

Proyecto 2: Informe

Integrantes

- David Rodriguez 14-10930
- Gregory Muñoz 16-11313

Resumen del proyecto

1.1 Generador de Votantes

Generar un conjunto de votantes ficticios, cada uno debe tener: - Nombre - Correo - Localidad Electoral - Clave Privada - Clave Publica - Address - Saldo Inicial de 10.000.000 wei (0.01 gwei)

Requerimiento:

- Definir un conjunto de Localidades Electorales
- Cada Votante debe asignarse a un centor de votacion en su localidad aleatoriamente
- 1% aprox de los votantes como candidatos

Ejecución

```
genVotante -f <archivo-localidades>
```

Ejemplo Archivo-Localidades

```
Local1 numVotantes1 numeroCentrosVotación1
...
Localn numVotantesn numeroCentrosVotaciónn
```

- Local-i: Representa Estado en caso de Venezuela
- numVotantes-i: Votantes inscritos en esa localidad
- numeroCentroVotacion-i: Centros de votacion disponibles en esa localidad

1.2 Generador de Votos

Generar y enviar votos a los centros de votacion

Requerimientos

- Voto aleatorio
- Voto enviado al centro correspondiente
- Envio de votos concurrente
- Porcetrnaje de abstencion entre 10% y 30%, debe ser de facil modificacion en el codigo (Usar variables globales)

Ejecucion

```
genVotos -n <archivo centro> -nc <n>
```

- n: Nivel de concurrencia de envio de votos
- arhcivo centro: Ubicacion del servidor del centro de votacion (nombre y puerto)

```
n
centro1 puerto1
centro2 puerto2
...
centron puerton
```

1.3 Escenario electoral

Responsable de interactuar con el centro1 para realizar el registro del escenario de votacion - Registrar votantes - Registrar Candidatos - Resgistrar Localidades

2 Centros de Votación

Servidor sencillo para procesar la intención de voto de los votantes pasos: - Recibir opción de voto - Crear transacción correspondiente (campo from del votante) - Enviar Transacción al nodo de la red

Ejecución

```
centroVotacion -n <nombreCentro> -p <puerto> -f <archivo datos nodo>
```

- nombreCentro: Nombre del centro de votación
- puerto: puerto para recibir peticiones
- archivo datos nodo: Datos de la ubicación del nodo Ethereum al cual se enviarán las transacciones

Nota: El centro1 recibirá la información de los escenarios del proceso electoral. En otras palabras, el proceso de ejercer los votos no puede comenzar antes de la configuración de los escenarios.

3 Software para Votación: Contrato Inteligente

- Inicializar el escenario: Crear el contrato e inicializar estructuras necesarias
- Reinicializar el escenario: Si el contrato ya está creado, se debe reinicializar las estructuras necesarias
- Registrar las localidades: Recibir localidades que participan en la elección
- Registrar Votantes: Registrar y asocia a cada votante con su localidad
- Registrar Candidatos: Registrar candidato en la localidad especificada (Deben ser votantes válidos)
- Recibir la intención de voto de votantes: Realizar verificaciones correspondientes al escenario asignado
- Cerrar el proceso de votación: Finaliza el proceso, no se pueden recibir más votos
- Reportar ganadores con sus porcentajes: Finalizado el proceso, retornar un resumen de los resultados, incluyendo abstención
- Dar un reporte por localidades: Finalizado el proceso, retornar un pequeño informe dando los resultados parciales (incluyendo abstención) para todas las localidades

4 Visualizador

Resumen como reporte detallado del proceso electoral * Extra: Posibilidad de presentar una interfaz web que le permita a los votantes ejercer su derecho al voto y obtener los resultados del proceso una vez finalizado

5 Escenario

Elecciones con una sola vuelta: Incluye elección del Presidente del país y de un Gobernador por localidad. Al finalizar el proceso, el nuevo presidente/gobernador es el candidato con mayor número de votos en su zona electoral.

Nota: En el caso del Gobernador, la zona electoral es la localidad y en caso del Presidente es la unión de las localidades.

