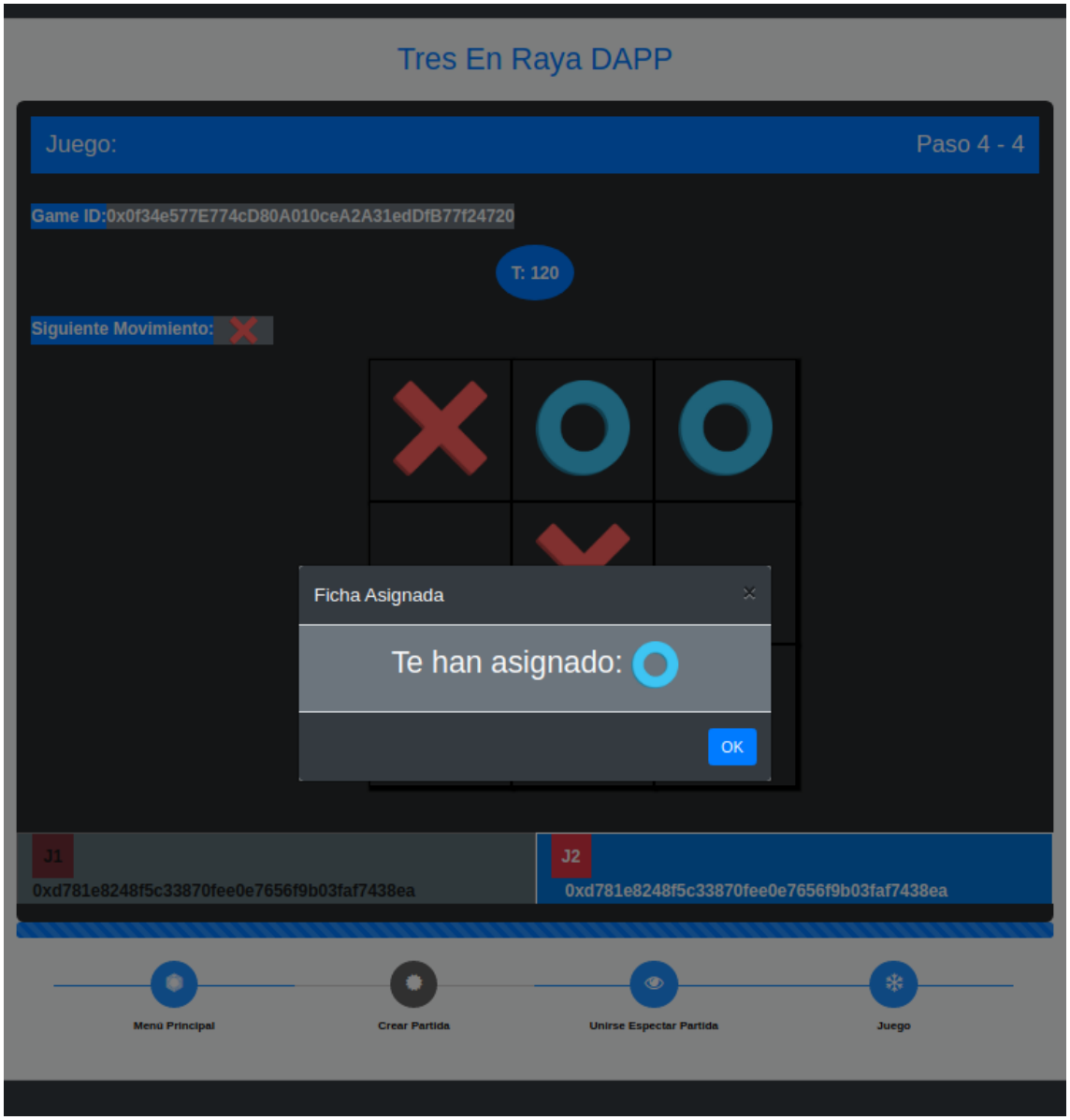


Figura 61: Prototipo Hi-Fi – Página de ficha asignada en juego (Mobile)

5) Ganador / Crear Otra Partida:

