Universidad ORT Uruguay

Facultad de Ingeniería

Tecnologías Blockchain para contratos inteligentes

Obligatorio

Marcel Cohen – 212426

Alan Monjardín – 214105

Federico Palermo – 212208

Docentes: David Rafael Giménez
Daniel Mariano

Código disponible en:
https://github.com/mcohen97/Blockchain-Inheritance-Management

Julio, 2020

Índice

Introducción

El trabajo consistió de la construcción de una plataforma basada en la tecnología Blockchain que permitiera gestionar de forma descentralizada la administración de herencias familiares valoradas en Ethers.

La plataforma se implementó con una aplicación construida en NodeJS, la cual despliega e interactúa con contratos inteligentes escritos en Solidity, los cuales se encuentran en la Blockchain de Ethereum. En esta versión inicial, los contratos no se despliegan en la red principal (Main Net) sino en una de las redes de prueba: Ropsten.

Cuentas y contratos inteligentes

El equipo acordó una Seed Phrase para generar cuentas de ethereum, la idea detrás de esta decisión fue el poder utilizar las mismas pruebas del sistema, trabajando en distintas computadoras y obtener los mismos resultados. Se utilizó la herramienta Ganache para simular una Blockchain local, y una colección de pruebas Postman para probar las funcionalidades, la herramienta Postman sincronizaba la colección entre el equipo. Entonces, con mismas pruebas, mismas Blockchains locales, y utilizando una herramienta de control de versiones (Git), se logró un trabajo fluido y totalmente remoto. La aplicación cuenta con un archivo .env donde se configura, mediante variables de entorno, cual Blockchain utilizar (la local que provee Ganache o la Testnet de Ropsten, accediendo a través de Infura).

La Seed Phrase fue la siguiente:

```
wolf cake game reason oyster clutch artefact prevent poverty flat stumble mansion
```

Las cuentas generadas por la Seed Phrase fueron las siguientes:

- `0x8c9a1E92F4101f4f5ab3A58b0963b1040e425b6F` (dueño del contrato)
- `0xF35b3ba2DE9dB8eEBcD5d128B53735e493c0BA64`
- `0x4A643DddC95aFa8A0f6E7439d9dc760aa84CdEB1`
- `0x72a5D1bd8c6e5Feb7399EacE1bcDC6cEcdD5a79C`
- `0x45F44Dec152224F336432568c8ACA7c9B98baFA8`
- `0xcC09F2d0d887f6098432E7C8BEc51b0B5aC205F9`
- `0xFEdae43693A435fb940a9881CCcD7f3Ff328c08F`
- `0xAe372f8F9697dc568AF1aD274351e56f5D467ceE`
- `0x5E6ecDA6875b4Dc8e8Ea6CC3De4b4E3c73453c0a`
- `0x1DfC1E4a154df46C0d9F0Bd35D8E9350A8187e28`

Como se especificó en la letra del obligatorio, la solución debe estar desplegada en la red de prueba de ropsten[1], las direcciones de los contratos en la red son los siguientes:

- `0x62E550A2321fBc87F227201f4336E7748774b8f1` (Testament)
- `0x62E550A2321fBc87F227201f4336E7748774b8f1` (Laws)

[1] Pueden visualizarse mejor en Etherscan.
Testament: https://ropsten.etherscan.io/address/0x62E550A2321fBc87F227201f4336E7748774b8f1
Laws: https://ropsten.etherscan.io/address/0x3edE31d488B41FceEee49Ee7feE927413B16C0c5

Decisiones de diseño

Estructura de contratos

Además de contratos para representar los testamentos (*Testament*) y la ley (*Laws*), se creó un tercer contrato: *DataStructures*, el cual contiene todas las estructuras que el equipo diseñó para capturar las distintas entidades del negocio. Se consideró que era una forma de mantener la información bien organizada, haciendo el código más fácil de mantener entre los distintos miembros del equipo. Este contrato es importado por Testament, quien utiliza las estructuras de datos, y esta dependencia se compila junto con Testament, por lo tanto ambos están contenidos en un mismo *bytecode* de contrato desplegado.

El contrato de la Ley, por otro lado, es desplegado por separado, y *Testament* realiza llamadas a este para consultar distintas reglas legales. *Testament* importa el *abi* del contrato de leyes para poder referenciarlo de forma segura.

