

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
DOBRE MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN
E ENXEÑARÍA DE COMPUTADORES

Pasaporte dixital de vacinación COVID-19 baseado en blockchain e aplicación móbil para o seu uso

Estudante: Mauro Alberto de los Santos Nodar

Dirección: Tiago Manuel Fernández Caramés

A Coruña, xuño de 2021.

*A todas e cada unha daquelas persoas que fixestes destes últimos cinco anos os mellores da
miña vida*

Agradecementos

Agradecido de por vida a meus pais, porque sen vós, nada disto tería sido posible. Grazas pola educación que me destes e os valores que me inculcastes, así como tamén pola vosa axuda de todo tipo a cal me acompañou cada día dos últimos cinco anos. A partir de agora, quedame demostrarvos que mereceu a pena e comezar devolvervos todo o que me destes.

Grazas tamén a todos aqueles profesores que me axudaron a crecer neste mundo da informática. Grazas a ese profesor que me fixo interesarme por unha asignatura ou poñerme cacharrear cun ordenador ao chegar á casa da facultade. E grazas en especial a Óscar, Fran e Jose, por como me tratastes, todo o que me ensinastes e a gran confianza e axuda que sempre me destes.

Por último, grazas Tiago, titor deste TFG e sen o que ningunha destas páxinas tería sentido. Mil grazas por toda a axuda, ideas e coñecemento que me aportache estes últimos catro meses, os cales espero que foran os primeiros de moitos.

Resumo

Debido á recente aparición da pandemia da COVID-19, as vidas de todos nós deron un xiro radical e na sociedade actual xurdiron novas etapas acompañadas de novas necesidades. Este traballo creará unha solución para unha delas, xa que aportará un sistema de certificados dixitais para o seguro almacenamento e compartición de datos médicos relacionados ca COVID-19, co fin de demostrar de forma inequívoca, infranqueable e segura a inmunidade ou falta de virus, baseándose dito sistema na tecnoloxía Blockchain e nos seus beneficios inherentes e presentando un sistema de incentivos cunha criptomoeda personalizada para recompensar aos usuarios que o empreguen.

Abstract

Due to the recent outbreak of the COVID-19 pandemic, everybody's life has changed dramatically, and nowadays society went through new stages accompanied by new needs. This work will create a solution for one of them, since it will provide a system of digital certificates for the secure storage and sharing of medical data related to COVID-19, with the goal of demonstrating immunity or lack of virus in a unequivocal, unbreakable and secure way. This system will be based on Blockchain technology and its inherent benefits, developing also an incentive system with a customized cryptocurrency in order to reward the users who use it.

Palabras chave:

- Blockchain
- Smart Contract
- Ethereum
- Criptomoeda
- COVID-19
- Pasaporte dixital de vacinación
- Certificado de inmunidade
- Aplicación móbil distribuída
- Kotlin Mobile Multiplatform

Keywords:

- Blockchain
- Smart Contract
- Ethereum
- Cryptocurrency
- COVID-19
- Digital vaccination passport
- Immunity certificate
- Distributed mobile application
- Kotlin Mobile Multiplatform

Índice Xeral

1	Introdución	1
1.1	Contextualización	1
1.2	Utilidade real do sistema	2
1.3	Implicacións éticas e morais	3
1.4	Obxectivos	5
1.5	Estrutura da memoria	6
2	Estado da arte	9
2.1	Pasaportes dixitais ou certificados de vacinación COVID-19	9
2.2	Blockchain e Dapps	10
2.3	Blockchain no ámbito sanitario	11
2.4	Blockchain e COVID-19	12
2.5	Análise, crítica e proposta	13
2.6	Solución europea	14
3	Análise e deseño do sistema	17
3.1	Análise	17
3.2	Deseño xenérico	19
3.2.1	Arquitectura do proxecto	19
3.2.2	Modelo de datos	21
3.2.3	Casos de uso e diagramas de secuencia	22
3.3	Deseño da app	27
3.3.1	Parte compartida	28
3.3.2	Parte paciente	28
3.3.3	Parte CDC	32
3.4	Proposta de deseño	33
3.5	Metodoloxía	34

4	Fundamentos Tecnolóxicos	35
4.1	Base Teórica	35
4.1.1	Blockchain	35
4.1.2	Desenvolvemento móbil multiplataforma	39
4.2	Selección tecnolóxica	41
4.2.1	Globais	41
4.2.2	Blockchain	41
4.2.3	Dapp	42
5	Blockchain: cadea privada e contratos intelixentes	43
5.1	Implementación	43
5.1.1	Creación da Blockchain privada	44
5.1.2	Smart Contracts: rexistro médico e criptomoeda	47
5.2	Probos	52
5.2.1	Blockchain Privada	52
5.2.2	Remix + Metamask: Cadea privada e primeiros contratos	55
5.2.3	Smart Contracts: rexistro médico e criptomoeda	56
6	Aplicación móbil distribuída	59
6.1	Implementación	59
6.1.1	Lóxica destacada	59
6.2	Probos	64
6.2.1	Fluxo de execución A: Primeiros pasos e App CDC	65
6.2.2	Fluxo de execución B: App paciente e Funcionalidades Extra	69
7	Conclusións, Incidencias e Traballo Futuro	73
7.1	Conclusións	73
7.1.1	Cumprimento dos obxectivos principais	73
7.1.2	Cumprimento dos requisitos técnicos	74
7.1.3	Cumprimento doutros obxectivos	75
7.1.4	Cumprimento dos requisitos de TFG de dobre mención	75
7.1.5	Conclusión final	76
7.2	Incidencias	76
7.3	Próximos pasos e liñas futuras de cara a produción	77
A	Planificación e avaliación de custos	83
A.1	Planificación do proxecto	83
A.2	Avaliación de custos do proxecto	86

B Alternativas ao deseño da arquitectura do proxecto	87
C Deseño: elementos restantes	91
C.1 Casos de uso e diagramas de secuencia	91
C.2 Deseño da app	92
C.2.1 Parte compartida	92
C.2.2 Parte paciente	92
C.2.3 Parte CDC	95
D Selección Tecnolóxica Restante	99
D.1 Selección tecnolóxica	99
D.1.1 Globais	99
D.1.2 Blockchain	100
D.1.3 Dapp	100
E Blockchain: elementos restantes	103
E.1 Implementación	103
E.1.1 Creación da Blockchain privada	103
E.1.2 Smart Contracts: rexistro médico e criptomoeda	105
E.2 Probas	112
E.2.1 Blockchain Privada	112
E.2.2 Remix + Metamask: Cadea privada e primeiros contratos	113
E.2.3 Smart Contracts: rexistro médico e criptomoeda	116
F Elementos restantes da Dapp	119
F.1 Implementación	119
F.1.1 Lóxica destacada	119
F.2 Probas	123
F.2.1 Fluxo de execución A: Primeiros pasos e App CDC	123
F.2.2 Fluxo de execución B: App paciente e Funcionalidades Extra	123
G Proxecto de código aberto	125
Relación de Acrónimos	127
Glosario	129
Bibliografía	131

Índice de Figuras

2.1	Certificado COVID-19 proposto pola Xunta de Galicia / Sergas - Abril 2021 [11].	10
2.2	Diagramas overview das interaccións ca Blockchain e dos Smart Contracts presentes nun sistema de votación seguro [24].	11
2.3	Arquitectura de comunicacións onde se diseña un sistema Blockchain con incentivos relacionado co ámbito sanitario [19].	12
2.4	Imaxe da app móbil usada en EEUU para axudar ao cidadán a xestionar a vacinación COVID-19 [13].	13
3.1	Modelo elixido de escritura na Blockchain.	19
3.2	Modelo de Datos.	21
3.3	Casos de uso de Alice comezando ca app e sendo engadida ao sistema.	23
3.4	Casos de uso de Bob enviando seu pasaporte e lendo o de Alice.	24
3.5	Casos de uso de Alice recibindo dose e tendo unha reacción adversa á mesma.	25
3.6	Casos de uso de Alice facendo unha proba e comerciando ca criptomoneda.	27
3.7	Mocks de Log In e do primeiro Welcome Slider.	28
3.8	Mocks de cifrar e compartir QR e de engadir información persoal do usuario.	30
3.9	Mocks de escáner QR e lectura de pasaporte incorporado ao sistema.	32
3.10	Mockups dos tipos de lectura de pasaporte dende CDC.	33
3.11	Proposta inicial de deseño.	34
4.1	Overview do funcionamento da tecnoloxía Blockchain [40].	37
4.2	Arquitectura KMM [28].	40
5.1	Comprobación de contas e saldo da coinbase na nosa Blockchain.	53
5.2	Proba múltiple do nodo despregado na Blockchain privada.	54
5.3	Logs da proba múltiple do nodo despregado na Blockchain privada.	54
5.4	Probando as contas da cadea con Metamask.	55

5.5	Comprobando os logs da Blockchain tras o envío da transacción.	55
5.6	Probando as contas da cadea con Metamask.	56
5.7	Resultado da proba mock de engadir conta CDC á Blockchain.	56
5.8	Probas na Blockchain privada: Primeiros pasos.	56
5.9	Probas na cadea privada: Primeiro paso e lectura gratuíta do usuario engadido.	57
5.10	Debug da lectura NON gratuíta.	58
5.11	Chamada á lectura non gratuíta e consulta de coins.	58
6.1	Primeiras pantallas da app: Sign Up e Welcome Slider 2.	66
6.2	Logcat ca creación da nova conta na Blockchain.	66
6.3	Logcat ca chamada ao Faucet.	66
6.4	Log da Blockchain ca transacción a cal engade a dose.	67
6.5	Engadir dose a un paciente dende app CDC.	68
6.6	Lectura do pasaporte do usuario creado, e Home CDC tras modificar o rexistro médico.	68
6.7	Detalles das doses recibidas, da wallet actual e do saldo de criptomoeda.	70
6.8	Probando as funcionalidades referentes á criptomoeda.	71
6.9	Logcat ca información referente á transacción de coins.	71
6.10	Funcionalidades extra non detalladas.	72
A.1	Diagrama de Gantt do Proxecto.	83
B.1	Modelo 1 non elixido de escritura na Blockchain.	87
B.2	Modelo 2 non elixido de escritura na Blockchain.	88
C.1	Alice consulta seu saldo de Coronacoins.	91
C.2	Mockups de Sign Up e da segunda Welcome Slider.	92
C.3	Mockups das pantallas iniciais posibles na app paciente.	93
C.4	Mockups das pantallas referentes á criptomoeda.	94
C.5	Mockups da lectura do pasaporte.	94
C.6	Mockups da importación do pasaporte e das doses en detalle.	95
C.7	Mockups da pantalla inicial e de perfil da app CDC.	96
C.8	Mockups de engadir datos de perfil e unha dose dende a app CDC.	96
C.9	Perfil CDC e Home CDC despois de transacción realizada correctamente.	97
E.1	Smart Contracts en Remix.	110
E.2	Inicio dun nodo na nosa Blockchain privada.	112
E.3	Logs do inicio dun nodo na nosa Blockchain privada.	112
E.4	Conectándonos á nosa cadea con Metamask.	113

E.5	Probando as contas da cadea con Metamask.	114
E.6	Probando a importar conta con Metamask e verificar saldo.	114
E.7	Enviando transacción dende Metamask e comprobando resultado.	115
E.8	Comprobando os logs da Blockchain tras o envío da transacción.	115
E.9	Primeiras probas mocks no despregue de Smart Contracts dende Remix.	116
E.10	Despregue e proba mock dende Remix dos primeiros Smart Contracts.	116
E.11	Probas mock de engadir rexistro e ler gratuitamente o mesmo.	117
E.12	Resultado da proba mock de engadir rexistro de paciente á Blockchain.	117
E.13	Log de Remix do despregue do contrato na nosa cadea.	118
F.1	Descargar ABI dende Remix.	122
F.2	Consola Firebase: confirmación da correcta creación da nova conta.	123
F.3	Log da Blockchain ao chamar ao Faucet.	123
F.4	Log da Blockchain ca transacción de envío de Criptomoedas.	123
F.5	Home da app paciente tras ser vacinado por primeira vez.	124

Índice de Táboas

2.1	Táboa comparativa do proxecto europeo de pasaporte COVID-19 co sistema proposto neste TFG.	16
-----	---	----

Introdución

1.1 Contextualización

DURANTE as primeiras semanas do 2020, a enfermidade coñecida como **COVID-19** (**Coronavirus Disease 2019**) comezou expandirse dende China cara máis de 20 países, creando así unha emerxencia sanitaria mundial declarada pola **OMS** (**Organización Mundial da Saúde**) o 20 de xaneiro dese mesmo ano [1]. Dende aquela, leva transcorrido ano e medio, e as consecuencias que a pandemia deixa na vida de todas as persoas ao redor do mundo son notábeis, véndose reflexadas a todos os niveis e áreas da nosa sociedade. De feito, orixináronse nela unha serie de fases, entre as que destacamos as actuais e futuras etapas de vacinación e post-vacinación da poboación. Con elas preséntase un problema co cal lidar: a adecuada trata da información respecto ás vacinas e a óptima utilización da mesma. Isto, sumado á clara precariedade tecnolóxica do sistema sanitario e ás cuestionábeis solucións adoptadas en certos países como as tarxetas ou pasaportes físicos de vacinación, anima pensar nunha solución que trate xa non só os datos respecto ás vacinas, se non todos aqueles que xiren en torno á **COVID-19**, dunha forma máis moderna, óptima, descentralizada e segura.

É por todo isto polo que se propón o seguinte traballo, o cal consistirá no deseño e depregue dunha infraestrutura que mediante a tecnoloxía **Blockchain** sexa capaz de almacenar a información médica do cidadán referente á **COVID-19**, centrándose sempre na integridade, fiabilidade, inmutabilidade e trazabilidade da mesma, grazas aos beneficios inherentes á **Blockchain** e á criptografía. Por outro lado, o sistema a desenvolver fará uso dos datos recollidos a través dunha aplicación móbil multi-plataforma co fin de mellorar accións á orde do día e que polo ben xeral da sociedade necesitamos optimizar, como son os controis de acceso, a detección precoz de situacións de risco e a análise dos datos médicos, xa non só da vacinación, senón de todos aqueles relacionados ca pandemia. A maiores, propónse un sistema de incentivos para que o cidadán de a pé faga uso da mesma, incorporando seus datos médicos referentes á **COVID-19** na **Blockchain**. Este sistema vese traducido na creación dunha

criptomoeda na propia Blockchain privada ca cal recompensar á persoa que use o sistema.

A utilidade deste traballo é maiúscula, ao ocuparse de varios dos problemas presentes e futuros da sociedade actual marcada fortemente pola COVID-19, como son as ben ditas etapas de vacinación e post-vacinación, as cales por outro lado serán con toda probabilidade cíclicas e anuais, polo que se axilizarán e mellorarán as tarefas de almacenamento de información crítica (vacinas, probas, reaccións adversas) que darán lugar do lado do paciente ou cidadán, á posibilidade de posuír un certificado dixital seguro co que demostrar a súa inmunidade, como do lado das autoridades sanitarias a unha mellor trata e análise da información, ao estar almacenada de forma compacta nun só sistema homoxéneo, seguro e infranqueable como é unha cadea de bloques.

1.2 Utilidade real do sistema

Vendo pois as situacións actuais e futuras que se presentan ao redor desta etapa, as posibilidades que se nos presentan son enormes debido ao gran potencial dun sistema coma o descrito. Polo que para explicar agora o seu impacto, non xa só no ámbito tecnolóxico, ao crear unha innovadora solución, se non tamén no social, con aquelas consecuencias derivadas do sistema que axuden á mellora da situación actual ca pandemia, propónse a seguinte sección, onde se detalla un uso completo do sistema onde quedan evidenciados os beneficios de cara a diferentes sectores da sociedade e a mellora de diversas situacións cotiás actuais que se ven limitadas pola convivencia ca pandemia, así como tamén se explica de forma detallada a idea proposta para unha mellor comprensión da mesma.

Caso práctico real

Imaxinemos que unha poboación está a ser vacinada masivamente, como por exemplo España nestes intres, e onde a información das vacinas recae unicamente nas entidades médicas estatais e da comunidade autónoma, aínda que a maioría de actividades sociais, dende museos e concertos ata viaxes, son xestionados por empresas privadas, as cales dependen directamente das medidas vixentes, acordadas en referencia a esta información médica, a cal non accesible polo cidadán de a pé. A idea é clara, como ben explican os fundamentos da Blockchain 4.1, a chave recaería en eliminar os intermediarios grazas á descentralización, para axilizar e securizar así o proceso de verificación da información médica, e neste caso, ademais, axudar á sociedade a retomar unha certa normalidade ao poder eliminar certas restricións grazas ao tratamento adecuado desta información.

Ca existencia dun sistema onde estiveran presentes todos estes datos críticos, de forma segura, inalterable e asociados de forma unívoca ao cidadán, e seguindo o modelo actual onde dita información só é engadida por autoridades certificadas, podería ser utilizada por calquera

que o necesitara sempre que o cidadán o consentira, e conseguir así maiores beneficios. Por exemplo, imaxinemos o caso dunha viaxe ás Illas Canarias: en primeiro lugar, as restricións de mobilidade poderían ser facilmente redistribuídas na poboación dependendo de se consta a existencia de inmunidade ou de probas negativas recentes, como xa se fai en varios países esixindo *PCRs* (*Polymerase Chain Reaction*) para certas accións ou inmunidade para outras, como é o caso de España [2] ou Gales [3].

Unha vez superada dita barreira da restricións da viaxe grazas á demostración de inmunidade ou inexistencia de virus co pasaporte descrito, chegaría a parte do beneficio mutuo entre cidadán e empresa, os cales establecerían unha relación de simbiose ao ter que: unha empresa calquera, que podería ser por exemplo a aeroliña encargada dos vós ou a cadea de hoteis que aloxa aos viaxeiros, cando sabe de forma inequívoca que os clientes ou ben presentan inmunidade por vacina ou ben constan dunha *PCR* negativa, pode aliviar as medidas restritivas de diferentes formas sen supor isto un risco sanitario, como xa se está probando e estudando en casos reais como [4], [5] ou [6], polo que esta *praxis* de eliminación de restricións, grazas ao ensino do pasaporte ca información das vacinas e probas, poderíase tamén levar a cabo. Con isto, a aeroliña ou o hotel beneficiaríase de diversas maneiras: tanto aliviando as medidas para os clientes, creando así neles unha mellor actitude e predisposición, ao xerar ambientes similares aos de antes da pandemia, así como permitindo un maior aforo, tendo polo tanto unha maior potencial ganancia económica, como por último, grazas ao mecanismo de incentivos descrito. Un caso a contemplar para beneficiar ao cidadán sería aquel onde dita empresa ou sector a aceptara para certas transaccións, por exemplo, no caso deste hotel ou aeroliña, para certos descontos en algúns servizos, e así fomentar no cidadán a actitude de compartir os seus datos médicos referentes á *COVID-19* para mellorar a loita fronte a pandemia e de forma análoga verse beneficiados tanto empresas como clientes de forma económica.

É dicir, aínda que este caso sexa unha mera suposición, podería ser unha situación real e factible no caso de ter este sistema vixente nuns meses, onde se faría un uso da totalidade de funcionalidades implementadas no proxecto e grazas ás cales o cidadán, as autoridades sanitarias e as empresas privadas se verían beneficiadas, cada un nun aspecto diferente, polo simple uso de dito pasaporte ou certificado dixital de vacinación.

1.3 Implicacións éticas e morais

Unha vez que sabemos a situación na que se presenta este proxecto, así como a súa potencial utilidade e entendemos o seu futuro funcionamento, paga a pena deixar momentaneamente de lado os retos e desafíos científicos e tecnolóxicos que tanto a *COVID-19* en xeral, como a idea dun pasaporte ou certificado dixital de vacinación supoñen, para discutir a serie de implicacións éticas e morais que esta idea trae consigo, as cales están sendo moi cuestio-

nadas a día de hoxe [7], e máis aínda ao ver a intensificación das campañas de vacinación e a súa efectividade contra o virus, dando lugar ao pensamento da recuperación da antiga normalidade o máis rápido posible [8]. Xa que aínda que un principio fundamental da medicina ética é intervir o menos posible nas liberdades das persoas, a COVID-19 está a producir certas restricións de dereitos fundamentais, e a existencia dun pasaporte ou certificado de inmuni-
dade podería dar lugar a diferencias e desigualdades en canto a estas restricións se tratara [7], como de feito xa se pode observar en diferentes ambientes de ocio en diferentes países ao longo do mundo que están optando pola adopción deste tipo de medidas (e.g., Israel, EEUU, Dinamarca) [8].

O uso destes pasaportes ou certificados para o establecemento ou non de restricións a certa poboación supón un debate ético e moral enorme, así como trae consigo unha serie de problemas asociados, e máis nas primeiras etapas destas novas fases onde os recursos son limitados (vacinas, probas...), os *ratios* de vacinación son variables en moitos países dependendo do núcleo social ao que se pertenza e os resultados médicos das mesmas aínda non están totalmente corroborados, así como a vacinación non está nin obrigada nin apoiada por toda a poboación [8]. Diversos artigos amosan a división de opinións respecto este aspecto, pero certo é que canto máis tempo pasa e máis evoluciona a pandemia, os beneficios para contemplar o seu uso comezan a aparecer, debido ao aumento exponencial de acceso a vacinas ao longo do mundo, á aparición dos primeiros estudos que confirman a eficacia das vacinas, polo menos contra episodios de enfermidade grave ou morte, e incluso á idea de que a súa presenza axudaría a mellorar a opinión social da vacinación así como o *castigo* pola irresponsabilidade de rexeitala, xa que, con este sistema, a elección da negativa a vacinarse suporía consecuencias traducidas no non outorgamento de beneficios, feito que asemella xusto [8].

A conclusión que se pode obter lendo as referencias, así como analizando as palabras do profesor na Universidade de Pensylvania e membro do comité médico para a COVID-19 de Joe Biden, Dr. Zeke Emanuel, é que ao uso destes mal-chamados pasaportes (o estritamente correcto sería nomealos como certificados) [9], non é inherente ningún tipo de discriminación, xa que a eliminación ou non de restricións ao cidadán dependerá só dos riscos sanitarios que isto signifique [7]. O Dr. Emanuel postula que o principio ético a seguir é aquel que se mencionou ao comezo da sección, o cal propón que se deberán manter as situacións no nivel menos restritivo posible, polo que, se para unha parte da poboación podemos mitigar ou incluso retirar estas restricións, todo isto co respaldo dunha xustificación médica, o uso dun pasaporte ou certificado que o permitira facer dunha forma organizada, segura e fiable, só sería un mecanismo máis para poder cumprir este principio [9], sempre e cando por suposto, se cumpriran certos límites coma o control ao acceso a dita información e o uso non lucrativo da mesma, se garantiran certos dereitos como son o acceso igualitario á posibilidade da realización de probas ou vacinación e a non obriga por parte do estado ao uso do mesmo [8].

