Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Sistema de votación electrónica baseado en blockchain

Estudante: Javier Díaz Santiso
Dirección: Paula Fraga Lamas

A Coruña, xuño de 2021.

A meus pais, polo seu apoio incondicional en todas as etapas da miña vida.

Agradecementos

En primeiro lugar, agradecer a meus pais o seu esforzo e sacrificio, que me fixo chegar ata aquí e ser quen son. A todos os compañeiros e amigos que me apoiaron durante esta etapa académica.

Por último, agradecer á titora deste proxecto, Paula, xa que sen o seu apoio e dedicación este proxecto non sería posible.

Resumo

O inexorable avance de internet e da tecnoloxía está cambiando os nosos costumes e como interactuamos entre nós. A pesar das inumerables innovacións tecnolóxicas da sociedade actual, todavía quedan procesos que empregan mecánicas obsoletas e ineficientes, como é o caso das votacións, que son realizadas maioritariamente mediante votos en papel.

A raíz deste problema xorde este proxecto, co obxectivo de crear unha aplicación de votacións electrónicas que garanta a inmutabilidade e mellore aos actuais sistemas electorais en termos de rendemento e fiabilidade. Para acometer esa premisa, implementarase un sistema de votación electrónica baseado na tecnoloxía blockchain. En concreto empregarase a plataforma Hyperledger Fabric, debido a que permite implementar redes permisionadas e ten gran aceptación nos entornos empresarias, outorgando maior alcance á aplicación. Sobre a rede de Fabric desplegaranse dous smart contracts para garantir o segredo de voto e o anonimato dos votantes.

Adicionalmente implementarase unha interfaz de usuario e un middleware para permitir aos usuarios comunicarse coa rede blockchain. Como complemento a esta aplicación, integraranse módulos de monitorización e de análise de rendemento, coa finalidade de avalar os requerimentos necesarios nos procesos electorais.

Abstract

The inexorable advance of the Internet and technology is changing our habits and the way we interact with each other. Despite the countless technological innovations of today's society, there are still processes that employ obsolete and inefficient mechanics, as is the case of voting, which are mostly carried out by paper ballots.

As a result of this problem, this project, aimed at creating an electronic voting application that guarantees immutability and improves the current electoral systems in terms of performance and reliability. To meet this premise, this project implements an electronic voting system based on blockchain technology.

Specifically, we would use the Hyperledger Fabric platform, on which two smart contracts would be distributed to ensure the security of the vote and the anonymity of voters.

As a complement to this application, monitoring and performance analysis modules will be integrated, in order to evaluate the necessary requirements in electoral processes.

Palabras chave:

- Blockchain
- Smart contract
- Hyperledger
- Orderer
- Peer

Keywords:

- Blockchain
- Smart contract
- Hyperledger
- Orderer
- Peer

Índice Xeral

Índice de Figuras

Capítulo 1

Introdución

A aparición da actual pandemia abocou a unha nova realidade na cal se viron afectados os trámites burocráticos: a nivel de seguridade sanitaria, de procedemento, de xestión de recursos, etc. Así, nos procesos electorais, nos que a dificultade de cumprimento do distanciamento social e as restriccións de movilidade reabren o debate sobre a implementación doutras alternativas máis avanzadas e modernas, como a **votación electrónica**, sistema de sufraxio que emprega unha combinación de procedementos, con compoñentes hardware, software e de rede de comunicacións que permiten automatizar os procesos de identificación do elector, da emisión de votos, de emisión de reportes e de presentación de resultados dun proceso electoral, referendo e outras consultas populares [1].

Existen casos de éxito no tocante á implantación destes sistemas, como ocorreu nas votacións parlamentarias do 3 de marzo de 2019 en Estonia, nas que aproximadamente a metade dos votantes se decantou por empregar esta tecnoloxía. Tras un previo rexistro do cidadán como votante, este queda habilitado para emitir o seu voto a través de Internet, o cal incluirá unha marca dixital que permitirá verificar a súa chegada a tempo e o formato co que chegou ao servidor.

Pola contra, existen numerosos exemplos a nivel global nos que a votación electrónica non acadou os resultados esperados, como o caso de Suíza, no que o proxecto para a dixitalización do proceso electoral viuse interrumpido debido ao gran potencial de manipulación de votos na infraestructura que se ía implantar, ou o caso de Estados Unidos, onde ao longo da súa historia electoral xurdiron escándalos derivados de manipulacións, destacando as últimas eleccións á presidencia, na que o ex presidente Donald Trump denunciou un fraude debido á eliminación de votos, xestionados electrónicamente mediante o sistema desenvolto por *Dominion Voting Systems*.

Estes sucesos, derivados dun hardening incompleto da infraestructura de votación electrónica e da falta de auditabilidade do mesmo, afianzaron a desconfianza do elector nestes sistemas, tanto pola falta de garantías do segredo do voto como pola posibilidade de manipu-

lación do mesmo, xurdindo así unha maior reticencia á implantación desta nova tecnoloxía. No caso particular de España, ademais destes inconvintes engádese o factor da lexislación, debido a que a partir da lei orgánica *LOREG* considérase que estes novos sistemas non garanten a non coacción do voto, a suplantación de identidade ou a compra de votos.

1.1 Obxectivos

Este proxecto cobre a necesidade de implementar unha solución de votación electrónica que outorgue fiabilidade, auditabilidade e inmutabilidade; empregando a tecnoloxía **blockchain**. Adicionalmente, tratarase de integrar mecanismos de validación e verificación das premisas que se acaban de mencionar, coa finalidade de aportar un maior grado de confianza no sistema.

En concreto, tratarase de cubrir as seguintes funcionalidades:

- **Xestión de votacións**, tanto a súa creación como a búsqueda de votacións por id ou unha colección completa.
- **Xestión de votantes**, ao igual que no punto anterior tanto nos aspectos da súa creación como no referente a listar un votante por dni ou todos os votantes rexistrados.
- Proporcionar un mecanismo de **emisión de voto** que garanta a autenticidade do usuario e a integridade do voto durante todo o proceso da votación.
- **Escrutinio** dunha votación.
- Integración de mecanismos de **monitorización** da rede blockchain.
- Integración dun sistema de **análise de rendemento**.

1.2 Estructura da memoria

Para facilitar a comprensión das tecnoloxías implicadas neste proxecto, realizarase unha explicación dos distintos módulos usados no sistema proposto:

- Estudo do estado da arte, explicación dos conceptos básicos das votacións electrónicas e de blockchain.
- Estudo en profundidade de Hyperledger Fabric.
- Deseño das compoñentes principais do sistema.
- Implementación das compoñentes do punto anterior.

- Integración de ferramentas auxiliares de monitorización do sistema e análise de rendemento.
- Explicación do procedemento seguido para a realización do proxecto, pasos a seguir no futuro e conclusións obtidas tras a realización deste proxecto.

Capítulo 2

Estudo do estado da arte

2.1 Tipos de sistemas de votación electrónica

Para poder afondar nunha solución para á problemática actual dos sistemas de votación é preciso profundizar na súa evolución e estado actual, sobre o que se distinguen 4 tipos segundo o seu mecanismo de funcionamento:

- **DRE**(*Direct Recording Electronics*) : Trátase de máquinas ubicadas en localizacións supervisadas non remotas que empregan un dispositivo electrónico para permitir que o votante realice a súa elección. Unha vez capturada a elección do votante, almacénase electrónicamente para a súa posterior transmisión a unha entidade centralizada unha vez finalice a votación para efectuar o escrutinio.
- **OMR**(*Optical Mark Recognition*): Mecanismo que permite a lectura do voto mediante máquinas capacitadas para o escaneo de papeletas electorais para o seu posterior escrutinio nunha entidade central ou en centros de votación.
- **EBPs**(*Electronic Ballots Printers*): Sistema que consiste nun enfoque similar ás máquinas DRE, nas que se emprega un dispositivo electrónico para capturar a elección do votante, pero en lugar de almacearse en memoria xérase un token ou recibo en papel, o cal se empregará para realizar o escrutinio de forma electrónica.
- **IVS**(*Internet Voting System*): Infraestructura que permite efectuar o proceso electoral de forma descentralizada desde calquer dispositivo con conexión a internet. Existen tamén IVS nos que a votación queda restrinxida só a certas ubicacións supervisadas.

2.2 Sistemas de votación electrónica baseados en blockchain

A implementación de sistemas de votación electrónica descentralizados con sistemas **DLT** (Distributed Ledger Technology), en concreto blockchain, é un tema que está adquirindo cer-

ta relevancia nos últimos anos, especialmente no sector público co obxectivo de garantir a seguridade e privacidade dos votantes e minimizar os casos de fraude e ciberataques nos procesos electorais. Así, na literatura atópanse artigos científicos recentes como por exemplo [2, 3, 4, 5, 6, 7, 8]. Zaghloul et al. [3] centranse en atopar solucións blockchain para garantir unha alta escalabilidade (e.g., votacións masivas) e minimizar o custo computacional da solución proposta. Autores como Han et al. [4] investigan sobre como crear un framework blockchain para interactuar con máquinas IoT. No caso de Shahzad et al. [5], os autores centranse en analizar as principais causas de desconfianza nos sistemas electrónicos actuais e revisan as experiencias piloto realizadas en países como Estonia, Irlandia, Suiza ou Noruega. No caso de Takabatake et al. [7], os investigadores propoñen un sistema de votación baseado en Zerocoin para aumentar a privacidade do votante. Song et al. [9] presenta unha implementación de un sistema de votación autocontabilizado sobre Ethereum. No caso de Gao et al. [8] centranse directamente na criptografía do protoloco de votación electrónica, creando unha solución moi sofisticada pensada para resistir ataques cuánticos.

Por outra banda, deben salientarse solucións comerciais representativas como como as plataformas Follow my vote [10], TIVI [11] ou Netvote [12].

Como conclusión deste breve resumo do estado do arte debe mencionarse que tras realizar unha búsqueda exhaustiva, a maioría dos sistemas plantexan o uso de redes de carácter público como Ethereum. Non se encontrou ningún sistema de votación electrónica similar o proposto neste TFG, enfocado tamén pensando en votacións para sistemas empresariais.

2.3 Blockchain

Como se avanzou ao comezo deste documento, a proposta deste traballo para solucionar as deficiencias dos sistemas de votación actuais consiste en desenvolver un sistema **DLT**, e dicir, un sistema de rexistro distribuído de transaccións mantidas por consenso sobre unha rede de nodos descentralizada, nas que se garante a integridade da información mediante hashes [1] criptográficos [13].

En concreto, empregaráse a tecnoloxía **blockchain**, unha estructura de datos distribuída e replicada entre todos os membros da rede [14]. Específicamente, está formada por un conxunto de rexistros agrupados en **bloques** cunha marca de tempo. Cada un destes bloques identificase mediante un hash criptográfico que referencia ao do bloque anterior, creando un vínculo entre bloques que da lugar á denominación desta tecnoloxía, a **cadea de bloques**. Para comprender mellor este mecanismo, representarase de forma gráfica mediante a figura 2.1.

[1] Algoritmo matemático que transforma calquer bloque de datos nunha serie de caracteres de lonxitude fixa.

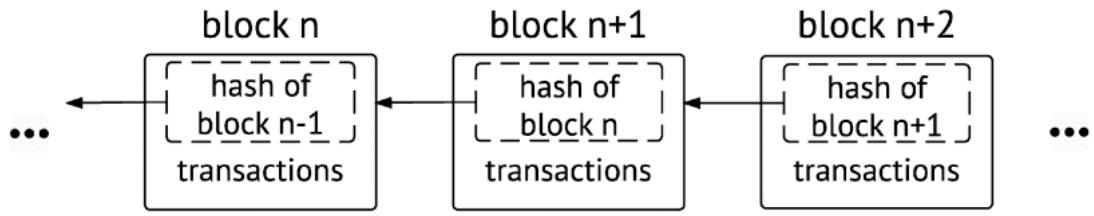

Figura 2.1: Estructura da cadea de bloques de blockchain [14].

2.3.1 Smart contracts

Para definir as lóxicas que gobernan unha rede blockchain empreganse os **smart contracts**, programas simples almacenados na blockchain que se executan ante unhas condicións determinadas, automatizando a execución dun acordo mediante o cal os usuarios teñen a seguridade do resultado dunha operación sen necesidade de intermediarios ou para automatizar un fluxo de traballo [15].

2.3.2 Taxonomía das blockchains

Para poder desenvolver unha solución adecuada, é preciso profundizar na taxonomía das blockchains, distinguíndose dous tipos segundo o criterio de clasificación [16]:

- **Segundo a súa privacidade**: *públicas ou sen permisos* e *privadas ou permisionadas*. Nas primeiras, calquera pode interactuar e formar parte da rede, tanto como un nodo como un mineiro/validador. Pola contra, cando se trata de blockchains privadas os propietarios restrinxen o seu uso, tanto na decisión dos usuarios que poden interactuar na rede como na elección dos nodos validadores.
- **Sgundo o uso de tokens:** *Baseadas en tokens*, as cales empregan un token como un incentivo na rede ou como representación dun activo, e *token-agnósticas*, blockchains que non empregan tokens para garantir o seu funcionamento.

As infraestructuras blockchain máis populares son as de caracter público con uso de token, destacando *Bitcoin, Litecoin e Dash* no tocante ás orientadas a transaccións e *Ethereum, Cardano e Tron* no ámbito das orientadas a procesos.

2.3.3 Algoritmos de consenso

Debido ao caracter descentralizado da tecnoloxía blockchain, no que a cadea de bloques está replicada en cada nodo da rede, resulta imprescindible definir un mecanismo que estatúa os criterios para determinar que información se indexa á blockchain. Os máis populares son os seguintes [17]:

- **Proof of Work(PoW)**: Mecanismo no que os nodos que queiran participar no consenso, chamados mineiros, deben verificar as transaccións que formarán parte do novo bloque e calcular a cabeceira asociada a este, para o que deben resolver un problema computacional. En concreto, compiten entre eles para resolver este cálculo. O primeiro en finalizar obtén unha recompensa e comunica o resultado ao resto de nodos, que verifican o cálculo e a correcta creación do bloque.

- **Proof of Stake(PoS)**: Este algoritmo basease na participación dentro da rede. Os nodos participantes no consenso, chamados validadores, almacenan unha cantidade determinada de tokens. Canto maior sexa esta cuantía, máis posibilidades de ser elexido para validar e realizar a creación do bloque.

- **Practical Byzantine Fault Tolerance(PBFT)**: Para esta estratexia de consenso, un nodo, chamado cliente, envía unha proposta de bloque a outra nodo, denominado como primario, que propaga a proposta a múltiples nodos, os backups. Cada un destes nodos verifica se a proposta é válida, e en caso de acadar un número determinado de validacións afirmativas o bloque enlázase á blockchain.

2.4 Hyperledger

Cabe destacar tamén o caso de **Hyperledger**, unha plataforma de blockchain de código aberto empregada no desenvolemento de solucións privadas, permisionadas e sen necesidade de implementación dun token, con especial empregabilidade no eido empresarial, que será a infraestructura a empregar para desenvolver a solución proposta [18].

2.4.1 Frameworks

Esta plataforma abrangue numerosos frameworks para a implementación da devandita tecnoloxía como se pode ver na figura 2.2:

- **Hyperledger Fabric**: Trátase dunha plataforma de desenvolvemento de solucións empresariais mediante a tecnoloxía DLT, estructuradas mediante unha arquitectura modular e permisionada. Entre as súas características destaca a súa ampla flexibilidade, tanto no tocante á **gobernanza da rede**, permitindo a súa xestión mediante diversos algoritmos de consenso, como a nivel de **código**, debido ao soporte de smart contracts en múltiples linguaxes (Java, Go, Solidity...)[20].

- **Hyperledger Sawtooth**: Comprende a infraestructura impulsada por *Intel* para a construción de redes blockchain privadas baseadas en Ethereum, polo que permite despregar

Figura 2.2: Frameworks Hyperledger [19]

smart contracts implementados en Solidity na EVM(*Ethereum Virtual Machine*), habilitando unha sinxela migración dende Ethereum a Hyperledger. Entre as súas características destaca a súa modularidade, abstraendo a implementación dos contratos intelixentes do resto do sistema [21].

- **Hyperledger Iroha**: Consiste nun conxunto de librerías implementadas en C++ dispoñibles para múltiples linguaxes como Java, Python e Javascript, que constan de comandos pre-definidos, queries e permisos predeterminados coa finalidade de complementar e simplificar o desenvolvemento de aplicacións móbiles e de escritorio que interactúen con sistemas creados a partir de *Hyperledger Fabric* e *Hyperledger Sawtooth* [22].

- **Hyperledger Indy**: Comprende o conxunto de ferramentas e librerías empregadas para prover identidades dixitais arraigadas en blockchains ou outro tipo de DLT para a súa interoperabilidade entre dominios administrativos, aplicacións e outros sistemas [23].

- **Hyperledger Besu**: A última incorporación aos frameworks con estado activo. Correspóndese cun cliente Ethereum baixo a licencia de Apache 2.0, cunha implementación de código aberto en Java. Opera tanto en redes públicas como privadas como testnets de Ethereum como *Rinkeby*, *Ropsten* ou *Göril*. Cabe destacar tamén a súa flexibilidade na implementación de algoritmos de consenso, permitindo tanto Proof or Work como Proof of Authority.

- **Hyperledger Burrow**: Consiste nun framework, que ao igual que *Sawtooth* permite despregar smart contracts na EVM. Ao contrario que os anteriores proxectos mencio-

nados, este está en estado de incubación, polo que todavía non aporta todas as funcionalidades esperadas [24].

- **Hyperledger Grid**: Trátase dunha plataforma empregada para o desenvolvemento de solucións baseadas en DLTs para as cadeas de suministro. Para realizar este cometido prové smart contracts de código aberto que aplican as boas prácticas propias do SCM(*Supply Chain Management*) e librerías para a implementación das súas correspondentes interfaces cliente [25].

2.4.2 Ferramentas adicionais

Como soporte adicional a estas utilidades, Hyperledger aporta unha serie de ferramentas en estado de incubación, deseñadas para ser independentes tanto da implementación do framework como do tipo de rede que empregan:

- **Hyperledger Caliper**: Comprende un conxunto de instrumentos de benchmarking empregados para medir o rendemento dunha implementación de blockchain, analizando as transaccións por segundo, a súa latencia, o uso de recursos, etc. [26]

- **Hyperledger Cello**: Mecanismo de despregue e xestión de blockchains de modo automático e eficiente sobre diversas infraestructuras, como Bare Metal, plataformas de máquinas virtuais ou clústers de contedores como Docker Swarm ou Kubernetes [27].

- **Hyperledger Composer**: Trátase dunha ferramenta de modelado de redes de negocio e implementación de aplicacións baseadas en blockchain. Dende 2019 as súas utilidades están obsoletas [28].

- **Hyperledger Explorer**: Consiste nunha aplicación web orientada á visualización de bloques, transaccións, información da rede e calquer outra información do sistema blockchain asociado [29].

- **Hyperledger Quilt**: Implementación en Java de *Interledger*, un conxunto de protocolos empregados para garantir interoperabilidade de pagos, tanto en moedas fiduciarias como criptográficas [30].

- **Hyperledger Ursa**: Ferramenta que abrangue un conxunto de librerías criptográficas compartidas, cuxa finalidade é evitar a duplicidade neste tipo de proxectos e aumentar a seguridade dos mesmos no proceso [31].

2.5 Hyperledger Fabric en profundidade

Para a realización deste traballo empregarase **Hyperledger Fabric**, polo que resulta imprescindible afondar nas particularidades desta plataforma DLT. A súa orixe radica na natureza dos contextos empresariais, nos que as necesidades de privacidade e de entornos permisionados non son garantidas polas blockchains públicas.

A raíz destas premisas, consolídanse os principais fundamentos desta plataforma, o **non anonimato** das entidades participantes, coa finalidade de garantir a autoridade dos datos, que as redes sexan **permisionadas**, outorgando aos membros da rede control sobre os posibles participantes da mesma, e un **alto rendemento** das transaccións, permitindo ademais a posibilidade de realizalas de forma **privada e confidencial**, garantindo a integridade de datos sensibles das entidades implicadas.

2.5.1 Certificación da autoría

Debido ao amplo abano de requerimentos dos entornos empresariais, Fabric está deseñado mediante unha estructura altamente modular e configurable, pero con certos aspectos básicos inherentes a todos os casos prácticos.

Para cumprir co non anonimato dos participantes da rede, todos (entidades, nodos, aplicacións...) contan con un **certificado X.509**, formato estándar para certificados de clave pública, documentos dixitais que asocian de forma segura pares de chaves criptográficas con identidades como sitios web, individuos ou organizacións [32]. Para levar a cabo este requerimento, a plataforma implementa unha compoñente para abstraer os mecanismos e protocolos criptográficos precisos para a emisión e validación dos certificados, e da autenticación dos usuarios, o **Membership Service Provider (MSP)** [33]. Dado que cada organización trátase dunha entidade legal dentro da rede, poden emitir os seus propios certificados para cada participante da mesma. Esta estructura vese reflexada na figura 2.3 .

