

**UNIVERSITAT POLITÈCNICA DE CATALUNYA**

**School of Professional & Executive Development**



**UNIVERSITAT POLITÈCNICA DE CATALUNYA**  
**BARCELONATECH**

---

**School of Professional & Executive Development**

**PROYECTO FIN DE POSTGRADO**  
**Tecnologías Blockchain**



**Jordi Girón**  
**Guillem Córdoba**  
**Mario Campo**

*Septiembre 2018*

## INDICE DE CONTENIDOS

1	Introducción.....	3
1.1	Resumen .....	3
1.2	Objetivos.....	4
1.3	Concepto y estructura .....	4
1.3.1	Actores .....	4
1.3.2	Periodos de interacción.....	5
2	Front-End.....	8
2.1	Aplicación web .....	8
2.1.1	Consideraciones generales.....	8
2.1.2	“Deployment” de la aplicación web .....	9
2.2	Estructura de la aplicación.....	9
2.2.1	Crear nuevo concurso .....	9
2.2.2	“Dashboard” principal .....	10
2.2.3	Vista de concurso.....	10
2.2.4	Vista de candidaturas .....	11
2.2.5	Concurso finalizado .....	12
3	Back-End .....	13
3.1	Sistema de archivos interplanetarios (IPFS).....	13
3.2	Contrato Inteligente .....	14
3.2.1	Interacción con el Frontend .....	14
3.2.2	Filosofía básica de contrato .....	14
3.2.3	Métodos de protección.....	15
4	Integración, pruebas y resultados .....	17
4.1	Herramientas y Utilidades .....	17
4.1.1	Ganache .....	17
4.1.2	Truffle .....	17
4.1.3	Metamask.....	18
4.2	Testing .....	18
4.2.1	Tests utilizados .....	18
4.2.2	Resultados.....	20
5	Conclusiones y trabajo futuro.....	22
5.1	Conclusiones.....	22
5.2	Posibles integraciones con proyectos externos.....	22
5.3	Trabajo futuro .....	22
	Referencias .....	24
	Glosario .....	I

## INDICE DE ILUSTRACIONES

ILUSTRACIÓN 1-1 – PERIODOS DE INTERACCIÓN.....	5
ILUSTRACIÓN 2-1 - EJEMPLO DE FUNCIÓN DESHABILITADA Y MENSAJE DE EXPLICACIÓN .....	8
ILUSTRACIÓN 2-2 – INTERFAZ NUEVO CONCURSO .....	9
ILUSTRACIÓN 2-3 – INTERFAZ PRINCIPAL O “DASHBOARD” .....	10
ILUSTRACIÓN 2-4 – INTERFAZ DE VISTA DE CONCURSO.....	11
ILUSTRACIÓN 2-5 – INTERFAZ DE VISTA DE CANDIDATURAS .....	11
ILUSTRACIÓN 2-6 – INTERFAZ DE CONCURSO FINALIZADO.....	12
ILUSTRACIÓN 3-1 – COMPARACIÓN IPFS VS HTTP.....	13
ILUSTRACIÓN 4-1 – GANACHE UI .....	17
ILUSTRACIÓN 4-2 – METAMASK UI.....	18
ILUSTRACIÓN 4-3 – GANACHE ACCOUNTS.....	19
ILUSTRACIÓN 4-4 – TRUFFLE TEST 1.....	20
ILUSTRACIÓN 4-5 – TRUFFLE TEST 2.....	21

# 1 Introducción

---

## 1.1 Resumen

La lista de los posibles usos de la tecnología blockchain abarca muchos sectores diferentes con la creencia es que casi todas las facetas de la vida, desde las finanzas, tecnología, economía y sociología, pueden ser afectadas por esta fuerza descentralizada.

Sin embargo, es en la política y especialmente en los métodos electivos, donde hay una expectativa creándose sobre la influencia positiva que la tecnología blockchain puede llevar a las urnas. Las elecciones han sido objeto de escrutinio y corrupción durante tanto tiempo que muchos en el ámbito de la tecnología, así como en los comités electorales, están viendo a la blockchain como el futuro de elecciones justas.

Ya hay algunos casos de uso en los que la blockchain ha llegado a la ayuda de las elecciones. Los diferentes países y organizaciones han empezado a experimentar con la inmutable contabilidad distribuida que ofrece transparencia y seguridad.

El interés en la tecnología blockchain relevando los métodos tradicionales de elección tiene ventajas potenciales debido a la gran actualización tecnológica de cómo las elecciones se mantienen actualmente. Muchas elecciones nacionales todavía se llevan a cabo mediante un sistema basado en el papel, dejando abierto enormes agujeros de seguridad, fraude, y corrupción.

La blockchain ofrece un sistema actualizado para los votantes que podría solucionar estos problemas.

Sus activos tradicionales, tales como su transparencia, permiten que los votos sean seguidos, contados y correlacionados con muchas fuentes diferentes, manteniendo la privacidad de los electores debido a las transacciones anónimas a lo largo de la blockchain.

Paralelamente, suele suceder lo mismo en procesos electivos más cotidianos como, por ejemplo, concursos de fotografía, selección de ideas o cualquier concepto que pueda ser objeto de voto), entre los que se debe elegir uno entre varios candidatos.

## 1.2 Objetivos

Por los motivos anteriormente comentados e intentando añadir confianza e integridad en los métodos electivos más cotidianos, se ha pensado en realizar una DApp que ayude a buscar transparencia y seguridad en estos casos.

Dicha Dapp tendrá un front-end basado en web, desde la cual se podrá generar un nuevo concurso o proceso electivo, indicando los diferentes parámetros para que dicho proceso quede sujeto a unas normas establecidas claras y transparentes.

Con ello obtenemos el objetivo principal de este proyecto el cual es permitir la creación de concursos entre desconocidos sin que tengan que confiar entre ellos, sino confiando en la plataforma la cual esté lo mayormente descentralizada posible y que nos proporcione la mayor transparencia en los procesos electivos.

## 1.3 Concepto y estructura

Se va a presentar una solución en la cual desde un front-end basado en un navegador web estándar se puede interactuar con toda la arquitectura del concurso.

### 1.3.1 Actores

Los diferentes actores que se involucrarán en el proyecto son:

- Creador del Concurso:  
Esta persona será la administradora del concurso y la que seleccionará los diferentes jueces.
- Jueces:  
Serán las personas que valorarán las diferentes candidaturas. Dichas personas podrán ser de dos tipos
  - Personas individuales: personas únicas identificadas por su ethereum address.
  - Grupo de personas: En éste caso será un contrato inteligente de terceros el cual podrá englobar a varios jueces, retornando a nuestro contrato el resultado final. Dicho contrato a terceros tendrá que encargarse de realizar sus propios criterios de selección.

Dado que no es lo mismo el voto de un juez individual, juez experto o una comunidad de jueces agrupados por un contrato inteligente, el administrador del concurso asignará una ponderación a cada juez para que exista un peso de voto justo.

- Participantes:  
Comprende los diferentes usuarios que realizan una candidatura para optar al premio del concurso.

### 1.3.2 Periodos de interacción

Los tiempos en la que los diferentes podrán interactuar con el sistema vendrán asignados por el creador del concurso y seguirá la siguiente estructura:

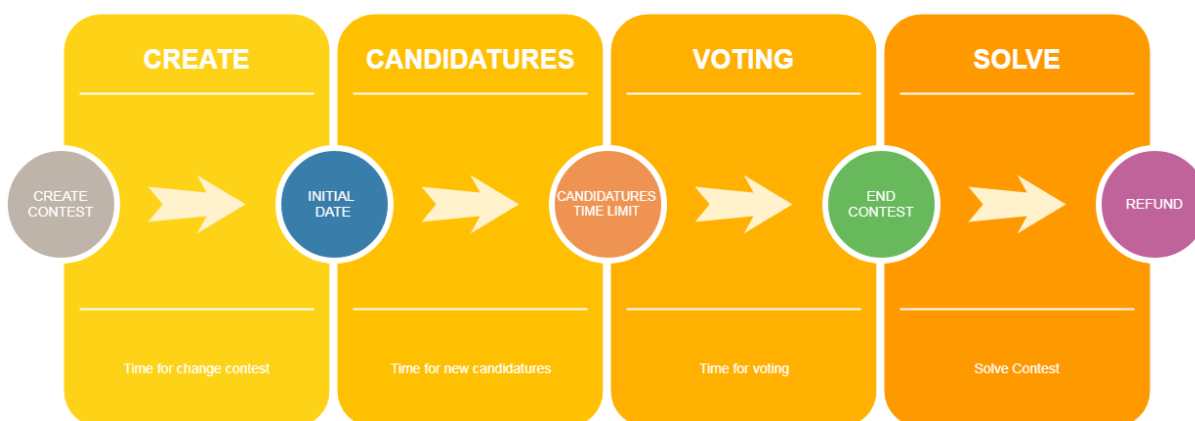


Ilustración 1-1 – Periodos de interacción

- **Fase Inicial:**

Este es el momento inicial de creación del concurso donde el administrador o creador del concurso establece las normas de tiempo y asignación de jueces.

