Implementación de contratos inteligentes para procesos colaborativos con Hyperledger Fabric como framework de blockchain

Pairetti Ribotta, Lucas José - ljpairetti@gmail.com

Peiretti, Tomás - tomaspeiretti@gmail.com

Universidad Tecnológica Nacional, Facultad Regional de Santa Fe

Abstract: *Junto al desarrollo de tecnologías relacionadas al almacenamiento de datos en donde se asegura la integridad y la privacidad de la información surge una nueva necesidad relacionada con la gestión de forma descentralizada. Estos requerimientos se lograron implementar mediante el uso de blockchain. Los procesos colaborativos entre organizaciones requieren de una gestión descentralizada de la información que se intercambia, para lo cual se propone aplicar la tecnología de blockchain mediante el uso de la plataforma HyperLedger Fabric. La propuesta describe la implementación de contratos inteligentes para la validación de una transacción enviada por algún miembro de la red, permitiendo un mecanismo de confianza en donde los participantes se conocen entre sí y el cumplimiento de las políticas de negocio que existen dentro de dicho proceso.*

Palabras clave: Blockchain, Contrato inteligente, proceso de negocio colaborativo, descentralización.

Introducción

El avance de las comunicaciones y la tecnología, permiten la cooperación entre distintas organizaciones para prestar nuevos y mejores servicios a sus clientes [1]. Así, las empresas conforman redes colaborativas integrando sus procesos entre sí, lo cual requiere un extenso intercambio de información entre las partes involucradas, en un marco que les asegure integridad y privacidad de la información, funcionando de modo descentralizado. Esto representa un desafío para que las organizaciones puedan lograr sus metas de negocio en común. Por lo tanto, es necesario adoptar el uso de tecnologías que permitan una comunicación fluida, confiable y privada para conseguir estos fines interorganizacionales.

La tecnología de blockchain (cadena de bloques) se presenta como un medio para estrechar los lazos entre organizaciones que colaboran en el desarrollo de sus procesos de negocios, al proporcionar más integridad, seguridad y transparencia a los datos [2,3]. Esta cadena de bloques permite capturar la historia y el estado actual de los procesos de negocio colaborativos, al registrar en cada bloque las transiciones de las transacciones ejecutadas entre las partes. Este registro distribuido es inmutable e invulnerable, garantizando así la confianza de que la información almacenada no ha sido alterada de forma deliberada o accidental [5]. La información sobre el estado de las instancias de cada proceso se puede compartir y actualizar localmente en cada nodo [4].

Así como ha existido a lo largo del tiempo el uso de contratos entre empresas y personas, en las cadenas de bloques se ha desarrollado la tecnología definida como contrato inteligente (smart contract), que permite a las organizaciones definir sus políticas de negocio a través de acuerdos escritos en código que se ejecutan automáticamente dentro de la cadena de bloques [6].

En este trabajo se propone la implementación de contratos inteligentes para la validación de mensajes enviados entre las

organizaciones al llevar a cabo sus procesos colaborativos, basados en una red de Blockchain, la cual emplea la plataforma y framework de HyperLedger Fabric (HF). Esta red permite que las organizaciones posean los permisos de proponer sus políticas de negocio y aplicarlas a una red de almacenamiento de información para el envío de mensajes entre los distintos participantes.

Arquitectura de una red blockchain para gestionar procesos colaborativos

Para poder definir la arquitectura de una blockchain en Hyperledger Fabric se emplea el uso de objetos denominados nodos, los cuales representan a una o más organizaciones (Figura 1) en un canal. Estos nodos se clasifican según su tarea; pueden ser los encargados de validar las transacciones provenientes de distintas aplicaciones, o pueden ser nodos que se encargan de ordenar las transacciones. Por un lado, se encuentran los nodos peer y de endorsamiento, que son los encargados de validar las transacciones provenientes de distintas aplicaciones. En estos se encuentran los contratos inteligentes, dominados Chaincodes. Por otro lado, se tienen los nodos de ordenamiento los cuales pueden pertenecer a alguna de las organizaciones. Estos son los encargados de recibir transacciones previamente aceptadas por los contratos de los nodos peers o endorsamiento y ordenarlos para su posterior almacenamiento en el Ledger de la blockchain, el cual es accesible a las distintas organizaciones.