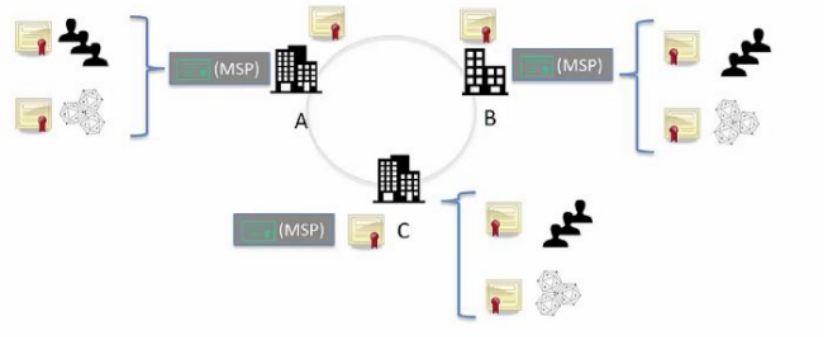

Figura 2.3: Estructura de MSPs dentro dunha rede blockchain de Hyperledger Fabric [34].

2.5.2 Compoñentes e tipos de nodos

Para comprender o funcionamento da rede blockchain de Fabric, é preciso coñecer os elementos que a conforman e como interactúan entre si.

As unidades elementais de toda rede blockchain de Fabric son o **ledger**, estructura que consta de dúas partes, a encargada de almacenar todas as transaccións dentro da rede, o **transaction log** e a encargada de rexistrar o estado das representación dos elementos do mundo real denominados assets, o **worldState**. Para definir os assets e interactúar co ledger cómpre programar a lóxica das transaccións, o que se coñece como **smart contracts**, os cales serán agrupados en **chaincodes**, estructuras empregadas para despregar grupos de smart contracts sobre unha rede blockchain para habilitar as súas funcionalidades [35].

Un aspecto fundamental é o mecanismo que emprega Hyperledger Fabric para establcer as comunicacións dentro da rede, a **canle**. Trátase dunha subrede privada que habilita a comunicación entre membros específicos da rede para a execución de transaccións confidenciais e privadas. Todas as transaccións executadas na rede blockchain realízanse sobre unha canle, na cal todos os integrantes deben estar auntenticados, motivo polo que se emprega o protocolo MSP.

A diferenza doutras redes blockchain, nas que todos os nodos son idénticos e realizan as mesmas funcións, en Fabric distínguense 2 tipos distintos, cada un cun papel a desenvolver dentro do fluxo das transaccións:

- **Orderers:** Son os nodos encargados de realizar a comunicación e mantemento da consistencia dos nodos dentro da rede mediante mecanismos de consenso e asegurando a orde das transaccións. Para acometer esta tarefa, reciben as transaccións, constrúen o bloque a partir destas e garantizan o envío de forma atómica aos *Anchor peers* das organizacións da rede. Dependendo do tipo de implementación existen 3 tipos de ordering service [36]:
  - *Solo:* Variante que consiste no uso dun só orderer. Trátase dunha implementación empregada en fases de desenvolvemento, pouco recomendable en produción debido á pouca tolerancia a fallos que aporta.
  - *Kafka*: Implementación de orderers nun cluster, dando lugar a maior concorrencia, escalabilidade e tolerancia a fallos. En concreto, basease nunha implementación do protocolo *Crash Fault Tolerant(CFT)*, mediante o cal é posible acadar consenso aínda que fallen certos nodos da rede. A pesar destas vantaxes ten os inconvenientes de ser dificil de implementar e ser desenvolto por unha empresa externa a Hyperledger.
  - *Raft*: Trátase da versión máis recente de ordering service, na cal se empregan tamén clusters pero mediante un mecanismo máis sinxelo, creado polo propio

equipo de Fabric. Ao igual que no caso de Kafka, basease no protocolo CFT, coa diferencia de que implementa a nivel de canle e que as organizacións poden aportar nodos ao ordering service.

- **Peers:** Trátase dun tipo de nodo que manten unha copia propia do ledger. En función do labor que desempeñan distínguense catro tipos:
  - *Regular peer / Commiting peer*: Consiste no tipo de nodo máis básico, que se encarga de manter o ledger actualizado e de sincronizarse co resto de peers.
  - *Anchor peer*: Son os únicos que son coñecidos por elementos externos á organización, polo que cada unha debe contar con un como mínimo.
  - *Leader peer*: Trátase do nodo encargado de distribuír os bloques ao resto de peers da súa organización.
  - *Endorsing peer*: A función deste tipo de nodos consiste en simular a transacción que recibe dos clientes ou aplicacións pero sen rexistralo no ledger. Si o resultado da simulación é válido, envíaa ao ordering service para que a a xestione e a transmita ao resto da rede. Mediante este mecanismo evitanse ataques e controlanse posibles erros dentro de configuración.

2.5.3 Fluxo de transaccións en Fabric

Unha vez explicados os compoñentes só resta xuntar as pezas do crebacabezas. Para centrarse no únicamenta na interacción dos nodos, asúmese que todos os elementos implicados nesta explicación foron correctamente certificados mediante o protocolo MSP e contan co material criptográfico preciso.

A transacción iníciase cando un cliente/aplicación invoca a funcionalidade dun API que expón as operacións a realizar sobre a blockchain [37]. A continuación, o servidor que expón este API, a partir dos métodos proporcionados por un dos SDK soportados por Hyperledger Fabric crea unha proposta de transacción. Ademais, este SDK emprega o material criptográfico asociado ao elemento que iniciou a transacción para xerar unha firma única sobre a proposta que acaba de formar.

Posteriormente medainte este SDK envíase a proposta a todos os endorsing peers da canle, os cales se encargarán de realizar o proceso coñecido como endorsement, que consiste en:

- Verificar que a proposta de transacción esté **ben formada**.
- Comprobar que **non fose commiteada** no pasado, para evitar os *Replay-attacks*[2].

[2] Ataque que consiste na retransmisión de datos dunha rede coa finalidade de suplantar a identidade dun usuario ou de provocar denegación de servizos

- Corrroborár que a **firma é válida**, mediante o mecanismo MSP.
- Que o remitente da proposta está **autorizado** para realizar esa operación.
- **Simular** o resultado da transacción contra o estado actual do ledger, para posteriormente enviar o resultado de volta ao SDK, asinado coa clave privada do peer.

O seguinte paso será a verificación das firmas e a comprobación das respostas dos peers no servidor mediante o SDK. Dependendo do tipo de función a realizar distínguense dous casos:

Para as operacións de **lectura**, a aplicación só inspeccionará as respostas dos peers e en caso de cumprirse as políticas de endorsement devolverá a a información correspondente ao usuario.

Para as operacións de **escritura**, en caso de cumprirse as políticas de endorsement, transmitirase ao ordering service a proposta de transacción e a resposta asociada dentro dunha ”mensaxe de transacción”, conterá información referente aos datos a escribir[3], a firma dos peers e o id da canle. O ordering service ordenará as transaccións cronolóxicamente e creará bloques de transaccións por canle.

A continuación, o ordering serivce transmitirá os bloques aos peers, que validarán as transaccións, marcandoas como válidas ou inválidas, verificando que se cumpre completamente a politica de endorsement e que o estado do ledger no tocante ás variables afectadas non foi modificado dende que se simularon as transaccións.

Finalmente, cada peer actualizará a súa réplica do ledger coas transaccións válidas, notificando á aplicación que invocou a transacción sobre si foi correctamente adherida á blockchain ou se foi invalidada. Na figura 2.4 podemos ver un diagrama representativo de toda a secuencia descrita.

[3] A documentación da cita desta subsección di lectura/escritura debido a que para as operacións de actualización envíase tamén un id usado para validar que o obxecto a actualizar é o mesmo que o que se leeu durante a simulación da transacción

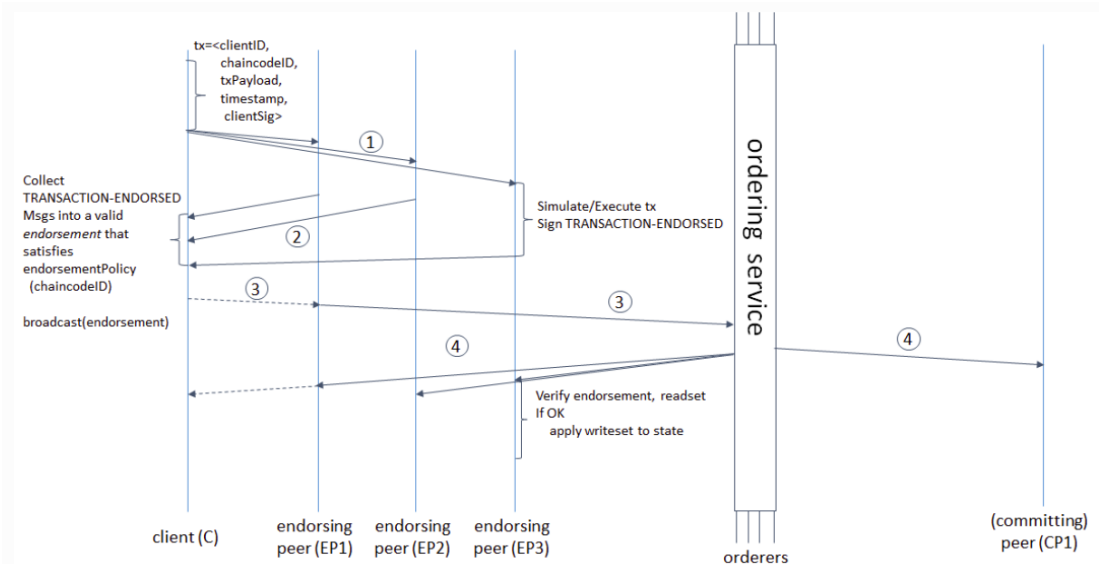

Figura 2.4: Diagrama de fluxo de Hyperledger Fabric [37]

2.5.4 Multiversion concurrency control

Unha das características fundamentais dun sistema de votación electrónico é a capacidade de soportar unha alta carga de votos. Como consecuencia, debe garantir unha concorrencia rápida e eficaz.

Hyperledger Fabric implementa un mecanismo para garantir a coherencia do ledger, o **Multiversion Concurrency Control(MVCC)**. Este sistema consiste en asociar a cada asset do ledger un identificador de versión, polo que cada vez que unha transacción actualiza un elemento, incrementase o seu identificador de versión. Na figura 2.5 podemos ver un exemplo de identificador dunha votación baixo o atributo *"_rev"*, o cal está composto por un contador de versión e un código asociado.

```
"_id": "BALLOT0",
"_rev": "3-67033275eaecfb9c49144e2d7968ecfb",
```

Figura 2.5: Exemplo de identificador dun asset.

Esta funcionalidade foi deseñada para protexer a integridade da información da blockchain fronte ao ataque do dobre gasto, que consiste en realizar múltiples transaccións simultáneas co obxectivo xerar información fraudulenta. O caso máis común é o uso dunha mesma moeda ou token para realizar múltiples pagos.

Como consecuencia deste control de versións, resulta imposible realizar actualizacións concurrentes dun asset, polo que non se podería levar un reconto de votacións mediante variables contador. En capítulos posteriores describiranse as incidencias e explicaranse as alternativas a este impedimento.

2.5.5 Políticas de Hyperledger Fabric

Debido ao caracter descentralizado de Hyperledger Fabric, resulta necesario definir unha serie de normas e restriccións, orixinandose así as **políticas**, conxunto de regras que definen a estructura de toma de decisións e si se alcanzan os resultados requeridos polo consorcio [38], e dicir, estas políticas representan o modo en que os membros da rede se poñen de acordo para aceptar ou rechazar cambios.

Esta característica é un dos principais **elementos diferenciadores** de Hyperledger Fabric con respecto a outras redes como as de **Bitcoin ou Ethereum**, que son rexidas por políticas fixas e só modificables mediante o proceso que goberna o código, ao contrario que en Fabric, que poden ser modificadas sempre que exista acordo entre os membros da rede.

Dentro das políticas de Fabric existe unha xerarquía en función dos aspectos da rede que goberna. Na figura 2.6 podemos observar de modo gráfico esta xerarquía:

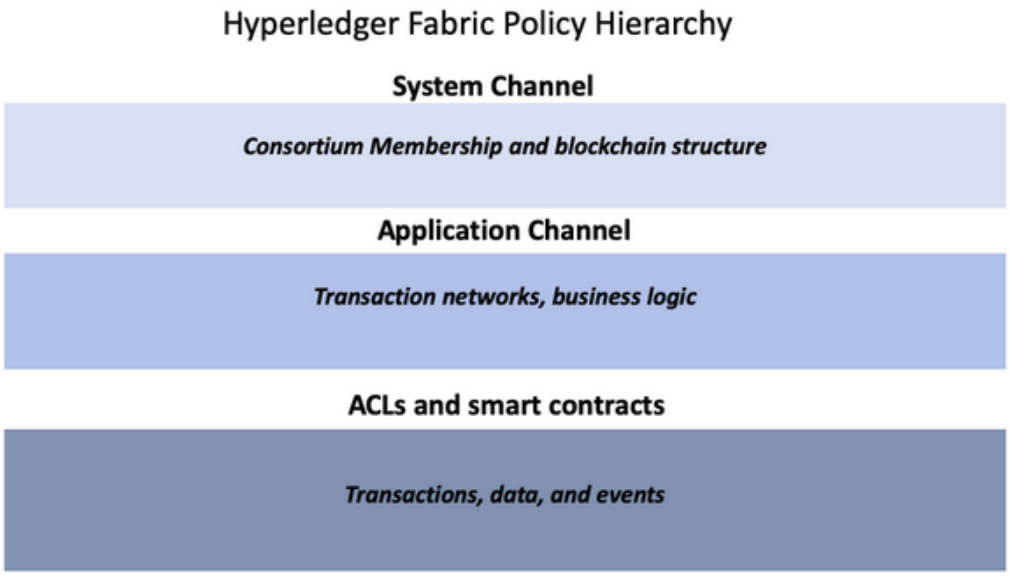

Figura 2.6: Xerarquía das políticas en Hyperledger Fabric [38].

- **System Channel**: As políticas asociadas a este estrato permiten establecer o consenso empregado no ordering service e definir como se crean os novos bloques. Ademais, as políticas a nivel de canle goberna que membros do consorcio poden crear novas canles.

- **Application Channel**: As normas asociadas a este nivel permiten xestionar si se integran ou eliminan membros da canle. Tamñen é neste punto no que se definen as organizacións habilitadas para aprobar un chaincode antes de despregalo.
- **ACLs e smart contracts**: Este estrato correspóndese coa configuración de acceso aos recursos, tanto no referente a funcións da rede como recibir eventos que se producen nesta, como por exemplo notificación de que se xerou un novo bloque.

Na figura 2.7 podemos observar sobre que elementos da rede recae o goberno dos distintos niveles de políticas.

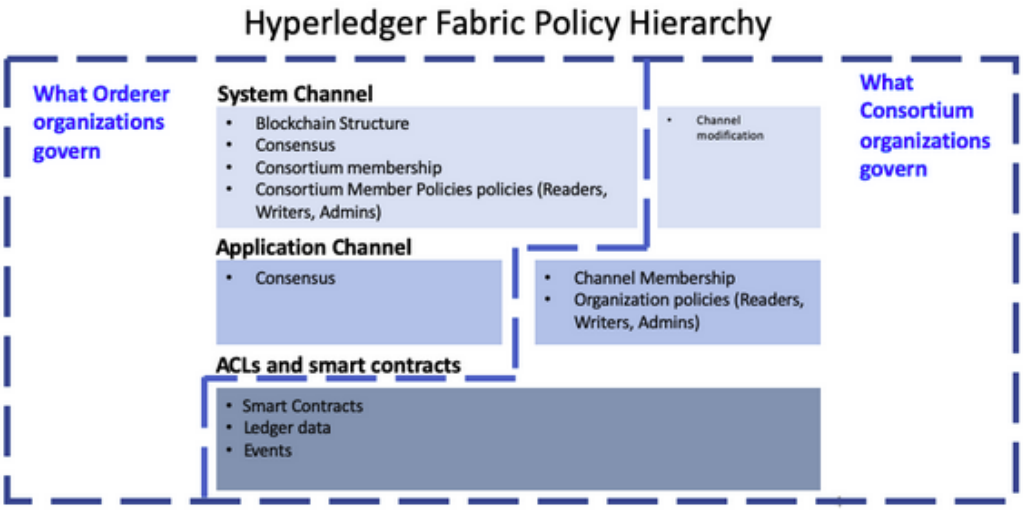

Figura 2.7: Xerarquía das políticas en Hyperledger Fabric [38].

Capítulo 3

Deseño do sistema

A solución proposta neste proxecto consiste no desenvolvemento dun sistema de votación electrónico descentralizado, construído sobre unha rede blockchain. Para habilitar a interacción con esta infraestructura implementarase un API REST, que servirá como ponte de comunicación entre a interfaz de usuario e a rede blockchain.

3.1 Requisitos

Os procesos electorais poden dar lugar a consecuencias de elevada importancia a nivel institucional, polo que un sistema de votación plenamente funcional debe cumprir cunha serie de requisitos reunidos neste capítulo. No caso da solución proposta neste proxecto, ademais de implementar estas premisas, está deseñado co obxectivo de cumprir coa lexislación española a data de realización desde documento no ámbito electoral.

3.1.1 Requisitos funcionais

Coa finalidade de estructurar correctamente os requisitos do sistema definiranse dous roles, **administrador** e **votante**.

No tocante ao primeiro rol, mostrado na figura 3.1, abranguirá as funcionalidades de xestión do sistema, tanto a nivel de **integración** de novos datos mediante a creación de votacións ou o rexistro de novos votantes, como a nivel de **monitorización**, tendo habilitada a visualización de todos os usuarios do sistema e de todas as votacións.

Pola contra, o rol de usuario mostrado na figura 3.2 comprende as competencias de **participación**, mediante a visualización das votacións permitidas, aportación ao proceso electoral a través do seu voto e a posterior revisión dos resultados tras o escrutinio.

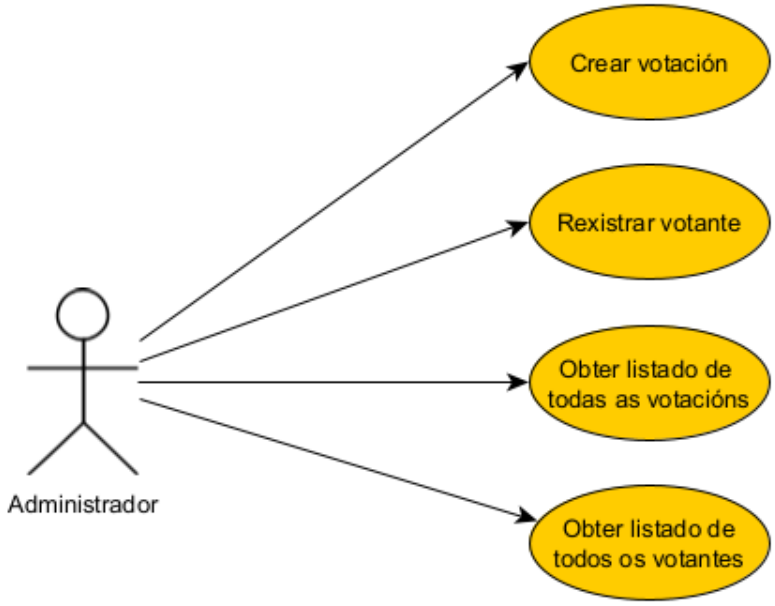

Figura 3.1: Requisitos funcionais do rol administrador.

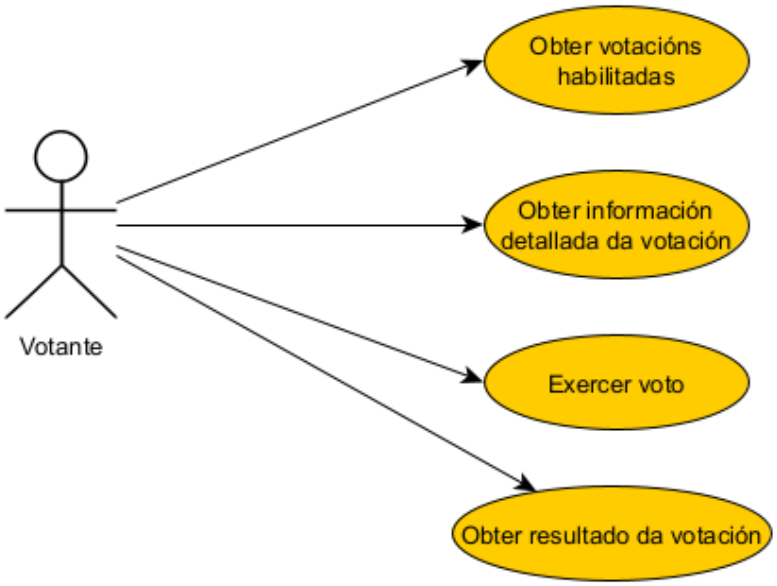

Figura 3.2: Requisitos funcionais do rol votante.

3.1.2 Requisitos non funcionais

Adicionalmente, o sistema debe garantizar unha serie de competencias transversais no ámbito de calidade.

Confidencialidade e anonimato

Trátanse de dúas características de vital importancia dentro deste tipo de sistemas, altamente cohesionadas entre si, que fan referencia á **protección do votante**, dado que avalan o segredo de voto e a custodia da información sensible dos mesmos, precisa para a súa autenticación.