Requerimientos para el proyecto:

1. Generador de Votantes:

- Método de generación random de personas con nombre, apellido y email
- Método para generar claves públicas y privadas para cada persona
- Método para leer el archivo de localidades
- Método para crear al votante usando los métodos anteriores
- Método para seleccionar candidatos aleatoriamente a partir de los votantes creados

2. Generador de Votos:

- Método para leer el archivo de centros
- Método para configurar en nivel de concurrencia y centros de votación
 - Se puede usar un Pool de Hilos
- Método para enviar votos al servidor de votación
 - Hacer uso del archivo de centros para saber en qué centro debe votar cada votante.

3. Servidor de Votación

- Método para inicializar el contrato
- Método para recibir petición de opción de voto
 - Debe recibir información del votante (Nombre, Email, Address)
 - Debe recibir las opciones de voto
- Método para crear la transacción según la opción elegida
- Método para crear la transacción registrar el derecho al voto del participante
 - Enviarla al contrato
- Método para enviar las transacciones generadas a la blockchain
- Nota:
 - Usar alguna librería para crear un servidor HTTP (Sugerido: Para python usar Flask, para JS usar Express)
 - Usar la librería web3 para la comunicación con el contrato inteligente

4. Contrato Inteligente

- Estructuras:
 - Votante
 - ID
 - Nombre
 - Localidad
 - Centro de Votación

- Candidato
 - ID
 - Nombre
 - Localidad
 - Cargo
 - Votos recibidos
 - Localidad
 - Nombre
 - Numero de Centros de Votacion
- Metodo para inicializar el escenario: (Puede ser el constructor del contrato)
- Metodo para Reinicializar el escenario: Mismo que inicializar pero un metodo a parte
 - Reiniciar el almacenamiento de los Candidatos, se puede dejar las Localidades y Votantes
- Metodo para Registrar las localidades:
 - Solo puede registrar el que despliego el contrato (O si tiene permiso)
 - Emitir evento de localidad Registrada
- Metodo para Registrar Votantes:
 - Solo puede registrar el que despliego el contrato (O si tiene permiso)
 - Emitir evento de Votante Registrado
- Metodo para Registrar Candidatos:
 - Solo puede registrar el que despliego el contrato (O si tiene permiso)
 - Emitir evento de Candidato Registrado
- Metodo para Recibir la intencion de voto de votantes (VOTAR):
 - Registrar la participacion del votante (sin el voto) y solamente aumentar en 1 segun el candidato votado, esto para preservar la propiedad de voto secreto
- Metodo para Cerrar el proceso de votacion:
 - Solo puede cerrarlo el que despliego el contrato
 - Emitir evento de Votacion Cerrada
 - Se puede definir una hora de cierre (Extra)
- Metodo para Reportar ganadores con sus porcentajes:
 - Colectar el ganador al puesto de Presidente y una lista de ganadores al puesto de Gobernador de cada Localidad
- Metodo para Dar un reporte por localidades
 - Colectar una lista con cada Candidato a la Gobernacion de una cierta localidad
 - Colectar una lista con cada Candidato a la Presidencia

5. Visualizador

- Generar una interfaz web haciendo uso de la Wallet de Metamask
- Hacer 3 estados de la app:
 1. Vista de Home:
 - Conectar con metamask y buscar con el address la informacion del Votante
 2. Vista de Votar:
 - Mostrar las opciones de Candidatos a Presidente y poder votar por uno solo
 - Mostrar las opciones de Candidatos a Gobernador segun la Localidad del Votante y poder votar por uno solo
 3. Vista de Resultados:
 - Mostrar resultados de la Votacion
 - Mostrar porcentajes de los votos recibidos de los Candidatos
 4. (EXTRA) Vista de fin de Votacion:
 - Ultima vista de fin del proceso Electoral

Inicializar el proyecto

1. Crear entorno virtual (VirtualEnv) e instalar requerimientos

```
virtualenv -p python3 env/
source env/bin/activate
pip install -r requirements.txt
```

2. Exportar Variables de entorno de Flask

```
export FLASK_APP=servidor/app.py
```

Nota: Si se quiere poner Flask en modo DEBUG, ejecutar

```
export FLASK_DEBUG=1
```

Correr el proyecto

1. Dentro de la raiz del proyecto, Generar los Votantes Ficticios

```
python3 genVotantes.py -f archivo-localidades-prueba.txt
```