Modelado de herederos y encargados

Para modelar a los encargados y herederos en el contrato, se utilizaron *structs*, definidos en el contrato *DataStructures*.

Las estructuras de encargados contienen los siguientes datos de estos:

- La dirección de su cuenta ethereum.
- La deuda que mantiene con el contrato, en cuanto a retiros.
- La fecha del primer retiro realizado, de los que están pendientes de pagar.
- Un flag que indica si anunció la muerte del dueño del contrato o no.

Las estructuras de los herederos contienen los siguientes datos:

- La dirección de su cuenta ethereum.
- Porcentaje de la herencia que le corresponde.
- Si falleció o no.
- Si tiene un hijo menor de edad anunciado.

El contrato mantiene un *array* de cada estructura. La cantidad de cada uno se obtiene a través del largo, y consultando por posiciones en el *array* se obtienen los distintos herederos/encargados. Se consideraron otras opciones, como tener *mappings* en lugar de *arrays*. Si bien estos dan la ventaja de hacer las consultas más rápidas, por ejemplo, obtener un encargado o heredero por su dirección en *O(1)*, se pierde la posibilidad de saber cuales son todos los datos almacenados, incluso teniendo un *mapping* de prioridad a heredero tiene casi la misma complejidad de cambiar prioridades que un *array*, por lo tanto se descartó. Otra opción era tener un combinado de *mappings* con *arrays*, donde los *mappings* tuvieran como key una dirección y cómo value una posición en un *array*. Con esa combinación se

lograrían mejores tiempos, pero agrega más complejidad. Se consideró la mejor opción utilizar *arrays* en esta versión.

Sin embargo, se incluyó un *mapping* de *address* a *int* (`users`) que almacena los addresses de usuarios involucrados y el rol que tienen en el contrato: (1 - dueño, 2 - encargado, 3 - heredero). Esto permite verificar que los encargados, beneficiarios y el dueño sean disjuntos, y también comprobar rápidamente que el address es efectivamente de un encargado, heredero o etc.

Modelado de la organización

La organización que mantiene los contratos es modelada dentro del contrato como una dirección pagable. Y existe un modificador que controla que las funcionalidades que sólo son accesibles a miembros de la organización (como el reclamo de la herencia pasados los 36 meses) sean realizadas por esta misma dirección.

Retiros

Como se mencionó antes, cada encargado tiene una variable que almacena su deuda y desde cuando la debe. Existe una funcionalidad de retiro, la cual es solo accesible a los encargados no suspendidos, para retirar dinero del testamento. Dicha función realiza las validaciones necesarias, hace la transferencia al encargado, y genera un evento de retiro y guarda en un *array* una *struct* que representa el retiro, conforme al requerimiento 23. Los eventos de retiro están indexados por dirección del que retira.

Manejo del tiempo y transcurso

Dentro del contrato se manejaron *timestamps* en formato *unix-time*, los cuales se representan como `uint256` (Con excepción de la fecha de nacimiento del dueño, la cual es `int256`, ya que puede ser negativa). Se tienen funciones que calculan la diferencia entre un *timestamp* y otro en segundos, esa diferencia es dividida entre variables que representan el largo de un día o un mes en segundos, y de esa forma se puede obtener la diferencia en meses o días entre 2 *timestamps*. Dichas variables pueden ser modificadas, para reducir el largo de los días o meses a minutos o segundos. Para saber cual es la fecha actual, se utilizó la primitiva `now` de Solidity, la cual devuelve el tiempo de minería del bloque actual. El contrato guarda un *timestamp* que representa la última señal de vida del dueño, o como le ha llamado el equipo a esta función: *Heartbeat*.

Multas y suspensiones a encargados

Existe un *modifiers* para controlar que sean encargados los que acceden a una función, y que a su vez no estén suspendidos. Se controla que el encargado no esté suspendido calculando la diferencia en meses entre hace cuanto tiempo tiene deuda y el tiempo del bloque (`now`), si ese tiempo supera los 3 meses (90 días) se corta la transacción y se devuelve un error indicando que el encargado está suspendido. Este modificador tiene una

limitante, ya que el tiempo del bloque puede ser el mismo por cierto tiempo, dependiendo de la blockchain, (véase la sección Limitantes). Cuando el encargado va a pagar su deuda, si está suspendido, debe pagar la cantidad que retiró más la multa, que se calcula utilizando la diferencia en días desde que se atrasó (es decir, la cantidad de días desde el préstamo menos 90). Del contrato de leyes desplegado se obtiene el porcentaje sobre lo retirado y el maximo de dias. Si el encargado que hace el pago no está suspendido, simplemente se le resta a su deuda. Cabe destacar que se trata de una función pagable.