Trazabilidade e inmutabilidade:

Todo proceso sufraxista debe ser transparente e expor de forma clara o traxecto dos votos dende a súa emisión ata a finalización da votación, garantindo así que os resultados non son adulterados. Estas características son inherentes á filosofía de blockchain, a través da cal se almacena a información de forma inalterable grazas á tecnoloxía da cadea de bloques, e se reflexa a trazabilidade dos datos de forma ostensible.

Accesibilidade

Debido ao amplo espectro de público para o que adoitan estar destinados os procesos electorais, o sistema debe habilitar un portal afable e doado para facilitar a interacción dos usuarios coa infraestructura independentemente das súas características.

3.2 Blockchain e smart contracts

Como se avanzou na sección 2.3, existe unha gran diversidade de plataformas que permiten construír solucións mediante a tecnoloxía blockchain. Dado que o obxectivo deste proxecto é deseñar unha aplicación que abarque a maior cantidade de escenarios posibles, a elección empregada será **Hyperledger Fabric**. A principal vantaxe desta plataforma é que permite desenvolver sistemas privados e permisionados, cun alto grado de configuración de mecanismos de consenso, permisionado, arquitectura de rede, etc. Estas características permiten cubrir tanto as necesidades específicas de votación en consorcios empresariais privados como as esixencias lexislativas nos procesos electorais relacionados coas institucións públicas.

No referente á linguaxe de programación dos smart contracts a seleccionar, Fabric soporta Java, Node.js e Go. Todas estas alternativas presentan as mesmas funcionalidades, polo

que debido a que durante as asignaturas cursadas durante grao a única da que non se impartiu formación é Go, optouse por empregalo coa finalidade de incrementar as habilidades de programación.

3.3 Arquitectura

Para acometer de forma estructurada a explicación da infraestructura deseñada, describiranse as distintas capas que a compoñen de forma modular, detallando as funcionalidades que desenvolven os elementos participantes no módulo, as súas características, como interactúan entre sí e a motivación de empregar as tecnoloxías elexidas para a súa integración. A infraestructura constará de 3 elementos principais, unha capa de implementación dos servizos, unha capa de acceso a estes servizos e unha capa de interfaz de usuario.

Adicionalmente, integraranse dous módulos complementarios ao sistema, un explorador de bloques para incrementar a monitorización do sistema e un módulo de benchmarking para realizar análises sobre o rendemento e o alcance do sistema.

3.3.1 Capa modelo

Trátase do módulo do sistema encargado de proporcionar un mecanismo de rexistro da información modelada a partir dos requisitos, habilitar un método de acceso a estes datos e de implementar os servizos requeridos polos usuarios do sistema. Polo tanto, dentro desta capa podemos diferenciar dúas categorías, o **acceso a datos** e a **lóxica de negocio**.

Capa de acceso a datos

Como se adiantou anteriormente, Fabric será a plataforma blockchain a empregar, a cal soporta como bases de datos para os nodos peer as bases de datos **CouchDB** e **LevelDB** [39]. A principal diferenza entre ambas reside en que *CouchDB* permite almacear a información en formato JSON, dando lugar á posibilidade de realizar consultas máis complexas que no caso de *LevelDB*, que almacena a información en pares chave-valor. Ademais, *CouchDB* permite desplegar índices sobre o modelo de datos implicado nos smart contracts, dando lugar á consultas máis eficientes. Dada a gran diversidade de necesidades que poderían ser requeridas e por motivos de eficiencia, optouse por empregar **CouchDB**.

Capa de lóxica de negocio

A implementación dos requisitos mencionados anteriormente na sección 3.1.1, realizarase mediante unha rede descentralizada de cadea de bloques, mediante a plataforma de Hyperledger Fabric sobre smart contracts en Go. Para acometer este obxectivo, implementaranse os

servizos mediante dous smart contracts, **Voter** e **Ballot**. O primeiro comprende as funcións de rexistro de votantes, da súa visualización e de habilitalos para participar en votacións. O segundo engloba as capacidades de creación de votacións, de permitir a súa visualización, levar a cabo o proceso de votación e de realizar o escrutinio.

A decisión de implementar a lóxica de negocio mediante dous smart contracts en lugar de empregar só un radica na necesidade de realizar a votación mantendo o anonimato do votante e o segredo de voto. No capítulo de implementación, sección 4.1.3, profundizarase nos detalles técnicos que se levan a cabo para acadar esta meta.

Como se pode ver na figura 3.3, os participantes desta rede serán 3 organizacións ficticias, as cales contarán cada unha con un peer. Cada un destes nodos contará cunha réplica do ledger e dos smart contracts, a partir dos cales encargarán de simular as transaccións para comprobar o seu correcto funcionamento e de actualizar o estado do ledger en caso de cumprirse os requisitos de consenso da rede.

Adicionalmente, a infraestructura contará cun ordering service composto por 3 nodos orderer independientes a estas organizacións, encargados de asegurar o cumprimento do mecanismo de consenso establecido e de transmitir aos peers as transaccións a escribir no ledger mediante bloques.

Finalmente, para avalar a autenticidade dos nodos empregaranse 4 autoridades certificadoras. Cada organización aportará unha destas CAs para validar a autenticidade dos nodos asociados á súa organización mediante a emisión dos certificados *X.509* e levar a cabo o mecanismo MSP, explicado no capítulo anterior 2.5.1. A autoridade certificadora restante simulará unha entidade independente elexida por consenso entre as organizacións implicadas, a cal será a encargada de emitir os certificados dos nodos participantes no ordering service.

A partir destas autoridades certificadoras, no proceso de despregue da rede xeraráse un usuario administrador por cada unha das organizacións. Cada un deles terá asociados unhas credenciais e un certificado X.509 xerado mediante a librería de OpenSSL, factor clave para a posterior autenticación de usuarios.

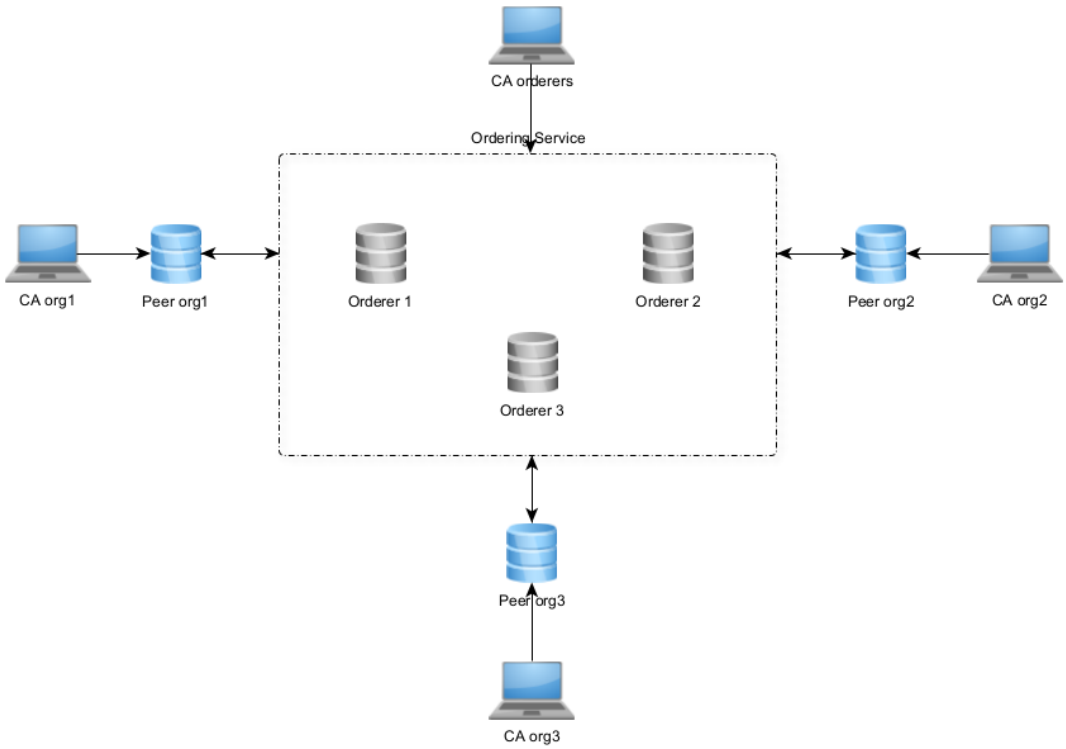

Figura 3.3: Arquitectura da rede blockchain.

3.3.2 Capa de acceso aos servizos

Tras deseñar e implementar o núcleo do sistema, o seguinte paso é habilitar un mecanismo que permita interactuar aos usuarios con esta infraestructura. Para realizar o acceso a estes servizos, exporase un API REST mediante Node.js, debido a que conta cun SDK[1] de Hyperledger Fabric para a comunicación coas APIs das autoridades certidicadoras da propia plataforma, *FabricCAClient*, e outro SDK que permite interactuar cos smart contracts despregados sobre a blockchain, *fabric-network*. Debido a que conta con estas características, será nesta capa na que se verificará a autenticación e a autorización no sistema.

API REST

O obxectivo de implementar o acceso ao servizo mediante unha API REST radica na necesidade de abstraer da complexidade tecnolóxica de blockchain aos programadores de capas superiores do sistema, creando unha interfaz programática accesible de forma remota mediante HTTP.

Este tipo de arquitecturas seguen catro principios fundamentais:

- Estructura cliente/servidor mediante peticións HTTP.
- Recursos direccionables únicamente a través da sua URI.
- Interfaz uniforme, baseada en recursos, transmitidos en formato entendible tanto por máquinas como por persoas. No caso do sistema a desenvolver nestro proxecto será JSON, debido a que é o formato en que almacenará CouchDB a información dos recursos.
- Peticións HTTP seguindo o protocolo RFC2616 [41]. En concreto, aplicaronse as referentes ás operacións CRUD, tratadas nas seccións do rfc 9.3, 9.5, 9.6, 9.7:
  - POST, para a creación dun determinado recurso.
  - GET, para a obtención da representación dun recurso.
  - PUT, para modificar a representación dun recurso existente.
  - DELETE, para eliminar un recurso existente.

Autenticación e autorización

Os sistemas de votación adoitan ser entornos nos que a seguridade é un aspecto crítico, polo que resulta imprescindible que o sistema conte con un robusto sistema de **autenticación** e **autorización**.

[1] Kit de ferramentas proporcionados por un fabricante para desenvolver aplicacións na súa plataforma [40].

Respecto ao primeiro, realizarase mediante a compoñente de Hyperledger Fabric MSP e os certificados X.509 (definido na sección 2.5.1), proceso que debido ao amplo abano de configuracións que brinda Hyperledger será tratado en detalle no capítulo de implementación.

No tocante á autorización, integraránse no sistema **JSON Web Tokens (JWT)**, estándar de código aberto para a xeración de tokens de acceso proposto IETF no documento RFC5719 [42]. A creación deste tipo de token xérase mediante 3 bloques principais:

- Algoritmo e tipo de token a empregar, no caso deste sistema, **SHA256** e token **JWT**
- Payload da petición, para o cal se empregará o nome e dni do usuario e a organización á que está asociado
- Hash dos dous elementos anteriores codificados en base64 e concatenados co contrasinal do usuario codificado tamén en base64

Na figura 3.4 podemos dun modo gráfico as compoñentes que integran o token.

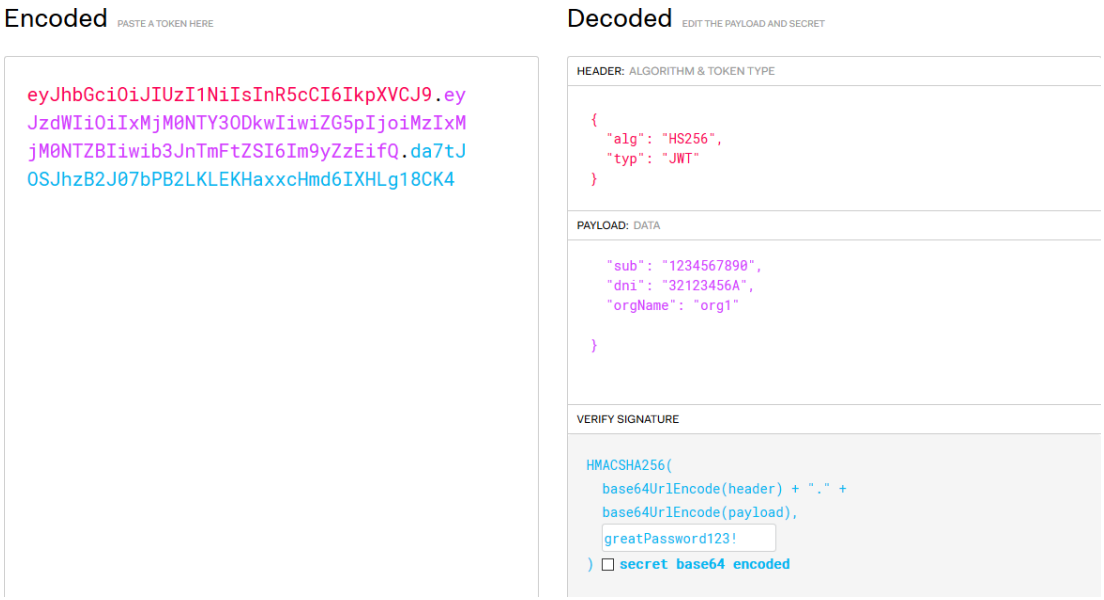

Figura 3.4: Estructura token JWT.

Descentralización do acceso aos servizos

Tras explicar as funcionalidades que exportará esta capa do sistema, só resta profundizar na estructura sobre as que serán soportadas. A diferencia doutras aplicacións tradicionais, nas que existe un único servidor como punto de acceso, o sistema a desenvolver neste proxecto contará con múltiples instancias, xa que integrar un único punto de acceso centralizado rompería coa filosofía de descentralización de blockchain.

3.3.3 Interfaz de Usuario

Tras implementar os requisitos e proporcionar mecanismos para acceder a estes, só resta integrar no sistema os medios para habilitar a interaccion dos usuarios.

Existen unha gran variedade de frameworks que permiten deseñar interfaces de usuario. Neste aspecto o principal requisito do sistema é que sexa unha interfaz sinxela e funcional, tratando de facilitar o uso da aplicación ao amplo espectro de posibles usuarios.

En base a estas premisas, a opción elexida foi a plataforma **Angular**, debido a que permite crear solucións SPA, aplicacións nunha única páxina, que outorgan á interfaz unha elevada fluidez. Ademais, Angular estructura o deseño das interfaces en compoñentes independientes, permitindo crear interfaces en módulos altamente desacoplados entre sí.

Dapp

Ao igual que ocorre co middleware encargado do acceso aos servizos, para manter a filosofía descentralizada de blockchain despregarase a interfaz de usuario en múltiples nodos do sistema. Este tipo de arquitectura na que todos os elementos están aloxados de forma descentralizada coñécese como **Dapp**.

A figura 3.5 reflexa unha visión global do sistema.

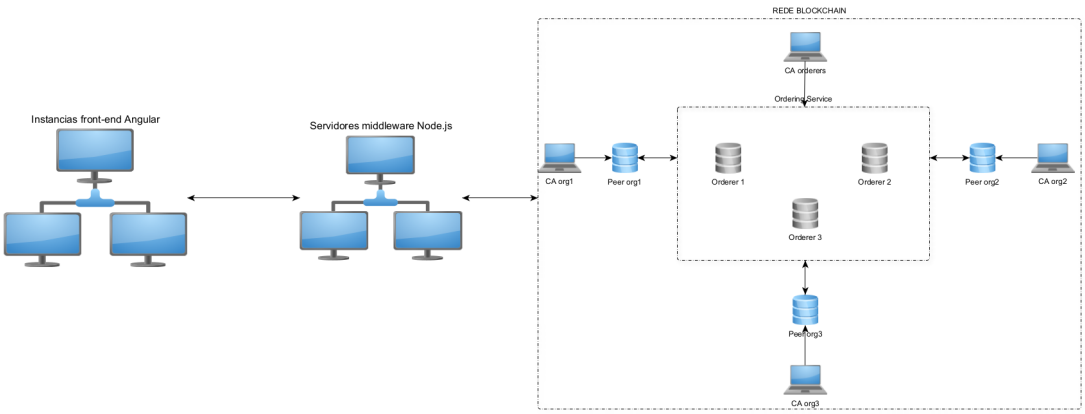

Figura 3.5: Arquitectura xeral do sistema.

3.4 Comunicación do sistema

Unha vez establecidos os requerimentos do sistema, definidos os módulos que o integran coas súas correspondentes funcionalidades e o medio físico sobre o que son soportadas, só resta especificar como interactúan entre si.

Para acometer este obxectivo, organizarase a explicación deste apartado por grupos funcionais coa finalidade de evitar expor información e diagramas redundantes e repetitivos, resaltando en groso os requisitos definidos no apartado 3.1.1 implicados en cada un dos bloques a definir e exemplificando cada un deles mediante un diagrama de secuencia .

3.4.1 Crear votación

O primeiro paso para poder levar a cabo un proceso electoral é definir os participantes, polo que o punto de partida será a funcionalidade de **creación dunha votación** no sistema. Este proceso vese reflexado na figura 3.6.

Un apartado a resaltar é o proceso de validación de usuario, que ademais de verificar as credenciais e o rol de administrador, é realizado no middleware sen necesidade de comunicarse coas autoridades certificadoras. Esto débese a que durante o procedemento de despregue do middleware impórtase o certificado X.509 dos administradores.

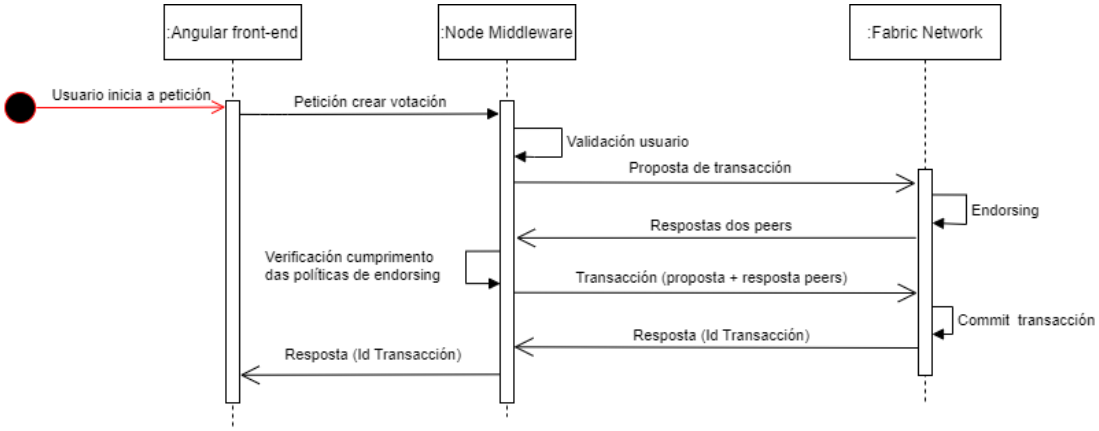

Figura 3.6: Diagrama de secuencia de creación de un elemento.

3.4.2 Rexistro de votantes

O seguinte aspecto preciso para poder levar a cabo unha elección é **rexistrar aos votantes**, proceso exemplificado na figura 3.7. Este proceso resulta semellante ao de creación de votacións, coa diferencia de que neste procedemento, tras a validación do administrador, sí se interactúa coa autoridade certificadora da entidade na que se rexistrará o votante, coa finalidade de crear e importar o certificado X.509 a unha wallet que se asociará a este novo usuario. O motivo desta dinámica e obter un maior rendemento posteriormente dado que non será preciso realizar unha comunicación coas CA's.

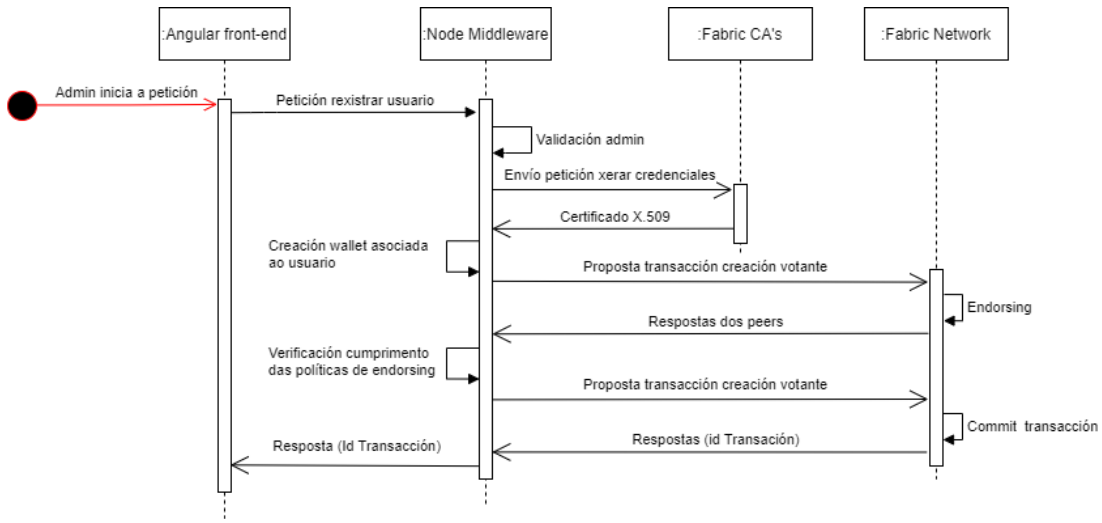

Figura 3.7: Diagrama de secuencia de rexistro de un usuario.