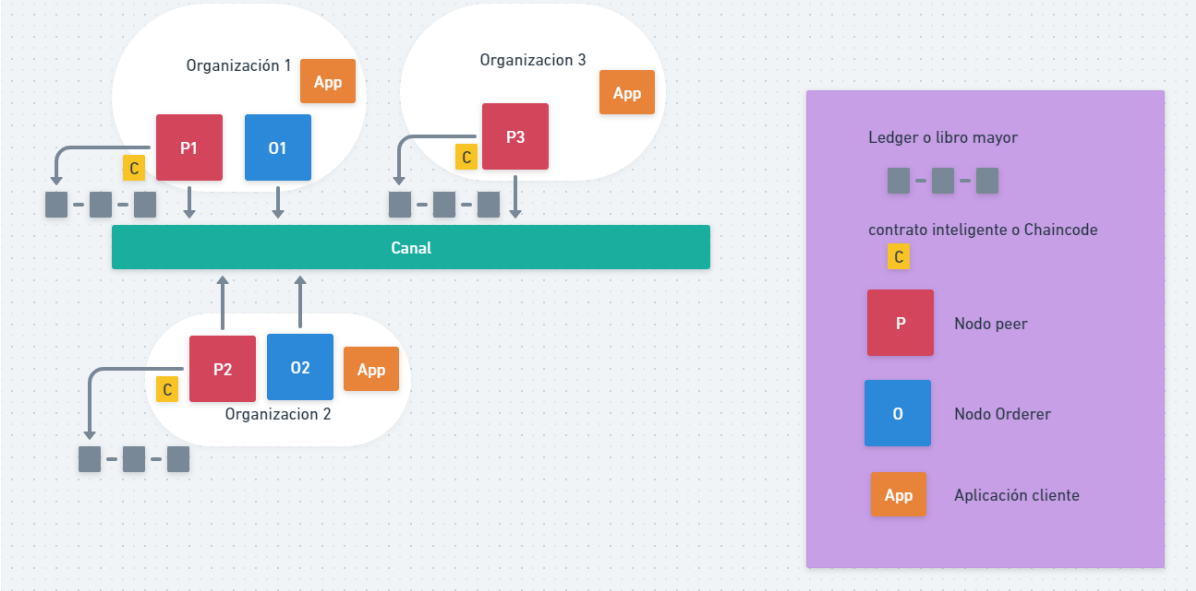

Figura 1: Red Blockchain de HyperLedger

En este trabajo se propone la definición de un canal por cada modelo de proceso colaborativo a ejecutar, a través del cual las distintas organizaciones transmitirán mensajes y estos se verificarán mediante contratos inteligentes para asegurar la integridad y validez del orden de los mensajes, para posteriormente ser almacenados en el ledger.

Definición de un contrato inteligente en HyperLedger Fabric

Un contrato inteligente (smart contract) permite modificar los estados de un objeto de negocio y gobierna los procesos que mueven dicho objeto entre los diferentes estados posibles. Los contratos permiten a desarrolladores implementar la lógica de los procesos de negocio colaborativos y los datos que serán compartidos a través de las diferentes organizaciones que participan en la red blockchain. Los contratos son implementados en paquetes denominados chaincode. Ante la llegada

de una transacción que representa el envío de un mensaje en un proceso, estos contratos revisan el formato, contenido y orden de dicho mensaje con el objetivo de verificar que cumpla con las políticas de negocio definidas por las organizaciones en el modelo de proceso colaborativo.

Despliegue de contratos inteligentes

Las organizaciones que desean validar transacciones en el ledger deben instalar un contrato inteligente (chaincode) en sus nodos Peers unidos a un canal asociado a un proceso colaborativo. Luego, los miembros del canal podrán implementar el chaincode y usarlo para crear o actualizar elementos del ledger en el canal, esto es enviar mensajes en la ejecución de los procesos. Un chaincode es implementado en un canal usando un proceso llamado Fabric chaincode lifecylcle en donde se permite que múltiples organizaciones acepten cómo un chaincode será operado antes de ser utilizado para aceptar transacciones.