1.4 Obxectivos

Queda agora, despois de explicar a situación na que nace o proxecto e de detallar a súa utilidade e implicacións sociais, definir claramente os obxectivos que queremos acadar ca realización do mesmo, centrándonos nesta sección na súa parte máis técnica, a cal consta claramente de 3 partes, onde en cada unha delas haberá que crear dende cero unha funcionalidade final ou obxectivo do sistema. Aínda que cada parte terá os seus diferentes sub-obxectivos, todas elas teñen en común que acaban levando á realización dun obxectivo xeral, polo que podemos dicir que, teríamos tres grandes liñas ou obxectivos a conseguir, estando posteriormente cada un deles sub-divididos en diferentes tarefas. Estes obxectivos fundamentais son:

- **A creación dun sistema descentralizado con intelixencia automatizable** que permita incorporar ao mesmo todos aqueles datos médicos e de vacinación máis relevantes, asociados sempre de forma unívoca e segura ao cidadán, con datos como poden ser as datas das doses recibidas, o tipo de vacina, o centro na que foi administrada (cos seus datos asociados) ou as últimas probas **COVID-19** realizadas, á súa vez co seu tipo, data e resultados.
- **A creación dunha criptomoeda**, despregada sobre o sistema anteriormente mencionado, que terá como obxectivo actuar a modo de incentivo para o cidadán, sendo adxudicada certa cantidade dela á conta de cada cidadán por diversos motivos: uso do pasaporte de vacinación, inoculación dunha dose, realización dunha proba, etcétera.
- **A creación dunha **Decentralized application (Dapp)** móbil**, deseñada e implementada nun entorno **multi-plataforma** (é dicir, que permita ser despregada tanto en iOS como en Android con facilidade), a cal se comunique co anterior sistema para facilitar e acercar o uso de toda a infraestrutura detallada anteriormente ao cidadán, permitindo realizar accións como por exemplo: lecturas de pasaportes, actualización dos mesmos ao acudir a centros médicos ou de probas, envío e recepción da criptomoeda entre usuarios, etcétera. Á súa vez, habería que crear a versión homóloga para as autoridades, con maiores permisos e funcionalidades á hora de escribir no sistema, polo que aínda que en concepto se trata dunha app, haberá dúas versións diferentes da mesma, con diferentes funcionalidades, diferenciándose entre elas na capacidade ou non de escritura no sistema dependendo de se se trata da versión para as autoridades ou para os cidadáns.

Por último, e no que respecta xa non só aos obxectivos técnicos, se non ao carácter do propio traballo, haberá que salientar o obxectivo a maiores de **cumprir os requisitos dun TFG de dobre mención**. No proxecto, deberanse abordar conceptos técnicos específicos referentes a ambas mencións, é dicir a **Tecnoloxías da Información** e a **Enxeñaría de**

Computadores, para xunto cas competencias obtidas grazas á realización de ambas, ser capaces de combinar ambos elementos para a elaboración dunha solución compacta, que se vexa potenciada por dita dualidade conceptual.

1.5 Estrutura da memoria

A memoria intentará reflexar da forma máis fiel posible o traballo realizado neste *Traballo Final de Grao*. Os primeiros capítulos terán un carácter máis teórico, para progresivamente ir acadando unha perspectiva máis práctica a medida que se avanza na memoria, como cronoloxicamente tamén aconteceu durante a realización do proxecto. Polo que despois deste primeiro Capítulo 1 de introdución, onde contextualizamos o proxecto na situación actual, explicamos a súa utilidade exemplificando un uso completo do mesmo, e discutimos as súas implicacións tanto a niveis sociais, como tecnolóxicos, aclarando tamén os obxectivos do mesmo, imos abordar os seguintes temas:

- **Estudo de aproximacións relacionadas:** Co Capítulo 2 tentaremos analizar todas aquelas solucións presentes a día de hoxe e que están relacionadas con algunha das temáticas abordadas no proxecto, para despois poñelas en conxunto para analízalas e facer unha crítica das mesmas, onde tras detallar a nosa proposta, comparáremola con ditas solucións e sacaremos as primeiras conclusións.
- **Primeiras fases do proxecto:** No Capítulo 3 repasaremos as dúas primeiras fases da metodoloxía incremental utilizada na realización do proxecto, aquelas referentes ao **Análise e Deseño** do mesmo. Onde, respectivamente, detectaremos os requisitos e funcionalidades a satisfacer polo noso traballo, así como deseñaremos as diferentes solucións para levar isto a cabo. Finalizaremos o capítulo ofrecendo a proposta xeral do sistema a crear, explicado dende unha vista xeral e acompañado da metodoloxía a usar para levala a cabo.
- **Fundamentos tecnolóxicos do proxecto:** No Capítulo 4 explicaremos todos aqueles conceptos teóricos necesarios para unha mellor comprensión do proxecto, os cales se deberon perfeccionar á hora de poder realizar o mesmo, debido ao seu descoñecemento inicial e gran complexidade. Unha vez detallados, comentaranse na última parte do capítulo todas aquelas ferramentas usadas no traballo, explicándoas brevemente e xustificando o seu uso fronte outras alternativas.
- **Segunda metade do proxecto:** Nos Capítulos 5 e 6 abordaranse respectivamente as últimas dúas fases do ciclo incremental utilizado, referentes á **Implementación e Probas**. Cada capítulo tratará ambas partes de cada un dos dous grandes módulos do traballo, cada un máis fortemente relacionado cunha das dúas mencións, detallando os

conceptos máis importantes e dando sempre unha explicación concreta do traballo elaborado acompañado dunha exemplificación visual con capturas da aplicación ou anacos do código da versión final.

- **Parte final:** No derradeiro Capítulo 7, farase unha recapitulación de todo o proxecto, sacando conclusións a diferentes niveis e comentando as incidencias atopadas e as principais liñas de investigación futuras para despregar o proxecto nun hipotético entorno de produción.
- **Apéndices:** Debido á gran extensión do proxecto, xa non só pola connotación da dobre mención, a cal implica poñer o dobre de traballo nel, senón tamén pola gran complexidade e tamaño do mesmo, houbo moita información que non puido ser incorporada á parte central da memoria, polo que utilizaremos os apéndices para poder ensinar e xustificar diferentes tarefas, decisións e información complementaria que dotan tanto ao TFG en xeral, como á memoria, dun maior valor. Exemplo disto poden ser todos aqueles apéndices con case o mesmo nome e estrutura que varios capítulos da memoria (por exemplo os Apéndices E e F, relacionados cos Capítulos 5 e 6 respectivamente), que se utilizarán para ou ben amosar aqueles conceptos de ditos capítulos que non foron elixidos para mostrar en primeira instancia na memoria, ou ben para complementar aqueles conceptos expostos na mesma. Por outro lado, un dos apéndices, en concreto o primeiro, o Apéndice A, encargárase de amosar a planificación do TFG (cun *Diagrama de Gantt*) e explicar as súas fases, así como de detallar o seu custo final. E por último, no Apéndice G, falaremos do carácter *Open Source* do proxecto así como daremos un *link* para consultar todo o código íntegro do mesmo, subido a un repositorio *Git* público baixo unha licenza de código libre.

Estado da arte

Neste capítulo levarase a cabo un estudo ou análise de antecedentes e alternativas relacionadas co traballo proposto. Debido á súa composición modular, farase un resumo separado por cada unha das partes do proxecto, indo dende unha aproximación máis xeral, co estudo do concepto ou idea na que se basea o TFG, neste caso a da creación dun pasaporte ou certificado de vacinación, até finalizar con partes cada vez máis específicas e relacionadas ca tecnoloxía, neste caso Blockchain aplicada ao ámbito socio-sanitario, para rematar facendo unha análise conxunta de todas elas, unha análise crítica das mesmas e unha proposta que as complementa. Por último, compararemos dita proposta ca solución máis recente e similar.

2.1 Pasaportes dixitais ou certificados de vacinación COVID-19

Antes de analizar os traballos e estudos técnicos referentes á parte de Blockchain ou desenvolvemento móbil, cabe mencionar a situación actual de investigación e discusión global sobre a posible instauración dun pasaporte ou certificado de vacinación. Aínda que non hai ningún sistema oficial nin estandarizado listo para ser usado, hai varias propostas saíndo á luz, a día de hoxe en Maio de 2021. Entre elas podemos destacar en primeiro lugar o pasaporte de vacinación proposto para Galicia e España, o cal está sendo desenvolto a día de hoxe e temos só a información dispoñible nos medios de prensa como [10] ou [11]. Outras institucións a nivel mundial como a Comisión Europea [12], están traballando en proxectos similares para un futuro próximo, que estudaremos na última sección 2.6 do capítulo, e en países como EEUU adoptáronse de momento medidas opcionais de empresas privadas como a comentada anteriormente [13]. Sen embargo, hai países que xa implantaron pasaportes de vacinación [14] de diversos tipos como China [15], co seu sistema de identificación mediante códigos QR (*Quick Response*), Israel [16], co *Green Pass*, ou Dinamarca [17], cunha app móbil para demostrar inmunidade. Á súa vez, moitos outros están a traballar para facer oficiais e despregar o antes posible as súas propostas, como por exemplo Australia, Suecia ou Alemaña [14].

XUNTA DE GALICIA **VACINACIÓN COVID**
XUNTA DE GALICIA

Certificado COVID / Certificado COVID / COVID Certificate

Consellería de Sanidade - Xunta de Galicia

Nome / Nombre / Name: USUARIO DE PROBAS APELIDO1 APELIDO2 DNI: 00000000X Data de nacemento / Fecha de Nacimiento / Date of birth: 24/07/1966 CIP: XXXX00XXXX0010
Enfermidade / Enfermedad / Disease: SARS-CoV-19 Vacina / Vacuna / Vaccine: COVID
Data / Fecha / Date: 04/02/2021 Laboratorio / Laboratorio / Laboratory: Pfizer/BioNTech Lote / Lote / Batch: EJ6796 Centro / Centro / Centre: CENTRO DE DIA DA CRUZ VERMELLA ESPAÑOLA Data / Fecha / Date: 04/01/2021 Laboratorio / Laboratorio / Laboratory: Pfizer/BioNTech Lote / Lote / Batch: PFIZER-LOTE1 Centro / Centro / Centre: CENTRO DE DIA DA CRUZ VERMELLA ESPAÑOLA

Figura 2.1: Certificado COVID-19 proposto pola Xunta de Galicia / Sergas - Abril 2021 [11].

2.2 Blockchain e Dapps

A Blockchain e consigo as **DLT** (**Distributed Ledger Techonology**), son unhas novas e disruptivas tecnoloxías introducidas na última década as cales non paran de medrar [18] en gran medida polo seu enorme potencial e as garantías de transparencia de datos, confianza na veracidade dos mesmos, privacidade, inmutabilidade e seguridade que ofrecen [19]. Xa non só son a base das criptomoedas, a día de hoxe en auxe, destacando os proxectos tanto de Bitcoin [20] como de Ethereum [21], o cal será chave neste traballo, se non tamén son usadas por cada vez máis sistemas co fin de xerar aplicacións descentralizadas que utilicen Blockchain para que os seus beneficios axuden a complementar e potenciar estes traballos. Exemplos disto podemos velos en sistemas de votación por Blockchain [22] [23] [24], melloras nas cadeas de suministro [25] e en proxectos onde se deseña e constrúe dende cero unha infraestrutura Blockchain privada para despois crear **Dapps** (**Decentralized application**) webs ou móbiles persoais que a utilicen, como vemos en [26], TFM da Universidade de Basilea o cal consiste no despregue dende cero dunha Blockchain privada que utiliza conceptos como ERC20 Tokens como criptomoedas, Raspberry Pies como **nodos mineiros** e unha páxina web para comunicarse ca

cadea de bloques a modo de *Dapp*. No que respecta ao *framework* utilizado neste traballo para implementar a *Dapp* que se comunice co Blockchain, *Kotlin Multiplatform Mobile (KMM)*, comentar que debido ao seu recente lanzamento en decembro de 2020 como versión *alfa*, practicamente non se atopan artigos nin proxectos que traten sobre isto, salvando excepcións como [27] ou [28], os cales foron útiles para a iniciación na tecnoloxía, aínda que como a información dispoñible sobre este *framework* é moi limitada neste momento, as consultas na súa propia documentación [29] son de onde podemos extraer maior información. Por riba, no que respecta a *KMM* relacionado con *Dapps* e/ou Blockchain, non se atopou ningún traballo coñecido.

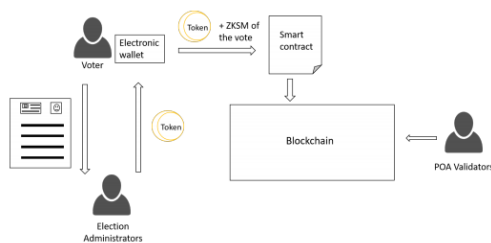


Figure 3: Blockchain-based voting system

Fig.5 represent different smart contracts that are going to be used in our voting system. Those contracts have different functionalities and they add the business logic to our Blockchain system.

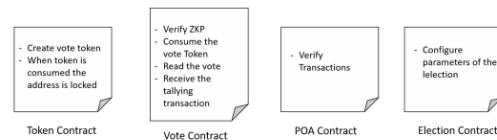


Figure 5: Smart Contracts of the system

Figura 2.2: Diagramas overview das interaccións co Blockchain e dos Smart Contracts presentes nun sistema de votación seguro [24].

2.3 Blockchain no ámbito sanitario

Acompañando este crecemento das Blockchains, ven tamén o crecemento e a aparición de máis solucións que as utilicen nun ámbito sanitario, como podemos ver en [30], onde se diseña un sistema intelixente baseado en Blockchain para o suministro e supervisión de vacinas xenéricas. Tamén observamos en [19], un dos traballos referentes e modelos a seguir para este TFG, como se diseña tanto un sistema baseado en Blockchain para a trazabilidade e xestión adecuada de información médica referente á diabetes, como unha app *Android* para as consultas do paciente como tamén un incentivo a modo dunha criptomoneda, todo isto nunha *Testnet* xa existente de *Ethereum*. En [31] atopamos un estudo teórico detallado da solución adoptada en China para mellorar a trazabilidade das vacinas grazas ás tecnoloxías Blockchain. E en canto a un proxecto práctico, onde atopamos unha serie de *Smart Contracts* programados en *Solidity* implementando un sistema de rexistros médicos con incentivos a base de *tokens* para os pacientes, podemos atopar un gran traballo en [32], o cal foi tamén un dos grandes puntos de partida para este proxecto.

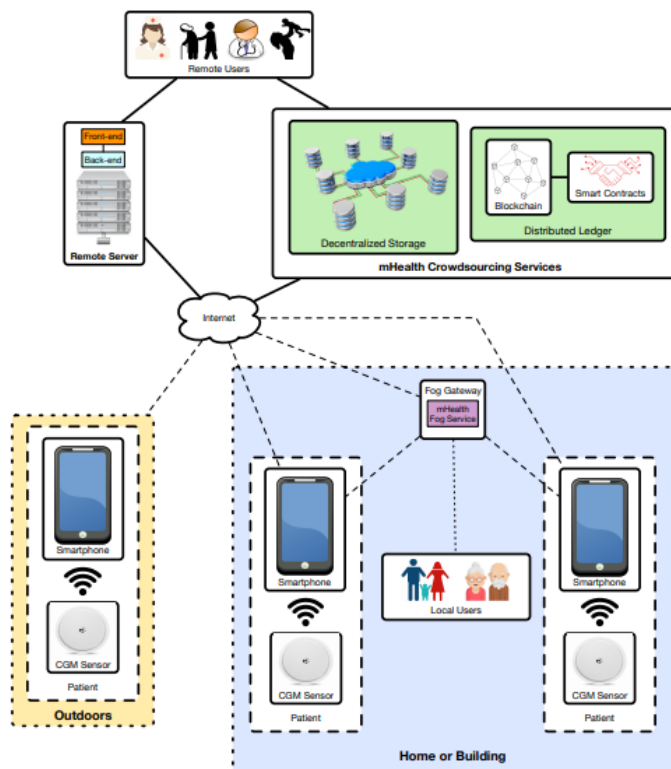


Figura 2.3: Arquitectura de comunicacións onde se diseña un sistema Blockchain con incentivos relacionado co ámbito sanitario [19].

2.4 Blockchain e COVID-19

No que respecta aos proxectos que mesturen conceptos de Blockchain ca recente emerxencia sanitaria da COVID-19, os cales se acerquen á proposta de diferentes mecanismos de pasaportes dixitais ou *e-certificates* de vacinación temos [13], unha app móbil usada en EEUU para verificar e xestionar de forma dixital a túa vacinación grazas á colaboración entre cidadán e centros de inoculación. Lemos tamén en [18] unha análise teórica de diversas solucións actuais ou recentes onde se empregou Blockchain para facilitar ou mellorar algunha situación motivada pola COVID-19, especialmente centrándose en solucións tanto de trazabilidade de contactos recentes con infectados, como por outra banda en pasaportes ou certificados dixitais de inmunización. É aquí onde se nomean outros traballos desta temática como [33], onde se propón un sistema baseado nunha Blockchain privada onde se despreguen pasaportes asociados aos cidadáns con información relativa á COVID-19, os calen serán accedidos por autoridades e sistemas sanitarios de diferentes países do mundo e tamén [34], onde se propón un sistema Blockchain dirixido polo goberno para almacenar certificados de anticorpos COVID-19, axudado todo este proceso por dispositivos IoT para a interacción co sistema.



Figura 2.4: Imaxe da app móbil usada en EEUU para axudar ao cidadán a xestionar a vacinación COVID-19 [13].

2.5 Análise, crítica e proposta

Analizando todas as anteriores e diferentes solucións, vemos como o enorme crecemento e potencial das tecnoloxías Blockchain, así como o seu encaixe na aplicación a solucións similares, confirma que é a base ideal para a partida deste proxecto. De feito, recentemente chegou a ser recoñecida como unha das 10 mellores tecnoloxías dispoñibles para mellorar a loita contra a pandemia [18]. Aínda así, vemos como moi poucos proxectos abordan a solución do pasaporte dixital de vacinación de forma directa, pois ben ou fan unha aproximación teórica ou ben na práctica deciden ir máis polos conceptos de trazabilidade das propias vacinas e o seu tratamento (conserva e distribución).

Por outro lado, no que respecta á pura parte de Blockchain, centrándose na creación dunha rede privada e despregue sobre ela de *Smart Contracts*, vemos que hai proxectos aínda que moi escasos xa que o máis habitual é atopar usos de *Testnets* xa despregadas ou incluso das redes públicas, e quedándonos naqueles que abordan a temática de creación e despregue dunha cadea de bloques propio, ningún deles fai referencia á COVID-19. Ben é certo que hai aproximacións médicas que incluso chegan explotar tamén a temática das criptomoedas como [19] ou [32], os cales supuxeron unha axuda fundamental pero que aínda distan da solución proposta.

Polo que vemos que por separado, en cada un dos tres módulos comentados na sección de obxectivos podemos atoparnos cunha serie, aínda que limitada, de grandes traballos, pero

con ningún que trate de acercar os tres conceptos de rede privada, criptomoeda e Dapp, e por enriba aplicar o resultado para a mellora de situacións motivadas pola COVID-19, polo que o que se propón neste traballo é o aproveitamento dos puntos fortes de cada unha das súas partes: Blockchain, Criptomoedas, Dapps e Desenvolvemento Multiplataforma (KMM), para nunha única e homoxénea solución, construír un sistema que axude á loita fronte a COVID-19 e á recuperación dunha nova normalidade social, próxima á que tiñamos antes da pandemia, ca creación dunha ferramenta que facilite aquelas tarefas dependentes da demostración de inmunidade ou falta de virus.

2.6 Solución europea

Unha semana despois da entrega deste traballo, a día 1 de xullo de 2021, entrará en vigor o certificado dixital europeo COVID-19 [35] en toda a Unión Europea. Esta sección, a través da Táboa 2.1, pretende comparar dita solución ca exposta neste *Traballo de Final de Grao*. O seu cometido será salientar as carencias de dito pasaporte e, por tanto, destacar as vantaxes do sistema proposto.

	Pasaporte Europeo	Pasaporte TFG
Código QR	Si	Si
Información médica	Vacinas (tipo, fabricante, número de doses e data), probas (tipo, data, lugar e resultado) e contracción da enfermidade (data).	Vacinas (fabricante, identificador único de cada dose, número de doses, data, lugar e centro onde se inoculou cada dose), probas (tipo, data, lugar, resultado, identificador da proba e centro de realización da mesma) e reaccións adversas (informe médico da reacción).
Información persoal	Nome e apelidos, data de nacemento, data de expedición, emisor do certificado e identificador único do emisor do certificado.	Nome e apelidos, data de nacemento, xénero, país, información de contacto, identificador persoal e identificador médico.
Criptografía	Si, usa chaves públicas e privadas para firmar e verificar o certificado.	Si, usa Criptografía Simétrica para enviar cifrado o pasaporte e para almacenar os datos na app. Úsaa tamén en todas as interaccións co sistema grazas ao uso da tecnoloxía Blockchain.