**Fechas:** Desde que se crea el concurso hasta la fecha inicial del concurso.

**Detalles técnicos:**

El creador del concurso define los siguientes parámetros:

- Título del concurso
- Lista de etiquetas
- Fecha de inicio del concurso
- Fecha límite de presentación de candidaturas
- Fecha límite de revisión y votación de candidaturas
- Imagen y descripción de soporte, donde se pueden especificar las bases del concurso.
- Juez inicial: dirección, nombre y peso en votos.
- Premio del concurso, en ETH
- “Stake” para participar en el concurso

Cada concurso se identifica con: la dirección del creador, la fecha de creación de concurso y su título. Por lo tanto, no podrá haber dos concursos que compartan estas tres propiedades.

Todos estos parámetros se guardan directamente en la blockchain, excepto la imagen y la descripción de soporte, ya que serían demasiado caros. Estos se guardan en IPFS, y su hash compartido se guarda en la blockchain como un parámetro más.

En esta fase el creador del concurso puede quitar y añadir jueces, asignándoles a cada uno un peso específico en votos. Hay una sola condición: siempre debe haber como mínimo un juez asignado al concurso. De lo contrario, se podría dar el caso que se entre en la fase de admisión de candidaturas sin ningún juez, y el premio del concurso no podría ser otorgado.

Cada juez tiene los siguientes parámetros: un nombre, una dirección y un peso específico en votos. Como ya hemos comentado antes, esta dirección puede pertenecer tanto a una persona individual o a un agente externo dentro del ecosistema de Ethereum. Más adelante se explorarán opciones de integración con otros proyectos.

- **Fecha de admisión de candidaturas**

Finaliza el periodo de revisiones y modificación de jueces, y en el que a partir de este momento cualquiera podrá presentar sus candidaturas. Para presentar las candidaturas, los propietarios tendrán que realizar un abono en concepto de “stake”

**Fechas:** desde que la fecha inicial del concurso hasta la fecha límite de admisión de candidaturas.

**Detalles técnicos:**

Al ser muy costoso guardar el contenido de una candidatura en la blockchain, este se guardará en IPFS. Sin embargo, en esta fase no se sube el contenido de la candidatura a IPFS, sino que solo se guarda su hash en la blockchain. De esta manera se evita la posibilidad de copia, ya que el contenido en sí de las candidaturas no es visible.

Otras restricciones de esta fase:

- A partir de esta fase ya no se podrán cambiar los jueces del concurso.
- El “stake” presentado por cada participante para cada una de las candidaturas tiene que coincidir con el especificado en el concurso.
- No se permite presentar el hash de una candidatura que ya haya sido presentado en el mismo concurso por cualquier participante.
- No hay límite de candidaturas por persona.

- **Fase de revisión:**

A partir de este momento no se podrán presentar más candidaturas y comenzará el periodo en el que los jueces pueden votar sus preferidas. De igual manera, los jueces pueden invalidar candidaturas que no cumplan las normas del concurso.

**Fechas:** desde la fecha límite de admisión de candidaturas hasta la fecha final del concurso.

**Detalles técnicos:**

En esta fase, los participantes del concurso **deben** subir sus candidaturas a IPFS. Sólo entonces podrán ser vistas por los jueces y votadas. Nuestra aplicación web ofrece la funcionalidad para hacerlo de la siguiente manera:

1. El usuario selecciona su foto de candidatura, y se calcula su hash.
2. Se compara el hash con todas las candidaturas presentadas en la fase anterior.
3. Si el hash coincide con alguna de las candidaturas, se sube el contenido a IPFS.

Cabe notar que el usuario puede hacer el proceso de manera externa sin ningún problema, pero solo si el hash del contenido coincide con el presente en la blockchain se podrá ver en la aplicación web, así como ser votado.

De otro lado, cualquier juez puede cancelar cualquier candidatura, dando una razón para ello. El “stake” de esa candidatura se pierde, para evitar conflicto de interés a los jueces y creadores del concurso, que se pueden ver incentivados a cancelar cuantas más candidaturas mejor.

- **Fase de finalización:**

En este periodo se finaliza la posibilidad de voto por parte de los jueces y se realiza el cálculo de ganadores.

Los diferentes participantes (exceptuando las candidaturas penalizadas) podrán solicitar las cantidades inicialmente ingresadas en concepto de “stake”. El ganador recibirá también el importe del premio.

**Fechas:** desde la fecha de finalización del concurso

**Detalles técnicos:**

Cualquier agente puede realizar la transacción para hacer el recuento de votos, ya que es automática y todos los votos están ya recogidos. Si solo los jueces o el creador del concurso tuvieran esa opción, se crearía un posible conflicto de intereses en caso de que no les gustara el ganador podrían decidir no resolver el concurso y el ganador se quedaría sin premio.

Dicho esto, solamente los participantes del concurso pueden retirar el “stake” que hayan puesto para cada una de sus candidaturas. En caso de que hayan ganado el concurso, también se les enviará el premio. Esto se hace mediante la técnica “Withdrawal from contract”, es decir, en lugar de enviar todos los “stake” a los participantes de una vez, se pone disponible para que los mismos puedan retirarlos con una transacción. De esta manera se evitan posibles ataques que puedan manipular todos los fondos gestionados por un concurso.



## 2 Front-End

---

### 2.1 Aplicación web

#### 2.1.1 Consideraciones generales

Hemos escogido construir una SPA a por el tipo de comunicación que debemos tener con el smart-contract. Si queremos una aplicación verdaderamente descentralizada y con la que podamos asegurar el desarrollo de concursos de manera segura, no podemos depender de funcionalidad que se ejecute en un servidor centralizado. Eso supondría introducir un SPOF (“Single Point of Failure”) dentro de un sistema que por lo demás no contiene ningún punto central de posible fallo.

Así, todas las operaciones necesarias des del punto de vista del front-end se ejecutarán directamente dentro del navegador del usuario a través de la aplicación web. Como consecuencia, debemos tener en cuenta que un usuario con conocimientos avanzados podría modificar nuestro código y intentar atacar el smart-contract a través de él. Por lo tanto, debemos tener mucha precaución en securizar el sistema dentro del smart-contract, exigiendo todos los *require* necesarios siempre y verificando que el creador de la transacción tiene permisos para ejecutarla.

Por otro lado, por consistencia y claridad para los usuarios bien intencionados, todas las acciones no permitidas por el smart-contract se han deshabilitado también en el frontend, dando una explicación de porqué esa función en concreto no está activada.

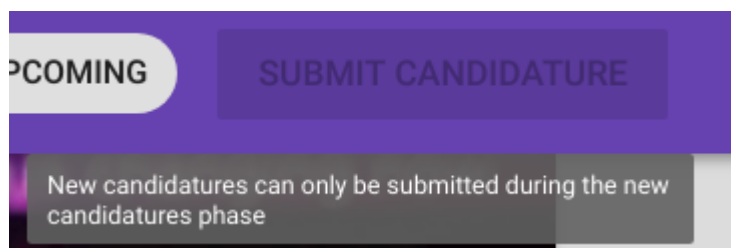


Ilustración 2-1 - Ejemplo de función deshabilitada y mensaje de explicación

Gran parte de funciones dependen del momento actual para poder activarse o desactivarse. En consecuencia, hemos preferido deshabilitar las funcionalidades en vez de esconderlas para que el usuario pueda saber dónde se encontrarán una vez estén activadas, dando indicaciones de ello al usuario.