Proceso de implementación:

1. Implementar en un package (chaincode) del contrato inteligente: en esta tarea, que puede ser realizada solo por una organización, se guarda el contrato en un paquete para poder enviarse al resto de organizaciones
2. Instalar el chaincode package: toda organización que requiera usar el contrato para validar las transacciones deberá realizar este paso e instalar el contrato en su nodo Peer
3. Aprobar el chaincode definido: Todas las organizaciones del canal deberán votar por dicho contrato. Para que un contrato sea aceptado debe obtener la aprobación de suficientes organizaciones para poder satisfacer las políticas de negocios de la red. Toda organización que acepte el contrato deberá instalarlo en sus nodos Peer.
4. Establecer el chaincode dentro del canal. Una vez el contrato consigue la suficiente cantidad de votos positivos, éste es subido por una organización al canal, para que cuando se reciba una transacción el contrato sea ejecutado en todos los nodos Peers que lo contengan.

Proceso de validación de una transacción y almacenamiento en el ledger

El proceso de recibir una transacción disparada por la aplicación de software de una de las organizaciones participantes de la red consta principalmente de tres etapas: ejecución, ordenamiento y validación. Como se muestra en la figura 2, el proceso comienza con un evento producido por una de las organizaciones participantes a través de una aplicación de software privada de la organización. Dicho evento dispara una transacción, la cual tiene la particularidad de tener un orden que debe cumplirse. Este orden es asegurado por los chaincodes, los cuales establecen cómo debe ser la estructura del contenido, cuál debe ser la organización origen y destino para dicho tipo de mensaje y otras particularidades, las cuales son dependientes del tipo de política de negocio del proceso.