Almacenamento da información	Centralizado. Cada país ten unha base de datos cos datos de cada centro emisor ou CDC (<i>Centros para o Control e Prevención de Enfermidades</i>).	Descentralizado. O sistema será único e idéntico para todos os que o usen e estará distribuído polos nodos e contas existentes.
App móbil	Non, aínda que a UE <i>“axuda aos países a desenvolverlas”</i> .	Si, o proxecto trae consigo unha aplicación móbil multiplataforma (iOS e Android) co fin de facer uso do sistema de pasaportes deseñado.
Escalabilidade internacional	Non. Unicamente válido para Europa. Terá problemas ca centralización dos datos médicos, os cales por natureza están distribuídos, xa non só entre os organismos sanitarios nacionais, senón tamén de menor nivel (e.g., en España as comunidades autónomas, co Sergas en Galicia). Europa di que <i>“están traballando para garantir compatibilidade”</i> .	Si. Non dependerá de ningún sistema nin de ningunha compatibilidade con ningunha outra implementación. Calquera sistema sanitario ao longo do mundo poderá facer uso do mesmo e brindará unha solución única, global e compacta.
Sistema de incentivos	Non.	Si. Criptomoeda personalizada outorgada aos usuarios que fagan uso de dito sistema.
Seguridade	Non se detalla. Só se fala de <i>“bases de datos seguras”</i> e se menciona a Pasarela da UE, a cal conectará as bases de datos nacionais que conteñen as chaves públicas dos CDCs para verificar os pasaportes.	Si, úsase un sistema distribuído e infranqueable como é Blockchain. A información almacénase cifrada, o envío do QR faise tamén cifrado. Non hai SPOFs (<i>Single Point of Failure</i>), nin puntos centrais ou críticos, tanto físicos como virtuais, aos que atacar para causar denegacións de servizo ou caídas do sistema.

<p>Transparencia e código <i>Open Source</i>.</p>	<p>Non ao completo. Certo <i>oscurantismo</i> tecnolóxico á hora de concretar as tecnoloxías e ferramentas empregadas, así como detalles de deseño e implementación.</p> <p>Non é <i>Open Source</i>.</p>	<p>Total. Sistema accesible publicamente onde a integridade e a transparencia de todos os datos serán características fundamentais. A totalidade do <i>software</i> utilizado é <i>Open Source</i> e o proxecto tamén o será, polo que poderá ser verificado, depurado e mellorado por unha comunidade, eliminando así toda traza de opacidade. Dita característica será explicada en maior detalle no Apéndice G correspondente, onde, entre outras cousas, teremos o <i>link</i> de acceso ao repositorio contedor de todo o código do proxecto baixo licenza <i>Open Source</i>.</p>
--	---	---

Táboa 2.1: Táboa comparativa do proxecto europeo de pasaporte COVID-19 co sistema proposto neste TFG.

Análise e deseño do sistema

Neste capítulo abordarase o proxecto dende unha perspectiva máis teórica, detallando as súas fases de Análise e Deseño. Antes de comezar, como con calquera outro proxecto informático, houbo que delimitar os requisitos a satisfacer polo sistema para posteriormente deseñar os seus compoñentes, relacións e comportamento para acometer estas funcionalidades dun xeito óptimo. A diferenza dos seguintes capítulos, onde se tratan de forma separada as partes de Blockchain e Desenvolvemento móbil, decidiuse comentar estas primeiras fases de forma conxunta para ambas debido a que nesta etapa do proxecto, o punto de vista sobre o mesmo debía ser compacto e xeral, para así facer un mellor estudo e deseño do mesmo.

3.1 Análise

O primeiro a definir no noso proxecto serán os **requisitos** a cumprir e as necesidades a satisfacer. Para isto, haberá que enumerar as funcionalidades que queremos que o noso sistema ofrezca, así como organizalas dependendo do seu tipo. Tras isto, haberá que avaliar as diferentes opcións para acometer estas tarefas e elixir a mellor perspectiva. Imos pois, ver as prestacións que tras a fase de análise, onde se levaron a cabo as tarefas anteriormente mencionadas, se decidiu que o noso sistema tiña que brindar, enumeradas por orde de importancia.

Requisitos principais

- **Lectura dende app móbil do pasaporte:** A función fundamental será poder ler dende un móbil de calquera plataforma, o pasaporte [COVID-19](#) de calquera presente na app de calquera outro cidadán, consultando os datos sobre o mesmo na Blockchain, co único parámetro necesario da dirección da *wallet* do cidadán que nos amosa o pasaporte.
- **Escritura dende app móbil do pasaporte:** As autoridades con dereitos de escritura (concepto abordado en detalle na sección de deseño [3.2](#)), poderán escribir na Blockchain os datos necesarios para a existencia deste sistema de certificados dixitais. Sendo estes:

- Engadir paciente ao sistema.
 - Engadir dose ao paciente.
 - Engadir proba ao paciente.
 - Engadir reacción adversa ao paciente.
- **Modelo de incentivos por uso do pasaporte:** Terá que existir un sistema de incentivos baseados nunha criptomoeda personalizada creada na Blockchain privada que premie cunha porción da mesma aos cidadáns que usen a app para tanto ler, como deixar ler seu pasaporte. Á súa vez, cada vez que incorporen ao sistema unha dose, proba ou reacción tamén serán incentivados, de igual maneira que a primeira vez que o seu rexistro médico sexa engadido ao sistema. Nestes tres casos, haberá tres cantidades de incentivo diferentes, indo de menor a maior cantidade.

Requisitos secundarios

- **Envío cifrado do pasaporte:** Non só se poderá ler o pasaporte de forma presencial escaneando a app, senón que o código QR poderá ser enviado, de forma segura, máis concretamente cifrado de forma simétrica cunha contrasinal consensuada previamente entre ambas partes, e recibido para ser descifrado e procesado pola app de igual maneira que se se acabara de ler en persoa.
- **Trading ca criptomoeda:** Transferirase sen limitación a cantidade desexada da criptodivisa á dirección que se indique, habilitando un microsistema económico *intra-app*.

Fundamentos dos requisitos

Para conseguir todas estas funcionalidades anteriormente mencionadas, fará falta unha infraestrutura que as soporte, a cal estará baseada na:

- **Creación dunha Blockchain privada:** deseñándoa, implementándoa e despregándoa dende cero, adaptada ao sistema, é dicir cos parámetros óptimos para o mesmo. Conterá a información relativa aos rexistros médicos COVID-19 e á criptomoeda creada.
- **Creación dunha Dapp multiplataforma:** haberá que deseñar, implementar e despregar unha app móbil distribuída nun entorno multiplataforma, é dicir, que funcione cos mínimos cambios posibles tanto para Android como para iOS cun só desenvolvemento.
- **Creación dun Faucet personalizado:** Será necesario a creación dun *Smart Contract* o cal dote da criptomoeda asociada á Blockchain elixida de forma gratuíta (é dicir, sen coste de transacción) ás contas que o soliciten, de igual maneira que o fan os chamados *Faucet* [36] nas *testnets* públicas de Ethereum, como Ropsten [37] ou Rinkeby [38].

3.2 Deseño xenérico

Imos agora enumerar por orde cronolóxica as diferentes tarefas de deseño realizadas como comentar as decisións de deseño máis relevantes que houbo que tomar.

3.2.1 Arquitectura do proxecto

O primeiro que houbo que deseñar foi a **infraestrutura da app**, decidindo quen ía poder escribir na Blockchain e quen non, así como a organización da **Dapp** para os perfíles diferentes en canto ao uso da mesma. A opción elixida represéntase no seguinte diagrama 3.1:

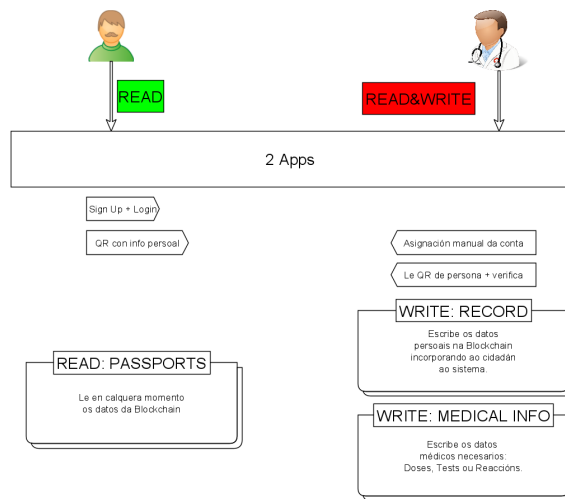


Figura 3.1: Modelo elixido de escritura na Blockchain.

Como vemos na figura 3.1, a decisión final consistiu na creación de dúas apps diferentes, aínda que idénticas no seu esqueleto xa que operan ambas sobre os mesmos datos e infraestrutura Blockchain, sendo unha para os cidadáns de a pé, ou pacientes, e outra para as entidades sanitarias ou autoridades. As súas diferencias son:

- **Pacientes:** Rexístranse co seu email e *password* para obter unha conta e loguéanse. Unha vez na app, deben introducir os seus datos persoais para xerar un QR que amosarán ás autoridades sanitarias. Non poderán crear nin modificar a súa información médica nin na app nin na Blockchain, xa que só terán permisos de lectura nela.
- **CDCs:** Contas previamente creadas e outorgadas individualmente a cada autoridade certificada cas cales se loguearán na app. As súas funcionalidades consistirán en ler os QRs dos cidadáns e, dependendo do seu estado, ou ben introducilos na Blockchain tras verificar a información persoal lida, ou ben para engadir información médica ao

registro médico xa presente na Blockchain, cunha Dose, Proba ou Reacción Adversa. Son as únicas contas con permisos de escritura na Blockchain.

En canto á elección deste modelo, comentar que foi a mellor opción de tres, amosando as outras dúas no Apéndice B, onde as detallamos xunto cos inconvenientes polos cales non foron elixidas. Comentar por último dous apuntes máis. O primeiro deles, a **xustificación** da elección do primeiro modelo. Debido ademais da inexistencia dos problemas das outras opcións, o primeiro modelo é unha aproximación a cal destaca pola súa sinxeleza á hora de ser implementada, a súa seguridade ao poderen ser feitas as escrituras só por organismos ou autoridades certificadas con contas asignadas de forma personalizada, non creadas baixo demanda, e as cales poden ser canceladas en caso de problemas con elas. É unha solución a cal verifica a información introducida polo cidadán, librando á súa vez ás autoridades do tratamento da información persoal do cidadán, tendo unicamente que verificala, non posuíla. Por último, non sobrecarga ningunha das dúas partes, ao ter cada unha, unha serie de tarefas repartidas, polo que asemella ser a máis práctica e eficiente. Por outro lado, **aclarar** que en todos aqueles casos onde se fale ou propoña a existencia de dúas apps, poderíamos estar a falar dunha soa app pero da incorporación de roles diferentes, que imitaría o caso de ter dúas apps distintas. A decisión de deseño foi optar por dúas aplicacións diferentes debido a:

- **Sinxeleza das apps:** Son dúas apps con escasas *activities*, con interfaces moi similares entre elas e funcionalidades básicas, polo que a realización de dous proxectos separados, ao seren deste tamaño, non supón unha carga de traballo extra significativa.
- **Inexistencia de funcionalidades comúns:** O modelado dunha soa app sería unha aproximación máis acertada se, ambas, a parte de parecerse esteticamente e basearse na mesma Blockchain, compartiran funcionalidades, para así non ter código repetido nelas. Pero ben é certo que a única función compartida é a lectura de pasaportes na Blockchain, e aínda esta, faise con fins diferentes: polos cidadáns para ver información doutros e dar/recibir incentivos, e polas entidades médicas para verificar e obter a información do paciente ao que instantes despois se lle vai modificar. O resto, como ben dixemos antes, son funcionalidades diferentes, ao ter obxectivos distintos.

Criptomoeda

Por último, haberá que falar do deseño das funcionalidades referentes á criptomoeda. Este será de menor dificultade, xa que só está presente no modelo da app dos cidadáns. Estes, poderán recibila e comerciar con ela, polo que no referente á criptomoeda, a cal estará separada nun *Smart Contract* diferente, si terán permisos de escritura na Blockchain. En canto aos detalles do deseño respecto ás operacións con ela, as cales serán consultar o saldo da mesma e enviar unha cantidade a outra *wallet*, detállanse no apartado 3.2.3.

3.2.2 Modelo de datos

Unha vez deseñadas as partes do sistema e os seus permisos, referentes tanto á app móbil como ás interaccións ca Blockchain, o segundo a modelar será o coñecido como o **modelo de datos**. Dito traballo consistiu na elección dos datos que imos almacenar na Blockchain, así como da relación entre eles e dos seus tipos. Baseándonos principalmente nos datos que certos mecanismos COVID-19 ao redor do mundo decidiron incorporar aos seus sistemas [10] [39], e en traballos que explotaban a dixitalización da xestión das vacinas en extensas poboacións, destacando [30] [32], ideamos o seguinte modelo de datos da Figura 3.2:

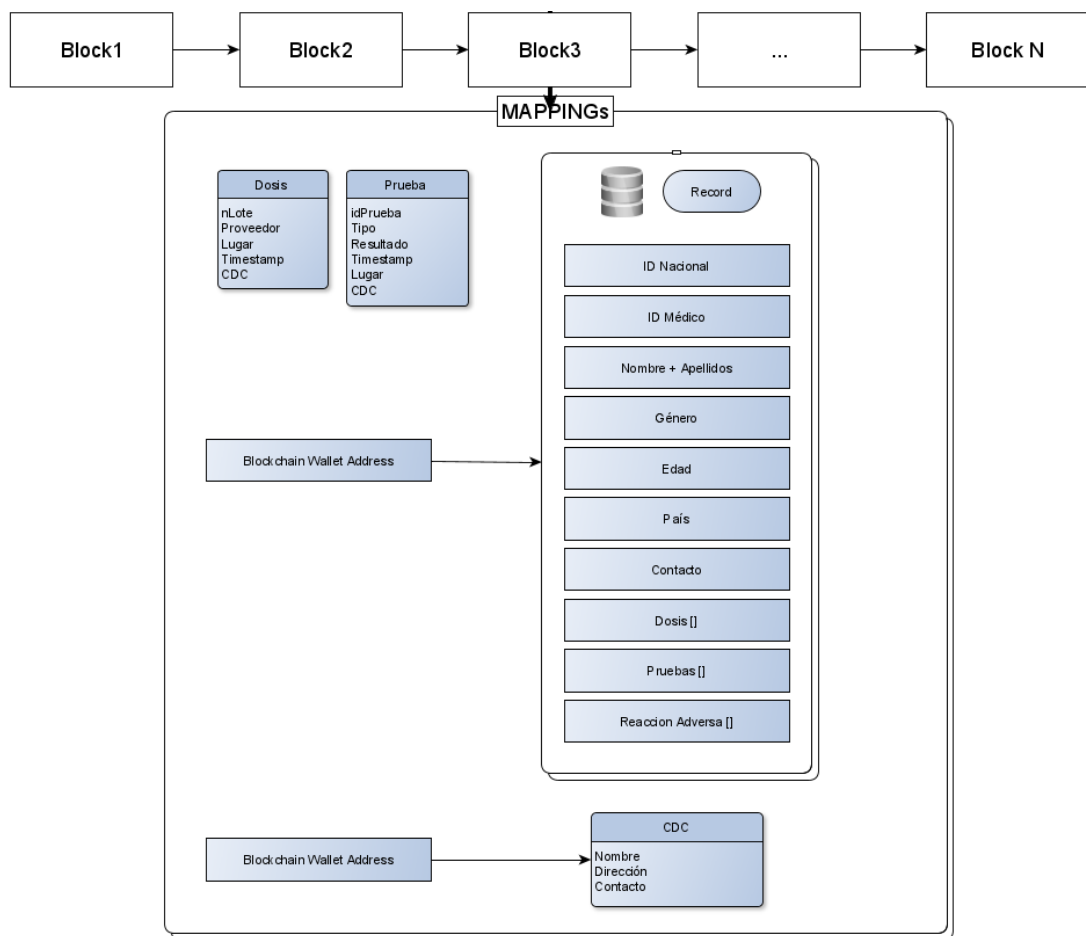


Figura 3.2: Modelo de Datos.

Destacar varios puntos para aclarar a organización dos datos e a elección da mesma:

- **Mappings:** A chave deste modelo recaerá no uso de *mappings*, os cales son asociacións entre dúas entidades ou tipos de datos. Máis en concreto con dous, o que asociará de forma inequívoca ao cidadán ca súa dirección de *wallet*, así como o segundo, máis

sinxelo, que servirá para ter información máis detallada dos CDCs, asociándoos á súa dirección única de *wallet*. Decídese incorporar pero simplificar moito os datos dos CDCs debido a que non é funcionalidade nin obxectivo principal do traballo, pero si que pode axudar a auditoría e trazabilidade de problemas ca xestión médica de vacinas ou tests.

- **Tipos de datos:** Crearanse 4 tipos complexos de datos, chamados **structs** en Solidity, que servirán por orde de importancia, para almacenar a información do rexistro médico (tipo **Record**), a información sobre as vacinas (Tipo **Dosis**), sobre os Tests (Tipo **Proba**) e sobre os CDCs (tipo **CDC**). Todos eles constan da información necesaria para cada caso, segundo se decidiu tras estudar diferentes deseños doutros proxectos xa instaurados en diferentes países, como por exemplo USA [39].
- **Strings:** O tipo de todos os datos será **String**, exceptuando por suposto os tipos complexos enumerados anteriormente e as direccións de *wallets*, as cales serán do tipo **address** de Solidity, xa creado para a adecuada trata de direccións de contas Blockchain. Esta decisión foi tomada xa que non hai ningún campo que claramente só conteña números ou caracteres, debido a que a maioría de identificadores, tanto nacionais como sanitarios a nivel mundial conteñen polo menos un carácter alfabético, e as datas en Solidity non teñen un tipo predeterminado. Ademais, o tratamento deles permite unha maior flexibilidade á hora de manexar os datos. Recalcar que o tipo *Reacción Adversa* é tamén un String, e asociado ao paciente teremos un array das mesmas, é dicir, de Strings.

3.2.3 Casos de uso e diagramas de secuencia

O seguinte a realizar sería identificar todos os casos de uso presentes no proxecto, así como acompañalos do seu diagrama de secuencia. Amósanse pois a continuación, xunto cunha explicación e xustificación dos mesmos.

Alice comeza ca app e crea QR cos seus datos persoais

Este caso de uso representado na Figura 3.3a, o cal será o primeiro en ser executado ao empezar ca **Dapp** móbil, constará de dúas fases: A primeira, tras a descarga da aplicación, consistirá na creación dunha conta (**Sign Up**) cun email e contrasinal asociadas e o **Log In** ca mesma. Despois de realizar isto por primeira vez, crearase en *Firebase* este usuario da app e na Blockchain unha *wallet* baleira. Despois disto, unha vez que a conta sexa verificada manualmente *via* email, medida tomada para evitar ataques de inanición de Ether, esta *wallet* recibirá Ether para poder comezar operar na Blockchain ao chamarse automaticamente á función *giveMeEther()*, a cal transferirá unha certa cantidade estándar de Ether á *wallet* solicitante, a modo de **Faucet**, como ben se explicou na sección anterior.

Tras isto, o usuario, neste caso Alice, poderá comezar usar a app cando desexe, introducindo os seus datos persoais e xerando un QR que actuará a modo de pasaporte provisional fronte as autoridades, o cal aínda non estará confirmado debido a non estar no sistema, pero será de gran utilidade cando acuda a calquera centro para escribir na Blockchain o seu rexistro. Este QR con datos persoais serán o que lerán as autoridades dende a súa app con permisos de escritura para verificar a información.

Alice é engadida ao sistema

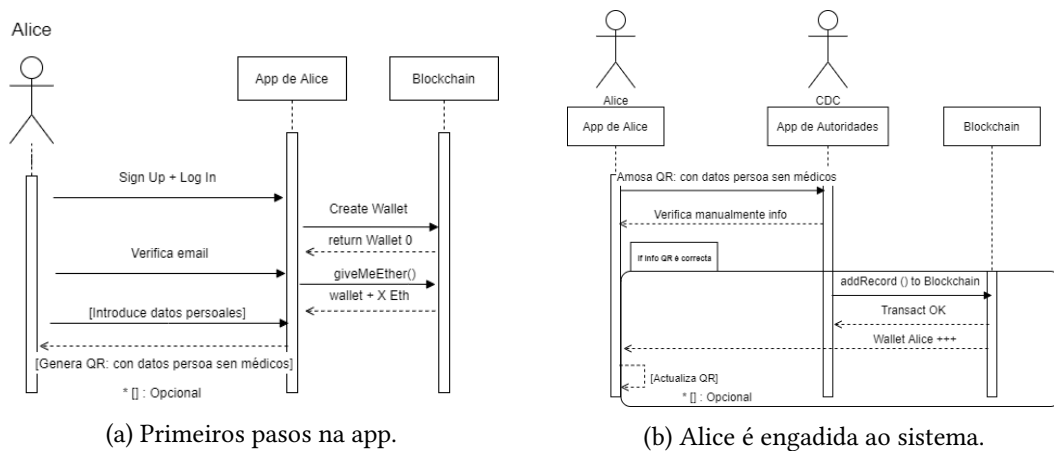


Figura 3.3: Casos de uso de Alice comezando coa app e sendo engadida ao sistema.

Neste caso de uso (Figura 3.3b), vemos como Alice, co QR previamente mencionado, acude a un centro autorizado en escrituras da Blockchain (CDC), no cal ensinará a app co seu QR. As autoridades, dende a súa app, accederán aos datos dispostos nel e corroborarán que son correctos ao comparalos coa documentación ofertada polo cidadán, mandándolle actualizalos no caso de que foran erróneos. Despois, estes CDCs escribirán na Blockchain a información chamando ao método *addRecord()*. Cando este se execute de forma satisfactoria, Alice recibirá un incentivo, en concreto o maior dos tres, como ben comentamos en apartados anteriores, por ser engadida ao sistema. Opcionalmente, Alice poderá actualizar a súa app refrescando a mesma, a cal fará unha chamada de lectura á Blockchain que devolverá o novo estado de Alice, agora verificada e presente no sistema.

Bob envíalle o pasaporte de Alice

No diagrama exposto na Figura 3.4a vemos representado como Bob lle envía á app de Alice, o seu pasaporte dixital (QR). Para elo, tras acordar previamente unha contrasinal con Alice, a app de Bob cifra simetricamente con esa chave (a cal é introducida manualmente por Bob) o QR e mediante unha app externa, este é enviado ata Alice. Alice só terá que, dende

a súa app, importar dito ficheiro cifrado recibido *via* externa, e introducindo manualmente a contrasinal, descifrar o ficheiro con ela para obter así o QR de Bob para poder lelo.