Ajuste de porcentajes

Se diseñó un algoritmo para ajuste de porcentajes, el cual es reutilizado por distintas funciones, como agregar y cambiar porcentaje de un heredero. El algoritmo consiste en dejar fijo uno de los porcentajes, y ajustar el resto para que sumen 100, esto se hace calculando la suma de todos los porcentajes y restándole 100, de ahí se obtiene la diferencia a ajustar, que puede ser positiva o negativa. Esa diferencia se divide entre la cantidad de los otros porcentajes que se tienen que ajustar (que no quedan fijos), dicha división se suma/resta a todos los otros porcentajes, a veces existe un resto que no se puede dividir, ya que Solidity trabaja sólo con enteros, ese resto, que es menor a la cantidad de porcentajes se distribuye entre los de mayor/menor prioridad, dependiendo del signo. Cuando se remueve un heredero, se le suma su porcentaje al siguiente conforme al requerimiento 9, cuando se declara muerto, simplemente se le asigna porcentaje 0, y se pone su flag de muerto como true.

Separación de conceptos de herencia y fondos del testamento

Tras un análisis del problema, el equipo observó que los cálculos que se realizan en función de la herencia del contrato, no pueden utilizar el balance de este (`address(this).balance`), ya que ese balance puede ser mucho menos de lo que realmente existe de herencia, dado que los encargados pueden hacer préstamos del dinero del contrato. Por lo tanto se decidió tener una variable que mantenga la cantidad total de la herencia, que puede ser más que lo que contiene el balance del contrato.

A continuación se especifica cómo se implementó cada requerimiento y regla de negocio. Luego de cada explicación se incluyen, si corresponden, los *endpoints* de la aplicación para poder probar el requerimiento o regla de negocio en cuestión. Todos los *endpoints* se encuentran en la colección de Postman incluída en la entrega. La colección incluye endpoints adicionales de obtención de datos que pueden no estar documentados a continuación dado que no eran necesarios para cumplir con lo solicitado, pero se utilizaron para verificar el correcto funcionamiento de la solución.

Requerimientos

1. *Manejar la gestión de contratos de herencia, adjudicando propiedad sobre una cantidad de Ethers a la auditoría de un contrato inteligente.*

   La aplicación implementada provee una API, construida en *express js*, que permite realizar la gestión de contratos de herencia, incluyendo la gestión de los herederos, encargados, los préstamos, y la herencia en sí misma. Al momento de desplegar los contratos se les transfiere el valor correspondiente a la herencia y desde ese momento queda bajo la administración del contrato.

2. *De la persona que contrata el servicio para administrar su herencia, quien será el dueño del contrato, se debe almacenar sus datos de identificación: Nombre completo, Documento de identidad, Fecha de nacimiento, Domicilio constituido, Teléfono, y mail de contacto, Fecha de contratación*

   Se decidió no incluir los datos de quien contrata el servicio en el constructor del contrato dado que el constructor de un contrato en Solidity está limitado a 16 parámetros. Para evitar mantener el contrato en un estado inconsistente, se decidió implementar una función `addOwnerData` para agregar los datos faltantes e invocar esta función en la misma función que realiza el despliegue del contrato. Por lo tanto, quien contrata el servicio lo realiza mediante una única llamada a la API de la aplicación incluyendo sus datos personales, y luego la aplicación se encarga de desplegar el contrato y asignarle sus datos personales. Puede suceder que se realice el despliegue y no la asignación de datos por algún problema técnico o insuficiencia de fondos, pero se intentó asegurar la consistencia lo máximo posible.

   Los datos del dueño se pueden consultar con un endpoint de la API (sólo encargados del contrato):

   ```
   GET {{URL}}/api/testament/owner
   ```

3. *De cada contrato de herencia se debe conocer la siguiente información: Monto de la herencia, Cantidad de beneficiarios (herederos), Cuentas de Ethereum de cada beneficiario, Cantidad de encargados, Cuentas de Ethereum de cada encargado.*

   El monto de la herencia se envía como valor al momento de desplegar el contrato y las cuentas de los herederos y encargados se envían como parámetros del

constructor del contrato. Estas cuentas se almacenan en arrays a partir de los cuales se puede obtener la cantidad de cuentas según el largo.