Por otro lado, se han integrado las rutas URL con el estado de la aplicación web, de tal manera que dependiendo de la ruta en la que se encuentre la URL, la aplicación mostrará un concurso determinado, el “dashboard” ... Así, se facilita mucho la compartición de los concursos fuera de la aplicación, ya que con un sólo link ya se puede acceder un concurso concreto. Creemos que esto puede beneficiar en gran medida que la aplicación sea utilizada por más usuarios.

Para el desarrollo de la aplicación web, se ha utilizado el framework *Angular*, por sus diversas herramientas de gestión de SPAs (“Single Page Applications”). También por su integración con *Angular Material* (<https://material.angular.io>), librería de componentes que nos han facilitado mucho la creación de la UI/UX.

## 2.1.2 “Deployment” de la aplicación web

Para poder acceder a la aplicación web desde un navegador, estas deben ser servidas por un servidor centralizado o, más recientemente, pueden estar guardadas en IPFS. Si se utiliza la segunda, ya no debemos confiar en que el servidor centralizado continúe sirviendo la aplicación, sino que está estará disponible en IPFS sin censura posible.

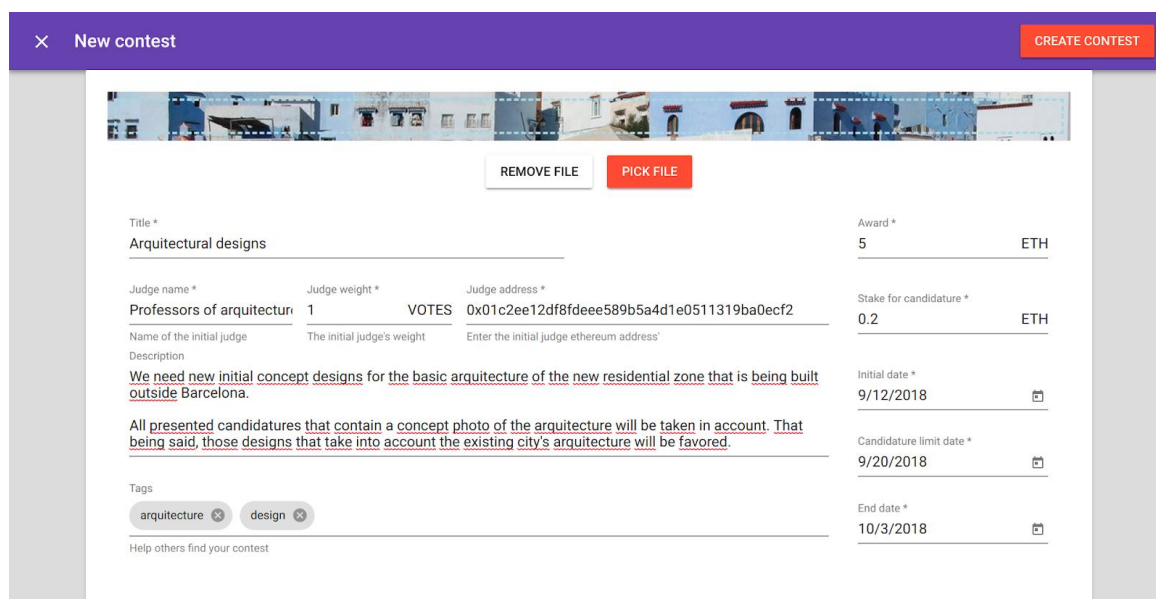
Desafortunadamente, subir la aplicación web a IPFS implicaría romper la integración que *Angular* hace con las rutas, ya que IPFS aún no soporta links del tipo “<https://ipfs.io/ipns/QmNx6UVnpmMDD1h7F8xwkPUdbstZjvEnksrJZcZM36dCoz/contests/ongoing>”. Cuando intentamos acceder a este link, IPFS entiende que debe haber un archivo en la ruta “contests/ongoing”, cuando en realidad esas son rutas añadidas por *Angular* que dan la integración con la aplicación web que comentamos en el apartado anterior.

Por consecuencia, por el momento hemos hecho el “deploy” de la aplicación en un servidor centralizado (<https://cryptocontests.github.io/cryptocontests/>), esperando a que la integración entre los dos sistemas mejore y podamos utilizar IPFS para ello.


## 2.2 Estructura de la aplicación

### 2.2.1 Crear nuevo concurso

En esta vista se deben introducir todos los datos ya comentados en apartados anteriores para la creación de un concurso. Es donde se determina el creador del concurso, y en este momento transfiere el premio del concurso al smart-contract, que a partir de la transacción queda bloqueado.



**New contest** CREATE CONTEST



REMOVE FILE PICK FILE

Title * Architectural designs			Award * 5 ETH
Judge name * Professors of architectur	Judge weight * 1	Judge address * 0x01c2ee12df8fdeee589b5a4d1e0511319ba0ecf2	Stake for candidature * 0.2 ETH
Name of the initial judge The initial judge's weight Enter the initial judge ethereum address*			Initial date * 9/12/2018
Description We need new initial concept designs for the basic architecture of the new residential zone that is being built outside Barcelona. All presented candidatures that contain a concept photo of the architecture will be taken in account. That being said, those designs that take into account the existing city's architecture will be favored.			Candidature limit date * 9/20/2018
Tags architecture design			End date * 10/3/2018

Help others find your contest

Ilustración 2-2 – Interfaz Nuevo Concurso

### 2.2.2 “Dashboard” principal

El “dashboard” constituye la pantalla inicial de la aplicación, desde donde se puede acceder a todas las otras funcionalidades y buscar concursos.

En este apartado se pueden visualizar todos los concursos que están disponibles, clasificados por fases. Así, cada tipo de usuario (juez, participante o creador de concursos) puede acceder a los concursos más relevantes para él de manera más fácil.

También hay opción de buscar concurso por título o filtrarlos por etiquetas, para facilitar también el acceso a los concursos interesantes al usuario cuando haya muchos concursos.

Por último, también se puede acceder a la vista de creación de un nuevo concurso, así como ver la lista de transacciones que ha realizado el usuario y poder visualizarlas en *Etherscan*.

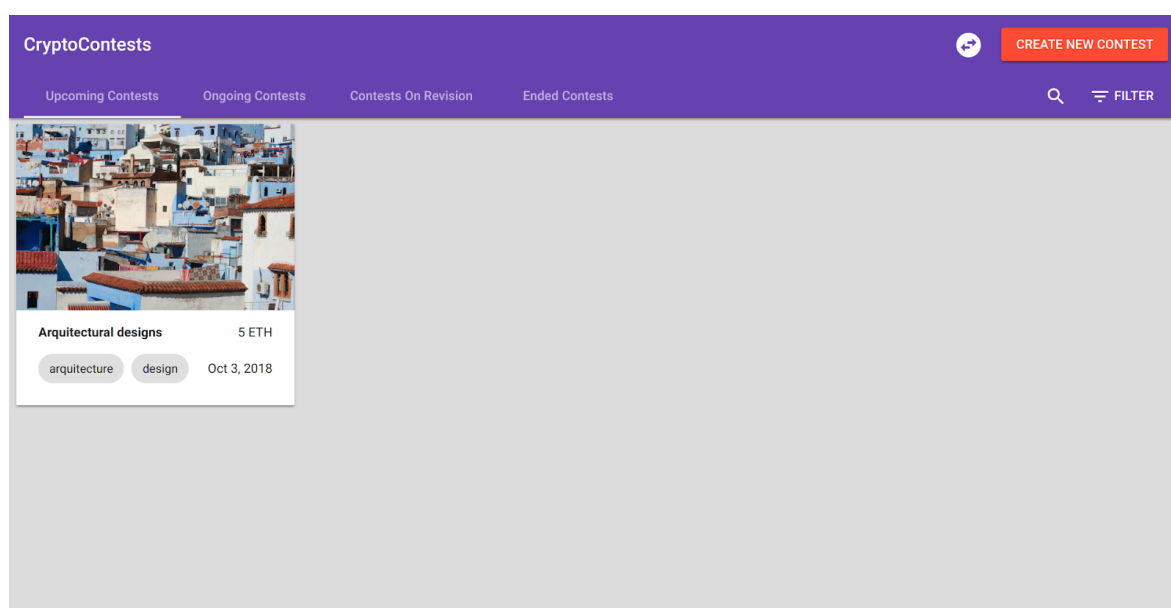


Ilustración 2-3 – Interfaz principal o “Dashboard”

### 2.2.3 Vista de concurso

Esta pantalla constituye el principal activo de la aplicación: se pueden visualizar todos los detalles de un concurso: título, descripción, jueces, premio, fase en la que se encuentra... A medida que vaya avanzando el ciclo de vida de un concurso, todos los usuarios podrán ver aquí su estado.