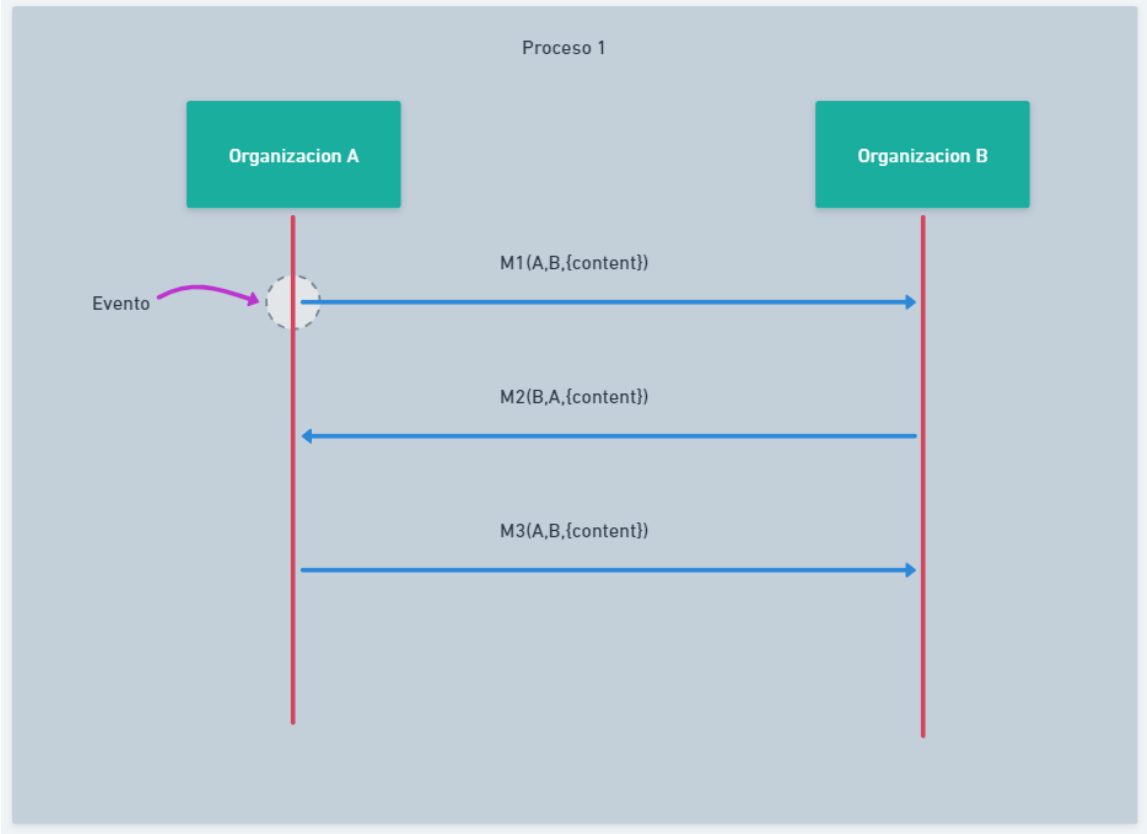

Figura 2: Ejemplo de comunicación entre 2 organizaciones para un evento

Teniendo en cuenta la estructura preestablecida en la figura 1, el procedimiento para la validación de una transacción comienza mediante el envío de la transacción por una aplicación de alguna de las organizaciones hacia el canal. Los nodos de endorsamiento reciben y ejecutan la transacción aplicando los chaincodes. Estos nodos capturan una simulación de la ejecución de la transacción con sus operaciones de lectura y escritura. Si la mayoría de estos nodos lo aceptan, devolverán la transacción firmada y encriptada. Luego la aplicación envía la transacción hacia los nodos de ordenamiento; en este momento varias aplicaciones podrían enviar transacciones en simultáneo hacia estos nodos. Estos nodos reenvían las transacciones de forma ordenada a todos los nodos de la red los cuales validaran a través de los chaincodes que la transacción cumple con sus políticas de negocio. Si las políticas se cumplen, las actualizaciones se retienen y se guardan en el ledger y se actualiza el "world state", el cual es el último estado del ledger. En caso de no aceptarse, se guardan de igual manera, pero no actualizan el world state. Posteriormente a esto la red envía una notificación sobre la situación a todos los nodos de las distintas organizaciones y a la aplicación cliente la cual envió la transacción para determinar el resultado final.

Desarrollo de smart contracts (chaincodes) para procesos colaborativos

En HyperLedger Fabric los chaincodes (implementaciones de smart contracts) pueden ser desarrollados en diversos lenguajes como Javascript, Java, Go, etc, a través del uso de librerías específicas (tales como "`org.example.ledgerapi.State`", "`org.hyperledger.fabric.contract`" para Java). Usando estas librerías se definen los datos de un contrato inteligente tales como el nombre, la versión, una descripción y demás datos.

Para la ejecución de procesos colaborativos, se define un chaincode por cada proceso colaborativo. El

chaincode contiene la lógica de validación que posibilita determinar si un envío de mensaje, a partir de una transacción disparada por alguna aplicación perteneciente una de las organizaciones que conforman la red en la blockchain, es correcto. Esto implica verificar que el formato, contenido y orden del mensaje son correctos respecto a lo definido por las organizaciones en un modelo de proceso colaborativo. A continuación, se describen partes de un chaincode en Java para un proceso colaborativo dado. Lo siguiente muestra la definición del contrato con información del mismo.

```
@Contract(
      name = "basic",
      info = @Info(
          title = "Order Management",
          description = "The logic of the collaborative process 'Order Management'",
          version = "0.0.1-SNAPSHOT",
          license = @License(name = "Apache 2.0 License",
url="http://www.apache.org/licenses/LICENSE2.0.html"),
          contact = @Contact(email = "info@collaborativenetwork.com",
                        name = "Collaborative Network",
                        url = "https://collaborativenetwork.com")))
```

Posterior a esto se crea la clase del contrato que implementa la interface "ContractInterface". Para nuestro caso se definió la siguiente clase:

```
public class CollaborativeOrderManagement implements ContractInterface
```