3.4.3 Obter a representación de elementos

Para poder interactuar coas votacións ou xestionar os usuarios, resulta primordial obter unha instancia coas súas características. A figura 3.8 reflexa o ciclo das funcionalidades de obter os **listados de todas as votacións e de todos os votantes**, as **votacións habilitadas** para un usuario en concreto e a de obter **información detallada dunha votación**.

Destacar que este diagrama é valido tanto para as lecturas dun único elemento como de varios debido a que tanto para as consultas individuais como para as coleccións de elementos realizarase unha única consulta, buscando por clave única ou por un rango de claves, acotado a todos os posibles, respectivamente.

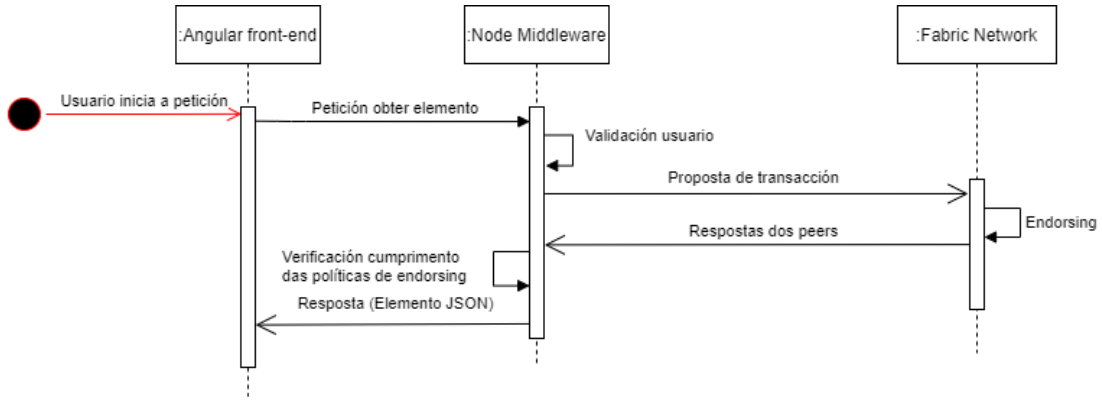

Figura 3.8: Diagrama de secuencia de lectura de entidades do ledger.

3.4.4 Emisión do voto

Tras organizar os elementos implicados na votación e os métodos necesarios para o seu acceso, é fundamental definir o mecanismo para **exercer o voto**. Como se pode observar na figura 3.9, as comprobacións de si existe a o proceso electoral e de se o usuario está habilitado para participar realizanse dentro da propia función de votar no canto de realizarse dende o middleware en pasos separados. O motivo desta decisión técnica radica na premisa de que o proceso de votación debe soportar un elevado número de transaccións simultáneas, condición ante a cal o deseño da figura ten un impacto óptimo en termos de rendemento, debido a que se eleminan dúas comunicacións entre o middle e a rede blockchain por cada petición de emitir un voto.

Un aspecto crucial é o procedemento empregado para almacenar os votos. Ao contrario que noutros sistemas nos que se vai incrementando unha variable que realiza a función de contador de votos, o sistema presentado neste proxecto almacena claves compostas sobre a votación en curso. Esta fórmula, que será tratada en profundidade no apartado de implementación debido a súa elevada importancia, deriva da xestión de control de versións que realiza Hyperledger Fabric, explicado na sección 2.5.4, que impide realizar un elevado número de transaccións concurrentes sobre un elemento de forma eficiente.

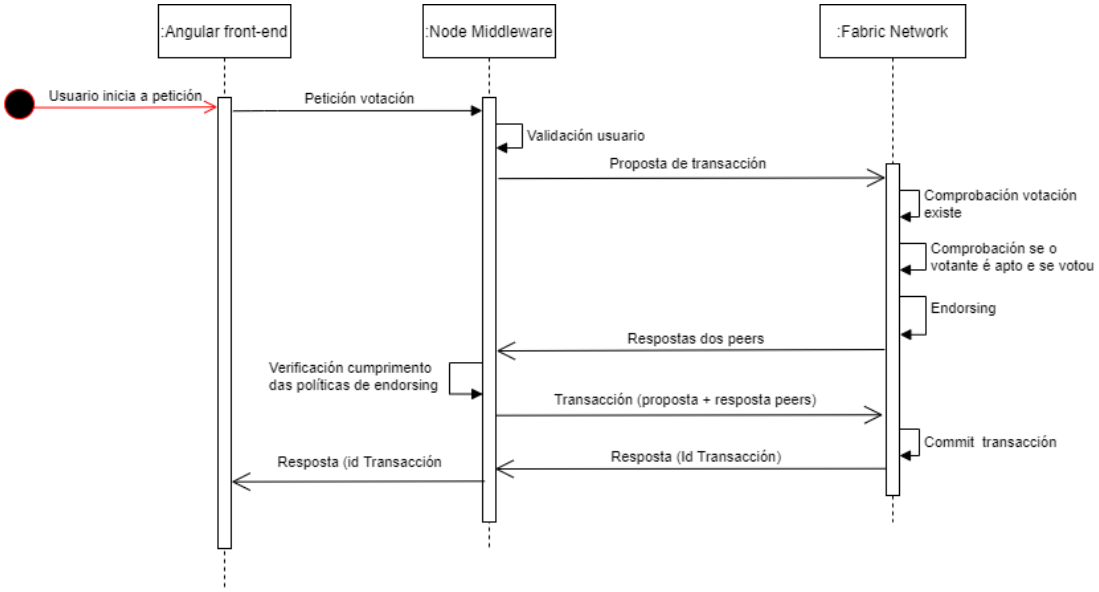

Figura 3.9: Diagrama de secuencia de emisión de un voto.

3.4.5 Visualización de resultados

Finalmente, para concluír un proceso electoral só resta realizar o escrutinio. Derivado do mecanismo de almacenamento de votos, o proceso de **obter os resultados** dunha votación emprega unha metodoloxía adaptada, que consistirá na lectura de todas as claves parciais compostas asociadas a unha votación para posteriormente iterarar sobre elas e sumar os resultados, é dicir, para unha clave composta pola clave da votación + id do candidato a votar + id da transacción de voto, para obter os votos realizariase unha operacion de lectura filtrando pola clave da votación e iterando os votos para cada candidato. Na figura 3.10 podemos observar a comunicación realizada entre os módulos do sistema durante este proceso.

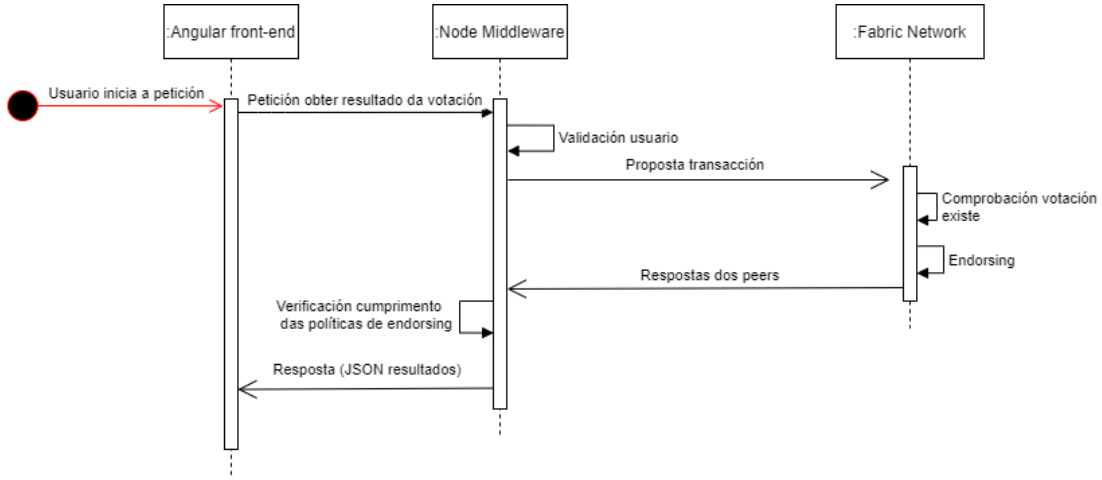

Figura 3.10: Diagrama de secuencia de visualización do escrutinio.

Capítulo 4

Implementación do sistema

O derradeiro paso para materializar o sistema será desenvolver as estructuras deseñadas e funcionalidades definidas sobre cada unha delas. Para dilucidar este proceso, organizarase a explicación seguindo unha metodoloxía modular dende as capas inferiores cara as superiores.

4.1 Back-end - Hyperledger Fabric

Neste estrato do sistema tratarase a creación das **autoridades certificadoras**, a implementación dos **smart contracts** e o proceso de integración destas compoñentes sobre a **rede blockchain** de Hyperledger Fabric, despregada sobre a plataforma Docker.

4.1.1 Autoridades certificadoras

Co obxectivo de facilitar a comprensión deste apartado e evitar información redundante, explicarase o proceso de implementación realizado sobre unha das autoridades certificadoras, dado que o proceso é análogo para as demais.

Para erixir as autoridades certificadoras empregarase a imaxe docker proporcionada por Hyperledger Fabric, *hyperledger/fabric-ca*, que será empregada no ficheiro docker-compose.yaml para realizar o seu despregue en Docker, sobre a que se definirán características básicas como os nomes do servidor e contedor, porto no que se expondrán os servicios da CA, o porto no que se exporá o servicio fora do contedor e o comando para inicializar o servidor.

```
ca_org1:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca.org1.example.com
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_PORT=7054
    ports:
```

```
      - "7054:7054"
    command: sh -c 'fabric-ca-server start -b admin:adminpw -d'
    volumes:
      - ./fabric-ca/org1:/etc/hyperledger/fabric-ca-server
    container_name: ca.org1.example.com
    hostname: ca.org1.example.com
    networks:
      - test
```

Para aplicar esta configuración empregaríase o comando **docker-compose up -d**.

Posteriormente, para que as autoridades certificadoras presenten garantías de autenticidade, xerarase as claves pública-privada e un certificado TLS mediante o comando:

```
fabric-ca-client enroll -u https://admin:adminpw@localhost:7054
    --caname ca.org1.example.com --tls.certfiles
    ${PWD}/fabric-ca/org1/tls-cert.pem
```

A continuación, o seguinte paso sería rexistrar cada entidade da organización na súa correspondente CA, para o cal empregaremos o comando mostrado a continuación, adaptando o tipo de elemento *id.type* co que se corresponda (e.g., peer, admin, etc.) e as credenciais.

```
fabric-ca-client register --caname ca.org1.example.com --id.name
    peer0 --id.secret peer0pw --id.type peer --tls.certfiles
    ${PWD}/fabric-ca/org1/tls-cert.pem
```

O derradeiro paso será xerar o material criptográfico necesario para levar a cabo o proceso de msp para cada un dos elementos rexistrados mediante o comando:

```
fabric-ca-client enroll -u
    https://org1admin:org1adminpw@localhost:7054 --caname
    ca.org1.example.com -M
    ${PWD}/../crypto-config/peerOrganizations/org1.example.com/users/
    Admin@org1.example.com/msp --tls.certfiles
    ${PWD}/fabric-ca/org1/tls-cert.pem
```

Adicionalmente, para os nodos que conforman a rede blockchain crearanse certificados tls, que habilitarán un fluxo de datos encriptado e seguro:

```
fabric-ca-client enroll -u https://peer0:peer0pw@localhost:7054
    --caname ca.org1.example.com -M
    ${PWD}/../crypto-config/peerOrganizations/org1.example.com/
    peers/peer0.org1.example.com/tls --enrollment.profile tls
    --csr.hosts peer0.org1.example.com --csr.hosts localhost
    --tls.certfiles ${PWD}/fabric-ca/org1/tls-cert.pem
```

4.1.2 Rede blockchain

O paso inicial para materializar a cadea de bloques será definir os participantes e os mecanismos de consenso asociados que se deberán cumprir dentro da rede. Para estructurar a explicación, agruparase segundo o tipo de elemento.

Creación de artefactos

- En primeiro lugar, especifícanse as **organizacións participantes**, asignando a cada unha un identificador de MSP e un directorio que contén os materiais criptográficos asociados a cada organización explicados no apartado anterior. Tamén se establecerán as regras que se deberán cumprir dentro da organización, relacionando cada grupo de políticas (e.g., lectura, escritura, administración, etc.) cos tipos de rol habilitados para realizar esa función, os cales se averiguarán mediante o certificado X.509. Neste apartado tamén se elexirá os anchor peers da organización. Na figura 4.1 vese exemplificada a configuración para a organización 1.

```
- &Org1
    Name: Org1MSP
    ID: Org1MSP
    MSPDir: crypto-config/peerOrganizations/org1.example.com/msp
    Policies:
        Readers:
            Type: Signature
            Rule: "OR('Org1MSP.admin', 'Org1MSP.peer', 'Org1MSP.client')"
        Writers:
            Type: Signature
            Rule: "OR('Org1MSP.admin', 'Org1MSP.client')"
        Admins:
            Type: Signature
            Rule: "OR('Org1MSP.admin')"
        Endorsement:
            Type: Signature
            Rule: "OR('Org1MSP.peer')"

    AnchorPeers:
        - Host: peer0.org1.example.com
          Port: 7051
```

Figura 4.1: Configuración das políticas da organización 1.

- Posteriormente definiranse as **capacidades** do sistema, as cales se engloban en tres estamentos: a nivel de **aplicación**, o referente aos **orderers** e as que aplican á **canle**.

  O primeiro comprende unha serie de **ACLs**[1] para axustar que grupos teñen acceso a determinadas funcionalidades. Na figura 4.2 podemos observar un exemplo de como se asocia o rol de escritura á operación encargada de realizar un commit da definición dun

[1] Regras para permitir o acceso a un determinado recurso

chaincode e de como se asocia o rol de lectura á función de lectura da definición dun chaincode.

```
# ACL policy for _lifecycle's "CommitChaincodeDefinition" function
_lifecycle/CommitChaincodeDefinition: /Channel/Application/Writers

# ACL policy for _lifecycle's "QueryChaincodeDefinition" function
_lifecycle/QueryChaincodeDefinition: /Channel/Application/Readers
```

Figura 4.2: Exemplo de definición de ACLs a nivel de aplicación da blockchain.

Neste nivel de definición de capacidades do sistema definirase tamén os requisitos que deben cumprir os diferentes roles. Por exemplo, para executar unha función que requira o rol de *Reader* só é preciso que participe un nodo con ese rol asignado. Sin embargo, para os procesos que requiran o privilexio de *Endorsement*, deberán ser invocados pola maioría de entidades con ese rol 4.3.

```
Organizations:
Policies:
    Readers:
        Type: ImplicitMeta
        Rule: "ANY Readers"
    Writers:
        Type: ImplicitMeta
        Rule: "ANY Writers"
    Admins:
        Type: ImplicitMeta
        Rule: "MAJORITY Admins"
    LifecycleEndorsement:
        Type: ImplicitMeta
        Rule: "MAJORITY Endorsement"
    Endorsement:
        Type: ImplicitMeta
        Rule: "MAJORITY Endorsement"
```

Figura 4.3: Definición dos requisitos de participantes que debe cumprir cada rol.

No referente ás capacidades dos orderers e ás da canle definiranse os perfís que debe cumprir cada rol de forma similar á mostrada na figura 4.3. Cabe destacar que non se definirán os roles de *LifeCycleEndorsement* e *Endorsement*, xa que aplican aos nodos peer, os cales non teñen implicacións a nivel de orderers ou de canle.

Outro aspecto a recalcar é a configuración do batch no apartado de orderers, a cal reflicte o número máximo de transaccións por bloque mediante a variable *MaxMessageCount*, o tamaño máximo e óptimo das transaccións, *AbsoluteMaxBytes* e *PreferredMaxBytes*, respectivamente, e o tempo entre batch *BatchTimeout*. Esta configuración amosase na

figura 4.4.

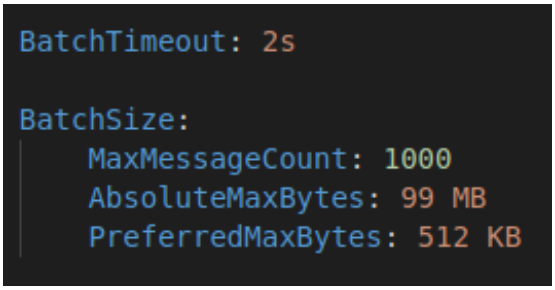

Figura 4.4: Configuración do batch dos orderers.

- Finalmente, estableceranse dous perfís que, aplicando as configuracións que se acaban de definir, serán os encargados da creación do bloque xénese e da configuración da canle.

  O primeiro perfil será *OrdererGenesis*, que como o seu propio nome indica, será o empregado de xerar o bloque xénese. Para acometer este obxectivo asociaraselle a configuración de participantes de orderers, as capacidades definidas para orderers e para a canle que se acaban de explicar e os certificados tls asociados aos nodos orderer xerados na sección das autoridades certificadoras (ver sección 4.1.1).

  No tocante ao segundo, *BasicChannel*, será o encargado de establecer a **configuración da canle** e definir os **anchor peer** de cada organización. Para materializar estas premisas, vincularase este perfil coa configuración dos participantes das distintas organizacións do consorcio e coas capacidades da canle e de aplicación.

  No seguinte fragmento do script empregado para xerar os artefactos da rede, podemos observar como se empregan os distintos perfís para crear os ficheiros *genesis.block*, *mychannel.tx* e *Org1MSPanchors.tx* (xeraranse ficheiros análogos a este último para o resto de organizacións).

```
1 configtxgen -profile OrdererGenesis -configPath . -channelID
      $SYS_CHANNEL  -outputBlock ./genesis.block
2
3 configtxgen -profile BasicChannel -configPath .
      -outputCreateChannelTx ./mychannel.tx -channelID $CHANNEL_NAME
4 configtxgen -profile BasicChannel -configPath .
      -outputAnchorPeersUpdate ./Org1MSPanchors.tx -channelID
      $CHANNEL_NAME -asOrg Org1MSP
```

Despregue da rede

A partir da configuración e os artefactos que se acaban de xerar, definirianse os tres tipos de elemento a despregar:

- O primeiro dos elementos a definir serán os **orderers**, a partir da imaxe docker *hyperledger/fabric-orderer*. Na figura 4.5 reflictese a configuración dun dos orderer, a cal podemos sintetizar na asignación dos certificados tls e na montaxe en volumes do bloque xénese e o material criptográfico asociado a este nodo, ademais de expoñer os portos necesarios para a comunicación entre o servizo do nodo e o exterior do contedor.

```
environment:
  - ORDERER_GENERAL_LOGLEVEL=info
  - FABRIC_LOGGING_SPEC=INFO
  - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
  - ORDERER_GENERAL_GENESISMETHOD=file
  - ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/genesis.block
  - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
  - ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp
  - ORDERER_GENERAL_TLS_ENABLED=true
  - ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/server.key
  - ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/server.crt
  - ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
  - ORDERER_KAFKA_VERBOSE=true
  - ORDERER_GENERAL_CLUSTER_CLIENTCERTIFICATE=/var/hyperledger/orderer/tls/server.crt
  - ORDERER_GENERAL_CLUSTER_CLIENTPRIVATEKEY=/var/hyperledger/orderer/tls/server.key
  - ORDERER_GENERAL_CLUSTER_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
  - ORDERER_METRICS_PROVIDER=prometheus
  - ORDERER_OPERATIONS_LISTENADDRESS=0.0.0.0:8443
  - ORDERER_GENERAL_LISTENPORT=7050
working_dir: /opt/gopath/src/github.com/hyperledger/fabric/orderers
command: orderer
ports:
  - 7050:7050
  - 8443:8443
networks:
  - test
volumes:
  - ./channel/genesis.block:/var/hyperledger/orderer/genesis.block
  - ./channel/crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp:/var/hy
  - ./channel/crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/tls:/var/hy
```

Figura 4.5: Exemplo definición dun orderer.

- Para poder commitear o resultado das transaccións, será preciso crear a base de datos que se asociará a cada peer, a **couchDB**, como se amosa na figura 4.6.

```
couchdb0:
  container_name: couchdb0
  image: couchdb:3.1.1
  environment:
    - COUCHDB_USER=admin
    - COUCHDB_PASSWORD=adminpw
  ports:
    - 5984:5984
  networks:
    - test
```

Figura 4.6: Exemplo de definición da couchDB asociada a un peer.