Esta pantalla irá cambiando ligeramente de funcionalidad para adaptarse a la fase en la que se encuentra el concurso. Dependiendo de ella, las funciones disponibles en esa fase concreta se activarán sólo para los usuarios adecuados (por ejemplo, añadir juez sólo se activa para el creador del concurso).

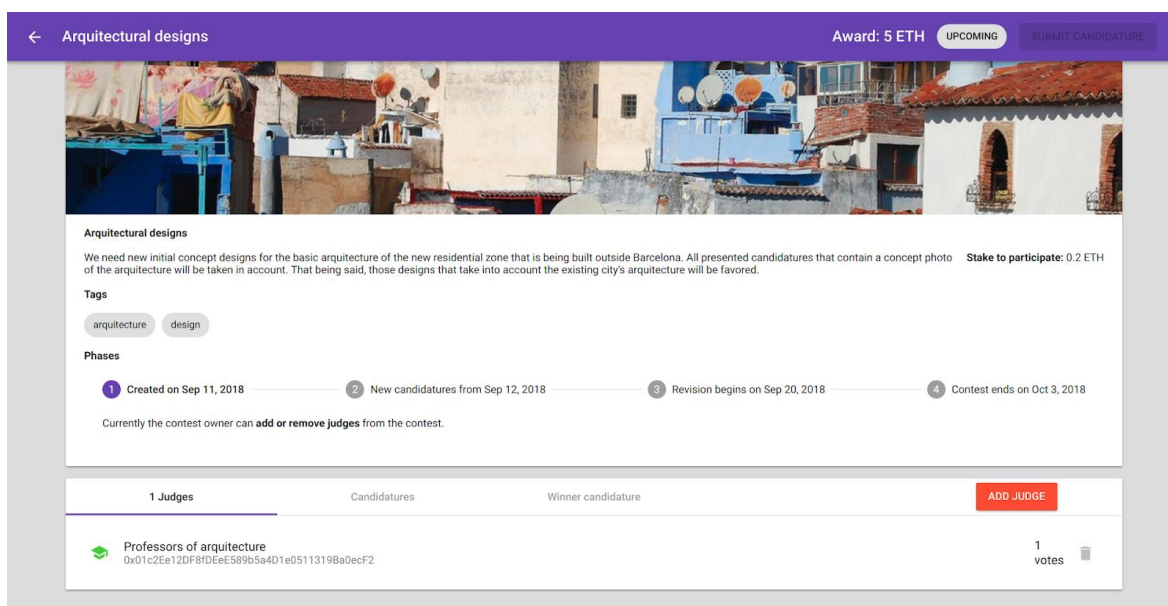


Ilustración 2-4 – Interfaz de vista de concurso

## 2.2.4 Vista de candidaturas

Una vez en la fase de revisión de candidaturas, éstas se podrán visualizar en forma de galería dentro de la vista de un concurso.

También aquí los jueces del concurso podrán votar o cancelar la candidatura, dando una razón para ello.

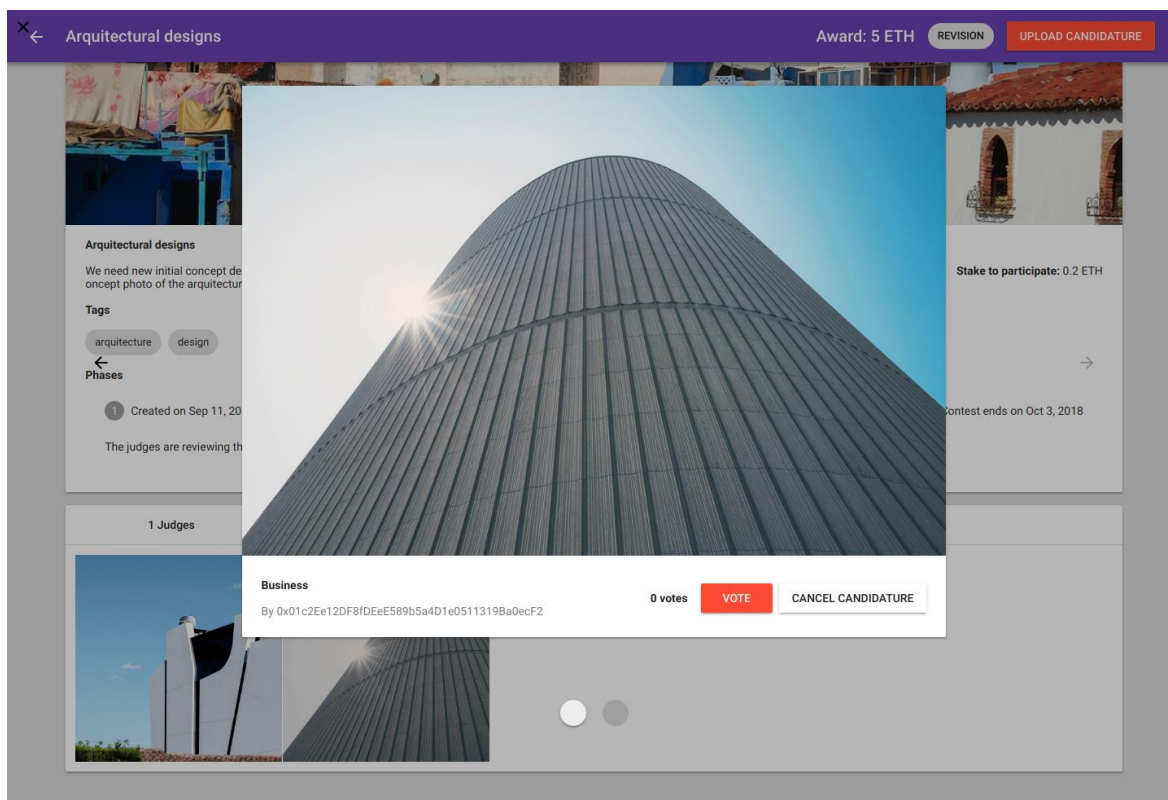


Ilustración 2-5 – Interfaz de vista de candidaturas

## 2.2.5 Concurso finalizado

Cuando el concurso finaliza, se permite a cualquier agente resolverlo. Seguidamente, se muestra el ganador del concurso, y se permite a los participantes retirar el “stake” y/o el premio del concurso.

