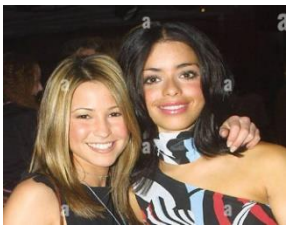
Trabajo 2 (25%)



Punto 1 (Multichain) (50%)

La academia de artes marciales mixtas de **Tina Barret** y **Rachel Stevens** ha decidido entregar títulos académicos en formato digital a quienes finalicen satisfactoriamente sus programas de formación. Estos certificados electrónicos no deben ser manipulados una vez sean expedidos (¡Tina y Rachel son muy exigentes en este aspecto!) por lo cual se tomó la decisión de crear una *blockchain* para este propósito.

a) (5%) En un equipo cree una *blockchain* en Multichain llamada "Titulos". El equipo puede ser una máquina virtual o un computador de uno de los miembros del grupo. Este nodo será el nodo admin de la *blockchain*. Una vez creada la *blockchain*, conecte otro equipo o máquina virtual como un segundo nodo a la *blockchain* "Titulos". Cree un *stream* llamado "Graduandos" y otro llamado "Certificados". Estos *streams* se deben crear desde el nodo admin, pero se manipularán desde el segundo nodo. **Conceda los permisos** necesarios para estas acciones.

(10%) Desde el nodo admin cree los siguientes *smart filters*:

- *Smart filter* de *stream* que valide lo siguiente para el *stream* "Graduandos":
  - Debe tener una única *key*. Esta *key* será el *pasaporte* con el que se identificará a cada graduando.
  - El pasaporte deberá ser numérico con mínimo 10 y máximo 15 dígitos.
  - Solo se permiten datos tipo JSON.
  - El JSON debe contener los siguientes campos (ni más, ni menos): *nombre, apellidos, fecha_nacimiento* y *celular*. El campo fecha_nacimiento debe ser una fecha válida (usted define el formato) y el campo celular debe ser numérico con mínimo 10 y máximo 15 dígitos.
- *Smart filter* de *stream* que valide lo siguiente para el *stream* "Certificados":
  - Debe tener exactamente dos *keys*. Estas *keys* corresponden al *pasaporte* del graduando y al *código* del programa del que se gradúa (la primera *key* corresponde al pasaporte y la segunda al código del programa).
  - Solo se permiten datos tipo JSON.
  - El JSON debe contener los siguientes campos (ni más, ni menos): *fecha_expedicion* y *diploma* (*diploma* es un PDF, ver punto b) donde se dan más detalles sobre este campo). El campo fecha_expedicion debe ser una fecha válida (usted define el formato).

Estos filtros deben ser activados solamente para el segundo nodo.

Para este punto, envíe un archivo de texto con los comandos que usó para hacer lo solicitado indicando claramente en qué nodo se ejecuta cada comando y el orden en el que se deben ejecutar. Para los *smart filters*, incluya por separado el código de la función de forma legible (archivo txt tal que al copiar y pegar no haya caracteres extraños que generen errores de compilación).

b)

Cree un programa en Java que permita interactuar con la *blockchain* creada en el punto anterior por medio de una interfaz gráfica. Dicho programa se deberá conectar a la *blockchain* desde el segundo nodo y deberá contar con las siguientes funcionalidades:

- **(10%)** **Registro de graduando***:* Permite ingresar los campos necesarios para almacenar un graduando en el *stream* "Graduandos", incluyendo el pasaporte que se usará como *key* y los campos *nombre, apellidos, fecha_nacimiento* y *celular*. El programa Java debe validar que el pasaporte no se repita en el stream "Graduandos".
- **(10%)** ***Subir certificado****:* Permite ingresar los campos necesarios para almacenar un certificado en el *stream* "Certificados", incluyendo sus dos *keys*, donde la *key* que hace referencia al graduando debe corresponder a un pasaporte (*key*) del *stream* "Graduandos" (el programa Java debe hacer la validación correspondiente). Para ingresar los certificados en la *blockchain* (campo *diploma* del JSON), se deben convertir los PDF a texto implementando un codificador y decodificador de PDF en su programa o usando una herramienta de conversión. Se garantiza que solo se ingresarán PDFs con un tamaño inferior a 50KB.
- **(15%)** ***Ver certificados****:* Permite listar los certificados "filtrando" por graduando (*pasaporte*) o por código del programa (o por ambos filtros a la vez) junto con la información almacenada de dicho certificado y su respectivo graduando. Se da libertad para decidir la forma en la que se muestra el campo "diploma" del stream "Certificados", ya sea permitir descargar el PDF original en una carpeta del sistema operativo o mostrarlo directamente a través de la interfaz.