- Finalmente, o terceiro elemento a definir serán os **peer**. Ao igual que cos orderers, asociarase con cada peer o seu material criptográfico e exporase un porto para comunicar o servizo do peer co exterior do contedor. Adicionalmente, relacionarase cada peer con unha instancia de couchDB 4.7. Unha vez establecidos todos os elementos e os seus parámetros, só resta levantar os contedores mediante o comando **docker-compose up -d**.

```
environment:
  - FABRIC_LOGGING_SPEC=info
  - ORDERER_GENERAL_LOGLEVEL=info
  - CORE_PEER_LOCALMSPID=Org1MSP

  - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=artifacts_test

  - CORE_PEER_ID=peer0.org1.example.com
  - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
  - CORE_PEER_LISTENADDRESS=0.0.0.0:7051
  - CORE_PEER_CHAINCODEADDRESS=peer0.org1.example.com:7052
  - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:7052
  - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:7051
  - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
  - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb0:5984
  - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=admin
  - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=adminpw
  - CORE_METRICS_PROVIDER=prometheus
  - CORE_PEER_TLS_ENABLED=true
  - CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/crypto/peer/tls/server.crt
  - CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/crypto/peer/tls/server.key
  - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/crypto/peer/tls/ca.crt
  - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/crypto/peer/msp
depends_on:
  - couchdb0
ports:
  - 7051:7051
volumes:
  - ./channel/crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/
  - ./channel/crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls:/
  - /var/run/:/host/var/run/
```

Figura 4.7: Exemplo definición peer en docker-compose.yaml .

Comunicación dentro da rede

Para habilitar a comunicación entre os distintos nodos dunha rede de Hyperledger Fabric, resulta esencial establecer unha canle entre eles.

O primeiro paso será **crear a canle**. Mediante o comando mostrado a continuación, o orderer despregado en localhost:7050 fará uso dos seus certificados TLS e da configuración definida na sección anterior no arquivo ${CHANNEL_NAME}.tx para crear a canle e escribir o bloque xénese, o cal estará composto pola información do consorcio implicado na rede e a configuración que goberna esta infraestructura blockchain.

```
peer channel create -o localhost:7050 -c $CHANNEL_NAME \
    --ordererTLSHostnameOverride orderer.example.com \
    -f ./artifacts/channel/${CHANNEL_NAME}.tx --outputBlock
    ./channel-artifacts/${CHANNEL_NAME}.block \
    --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA
```

O seguinte paso para establecer unha contorna de comunicación dentro da rede blockchain, será enlazar os nodos peer ao escenario que se acaba de crear. Para efectuar este mester, para cada peer a vincular coa canle estableceranse en variables as características da CA

asociada á organización á que pertence e a dirección do peer. Posteriormente, mediante o comando *peer channel join* quedará integrado na canle, figura 4.8.

```
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=$PEER0_ORG1_CA
export CORE_PEER_MSPCONFIGPATH=${PWD}/artifacts/channel/crypto-config
/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7051

peer channel join -b ./channel-artifacts/$CHANNEL_NAME.block
```

Figura 4.8: Join dun peer a unha canle.

Por último, só resta informar ao resto de participantes da canle sobre o anchor peer que se acaba de unir á rede. Este proceso levarase a cabo mediante o envío sobre tls da información do anchor peer definida na creación dos artefactos. A continuación móstrase un exemplo da actualización a realizar, o cal sería executado para cada peer que se adheriu á canle, adaptando a variable ${CORE_PEER_LOCALMSPID} á organización do anchor peer vinculado.

```
peer channel update -o localhost:7050 --ordererTLSHostnameOverride
    orderer.example.com -c $CHANNEL_NAME -f
    ./artifacts/channel/${CORE_PEER_LOCALMSPID}anchors.tx --tls
    $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA
```

4.1.3 Implementación dos smart contracts

Garantir o **anonimato** e o **segredo de voto** son os dous aspectos máis críticos dentro dun proceso electoral. Polo tanto, garantir este requisito é un dos obxectivos fundamentais do sistema a desenvolver.

Para levar a cabo este propósto, implementaranse dous smart contracts, un encargado de rexistrar usuarios e habilitalos a participar en procesos electorais,*Voter*, e outro coa finalidade de almacenar votacións e os votos asociados a cada unha delas, *Ballot*. Desta forma, os sistema garante o anonimato dos votantes, dado que só se mostra aos usuarios as votacións e os seus resultados. Ademais, tamén avala o segredo de voto, xa que non se relaciona ao votante co seu voto.

A única relación que existe entre o votante e o voto é que si se ten coñecemento sobre si exerceu ou non o dereito a voto. Este modelo é análogo ao realizado nas votacións tradicionais, nas que os membros da mesa electoral teñen un listado de votantes e marcan aqueles que efecúan o seu voto. Nas figuras 4.9 e 4.10 podemos observar un exemplo dos modelos de datos que empregarán.

```
{
  "_id": "5111111A",
  "_rev": "2-8e72bb47e91faac4b957faa68cf5821f",
  "address": "address5",
  "dni": "5111111A",
  "name": "voter5",
  "phoneNumber": 1234567891,
  "voterBallots": [
    {
      "ballotName": "BALLOT1",
      "hasVoted": false
    }
  ],
  "~version": "CgMBHAA="
}
```


Figura 4.9: Exemplo de votante gardado nunha das couchDB.


```
{
  "_id": "BALLOT11",
  "_rev": "1-7ddb036d3dde346d6244ddc276595f92",
  "ballotName": "First ballot",
  "candidates": [
    {
      "candidateName": "candidate0",
      "idCandidate": 0,
      "leader": "leader0",
      "votes": 0
    },
    {
      "candidateName": "candidate1",
      "idCandidate": 1,
      "leader": "leader1",
      "votes": 0
    }
  ],
  "endTime": 1626390166000,
  "startTime": 1609459200000,
  "~version": "CgMBIQA="
}
```


Figura 4.10: Exemplo de votación almacenada nunha das couchDB.

No referente ao código, non se entrará en detalles específicos (como que se comproben os parámetros de entrada, ou que "x" elemento existe, si o votante efectuou o voto, etc.), se non que se enfatizarán os conceptos clave que caracterizan á rede blockchain de Hyperledger Fabric.

Para materializar a lóxica de negocio dos smart contracts, empregarase a api facilitada por Fabric, *contratapi*. As principais funcionalidades que se empregaran serán os métodos *putState* e *getState* para escribir e leer o estado no ledger dun determinado elemento, respectivamente.

Existe unha funcionalidade do sistema que emprega unha mecánica diferente, o proceso de **votación**. Ao contrario que noutros sistemas nos que se actualiza unha variable constantemente que funciona como un contador, o sistema a desenvolver crea **claves compostas**, mediante a función *CreateCompositeKey* para almacenar os votos. En concreto a formada pola propia definición da clave composta, o id de votación, o id de candidato a votar e o id da transacción do voto. Na figura 4.11 pódese observar como o atributo id mostra mediante a cor verde a definición das compoñentes da clave composta e os valores de cada unha delas.

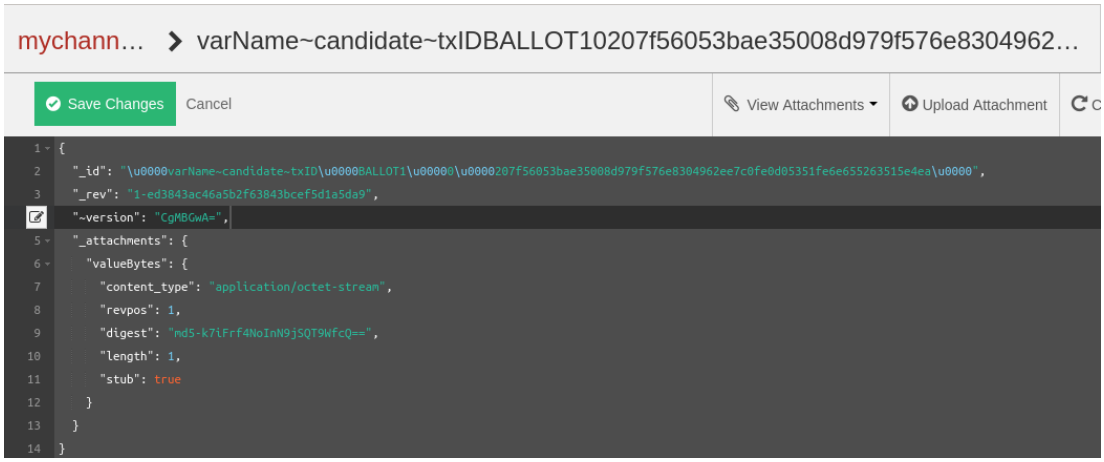

Figura 4.11: Exemplo de voto almacenado como clave composta.

Como consecuencia, para realizar o escrutinio deberanse ler todas as claves compostas asociadas a unha votación. Para acometer esta premisa, a api contractapi proporciona a función *GetStateByPartialCompositeKey*, a cal se empregará buscando por id de votación para obter os votos que ten asociados, para posteriormente agrupalos e sumar o número de votos segundo o id de candidato que forma parte da clave composta.

4.1.4 Despregue dos smart contracts

O último paso para materializar unha rede blockchain plenamente operativa é realizar a integración dos smart contracts. Este proceso consistirá en **empaquetar** cada un dos smart contracts, **instalar** ambos en cada peer, **aprobar** que se realizou correctamente, **commitear**

nos ledgers este cambio e **inicializar** ambos smart contracts.

Para poder realizar o endorsing deberán ser despregados sobre cada peer, polo que para evitar expor información redundante, cando se referencie algunha operación sobre un peer (instalación, verificación ou commit) asumirase que se exportaron as variables de entorno precisas para interactuar con él. Estas serán o id do msp da organización, o directorio co material criptográfico preciso para interactuar con este, os certificados tls do peer e a súa dirección 4.12.

```
setGlobalsForPeer0Org1() {
    export CORE_PEER_LOCALMSPID="Org1MSP"
    export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/artifacts/channel/crypto-config/peerOrganizations/org1.
    export CORE_PEER_MSPCONFIGPATH=${PWD}/artifacts/channel/crypto-config/peerOrganizations/org1.exam
    export CORE_PEER_ADDRESS=localhost:7051
}
```

Figura 4.12: Variables de entorno para interactuar cun peer.

Empaquetado dos contratos intelixentes

Para levar a cabo esta premisa, descargaranse as dependencias dos módulos de go precisas para executar o código para posteriormente empaquetalas xunto co smart contract mediante o comando *peer lyfecycle chaincode package* 4.13.

```
presetup() {
    echo Vendoring Go dependencies ...
    pushd ./artifacts/src/github.com/ballot/go
    GO111MODULE=on go mod vendor
    popd
    echo Finished vendoring Go dependencies
}

CHANNEL_NAME="mychannel"
CC_RUNTIME_LANGUAGE="golang"
VERSION="1"
SEQUENCE="1"
CC_SRC_PATH="./artifacts/src/github.com/ballot/go"
CC_NAME="ballot"

packageChaincode() {
    peer lifecycle chaincode package ${CC_NAME}.tar.gz \
        --path ${CC_SRC_PATH} --lang ${CC_RUNTIME_LANGUAGE} \
        --label ${CC_NAME}_${VERSION}
    echo "===================== Chaincode is packaged ===================== "
}
```

Figura 4.13: Proceso empaquetar o smart contract Ballot.

Instalación do smart contract

Tras crear un paquete asociado a cada smart contract e ás súas correspondentes dependencias, estes serán instalados en cada peer. Para realizar este paso, tras importar as variables de entorno precisas instalaranse mediante o comando:

```
peer lifecycle chaincode install \$\{NOME_SMARTCONTRACT\}.tar.gz
```

Aprobación da instalación

Para poder verificar a instalación, o primeiro paso é listar os smart contracts instalados no peer correspondente mediante o comando *peer lifecycle chaincode queryinstalled*, a partir do cal obteremos o id do paquete do smart contract a despregar, necesario para posteriormente aprobar a instalación. Unha vez obtido, mediante o comando *peer lifecycle chaincode approveformyorg* e as referencias ao peer, certificados tls e metadatos do paquete aprobarase a instalación 4.14.

```
queryInstalled() {
    setGlobalsForPeer0Org1

    peer lifecycle chaincode queryinstalled >&log.txt
    cat log.txt
    PACKAGE_ID=$(sed -n "/${CC_NAME}_${VERSION}/{s/^Package ID: //; s/, Label:.*$//; p;}" log.txt)
    echo PackageID is ${PACKAGE_ID}
    echo "===================== Query installed successful on peer0.org1 on channel ==============
}

approveForMyOrg1() {
    setGlobalsForPeer0Org1

    peer lifecycle chaincode approveformyorg -o localhost:7050 \
        --ordererTLSHostnameOverride orderer.example.com --tls \
        --cafile $ORDERER_CA --channelID $CHANNEL_NAME --name ${CC_NAME} --version ${VERSION} \
        --init-required --package-id ${PACKAGE_ID} \
        --sequence ${SEQUENCE}

    echo "===================== chaincode approved from org 1 ===================== "

}
```

Figura 4.14: Aprobación do smart contract Ballot.

Commit da instalación

Antes de poder confirmar os datos no ledger, será necesario verificar as políticas de endorsement do ciclo de vida da rede, que no caso deste proxecto é que a maioría de peers. Para realizar esta función, empregarase o comando *peer lifecycle chaincode checkcommitreadiness*, asociando os metadatos da canle e o smart contract.

No caso de cumprirse estas políticas, realizarase o commit sobre **todos os peers** da rede 4.15.

```
checkCommitReadyness() {
    setGlobalsForPeer0Org1
    peer lifecycle chaincode checkcommitreadiness \
        --channelID $CHANNEL_NAME --name ${CC_NAME} --version ${VERSION} \
        --sequence ${VERSION} --output json --init-required
    echo "==================== checking commit readyness from org 1 ==================== "
}

commitChaincodeDefination() {
    setGlobalsForPeer0Org1
    peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com
        --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA \
        --channelID $CHANNEL_NAME --name ${CC_NAME} \
        --peerAddresses localhost:7051 --tlsRootCertFiles $PEER0_ORG1_CA \
        --peerAddresses localhost:9051 --tlsRootCertFiles $PEER0_ORG2_CA \
        --peerAddresses localhost:11051 --tlsRootCertFiles $PEER0_ORG3_CA \
        --version ${VERSION} --sequence ${SEQUENCE} --init-required

}
```

Figura 4.15: Commit da instalación do smart contract Ballot.

Inicialización dos smart contracts

Para poder interactuar cos smart contracts instalados, o derradeiro paso será inicializar o smart contract en todos os peers da rede 4.16.

```
chaincodeInvokeInit() {

    peer chaincode invoke -o localhost:7050 \
        --ordererTLSHostnameOverride orderer.example.com \
        --tls $CORE_PEER_TLS_ENABLED --cafile $ORDERER_CA \
        -C $CHANNEL_NAME -n ${CC_NAME} \
        --peerAddresses localhost:7051 --tlsRootCertFiles $PEER0_ORG1_CA \
        --peerAddresses localhost:9051 --tlsRootCertFiles $PEER0_ORG2_CA \
        --peerAddresses localhost:11051 --tlsRootCertFiles $PEER0_ORG3_CA \
        --isInit -c '{"Args":[]}'

}
```

Figura 4.16: Inicialización do smart contract en todos os peers.

4.2 Middleware - Servidores en Node

Para poder interactuar coa rede blockchain, será preciso desenvolver unha infraestructura para habilitar esta comunicación.

Para realizar esta función desenvolverase un servidor en Node mediante o software *Express.js*. Para manter o carácter descentralizado imperante neste sistema, exporanse varias instancias deste servidor.

Consta principalmente de 3 aspectos a destacar, a xestión da **autorización**, verificación da **autenticidade** e a interacción coa **blockchain**.

No tocante á primeira característica, tal e como se tratou no capítulo anterior 4 empregaranse JSON Web Tokens. En concreto, empregaranse as librerías *expressJWT*, para condicionar o uso do token para todas as funcionalidades agás a de login e *jsonwebtoken*, para crear e verificar os tokens.

Respecto á autenticación, empregaranse os certificados X.509. Cando un usuario administrador rexistra un usuario, xérase un certificado X.509, o cal será almacenado nunha wallet xerada a través do SDK de node *fabric-network*, ademais de na propia CA. A finalidade deste mecanismo e mitigar o tráfico de rede necesario para interactuar coa blockchain, dado que cando un usuario acceda á plataforma só será preciso verificar a súa wallet. Na figura 4.17 podemos observar a wallet do usuario *user000*, a cal contén o seu certificado. Para a comunicación contra as CA's empregarase o SDK *fabric-ca-client*.

Figura 4.17: Contido da wallet do usuario user000.

En relación coa comunicación coa rede blockchain, establecerase mediante o SDK *fabric-network*. En primeiro lugar establecerase a conexión coa canle mediante o módulo *Gateway* para obter a referencia da rede e posteriormente a do contrato intelixente requerido, reflexado na figura 4.18).

```
const network = await gateway.getNetwork(channelName);
const contract = network.getContract(chaincodeName);
```

Figura 4.18: Conexión do middle coa reed blockchain.

Posteriormente, empregarase o método *submitTransaction* cos parámetros necesarios para as transaccións de escritura e *evaluateTransaction* para as de lectura.

Como se mencionou no capítulo 4, exporase un API REST para que o usuario poida acceder a estes servizos. Na figura 4.19 podemos observar a petición de crear votación, para a cal se precisan a canle, o smartcontract a invocar, a función requerida e os argumentos precisos para levala a cabo.

Figura 4.19: Chamada a endpoint de creación de votación.

A continuación, exponse na figura 4.20 o resultado desta petición, a cal devolverá o id da transacción.

Figura 4.20: Resposta endpoint de creación de votación.

Finalmente, para contar tamén cun exemplo dunha operación de lectura, representarase sobre a figura 4.21 a petición e a resposta dunha petición de buscar unha votaciñon por id. A resposta será un json cos atributos da entidade de votación.

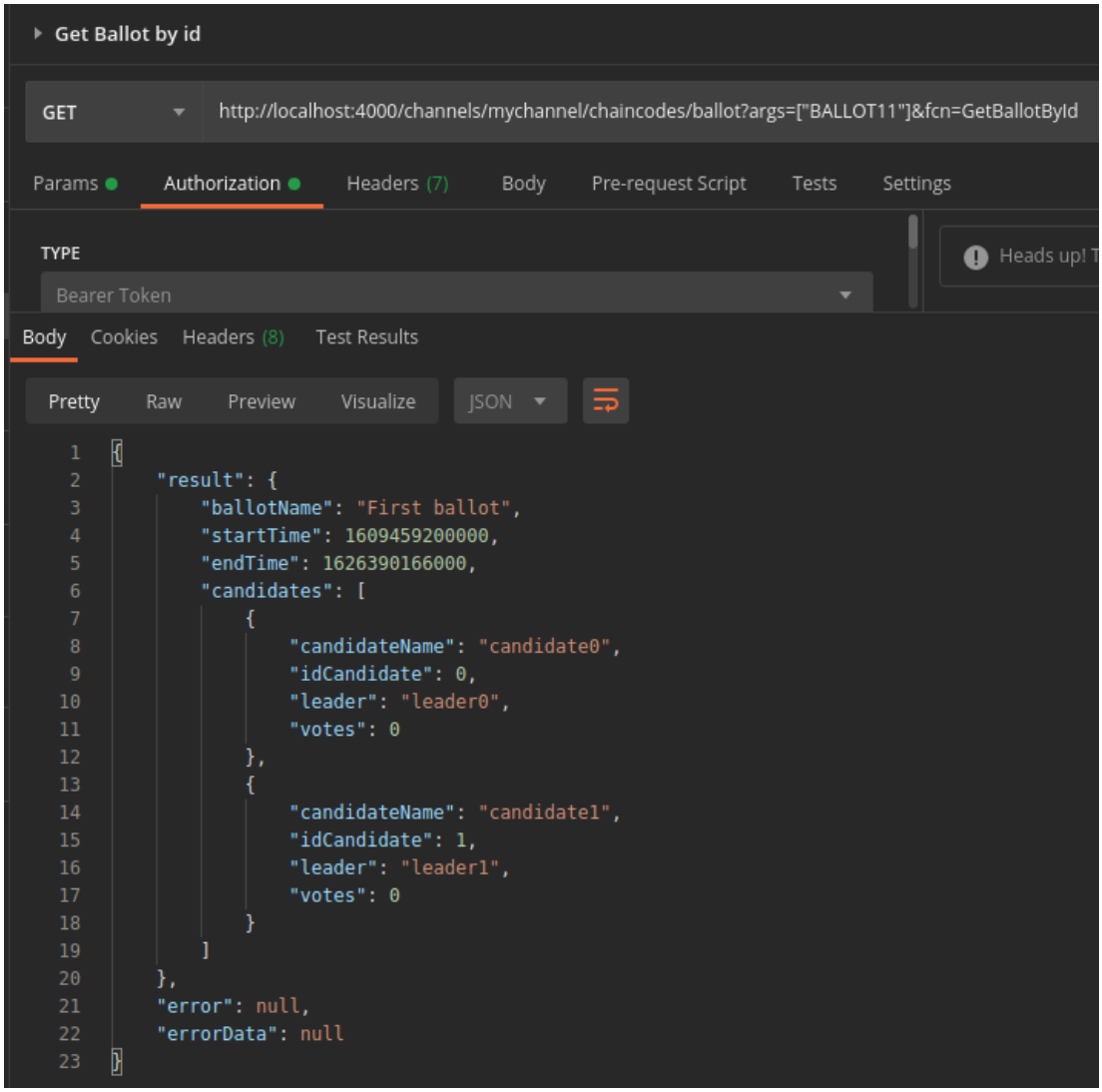

Figura 4.21: Resposta endpoint de creación de votación.