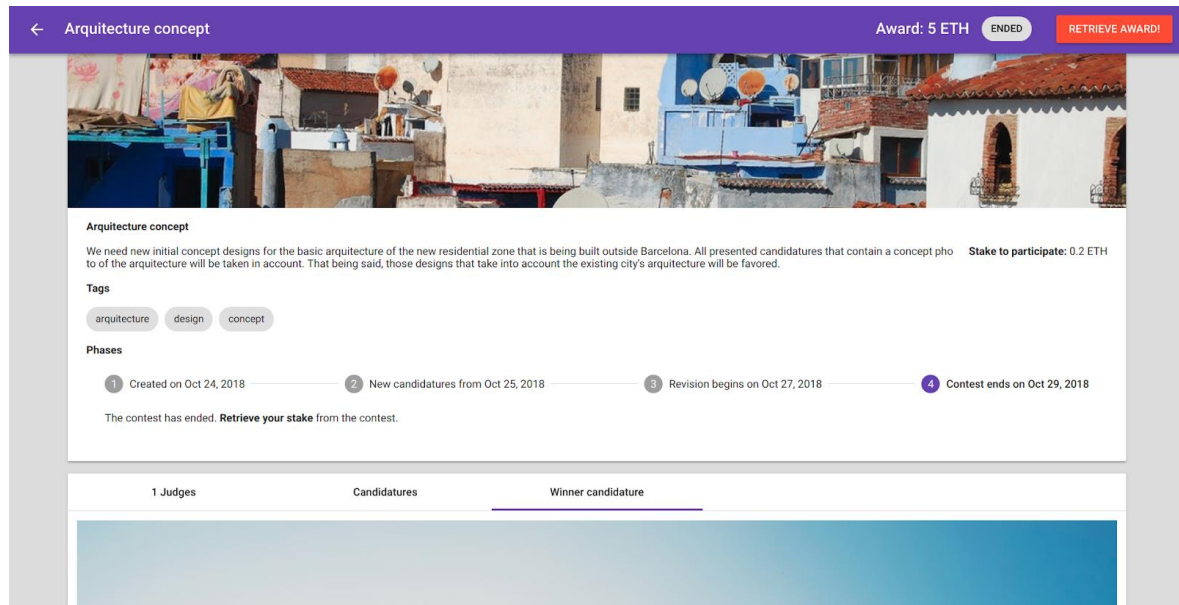


Ilustración 2-6 – Interfaz de concurso finalizado

## 3 Back-End

### 3.1 Sistema de archivos interplanetarios (IPFS)

*IPFS* es un protocolo y una red diseñados para crear un método peer-to-peer direccionable por contenido para almacenar y compartir hipermedia en un sistema de archivos distribuidos. En otras palabras, *IPFS* proporciona un modelo de almacenamiento en bloque de alto rendimiento y contenido direccionado, con hipervínculos dirigidos al contenido. Esto forma un gráfico acíclico dirigido por Merkle generalizado (DAG). *IPFS* combina una tabla hash distribuida, un intercambio de bloques incentivado y un espacio de nombres de autocertificación. Los archivos se identifican por sus valores hash, por lo que es fácil de almacenar en caché.

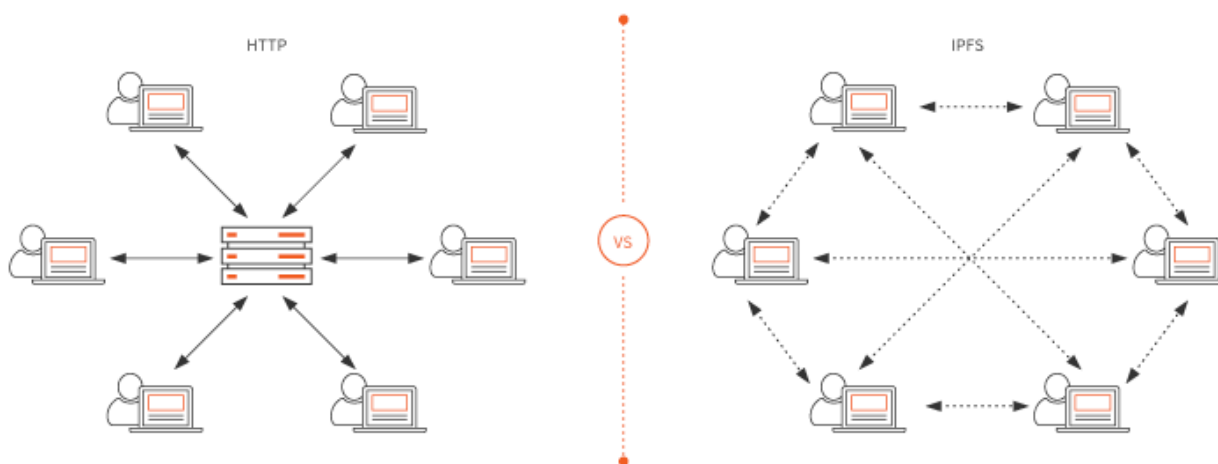


Ilustración 3-1 – Comparación IPFS vs HTTP

El propósito de uso de éste protocolo es la capacidad de descentralización y de veracidad. Mediante el uso del protocolo IPFS se van a almacenar los objetos a ser elegidos en el proceso electivo, ya sea una fotografía, un documento redactando un proyecto o idea, etc... Dicho documento/s vendrán identificados con un hash único con el cual puede ser identificado por cualquier miembro para comprobar el contenido.

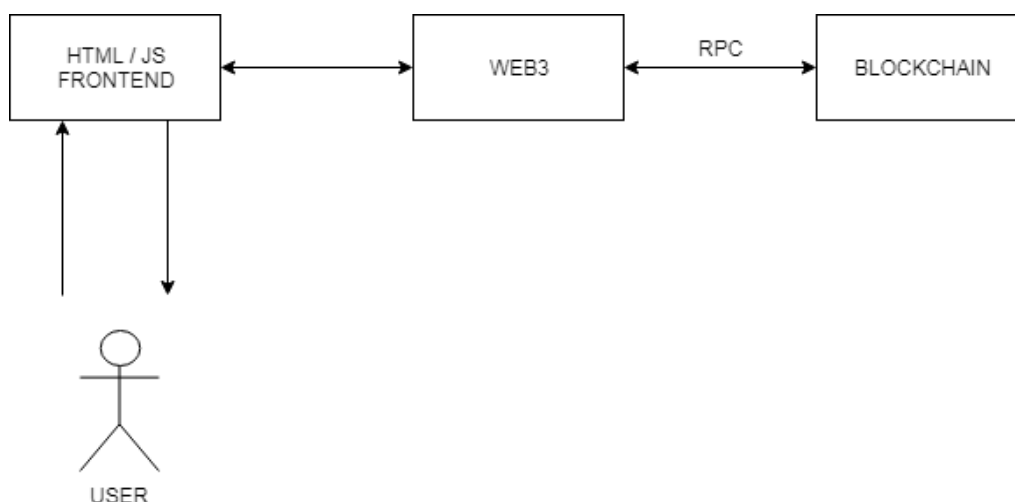


## 3.2 Contrato Inteligente

### 3.2.1 Interacción con el Frontend

Todo lo que hacemos en la blockchain de Ethereum ocurre a nivel de *geth*, que es una interfaz de nivel de línea de comandos en la consola *geth*. No esperamos que un usuario final escriba comandos en una terminal para cada pequeña cosa que desee realizar en ethereum. Por lo tanto, es necesario que haya una forma de interacción con la interfaz de línea de comandos utilizando los códigos que se ejecutan en los navegadores. La forma de asegurarnos de que con un simple click del usuario se interprete como un comando de terminal *geth* es usando la librería *web3.js*.

*Web3.js* es una colección de bibliotecas que nos permite interactuar con un nodo ethereum local o remoto, utilizando una conexión HTTP o IPC. Es decir, nos proporciona una API de JavaScript para comunicarse con *geth*. Utiliza JSON-RPC internamente para comunicarse con *geth*, que es un protocolo de llamada de procedimiento remoto (RPC) ligero.



### 3.2.2 Filosofía básica de contrato

El contrato inteligente creado para *cryptocontest*, establece las funciones básicas para la creación de concursos o métodos electivos. Para ello el creador del concurso establece el concurso basándose en unos criterios y normas iniciales. Los jueces serán las personas u otros contratos inteligentes que votarán las candidaturas preferidas. En el caso de añadir como juez otro contrato inteligente, éste dispondrá plena facultad para añadir los jueces que considere oportuno. En este caso, nuestro contrato obtendrá de este juez basado en contrato la candidatura más votada. El creador del concurso ponderará a los jueces según su experiencia, o caso de uso, quedando claramente inicialmente sus poder o capacidad de voto.

Las diferentes candidaturas podrán adherirse al concurso en tiempo y forma, realizando un pago de fee/"stake" para cada candidatura realizada. También tendrán que añadir el IPFS hash de la imagen/documento subido a la red IPFS.

Posteriormente y ya finalizado el tiempo de candidaturas, los jueces pueden votar las candidaturas de su elección. Los jueces también pueden invalidar cualquier candidatura por incumplir las normas del concurso, quedando registrado el juez que anuló la candidatura y motivo. La invalidación de la candidatura implica perder el “stake” aportado inicialmente.

Una vez finalizado el concurso, se elige la candidatura (no invalidada) con más votaciones. A partir de ese momento todas las candidaturas no invalidadas pueden recoger el “stake” aportado. En el caso del ganador, a parte del “stake” obtendrá el premio.

### 3.2.3 Métodos de protección

El contrato está fuertemente ligado a periodos de fechas, por lo que se han utilizado múltiples *#require* para proteger dichos rangos de fechas.

De igual forma, se ha utilizado la librería SafeMath desarrollada por OpenZeppelin que se utiliza para escribir contratos inteligentes seguros en Solidity utilizando patrones de seguridad de contratos comunes, para asegurarnos de que no hay ninguna posible falla de cálculo en la utilización de operadores matemáticos.