4. *Cada heredero debe ser designado con el porcentaje que debe recibir de herencia.*

   Los porcentajes de herencia se envían como parámetro del constructor del contrato, el cual requiere que la cantidad de herederos y de porcentajes sea la misma. Luego se realiza la asignación de porcentajes a herederos uno a uno, guardando el porcentaje en el campo `percentage` de cada heredero.

5. *Cada heredero tendrá un orden de prelación para recibir su parte de la herencia.*

   El orden de prelación corresponde al orden en el que se almacenan los herederos. Se entendió que cada heredero tiene un único orden de prelación.

6. *El contrato debe poder recibir cualquier cantidad de herederos, pero debe existir al menos 1.*

   El constructor del contrato requiere que el largo del array de herederos sea mayor o igual a 1 y no se establece un máximo.

7. *El contrato debe poder recibir hasta 5 encargados, pero debe recibir un mínimo de 2.*

   El constructor del contrato requiere que el largo del array de encargados sea entre 2 y 5.

8. *El dueño del contrato debe poder dar alta y baja de herederos y encargados en todo momento, cumpliendo con las restricciones de cantidades mínimas.*

   La la aplicación permite para agregar o eliminar herederos y encargados, invocando funciones del contrato que agregan o remueven elementos de los arrays correspondientes.

   Para el alta de herederos se requiere que la prioridad solicitada no exceda la cantidad de herederos incluyendo el heredero nuevo, y que el nuevo heredero no se encuentre registrado con otro rol. Los porcentajes de la herencia se ajustan en base al porcentaje del nuevo heredero (con el algoritmo de ajuste de porcentajes). Para la baja de herederos se requiere que se cumpla con el mínimo de 1. El porcentaje del heredero dado de baja se asigna al heredero con orden de prelación precedente, o al de orden 2 si se trata del de orden 1.

   Para el alta de encargados se requiere que no se exceda el máximo de 5, que su adición no implique que las comisiones de los encargados excedan el 100% del balance y que el nuevo encargado no se encuentre registrado con otro rol. Para la baja de encargados se requiere que se cumpla con el mínimo de 2.

```
POST {{URL}}/api/testament/heirs
DELETE {{URL}}/api/testament/heirs/{{ADDRESS}}
POST {{URL}}/api/testament/managers
DELETE {{URL}}/api/testament/managers/{{ADDRESS}}
```

9. *En caso que un heredero no pueda recibir su parte de la herencia, esta será entregada al heredero con orden de prelación precedente, o al de orden 2 si se trata del de orden 1.*

   Se considera que un heredero no puede recibir su parte de la herencia si es señalado como fallecido, o fue removido. En tal caso, su porcentaje de la herencia es tratado como se solicita.

10. *El contrato de herencia debe poder ser cancelado por su dueño, con la devolución del monto depositado, menos la cantidad o porcentaje acordada al momento de la contratación.*
11. *Si el contrato es cancelado ya no deben poder hacerse operaciones sobre él y el saldo del mismo debe ser cero.*

    La aplicación permite cancelar un contrato de herencia invocando una función que del contrato, solo para el dueño del mismo, que lo destruye. La comisión de cancelación del contrato se envía como parámetro del constructor del mismo al momento de desplegarlo (`cancelFee`). Tal comisión puede especificarse como un valor fijo o de lo contrario se interpreta como un porcentaje debiendo ser menor o igual a 100. Antes de destruir el contrato se transfiere el valor correspondiente a la comisión a la cuenta de la organización y el saldo restante se transfiere devuelta al dueño del contrato. Cabe destacar que esta función está protegida por el modifier que controla que sea el dueño del contrato.

    ```
    DELETE {{URL}}/api/testament
    ```

12. *El dueño del contrato, debe poder cambiar en cualquier momento el orden de prelación de sus herederos y sus porcentajes.*

    La aplicación provee operaciones para modificar la prioridad y porcentajes de los herederos. Para modificar la prioridad, el contrato realiza el desplazamiento correspondiente en su array de herederos y ubica al heredero en cuestión en su nueva posición. Para modificar el porcentaje, el contrato le asigna el nuevo porcentaje al heredero en cuestión e invoca al algoritmo de ajuste de porcentajes.