Bob le o pasaporte de Alice

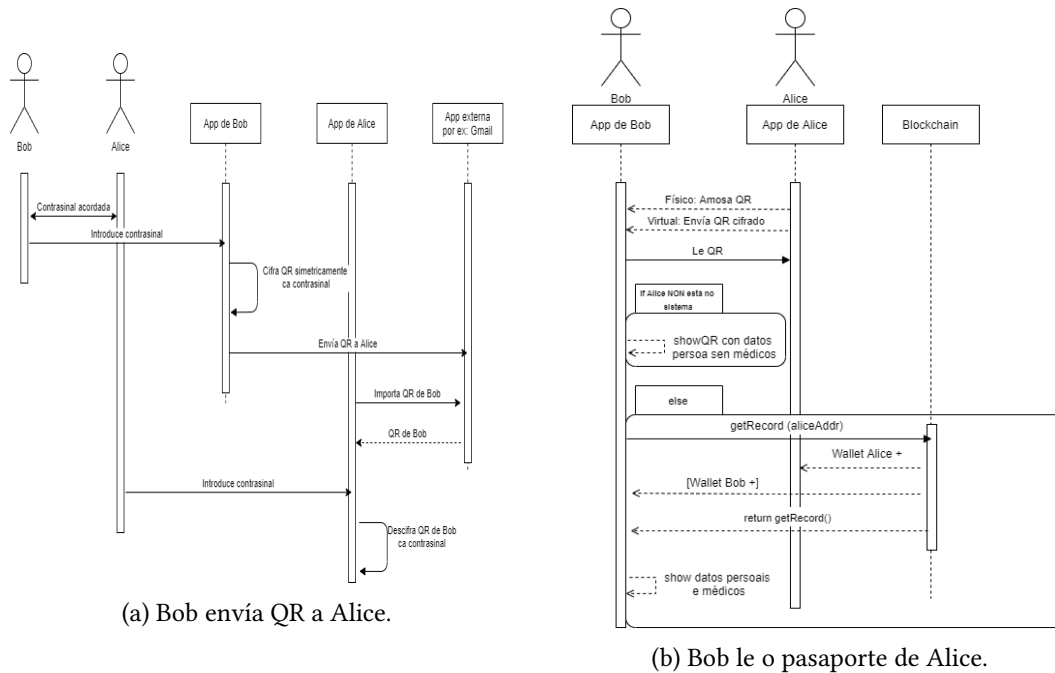


Figura 3.4: Casos de uso de Bob enviando seu pasaporte e lendo o de Alice.

Agora, neste caso representando na Figura 3.4b, temos a acción onde Bob le o pasaporte de Alice. Pode facelo de dúas maneiras: activando o lector de QRs e lendo de forma física o QR da app de Alice, ou importando o QR cifrado como se explicou anteriormente, para descifralo e acceder á súa información. Unha vez que temos o QR, comprobarase cunha chamada de lectura á Blockchain se Alice está no sistema, é dicir, se é paciente (*isPatient()*), equivalente a se a súa dirección de *wallet* foi engadida ao rexistro. Se é falso, mostrarase a información do QR de Alice, pero cun aviso claro de que non é información verificada nin está no sistema, simplemente son os datos persoais introducidos por Alice. En caso contrario, farase unha petición de lectura á Blockchain, a cal empregará parte do Ether de Bob. Esta consistirá en, enviando a dirección de Alice, ser capaces de obter toda a información dispoñible na Blockchain sobre ela, así como gracias ao Ether enviado, facer que a *wallet* de Alice sexa incentivada ca porción máis pequena da criptomoneda. Tras isto, e destacando que nalgún caso podería ser efectivo incentivar tamén ao que le, o lector recibirá os datos de Alice directamente da Blockchain de forma segura e mostraránselle por pantalla nun formato adecuado.

Alice recibe dose de vacina

Neste caso de uso representado na Figura 3.5a, as primeiras interaccións son moi similares ás do amosado na Figura 3.3b, de feito, adoitan ir de forma consecutiva no caso de ser a primeira dose. Alice chegaría ao centro de vacinación autorizado para escribir na Blockchain e as autoridades médicas verificarán que está no sistema de forma correcta. En caso negativo, habería que executar o caso de uso de introducir a Alice no sistema, ilustrado previamente na figura 3.3b, e en caso positivo, despois de inocular a dose ao paciente, dende a app do CDC, chamaríase á función do *Smart Contract* do rexistro médico na Blockchain *addDosis()*, para escribir os datos da vacina recibida. Cando a transacción fora satisfactoria, Alice recibiría dous incentivos de criptomoneda, o máis baixo debido á lectura do seu pasaporte, e o de tamaño medio por facer unha transacción de escritura no seu rexistro médico. A app do centro recibiría a confirmación da mesma transacción na súa app, con datos como o número de bloque escrito e o *hash* da transacción e Alice opcionalmente podería actualizar o estado do seu QR.

Alice sofre unha reacción adversa á vacina

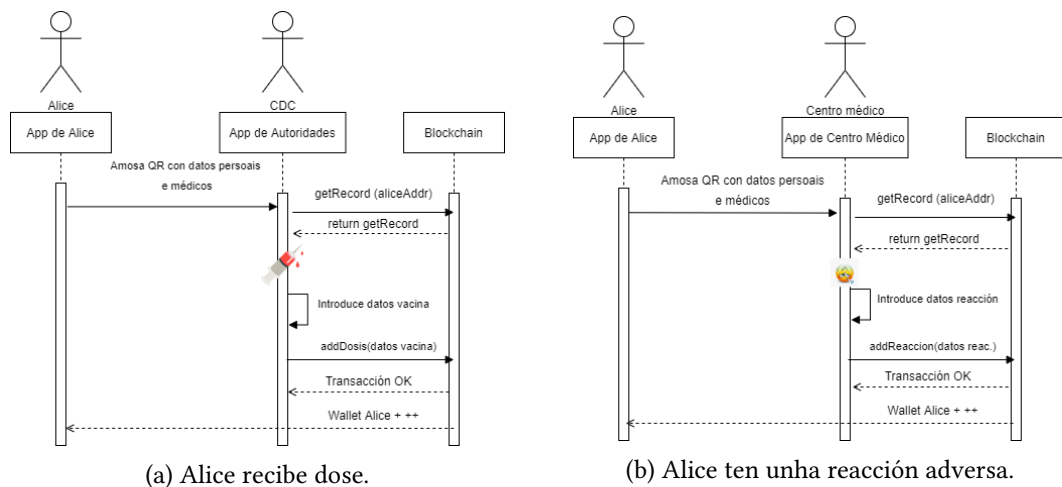


Figura 3.5: Casos de uso de Alice recibindo dose e tendo unha reacción adversa á mesma.

Este caso de uso ilustrado na Figura 3.5b será idéntico ao anterior. Só cambiará que tras a verificación e consulta da información de Alice na Blockchain, chamarase a un método diferente, esta vez a *addReaccion()*, o cal escribirá no rexistro médico de Alice unha nova reacción adversa, redactada e corroborada pola entidade médica, así como tamén escrita na Blockchain por ela dende a súa app. Destacar que esta decisión de deseño débese a dúas xustificacións: primeiro, porque deixar do lado do paciente o autodiagnóstico de reaccións adversas á vacina sen a supervisión dun profesional podería dar lugar a falsos positivos, efectos placebo e información enganosa, e por outro lado, a decisión de modelalo como un array de Strings, débese á

facilitación de tarefas posteriores de auditoría ou análise, ao ser moi doado contabilizar cantos cidadáns tiveron reaccións ($[size] > 0$) e os que as tiveron, cantas foron ($[size]$).

Alice faise unha proba

Neste caso, ilustrado na Figura 3.6a vemos como se volve repetir o patrón dos dous anteriores, volvendo de novo cambiar unicamente o método ao que chama o centro autorizado, esta vez *addTest()*, para engadir a información referente á proba feita por Alice. Neste caso hai que resaltar dúas cousas:

- **A autoridade pode levar incentivo:** Debido a que, a diferenza de ca administración das vacinas e o diagnóstico de reaccións adversas, unha proba pode ser feita por unha entidade privada non sanitaria ou mesmo polo propio cidadán. Cabería entón a posibilidade de opcionalmente incluír no método *addTest()* que aquela conta que rexistre o tests tamén sexa incentivada cunha porción de criptodivisa, para así, por exemplo, promover as campañas de testaxe ou os cribados por parte de institucións non sanitarias: empresas, colexios, ximnasios...
- **O resultado pode non ser inmediato:** Algunhas probas contra a COVID-19 poden tardar varias horas en ofrecer un resultado, polo que nestes casos teríamos dúas posibilidades: ou ben a opción de que a transacción non fora enviada polo CDC até que se reciba o resultado, facéndoo sen que o paciente estea presente, o que podería ser un pequeno defecto de seguridade, ou poderíase incorporar un método a parte para certos tipos de tests, só para engadir o resultado, chamado de feito remoto pola entidade de testaxe cando os resultados foran recibidos, e o resto de información estar xa presente no sistema na visita do paciente, de igual maneira que nos dous casos anteriores.

Se a proba é realizada por unha entidade sanitaria, ao igual que a vacinación ou as reaccións adversas, o incentivo á mesma é inútil, e o caso de uso será o mesmo só que sen o incremento da *wallet* da entidade que realizou o test, polo que ou ben se fan dous métodos diferentes para que a entidade de tests reciba incentivo ou non, dependendo de se é pública ou privada, ou ben se da o incentivo en todos os casos aínda que neste último caso sexa irrelevante. Outra opción sería engadir un campo extra ao dato CDC indicando se é público ou privado, e ao chamar ao método *addTest()* decidírase se outorgar o incentivo ou non.

Alice comercia ca criptomoeda

O diagrama incluído na Figura 3.6b fai referencia ao caso de uso relacionado ca criptomoeda, onde Alice comerciará con ela ao poder enviala a outra *wallet*, facendo así posibles as transaccións económicas mencionadas no primeiro capítulo. Destacar deste caso, que as

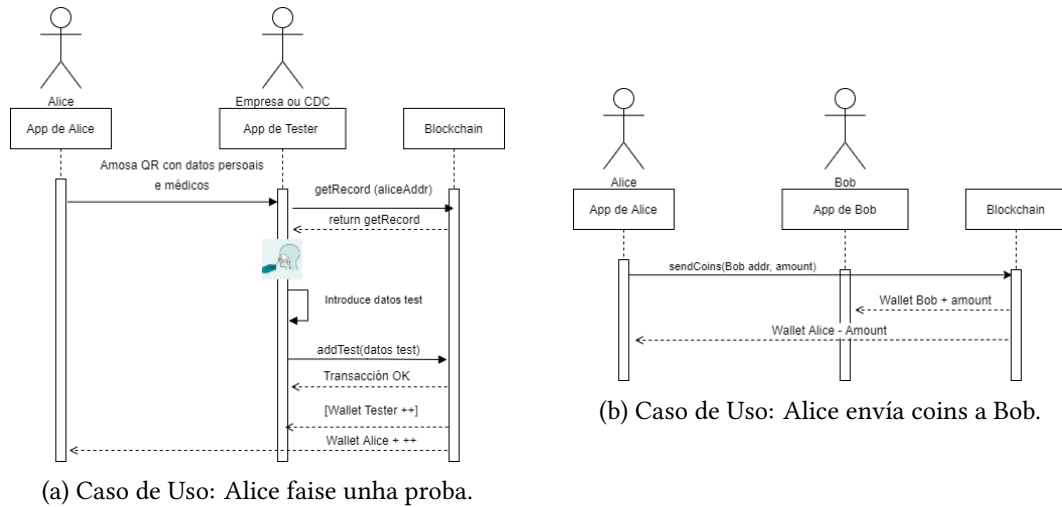


Figura 3.6: Casos de uso de Alice facendo unha proba e comerciando ca criptomoea.

chamadas serán a un *Smart Contract* diferente, encargado unicamente de xestionar a criptomoea, e só poderán ser feitas pola versión da app para os cidadáns/pacientes. A outra funcionalidade respecto a este contrato será a de consultar o saldo de criptodivisa dispoñible e debido á súa sinxeleza estará detallada no Apéndice C, na Figura C.1.

3.3 Deseño da app

Imos a continuación, e como última parte do deseño, falar da app móbil. Nos estándares de desenvolvemento de aplicacións móbiles, o primeiro a facer despois da análise de requisitos e funcionalidades, así como dos diagramas de secuencia dos casos de uso, son os chamados *mockups* ou *wireframes*, onde cunha ferramenta de deseño, faremos unha réplica en aspecto da nosa app, para mostrar a súa estética e fluxo entre compoñentes. No caso deste traballo, optouse pola aproximación de facer *wireframes*, sendo estes aquela versión máis xeral, cunha estética básica e pouco detallada, sen afondar na elección de cores, estilos ou detalles estéticos similares, para así centrarse na organización dos elementos por pantalla e dos fluxos entre as mesmas. Esta decisión foi tomada debido á sinxeleza da app, a cal non conta con demasiadas pantallas diferentes nin elementos a deseñar, polas restricións de tempo, o cal era limitado, e debido tamén a que a estética final non era un obxectivo primordial, como si que o era pola contra conseguir o 100% de funcionalidades operativas.

Imos ver entón unha selección dos *mockups* deseñados e unha explicación dos seus elementos, separándoos á súa vez en tres partes: a **global**, común para apps de cidadáns e autoridades, a do **ciudadán ou paciente**, sen permisos de escritura, e a das **autoridades**, con ditos permisos pero sen funcionalidades referentes á criptomoea. Puntualizar que nesta sec-

ción aínda que se explicarán todas, só se amosarán capturas dunha parte das pantallas, máis concretamente daquelas que se consideran **esenciais** e que **non foron ensinadas nos capítulos de probas** finais da app, aínda que no Apéndice C poderemos atopar todos os *mockups* restantes. Esta decisión foi tomada por motivos de optimización da extensión da memoria.

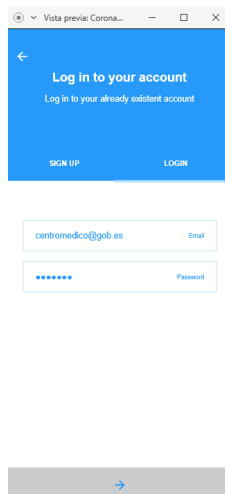
3.3.1 Parte compartida

Xestión de usuarios

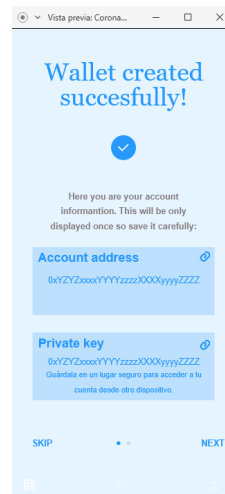
As primeiras pantallas despois de abrir a aplicación serán ou ben a de **Sign Up** (dispoñible no Apéndice C na Figura C.2a ou nas probas finais da app no Capítulo 6) ou ben a de **Log In** (dispoñible nesta sección na Figura 3.7a). A transición entre elas será clickando na parte superior da pantalla, no seu nome, e o seu funcionamento moi básico. **Sign Up** creará unha conta de Firebase asociada á app, para entrar nela e ter un email con contrasinal asociado. Á súa vez, interactuará ca Blockchain xa que creará unha nova *wallet* sen fondos. Esta pantalla levará ás **Welcome Sliders** que comentaremos no apartado 3.3.2. Pola outra banda, a pantalla de **Log In** será utilizada para loguearnos ca nosa conta de Firebase creada previamente e levará directamente á pantalla inicial da nosa versión da app que veremos posteriormente.

3.3.2 Parte paciente

Welcome Sliders



(a) Mockup: Log In.



(b) Mockup: Welcome Slider 1.

Figura 3.7: Mocks de Log In e do primeiro Welcome Slider.

O primeiro que verá o paciente despois de crear a súa conta, serán dúas **Welcome Sliders** ca información básica necesaria para comezar usar o pasaporte. Pódense ver por separado,

a primeira na Figura 3.7b e a segunda ou ben no Apéndice C ou nas probas finais da app no Capítulo 6. Estas dúas pantallas, as cales aparecerán unicamente por primeira vez ao crear unha conta (a menos que desexemos revisualizar mais tarde o tutorial de iniciación na app dende o perfil), encargaranse de, a primeira delas, Figura 3.7b, darlle a información relativa á nova *wallet* creada ao usuario (en concreto a dirección da súa conta na Blockchain e a súa **chave privada**). Dita chave privada móstrase por pantalla así como se da a posibilidade de copiar ao portapapeis para gardala onde o usuario desexe para poder *loguearse* posteriormente dende outros dispositivos, modelo moi parecido ao que nos oferta *Binance*, por exemplo. E a segunda, Figura C.2b, será a encargada de enviar un email de verificación ao correo introducido ao crear a conta, para que manualmente este sexa verificado e así engadir unha medida de seguridade contra a creación masiva de contas (a maiores das xa ofertadas por Firebase), e para mitigar tamén potenciais problemas de inanición de Ether a costa de peticións masivas ao noso *Faucet*. Unha vez feito isto, poderase pedir Ether ao noso *Faucet* personalizado, sempre que a nosa conta estea verificada e non conte xa con fondos. Cumpridos os requisitos, transferirásenos unha cantidade inicial de Ether para poder firmar transaccións na cadea de bloques e comezar usar todas as funcionalidades do sistema.

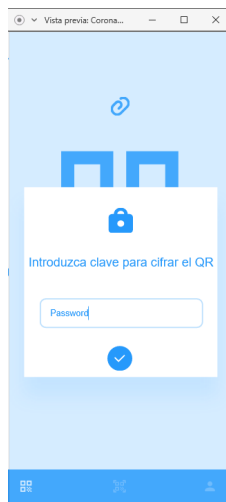
Patient Home Screen

Na pantalla inicial da app paciente, da cal teremos exemplos nas probas do Capítulo 6 e nas Figuras C.3a e C.3b do Apéndice C, vemos dous posibles estados. Antes de nada, imos comentar os elementos comúns en ambas, que son a **Bottom Navigation Bar** para navegar polas diferentes pantallas dispoñibles na app, seguindo as tendencias de deseño actuais que vemos en apps como *Instagram* ou *Spotify*. Esta barra actuará como elemento de transición entre a pantalla principal e as secundarias, podéndonos levar tanto ao apartado de lector de pasaportes, que comentaremos en detalle a continuación, como ao perfil da conta. Por outro lado, teremos outro elemento común como é o **Código QR**, que por defecto, mostrará un JSON cos campos da información persoal inicializados co valor X e co campo *Wallet* contendo o valor da dirección da conta (0x...). Dous exemplos dos JSON a atopar en ditos QRs, tanto o por defecto como o editado, pódense consultar no apartado do Apéndice C.2.2.

En canto aos elementos diferentes entre elas, teremos que ter un botón de **actualizar ou refrescar**, o cal se encargará de consultar á Blockchain se a *wallet* da conta actual foi engadida ao sistema, para dependendo da resposta cambiar a mensaxe aclarativa de debaixo do QR (principal diferenza entre ambas). Cando dito pasaporte estea na cadea, teremos que contar co botón de **compartir**, o cal veremos máis adiante en detalle, pero que será o encargado de compartir mediante unha app externa o QR presente na app, cifrado simetricamente ca clave introducida na nosa app, para cumprir así o caso de uso de envío do pasaporte seguro. A pantalla ao clicar para compartir sería a amosada na Figura 3.8a, onde vemos como aparecerá un

pop-up previo á lapela por defecto de compartir de cada sistema operativo, onde se introducirá a chave ca que cifrar o QR e a cal terá que ser coñecida por destino á hora de descifrar o QR. Pulsar no *tick* creará un ficheiro co QR cifrado, o cal se compartirá pola app externa elixida polo usuario (Gmail, por exemplo), para máis tarde ser importado de forma interna pola app en destino e descifrado de novo ca chave introducida da mesma forma que na Figura 3.8a.

Perfil do usuario



(a) Mockup: Compartir QR.



(b) Mockup: Perfil.

Figura 3.8: Mocks de cifrar e compartir QR e de engadir información persoal do usuario.

Imos agora con outra das pantallas que contén unha funcionalidade chave do sistema, a de **perfil de usuario**. Unha das primeiras tarefas a facer polo usuario será introducir os seus datos persoais para crear o QR que lerá a autoridade que o introduza no sistema. Para isto, bastará con *clickar* na icona de perfil na *Botom Navigation Bar* e atoparémonos coa pantalla amosada na Figura 3.8b. Nesta pantalla introducíranse manualmente os datos persoais para así actualizar o QR disposto na pantalla principal. Aínda que se pode actualizar cando sexa necesario, rexenerando así o QR, o instante crítico é o previo ao entrar no sistema, xa que unha vez o CDC introduza os datos na Blockchain, por moito que os cambiemos, só cambiarán no noso QR, e o único dato utilizado á hora de lelo unha vez que conste no sistema, será a dirección de *wallet*, a cal non se pode introducir manualmente e que serve para obter o resto de datos, tanto persoais como médicos, que se obterán polo tanto directamente da Blockchain e non do noso QR, evitando así erros e cambios de información.

Parte criptomoeda

Nunha mesma pantalla ocuparémonos desta parte, onde teremos as funcionalidades de:

- **Saldo actual de criptodivisa:** Poderemos consultar a cantidade posuída da criptomoe-da pola carteira da conta da aplicación logueada.
- **Envío de criptos:** *Clickando* na icona correspondente, saltará un *pop-up* (como podería ser o da Figura 3.8a), o cal servirá para transferir a cantidade indicada de criptomoneda á dirección de *wallet* introducida.

Ambos *mockups* pódense consultar no Apéndice C, no apartado C.2.2, xa que as pantallas serán amosadas na app real no capítulo 6.

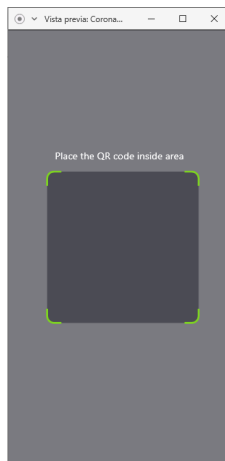
Ler pasaporte e mostrar información

Por último, na parte do cidadán ou paciente teremos a funcionalidade de **lectura de pasaporte**. Pode ser feita de dúas maneiras, importando un QR cifrado, ou lendo directamente o QR amosado dende a app por outra persoa. Para facer uso delas, teremos que ou clicar na *Bottom Navigation Bar* na icona do lector de códigos QR que vemos na Figura 3.8b, ou ben importar o QR cifrado recibido como ficheiro, introducindo previamente a chave pre-acordada ca que se fixo o cifrado simétrico da forma amosada na Figura 3.8a.