Para garantizar unos concursos seguros, hemos analizado los riesgos y posibles ataques a la plataforma que proponemos, y las soluciones que hemos encontrado son las siguiente:

- **Manipulación de votaciones públicas**

Descripción: en un concurso público con votación popular, en que todas las direcciones pudieran votar, se podría manipular muy fácilmente el ganador del concurso. Esto es porque cualquier participante podría crear múltiples direcciones anónimas y votar a su propia candidatura, y técnicamente no podemos diferenciar si se trata de la misma persona o no.

Solución: todos los concursos deben tener unos jueces determinados. Para permitir votaciones con un gran número de posibles votantes, se deberá crear un concurso en el que el juez sea una dirección de smart-contract, que sea el responsable de gestionar las identidades de los votantes.

En cuanto se solucione el problema de la identidad en la blockchain y en Ethereum, se debería poder implementar de forma segura votaciones abiertas y populares.

- **Ataque de copia**

Descripción: un agente maligno podría presentar una copia de una candidatura ya presentada, así optando a ganar el concurso con el trabajo de otra persona.

Solución: no se puede presentar una candidatura cuyo hash ya esté inscrito en el concurso anteriormente.



- **Ataque de modificación**

Descripción: similar al anterior, un agente maligno podría escanear todos los concursos para ver las nuevas candidaturas presentadas, modificar una parte muy pequeña del contenido de la candidatura y presentarla en su nombre. Así, podría optar a ganar los concursos con el trabajo de los otros. En su versión extrema, se podría escribir un script que automatice las copias y maximizar el beneficio.

Solución: en la fase de presentación de candidaturas, no se sube el contenido de la candidatura, sino solamente su hash. El contenido sólo se sube en la fase siguiente, cuando ya no se pueden presentar nuevas candidaturas.

Riesgos: para evitar este ataque se obliga a los participantes a hacer dos pasos para la presentación de las candidaturas. Por otro lado, si se presenta una candidatura ya utilizada para otro concurso, ya será visible para otros agentes. En ese caso, los ataques de modificación serían posibles, y sería la responsabilidad de los jueces cancelar la falsificación.

- **Ataque del “creador arrepentido”**

Descripción: en el caso que el creador de un concurso sea asimismo un juez, se puede dar la siguiente posibilidad: puede presentar su propia candidatura, y si no le satisfacen las otras candidaturas presentadas, puede votar a su propia, y reembolsarse el importe del concurso. Además, así puede obtener todas las candidaturas y beneficiarse de ellas.

Solución: desde el punto de vista técnico, no podemos saber si detrás de la dirección de un juez hay la misma persona creadora del concurso. Lo mejor que podemos hacer es recomendar a los participantes que participen en concursos que tengan los jueces bien identificados en plataformas externas, o que no tengan un juez único. Cuando el problema de la identificación en la blockchain esté resuelto, se podrá encontrar una solución mucho mejor.

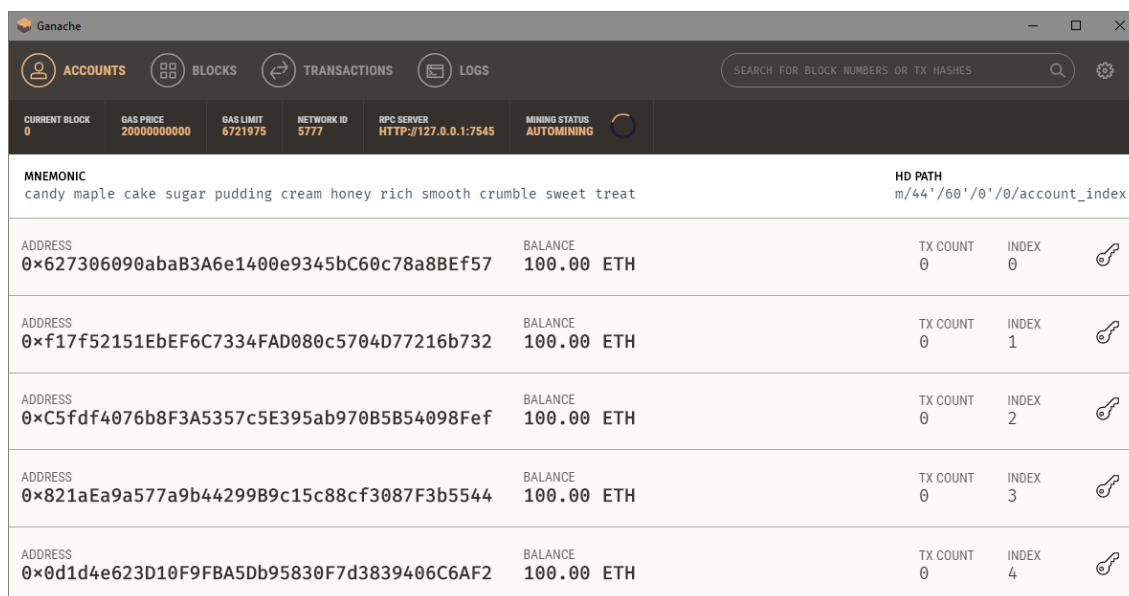
## 4 Integración, pruebas y resultados

### 4.1 Herramientas y Utilidades

#### 4.1.1 Ganache

Ganache nos permitirá ejecutar una blockchain en local, donde podremos desplegar nuestros contratos inteligentes, ejecutar tests e inspeccionar de un modo visual el estado de la blockchain mientras hacemos nuestras operaciones de desarrollo.

Existe versión de consola de comandos y otra en modo versión gráfica.



Ganache					
ACCOUNTS		BLOCKS	TRANSACTIONS	LOGS	SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK 0	GAS PRICE 20000000000	GAS LIMIT 6721975	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING
<b>MNEMONIC</b> candy maple cake sugar pudding cream honey rich smooth crumble sweet treat					<b>HD PATH</b> m/44'/60'/0'/0/account_index
ADDRESS 0x627306090abaB3A6e1400e9345bC60c78a8BEf57	BALANCE 100.00 ETH	TX COUNT 0	INDEX 0	🔑	
ADDRESS 0xf17f52151EbEF6C7334FAD080c5704D77216b732	BALANCE 100.00 ETH	TX COUNT 0	INDEX 1	🔑	
ADDRESS 0xC5fdf4076b8F3A5357c5E395ab970B5B54098Fef	BALANCE 100.00 ETH	TX COUNT 0	INDEX 2	🔑	
ADDRESS 0x821aEa9a577a9b44299B9c15c88cf3087F3b5544	BALANCE 100.00 ETH	TX COUNT 0	INDEX 3	🔑	
ADDRESS 0x0d1d4e623D10F9FBA5Db95830F7d3839406C6AF2	BALANCE 100.00 ETH	TX COUNT 0	INDEX 4	🔑	

Ilustración 4-1 – Ganache UI

Para nuestro desarrollo se utilizará la versión en modo consola y se especificarán inicialmente varias cuentas de direcciones fijas para facilitar el testing.

#### 4.1.2 Truffle

Truffle es un framework de desarrollo que nos permitirá compilar, desplegar y/o testear nuestros contratos inteligentes. Está desarrollado en Node.js y lo usaremos mediante línea de comandos. Truffle hace un wrapper sobre el framework Javascript de desarrollo de test Mocha.

### 4.1.3 Metamask

Es una herramienta que nos permitirá interactuar con nuestros contratos inteligentes desde el navegador web. Para su instalación tan solo se necesita buscarla como extensión para navegadores como Mozilla Firefox o Google Chrome.