Envíe el proyecto Java SIN construir, de modo que se puedan modificar los parámetros de la conexión. Si se usan librerías externas adjúntelas al proyecto e incluya instrucciones que considere necesarias para hacer el *build* del proyecto.

Punto 2 (Solidity) (50%)

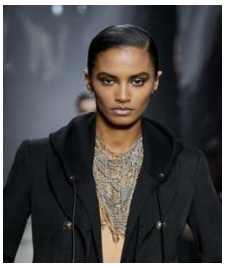
Elabore un contrato en Solidity que cumpla los siguientes requisitos:

(4%) Cuando un contrato se instancia, el usuario que lo instancia queda como el presidente. El contrato puede ser instanciado para manejar una, dos o tres propuestas. Una propuesta contiene lo siguiente: código de la propuesta (un número entero), texto de la propuesta, número de opciones de la propuesta (se debe validar que sea un número entero entre 2 y 10), cuánto cuesta votar (se debe validar que sea un número entero entre 1 y 5, representa ethers) y tiempo que estará abierta la propuesta para ser votada (se debe validar que sea un número entero mayor a 120, representa segundos). Veamos un ejemplo concreto:

Supongamos que el contrato se va a instanciar para las siguientes **dos** propuestas:

Primera propuesta:
Código de la propuesta: 985
Texto de la propuesta: “Contratar a **Shalom Harlow** para la inauguración”
Número de opciones de la propuesta: 6
Cuánto cuesta votar: 3 (ethers)
Tiempo de la propuesta: 500 (segundos)

Segunda propuesta:
Código de la propuesta: 432
Texto de la propuesta: “Traer a **Sacha Quenby** para el desfile”
Número de opciones de la propuesta: 4
Cuánto cuesta votar: 2 (ethers)
Tiempo de la propuesta: 300 (segundos)



Es decir, en este contrato los usuarios pueden votar por dos propuestas: en la primera disponen de 6 opciones (1, 2, 3, 4, 5 y 6) y emitir su voto les cuesta 3 ethers. En la segunda disponen de 4 opciones (1, 2, 3 y 4) y emitir su voto les cuesta 2 ethers.

El contrato debe tener las siguientes funciones:

- **(10%) Votar**: esta función recibe el código de una propuesta y la opción de la propuesta por la cual el usuario desea votar. El presidente no puede votar. Un mismo usuario puede votar varias veces en una misma propuesta pero con estas condiciones: la primera vez que vota le cuesta el valor indicado por la propuesta (por ejemplo, 3 ethers en la propuesta 985). Si desea votar por segunda vez en esta misma propuesta le cuesta el doble (es decir, 6 ethers), si desea votar por tercera vez le cuesta el doble (es decir, 12 ethers), si desea votar por cuarta vez le cuesta 24 ethers y así sucesivamente. Un mismo usuario puede votar como máximo 5 veces en una misma propuesta. Cada vez que un usuario vota en una propuesta, puede votar por la misma opción o elegir una opción distinta. Así vote por la misma opción en una propuesta se cuentan como distintos votos para la propuesta.
- **(5%) NombrarBeneficiario**: (solamente para propuestas cuyo tiempo **NO** haya acabado y solamente el presidente está autorizado para esta función). Esta función recibe la dirección de un usuario y el código de una propuesta. El usuario correspondiente a la dirección queda entonces nombrado como beneficiario de la propuesta. Una propuesta puede tener como máximo cuatro (distintos) beneficiarios.
- **(7%) AumentaTiempo:** (solamente para propuestas cuyo tiempo **NO** haya acabado y solamente el presidente y los beneficiarios de la propuesta están autorizados para esta función). Esta función recibe el código de una propuesta y un número (un múltiplo de 60) con la cual el usuario desea incrementar el tiempo de la propuesta. Aumentar el tiempo en 60 segundos cuesta 1 ether, aumentarlo en 120 segundos cuesta 2 ethers y así sucesivamente. Una propuesta solamente puede recibir en total un máximo de 5 aumentos (ya sean hechos por el mismo o por diferentes usuarios). Cada aumento debe ser como máximo de 300 segundos.