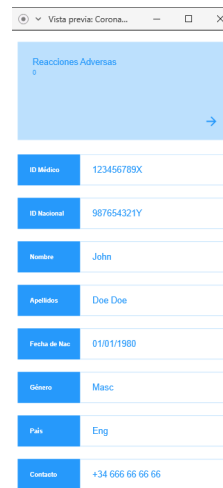
Para a opción de escanear un QR (Figura 3.9a), simplemente abríranos a cámara cun lector QR, no que teremos que colocar o QR amosado na app da persoa a cal queremos ler o seu pasaporte. Por outra banda, se *clickáramos* en importar, tras abrírenos a lapela por defecto do **SO** para seleccionar un arquivo existente no noso móbil, pantalla non modelada xa que é externa á app, o amosado pola aplicación tras escollelo sería da forma que vemos na Figura 3.8a, onde teríamos que introducir o contrasinal acordado con orixe, co cal se cifrou o pasaporte QR, para poder así descifralo clickando no *tick* e acceder aos datos de igual maneira que se o leramos co lector. Ambos casos, ao finalizar, levan á mesma pantalla, aquela referente á información obtida de dito QR, da forma que vemos na Figura 3.9b.

Á hora de ver os resultados, temos dúas posibilidades: a visualización do QR dunha *wallet* non incorporada ao sistema, é dicir, que ten os datos introducidos manualmente polo usuario no perfil ou os por defecto, ou ben ver o pasaporte de alguén que si está dado de alta e ten información médica asociada. A diferenza de ambas pantallas podémola consultar en detalle no Apéndice C: o caso de lectura dun pasaporte que non consta na Blockchain ilústrase no *mockup* da Figura C.5a e o caso contrario, podemos velo en detalle tamén na Figura C.5b. Aínda que tamén poderemos ver isto na sección de probas da nosa **Dapp**, podemos consúltalas como terceira opción nas Figuras 3.10a e 3.10b da parte CDC desta sección. No caso da lectura dun pasaporte engadido (Figuras 3.9b e 3.10b), teríamos unha mensaxe aclaratoria corroborando que dita conta está no sistema, seguida primeiro, da información médica dispoñible, e por último, da información persoal. Neste caso, todos os datos amosados atópanse na Blockchain, o que quere dicir que foron verificados e escritos por unha autoridade. Por último, destacar que

se quixeramos ver en detalle tanto as doses, como os tests ou as reaccións, só teríamos que pulsar sobre calquera delas para levarnos a unha pantalla idéntica para os tres casos, onde só cambiaría a información médica de cada obxecto, dependendo de se tratan de vacinas, tests ou de reaccións adversas. Isto, verémolo na sección final de probas da app no Capítulo 6 pero tamén se poden consultar os *mockups* no Apéndice C, no apartado C.2.2.



(a) Mockup: Ler Pasaporte. Escanear QR.



(b) Mockup: Ler pasaporte oficial.

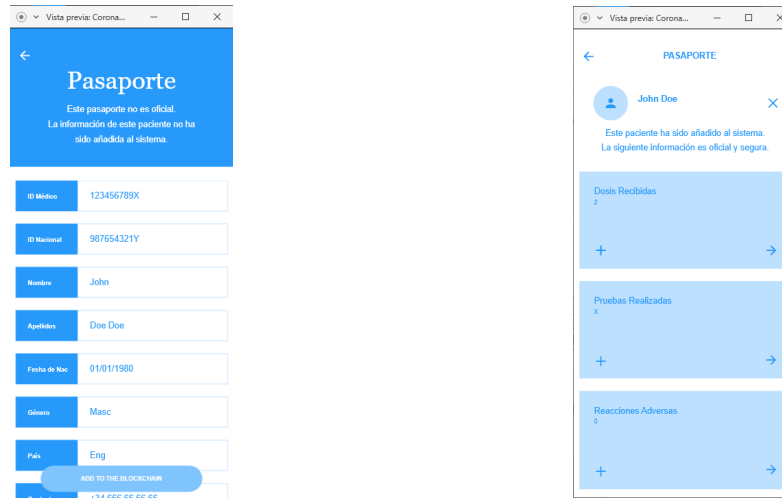
Figura 3.9: Mocks de escáner QR e lectura de pasaporte incorporado ao sistema.

3.3.3 Parte CDC

No que respecta á app para os CDCs ou autoridades sanitarias, é moi similar en canto estética á app de pacientes previamente comentada, pero diferénciase principalmente nas funcionalidades extra ao poder escribir diversos datos na Blockchain. De igual maneira, poderemos consultar todos os mockups referentes a ela no Apéndice C, pero imos mencionar a continuación os aspectos esenciais deseñados de dita versión.

Escribir no rexistro médico escaneado

Seleccionando na *Bottom Navigation Bar* a opción de lector QR, aparecerá unha pantalla idéntica ao escáner do paciente, amosado na Figura 3.9a. Descártase a opción de importar QR, xa que para as tres tarefas (inoculación de dose, realización de test ou diagnóstico de reacción adversa), é imprescindible a presencialidade do usuario. Unha vez escaneamos o QR do paciente, haberá, ao igual que no caso anterior, dúas posibilidades: a de tratarse dun pasaporte xa no sistema, ou dun aínda non engadido. As pantallas, aínda que case idénticas ás da versión paciente conteñen lixeiros cambios. Estes poden verse nas Figuras 3.10a e 3.10b.



(a) Ler pasaporte non presente no sistema.

(b) Ler pasaporte presente no sistema.

Figura 3.10: Mockups dos tipos de lectura de pasaporte dende CDC.

Ditos cambios son: no caso de lectura dun pasaporte non engadido ao sistema (ilustrado na Figura 3.10a), a aparición dun botón para facelo, e no caso onde o pasaporte foi xa engadido (na Figura 3.10b), a dun botón por cada un dos campos médicos que servirá para engadir un elemento máis do tipo elixido ao rexistro do cidadán. No primeiro deles, cando *clickamos* para engadir dita información á Blockchain, aparecerá unha lapela de confirmación de dita información estilo *pop-up*, de igual maneira que en todos os casos de escrituras na Blockchain, para que estas sexan corroboradas polos CDCs. No segundo caso, ao pulsar o botón de engadir dose/proba/reacción, a aplicación mostrará unha nova pantalla para introducir manualmente a información do obxecto médico a engadir (e.g., unha dose) para de igual maneira que no caso anterior, posteriormente facer unha confirmación, sendo estándar así no sistema que para a introdución de calquera dato na Blockchain, fagan falta polo menos dúas confirmacións, evitando escrituras erróneas. Ditos *mockups* vémoslos no apartado do Apéndice C.2.3.

3.4 Proposta de deseño

Vistas as anteriores sub-etapas de deseño, contamos ca información suficiente para elaborar unha **proposta xeral de deseño** onde se amose dende unha perspectiva global o que vai ser o proxecto, vendo de forma gráfica as súas partes e funcionalidades principais. En concreto, na Figura 3.11 podemos ver a proposta de deseño final do traballo, onde vemos a tecnoloxía a usar (**Blockchain Ethereum**) así como os dous tipos de apps e usuarios que usarán o sistema acompañados das súas funcionalidades básicas, por un lado os cidadáns/pacientes, que poderán ler pasaportes así como comerciar ca criptomoneda, como por outro lado as autorida-

des sanitarias, con contas autorizadas para escribir na Blockchain a información médica dos cidadáns. Tamén apreciamos na imaxe as **dúas grandes partes** do proxecto, relacionadas cada unha cunha das dúas mencións realizadas. Esta división, por un lado ca parte de **Blockchain** e por outro lado ca **Dapp móbil Multiplataforma**, pode apreciarse ao longo de todo o proxecto e influíu notablemente tanto na súa metodoloxía como na súa planificación.

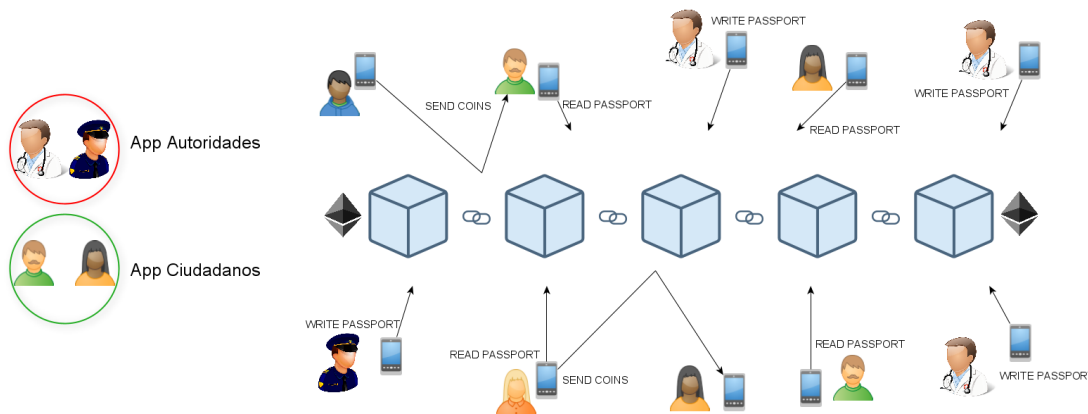


Figura 3.11: Proposta inicial de deseño.

3.5 Metodoloxía

Aínda que a metodoloxía utilizada non foi a típica dun proxecto software debido á significativa parte de investigación deste TFG, si contén fases comúns e pode considerarse na parte de desenvolvemento como unha metodoloxía **incremental**, dividida nas clásicas etapas do proceso de desenvolvemento software de **Análise-Deseño-Implementación-Probos**. Sumado a isto, tívose en conta o que mencionabamos no final da sección anterior: a división de todo o proxecto en dous grandes módulos, o referente a Blockchain e aquel ocupado do desenvolvemento da app móbil na ferramenta multiplataforma elixida, polo que este feito tivo un impacto na metodoloxía, ao separar as dúas últimas fases do ciclo de vida do proxecto (é dicir a implementación e probas) para cada unha destas partes e non facelas de forma conxunta como si que ocorreu nas fases de análise e deseño. Ben é certo, que ademais de separadas, ambas fases foron **iterativas**, é dicir, non se comezou ca segunda (desenvolvemento **Dapp**) até ter unha primeira versión *alfa* da primeira, referente á Blockchain. Como apunte final, destacar que unha vez chegados a ambas versións alfa separadas, programouse un **último ciclo completo** extra de Análise-Deseño-Implementación-Probos, para levar ao sistema en conxunto á unha segunda versión final, a *beta*, a cal será entregada como TFG.

Fundamentos Tecnolóxicos

O seguinte capítulo pretende adentrarse en todos os conceptos relacionados ca tecnoloxía utilizada no proxecto. En primeiro lugar, explícanse na sección **Base Teórica** os principais conceptos a dominar para unha mellor comprensión do traballo, os cales, ou gozan dunha especial complexidade, ou son o suficientemente recentes como para non ser coñecidos por unha ampla maioría. En segundo lugar, na **Selección Tecnolóxica**, para cada subsistema do proxecto, enumeraranse as ferramentas escollidas, tanto de soporte como de desenvolvemento, as cales estarán acompañadas dunha breve descrición das mesmas e unha xustificación do seu uso fronte outras alternativas.

4.1 Base Teórica

Unha vez chegado ao punto onde temos a fase de análise e a de deseño totalmente completas, antes de comezar ca implementación, hai que realizar unha serie de tarefas que serán determinantes á hora do resultado final a obter co proxecto. A primeira consistirá en coñecer e dominar as tecnoloxías a usar para poder facer unha selección das mellores e que mellor encaixen no sistema. Claramente, neste traballo pódense discernir dous conceptos chave como son o de **Blockchain** e o de **Desenvolvemento Multiplataforma**, os cales, debido á súa complexidade e á súa gran novidade, fixeron que antes de comezar coa implementación houbera que adentrarse neles para coñecelos e dominalos. Por isto, na seguinte sección téntase explicar en detalle estes conceptos teóricos co fin de que, debido a que de forma xeral son conceptos pouco coñecidos pola inmensa maioría, comprendelos a fondo pode ser un elemento chave á hora de entender o traballo restante.

4.1.1 Blockchain

Blockchain, que en galego se traduce literalmente por **cadea de bloques**, é unha **DLT** (**Distributed Ledger Techonology**), ou o que é o mesmo unha tecnoloxía distribuída (*Distri-*

buted Technology), é dicir descentralizada, sen ningunha dependencia nin punto crítico ou autoridade central, na cal se almacena información (a palabra *Ledger* en inglés poderíase traducir como un libro onde diferentes actividades, normalmente económicas, son almacenadas). Esta información é almacenada no que se denominan **bloques**, os cales están relacionados entre si co inmediatamente anterior e posterior, simulando así a estética dunha cadea, neste caso de bloques, onde se garda información, a **Blockchain**.

Estes bloques están directamente relacionados cas **transaccións**, as cales, para asegurar a integridade da información que conteñen, son verificadas por **todos os nodos** da Blockchain [24]. Temos así dous novos conceptos, primeiro, o de **nodos** da Blockchain, os cales serán os diferentes *end-points* que terán almacenada a cadea de bloques e estarán conectados a ela, xa que como recordamos, é un mecanismo distribuído, neste caso distribuído por todos os nodos que a forman e os cales serán os encargados de verificar que a información dispoñible no sistema é correcta, fiable e segura. Por outro lado temos o concepto de **transaccións**, as cales serán interaccións ca cadea de bloques, creadas polos usuarios que fagan uso desta tecnoloxía e as cales conteñen información a engadir á cadea.

Estas transaccións son engadidas aos bloques por nodos que no caso de Blockchains coma Bitcoin se soen chamar **mineiros**, aqueles que escriben na Blockchain. Para minar estas transaccións, é dicir, ser aquel nodo que consegue escribir o bloque na cadea, os nodos deben completar un traballo ou proba, a denominada **proof** en inglés. Estas probas son de diferente tipo dependendo da Blockchain a usar e serven para determinar un algoritmo de consenso de que nodo é o elixido para engadir dita información. Hainos de varios tipos, aínda que os máis comúns son o **Proof-of-Work**, o cal consiste na resolución dun problema matemático dunha elevadísima complexidade computacional, e o **Proof-of-Stake**, que, en vez de basearse na resolución dun problema coma o anterior, o que se traduce en poder ter a suficiente potencia computacional para conseguilo, baséase noutros conceptos como pode ser a cantidade de criptomoeda que ten cada nodo, concepto da cal falaremos no seguinte parágrafo. Este modelo de consenso mediante proba tradúcese en que para escribir información corrupta no sistema, non só hai que ser o nodo que consiga superar en primeiro lugar dito *proof*, senón que despois a transacción que escriba dita información ten que ser verificada polo menos polo **51%** dos nodos da cadea de bloques, polo que, cun número suficiente de **nodos mineiros** fiables, as Blockchains son consideradas estruturas de datos **seguras e inmutables** [24].

Aínda así, segue a haber unha serie de conceptos que hai que coñecer, como por exemplo a **criptomoeda** da Blockchain. Cada cadea de bloques ten unha criptodivisa asociada, a cal usa a modo de incentivo para darlle a aquel nodo mineiro que consegue engadir un bloque á cadea e así compensar a inversión feita por dito nodo. Exemplo disto pode ser o **Ether** en Ethereum ou o **Bitcoin** en Bitcoin.

As transaccións que escriben datos na cadea de bloques teñen un custo, o cal por exemplo

en Ethereum se adoita chamar **gas** ou **gas price**. Este concepto basicamente consiste nunha porción da criptodivisa da cadea, no caso de Ethereum de Ether, que o emisor da transacción **paga** para que esta sexa minada por algún nodo mineiro e polo tanto engadida ao sistema. É dicir, poderíamos emitir transaccións *gratis*, é dicir con prezo (*gas price*) 0, pero non serían minadas por ningún nodo, polo que a nosa información nunca constaría no sistema. Este concepto é chave tanto como para protexer o sistema contra usuarios malintencionados [24] como para soste o modelo de incentivos aos **nodos mineiros** explicado antes.

Por último, outros dous conceptos importantes á hora de entender as cadeas de bloques son os de *wallets* e/ou contas dos usuarios. Como vimos, os usuarios interactuaban ca Blockchain enviando transaccións. Para facelo, necesitarán ter unha **account** na Blockchain, con fondos da criptodivisa nela, para poder firmar as transaccións. Unha **wallet** simplemente será un mecanismo onde un usuario poida importar a súa ou súas contas para manexalas. Ambas están fortemente relacionadas con conceptos criptográficos como poden ser chaves públicas para firmar, chaves privadas para facer uso das contas, *passphrases*, etcétera. Podemos ver un resumo dos conceptos expostos na Figura 4.1, obtida de [40].

How does a transaction get into the blockchain?

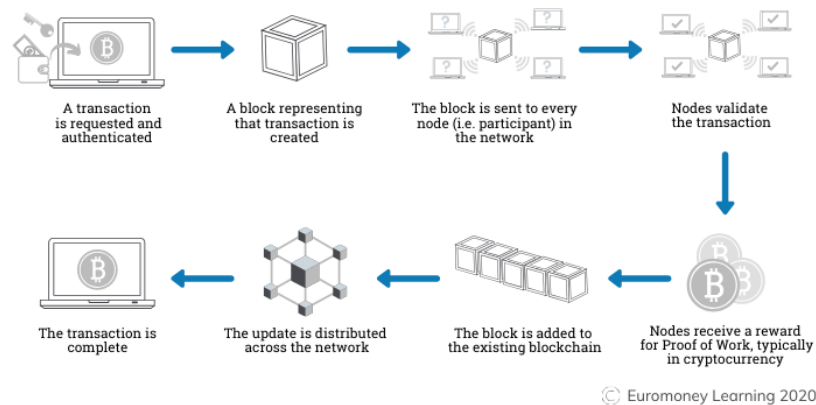


Figura 4.1: Overview do funcionamento da tecnoloxía Blockchain [40].

Vemos como **Blockchain** é a tecnoloxía exacta que necesitamos para este traballo, xa que permite almacenar a información dun xeito totalmente seguro, así como dotala de integridade e inmutabilidade, características fundamentais para a información médica referente á COVID-19. Agora, quedaría elixir a cadea de bloques a utilizar e, tras analizar varias posibilidades, elixiuse **Ethereum** debido a que trae consigo unha serie de capacidades que a fan a aproximación ideal para este TFG. Vemos as fundamentais a continuación.

Blockchains Privadas

As **Blockchains Privadas** son aquelas que só son accesibles por unha organización, feito que aínda que sacrifique parte da propiedade de descentralización inherente a Blockchain (ao quitarlle o carácter público), mellora a privacidade da mesma [24]. Ethereum, a parte da súa cadea de bloques pública e principal, a chamada *mainnet*, ofrece tanto *testnets* públicas, como *Rinkeby* ou *Ropsten*, as cales son extremadamente útiles para probar conceptos de desenvolvemento e despregue de *Smart Contracts*, nun sistema que xa conta con *nodos mineiros* establecidos, aínda que obviamente nel a criptodivisa carece de ningún tipo de valor real, como tamén ofrece a posibilidade de crear dende cero a túa propia **Blockchain privada**, obviamente sempre baseada en Ethereum, pero feita á túa medida, e cunha gran variedade de parámetros modificables ao gusto de cada caso.

Smart Contracts e Tokens ERC-20

Os **Smart Contracts** ou contratos intelixentes son bloques de código que se almacenan na Blockchain. Consisten en funcións ou eventos que permiten aos usuarios interactuar entre eles usando a cadea de bloques como medio [24]. O seu código, debido a que é almacenado na Blockchain, a cal goza das características explicadas no apartado anterior, é inmodificable e está dispoñible para todos aqueles nodos conectados á cadea de bloques. Ethereum goza dunha alta programabilidade ao poder de forma moi sinxela crear, probar e despregar contratos intelixentes en calquera tipo de cadea de bloques baseada nela, tanto pública como privada.

Por último, hai un concepto extra do cal falar, os **Tokens ERC-20**. Estes *tokens* son uns *Smart Contracts* estándar os cales describen unha serie de funcións e eventos a implementar á hora de usar un deles no teu proxecto, co fin de crear *coins* ou criptomoedas personalizadas, que non teñan que ver nin estean asociadas ca criptomoeda principal da cadea de bloques [30]. Pódense ver como un obxecto programático, o cal ten un posuidor e co cal se pode comerciar, sendo enviado e recibido, estando todos os datos asociados a dito *token* na Blockchain, é dicir, son o mecanismo ideal para a creación dunha moeda virtual que posúa os beneficios inherentes á tecnoloxía Blockchain: seguridade, descentralización e eliminación de intermediarios, así como tamén inmutabilidade e integridade. *Ethereum* facilita a súa creación tendo o desenvolvemento que preocuparse escasamente de maiores detalles ca o nome e a cantidade inicial da criptodivisa para poder comezar empregala, pero dando tamén posibilidades de programala detalladamente no caso de que fora necesario.

É por todo isto polo que **Ethereum** foi a cadea de bloques elixida, xa que ofrece absolutamente todas as facilidades posibles para a implementación do sistema deseñado, así como encaixa perfectamente nas súas liñas e obxectivos principais.

4.1.2 Desenvolvemento móbil multiplataforma

O desenvolvemento en plataformas cruzadas (***cross-platforms development***) é un paradigma de desenvolvemento de apps móbiles o cal está cobrando cada vez máis importancia por factores como son o crecemento exponencial das aplicacións móbiles dispoñibles, así como o feito de que ditas tecnoloxías están por postularse como as principais e máis usadas na actualidade, desbancando ás *desktop* e outras do denominado ***mainstream*** tecnolóxico [41].

Dito paradigma é oposto á natureza heteroxénea do **desenvolvemento nativo**, maioritaria solución a día de hoxe e a cal propón desenvolver cada app exclusivamente para o sistema operativo onde vai ser despregada, e se vai ser destinada a varios, facer tantos desenvolvementos como sexan necesarios, iso si, sempre aproveitando ao máximo as capacidades que nos brinda implementar a app co código nativo da plataforma onde vai ser empregada.

As novas plataformas de desenvolvemento cruzado están suscitando cada vez máis e máis interese debido aos seus grandes beneficios, inherentes ao concepto de compartición de código, o cal por outra banda sempre foi un criterio nada pragmático e cheo de inconvenientes, dos cales algúns seguen vixentes, como por exemplo o feito de non poder facer a totalidade do código da app deste modo ou o feito de que cando a complexidade do sistema a deseñar é moi elevada, pode non ser suficiente e dar lugar á necesidade do uso da linguaxe nativa [42]. Aínda así, os beneficios que ofrece, e o feito de que segue a ser unha aproximación válida para unha gran maioría de proxectos fan que sexa un concepto en auge hoxe en día.