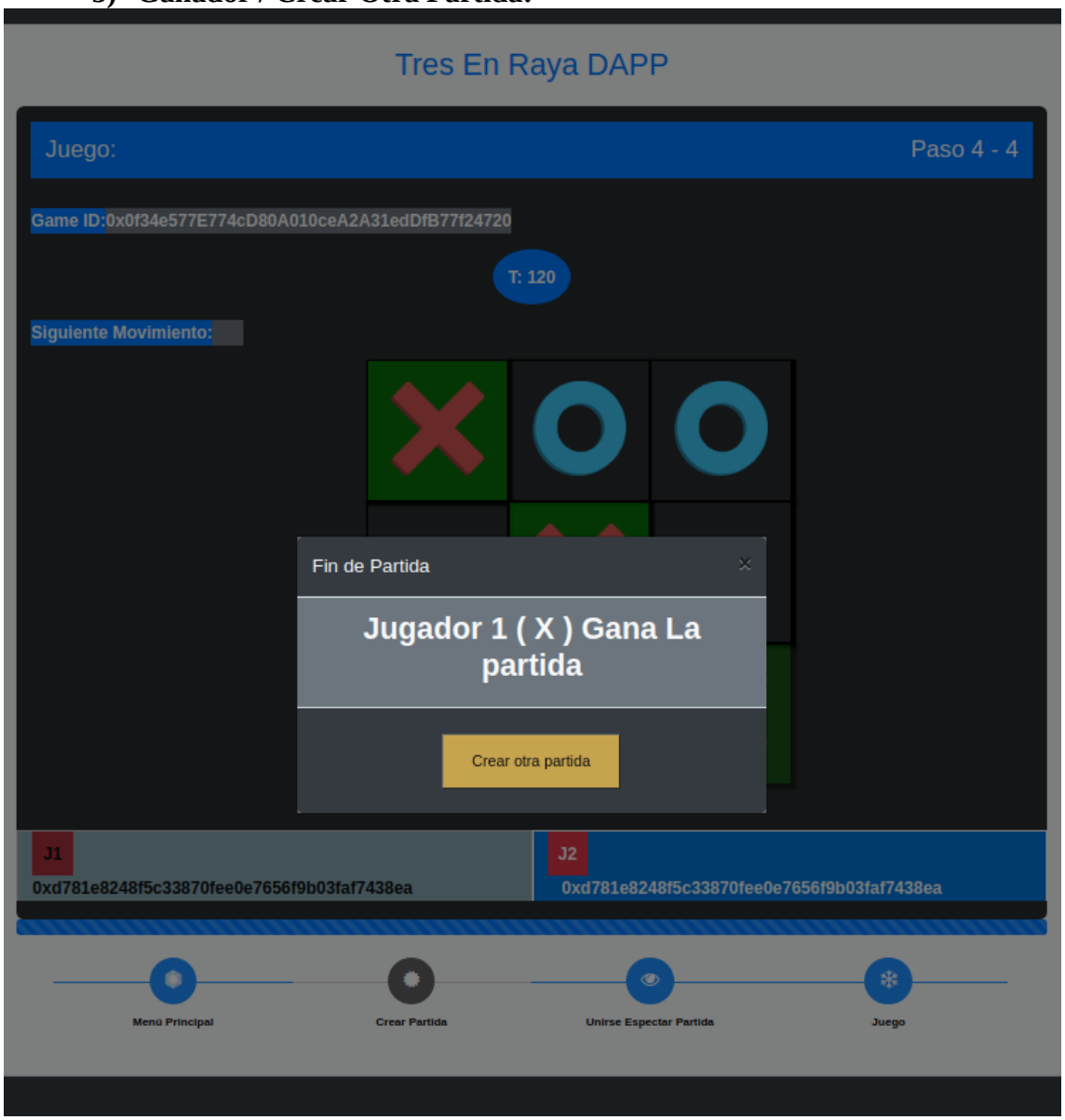


Figura 62: Prototipo Hi-Fi – Página de fin partida en juego (Mobile)

5.6 Pruebas

P_01: Acceder a la página principal	
Nombre:	Acceder a la página principal
Actores:	Usuario con Wallet en metamask.
Descripción:	• Se verifica que la aplicación metamask esté abierta y el usuario tenga una wallet en ella. • Se verifica que la acción de botones en la primera página funcione correctamente y lleven a los sitios correspondientes donde tienen que llevar. Crear Partida o Unirse a Partida.
Verificaciones:	1. Botón crear partida funciona. 2. Botón unirse a partida funciona. 3. La DAPP detecta la wallet del usuario. 4. Diseño responsive.