En esta clase se definen los métodos deseados para la interacción de las distintas organizaciones con la blockchain. Los métodos que se requieren son con el objetivo de una inserción, modificación, eliminación o lectura. Estos métodos son específicos de cada proceso, donde se indican las condiciones que debe cumplir la transacción para poder ser validada. Para la definición de métodos de transacciones se emplea la palabra reservada `@Transaction` para indicar que el siguiente método es una transacción. De esta manera, para el caso de envío de mensajes, se define un método:

```
public Message sendMessage(final Context ctx, final String idProceso, final String emisor, final String receptor, final Document documento) {

        /* A continuación la lógica que valida el cumplimiento de las políticas de negocio definidas en el proceso. En caso de no cumplirlas el sistema emplearía el uso de excepciones para enviar mensajes de alerta y rechazar dichas transacciones, empleando el uso de excepciones dadas por la API de hyperledger */

        /* esto especifica la creación de un mensaje para una instancia de un proceso, es decir, lo que se almacena en el ledger */

        Message processMessage = new ProcessMessage (idProceso, emisor,

                receptor, documento);

/* Serializa el objeto a un formato JSON (esto podría variar en función de cómo se desea estructurar la información */

        String messageJson = genson.serialize(processMessage);

        /* Se guarda el nuevo estado en el ledger y se retorna el mensaje y el control al usuario /*

        stub.putStringState(idProceso, messageJson);

        return processMessage;

}
```

Dependiendo de las necesidades de las organizaciones, se pueden emplear diferentes elementos de la API de hyperledger para formar métodos más específicos para políticas de negocio más estrictas que impliquen la necesidad de un grado de complejidad en la codificación más elevado.

Estructura y consulta de la instancia del proceso y sus mensajes

Además de los chaincodes, resulta importante definir como almacenar cada una de las instancias del proceso junto con los mensajes asociados. Se propone entonces, la siguiente estructura:

```
Proceso {

	docType (proceso); // Se utiliza para indicar el tipo de objeto

	idProceso;	//	Útil	para identificar unívocamente a la instancia

	variablesProceso	//	Estado lógico del proceso, útil para analizar al proceso

	nombreProceso; // Nombre de la instancia del proceso

}

Mensaje {

docType (mensaje) // Se utiliza para indicar el tipo de objeto

	IdMensaje; // Identificador del mensaje.

	IdDeProceso; // Identificador de la instancia del proceso asociado

Emisor; // Emisor del mensaje

Receptor; // Receptor del mensaje

Contenido; // Documento, archivo o información que se envía en el mensaje

}
```

Hyperledger Fabric permite utilizar CouchDB, una base de datos no relacional, por lo que es posible guardar y consultar a través de estructuras JSON. Esta base de datos permite la utilización de índices para eficientizar las consultas, por ende, en conjunto con la estructura dada, se plantea el siguiente índice:

```
{
  "index":{
      "fields":["docType","idProceso"]
// Nombre de los campos a ser consultados
  },

"ddoc":"allMessagesFromIdProcesoIndex",
// (opcional) Nombre del design document donde se guardará el índice
  "name":"indexMessages", // Nombre del índice
  "type":"json" // Tipo, siempre debe ser "json".
}
```

Este índice soporta la consulta de todos los mensajes asociados a una instancia del proceso. Debe ser guardado en la ruta *META-INF/statedb/couchdb/indexes* y empaquetado junto con el chaincode asociado poder ser utilizado.

Respecto al chaincode, es posible definir dentro de este una función similar a la que se presenta a continuación para poder realizar consultas al Ledger. Esta función permite, dada una cadena de caracteres que defina la consulta, ejecutar dicha consulta sobre el Ledger.

```go
func (t * SampleChaincode) QueryProcess(ctx contractapi.TransactionContextInterface,
                                        queryString string) ([]*Mensaje, error){
    //Consultar el Ledger
    resultsIterator, err := ctx.GetStub().GetQueryResult(queryString)
    if err != nil{
        return nil, err
    }
    defer resultsIterator.Close()

    //Procesar resultados
    var mensajes []*Mensaje
    for resultsIterator.HasNext(){
        queryResult, err := resultsIterator.Next()
        if err != nil {
            return nil, err
        }
        var msje Mensaje
        err = json.Unmarshal(queryResult.Value, &msje)
        if err != nil {
            return nil, err
        }
        mensajes = append(mensajes, &msje)
    }

    return mensajes, nil
}
```