Tecnicamente, o desenvolvemento cruzado consiste en codificar a app nunha linguaxe neutra, que a través de mecanismos propios saiba xerar dous resultados finais dende o mesmo código, grazas á compilación do código fonte no entorno nativo de cada plataforma [43].

KMM

KMM [29], ou **Kotlin Multiplatform Mobile**, ferramenta en ascenso no espazo multiplataforma, é unha versión *alfa* dende finais de 2020 a cal permite executar código escrito en **Kotlin** en iOS, Android e máis. Trátase dun **SDK (Software Development Kit)** para o desenvolvemento multiplataforma suministrado por **JetBrains**, o cal utiliza as capacidades e potencial de Kotlin xunto con outras funcionalidades engadidas para mellorar a experiencia de crear apps multiplataforma [28]. Con **KMM** pódese aproveitar a flexibilidade dun desenvolvemento entre plataformas sen perder as vantaxes do potencial do desenvolvemento nativo, xa que se utiliza, na gran maioría, unha base de código única para a lóxica da app iOS e Android, e escríbese código específico de cada plataforma só cando é estritamente necesario, por exemplo para implementar unha interface nativa ou para traballar con **APIs (Application Programming Interfaces)** específicas, como podemos apreciar na Figura 4.2. Despois de dita figura veremos algunhas das características máis importantes de **KMM** [28]:



Figura 4.2: Arquitectura KMM [28].

- **Traballo dende un só IDE (Integrated Development Environment)**, xa que existe un complemento para Android Studio.
- **Apoio de Google** xa que dende finais de 2020, mostrou o soporte e apoio a esta disruptiva tecnoloxía, establecéndoa como o futuro do desenvolvemento móbil.
- **Interoperabilidade**, xa que ofrece unha sólida integración co proceso de desenvolvemento iOS, grazas á interoperabilidade bidireccional de *Kotlin/Native* con *Objective-C/Swift*.
- **Facilidade á hora de usar de APIs específicas** de calquera plataforma.

Por todo o comentado até agora, o desenvolvemento multiplataforma foi o elixido para o proxecto debido á necesidade fundamental da dispoñibilidade para ambos sistemas operativos da app, porque nun tema como o que trata este TFG: referente ao mundo médico e que afecta a toda a poboación, como é a pandemia actual, non se podería eliminar unha porcentaxe tan alta de usuarios como suporía non dar soporte a un destes dous sistemas. Por outro lado, dentro do desenvolvemento cruzado, aínda que se sopesaron varias alternativas, como **React Native** ou **Flutter** [43], optouse por **KMM**, maioritariamente polas opcións e vantaxes destacadas anteriormente, xa que aínda que Kotlin era un *framework* e linguaxe moi nova e non coñecida, o potencial que ofrece o seu coñecemento así como as proxeccións de futuro que lle da ao traballo, fixeron que fora a opción escollida.

4.2 Selección tecnolóxica

Nesta sección discutiranse as principais tecnoloxías e ferramentas utilizadas, aportando unha breve descrición das mesmas así como unha xustificación do seu uso. Enumeraranse aquelas que xogaron un papel máis crítico á hora de desenvolver o proxecto, e detallaremos as restantes, que aínda que non vitais tamén foron importantes, no Apéndice D.

4.2.1 Globais

AES

[Advanced Encryption Standard \(AES\)](#), tamén coñecido como *Rijndael*, é un dos algoritmos máis populares utilizados en criptografía **simétrica** [44]. Dito algoritmo é un esquema de cifrado por bloques adoptado como estándar de cifrado por múltiples institucións a nivel mundial, por exemplo, a [NSA](#). É utilizado no proxecto tanto para cifrar as *Shared Preferences*, que discutiremos máis adiante, como tamén para enviar dende a app o pasaporte cifrado cunha chave previamente acordada entre ambos extremos.

4.2.2 Blockchain

Solidity

Linguaxe de alto nivel orientado a *Smart Contracts* [45]. Sintaxis similar a *JavaScript* e enfocado ás [EVM \(Ethereum Virtual Machine\)](#). Tipado estático e algunhas das súas *features* son heranzas, librerías e definición de tipos complexos. Aínda que se poden programar contratos intelixentes sobre Ethereum en diferentes linguaxes, esta asemella a mellor opción para o traballo debido á súa curva de aprendizaxe e a súa compatibilidade con Ethereum.

Geth

Geth [46], ou **Go Ethereum**, é unha das implementacións orixinais do protocolo Ethereum, concretamente a escrita en *Go*. É *open-source* e pódese instalar en case calquera sistema operativo co fin de comunicarse cas redes Ethereum: a *public mainnet*, calquera das *testnet* ou para facilitar a creación e despregue de *testnets* privadas, como foi o caso deste TFG.

Remix e Metamask

Por un lado temos que **Remix** é un **IDE** que permite desenvolver, despregar e administrar *Smart Contracts* para calquera Blockchain Ethereum [47]. Ferramenta moi potente grazas á cal se pode implementar e *testear* online sobre máquinas [EVM](#) os *Smart Contracts* desenvoltos en *Solidity* para despois usala para despregalos nunha *Testnet* Privada e realizar transaccións de

calquera tipo. Por outra banda, **Metamask** é unha extensión de navegador web que permite de forma intuitiva e sinxela comunicarse con aplicacións baseadas en Blockchain actuando de ponte entre o navegador e Ethereum [48]. Permite importar contas desas cadeas de diferentes formas, realizar transaccións interactuando con **Dapps** así como mesmo intercambiar Ether ou *tokens*. O uso combinado de ambas ferramentas é necesario para poder operar en Blockchains privadas, como foi o noso caso.

4.2.3 Dapp

Firestore

Plataforma para o desenvolvemento de apps web e móbiles ubicada na nube e que consta dunha serie de ferramentas para a creación e sincronización de proxectos que dotan a estes dunha maior calidade [49]. Algunhas das súas múltiples vantaxes son a súa extensa e detallada documentación, que é multiplataforma e tamén que usa a infraestrutura de Google, garantindo pois unha escalabilidade total. Firestore crea proxectos sen necesidade de servidor e sincroniza de forma moi doada os datos das apps sen ter que administrar conexións nin escribir lóxica complexa. Todo isto fixo que fora a ferramenta usada para o tratamento dos usuarios na nosa **Dapp** móbil, é dicir empregándoa como ferramenta para a persistencia online de datos.

Encrypted Shared Preferences

Implementación que envolve a clase *Shared Preferences* [50] dotándoa dun maior nivel de seguridade ao cifrar automaticamente as chaves e valores utilizando no noso caso, valores con **AES 256 GCM** de maneira non determinista [51]. É o segundo e último mecanismo de persistencia utilizado no TFG, sendo as *Encrypted Shared Preferences* un método lixeiro de almacenar localmente pares de chave-valor, aproximación máis que suficiente para gardar a información necesaria para o sistema a parte da existente na Blockchain, é dicir a dirección da *wallet* e a chave privada, principalmente. Por isto, e pola seguridade necesaria para gardar este tipo de datos, foi a aproximación elixida no traballo.

Web3J

Biblioteca Java e Android altamente modular, reactiva e segura para traballar con *Smart Contracts* e integrarse con clientes, é dicir, nodos da rede Ethereum [52]. Ideal para este proxecto para facer de ponte e conectar a **Dapp** móbil cos **nodos mineiros** na Blockchain privada, así como para xerar *wrappers* Java dos *Smart Contracts* para interactuar con eles e modificalos dende Android Studio.

Blockchain: cadea privada e contratos intelixentes

Proponse no seguinte capítulo a discusión de todos os elementos referentes á implementación e probas da parte de Blockchain do traballo. Ademais de ser unha das dúas grandes temáticas do proxecto, podemos dividila en dous módulos máis: un referente á propia **creación e despregue dunha cadea de bloques personalizada e privada**, e outro relacionado ca **escritura en *Solidity* dos Smart Contracts** correspondentes para o seu posterior despregue na Blockchain creada. Imos ver as diferentes fases e etapas polas que se pasou tanto na implementación como nas probas de ambos módulos, destacando os conceptos máis importantes de cada un, así como naqueles nos que se inverteu máis tempo, mostrando unha selección de exemplos do código e comandos necesarios para a correcta finalización de ambas fases, acompañados sempre dunha explicación en detalle así como dunha xustificación nos casos que fora necesario. O resto de casos, debido ao seu carácter menos crítico, así como certos anacos de código ou imaxes das que se falan nas seguintes seccións, poderanse atopar detallados no Apéndice E. De todas maneiras, todo o código tanto do proxecto, como da parte Blockchain en particular, poderá ser atopado baixo unha licenza *Open Source* no *link* dispoñible no Apéndice G.

5.1 Implementación

Como ben dixemos, o primeiro de todo será implementar a nosa propia cadea de bloques, é dicir, crear a nosa Blockchain privada baseada en Ethereum. Para elo, consultáronse traballos similares en [53] [54] [55] e [26], onde todos eles, aínda compartindo un esqueleto común, amosaban diferentes aproximacións para levar a cabo dita tarefa.

O primeiro de todo será instalar o software necesario. Neste caso, bastará con instalar **Geth**, xa que será cos comandos deste *framework* xunto ca edición duns poucos ficheiros de texto as ferramentas cas que despreguemos dita Blockchain. Unha vez temos dito software descargado e sincronizado con Ethereum, da forma que se expón no Apéndice E, estaremos listos para comezar as principais tarefas de configuración e creación da Blockchain, comezando polo seu bloque xénese.

redes de proba ou privadas adoita ser un valor baixo para evitar esperas, xa que ata que non se descubre o bloque válido non se executa a transacción. Na nosa rede, tamén se elixiu un valor baixo debido a que é prioritaria a velocidade de escritura dos datos médicos, así como que o Ether (recompensa do minado destes bloques) carece de valor.

- **gasLimit:** Valor escalar igual ao límite de gasto de gas por cada bloque en toda a cadea de bloques. No noso caso é un valor maior debido a que serán habituais altos prezos de gas nas transaccións debido a que escribirán moita información e poderán ser significativamente complexas, como por exemplo aquela que escriba todo un rexistro médico dun paciente na Blockchain.
- **alloc:** Permite definir unha lista de *wallets* precargadas co Ether especificado. É específico de Ethereum e moi útil no noso caso para dotar de Ether ás principais contas que minarán na cadea. Inicialmente, especificamos dúas contas, as cales traballaremos nesta memoria, unha con Ether e outra sen el.
- **chainId:** Identificador único da cadea. Por exemplo, a *public mainnet* de Ethereum ten o 0x1, e diversas *testnets* recoñecidas teñen outros valores. No noso caso, é o día da declaración do estado de alarma pola [COVID-19](#) en España, o 15 de Marzo de 2020. A súa utilidade é para securizar a Blockchain ante **ataques de tipo replay**, o cal un exemplo podería ser executar unha transacción nunha cadea de bloques privada e intentar facerlle un *broadcast* a outra cadea, por exemplo, a pública, para así corrompela. Ca existencia deste identificador, cada transacción leva intrinsecamente asociada a cadea á que pertence, para evitar así este tipo de ataques.
- **Blocks:** Referentes a conceptos de *chain forking* e versións. Deixámoslos en 0 porque estamos empezando unha nova Blockchain, pero inicializámoslos cun valor para que este deixe de ser nulo e así asegurar compatibilidade con ditas versións de Ethereum. Algúns deles foron engadidos a medida que se atoparon problemas na implementación dos Smart Contracts, como por exemplo o *Byzantium* e *Constantinople*, inicializados para soportar operacións de baixo nivel non incluídas noutras versións anteriores, en concreto a operación de ensamblador **SHR**.

Unha vez que temos Geth descargado, estamos sincronizados con Ethereum e temos o noso ficheiro da configuración do *Genesis Block* listo, estamos preparados para o despregue da cadea. Para isto, teremos que executar unha serie de **comandos Geth**. O primeiro, será inicializar o directorio que conterá os datos da nosa Blockchain e a configuración do bloque xénese. Para isto, teremos que ter unha carpeta que conteña o directorio para a información da cadea e o JSON anterior ca configuración do *Genesis Block*. Unha vez creado, haberá que executar os comandos amosados a continuación. Ditos comandos poden irse complicando

debido ás necesidades da Blockchain [57], como ocorreu neste proxecto. Dicar, ao igual que se comentará na parte de liñas futuras, que non todos os futuros nodos despregados terán que ter esta configuración, se non só unha parte dela dependendo da súa función, pero ningún deles necesitará parámetros diferentes aos explicados. Para ver unha explicación en detalle de **todos** os parámetros podemos acudir ao Apéndice E, aínda que a continuación amosamos o comando na súa totalidade e explicamos os parámetros máis críticos do mesmo:

```
1 $ geth --datadir ./chaindata/ init ./genesis.json
2 $ geth --datadir ./chaindata --networkid 15032020 --miner.gasprice 0 --txpool.pricelimit 0 --nat any
   --identity private_chain --nodiscover --port=30342 --http --http.port 9545 --http.api
   'admin,shh,txpool,debug,db,eth,net,web3,personal,miner' --http.corsdomain '' --rpc
   --rpc.allow-unprotected-txs=true
   --rpcapi='admin,shh,txpool,debug,db,eth,net,web3,personal,miner' --rpcport 9545 --rpcaddr
   '0.0.0.0' --rpccorsdomain "" --allow-insecure-unlock --unlock 1 console 2>> myEth.log
```

- **--datadir:** *Path* ao directorio onde a información da Blockchain está almacenada.
- **--networkid:** Especifica o valor do identificador da nosa cadea de bloques configurado no bloque xénese á hora da súa creación. Os nodos, para comunicarse entre eles, necesitan ter o mesmo identificador e versión da rede.
- **--miner.gasprice:** Establece o prezo mínimo de gas para que o nodo despregado mine unha transacción. No noso caso, importante que o valor sexa 0 nalgúns nodos, para poder así minar as transaccións gratuítas ao noso Faucet personalizado e poder entregar Ether. Tamén foi unha decisión que non todos os nodos o teñan configurado, tanto para así evitar posibles ataques de denegación de servizo contra todos os nodos da cadea e polo tanto da mesma, inunándoa de transaccións a prezo 0. Ao eliminar este parámetro no despregue de certos nodos, bloquearíanse só aqueles que acepten ditas transaccións sen coste, pero quedarían operativos o resto, libres de perigo ao rexeitalas directamente por teren un valor configurado de *miner.gasPrice* maior que 0.
- **--txpool.pricelimit & --rpc.allow-unprotected-txs:** De igual forma ca o anterior parámetro, son necesarios para a configuración e permiso da existencia de transaccións a custo 0 de gas nos nodos elixidos. Concretamente, para permitir que ditas transaccións se engadan á *pool* de transaccións para ser procesadas e para que sexan compatibles cas funcionalidades [RPC \(Remote Procedure Call\)](#) dos nodos, como debe ser no noso caso concreto.
- **console 2» myEth.log:** Establece o ficheiro *myEth.log* como arquivo de logs da Blockchain e de dito nodo. Serviranos para ver as transaccións, erros, e o estado en xeral da Blockchain, así como o histórico de eventos ocorridos nela, entre outras moitas cousas. Poderemos usar un mesmo ficheiro para varios nodos, ou como é o noso caso, un por nodo.

Os seguintes parámetros fan referencia ás conexións externas aos nodos da rede, por exemplo dende MetaMask ou dende a nosa futura [Dapp](#), polo que son valores críticos para unha adecuada configuración:

- **-rpc & -http**: Activa os servidores HTTP e o HTTP JSON-RPC para recibir peticións.
- **-rpcport & -http.port & -port**: Especifican os portos para que ditos servidores escoiten, así como **port** especifica o porto no que escoita a Blockchain.
- **-rpcapi & -http.api**: Lista de [APIs](#) ofrecidas polas interfaces [RPC](#) e HTTP para interactuar con dito nodo. É unha forma de restrinxir ou permitir a execución de certos comandos dispoñibles en ditas [APIs](#). Cada caso de nodo podería ter unhas [APIs](#) diferentes.
- **-rpcaddr**: Dirección na que o server está escoitando peticións. Utilizamos a dirección comodín por sinxeleza para o noso caso, pero nun entorno de produción habería que ser máis específico.
- **-rpcorsdomain & -http.corsdomain**: Lista de dominios dende os cales o servidor acepta peticións. O asterisco (*) indica que se permiten peticións dende calquera dominio, característica necesaria para o noso proxecto xa que virán das diferentes [Dapps](#).

Con isto, teríamos a nosa Blockchain privada creada a medida gracias á nosa configuración do **Genesis Block, despregada xunto cun nodo** (*end-point*) conectado á mesma e listo para poder recibir peticións. Polo que agora, o seguinte paso será implementar os **Smart Contracts** necesarios para o noso traballo, para así despregalos sobre esta nova Blockchain creada e poder comezar traballar coa información almacenada nela, ao ter xa un nodo minando peticións e facendo posible as futuras transaccións cas funcións e eventos de ditos contratos, os cales implementarán as funcionalidades do sistema referentes ao rexistro médico cos datos [COVID-19](#) e á nosa criptomoeda.

5.1.2 Smart Contracts: rexistro médico e criptomoeda

Para o desenvolvemento destes contratos, leváronse a cabo diferentes fases, sabendo que as dúas grandes funcionalidades a conseguir con eles eran a dun rexistro médico cos datos necesarios así como a da creación dunha criptomoeda que funcionara a modo de incentivo. Os seguintes subapartados describen os detalles da implementación de ambos.

Structs e modifiers do rexistro Médico

Hai que salientar a importancia do proxecto [32], o cal ofrece unha aproximación moi similar á que necesitamos, levando un rexistro médico de pacientes, os cales por certas ac-

cións recibían un incentivo a modo de *Token ERC-20*. Aínda que o motivo de ditos incentivos, os campos a gardar nos rexistros e certas funcionalidades máis, como a forma de verificar os pacientes ou a ausencia de centros médicos no sistema, non seguían a aproximación do noso modelo, foi un bo punto de partida para a partir dun esqueleto común, conseguir eses primeiros Smart Contracts. O primeiro, foi elixir as estruturas de datos acorde ao modelo de datos deseñado. Para elo, creamos o ficheiro do noso primeiro contrato, **MedicalRecord.sol** e nel comezamos cos **tipos de datos complexos** (*structs*) necesarios, en concreto os tipos **Dosis**, **Prueba**, **Record** e **Cdc**. Podemos consultar a súa implementación no apartado E.1.2 do Apéndice E. En dito código vemos como seguimos o exposto no modelo de datos, contendo todos estes tipos de datos creados datos de tipo *string* ou *address*. Tamén se seguiron certas convencións á hora de implementar ditos tipos de datos, como foron xuntar aqueles do mesmo tipo na declaración da propia estrutura, ou usar o tipo *address* para as direccións da Blockchain. Unha vez que os temos, haberá que deseñar a relación entre eles para crear así os **mappings** que amosabamos no deseño:

```
1 //mapping (address => Record) records; //opción tamén válida
2 mapping (uint256 => mapping (address => Record)) records;
3 mapping (address => Cdc) cdc;
```

Destacar agora que consultando diferentes fontes, vimos dúas principais aproximacións á hora de implementar os mappings, a que chamamos implementación **simple**, utilizada no mapping *cdcs*, ou a **dobre**, seguida co de *records*. Como ambas son válidas e ofrecen as súas respectivas vantaxes, optouse por implementar cada mapping dunha maneira, seguindo no principal, o modelo dobre empregado no proxecto de [32], e no dos centros médicos, creado por nós, a aproximación simple vista por exemplo en [58]. Consultando o anterior código e a maiores o relacionado con este no apartado E.1.2 do Apéndice E, vemos como temos os dous mappings deseñados no modelo para asociar tanto a un paciente co seu rexistro médico a través da súa dirección de *wallet*, como tamén o mesmo caso pero agora para os CDCs ca súa información en detalle. A maiores, haberá dous mappings extra, os cales relacionan unha dirección de *wallet* cun *booleano*, que nos será de axuda para facer comprobacións de se unha dirección consta no sistema como un paciente ou un CDC.

Con isto, xa estaría todo o traballo previo a comezar cas funcións. Aínda que antes de nada haberá que escribir os chamados **modifiers** en *Solidity*. Estes, son unhas funcións que actúan de comprobantes, ao ser chamadas nas cabeceiras doutras funcións co fin de, se non se cumpren, non executar dita función. Un exemplo co que se explica de forma sinxela este comportamento é o seguinte:

```
1 modifier patientExist(address patient) {
2     require(isPatient[patient]);
3     _;
```

O *modifier* será a función *patientExist*, cuxa sintaxe só quere dicir que é necesario que a dirección de *wallet* conste no sistema como a dun paciente, é dicir, teña asociado un *true* no

mapping *isPatient*. O guión baixo co que acaba é unha convención en *Solidity* para indicar que é unha función de dito tipo. Por outro lado, tanto no apéndice E, como no seguinte anaco de código mostrado nesta sección, as funcións *addDosis()* e *addRecord()*, antes de comezar ca súa lóxica, chaman a estes *modifiers*, feito que no primeiro caso se vería traducido en: sempre que se intente engadir unha dose a un paciente, compróbase que está no sistema e figura como paciente, e, en caso negativo, a función non é executada. Utilízanse varias funcións deste tipo ao longo do Smart Contract do rexistro médico, e as funcionalidades máis destacadas poden ser as de comprobar se unha dirección **é dun paciente**, se **é dun CDC**, se **xa existe no sistema**, se **é non nula**, se **é maior que 0** ou se xa se **recibiu algunha unha dose**, este último *modifier* moi útil para evitar engadir reaccións adversas por error a pacientes que aínda non tiveron unha dose inoculada. Podemos analizar máis en detalle o código de gran parte deles no Apéndice E ou ver na subsección seguinte outro exemplo do seu uso.