Figura 63: Pruebas – P_01 Acceder a la página principal

P_02: Crear Partida	
Nombre:	Crear Partida
Actores:	Usuario con Wallet en metamask.
Descripción:	• Se verifica que la aplicación metamask esté abierta y el usuario tenga una wallet en ella. • Se verifica que la acción de botones volver y crear funcionen correctamente.
Verificaciones:	1. Botón crear partida funciona. 2. Botón volver funciona 3. La DAPP detecta la wallet del usuario. 4. Diseño responsive.

Figura 64: Pruebas – P_02 Crear Partida

P_03: Unirse a partida	
Nombre:	Unirse a partida
Actores:	Usuario con Wallet en metamask.
Descripción:	• Se verifica que la aplicación metamask esté abierta y el usuario tenga una wallet en ella. • Se verifica que la acción de botones volver y Entrar funcionen correctamente.
Verificaciones:	1. Botón volver funciona. 2. Botón entrar funciona 3. La DAPP detecta la wallet del usuario. 4. Diseño responsive.

Figura 65: Pruebas – P_03 Unirse a partida

P_04: Acciones de tiempo transcurrido en partida	
Nombre:	Acciones de tiempo transcurrido en partida
Actores:	Usuario con Wallet en metamask.
Descripción:	• Se verifica que la aplicación metamask esté abierta y el usuario tenga una wallet en ella. • Se verifica que la acción de botones Pasar turno funcionen correctamente. • Se verifica que el contador haga la regresión correctamente en ambos jugadores.
Verificaciones:	1. Contador hace la regresión de 120 a 0 2. Contador = 0, aparece un botón al jugador contrario para pasar turno 3. Botón pasar turno genera la transacción correctamente y se asigna el siguiente turno. 4. Diseño responsive.

Figura 66: Pruebas – P_04 Acciones de tiempo transcurrido en partida

P_05: Cuadro de juego	
Nombre:	Cuadro de juego
Actores:	Usuario con Wallet en metamask.
Descripción:	• Se verifica que la aplicación metamask esté abierta y el usuario tenga una wallet en ella. • Se verifica que se pueda realizar tiradas correctamente. • Se verifica seguimiento de turnos.
Verificaciones:	1. Se verifica que al clicar en el cuadro del *front end* genera la transacción correcta en el contrato para posteriormente mostrar el resultado en el *front*. 2. Se verifica que el jugador contrario no pueda lanzar ficha hasta que no llegue su turno. 3. Se verifican animaciones al hacer un 3 en línea. 4. Diseño responsive.

Figura 67: Pruebas – P_05 Cuadro de juego

P_06: Gestión de turnos	
Nombre:	Gestión de turnos
Actores:	Usuario con Wallet en metamask.
Descripción:	• Se verifica que la aplicación metamask esté abierta y el usuario tenga una wallet en ella. • Se verifica que la gestión de turnos entre tiradas funcione correctamente.
Verificaciones:	1. Se verifica que después de cada tirada otorga el turno a quien corresponde. 2. Se verifica que el jugador contrario no pueda lanzar ficha hasta que no llegue su turno.

	3. Se verifican que si la partida ha finalizado no haya más gestión de turno entre jugadores. 4. Se verifica que transcurrir los 120 segundos de un turno y pulsar sobre el botón “Pasar Turno”, se otorgue el turno al jugador contrario. 5. Diseño responsive.