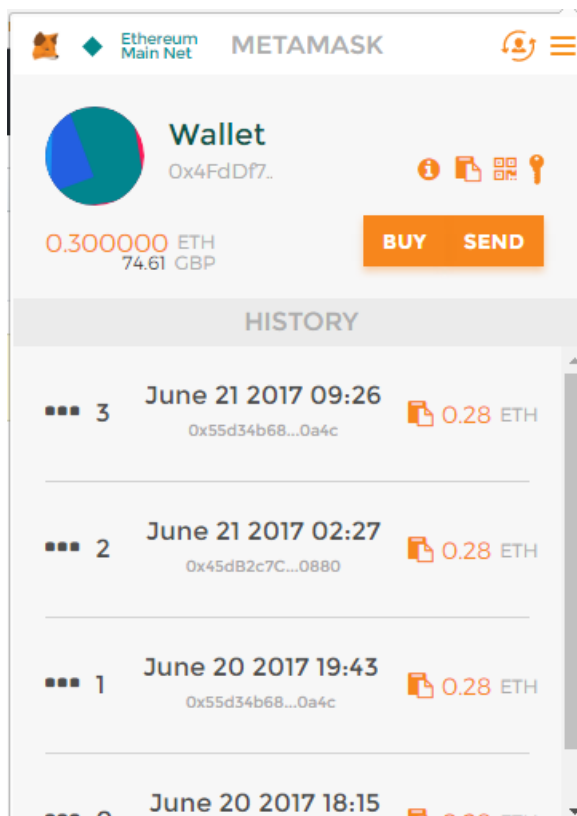


Ilustración 4-2 – Metamask UI

## 4.2 Testing

### 4.2.1 Tests utilizados

El objetivo de los test realizados es asegurar la robustez del smart-contract y todas sus funciones, tanto que funcionen como deben como para capturar los posibles errores que debería encontrar un usuario, ya que sabemos que es un código muy vulnerable y los usuarios aprovechan cualquier error del código en su beneficio.

Para realizar los test hemos utilizado la red Ganache personalizada con 30 cuentas para tener más diversidad y ejecutar más test. Además, hemos generado un Json con valores predeterminados para utilizarlos sobre los concursos, jueces y candidaturas.

```
^Cjordi@jordi-VirtualBox:~/cryptocontests/scripts$ ./run_ganache.sh
Ganache CLI v6.1.0 (ganache-core: 2.1.0)
```

Available Accounts

=====

```
(0) 0x8a838b1722750ba185f189092833791adb98955f
(1) 0x5f61a7da478e982bbd147201380c089e34543ab4
(2) 0x790d74d4a69a526b6c30401eca367dad499455dd
(3) 0x4522a965f4c12683a8ccaf2f43446a5e97c0364a
(4) 0x60971b70bef61aab41af496ed35e2e23ec8ff9ba
(5) 0x6b67d256bb78de2b16f5d6d439bd017b76891007
(6) 0xd5ed214251281875914e3632d9982839be804fba
(7) 0xae5f8cb3d2ed1bfb0f1b99ef5516c4569ef6e4a0
(8) 0x1586d0fe9f693f2182ab1963aa8f1268918a0330
(9) 0x8634bf5a0cb6de34f0c12e28d7f61f586c7cdb64
(10) 0xbd8a88bdac32f123008db768fee34211130e9cbb
(11) 0x0e9d2c00cd736e27e39c394870c78e5dd430a771
(12) 0xd1d1bc94567bceb24cb4ba06d9a8c364a20881da
(13) 0x128a906fbfd8497f0adc03d523d656ef16dfdf6
(14) 0x6dcdd8dd7377e4dfcd03527c26adab74ada6613c
(15) 0xaba60c4c7f47c4a765c27c218610a3e530d5d038
(16) 0x2568599d312bbc200bfdaa76652b80da51161523
(17) 0x10ff6d54239b2aea4c9941882e1e93bdf9fcc6a2
(18) 0x2f381dc91b5b2c1c0be4a6a2861144ce85f86ace
(19) 0xed80dc78f6291975ee484b5407fcc0d2b7ca506f
(20) 0x9dfe0b99a25218aa349bfd5e0fde7af0801a1ec2
(21) 0xe3106ed542b9f25a1d79d3c8e98982df4dea01eb
(22) 0xe56e0ff39b6bd125f03f1931414aef9f6133c18f
(23) 0xcaaebd1734a96dedb213d17b4e59c89e70e7a004
(24) 0xb36cf830cbb23aa89ab5e8d1d992ae33fafc45f3
(25) 0x102cd757b8460cd10ce6a9977ca491e00983cba3
(26) 0x231b5518d54ab5478b4d23a1b0f6448804c48368
(27) 0xa3f9a124f249bb7231ffaeeb7ea096f88a9afc4d
(28) 0x20b85fefe92f9829f8524c42b489d214b8a5b340
(29) 0xc0ec23f5e23d6014e35424267a99dcabd0816ce5
(30) 0xbf450fdf0af3118ae5dd94c79f52f296d2855024
(31) 0xa6af4e8a7561188ad87d8895cac3e9dfa83085f2
(32) 0x335a941ac5f1cb73378865b40f7c1a180eec5b22
(33) 0x353839f39e01f2526381dcef95e424f7467de49a
(34) 0xa7f6d3f615b7c146753f6683047ee978754bf76a
(35) 0x7e647f2469188d1fd41ea960b643185fbd936346
(36) 0x28ee7fcc86e1d371892349f9fdb86a2ae10e13ec
(37) 0x52b93a06df9884db1672439c355230b6c295d776
(38) 0x9449af28c0a43acd7782ceddd7406b28bdb1de87
(39) 0x75c31c112d82bc8ea9885f8570ad04f2728410ed
```

Private Keys

Ilustración 4-3 – Ganache Accounts

Para ejecutar los test disponemos de dos scripts:

- *run\_ganache.sh*: El cual nos va a permitir ejecutar la red ganache con las cuentas creadas, sobre la que ejecutaremos los test.
- *truffle\_test.sh*: Este script arranca el test utilizando la red ganache.

El fichero json que contiene la información que se utilizará:

- *contestBulkData.json*

La lógica de los test:

- *CryptoContest.test.js*: El cual se ejecutará una vez iniciemos el script *truffle\_test.sh*

## 4.2.2 Resultados

Los test comprobarán que todas las funciones emiten los resultados esperados, ya sea que dan un buen resultado y la transacción se ejecuta correctamente o captura algún tipo de error. Los podemos dividir en cinco apartados, los test de los concursos, de los jueces, de las candidaturas, de las votaciones y de resolución y cobro del premio. A continuación, mostramos los resultados y la descripción de los test realizados.

```
Contract: ContestController
Contest functions without errors:
  ✓ Should set a new contest (5249ms)
  ✓ Should get all contests by hash (954ms)
  ✓ Should get total contest count
  ✓ Should get all tags (60ms)
Contest functions catching errors:
  ✓ Should not set a new contest because award is 0 (971ms)
  ✓ Should not set a new contest because missing parameters (750ms)
  ✓ Should not get a contest by hash because contest hash is wrong (341ms)
Judge functions without errors:
  ✓ Should add a new judge member (1864ms)
  ✓ Should remove a judge member from contest (1295ms)
  ✓ Should get judge details (430ms)
  ✓ Should get judge by contest (408ms)
Judge functions catching errors:
  ✓ Should not add a new judge member because out of time (1580ms)
  ✓ Should not add a new judge member because missing parameters (627ms)
  ✓ Should not add a new judge member because contest hash is wrong (1723ms)
  ✓ Should not add a new judge member because the JudgeAddress already exists in the contest (1880ms)
  ✓ Should not remove a judge member from contest because missing parameters (1601ms)
  ✓ Should not remove a judge member from contest because the contest hash is wrong (2374ms)
  ✓ Should not remove a judge member from contest because the judge has already been eliminated (1979ms)
  ✓ Should not remove a judge member from contest because out of time (1708ms)
  ✓ Should not get a judge details because missing parameters
  ✓ Should not get a judge details because does not exists (891ms)
  ✓ Should not get a judge details because contest hash is wrong (511ms)
  ✓ Should not get judge by contest because missing parameters
  ✓ Should not get judge by contest because contest hash is wrong (453ms)
```