4.3 Front-end - Interfaz de Usuarios

O derradeiro paso para desenvolver un sistema plenamente funcional é dispensar aos usuarios un mecanismo para poder interactúar co sistema. Debido ao amplo espectro poboacional que podería interactúar co sistema, este módulo busca ser o máis sinxelo posible, mantendo unha experiencia de usuario óptima.

A figura 4.22 mostra a primeira vista do sistema, común a todos os usuarios que accedan á web. Mediante esta vista o usuario **iniciará sesión** no sistema. Cabe destacar o funcionamento do header, o cal se modifica de forma dinámica en función do rol do usuario como se

verá ao longo desta sección.

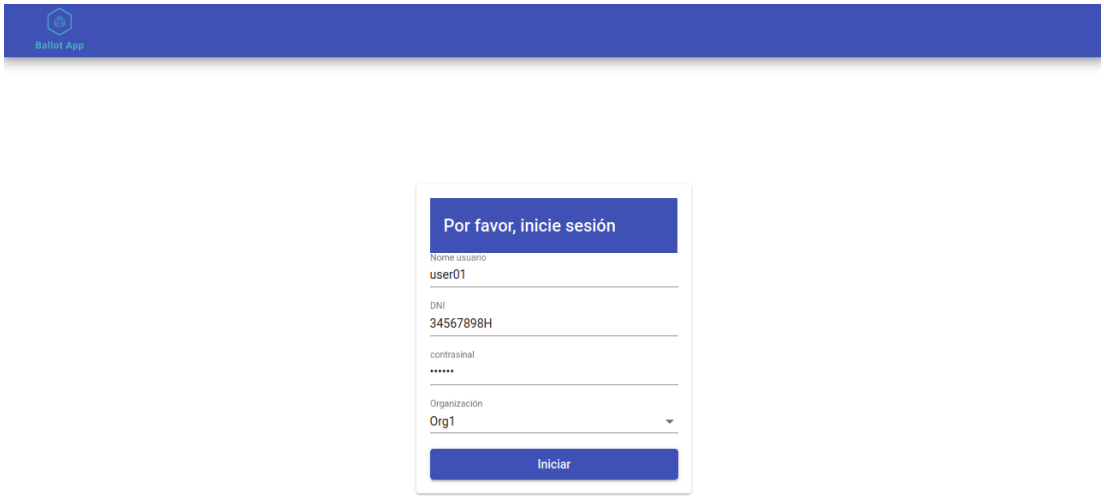

Figura 4.22: Formulario de inicio de sesión no sistema.

A continuación trataranse as vistas do rol **administrador**. A seguinte vista do sistema a tratar é moi semellante á que se acaba de mostrar, debido a que durante a implementación da interfaz de usuario aprobeitouse a alta modularidade que proporcionan as compoñentes de Angular. Na figura 4.23 podemos observar o formulario de **rexistro** de votantes, o cal só se diferencia do de inicio de sesión nos campos requeridos. Tamñen podemos observar como no header se atopan habilitadas as funcionalidades do rol administrador.

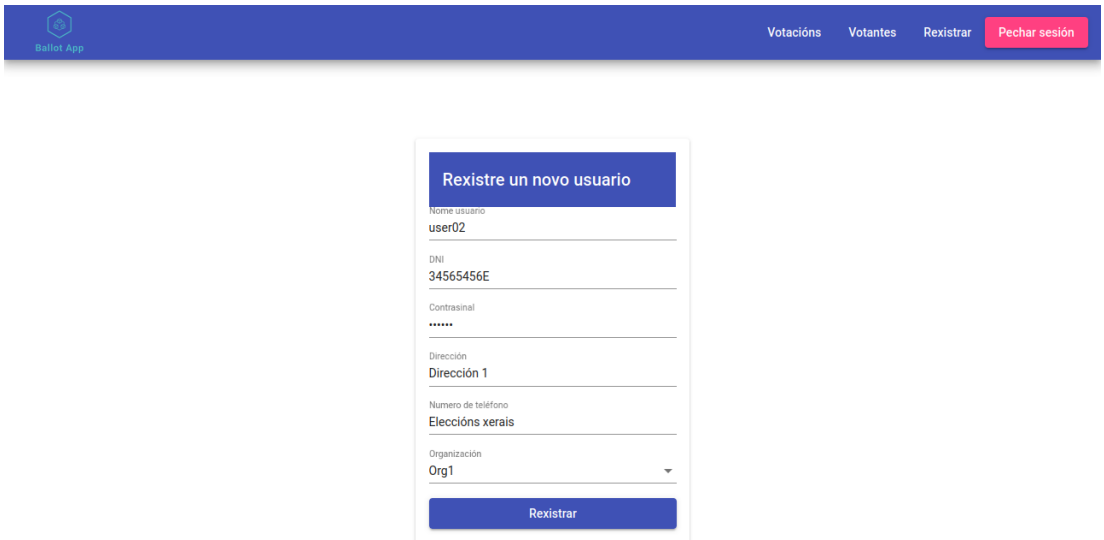

Figura 4.23: Formulario de rexistro dun votante.

As dúas seguintes figuras mostradas 4.24 e 4.25 correspóndense coas vistas de **listar todos**

os usuarios e a de **listar todas as votacións**, respectivamente. Ao igual que ocorría cos formularios, son compoñentes moi semellantes. Cabe destacar o funcionamento do icono de persoas, asociado ás columnas ”Votacións autorizadas” e ”Lista candidatos”. Este icono funciona como un despregable para evitar sobrecargar a vista de información, permitindo visualizar os datos importantes e profundizar en aqueles que se desexe.

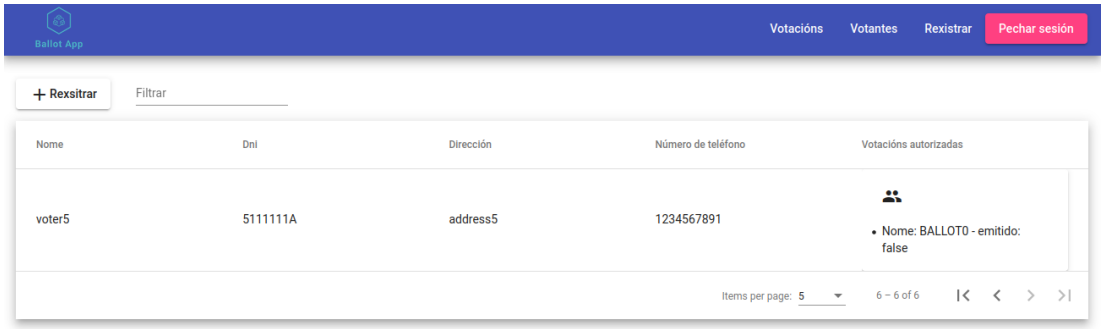

Figura 4.24: Listado de votantes do sistema.

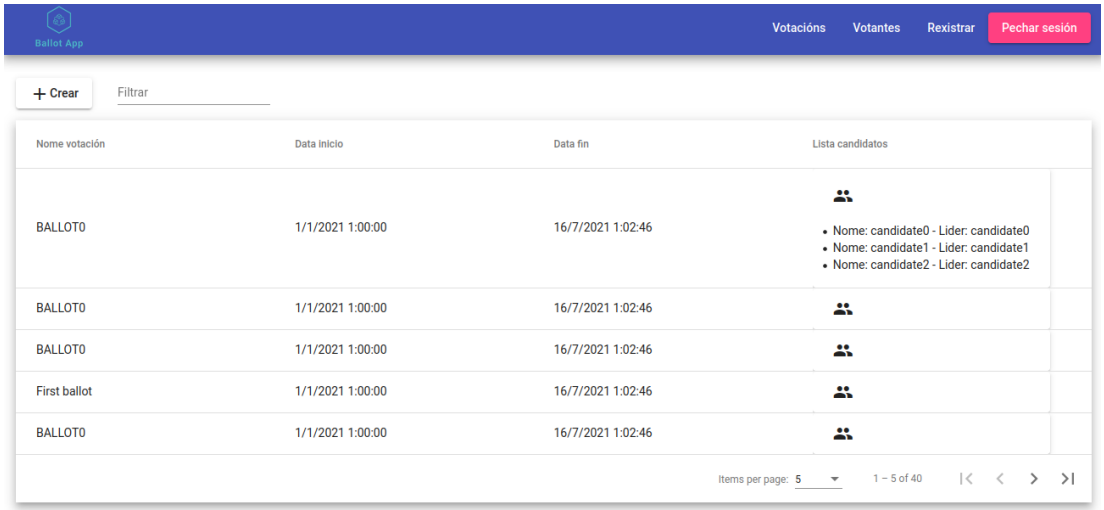

Figura 4.25: Listado de votacións do sistema.

A derradeira funcionalidade de administrador é a de **crear votación**, representada mediante a figura 4.26. Cabe destacar que o apartado de incorporar candidatos funciona de forma dinámica en función dos que se desexe engadir.

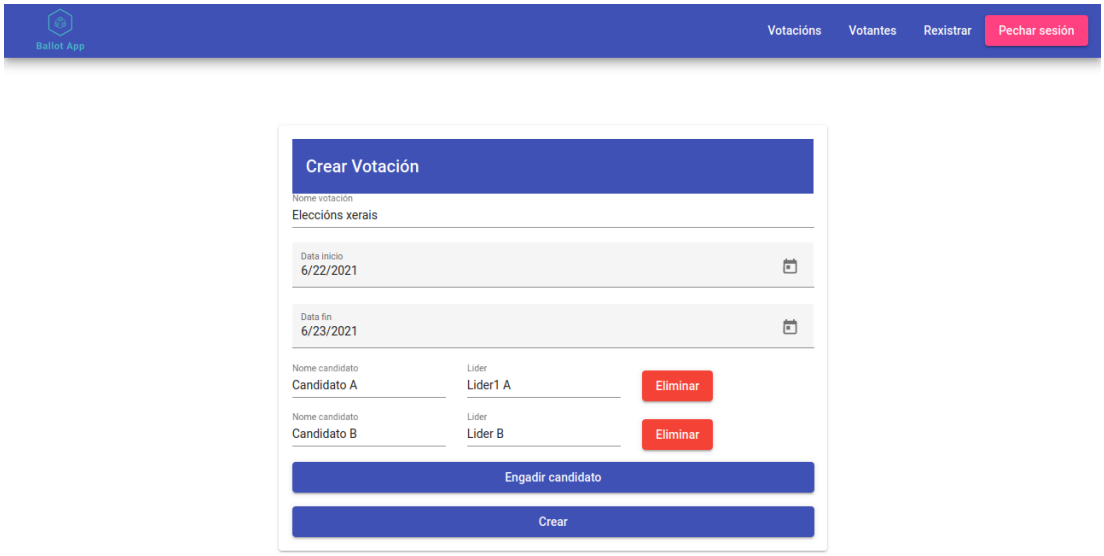

Figura 4.26: Formulario creación votación.

No tocante ás funcionalidades dos **votantes**, a primeira vista que visualizará será a de todas as votacións nas que pode participar. Tratase dunha compoñente similar as de listar coleccións do rol administrador, coa diferenza de que ten un apartado para emitir o seu voto e outro para visualizar os resultados da mesma, sempre e cando finalizase. Pódese observar a súa apariencia na figura 4.27.

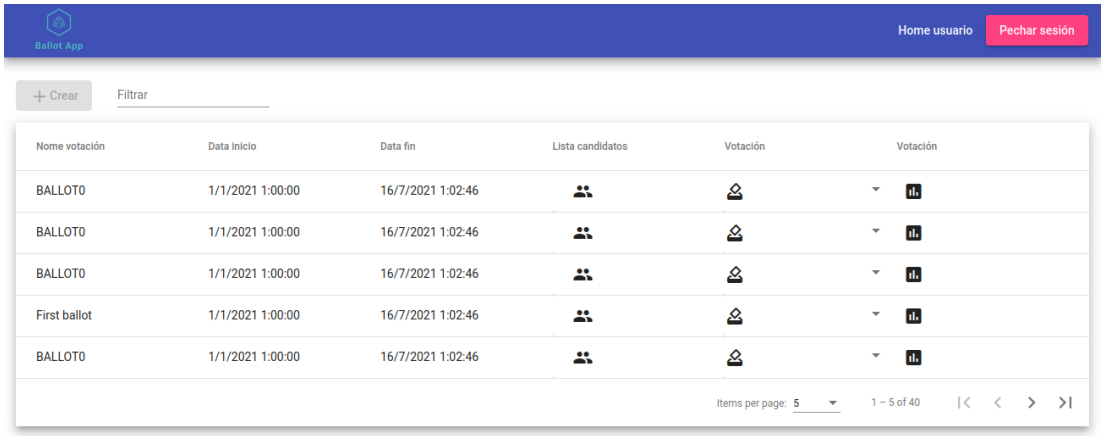

Figura 4.27: Votacións habilitadas para un usuario.

Para o proceso de votación, representado na figura 4.28, podese observar, tras elexir o candidato a votar mediante un selector, unha mensaxe de precaución para evitar que o votante emita o voto por erro.

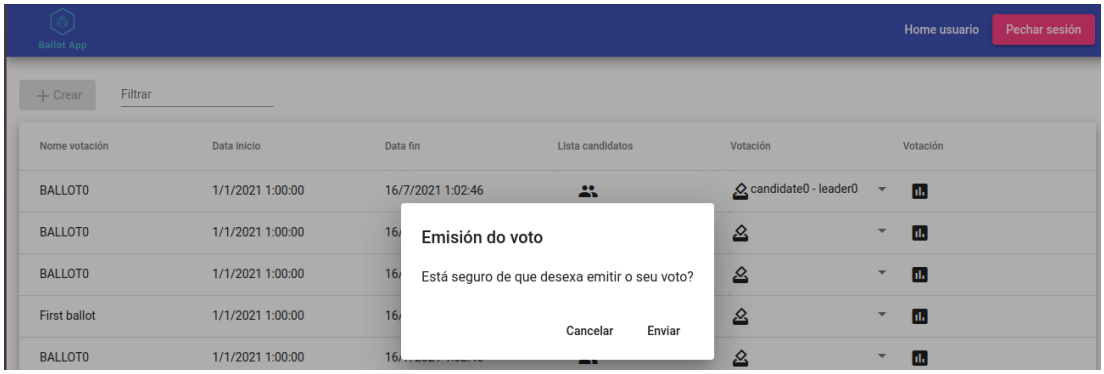

Figura 4.28: Emisión dun voto.

O derradeiro paso para concluír o proceso electoral sería visualizar os resultados da votación, para o que se emprega un pop-up cos resultados do escrutinio, tal e como se amosa na figura 4.29.

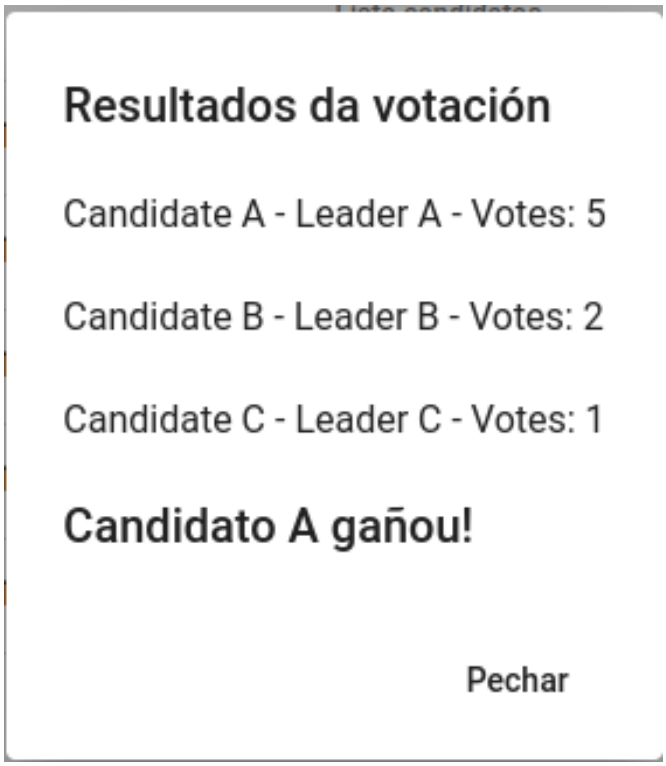

Figura 4.29: Resultado do escrutinio dunha votación.

Capítulo 5

Monitorización e rendemento do sistema

Os resultados dunha votación resultan críticos na evolución das entidades implicadas, polo que é preciso complementar o sistema descrito durante os capítulos anteriores con mecanismos que permitan monitorizar a infraestructura durante a votación e que permitan validar que está capacitada para soportar os requerimentos de rendemento.

5.1 Hyperledger Explorer

Un dos principais problemas dos sistemas de votación electrónicos é a desconfianza na integridade do voto dende a súa emisión ata o escrutinio. A tecnoloxía blockchain soluciona este problema mediante a cadea de bloques, pero acceder a esta información pode resultar complicado.

Froito deste lance xorde **Hyperledger Explorer**, unha ferramenta de visualización da cadea de bloques de forma gráfica, que será de gran utilidade no sistema proposto neste proxecto para a monitorización das votacións.

Para a integración desta ferramenta só é preciso clonar o proxecto do repositorio oficial de Hyperledger Explorer [43], importar o directorio que contén o material criptográfico da rede para habilitar a conexión con esta, e seguir os pasos que se indican na documentación do directorio para instalar as dependencias precisas.

Na vista principal da ferramenta, ilustrada na figura 5.1, podemos observar un dashboard que mostra os elementos actuais da rede (bloques, número de transaccións, nodos e chaincodes), unha lista de todos os nodos, o ratio de transaccións por unidade de tempo, un gráfico sobre as transaccións xeradas por cada organización e un listado dos bloques inseridos na blockchain.

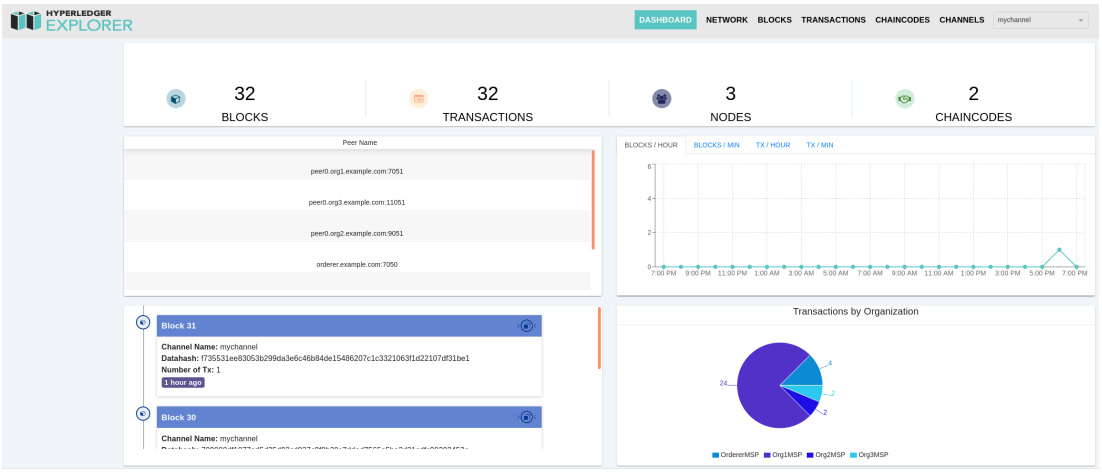

Figura 5.1: Dashboard de Hyperledger Explorer.

As vistas restantes amosan información máis detallada sobre os elementos da vista principal. Polo tanto, para mostrar os aspectos fundamentais desta ferramenta, profundizarase nas que presenten información máis relevante para a monitorización da rede.

Na pestaña de *Network*, representada na figura 5.2, podemos observar información detallada sobre todos os nodos da rede.

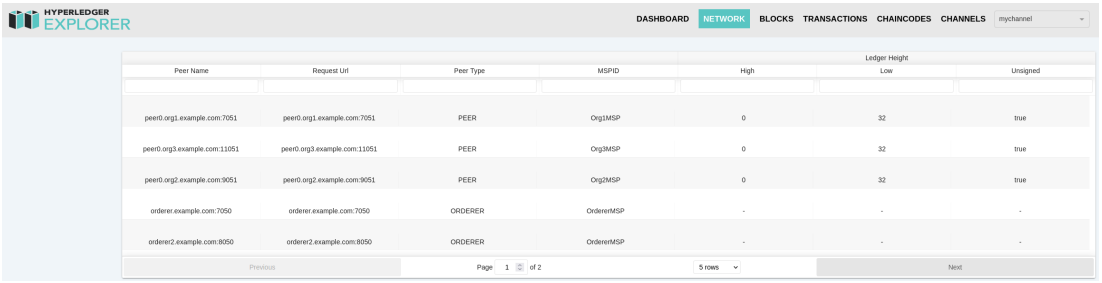

Figura 5.2: Visualización nodos da rede en Hyperledger Explorer.

Na ventá de *Blocks*, presentada na figura 5.3, lístanse todos os bloques inseridos na blockchain filtrando por un rango de fechas a elexir polo usuario. Esta vista é de gran axuda para facilitar tarefas de auditoría xa que se trata dun mecanismo para obter un histórico dos eventos da rede dun modo moi sinxelo e visual.