Figura 68: Pruebas – P_06 Gestión de turnos

P_07: Gestión de ganador	
Nombre:	Gestión Ganador
Actores:	Usuario con Wallet en metamask.
Descripción:	• Se verifica que la partida no ha acabado mientras haya espacio de tiradas en el tablero. • Se verifica que se cumplen los 3 posibles estados del tablero: Empate, victoria Jug1 o victoria Jug2. • Se verifica en cada turno si existe un ganador • Se verifica en caso de existir ganador, parar la partida y empezar otra.
Verificaciones:	1. Se verifica que si existen >=8 posiciones ocupadas en el tablero la partida es empate. 2. Se verifica que una vez haya un 3 en raya se le otorga la victoria al primer jugador que lo haya logrado. 3. Se verifica que si hay ganador, aparece un *popup* para gestionar una nueva partida. 4. Diseño responsive.

Figura 69: Pruebas – P_07 Gestión de ganador

P_08: Despliegue DAPP en testnet	
Nombre:	Despliegue DAPP en testnet
Actores:	Usuario con Wallet en metamask.
Descripción:	• Se comprueban que todas las pruebas P_01 hasta P_08 funcionen de igual manera fuera de la red local creada con Ganache y que funcione exactamente de la misma manera en una testnet como podría ser *Kovan*.
Verificaciones:	1. Se verifican todos los puntos de P_01 hasta P_08.

Figura 70: Pruebas – P_08 Despliegue DAPP en testnet

5.7 GitHub

Se adjunta enlace a plataforma GitHub donde se puede descargar todo el proyecto, así como código fuente, contratos, mockups, etc.

Además en el propio enlace de GitHub se explica brevemente como ejecutar el proyecto.

https://github.com/StRuKeRUoC/3EnRayaDAPP

6. Conclusiones y líneas de futuro

6.1 Conclusiones

A continuación, se explican las conclusiones del trabajo.

6.1.1 Objetivos Personales y del proyecto

Durante la realización de este proyecto he consolidado los conocimientos adquiridos en el grado. Así como, aportándome la capacidad de investigar sobre las tecnologías web y de *blockhain* existentes, que con tiempo y paciencia he ido sacando satisfactoriamente. Si bien es cierto, profesionalmente no me dedico a la programación, ya que vengo más de un sector IT donde llevo más temas de Arquitectura de la red, *cloud*, etc. Pero con el desarrollo de este proyecto, me ha ayudado abrir los ojos hacia otras áreas de la informática como es la programación web y el mundo de las criptomonedas.

Otro punto que destacar es la capacidad de planificar y gestionar el proyecto en unas fechas, diseñarlo e implementarlo. Mediante el seguimiento del diagrama de Gantt. Y desde la planificación del proyecto se ha definido que es exactamente lo que se necesitaba, analizando los requerimientos y funcionalidades y trabajando sobre lo que estaba planeado, sin excederse ni limitarse, con el objetivo de cumplir todos los objetivos del proyecto a tiempo.

Personalmente, creo que he alcanzado los objetivos personales que me había marcado. Así como los del proyecto. Es cierto, que se podría haber utilizado otras tecnologías para el desarrollo del mismo como hubiera podido ser *socketIO* para interconectar sesiones de usuarios en la blockchain **y por el lado del** *Front-end* como: **Angular**…

Me estuve informando un poco sobre ellos antes de empezar a desarrollar el proyecto, pero por el limitado tiempo del que disponíamos, no me permitió investigar y aprender a fondo estas dos tecnologías. Ya que, me iba a llevar más tiempo del planificado.

6.1.2 Planificación y metodología

Desde la fase inicial del proyecto se ha seguido la planificación sin problemas, aunque quizás deba reconocer que para la fase de implementación haya dedicado un poco más de tiempo del estipulado en el diagrama de Gantt. Ya que, entre investigación, diseño, vida personal, situación actual COVID-19… Me han limitado un poco a la hora de cumplir los objetivos diarios propuestos. Pero esto era un riesgo con él que ya contaba desde un principio de la planificación del proyecto y es por eso por lo que dejé días sobrantes para dedicar a las partes que realmente tenían la necesidad de tener esta dedicación, como lo han sido la implementación y la documentación.