    ```
    PUT {{URL}}/api/testament/heirs/{{ADDRESS}}/percentage
    PUT {{URL}}/api/testament/heirs/{{ADDRESS}}/priority
    ```

13. *Los encargados del contrato pueden consultar todos los datos del contrato, salvo el monto total de la herencia, si el dueño lo prohíbe.*

Este requerimiento se cumple mediante un conjunto de operaciones que permite la aplicación. Si se consulta la información del contrato, se obtiene la cantidad de herederos y de encargados. A partir de esto se puede consultar por un heredero o un encargado en particular según su posición en la lista del contrato, o bien consultar por el listado de herederos y el de encargados. También se permite consultar los datos del dueño del contrato.

Se provee además la posibilidad de consultar el monto total de la herencia y el monto actual descontando los préstamos. Para esto se necesita que el balance sea visible el cual por defecto está oculto. Se provee una operación exclusiva para el dueño del contrato que permite cambiar la visibilidad.

```
GET {{URL}}/api/testament/information
GET {{URL}}/api/testament/heirs/{{position}}
GET {{URL}}/api/testament/managers/{{position}}
GET {{URL}}/api/testament/owner
GET {{URL}}/api/testament/inheritance

POST {{URL}}/api/testament/inheritance/visibility
```

14. *Los encargados del contrato pueden retirar un porcentaje del valor del contrato, establecido por el dueño del contrato, comprometiéndose a devolverlo dentro de los 90 días siguientes, con la penalización, en caso de incumplimiento, de la pérdida de su condición de encargado, por lo que se le debe prohibir todo acceso a la información del contrato, así como a sus poderes como encargado. Del monto a retirar se cobrará un porcentaje por la operación, el cual es definido por las normas legales.*

    El contrato provee una función para realizar préstamos sobre el balance del mismo, solo disponible para encargados no suspendidos. Los préstamos se almacenan en un *array* del contrato. Se requiere que el monto del préstamo no exceda el `maxWithdrawalPercentage` establecido en el constructor del contrato por su dueño. Al momento de realizar un préstamo se actualiza la deuda del encargado y en caso de que no tuviera deuda previa también se actualiza la `withdrawalDate` del encargado con la directiva `now`. Esta fecha se utiliza para determinar si se cumplieron 90 días (3 meses) desde que realizó el préstamo.

    Se decidió mantener una única fecha de préstamo por encargado, que corresponde al momento en el que realiza el primer préstamo desde deuda cero y rige para todos los préstamos posteriores hasta que pague la deuda total. Supóngase un encargado sin deuda previa realiza dos préstamos en fechas distintas, su nueva deuda consistirá del monto de ambos préstamos pero la fecha que se tomará en cuenta para evaluar su estado de habilitación corresponderá a la fecha en la que realizó el primero de los dos. Si paga únicamente el monto del primer préstamo, la fecha de tal préstamo se mantendrá, no se sustituirá por la fecha en que realizó el segundo. Eventualmente podría ser suspendido si aún debe el monto del segundo préstamo y se cumplieron los 90 días desde que realizó el primero.

Para cada préstamo se determina una comisión en base al porcentaje correspondiente del contrato de leyes y se lo transfiere a la organización previo a efectuar al préstamo.
El contrato también provee una función para que cada encargado pueda pagar su deuda. Si se paga un monto inferior a la deuda, el monto pago se descuenta de la misma. Si se paga un monto superior a la misma, el monto excedente se incluye en el balance del contrato.

Véase multas y suspensiones a encargados.

```
POST {{URL}}/api/testament/withdrawals
POST {{URL}}/api/testament/withdrawals/pay
```

15. *Si un encargado fue suspendido por incumplimiento de un retiro, para recuperar su posición de encargado debe ingresar un monto igual al retirado más una multa equivalente a un porcentaje estipulado por ley sobre lo retirado, por cada día de atraso, con un máximo de días según lo estipulado por la ley.*

    Al momento de pagar una deuda, se evalúa si el encargado está suspendido y en ese caso se requiere que el monto pagado sea igual o superior a la deuda más una multa. La multa se determina según lo indicado en el requerimiento, haciendo uso del contrato de leyes.

16. *El contrato puede ser exigido por cualquiera de los beneficiarios o los encargados en todo momento, pero para que se valide su ejecución, debe cumplirse alguna de las siguientes condiciones:*
    - *Que el dueño no haya enviado señales de vida al contrato en los últimos 6 meses*
    - *Que todos los encargados del contrato señalen que el dueño ha fallecido y este no deseñale de vida en los 3 meses consecutivos.*

    El contrato permite reclamar que se liquide la herencia si se cumplen las condiciones establecidas por el requerimiento.