- **(7%) DecrementaTiempo:** (solamente para propuestas cuyo tiempo **NO** haya acabado y solamente el presidente y los beneficiarios de la propuesta están autorizados para esta función). Esta función recibe el código de una propuesta y un número (un múltiplo de 60) con la cual el usuario desea decrementar el tiempo de la propuesta. Decrementar el tiempo en 60 segundos cuesta 2 ethers, decrementarlo en 120 segundos cuesta 4 ethers y así sucesivamente. Una propuesta solamente puede recibir en total un máximo de 3 decrementos (ya sean hechos por el mismo o por diferentes usuarios). Cada decremento debe ser como máximo de 180 segundos. En el momento que se va a decrementar el tiempo de una propuesta, a la propuesta le debe faltar más de 2 minutos para terminar, de lo contrario el decremento se rechaza.

  Nota: ofrezca funciones o variables públicas donde todos los usuarios puedan ver el tiempo de cada propuesta y así poder saber si el tiempo de una propuesta ha aumentado o disminuido.

- **(5%) EntregaResultados**: (solamente para propuestas cuyo tiempo haya acabado y solamente el presidente está autorizado para esta función). Esta función recibe el código de una propuesta y la función devuelve cada opción de la propuesta cuantos votos recibió y el total de dinero que se recaudó en dicha propuesta.
- **(5%) TransferirDinero**: (solamente para propuestas cuyo tiempo haya acabado y solamente el presidente está autorizado para esta función). Esta función recibe el código de una propuesta y la función transfiere la mitad del dinero recaudado al presidente. La otra mitad del dinero se reparte equitativamente entre los beneficiarios de la propuesta. Si la propuesta no tiene beneficiarios, todo el dinero se transfiere al presidente.
- **(3%) ConsultarDineroPropuesta**: (solamente para propuestas cuyo tiempo haya acabado y solamente el presidente y los beneficiarios de la propuesta están autorizados para esta función). Esta función recibe el código de una propuesta y la función devuelve cuanto dinero recibe el presidente, cuanto recibió cada beneficiario y cuantos beneficiarios tuvo la propuesta.
- **(4%) ConsultarDineroTodas**: (solamente si todas las propuestas han acabado y solamente el presidente está autorizado para esta función). Esta función recibe el código de una propuesta y la función devuelve cuánto dinero recibe en total el presidente considerando todas las propuestas y cuanto recaudó el presidente en cada una de las propuestas.

Notas adicionales:

- Para enviar por email a *fjmoreno@unal.edu.co* y a *caaruizra@unal.edu.co* el jueves 13 de abril **hasta las 11 am**. Solo se califican trabajos enviados a esos dos correos.
- **No se reciben trabajos en hora posterior**. No se reciben versiones "mejoradas". No se califican trabajos enviados "por accidente" a otros correos.
- Grupos máximo de **tres** personas.
- Los trabajos deben ser independientes entre los grupos. Trabajos copiados **así sea en una SOLA parte** se califican con 0 (cero) en su totalidad para todos los integrantes.
- El monitor les puede ayudar con los aspectos técnicos, **en especial de instalación y explicación de comandos**, pero su función **no** es hacerle el trabajo **ni está autorizado** para **cambiar las condiciones del trabajo, pero sí les puede hacer aclaraciones con respecto**

al enunciado en especial en la parte de Multichain ya que en gran parte fue elaborado por él.

- **Si trabaja con otros lenguajes o herramientas**, así su trabajo funcione y sea "espectacular", el trabajo se califica con cero.
- Si encuentra errores en el enunciado por favor avisar lo más pronto posible para hacer las correcciones respectivas.
- Favor enviar en **un solo mensaje** todo el trabajo.
- **Incluya el nombre de todos los integrantes en el mensaje.**

Agradecimientos al monitor por la elaboración del enunciado para la parte de Multichain; (revisado y modificado levemente por Francisco Moreno)
24 de marzo de 2023.