La metodología en cascada ha ayudado mucho a dividir la planificación del proyecto dentro de todas sus fases y poder recorrer todas las etapas cumpliendo los objetivos propuestos. Como ya mencioné anteriormente, el período de implementación se hizo bastante corto. Pero gracias a la planificación, a la división y subdivisión de los trabajos hizo que todo el proceso de iteración de la metodología cascada fuera avanzando dentro de los tiempos establecidos y sin perder tiempo en tomas de decisiones u otros extras.

En definitiva, creo que la planificación aportada y la metodología elegida han dado sus frutos a cumplir con los objetivos principales propuestos, desarrollo del proyecto y finalmente, entrega del producto final.

6.2 Líneas de futuro

Debido al limitado tiempo para llevar a cabo todo el proyecto, la aplicación tiene bastantes aspectos a mejorar que se proponen cómo trabajo a futuro.

A nivel de aplicación se pueden incorporar diferentes funcionalidades como podrían ser:

- **Interconectar usuarios:** *SocketIO* es un websocket que hubiera permitido interconectar usuarios entre sí y saber cuándo se crean y se borran las sesiones de conexión de los usuarios.

- **Crear una interfaz de chat para publicar hashes de partidas:**

  Con el fin de tener la comunicación de la plataforma entre usuarios de forma centralizada y no utilizar otros métodos de comunicación, podría ser una buena idea implementar una pequeña interfaz de chat donde los usuarios pudieran hablar e intercambiar sus wallets con el fin de poder realizar partidas.

- **Uso de meta transacciones:** Con el fin de evitar que los usuarios firmen transacciones entre tiradas, sería interesante añadir un mecanismo mediante meta transacciones donde las tiradas se pudieran hacer automáticamente sin pedir permiso por tirada desde metamask.

- **Mejorar aspectos de diseño responsive:** Se podría mejorar el encuadre de la información a la hora de mostrarse, sobre todo en dispositivos móviles, donde el contenido varía de posición o desaparece por su limitada capacidad de pantalla.

7. Glosario

NFT: (Non – Fungible Token) es un tipo esencial de token criptográfico que representa algo único.

DappRadar: Una popular plataforma que ofrece análisis y seguimiento de billeteras para los mercados de aplicaciones descentralizadas (App) y tokens no fungibles (NFT).

Contratos inteligentes (*Smart contracts*): Por «contratos inteligentes» se entiende cualquier tipo de contrato cuya característica principal es que se puede ejecutar de forma automática sin que sea necesaria la intervención de un tercero.

Solidity: Solidity es un lenguaje de programación de tipo estático diseñado para desarrollar contratos inteligentes que se ejecutan en la máquina virtual de Ethereum, también conocida como EVM.

8. Bibliografía

Ethereum. *(2017). OBS.* Recuperado de https://ethereum.org/es/developers/docs/dapps/

Comillas. *(2019). El papel transformador del BlockChain en los servicios financieros.* Recuperado De https://repositorio.comillas.edu/xmlui/bitstream/handle/11531/27373/TFG%20-%20Garcia%20Joga%2C%20Iciar.pdf?sequence=1&isAllowed=y

Techedgegroup.com. (2019). *Introducción a la blockchain*. Recuperado de https://www.techedgegroup.com/es/blog/introduccion-blockchain-explicacion-evolucion

Platzi.com . *(2015). Que es Bitcoin, cómo funciona y Todo lo que debes saber.* Recuperado *de* https://platzi.com/blog/que-es-bitcoin-btc/?utm_source=google&utm_medium=paid&utm_campaign=14603491644&utm_adgroup=&utm_content=&gclid=EAIaIQobChMInOr_isqL9gIVrmxvBB0sjQZuEAAYASAAEgK9SvD_BwE&gclsrc=aw.ds

Ethereum.org . *(2022). Que es Ethereum.* Recuperado de https://ethereum.org/es/what-is-ethereum/

Ethereum.org. *(2022). Introducción a Ethereum.* Recuperado de https://ethereum.org/es/developers/docs/intro-to-ethereum/