Funcións de escritura no rexistro médico

Cos tipos de datos listos e os *modifiers* implementados, quedará comezar cas funcións que engadan e lean a información dos rexistros médicos. Aínda que todas seguen un mesmo patrón e son sinxelas, imos ver aquela que engadir un novo paciente ao sistema:

```
1 function addRecord (uint32 _idNacional, uint32 _idMedico, address _diPac, (...))
2     public patientNotExist(_dirPac) cdcExist(msg.sender) {
3         records[recordCount][_dirPac].idNacional = _idNacional;
4         (...)
5         isPatient[_dirPac] = true; dirToRecord[_dirPac] = recordCount;
6         emit PatientRecordAdded(recordCount, _dirPac);
7         patientCount += 1; recordCount += 1;
8         payReward(_dirPac, 3);}
```

Aínda que é un código moi intuitivo, podemos ver como consta cos parámetros de entrada iguais aos campos do rexistro médico referentes aos datos persoais, xa que os médicos serán engadidos *a posteriori* co resto de métodos a implementar. Vemos que consta de **dous modifiers**, comprobando que o paciente non foi xa engadido, así como que a conta que está chamando a dita función, reflexado na palabra chave de *Solidity* **msg.sender**, cuxo valor é a dirección da conta que chamou a dita función, é un CDC, é dicir, unha conta autorizada para escribir no sistema. Despois disto, créase no **mapping** dos rexistros médicos aquel asociado ao número de rexistro actual e á dirección *wallet* do paciente, e énchense os campos ca información pasada como parámetro. Por último, convértese a **true** o valor de que dita dirección de wallet foi engadida ao sistema como paciente, e **asóciase** ao número de rexistro actual. **Súmase un** ao número de pacientes e rexistros presentes no sistema, **émítese un evento** ca información engadida a modo de resumen, dos cales falaremos en detalle a final desta sección, e **págase o incentivo** de maior valor, é dicir o terceiro, parte referente á criptomoeda que tamén comentaremos en detalle ao final deste capítulo. O resto de funcións que escriben na Blockchain, tanto doses, como probas ou reaccións, seguen exactamente o mesmo

comportamento. Para ver un exemplo delas só teremos que consultar de novo o Apéndice E.

Funcións de lectura do rexistro médico

Quedan por comentar as funcións de consulta á Blockchain, usadas para ler pasaportes. Delas haberá dous tipos, e podemos consultar seu código completo da mesma forma que ata agora no apartado E.1.2 do Apéndice E. O primeiro é o de función gratuíta, é dicir, que non costa Ether debido a que é de tipo **View**, o que quere dicir que só lee na Blockchain e non escribe. A segunda, aínda que fai o mesmo, escribe na cadea de bloques debido a que a maiores da un incentivo ao paciente ao cal se lle leu o pasaporte, polo tanto xa non ten 0 custo de Ether. A diferenza entre elas tamén recaerá en que na primeira, un evento é disparado a modo de resumo da transacción, neste caso, devolvendo os datos do pasaporte. Polo que, sen esas últimas dúas liñas, a súa lóxica é idéntica.

```

1  function getRecord(address _direccionPaciente) public returns(
2      uint32 _idNacional, (...), string[] memory _reaccionesAdversas){
3      _idNacional = records[recordID][_direccionPaciente].idNacional;
4      (...)
5      _reaccionesAdversas = records[recordID][_direccionPaciente].reaccionesAdversas;
6      emit PatientRecordAccesed(_idNacional, (...), _reaccionesAdversas);
7      payReward(_direccionPaciente, 1);}

```

Criptomoeda

Imos ver agora a parte referente á criptomoeda. Constará principalmente de dous contratos, un referente ao *Token*, e outro para o Faucet personalizado. Por último, faltaría engadir ao anterior contrato do rexistro médico a información referente á criptomoeda personalizada. O contrato do *Token*, é dicir, o que implementa a criptomoeda, aínda que se pode consultar íntegro xunto ao resto no Apéndice E, será da forma:

```

1  contract Coronacoin is StandardToken, Ownable {
2      constructor() public {
3          totalSupply_ = INITIAL_SUPPLY;
4          balances[msg.sender] = INITIAL_SUPPLY;
5          emit Transfer(address(0x0), msg.sender, INITIAL_SUPPLY);}}}

```

Só teremos que definir o seu nome, o acrónimo e a cantidade inicial dispoñible, a cal será transferida á dirección que chame ao construtor do mesmo (*msg.sender*). Para entender isto mellor, podemos consultar o Apéndice E e ver o código ao completo en detalle. No contrato do rexistro médico haberá que referenciar á criptomoeda definindo unhas variables, as cales servirán para xestionar tanto a criptomoeda coma os incentivos a outorgar. Despois, para inicializar dita criptodivisa, teremos que chamar ao construtor apropiado. De novo, todo este código pódese consultar no apartado do Apéndice E.1.2. Inicializaremos tamén os tres valores de incentivos, de menor a maior (10, 100 e 1000), así como o propio *Token* ca súa dirección. Para comezar agora traballar con el, haberá que implementar diversas funcións

para pagar os incentivos, consultar o saldo (estas dúas vémolas a continuación) e para cambiar a configuración cando se desexe.

```
1 function payReward(address _patientAddress, int number){...}{
2   if(number == 3){coronacoin.transfer(_patientAddress, tokenRewardBigAmount);
3   emit RewardPaid(_patientAddress);}
4   function balanceOfCoronacoins(address _patientAddress) public view (...){
5     return coronacoin.balanceOf(_patientAddress);}
```

Antes de comezar co Faucet, hai que mencionar un aspecto importante como é o da **herdanza**. Aínda que tan só tres Smart Contracts son os críticos no proxecto, os do rexistro médico, da criptomoeda e do Faucet, todos herdán e utilizan outros contratos intelixentes xa dispoñibles para os desenvolvedores en *Solidity*, os cales brindan unha serie de funcionalidades pre-establecidas como poden ser as do contrato **Ownable**: herdando de dito contrato, poderemos tratar casos de uso onde sexa útil ter un dono ou posuidor, por exemplo no anterior caso, utilizando o *modifier onlyOwner* para asegurarnos de que tan só o dono de dito contrato, é dicir, quen o despregou, pode chamar a ditas funcións. Debido a isto, a estrutura final dos Smart Contracts non estará formada só polos tres comentados, senón que contará con todos aqueles contratos necesarios para conseguir as funcionalidades implementadas. No Apéndice E, na Figura E.1, pódese observar a organización final dos mesmos, onde vemos todos aqueles contratos que nos axudan a mellorar as funcionalidades do sistema. Estes, os cales non foron desenvolto por nós e pódense ver como librerías externas, axudarán a maiores do comentado a tarefas como xestionar os *tokens*, facilitando o seu envío e recepción (e.g., *ERC20.sol*), ou a securizar diversas funcionalidades como as operacións aritméticas (e.g., *SafeMath.sol*).

Por último, e para acabar ca parte referente aos contratos relacionados ca criptomoeda, teremos aquel que implementa a funcionalidade do noso **Faucet personalizado**, a cal consistirá en dar Ether baixo demanda ás *wallets* que o soliciten mediante transaccións a prezo cero, para dotalas así dunha cantidade inicial de Ether para que poidan comezar enviar transaccións con custo e aproveitar así todas as funcionalidades do sistema. Parte da función máis importante dentro deste contrato, a cal vemos íntegra no apéndice E, será:

```
1 uint balance = address(this).balance;
2 if (balance < 1 ether){ emit etherOutOfStock(balance);
3 }else(...){
4   msg.sender.transfer(100 ether);
5   emit etherGiven(msg.sender);}}
```

Dita función dará ou unha cantidade fixa de Ether por defecto, ou ben unha porcentaxe se a cantidade de Ether dispoñible no contrato é menor a un limiar, para evitar así a inanición do mesmo. Implementará tamén a capacidade de recibir Ether ca función de *receive()*, xa que dito Smart Contract terá que poder facelo para despois transferilo, así como contará tamén cunha serie de eventos para informar de todos aqueles sucesos que ocorran arredor do *CoronaFaucet*.

Eventos

Imos para acabar con esta sección falar dos **eventos**. Os eventos dos Smart Contracts son un tipo de función á cal se chama durante a execución dos mesmos, por exemplo no código dun método, e os cales emiten unha notificación na Blockchain coa información que nos queiramos asociar a eles. Por exemplo, no contrato do Faucet anterior temos un evento para notificar que se acabou o Ether. Os eventos (os cales podemos consultar íntegros no Apéndice E), servirán para notificar na cadea de bloques aquelas accións máis relevantes, que van dende a escritura dun novo paciente ou dunha dose/proba/reacción alérxica no rexistro do mesmo até a devolución da información dun paciente ao ler seu pasaporte ou a emisión de Ether por parte do noso Faucet. Estes eventos, facilitarán por tanto diversas xestións na Blockchain e poderán ser consultados en calquera momento xa que se almacenan na mesma, proporcionando así unha potente ferramenta de auditoría e análise das transaccións, así como tamén unha valiosa fonte de información acerca dos sucesos ocorridos na cadea. Serán fundamentais en practicamente todos os Smart Contracts e definiranse da seguinte forma:

```

1  event PatientRecordAccesed(uint32 _idNacional, (...) string[] _reaccionesAdversas);
2  event TestAdded(uint256 recordID, address patientAddress, Prueba newTest);
3  event RewardPaid(address patientAddress);
4  event etherGiven(address patientAddress);

```

5.2 Probas

En canto ás probas, aínda que todas elas teñen un carácter similar, é certo que se poden separar nestes dous mesmos módulos, aínda que para realizar as do segundo, necesitan estar listas as do primeiro, pois, se a Blockchain non foi ben despregada, os Smart Contracts serán inútiles xa que non se poderán engadir á Blockchain. Imos pois, ver as probas individuais realizadas de cada módulo, para finalizar cunha proba conxunta probando un fluxo completo da nosa cadea de bloques, recibindo peticións, minando transaccións e facendo uso das funcionalidades implementadas. Comentaranse todas as probas realizadas, e poderemos atopar imaxes que as ilustren no apartado E.2 do Apéndice E, pero, por limitacións de extensión, nesta sección só se acompañará con imaxes ás explicacións daquelas probas máis relevantes.

5.2.1 Blockchain Privada

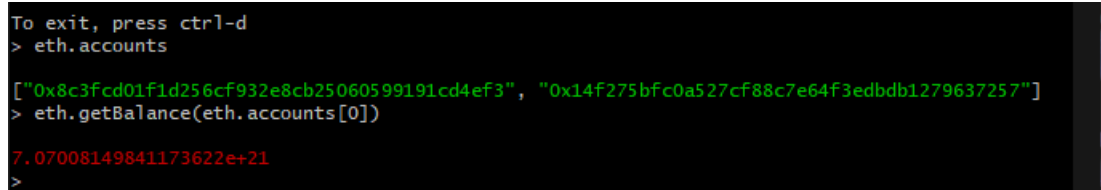
O primeiro de todo, tras executar os comandos de despregue sen erros, consistirá en executar comandos Geth na terminal que se nos abre despois de despregar o nodo, facendo uso dos mesmos para comprobar que os funcionamentos básicos da cadea de bloques son correctos. Resúmense en:

Post-execución dos comandos

Tras executar o comando de inicio do nodo do que falamos na sección de implementación e amosamos no apartado 5.1.1, deberemos observar o ficheiro ao que enviamos os logs para ver que ocorre en dito proceso. O resultado vémosto na Figura E.3 do noso Apéndice E, onde, xunto ca Figura E.2, onde temos a execución do comando da que falamos, podemos analizar que módulos se cargaron, o *datadir* da cadea de bloques, o último bloque engadido e o *coinbase*, establecido na primeira conta por defecto. Teremos tamén a información dos puntos *RPC* e *HTTP* despregados, a confirmación da conta desbloqueada así como do *etherbase* ou *coinbase* do nodo, entre outras cousas. Todo isto en conxunto, é información máis que suficiente para probar que o despregue do nodo foi exitoso e estamos preparados para comezar interactuar ca Blockchain. Para continuar cas probas, testemos os comandos máis básicos da consola Geth a continuación, onde veremos ilustrados gran parte dos datos anteriores.

Comprobación de contas e saldo

Consultando a Figura 5.1, poderemos ver a saída que nos confirma que as contas especificadas no *Genesis Block* foron creadas satisfactoriamente, así como que a primeira deles ten o Ether especificado no mesmo (o 0.7 extra débese a uns segundos de minado). Vemos polo tanto, que a configuración especificada no *Genesis Block* está presente na nosa cadea e pode ser probada a través destes comandos Geth.

A screenshot of a Geth console window with a black background and white text. The text shows the following sequence of commands and outputs: 'To exit, press ctrl-d' followed by a prompt '>'. Then the command 'eth.accounts' is entered, resulting in an output of two hexadecimal strings in green text: '["0x8c3fcd01f1d256cf932e8cb25060599191cd4ef3", "0x14f275bfc0a527cf88c7e64f3edbdb1279637257"]'. Next, the command 'eth.getBalance(eth.accounts[0])' is entered, resulting in an output of '7.07008149841173622e+21' in red text. The prompt '>' appears again at the end.

```
To exit, press ctrl-d
> eth.accounts

["0x8c3fcd01f1d256cf932e8cb25060599191cd4ef3", "0x14f275bfc0a527cf88c7e64f3edbdb1279637257"]
> eth.getBalance(eth.accounts[0])

7.07008149841173622e+21
>
```

Figura 5.1: Comprobación de contas e saldo da coinbase na nosa Blockchain.

Creación dunha nova conta e saldo, establecemento como coinbase e minado

Imos por último, probar a nosa Blockchain cun fluxo completo encargado de crear unha nova conta dende o nodo, ver o seu saldo inicial, o cal debería ser 0, establecela como *coinbase* do nodo para que reciba os incentivos dos bloques minados por el, e comezar a minar, para comprobar tanto no log, como tamén no seu saldo, o que ocorre. Todo isto atópase nas Figuras 5.2 e 5.3 amosadas a continuación.

Vemos na primeira das figuras, na 5.2, como a consecución da execución e saída dos comandos foi a desexada. En primeiro lugar, creamos unha nova conta co seu *passphrase* e

comprobamos que se engadiu ás dúas xa creadas no xénese. Vemos tamén como o seu balance inicial é 0 ether, pero tras establecela como *coinbase* e comezar minar, tras uns segundos, vemos como aumentou chegando case aos 20 de Ether, confirmando o correcto minado de bloques da cadea. Aquí vese tamén reflexado ese parámetro de **difficulty** configurado no *Genesis Block*, xa que en escasos segundos, se conseguiu unha gran cantidade de criptodivisa, debido a que o minado de bloques é sinxelo e polo tanto rápido, comportamento que como ben explicamos, queremos na nosa cadea.

```
To exit, press ctrl-d
> personal.newAccount("memoria")

"0x73c8b0e928823c597b4c079fd84b0dbd220642cf"
> eth.accounts

["0x8c3fcd01f1d256cf932e8cb25060599191cd4ef3", "0x14f275bfc0a527cf88c7e64f3edbdb1279637257", "0x73c8b0e928823c597b4c079fd84b0dbd220642cf"]
> eth.getBalance(eth.accounts[2])

0
> miner.setEtherbase(eth.accounts[2])

true
> miner.start()

null
> eth.getBalance(eth.accounts[2])

18000000000000000000
> miner.stop()

null
> |
```

Figura 5.2: Proba múltiple do nodo despregado na Blockchain privada.

```
INFO [06-01|18:49:25.427] Your new key was generated      address=0x73C8B0E928823C597B4C079FD84B0DBD220642CF
WARN [06-01|18:49:25.427] Please backup your key file!      path=C:\Users\Usuario\Desktop\TFG\privateChain\chaindata\keystore
UTC--2021-06-01T16:49:23.660436900Z--73c8b0e928823c597b4c079fd84b0dbd220642cf
WARN [06-01|18:49:25.427] Please remember your password!
```

Figura 5.3: Logs da proba múltiple do nodo despregado na Blockchain privada.

Por outro lado, nos logs da Figura 5.3, vemos información referente ás accións que ocorreron. En primeiro lugar, como se xerou a nosa chave e o *path* do noso ficheiro JSON cuxo nome comeza por *UTC*. Dito ficheiro é de gran importancia, xa que este directorio de **keystore** haberá que manexalo en varias ocasións durante o proxecto. Nel, haberá ficheiros con nomes que comezan polo patrón dun instante temporal da forma de **UTC-ano-mes-dia-hora-dir_wallet**. Estes ficheiros serán os que representen as contas da Blockchain e serán necesarios para importalas en Metamask. Haberá que incorporalos manualmente a dito directorio se as contas especificadas no *Genesis Block* non son creadas manualmente como se fixo da forma ca conta *0x73..cf* mostrada no exemplo anterior.

Con estes comandos será suficiente, xa que serán os únicos que usemos a maiores das in-

teraccións cos contratos, así que serven para probar que a Blockchain foi despregada e funciona correctamente. As seguinte fases para probar o despregue serán cambiando de paradigma de proba, e dende Remix e Metamask, ver que podemos conectarnos a ela así como importar unha conta da mesma e conectarnos con ela para enviar transaccións. Posteriormente, poderemos probar o despregue e funcionamento dos nosos contratos na nosa cadea.

5.2.2 Remix + Metamask: Cadea privada e primeiros contratos

Nesta sección faremos dúas probas, a primeira, tras instalar e configurar Metamask da forma exposta no Apéndice E, conectarámonos á nosa cadea despregada da forma amosada na Figura 5.4a para enviar 6 de Ether probando as conexións e transaccións na mesma, como vemos na Figura 5.4b. A maiores, nesta última imaxe podemos corroborar a correcta importación de contas da cadea (*TFG 2nd Account*) e o correcto saldo das mesmas (*20 Eth*).

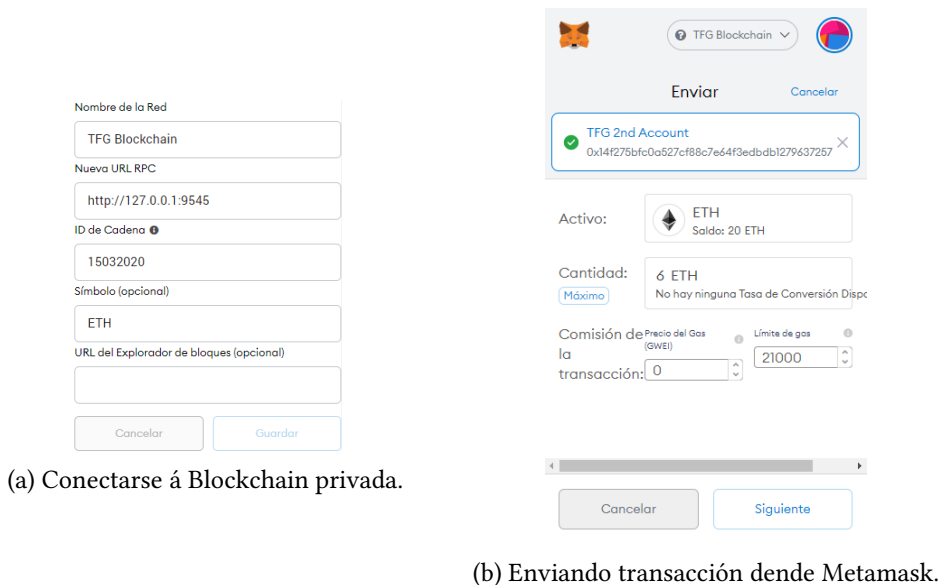


Figura 5.4: Probando as contas da cadea con Metamask.

```
INFO [06-01|19:19:19.612] Submitted transaction hash=0x337a3a14c9fae8eb5835d1f5598c6fc950588bf21175715e0fda2bed5a
bbf26d from=0x73C880e928823C597b4C079Fd8480bd220642cF nonce=0 recipient=0x14f275bfc0A527Cf88C7E64F3Edbd1279637257 value=6000000000
000000000
```

Figura 5.5: Comprobando os logs da Blockchain tras o envío da transacción.

Analizando os logs da cadea na Figura 5.5, vemos como de forma correcta as transaccións chegan para ser minadas, confirmando así unha importante parte do funcionamento da nosa infraestrutura. Quedará agora facer as primeiras probas cos contratos, as cales consistirán en despregalos de forma *mock*, como vemos na Figura 5.6a, onde consultamos a dirección do noso contrato despregado nunha EVM de Remix, e en facer interaccións co mesmo, en

concreto chamando ao método `addCdc()` que vemos na Figura 5.6b, exemplificando a proba dos contratos *via* Remix e a súa correcta execución, co log de Remix da Figura 5.7. Salientar que no subapartado E.2.2 do Apéndice E podemos atopar máis probas deste tipo.

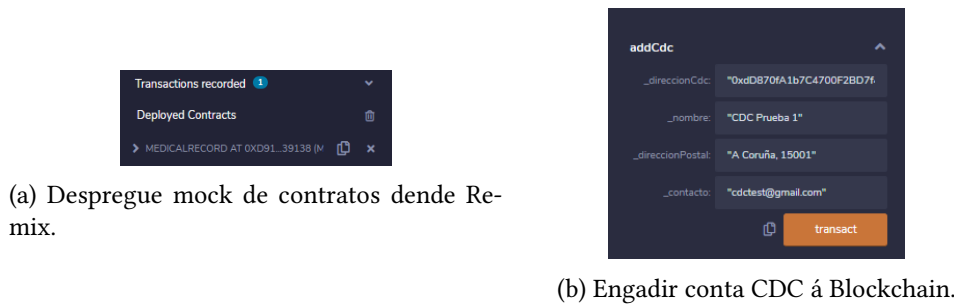


Figura 5.6: Probando as contas da cadea con Metamask.

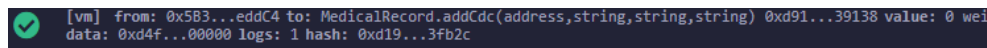


Figura 5.7: Resultado da proba mock de engadir conta CDC á Blockchain.

5.2.3 Smart Contracts: rexistro médico e criptomoeda

Finalizada a codificación dos contratos mencionados e comprobada a compilación sen erros, así como que funcionan correctamente cas probas *mock* anteriormente amosadas, chegará o momento de despregalos e probar o seu comportamento na nosa Blockchain.

Despregue e proba na Blockchain privada

Neste apartado probaremos parte dos métodos dos contratos da Blockchain, facendo unha proba conxunta de ambos módulos desta parte. Para elo, o primeiro paso será configurar Remix para que se comunique ca nosa cadea de bloques, testeándoa tamén a ela. Para conseguilo, no modo de despregue elixiremos **Injected Web3**, á vez que dende MetaMask nos conectaremos a unha das diferentes contas importadas como vemos na Figura 5.9a, para ir enviando así transaccións á Blockchain dende ela. Todos estes pasos vense íntegros a continuación.