Se deben generar 2 archivos: - wallets.json: Contiene la informacion basica de un Votante - localidades.txt: Contiene las localidades usadas

2. En otra terminal, inicializar el servidor de Flask

```
flask run
```

Se debe inicializar el servidor a la espera de nuevas solicitudes

3. En otra terminal, cambiarse al directorio "votacion" e inicializar el entorno del Contrato

```
cd votacion
brownie run build
```

Se debe recibir los logs de cada paso completado, al final sale un log de "Listo"

4. En otra terminal (Puede ser en la primera, donde se corrió el script para generar los votantes), generar los votos:

```
python3 genVotos.py -n centros.txt -nc 2
```

Va a empezar a aparecer logs de lo que va recibiendo cada Hilo de ejecución

5. (Extra) Para poder desde la interfaz web se puede acceder desde el navegador, teniendo el servidor de flask corriendo, a la URL "http://127.0.0.1:5000/"

Consideraciones

Decisiones de implementacion

- Se tomaron varias decisiones de implementación un poco distinto a lo requerido, por ejemplo, el requerimiento de crear un servidor por centro de votación, para la prueba se está usando un solo servidor central por temas de hardware, sin embargo con muy pocas modificaciones, por ejemplo, agregar los parámetros necesarios de "nombre del centro", "puerto" y "archivo de datos nodo", se puede levantar un conjunto de servidores representando cada uno un servidor dedicado a cada centro de votación, de esta manera se puede descentralizar a los votantes y evitar fallas de seguridad.
- Otra decisión fue, como se está trabajando en un ambiente local usando Ganache-cli como nodo de ethereum, es requerido que se las cuentas estén registradas en Ganache, por lo que el registro se hace con las claves privadas de los votantes generados, esto en la práctica no es necesario, incluso es muy peligroso, en la práctica basta con conectar tu wallet al nodo servidor y listo.
- En el Escenario Electoral se está comunicando directamente con el contrato con la librería web3.py, no se está pasando por un servidor de algún centro de votación ya que este proceso podría ser muy engorroso y probablemente menos seguro que la comunicación directa con el contrato.
- Se realizaron un conjunto pequeño de UnitTests para comprobar correctitud de contrato
- Para los patrones de Almacenamiento de datos en el contrato, se implementaron de la siguiente forma: En general cada estructura (Localidad, Votante y Candidato) tiene el siguiente patrón de almacenamiento (Usando el caso de Votante): - mapping (address => uint256) indexVotante; - Votante[] votantes; Esto con la finalidad de poder buscar los datos de un Votante, en este caso, en $O(1)$ ya que se pregunta el índice del votante dado el address y con índice se busca después en el Arreglo, para agregar un nuevo Votante, teóricamente al ser dinámico, inserta en $O(1)$ (A menos de que se tenga una abstracción de la implementación distinta) y agregar la entrada en el mapping igualmente $O(1)$. Finalmente para poder recorrer el arreglo de Votantes, en $O(n)$
- Hay un pequeño fallo en todo esta implementación y es que las cuentas que no son generadas al inicio por ganache no son reconocidas para ejecutar el contrato, es decir, cuando se corre el generador de Votos, da un response_code 400 desde el servidor ya que al ejecutar la función de Votar firmada por un address que no se generó al principio no la reconoce, devuelve un error de "ValueError: {'message': 'sender account not recognized'}". Lo que se cree, primero, es que hay una configuración de ganache que no se tiene muy claro y esto impide que se usen cuentas "externas", lo segundo es que, según la investigación que se hizo sobre este error es que ganache, al ser un simulador de un nodo de ethereum tiene quizás otras restricciones que no están en un nodo real, por lo que puede ser tanto un error como una decisión de implementación, totalmente válido.
- Por último, para probar desde la interfaz web con una cuenta de brownie, se tiene que acceder a la consola de brownie, obtener el address de cualquiera de las cuentas base, por ejemplo la `accounts[0].address`, y colocarla en la variable `metamaskConectado` dentro de `servidor/templates/index.html`, seguido recargar la interfaz web y NO DARLE AL BOTÓN "CONECTAR CON METAMASK" ya que se cambiaría por el address de metamask, solamente darle al botón de Votar, después es el mismo proceso de votar.