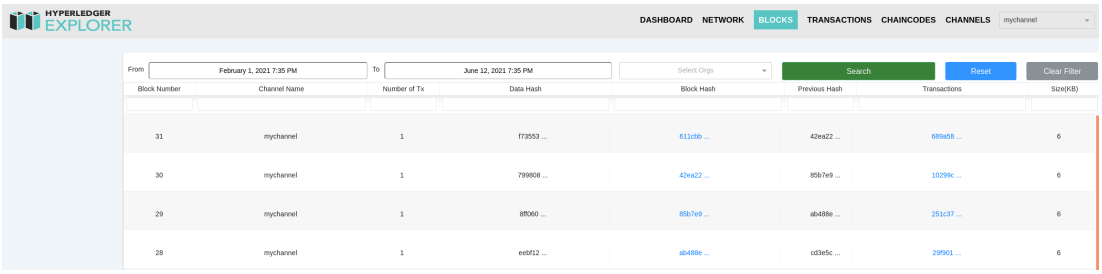

Figura 5.3: Visualización bloques da rede en Hyperledger Explorer.

Finalmente, na figura 5.4 representase a vista de transaccións, na que se mostra un listado das transaccións dentro dun rango de fechas fixado polo usuario.

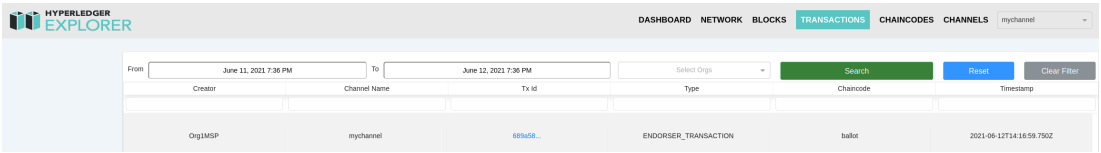

Figura 5.4: Visualización transaccións da rede en Hyperledger Explorer.

Cabe destacar que tanto na vista de bloques como na de transaccións podemos observar en detalle as transaccións implicadas, tal e como se mostra na figura 5.5.

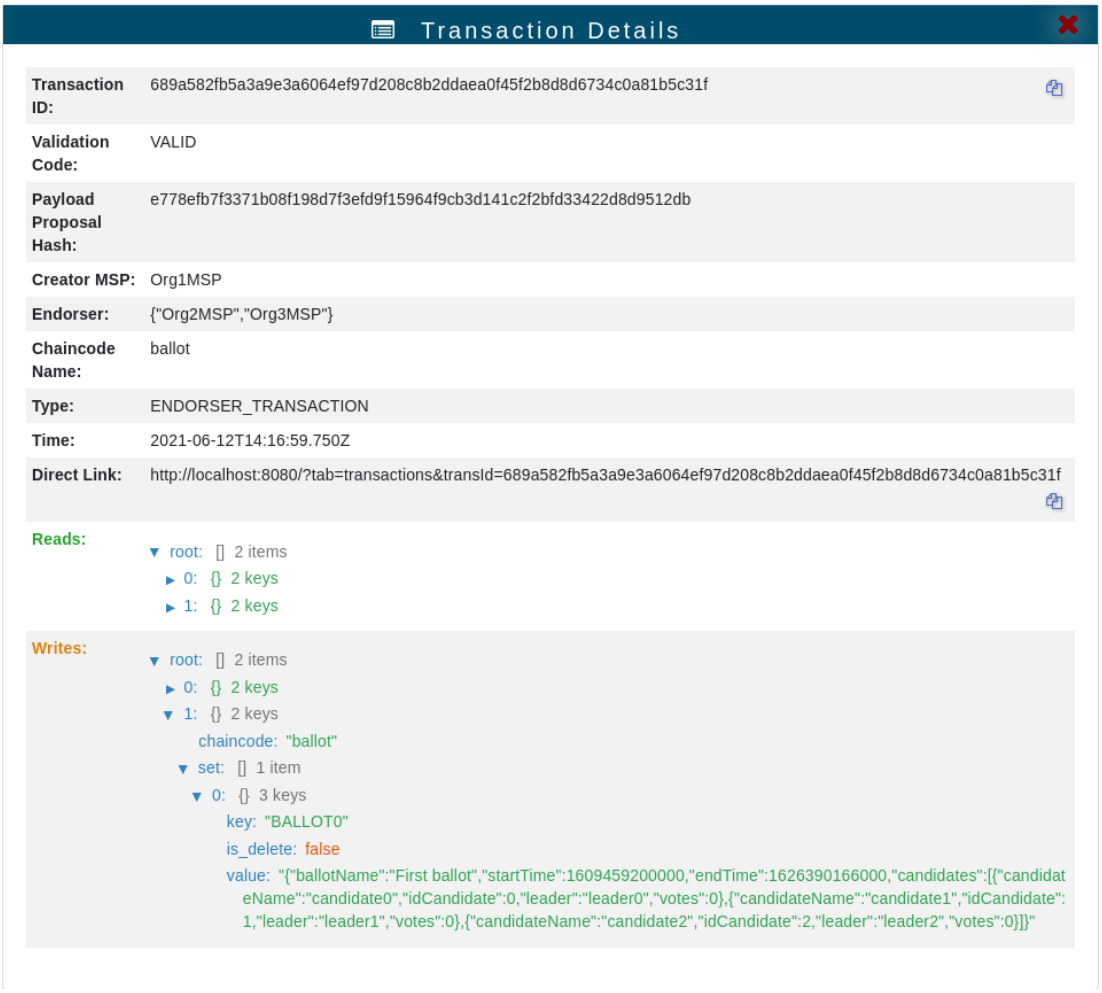

Figura 5.5: Exemplo transacción da rede en Hyperledger Explorer.

5.2 Hyperledger Caliper

Un dos principais problemas dos sistemas de votación tradicionais é a masificación de votantes nos recintos electorais, que da lugar a gran cantidade de tempo transcurrido entre cada voto. Polo tanto, un sistema de votación electrónico debe ser capaz de xestionar unha alta carga de transaccións á vez que mantén un rendemento e un tempo de resposta mínimo.

Para validar o rendemento do sistema empregarase **Hyperledger Caliper**, ferramenta que permite realizar informes sobre diferentes métricas de rendemento de redes blockchain.

5.2.1 Implementación Hyperledger Caliper

Para a integración de desta ferramenta, é preciso realizar os seguintes pasos:

- Importar o material criptográfico dos nodos da rede blockchain, necesario para recrear a autenticación da rede de producción.
- Definir a estructura da rede de test, que debe ser a mesma que a de producción, e asociar o material criptográfico importado cos nodos correspondentes. Realizarase sobre o ficheiro *network-config_2.2.yaml*.
- Implementar os casos de proba. Agruparanse sobre o directorio *caliper-benchmarks-local*.
- Establecer os test que van ser evaluados e as características a aplicar (transaccións por segundo, número total de transaccións, etc.). Serán englobados no ficheiro *config.yaml*.

Finalmente só restaría levantar un contedor docker coa imaxe de Hyperledger Caliper e establecer nas variables de entorno os correspondentes os ficheiros descritos 5.6.

```yaml
version: "2"

services:
  caliper_2.2:
    container_name: caliper_2.2
    image: hyperledger/caliper:0.4.1
    command: launch manager --caliper-fabric-gateway-enabled
    environment:
      - CALIPER_BIND_SUT=fabric:2.1.0
      - CALIPER_BENCHCONFIG=benchmarks/scenario/simple/ballots-v2.2/config.yaml
      - CALIPER_NETWORKCONFIG=networks/fabric/v2.1/network-config_2.2.yaml
    volumes:
      - ./caliper-benchmarks-local:/hyperledger/caliper/workspace
    network_mode: host
```

Figura 5.6: Configuración en Docker de Hyperledger Caliper.

5.2.2 Conceptos previos

Antes de estudar os resultados de rendemento do sistema, cómpre definir as métricas a examinar:

- Send Rate (TPS): Número de transaccións por segundo que o framework lanzará sobre un escenario de test.
- Max Latency (s): Trátase do tempo máximo que se tardou en realizar unha transacción, medido en segundos.
- Min Latency (s): Tempo mínimo que se tardou en executar unha transacción, medido en segundos.

- Avg Latency (s): Métrica que consiste no tempo medio das transaccións executadas durante un test, medido en segundos.
- Throughtput (TPS): Trátase das transaccións por segundo que a rede blockchain é capaz de soportar para un determinado test.

Para poder considerar as probas de rendemento como válidas, organizaranse de modo independente e baixo a mesma estructura.

En primeiro lugar, inicializarase o estado do ledger no caso de ser preciso (e.g., creando unha votación para evaluar a función de emisión de voto) mediante a función predeterminada da ferramenta, *initializeWorkloadModule()*. Na figura 5.7 podemos observar un exemplo de como se definiría un test, para o cal só se precisa definir o smart contract e a versión sobre o que se vai realizar, a función a analizar e os parámetros que precisa.

```
/**
 * Assemble TXs for the round.
 * @return {Promise<TxStatus[]>}
 */
async submitTransaction() {
    this.txIndex++;

    let args = {
        contractId: 'ballot',
        contractVersion: 'v1',
        contractFunction: 'vote',
        contractArguments: ["BALLOT0","0","1A"],
        timeout: 30,
        readOnly: true
    };


    await this.sutAdapter.sendRequests(args);
}
```

Figura 5.7: Exemplo de parámetros do test de caliper para a emisión de votos.

Cabe destacar que, ao igual que ocorre con outras plataformas de análise de rendemento e benchmarking, o tempo empregado para a inicialización lo ledger non se inclúe dentro da proba.

5.2.3 Análise dos resultados

Hyperledger Caliper xera un informe en html, que consta fundamentalmente de 3 bloques:

- **Información da estructura** sobre a que se realizan os tests, representada na figura 5.8, mostrando datos básicos sobre o test, un listado de enlaces aos resultados de cada tests e información sobre as características do sistema a evaluar.

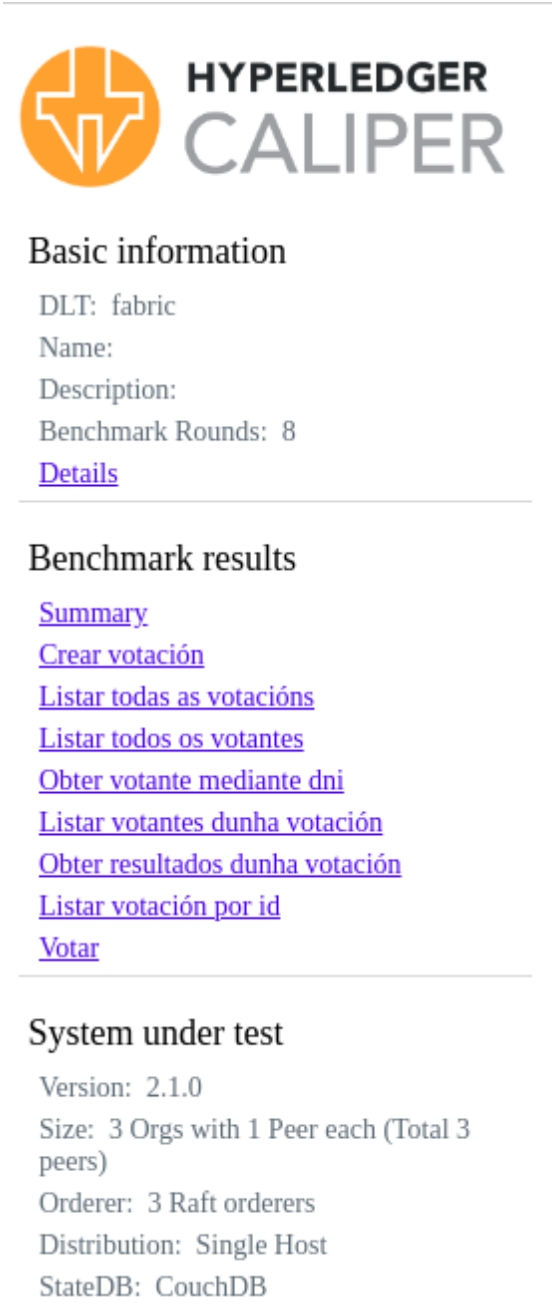

Figura 5.8: Información da rede de testing de Caliper.

- Na figura 5.9 podese observar un **resumo xeral** de todos os tests realizados. Cabe destacar que se empregarán dous ratios de envío de transaccións. Para operacións que serán realizadas por **administradores** realizaranse tests emitindo **50 transaccións por segundo**, dado que serán empregadas por poucos usuarios e de forma pouco recurrente. No tocante ás funcionalidades habilitadas para os usuarios, analizarase o rendemento do sistema ante **500 transaccións por segundo**.

Caliper report

Summary of performance metrics

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
Crear votación	1000	0	49.9	13.33	2.30	8.73	31.3
Listar todas as votacións	1000	0	50.1	0.11	0.01	0.03	50.0
Listar todos os votantes	1000	0	50.1	0.17	0.01	0.03	50.0
Obter votante mediante dni	1000	0	50.1	0.11	0.01	0.02	50.0
Listar votantes dunha votación	1000	0	50.1	0.11	0.01	0.03	49.9
Obter resultados dunha votación	1000	0	50.1	1.81	0.08	1.17	48.0
Listar votación por id	2500	0	307.8	10.21	2.20	5.84	239.5
Votar	2500	0	297.4	10.34	2.06	5.88	231.9

Figura 5.9: Resumo xeral dos tests.

- Adicionalmente, o informe xerado mostra tamén os **tests individualmente**, tal e como se representa na figura 5.10.

Benchmark round: Listar todas as votacións

Operación de listar todas as votacións

```
rateControl:
  type: fixed-rate
  opts:
    tps: 50
```

Performance metrics for Listar todas as votacións

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
Listar todas as votacións	1000	0	50.1	0.11	0.01	0.03	50.0

Figura 5.10: Desglose individual do test de listar todas as votacións.

Para analizar os resultados, desglosarase a análise segundo o rol implicado dado que o número de transaccións afectadas non é o mesmo.

Respecto ao rol **administrador**, podemos observar que as operacións de **lectura** sobre unha lista ou sobre unha colección de elementos cumpren as expectativas, presentando latencias de **30 ms de media**, aínda que no caso das de **escritura**, a creación de votacións ten unha latencia elevada, de **8,73 s de media**, debido a que o sistema non é capaz de soportar as 50 transaccións por segundo para esta funcionalidade, dando lugar a que queden transaccións pendentes de procesar e en consecuencia aumente o seu tempo de execución.

No referente ao rol de **votante**, destacan as elevadas transaccións por segundo que soporta o sistema para obter as votacións por id e para votar, con latencias medias de **5,8 s de media**. Ao igual que para a creación de votacións, a latencia é relativamente elevada debido a que o sistema non é capaz de xestionar todas as transaccións enviadas.

Na figura 5.11 reflictese o número máximo de transaccións por segundo que poden soportar as funcións de crear votacións, obter o detalle dunha votación e de emisión de voto sen penalizar a latencia. Só se inclúen as de estes tres métodos debido a que o resto cumpre cos requerimentos previstos.

Caliper report

Summary of performance metrics

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
Crear votación	998	2	20.0	6.53	1.62	3.33	18.8
Listar votación por id	2500	0	150.1	0.24	0.01	0.02	150.0
Votar	2500	0	200.0	1.21	0.01	0.14	199.7

Figura 5.11: Analise de tps para obter latencias óptimas.

Cabe resaltar dúas características dos tests que non se ven reflexadas na figura 5.9, as **transaccións por segundo** das funcionalidades do rol votante e a emisión de só 50 transaccións por segundo na función de **obter os resulados da votación**. Este ratio de emisión é o máximo soportado polo sistema para esta funcionalidade, debido a motivos hardware e limitacións no funcionamento de Hyperledger Caliper, o cal trataremos no seguinte apartado 5.2.4.

Á vista dos resultados, podemos concluír que se cumpren os requisitos de rendemento, dando lugar a un **sistema escalable**.

5.2.4 Limitacións de Hyperledger Caliper

Pese a gran utilidade que aporta esta ferramenta, durante o seu uso para analizar o rendemento do sistema detectáronse múltiples limitacións:

- **Número máximo de transaccións procesables**. Por defecto, os nodos de Fabric soportan un máximo de 2500 transaccións concurrentes. Dado que se procesan múltiples transaccións por segundo deberíanse poder facer tests con un número total de transaccións maior, sen embargo tal e como se mostra na figura 5.12, na que se configurou un test de 5000 transaccións, esto non ocorre. O motivo é que Hyperledger Caliper non reutiliza as conexións, polo que aínda que unha transacción finalice, os nodos do sistema non o interpretan así. Este problema trátase dun erro coñecido [44], mais aínda que propoñen unha solución parchear esta limitación, non é aplicable aos tests do sistema, xa que se recrea o proceso de transacción completo en lugar de realizalo directamente como no caso do fío de github.

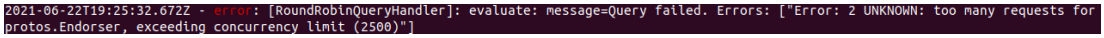

Figura 5.12: Erro número máximo de transacción totales superado.

- **Límite no envío de transaccións:** A pesar de que os test do rol votante están configurados para que se envíen 500 transaccións por segundo, na realidade envíanse en torno a 300. Esto ocorre por motivos hardware do sistema, xa que todas as compoñentes da rede están despregadas sobre o mesmo ordenador, polo que comparten recursos, ademais da perda de recursos derivada da incidencia do punto anterior.

Estas limitacións reflicten o estado actual da ferramenta, etiquetada co estado de **incubación** [45]. Esta etiqueta dos proxectos de Hyperledger indica que o proxecto non está completamente funcional, se non que pese a presentar certas funcionalidades, todavía se están explorando distinta capacidades, polo que estas non teñen un alcance e uso plenamente definidos nin garanten un funcionamento óptimo.

Capítulo 6

Planificación e incidencias

Un aspecto fundamental para ter éxito á hora de realizar un proxecto é trazar un plan de actuación, englobando as tarefas a realizar e o tempo de dedicación para poder afrontalo de forma organizada e adiantarse a posibles contratempos. Neste capítulos abordarase a **metodoloxía** empregada, a **planificación** seguida e as **incidencias** rexistradas durante a execución de todo o proxecto.

6.1 Metodoloxía de desenvolvemento

O primeiro paso para deseñar unha planificación dun proxecto é elexir o procedemento que se levará á hora de implementar a solución requerida. Debido ao baixo nivel de acoplamento dos módulos deste proxecto (a excepción da rede blockchain, que é o epicentro do sistema) decantouse por desenvolvemento **incremental**, asociando un dos módulos a cada incremento. En concreto, cada incremento consta dos seguintes pasos:

- Definición dos **obxectivos**.
- Organización en **tarefas** e **estimación** de tempo.
- **Implementación** do módulo asociado ao incremento.
- **Documentación** do proceso realizado.
- **Validación** do incremento.

6.1.1 GIT Flow

Establecer un fluxo de traballo e levar a cabo un control de versións son aspectos clave para poder levar a cabo a planificación de forma eficaz. Debido ao auxe de GIT, unha plataforma de código aberto que prove servizos de control de versións, elexiuse un fluxo de traballo asociado a esta tecnoloxía, **GIT Flow**.

Neste paradigma, cada funcionalidade debe ser realizada nunha rama independente, cun obxetivo claro e acotado. Adicionalmente establécense as seguintes normas en función do tipo de rama:

- Master: Rama estable que sempre debe funcionar correctamente. Sería a asociada a producción.
- Develop: Rama de desenvolvemento na que se integrarán as distintas ramas de funcionalidades.
- Features: Ramas nas que se desenvolverán as funcionalidades.
- Hotfix: Rama empregada para realizar arreglos erros en entornos de producción.

Na figura 6.1 podemos observar un exemplo de como se seguiu este paradigma durante o desenvolvemento do sistema.

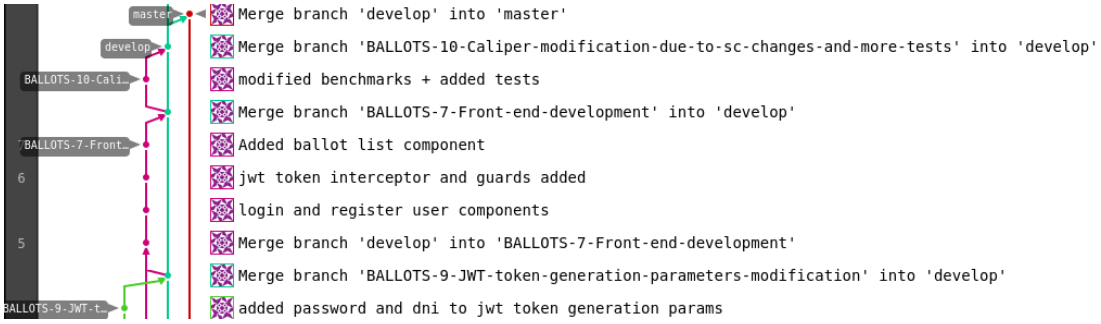

Figura 6.1: Exemplo seguemento paradigma GIT Flow.

Adicionalmente seguiuse un nomeamento de ramas asociando a tarefa a un id ficticio, simulando que a rama é creada a partir dunha tarefa xerada mediante algún software de xestión de incidencias e xestión de proxectos como Jira. Na figura 6.2 móstrase un exemplo desta práctica.