    ```
    GET {{URL}}/api/testament/inheritance/claim
    ```

17. *El dueño del contrato debe poder llamar a una función en el contrato que sirva como prueba de vida para este.*

    El contrato registra en una variable `lastLifeSignal` el momento en que recibió la última prueba de vida. Se inicializa en `now` cuando se despliega el contrato. Se provee una función para actualizar su valor al ejecutarse. También, protegida por un *modifier* que restringe el acceso sólo al dueño.

    ```
    POST {{URL}}/api/testament/heartbeat
    ```

19. *Los encargados del contrato deben poder llamar a una función para indicar que el dueño de la herencia ha fallecido.*

    Cada encargado tiene la posibilidad de indicar el fallecimiento del dueño del contrato de forma independiente mediante una función que provee el contrato, que actualiza la variable `hasInformedDecease` del encargado correspondiente a verdadero.

    ```
    POST {{URL}}/api/testament/inform_owner_decease
    ```

20. *Los encargados del contrato deben poder llamar a una función para indicar que un heredero de la herencia ha fallecido.*

    La aplicación permite que cualquier encargado habilitado señale el fallecimiento de un heredero. Se decidió que alcanza con que solo un encargado indique que falleció un heredero para darlo por fallecido, a diferencia del requerimiento 16 que indica que todos los encargados deben hacer la indicación para dar por fallecido al dueño del contrato. El porcentaje de la herencia del heredero fallecido se ajusta según el requerimiento 9.

    ```
    POST {{URL}}/api/testament/inform_heir_decease
    ```

21. *Si pasan 36 meses sin recibir señales de vida del dueño del contrato, o exigencias de liquidación por parte de los beneficiarios o los encargados, el personal de la empresa debe poder liquidar el contrato y hacerse propietario de sus valores.*

    La aplicación permite liquidar el contrato a favor de la organización. Esta operación solo puede ser invocada por la cuenta de la organización y requiere que se cumpla la condición temporal respecto a las señales de vida. El contrato no se liquida de igual manera que la indicada en los demás requerimientos, sino que el contrato directamente se destruye y el balance se transfiere a la cuenta de la organización.

    ```
    POST {{URL}}/api/testament/inheritance/organization_claim
    ```

22. *Todos los retiros realizados por parte de los encargados deben quedar registrados en eventos en la blockchain con la siguiente información: Address del encargado que retira, Monto retirado, Motivo del retiro*

    Se creó un evento withdrawal, que contiene dichos datos. El evento está indexado por *address* del encargado que retira y existe un endpoint que permite visualizar los eventos de retiro del contrato, filtrados por *address* de encargado.

23. *Toda la información de los retiros realizados por parte de los encargados debe ser almacenada en alguna estructura dentro del contrato que permita su consulta por parte del dueño de la herencia: Address del encargado que retira, Monto retirado, Motivo del retiro*

El contrato tiene un array de retiros con la estructura de datos `Widthdrawal`, la cual contiene las variables `manager`, `ammount` y `reason` que corresponden a la dirección del encargado, el monto retirado y el motivo de retiro respectivamente.

24. *El dueño de la herencia debe poder aumentar el monto de esta a voluntad, conllevando con ello los cambios en porcentajes en las reglas correspondientes.*

    El contrato permite aumentar el monto total de la herencia mediante una función exclusiva para el dueño del mismo.

    ```
    POST {{URL}}/api/testament/inheritance/increase
    ```

25. *El dueño de la herencia podrá reducir el monto de esta, retirando una parte, con el pago de una multa establecida en porcentaje o monto fijo al momento de la contratación. Tener en cuenta que si se paga monto fijo, no se debe pagar porcentaje y viceversa.*

    El contrato permite reducir el monto de la herencia mediante una función exclusiva para el dueño del mismo. Se requiere que el monto sea menor al balance actual del contrato y que el porcentaje menor o igual al 100%. Antes de realizar la reducción, se transfiere a la organización la multa correspondiente. Esta multa por reducción de la herencia se envía como parámetro del constructor del contrato al momento de desplegarlo (`redFeeVal`) junto con una variable bool indicando si corresponde a un porcentaje (`redFeePercent`) o de lo contrario se considera monto fijo.