Ilustración 4-4 – Truffle Test 1

```

Candidatures functions without errors:
✓ Should add 10 new candidature for each contest (19398ms)
✓ Should get a candidature by hash contest and hash candidature (6284ms)
✓ Should get the candidatures by contest (523ms)
✓ Should get the total candidatures by contest (304ms)
✓ Should get own candidatures by contest and candidate (3807ms)
✓ Should cancel a candidature (1110ms)
Candidatures functions catching errors:
✓ Should not add 10 new candidature for each contest because missing (5532ms)
✓ Should not add 10 new candidature for each contest because the stack is over required (5202ms)
✓ Should not add 10 new candidature for each contest because the caller is wrong (6993ms)
✓ Should not add 10 new candidature for each contest because the candidature already exists (6613ms)
✓ Should not add 10 new candidature for each contest because the contest hash is wrong (4867ms)
✓ Should not add 10 new candidature for each contest because the time is over (10454ms)
✓ Should not get a candidature by hash contest and hash candidature because missing parameters
✓ Should not get a candidature by hash contest and hash candidature because the contest hash is wrong (3063ms)
✓ Should not get the total candidatures by contest because the contest hash is wrong (245ms)
✓ Should not get the total candidatures by contest because missing parameters
✓ Should not cancel a candidature because the time is over (575ms)
✓ Should not cancel a candidature because the judge account is wrong (1195ms)
✓ Should not cancel a candidature because missing parameters
✓ Should not cancel a candidature because the contest hash is wrong (435ms)
✓ Should not get own candidatures by contest and candidate because missing parameters (498ms)
✓ Should not get own candidatures by contest and candidate because caller doesn't have candidature (357ms)
✓ Should not get own candidatures by contest and candidate because the contest hash is wrong (293ms)
Votation functions without errors:
✓ Should add a new vote (1564ms)
Votation functions catching errors:
✓ Should not add a new vote because missing parameters (520ms)
✓ Should not add a new vote because the contest hash is wrong (1012ms)
✓ Should not add a new vote because the judge does not exists (980ms)
✓ Should not add a new vote because the judge has already voted (1105ms)
Solve Contest and Refund functions without errors:
✓ Should solve all contests (1525ms)
✓ Should get winner of all contests (1083ms)
✓ Should get winner of all contests (836ms)
✓ Should refund award to winner of all contests (1360ms)
✓ Should refund stake to candidates of all contests (1674ms)
Solve Contest and Refund functions catching errors:
✓ Should not solve all contests because is not time to do (1290ms)
✓ Should not solve all contests because missing parameters (539ms)
✓ Should not solve all contests because the contest hash is wrong (1148ms)
✓ Should not solve all contests because the contest has already solved (1054ms)
✓ Should not get winner of all contests because missing parameters (473ms)
✓ Should not get winner of all contests because the contest hash is wrong (757ms)
✓ Should not get winner of all contests because missing parameters (704ms)
✓ Should not get winner of all contests because the contest hash is wrong (900ms)
✓ Should not refund to candidates of all contests because missing parameters (994ms)
✓ Should not refund to candidates of all contests because the contest hash is wrong (971ms)
✓ Should not refund stake to candidate of all contests because the stake has already refunded (1121ms)
✓ Should not refund award to winner of all contests because the award has already refunded (2414ms)
✓ Should not refund to candidates of all contests because is not time to do (1336ms)
70 passing (2m)

```

Ilustración 4-5 – Truffle Test 2

## 5 Conclusiones y trabajo futuro

---

### 5.1 Conclusiones

Los resultados obtenidos nos permiten concluir desde nuestra perspectiva personal, que, por lo general el desarrollo actual en tecnologías blockchain y en concreto con Ethereum, aún se encuentra en sus primeros pasos de evolución ya que los actuales entornos de desarrollo, frameworks, y lenguajes de desarrollo no cuentan con las facilidades de desarrollo con la que cuentan otros lenguajes que llevan mayor tiempo en el mercado. No obstante, comprendemos que son las primeras etapas, y es un daño colateral que hay que sufrir hasta que la adopción de esta tecnología vaya incrementando y sea mayormente adoptada por mas desarrolladores.

### 5.2 Posibles integraciones con proyectos externos

Una de las funciones más útiles que tiene el sistema resultante es la posibilidad de integración con otros proyectos basados en smart-contracts de Ethereum. Desafortunadamente, no hemos tenido tiempo de implementar ejemplos de este tipo de integraciones. Como comentamos en el último punto, esto supone una de las principales posibilidades de extensión del sistema.

Estos son algunos de los proyectos con los que tendría sentido poder integrar nuestro sistema en un futuro:

- *Aragon*: Proyecto que pretende desintermediar la creación y el mantenimiento de las estructuras organizativas mediante el uso de la tecnología blockchain.
- *Augur*: Plataforma de mercado de predicción y oráculo descentralizado construido sobre blockchain.
- *Gnosis*: Construye nuevos mecanismos de mercado para permitir la distribución de recursos, desde activos a incentivos, e información a ideas.
- *Daostack*: Herramienta para inteligencia colectiva para que se puedan tomar decisiones rápidas e innovadoras a escala.

### 5.3 Trabajo futuro

En base a las mejoras a considerar se han barajado las diferentes opciones que podrían mejorar el sistema:

- **Generalizar el tipo de concurso:** De momento el sistema solo soporta concursos de fotografías, como hemos explicado anteriormente. Para que se pudieran hacer concursos de cualquier tipo se tendrían que hacer algunas modificaciones, pero solo en la parte de “frontend”. El smart contract no necesita cambios ya solo guarda el hash del contenido de la candidatura subida a IPFS, y ese contenido podría tener cualquier formato.



- **Opciones de concurso:** Se podrían añadir diversas opciones de configuración para los concursos, así se podrían ajustar más a cada caso. Ejemplos serían: opción de primer, segundo y tercer premio; diferentes tipos de concurso (fotografía, libros, planos de obra...); etc.
- **Unidad de criptodivisa:** Actualmente se está utilizando como única medida de pago el Ether. Dada la cotización actual del ether, sería buena mejora dejar al usuario seleccionar entre ETHER y WEI ( $10^{-18}$ ).
- **Voto Cuadrático:** Una mejora a implementar podría ser implementar el voto cuadrático obteniendo que en lugar de que cada juez pueda votar a un único candidato, pueda asignar votos a diferentes candidatos teniendo en cuenta que cada voto le costará el cuadrado de "Voice Credit". El "Voice Credit" es una cantidad igual asignada inicialmente a todos los jueces.

Number of votes	"Voice Credit" cost
1	1
2	4
3	9
4	16
5	25

Tabla 5-1 – Voto Cuadrático

- **Mejorar la escalabilidad del smart-contract:** Al finalizar el proyecto, este está preparado para gestionar un número relativamente pequeño de concursos (del orden de centenares o miles). Para posibilitar un número mucho mayor de concursos al mismo tiempo, se deberían hacer cambios sobretodo en el aspecto de la comunicación entre el "front-end" y el smart contract, cómo introducir paginación, posibilitar el filtro de concursos a través del smart contract, etc.
- **Sistema de notificaciones:** dado que muchas acciones dentro del sistema de concursos se tienen que ejecutar en una determinada fase pasado un tiempo, se podría utilizar el sistema de eventos de Ethereum para notificar al front-end de que el usuario tiene que hacer una acción específica.



## Referencias

---

- [1] "A peer-to-peer hypermedia protocol to make the web faster, safer, and more open."  
<https://ipfs.io>
- [2] "TypeScript-based open-source front-end web application platform led". <https://angular.io>
- [3] "Language characterizes as dynamic, weakly typed, prototype-based and multi-paradigm."  
<https://wikipedia.org/wiki/JavaScript>
- [4] "Árbol de Merkle". [https://es.wikipedia.org/wiki/Árbol\\_de\\_Merkle](https://es.wikipedia.org/wiki/Árbol_de_Merkle)
- [5] "Ganache", <https://truffleframework.com/docs/ganache/overview>
- [6] "Truffle", <https://truffleframework.com/docs/truffle/overview>
- [7] "Metamask", <https://metamask.io/>
- [8] "Quadratic Voting", [https://en.wikipedia.org/wiki/Quadratic\\_voting](https://en.wikipedia.org/wiki/Quadratic_voting)
- [9] "web3.js – Ethereum JavaScript API", <https://web3js.readthedocs.io/en/1.0/>
- [10] "Official Go implementation of the Ethereum protocol", <https://geth.ethereum.org>
- [11] "OpenZeppelin", <https://openzeppelin.org>
- [12] "Framework Angular", <https://angular.io>
- [13] "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation 6 October 2000 <http://www.w3.org/TR/REC-xml>
- [14] "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation 6 October 2000 <http://www.w3.org/TR/REC-xml>
- [15] "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation 6 October 2000 <http://www.w3.org/TR/REC-xml>

## Glosario

---

API	Application Programming Interface
IPFS	IntePlanetary File System
IPC	Inter.process communication
RPC	Remote Procedure Call
JSON	JavaScript Object Notation
ABI	Application Binary Interface
SPOF	Single Point of Failure
URL	Uniform Resource Locator
UI/UX	User Interface / User Experience