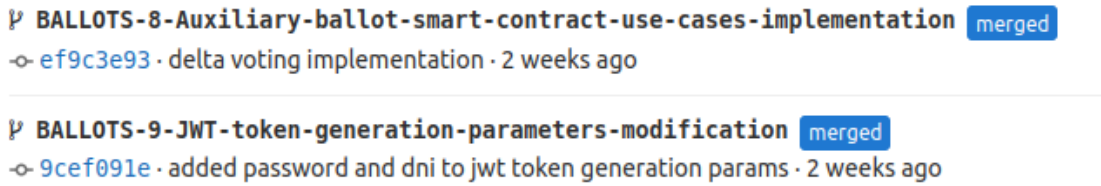

Figura 6.2: Exemplo de definición de ramas simulando tarefas de Jira.

6.2 Planificación do proxecto

Neste apartado da memoria tratarase a división do proxecto en tarefas e a estimación da súa duración. En concreto a realización deste tivo unha duración de 308 distribuídas nas tarefas da figura 6.3.

Nombre	Fecha de inicio	Fecha de fin
Estudo viabilidade + redacción anteproxecto	8/3/21	10/3/21
Introducción e definición obxectivos	10/3/21	20/3/21
Estudo da arte	20/3/21	2/4/21
Deseño sistema	2/4/21	24/4/21
Fin iteración 1: Diseño sistema	24/4/21	24/4/21
Implementación rede blockchain	25/4/21	30/4/21
Implementación smart contracts	1/5/21	6/5/21
Implementación Middleware	7/5/21	13/5/21
Fin it.2 : Implementación blockchain + middleware	14/5/21	19/5/21
Integración Hyperledger Explorer	14/5/21	18/5/21
Integración Hyperledger Caliper	19/5/21	29/5/21
Fin it.3 : Integración Explorer + Caliper	29/5/21	29/5/21
Novo deseño sistema	30/5/21	2/6/21
Implementación novo sistema	3/6/21	5/6/21
Implementacion front-end	6/6/21	8/6/21
Fin it.4 : Sistema funcional v1	8/6/21	8/6/21
Documentación do sistema	9/6/21	23/6/21
Fin it.5 : Documentación final do sistema	23/6/21	23/6/21

Figura 6.3: Tarefas realizadas durante o proxecto.

Para obter unha visión temporal do desenvolvemento do proxecto, na figura 6.4 móstrase un diagrama de Gantt que expon a duración das tarefas que acabamos de mencionar. Destacar que os hitos do diagrama representan a fin dunha iteración.

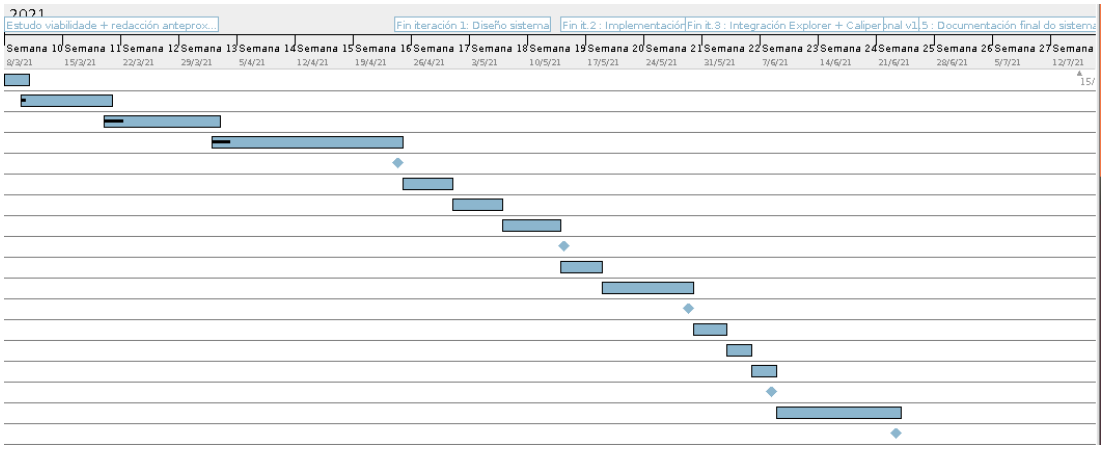

Figura 6.4: Diagrama de Gantt do proxecto.

6.3 Estimación de costes

Un dos obxectivos de levar a cabo un proxecto adoita ser obter un beneficio económico, polo que estimar os costes asociados á súa realización resulta fundamental para poder estudar a súa viabilidade.

Neste proxecto participaron tres recursos, un xefe de proxecto, un analista e un programador. O xefe de proxecto foi o encargado do deseño do sistema e da súa validación, o analista mentres que o programador foi o encargado de realizar a implementación. Na figura 6.5 podemos observar unha táboa resumo da estimación de costes.

Recurso	Coste por hora	Horas dedicadas	Coste total por recurso
Xefe de proxecto	50	57,5	2875
Analista	40	72,5	2900
Programador	20	178	3560
Total:		308 horas	9335 euros

Figura 6.5: Estimación de costes do proxecto.

6.4 Incidencias rexistradas

Durante o transcurso dun proxecto, na gran maioría adoitan xurdir contratempos. Nesta sección trataremos as principais incidencias atopadas durante o desenvolvemento do proxecto.

Debido a que todos os frameworks e linguaxes de programación usados neste TFG non foron vistos durante o grao ou foron tratados de forma moi superficial, xurdiron incidencias e dificultades relacionadas coa aprendizaxe das mesmas. Sen embargo, este tipo de dificultades

non serán contempladas nesta apartado dado que non son problemas derivados do proxecto en si.

6.4.1 Incidencia de deseño

Durante as dúas primeiras iteraccións do proxecto, o proceso de votación cosnistía na actualización dunha variable contador asociada ao candidato a votar. Tras a integración do sistema con Hyperledger Caliper, detectouse que apenas se podían relizar transaccións de votación concurrentes. Esto debese á funcionalidade de Hyperledger Fabric **Multiversion Concurrency Control(MVCC)**, explicada na sección 2.5.4, que impide modificar múltiples veces a mesma entidade de forma concurrente, dado que ao commitearse a primeira transacción modificaba a versión do elemento en cuestión, inhabilitando o resto de transaccións sobre este debido a que o valor leído ao comezo da transacción é diferente ao valor a escribir no momento de realizar a transacción.

Como consecuencia, foi preciso realizar un redeseño da rede blockchain para realizar a votación mediante claves compostas, tal e como se explica no apartado de deseño 3.4.4 relacionado coa emisión do voto.

6.4.2 Métricas de rendemento

Pese ao gran impacto que tivo a integración de Hyperledger Caliper á hora de detectar problemas de rendemento e o beneficio que aportan as métricas que proporciona, as limitacións que posúe, descritas na sección 5.2.4, deron lugar a ter que empregar gran cantidade de horas en realizar numerosas probas sobre o sistema ata detectar os problemas da propia ferramenta.

Capítulo 7

Liñas futuras e conclusións

7.1 Liñas futuras

Debido ao caracter xenérico do sistema, os sistema pode tomar diferentes camiños a nivel **administrativo**.

Por un lado, si se dirixe ao eido da **administración pública**, para empregalo en procesos de electorais a gran escala como votacións nacionais ou autonómicas, o sistema debería ser sometido a múltiples probas de auditoría e a tests de rendemento máis severos dado o elevado impacto que podería xerar un ataque ou mal funcionamento do sistema.

Por outro lado, si se enfoca a aplicación a nivel **empresarial**, debido ao entorno cambiante dos acordos comerciais entre empresas, sería preciso implementar un sistema para engadir ou eliminar membros da rede. Adicionalmente, unha funcionaliade de gran utilidade sería a posibilidade de crear canles privadas asociadas ás empresas dun consorcio, evitando ter que despregar múltiples instancias do sistema.

En caso de querer seguir ambos enfoques sería preciso incrementar o equipo de desenvolvemento, debido ao alcance e impacto do proxecto.

A nivel **técnico**, unha modificación importante que se debería realizar no sistema é implementar un proceso independente que se active cando unha modificación finaliza para realizar o escrutinio e actualizar os valores dos candidatos asociados a esa votación. Esto provocaría un gran incremento no rendemento do sistema, debido a que a función de realizar o escrutinio realizaríase solo unha vez, e a función que se lanzaría múltiples veces sería a de obter a votación por id, a cal ten unha latencia moito menor e soporta un maior número de transaccións.

Por outra parte, o método de rexistro de votantes resulta ineficiente ante votacións a gran escala, dado que de non estar rexistrados na plataforma habería que engadilos un a un. Polo tanto, outro dos desenvolvementos futuros debería ser implementar un proceso de rexistro de usuarios a partir dun ficheiro ou mediante outras alternativas automatizadas.

Finalmente, no tocante á **infraestructura**, sería un gran avance despregar o sistema sobre

algunha plataforma de integración continua como **Bamboo**. Así, poderíase actualizar os servizos con tempos de caída mínimos e lanzar de forma automática tests sobre o sistema. Outra opción sería integrar o sistema en plataformas cloud como **AWS**, que conta cunha plataforma para a creación de sistemas sobre Hyperledger Fabric.

7.2 Conclusións

En relación coa analise exposta e co sistema desenvolto, podemos extraer varias inferencias sobre o estado actual e o posible alcance de blockchain.

Por un lado, pese aos avances e melloras que se van integrando nesta tecnoloxía, debemos ser conscientes de que todavía existen moitos aspectos que non contan cunha solidez adecuada. Exemplos desta dinámica son os casos de Hyperledger Caliper ou a xestión da concurrencia en Hyperledger Fabric, que pese a outorgar unhas capacidades excepcionales teñen un gran camiño por recorrer.

Por outro lado, o sistema desenvolto neste proxecto deixa patente que o uso de solucións baseadas nunha filosofía descentralizada e transparente están chamadas a substituír moitos dos procesos administrativos actuais, liberandoos da saturación que adoitan presentar e de intermediarios innecesarios.

En base aos resultados obtidos, podemos resaltar tamén os elevados requisitos hardware precisos para obter unha descentralización plena e con gran tolerancia a fallos, motivo polo que considero que o uso de blockchain baixo plataformas propias está nunha fase moi temprana, debido ao elevado custo que suporía dispoñer dun gran número de nodos en funcionamento.

Sen embargo, o caracter innovador desta tecnoloxía resulta innegable. Cada vez son máis as empresas e as institucións públicas as que apostan por desenvolver solucións mediante a blockchain, seguindo a crecente corrente hacia a automatización e a menor dependencia humana na xestión de procesos.

A pesar da reticencia a modificar procesos que levan moitos anos arraigados na nosa sociedade, resulta intanxible un futuro adverso ao avance inexorable da tecnoloxía **blockchain**.

Apéndices

Apéndice A

Material adicional

A.1 Ferramentas e linguaxes de programación empregadas

Neste apartado trataranse as distintas ferramentas e linguaxes de programación empregadas durante o desenvolvemento deste proxecto. Seguindo a metodoloxía aplicada a o longo de todo o documento, comezarase dende as capas inferiores do sistema ata as superiores.

A.1.1 Rede blockchain

Para a imlementación das CA's e da rede blockchain empregáronse as librerías e imaxes docker proporcionadas pola plataforma de **Hyperledger Fabric**. En concreto, utilizouse o SDK para linguaxe **Go**, linguaxe de programación de código aberto que destaca polo seu tipado estático e polos seus potentes mecanismos de xestión de concorrencia [46], que tamén foi a empregada para o desenvolvemento dos smart contracts. Para os desenvolvementos relacionados con docker a linguaxe aplicada foi **yaml**.

Para a creación dos artefactos, da canle e o despregue dos smart contracts, realizouse mediante **Shell scripting** en conxunto coas librerías proporcionadas por Hyperledger Fabric para interactuar coa rede blockchain.

Un aspecto importante neste punto é o uso de certificados X.509 para realizar a autenticación dos elementos que interactúen coa blockchain.

A.1.2 Middleware

Para a implementación do servidor preciso para establecer a comunicación entre a interfaz de usuario e a rede blockchain empregouse a linguaxe de programación **Node.js**, entorno de ejecución de Javascript orientado a eventos asíncronos para a creación de aplicacións en rede escalables [47]. Adicionalmente empregaronse os SDKS **fabric-ca-client** para realizar a comunicación coas CA's e **fabric-network** para a creación das wallets e a interacción coa

rede blockchain.

Cabe destacar tamén o uso de **tokens JWT**, seguindo as boas prácticas definidas no documento RFC5719.

A.1.3 Interfaz de usuario

Para este módulo do sistema empregouse o framework **Angular**, un framework que permite crear aplicacións web nunha única páxina, cun deseño altamente desacoplado debido a que se segue un deseño por compoñentes, mecanismo que axiliza enormemente o desenvolvemento [48].

A.1.4 Ferramentas adicionais

Adicionalmente os tres bloques centrais que se acaban de mencionar, empregaronse as seguintes ferramentas complementarias:

- **Hyperledger explorer**, unha ferramenta da plataforma de Hyperledger para a monitorización e visualización de forma gráfica da rede blockchain.
- **Hyperledger Caliper**, outra ferramenta desenvolta por Hyperledger para realizar métricas de rendemento.
- **CouchDB**, peza importante do sistema debido a que será donde os peers commiteen o resultado das transaccións.
- **Visual Studio Code**, un editor de código lixeiro pero potente, con soporte e extensións para múltiples linguaxes [49].
- **Postman**, plataforma colaborativa de desenvolvemento de APIs [50].

Apéndice B

Acrónimos

ACLs: Access-Control lists
CA: Certification Authority
CFT: Crash Fault Tolerant
Dapp: Decentralized application
DLT: Distributed Ledger Technology
DRE: Direct Recording Electronics
EBPs: Electronic Ballots Printers
EVM: Ethereum Virtual Machine
IVS: Internet Voting System
JWT: JSON Web Tokens
MSP: Membership Service Provider
MVCC: Multiversion Concurrency Control
OMR: Optical Mark Recognition
P2P: Peer-to-Peer
PoW: Proof of Work
PoS: Proof of Stake
PBFT: Practical Byzantine Fault Tolerance
SCM: Supply Chain Management
TPS: Transactions per Second

Bibliografía

[1] J. Y. Espinoza, "A puertas del voto electrónico: un estudio preliminar," 2011, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: http://hdl.handle.net/10757/552727

[2] A. Fatrah, S. El Kafhali, A. Haqiq, and K. Salah, "Proof of concept blockchain-based voting system," in *Proceedings of the 4th International Conference on Big Data and Internet of Things*, ser. BDIoT'19. New York, NY, USA: Association for Computing Machinery, 2019. [En liña]. Dispoñible en: https://doi.org/10.1145/3372938.3372969

[3] E. Zaghloul, T. Li, and J. Ren, "d-bame: Distributed blockchain-based anonymous mobile electronic voting," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[4] G. Han, Y. Li, Y. Yu, K.-K. R. Choo, and N. Guizani, "Blockchain-based self-tallying voting system with software updates in decentralized iot," *IEEE Network*, vol. 34, no. 4, pp. 166–172, 2020.

[5] B. Shahzad and J. Crowcroft, "Trustworthy electronic voting using adjusted blockchain technology," *IEEE Access*, vol. 7, pp. 24 477–24 488, 2019.

[6] F. D. Giraldo, B. Milton C., and C. E. Gamboa, "Electronic voting using blockchain and smart contracts: Proof of concept," *IEEE Latin America Transactions*, vol. 18, no. 10, pp. 1743–1751, 2020.

[7] Y. Takabatake and Y. Okabe, "An anonymous distributed electronic voting system using zerocoin," in *2021 International Conference on Information Networking (ICOIN)*, 2021, pp. 163–168.

[8] S. Gao, D. Zheng, R. Guo, C. Jing, and C. Hu, "An anti-quantum e-voting protocol in blockchain with audit function," *IEEE Access*, vol. 7, pp. 115 304–115 316, 2019.

[9] J.-G. Song, S.-J. Moon, and J.-W. Jang, "A scalable implementation of anonymous voting over ethereum blockchain," *Sensors*, vol. 21, no. 12, 2021. [En liña]. Dispoñible en: https://www.mdpi.com/1424-8220/21/12/3958

[10] "followmyvote," 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://followmyvote.com/

[11] S. Team, "Tivi - convenient, secure and fully verifiable online voting," 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://www.smartmatic.com/elections/remote-voting/tivi/

[12] "Netvote," 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://citizendata.network/netvote/

[13] R. Kuhn, D. Yaga, and J. Voas, "Rethinking distributed ledger technology," *Computer*, vol. 52, no. 2, pp. 68–72, 2019.

[14] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[15] IBM, "What are smart contracts on blockchain?" 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://www.ibm.com/topics/smart-contracts

[16] T. M. Fernández-Caramés and P. Fraga-Lamas, "A review on the use of blockchain for the internet of things," *IEEE Access*, vol. 6, pp. 32 979–33 001, 2018.

[17] S. Bouraga, "A taxonomy of blockchain consensus protocols: A survey and classification framework," *Expert Systems with Applications*, vol. 168, p. 114384, 2021. [En liña]. Dispoñible en: https://www.sciencedirect.com/science/article/pii/S0957417420310587

[18] V. Dhillon, D. Metcalf, and M. Hooper, *The Hyperledger Project*. Berkeley, CA: Apress, 2017, pp. 139–149.

[19] *Hyperledger Fabric Uses*. The Linux Foundation, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://www.hyperledger.org/use

[20] *Hyperledger Fabric Whitepaper*. The Linux Foundation, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger_fabric_whitepaper.pdf

[21] *Sawtooth doc introduction*. Intel Corporation, 2017, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://sawtooth.hyperledger.org/docs/core/releases/1.0/introduction.html

[22] T. Kuhrt, "Hyperledger iroha," 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://wiki.hyperledger.org/display/iroha/Hyperledger+Iroha

[23] "Hyperledger indy," 2018, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://indy.readthedocs.io/en/latest/

[24] T. Kuhrt, "Hyperledger burrow," 2019, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://wiki.hyperledger.org/display/burrow/Hyperledger+Burrow

[25] "Hyperledger grid," 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://grid.hyperledger.org/

[26] T. Kuhrt, "Hyperledger caliper," 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://wiki.hyperledger.org/display/caliper/Hyperledger+Caliper

[27] ——, "Hyperledger cello," 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://wiki.hyperledger.org/display/cello/Hyperledger+Cello

[28] "Hyperledger composer," 2019, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://hyperledger.github.io/composer/latest/introduction/introduction.html

[29] T. Kuhrt, "Hyperledger explorer," 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://wiki.hyperledger.org/display/explorer/Hyperledger+Explorer

[30] ——, "Hyperledger quilt," 2019, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://wiki.hyperledger.org/display/quilt/Hyperledger+Quilt

[31] ——, "Hyperledger ursa," 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://wiki.hyperledger.org/display/ursa/Hyperledger+Ursa

[32] A. Russell, "¿que es un certificado x.509?" 2019, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://www.ssl.com/es/preguntas-frecuentes/%C2%BFQu%C3%A9-es-un-certificado-x-509%3F/

[33] hyperledger, "Membership service providers (msp)," 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://hyperledger-fabric.readthedocs.io/en/release-2.2/msp.html

[34] G. Araujo, "Hyperledger fabric: Conceptos y tipos de nodos," 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://babel.es/es/Media/Blog/Agosto-2019/Hyperledger-Fabric-Conceptos-y-tipos-de-nodos

[35] H. Fabric, "Smart contracts and chaincode," 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://hyperledger-fabric.readthedocs.io/en/release-2.2/smartcontract/smartcontract.html#terminology

[36] ——, "Ordering service implementations," 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://hyperledger-fabric.readthedocs.io/en/release-2.2/orderer/ordering_service.html#ordering-service-implementations

[37] ——, "Transaction flow," 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://hyperledger-fabric.readthedocs.io/en/release-2.2/txflow.html

[38] Hyperledger, "Policies -," 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://hyperledger-fabric.readthedocs.io/es/latest/policies/policies.html

[39] ——, "Couchdb as the state database," 2020, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://hyperledger-fabric.readthedocs.io/en/release-2.2/couchdb_as_state_database.html

[40] *¿Qué es un SDK?* Red Hat, 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://www.redhat.com/es/topics/cloud-native-apps/what-is-SDK

[41] *rfc2616.* The Internet Society, 1999, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://datatracker.ietf.org/doc/html/rfc2616#page-18

[42] *rfc5719.* Internet Engineering Task Force (IETF), 2015, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://datatracker.ietf.org/doc/html/rfc7519

[43] Hyperledger, "Hyperledger/blockchain-explorer," 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://github.com/hyperledger/blockchain-explorer

[44] aldredb, "Caliper does not reuse connections when sending transactions to hf orderer," 2019, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://github.com/hyperledger/caliper/issues/690

[45] H. F. Team, "Project lifecycle - tsc," 2019, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://tsc.hyperledger.org/project-lifecycle.html#incubation

[46] G. team, "Documentation - the go programming language," 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://golang.org/doc/

[47] O. J. Foundation, "Acerca de nopde.js," 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://nodejs.org/es/about/

[48] Google, "Angular - introduction to de angular docs," 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://angular.io/docs

[49] Microsoft, "Documentation for visual studio code," 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://code.visualstudio.com/docs

[50] P. Team, “Postman - the collaboration platform for api development,” 2021, consultado o 23 de xuño de 2021. [En liña]. Dispoñible en: https://www.postman.com/