    ```
    POST {{URL}}/api/testament/inheritance/reduce
    ```

26. *Deben escribirse todos los modificadores que sean necesarios para dar cumplimiento a todos los requerimientos y reglas del negocio.*

    Se implementaron modificadores para distintos niveles de acceso a los métodos:
    - `onlyNotSuspendedManager`: controla que solo sea un encargado no suspendido quien realiza la acción.
    - `onlyOwner`: controla que sea el dueño del contrato quien realiza la acción.
    - `onlyOrganization`: controla que sea un miembro de la organización que realiza la acción.
    - `onlyJudiciaryEmployee`: controla que sea personal judicial quien realiza la acción (para modificación de reglas legales).

27. *Crear una aplicación que pueda ejecutar las operaciones necesarias para dar cumplimiento a los requerimientos y las reglas del negocio.*

    Este requerimiento fue satisfecho por la aplicación Node JS construida, la cual interactúa con los contratos, y expone endpoints REST para que los usuarios puedan ejecutar las operaciones sobre los contratos.

28. *En caso de que ningún beneficiario pueda cobrar la herencia, luego de pagar los costos y a los encargados, se entregará el saldo a una dirección de beneficencia determinada por las normas legales.*

    Se considera que ningún heredero puede cobrar la herencia si fueron todos señalados como fallecidos. En tal caso, previo a destruir el contrato en la liquidación del mismo, se transfieren los fondos con los costos descontados a la organización benéfica que establece el contrato de leyes.

29. *Es de significar que el monto que los encargados pueden retirar como adelante no puede ser fijado en un monto igual o superior al monto que recibirán como pago de sus servicios.*

    Esto se controló en el constructor del contrato de testamento donde se requiere que el porcentaje máximo que pueden retirar no puede ser mayor a la comisión.

Reglas del negocio

- Como fue establecido en la letra del obligatorio, las reglas legales que el contrato debe seguir deben estar estipuladas en otro contrato inteligente, que debe poder ser consultado por los contratos de herencia para saber cómo proceder en ciertos casos.

  El contrato de reglas legales[2] contiene una lista de cuentas que corresponde al personal del poder judicial. La dirección de quien publique el contrato es agregada automáticamente a tal lista y se provee la posibilidad de agregar más cuentas. El contrato de herencia utiliza este contrato de leyes importando su *ABI* y recibiendo su dirección por parámetro en el constructor.

  ```
  POST {{URL}}/api/laws/judiciaryEmployees
  ```

- Las reglas legales deben poder ser actualizadas según así lo requiera el dictamen de nuevas leyes. Esta actualización debe poder ser realizada sólo por personal del poder judicial.

  Las variables de las reglas legales son actualizadas de forma independiente. La actualización de cada variable se restringe a sólo personal del poder judicial.

  ```
  PUT {{URL}}/api/laws/dollarToWeiConversion
  PUT {{URL}}/api/laws/withdrawalFeePercent
  PUT {{URL}}/api/laws/withdrawalFinePercent
  PUT {{URL}}/api/laws/withdrawalFineMaxDays
  PUT {{URL}}/api/laws/charitableOrganization
  ```

- La herencia se pagara a los beneficiarios en el orden de prelación establecido por el dueño de la herencia, a menos que alguno de los encargados del contrato señale la existencia de hijos menores de edad de alguno de los beneficiarios, en cuyo caso estos beneficiarios pasaran a orden 1 en la prelación. En caso de haber más de un beneficiario con hijos menores de edad entonces su orden de prelación original será el que se use para ordenarlos dentro de la posición 1.

  Los encargados pueden anunciar la existencia de hijos menores de edad para un determinado heredero, lo cual marca su `hasMinorChild` en verdadero y modifica el orden de prelación a utilizarse al momento de pagar la herencia según corresponda.

  ```
  PUT {{URL}}/api/testament/heirs/{{ADDRESS}}/minor
  ```

- Cuando el contrato de herencia es creado debe cobrarse al dueño la cantidad de 200 dólares americanos, en base a la cotización establecida en el contrato que debe poseer las reglas legales.