Luego, retomando el ejemplo presentado en la figura 2 y asumiendo que el id de proceso es 1313, si se realiza la siguiente consulta *'{\"selector\":{\"docType\":\"mensaje\", \"idDeProceso\":\"1313\"}, \"use_index\":[\"_design/allMessagesFromIdProcesoIndex\", \"indexMessages\"]}"]}* se obtendría un resultado similar al siguiente:


```
[{"docType":"mensaje","IdMensaje":"msj1
","IdProceso":"1313","Emisor":"A","Rece
ptor":"B","Contenido":"Contenido  Msje
1"},

{"docType":"mensaje","IdMensaje":"msj2"
,"IdProceso":"1313","Emisor":"B","Recep
tor":"A","Contenido":"Contenido  Msje
2"},

{"docType":"mensaje","IdMensaje":"msj3"
,"IdProceso":"1313","Emisor":"A","Recep
tor":"B","Contenido":"Contenido  Msje
3"},]
```


Siguiendo la misma idea, podría consultarse por ejemplo cuál fue el último mensaje enviado para cierto ID de proceso (tomando el que tenga mayor IdMensaje si este es secuencial, por ejemplo), u obtener el conjunto de participantes en el proceso a partir de todos los mensajes enviados, los emisores y receptores.

Conclusiones y Trabajos Futuros

En el presente trabajo se definió la estructura que debería poseer una red de Hyperledger Fabric y las herramientas que emplea para satisfacer las necesidades de las organizaciones acerca de cómo transmitir la información de forma que se mantenga la privacidad e integridad de la misma, manteniendo la descentralización de la toma de decisiones a la hora de realizar un proceso colaborativo entre las mismas.

El diseño propuesto propone emplear el uso de contratos inteligentes (chaincodes) como herramienta para asegurar que las políticas de negocio se respeten, dándole una estructura correcta y un orden en el que las diferentes transacciones deben enviarse. La ejecución de dichos contratos inteligentes valida todas estas políticas, permitiendo que la red guarde en su Ledger o base de datos distribuida todas las transacciones. A su vez, permite que las transacciones validadas por las políticas de negocio actualicen el world state del sistema. Mediante las APIs que provee HyperLedger Fabric se da la

posibilidad de crear los contratos inteligentes teniendo la capacidad de emplear diferentes lenguajes de programación para la realización de esta tarea.
Para el futuro de este trabajo, se espera implementar una red colaborativa empleando estas herramientas de contratos inteligentes en la plataforma de Hyperledger Fabric. Se espera poder definir estructuras para diferentes tipos de contratos inteligentes para los procesos colaborativos con el objetivo de definir correctamente que transacciones permitirán actualizar el world state.

Agradecimientos

Este trabajo de investigación se lleva a cabo en el marco del Proyecto PID UTN – SIUTIFE0007771TC "Tecnologías de blockchain y computación en la Nube para la gestión de procesos de negocio colaborativos".

Referencias

[1] Snyder LV, Shen ZJM (2011) Fundamentals of supply chain theory. Wiley, Hoboken.

[2] Mendling J, Weber I, van der Aalst WMP, vom Brocke J, Cabanillas C, Daniel F, Debois S, Di Ciccio C, Dumas M, Dustdar S, Gal A, Garc´ia-Bañuelos L, Governatori G, Hull R, La 35 (1).

Rosa M, Leopold H, Leymann F, Recker J, Reichert M, Reijers HA, Rinderle-Ma S, Solti A, Rosemann M, Schulte S, Singh MP, Slaats T, Staples M, Weber B, Weidlich M, Weske M, Xu X, Zhu L (2018) Blockchains for business process management – challenges and opportunities. ACM Trans Manag Inf Syst 9(1):4:1–4:16.

[3] Xian Rong Zheng & Yang Lu (2021) Blockchain technology – recent research and future trend, Enterprise Information Systems, DOI: 10.1080/17517575.2021.1939895.

[4] Di Ciccio, C; Cecconi, A; Dumas, M; García-Bañuelos, L;Lopez-Pintado, O; Lu,Q; Ponomarev,A; Trans,A; Weber,I (2019)Blockchain Support for Collaborative Business Processes Informatik Spektrum Vol.42 .

[5] Beck,R; Muller-Bloch,C; King JL (2018) Governance in the blockchain economy: a framework and research agenda. J. Assoc Inf. Sust., 19 (10).

[6] Eenmaa-Dimitrieva H; Schmidt-Kessen MJ (2019). Creating markets in no-trust environments: the law and economics of Smart contracts. Computer Law& Security Review, Vol