Bit2me.com. *(2021). Qué son las DAPPS?.* Recuperado de https://academy.bit2me.com/que-son-las-dapps/

Businessinsider.es. *(2021). Top juegos con los que ganarás criptomonedas.* Recuperado de https://www.businessinsider.es/7-juegos-blockchain-ganar-criptomonedas-cada-victoria-935367

Github. *(2018). Referencia*: Tictac chain. Recuperado de https://github.com/beyretb/Tic-Tac-Chain

Cointelegrah.com. *(2021). 5 razones por las que las economías de videojuegos basadas en blockchain son el futuro.* Recuperado de https://es.cointelegraph.com/news/5-reasons-why-blockchain-based-gaming-economies-are-the-future

Observatorioblockchain. *(2021). Lista de juegos play to earn.* Recuperado de https://observatorioblockchain.com/blockchain/la-lista-de-los-5-juegos-blockchain-play-to-earn-que-mas-usuarios-tienen/

Bit2me.com. *(2021).Blockchain y mercados financieros.* Recuperado de http://dev.scielo.org.pe/scielo.php?script=sci_arttext&pid=S0251-34202018000200013

Parásito virtual. *(2010).* Modelo cascada. Recuperado de https://parasitovirtual.wordpress.com/tag/modelo-en-cascada/

Ionos. *(2019). Análisis de mercado.* Recuperado de https://www.ionos.es/startupguide/gestion/que-es-el-analisis-de-mercado/

imgbin. *(2019). Imágenes Tres en Raya.* Recuperado de **https://imgbin.com/download/y4CKNtjH**

Draw.io. *(2020). Wireframes, esquemas, diagramas. Recuperado* de http://draw.io

Médium.com *(2018).* SQL vs NoSQL ventajas/desventajas. Recuperado de https://medium.com/@marlonmanzo/sql-vs-nosql-ventajas-y-desventajas-849ccc9db3d4

Nextu. *(2018). Ventajas y desventajas del javascript*. Recuperado de https://www.nextu.com/blog/conoce-las-ventajas-y-desventajas-de-javascript/

Platzi.com. *(2015). Ajax con jquery.* Recuperado de https://platzi.com/blog/ajax-con-jquery/

logomaster. *(2016). Diseño de logotipos.* Recuperado de https://logomaster.ai/es/

Solidity. *(2022). Solidity.* Recuperado de https://es.wikipedia.org/wiki/Solidity

Jquery. *(2022). Jquery.* Recuperado de https://jquery.com/

Brackets. *(2022). Solidity.* Recuperado de http://brackets.io/

Webpack. *(2020). ¿Qué es Webpack?*. Recuperado de https://medium.com/@victor.valencia.rico/qu%C3%A9-es-webpack-75cb56559759

CiberNinjas. *(2020). Remix IDE*. Recuperado de https://ciberninjas.com/editor-remix-ide-ethereum-dapps/#:~:text=Remix%20IDE%20es%20una%20aplicaci%C3%B3n,en%20Solidity%20y%20aplicaciones%20DAPPS

Beincrypto. *(2021). Importancia de Solidity.* Recuperado de https://es.beincrypto.com/importancia-solidity-lenguaje-programacion-ethereum-eth/

programmerclick. *(2021). Introducción a Bytcode y Opcode.* Recuperado de https://programmerclick.com/article/72171909695/

GeeksforGeeks. *(2020). Application Binary Interface(ABI) in Ethereum Virtual Machine.* Recuperado de https://www.geeksforgeeks.org/application-binary-interfaceabi-in-ethereum-virtual-machine/

9. Anexos

Lista de anexos:

9.1 ANEXO 1: Planificación del Proyecto

	EDT	Modo Tarea	Nombre de tarea	Duración	Comienzo	Fin
1	**0**		**Proyecto Tres En Raya**	**118,04 días**	**mi. 16/02/22 0:00**	**sá. 24/06/23 11:01**
2	**1**		**Plan de trabajo**	**13,96 días**	**mi. 16/02/22 0:00**	**ma. 01/03/22 23:12**
3	1.1		Elección del tema	1 día	mi. 16/02/22 0:00	ju. 17/02/22 0:01
4	1.2		Búsqueda de proyectos similares	2,33 días	ju. 17/02/22 0:01	sá. 19/02/22 8:02
5	1.3		Definir las herramientas necesarias	2,96 días	sá. 19/02/22 8:02	ma. 22/02/22 7:04
6	1.4		Enfoque y método	1 día	ma. 22/02/22 7:04	mi. 23/02/22 7:05
7	1.5		Elaboración de la memória PAC 1	6,29 días	mi. 23/02/22 7:05	ma. 01/03/22 14:06
8	**1**		**Definición de los requisitos**	**27,95 días**	**mi. 02/03/22 0:00**	**ma. 29/03/22 23:15**
9	2.1		Definición detallada de requisitos	2,62 días	ma. 01/03/22 14:06	vi. 04/03/22 5:07
10	2.2		Definición de tareas y planificación	4,62 días	sá. 05/03/22 5:07	mi. 09/03/22 20:09
11	2.3		Definición de objetivos	3,62 días	mi. 09/03/22 20:09	do. 13/03/22 11:06
12	2.4		Definición de alcance de proyecto	3,96 días	mi. 16/03/22 21:03	do. 20/03/22 20:10
13	2.5		Elaboración de la memória PAC 2	8,62 días	do. 20/03/22 20:04	ma. 29/03/22 11:07
14	**3**		**Diseño**	**26,98 días**	**mi. 30/03/22 0:00**	**ma. 26/04/22 0:02**
15	3.1		Definición perfiles de usuario	3,95 días	ma. 29/03/22 11:07	sá. 02/04/22 10:02
16	3.2		Diseño conceptual	4,62 días	ma. 05/04/22 10:11	do. 10/04/22 1:10
17	3.3		Prototipo	2,96 días	lu. 11/04/22 1:12	ju. 14/04/22 0:18
18	3.4		Casos de uso	4,62 días	vi. 15/04/22 0:13	ma. 19/04/22 15:09
19	3.5		Descripción de la arquitectura	2,62 días	mi. 20/04/22 15:13	sá. 23/04/22 6:09
20	3.6		Elaboración memória PAC 3	1,96 días	do. 24/04/22 6:15	ma. 26/04/22 5:16
21	**1.2.7**		**Implementación**	**34,61 días**	**mi. 27/04/22 5:16**	**ma. 31/05/22 20:23**
22	4.1		Instalación de todo el software necesario	2,63 días	ma. 26/04/22 5:16	ju. 28/04/22 20:25
23	**1.2.7.2**		**Desarrollo de la DAPP**	**27,13 días**	**sá. 30/04/22 20:23**	**vi. 27/05/22 23:59**
24	4.2.1		Back-End	14,62 días	sá. 30/04/22 20:23	do. 15/05/22 11:31
25	4.2.2		Front-End	9,63 días	do. 15/05/22 11:31	mi. 25/05/22 2:44
26	**5**		**Memoria Final**	**5 días**	**ju. 26/05/22 0:00**	**ma. 31/05/22 0:02**
27	5.1		Documentación Memoria Final (PAC4)	2,96 días	mi. 25/05/22 2:44	sá. 28/05/22 1:51
28	5.2		Revisión de la documentación	0,96 días	lu. 30/05/22 1:42	ma. 31/05/22 0:47
29	5.3		Entregable del Producto	1 día	ma. 31/05/22 0:45	mi. 01/06/22 0:46
30	**6**		**Vídeo de presentación**	**6 días**	**mi. 01/06/22 8:45**	**ma. 07/06/22 8:51**
31	6.1		Elaboración de diapositivas	2 días	mi. 01/06/22 8:45	vi. 03/06/22 8:47
32	6.2		Video + Entregable	3,29 días	sá. 04/06/22 8:00	ma. 07/06/22 15:00
33	**7**		**Defensa Proyecto**	**11,74 días**	**lu. 13/06/22 6:05**	**vi. 24/06/22 23:59**

9.2 ANEXO 2: GANTT