---

[2] El contrato se encuentra en el archivo contracts/Laws.sol de la solución, disponible en: https://github.com/mcohen97/Blockchain-Inheritance-Management/blob/develop/contracts/Laws.sol

Para desplegar el contrato de herencia se requiere enviar como mínimo en el valor para la herencia el equivalente en weis de 200 dólares, de lo contrario el contrato no puede desplegarse. El costo se transfiere de la herencia a la cuenta de la organización. Se utiliza la siguiente `dollarToWeiConversion` en el contrato de leyes, por la cual se requiere una herencia no menor a 1 ETH para cumplir con esta regla.

USD	**ETH**	**WEI**
1	0.005	5000000000000000
200	1	1000000000000000000

- Cuando el contrato de subasta es liquidado se debe descontar del monto total a pagar las comisiones definidas al momento de la contratación como un porcentaje de los Ethers entregados al momento de la liquidación.
  Cuando el contrato de subasta es liquidado se debe pagar a cada uno de los encargados del contrato la suma establecida al momento de la creación del contrato. Este monto debe ser un porcentaje igual para cada encargado.

  Al momento de crear el contrato de subasta se indica una `managerFee` que representa el porcentaje de comisión de los encargados. Al momento de liquidar el contrato, primero se realizan las transferencias de la comisión correspondiente a cada encargado y luego se continúa con la liquidación del valor restante.

- En caso de que un encargado mantenga deuda con el contrato de herencia al momento de la liquidación:
  - Si la deuda es menor al monto que le corresponde como pago, entonces esta se le descontara del monto a pagar
  - Si la deuda es igual no se le pagara el porcentaje acordado y su deuda deberá ser manejada por la vía legal correspondiente, fuera del entorno de la blockchain.

  Como se mencionó anteriormente, se modelan a los encargados como objetos con propiedades, siendo su deuda actual una de ellas. Al hacer la liquidación se itera sobre los encargados del contrato y se les transfiere a su cuenta un monto igual al pago que le corresponde. En caso de que la propiedad deuda del encargado sea mayor o igual a lo que se le pagara, no se le transfiere nada y esa deuda es manejada por fuera del sistema, si es menos simplemente se le descuenta del monto a transferir.

- Una vez finalizada la liquidación del contrato de herencia, el contrato inteligente no debe mantener ningún fondo.
  El contrato inteligente debe quedar inhabilitado para recibir nuevas instrucciones o fondos una vez finalizada la liquidación.

  Al finalizar la liquidación el contrato de herencia, se destruye el contrato y el balance restante se envía al último heredero de forma de que no mantenga ningún fondo.

Limitantes

Durante el desarrollo, el equipo se encontró con algunas limitantes técnicas. Las principales fueron:

- La primitiva now, no utiliza el tiempo actual, sino el tiempo del último bloque minado[3], lo que significa que a menos que se realicen transacciones que cambien el estado del contrato, el heredero que debe dinero de hace más de 90 días puede seguir accediendo a la información estando teóricamente suspendido. Esto pasa poco frecuente en las redes de Ethereum, pero en la blockchain locales es más posible que suceda.

- El equipo descubrió en una etapa muy tardía del desarrollo que el contrato tiene un límite de tamaño de 24 Kb[4], lo cual implicó que se tuvieran que hacer recortes, y reducir algunos componentes que, si bien no eran parte de la funcionalidad, permitían visualizar mejor el estado del contrato. También se redujeron largos de strings y tamaños de variables.

- Al desplegar en la testnet de ropsten, se utilizó Ropsten Ethereum Faucet[5] para cargar las cuentas que desplegaban los contratos de ley y testamento. Cuando se carga una cuenta una segunda vez en un intervalo corto de tiempo, esta bloquea la IP que realizó la recarga y el address a quien se lo realizó por 24 hs. Por lo tanto, se volvió algo tedioso conseguir ethers para las cuentas que el equipo tenía, teniendo en cuenta que cada despliegue de contrato requiere un mínimo de un ether aproximadamente (200 dólares).

[3] Documentacion de Solidity: https://solidity.readthedocs.io/en/v0.5.3/units-and-global-variables.html#block-and-transaction-properties

[4]Articulo que habala sobre el maximo de tamaño: https://dev.to/mudgen/ethereum-s-maximum-contract-size-limit-is-solved-with-the-diamond-standard-2189#:~:text=Ethereum's%20Maximum%20Contract%20Size%20Limit%20is%20Solved%20with%20the%20Diamond%20Standard,-%23ethereum%20%23solidity%20%23&text=Ethereum%20contracts%20with%20too%20many,contract%20standards%20require%20many%20functions.

[5] https://faucet.ropsten.be/