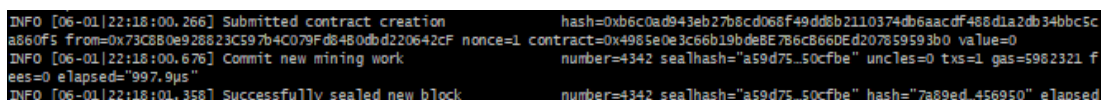
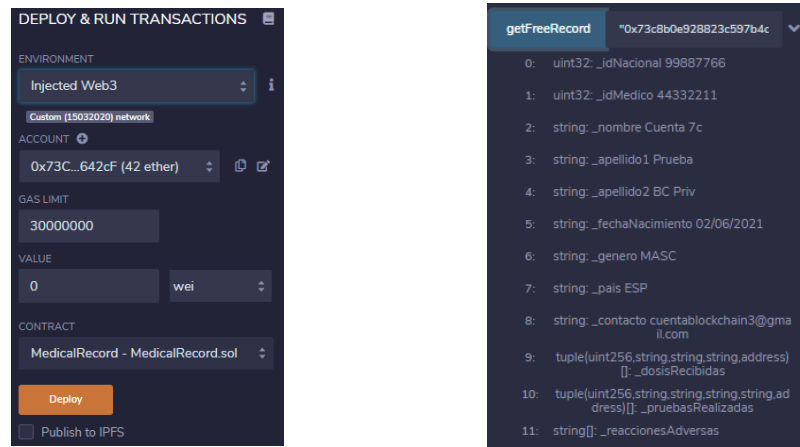


Figura 5.8: Probas na Blockchain privada: Primeiros pasos.

Como vemos na Figura 5.9a, tras elixir a configuración correcta e clicar no botón despregar, de igual maneira que no caso anterior, o contrato desprégase na nosa cadea de bloques



(a) Configuración Remix para a cadea privada.

(b) Lectura gratuíta do rexistro.

Figura 5.9: Probas na cadea privada: Primeiro paso e lectura gratuíta do usuario engadido.

correctamente, feito que podemos corroborar a maiores que dende Remix (Figura E.13 dispoñible no Apéndice E), tamén dende o log da Blockchain, onde como vemos na Figura 5.8, a transacción foi recibida e minada correctamente. Unha vez o temos despregado, ca súa dirección na Blockchain incluída, poderemos comezar cas probas, as cales poderemos analizar en detalle co log da Blockchain a maiores da información de Remix, como na Figura 5.8. Imos probar pois nesta sección, as funcionalidades de lectura do pasaporte xunto ca emisión de incentivos, para posteriormente consultar o saldo de criptomoedas e probar así tamén funcionalidades referentes ao contrato referente á criptomoeda. Antes de nada, como estamos na nosa Blockchain, tívose que incorporar información á mesma para poder lela, polo que se engadiu a conta principal, a `0x8c3...`, como CDC, e a conta creada dende cero nas anteriores seccións, a `0x73x...`, como paciente. Vemos a información da mesma á vez que probamos a chamada á lectura gratuíta do pasaporte na Blockchain dende Remix na Figura 5.9b.

Con esta información, procedemos facer unha chamada á lectura **non gratuíta** do pasaporte, como vemos na Figura 5.11a. Recordamos que cando dividimos as lecturas en gratuíta e non gratuíta, referímonos á súa división segundo se o método ao que se chama é de tipo **View**, é dicir só de lectura, sen coste de transacción, ou polo contrario, se a lectura leva implícita unha escritura na Blockchain, referente ao outorgamento dun incentivo. Esta última chamada, será aquela que se faga primeiro dende a app ao ler un pasaporte, para darlle así o incentivo ao paciente que amosa o pasaporte en primeira instancia, para ben posteriormente chamar en segundo lugar ao método gratuíta para así mostrar a información na Dapp. Poderíamos tamén obter esta información do paciente grazas ao evento emitido na primeira das lecturas, na non gratuíta, o cal podemos consultar en detalle ampliando a imaxe 5.11a, pero polos motivos expostos no último capítulo da memoria decidimos implementalo desta segunda maneira.

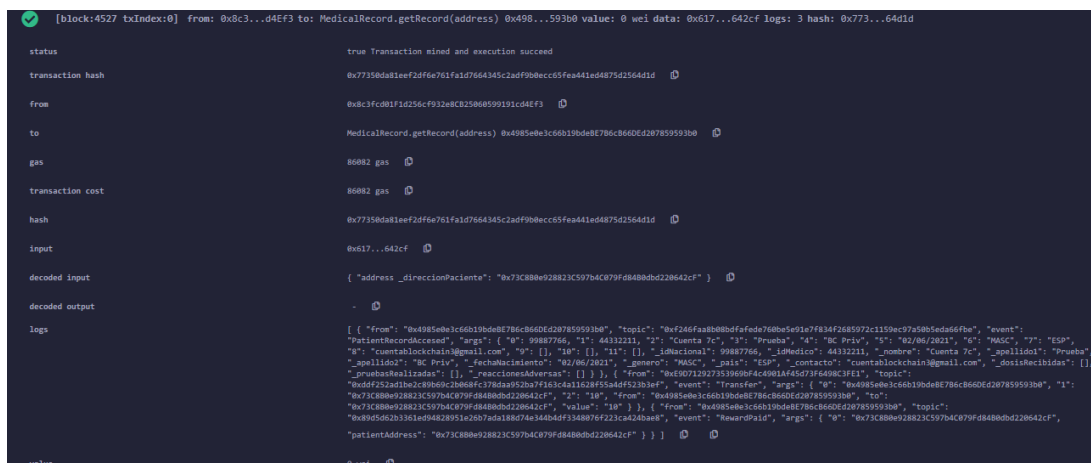
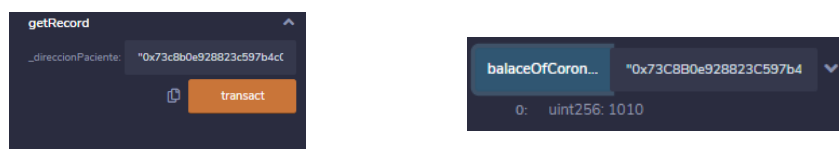


Figura 5.10: Debug da lectura NON gratuita.



(a) Chamada á lectura NON gratuita.

(b) Consulta do balance de criptomonedas.

Figura 5.11: Chamada á lectura non gratuita e consulta de coins.

Vemos na Figura 5.11a como introducindo a dirección do paciente lemos o seu pasaporte, agora ben, a información devolverase nun evento con ditos campos habendo un coste por transacción debido aos incentivos como explicamos e vemos na Figura 5.10. Aínda que xa vimos a execución correcta da transacción e a visualización do seu resultado, para corroborar o seu funcionamento e probar unha funcionalidade do contrato da criptomoneda (pago de incentivos), imos chamar ao método encargado de obter o balance de criptomonedas dunha conta, o da `0x73...`, é dicir, a do paciente que foi engadido e do cal se leu o pasaporte. O resultado de dita chamada, como vemos na Figura 5.11b, foi **1010**, que se repasamos a parte de implementación, vemos que é igual á suma de 1000, o maior dos incentivos, provinte de engadir un paciente ao sistema, máis 10, o menor incentivo, outorgado en cada lectura do pasaporte.

Quedan probadas así ditas funcionalidades, tanto do rexistro médico como da criptomonedas, así como tamén o correcto funcionamento da Blockchain, á cal nos podemos conectar e enviar transaccións, as cales son minadas exitosamente. Por outro lado, amosáronse ambos paradigmas de probas, tanto directamente contra á Blockchain dende Geth, como dende Remix e Metamask, así como se complementaron cas probas do Apéndice E para detallar todos os tests feitos ao noso sistema. Toca polo tanto, comezar ca app móbil que o use, ca súa implementación e cas súas probas, para así probar o resto de funcionalidades dispoñibles nos Smart Contracts despregados e as diferentes comunicacións dende a app cos mesmos.

Aplicación móbil distribuída

Este capítulo fará referencia a todos os conceptos involucrados na implementación e probas da parte do proxecto referente á **aplicación móbil**. Haberá **dúas versións** da mesma, é dicir, dúas aplicacións diferentes que como ben explicamos anteriormente corresponderanse ca versión para os pacientes e a versión para as autoridades. Ambas, serán moi similares e compartirán un esqueleto común, aínda que terán grandes diferencias nos permisos á hora de traballar sobre a Blockchain. Por último, mencionar que ditas aplicacións foron elaboradas nun entorno multiplataforma, concretamente ca ferramenta [Kotlin Multiplatform Mobile](#).

6.1 Implementación

Á hora de falar da implementación da app (a partir de agora imos tratala de forma xeral como unha, menos cando especifiquemos unha versión), podemos dividila en varios módulos. Por un lado, temos as tarefas puramente programáticas, que consistirán en desenvolver o código de Kotlin que se encargue das interaccións ca Blockchain así como de crear os diferentes *layouts* ou interfaces, disto falaremos na sección de lóxica destacada 6.1.1. Por outro lado, temos as partes máis xenéricas da implementación, como a organización do código, chave en [KMM](#) á hora de optimizar o despregue híbrido, ou a utilización de certos frameworks externos que nos permitan adaptar conceptos referentes á Blockchain, como por exemplo *Web3j*. Imos ver as tarefas de implementación máis relevantes de ambos módulos, para despois amosar as probas resultantes de dita implementación na última sección do capítulo.

6.1.1 Lóxica destacada

A seguinte sección fará referencia ao código elaborado, e xa que é imposible mostralo todo pola limitación de extensión, tratará de amosar aquela lóxica máis destacada, sendo esta ben ou a máis crítica, debido a que **implementa funcionalidades chave**, ou ben **a máis habitual**, detallando a máis utilizada polas apps. Por outro lado, amosará anacos de código

de todas as partes ou módulos da app, dende o referente á Blockchain, pasando polo tratamento de QRs até as comunicacións con Firebase ou a utilización de métodos criptográficos, para aínda que non en tanto detalle, deixar constancia da heteroxeneidade do traballo de implementación que houbo no proxecto. Darase sempre unha explicación de toda esta lóxica, e como no capítulo anterior, a que non vaia acompañada dun exemplo visual ou a que estea incompleta nesta sección, terá os elementos restantes no seu apéndice correspondente, neste caso o [F](#), o cal complementará esta sección. Recordamos de novo que todo aquel código non amosado na memoria, nin nos capítulos centrais nin nos apéndices, atópase no repositorio público amosado no Apéndice [G](#) baixo licenza *Open Source*.

Conexión da Blockchain

Comezamos ca primeira funcionalidade a implementar: conectarse á Blockchain privada con éxito, para facer así algunha lectura exitosa na mesma. Grazas á consulta de artigos como [\[59\]](#) ou [\[60\]](#), puidemos ter a nosa primeira implementación, a cal, do mesmo modo que ocorreu ca implementación da parte Blockchain, comezou ca xestión das funcionalidades máis simples: lecturas da propia Blockchain e comandos referentes a Geth, deixando en primeira instancia de lado os contratos intelixentes. Nesta fase, as funcionalidades a implementar baseáronse en conseguir conectarnos con éxito á Blockchain e crear unha *wallet* na mesma, xa que son tarefas moi habituais e críticas no proxecto. Delas podemos destacar a seguinte lóxica aínda que podemos ver o código completo no apartado [F.1.1](#) do Apéndice [F](#).

```

1  var web3: Web3j =
2      Web3j.build(HttpService("http://192.168.0.14:9545"))
3      (...)
4      web3.web3ClientVersion().sendAsync().get()
5      (...)
6      setupBouncyCastle();

```

Este anaco de código será aquel ao que teremos que acudir cada vez que queiramos conectarnos á nosa Blockchain para realizar calquera acción. Vemos como indicamos o *end-point* do nodo despregado e conectado á Blockchain, explicado no anterior capítulo, para despois utilizar certas librerías chave para as diferentes tarefas a implementar. En concreto:

- **Web3j:** Punto central sobre o que xirará toda a lóxica do proxecto ao brindarnos as funcionalidades básicas referentes a todos os tipos de conexións e interaccións ca Blockchain.
- **HttpService:** Socket HTTP para conectarnos ao *end-point* do noso nodo e enviar as peticións ca información pertinente (transaccións).
- **Web3ClientVersion:** Facilitaranos a comprobación da correcta conexión á Blockchain ao realizar unha sinxela lectura na mesma.

- **setupBouncyCastle**: Función cuxa implementación podemos atopar en [59], e a cal será chave debido a que implementará aqueles algoritmos criptográficos necesarios para manexar a seguridade da Blockchain e a compatibilidade con *Web3j* [59]. Sen ela, as conexións ca nosa cadea Ethereum serían imposibles de realizar.

Escrituras básicas e Encrypted Shared Preferences

Unha vez temos a conexión establecida, tocará facer as primeiras lecturas e escrituras na cadea cos comandos Geth, antes de abordar os Smart Contracts. Para isto, implementamos a lóxica da escritura dunha nova *wallet* ou conta na cadea.

```
1  fileName = WalletUtils.generateLightNewWalletFile(pwd, walletDir)
2  var credentials: Credentials = WalletUtils.loadCredentials(pwd, walletDir)
3  val sharedPreference: SharedPreferences = EncryptedSharedPreferences.create(...)
4  with(sharedPref.edit()) {...}
5  return credentials
```

Neste fragmento de código (o cal atopamos completo no apartado F.1.1 co Apéndice F) destacamos a creación da *wallet* na Blockchain, a cal dependerá de dúas chamadas críticas, aquela á función **generateLightNewWalletFile**, a cal pertence á *suite* de **WalletUtils**, utilizada para xestionar as contas da Blockchain, e que recibirá o contrasinal da conta a crear así como o directorio onde crear o ficheiro de dita conta. Este ficheiro, será da forma que comentabamos en capítulos anteriores: **UTC-xxx**, os cales se atopaban na *keystore* da Blockchain. Desta vez, dito ficheiro será almacenado no dispositivo, pero para asegurar a non dependencia co móbil dende onde se creou dita conta, almacenaremos tamén a chave privada da mesma, a cal se poderá utilizar para loguearse dende outro móbil, xa que como recordamos eran as dúas principais maneiras de facelo. En canto a esta chave privada hai que salientar que será ofrecida ao usuario para que a conserve nun lugar seguro para poder acceder á súa conta en caso de cambio de dispositivo, e que será almacenada de forma segura nas **Encrypted Shared Preferences** do dispositivo, segunda chamada crítica, para así poder lela cada vez que se necesite firmar unha transacción. Para usar dita clase, a cal será moi similar ás *Shared Preferences* convencionais, haberá que acceder ao ficheiro de datos chave-valor e facer uso das librerías propias, as cales se encargan de que dito ficheiro estea cifrado co algoritmo que indiquemos (no noso caso, **AES**), para aínda podendo editalo en texto claro de forma sinxela co obxecto *editor()*, seguir tendo a información almacenada de xeito seguro.

Miscelánea: Firebase, Algoritmos Criptográficos e Manexo de códigos QR

Antes de comezar ca lóxica vital do proxecto, aquela referente ás transaccións na Blockchain, hai que mencionar rapidamente tres conceptos: o primeiro deles será **Firebase**, o cal como ben explicamos na sección 4.2, será o software elixido para facerse cargo da xestión de usuarios no noso sistema. Con Firebase, encargaremos de crear usuarios, loguearnos na app

cas nosas contas e verificar as mesmas. Todo isto, co respaldo da súa consola online e a gran potencia de combinalo xunto con Android Studio, que co seu plugin facilita en gran medida todas estas tarefas. Aínda que en [61] e [62] atopamos dous artigos onde se explica en detalle a xestión de usuarios de Firebase dende Kotlin, amosase a continuación un fragmento de código relevante, podendo consultar o resto no apartado F.1.1 do Apéndice correspondente a este capítulo.

```

1      firebaseAuth.createUserWithEmailAndPassword(userEmail, userPassword)
2      .addOnCompleteListener { task -> if (task.isSuccessful) {
3  private fun sendEmailVerification() { firebaseAuth.currentUser?.let {
4      it.sendEmailVerification().addOnCompleteListener { task ->

```

Vemos dúas das funcións máis importantes desta parte, a da creación dunha nova conta así como a encargada de verificala *via* email, tarefa chave á hora de decidir se outorgar Ether ou non á conta, como veremos no capítulo de probas. No apéndice F, vemos uns fragmentos de código referentes tanto ao cifrado simétrico do pasaporte, para o seu posterior envío seguro, como á xestión dos códigos QR. Nos artigos [63] e [64] podemos ver os exemplos de forma máis extensa e detallada. En ditos fragmentos vemos código encargado do correcto funcionamento de funcionalidades accesorias da app que por orde encargárase de: abrir o **escáner QR** que a librería **ZXGing** nos facilita, **crear un QR** a modo de imaxe (*bitmap*) a partir de calquera *string*, así como por último **cifrar simetricamente unha mensaxe** (parte do código é dunha clase Java que nos brinda ditas funcionalidades), as cales usaremos para cifrar a nosa dirección de *wallet* co contrasinal pre-acordado con destino e enviar o resultado de dito cifrado a modo de ficheiro, conseguindo así a funcionalidade de envío seguro do pasaporte.

Transaccións ca Blockchain

Queda por tanto a parte máis importante do código a realizar na **Dapp**, aquel que se encargue de **enviar transaccións á Blockchain** para comunicarse así cos Smart Contracts e chamar aos seus métodos. O obxectivo desta parte da implementación será que da forma máis óptima posible, dende Kotlin, se fagan as chamadas cara os métodos de ditos contratos, agora invocándoos integramente dende a lóxica da aplicación móbil e non dende Remix ou Geth, para poder utilizar así a infraestrutura Blockchain dende calquera móbil que teña dita **Dapp**. Antes de nada, deberemos saber que para interactuar cos contratos, necesitaremos xerar uns **Wrappers Java** a partir do código *Solidity*, para así poder acceder aos seus métodos, eventos, variables. Dito proceso será chave na fase de implementación e conseguirase tras facer o seguinte, exemplificado con comandos e capturas no Apéndice F.

- **Copiar o ABI (Contract Application Binary Interface) do Smart Contract a un ficheiro:** En Remix, teremos que ir á xanela de compilación, para facer click no botón do portapapeis xunto a palabra **ABI** para copiar o mesmo a un ficheiro (Figura F.1). O **ABI**, non é máis que o **modo estándar de interactuar con contratos** no ecosistema

Ethereum [45]. É dicir, é como unha especie de [ABI](#) ou interface para que calquera elemento interactúe cas funcionalidades outorgadas por un Smart Contract.

- **Xerar os wrappers con Web3j e utilízalos:** Unha vez que temos o [ABI](#) do contrato, deberemos usar os comandos de consola *Web3j* para que automaticamente xere os *wrappers* dos mesmos e os importe no proxecto. Todo isto, fariase co comando amosado no apartado F.1.1. No cal vemos como bastará con indicar o *path* do [ABI](#), o paquete onde queremos gardar *wrappers* e o directorio de saída pertencente ao proxecto. Agora, unha vez que temos os *wrappers* importados no proxecto, tan só haberá que utilízalos como clases Java, traducidas a código listo para importar no proxecto e ser chamado.

Unha vez temos isto, só quedará realizar as transaccións que necesitemos contra a Blockchain, utilizando os métodos apropiados en cada momento. Veremos tanto transaccións de lectura, como de escritura, xa que ambas serán moi similares. Grazas ao artigo [60] puídemos elaborar a lóxica adecuada para facer ditas chamadas, a cal poderemos consultar na súa totalidade no Apéndice F e ver un breve resumo a continuación.

```
1 val credentials: Credentials = Credentials.create(ECPair.create(BigInteger(key)))
2 val contractAddress = "0x0c8E392EF799F3EADBAE7B4d780716533ad03347"
3 val gasProvider = StaticGasProvider(...)
4 val medicalRecord = MedicalRecord3.load(contractAddress, credentials, (...), gasProvider)
5 val transactionReceipt (...) = medicalRecord.getFreeRecord(wallet).sendAsync().get()
6 val transactionReceipt0 (...) = medicalRecord.addDosis(acc, nLote, (...), lugar).sendAsync().get()
7 val balance: EthGetBalance = web3.ethGetBalance(...).sendAsync().get()
```

A situación tanto á hora de facer lecturas como escrituras será practicamente idéntica, e para todas as chamadas aos contratos haberá que seguir un patrón común, variando tan só o acceso e a xestión dos datos devoltos por cada método. As partes para facer a conexión e envío das transaccións de forma correcta resúmense en:

- **Credentials:** Deberemos indicar dende que conta imos realizar dita transacción. Para facer isto poderíamos elixir ou ben usar o noso ficheiro JSON da *wallet* (da forma *UTC-xx*, do cal xa falamos) ou a nosa chave privada (almacenada de forma segura nas *Encrypted Shared Preferences*). Polo que, como se contempla un hipotético caso de cambio de dispositivo onde só valería a segunda opción, utilizaremos a chave privada do usuario para crear as credenciais cas que firmar a transacción.
- **Contract Address:** Dirección do Smart Contract despregado na Blockchain cuxa interface é a mesma ca o da [ABI](#) que xerou os *wrappers* cos que nos imos comunicar.
- **gasProvider:** Mecanismo para indicarlle ás transaccións o límite e prezo de gas. Moi útil para manexar transaccións moi custosas ou pola contra, gratuítas.

- **medicalRecord:** Instancia do propio contrato. Con algunhas das variables mencionadas ata agora, chamaremos ao método *load* do *wrapper* de dito contrato, para cargar así a interface do mesmo e poder traballar cos elementos que o forman.
- **Envío da transacción:** Unha vez con iso, só teremos que mediante o tipo de dato *TransactionsReceipt*, e xestionando o asincronismo (no noso caso facémolas síncronas ao chamar a *get()*), enviar as transaccións desexadas aos métodos do contrato indicado.
- **Uso de tipos específicos:** Para aquelas transaccións ca cadea máis sinxelas e non referentes aos Smart Contracts, *Web3j* brinda unha serie de tipos predefinidos para tratar de forma máis óptima as peticións e respostas á Blockchain (balances, contas, etcétera), por exemplo, o *EthGetBalance*.

CoronaFaucet: Transaccións gratuítas ou con **gasPrice de 0**

A maiores destas transaccións, vai haber un tipo extra: as gratuítas ou con **gasPrice de cero**. Serán implementadas para conseguir Ether cando a *wallet* do usuario estea baleira, facendo uso do noso Faucet. Para elo, o *modus operandi* será idéntico, cambiando unha serie de parámetros como son a dirección de contrato e o *wrapper* a utilizar, e engadindo o punto crítico de indicar no *gasProvider* que dita transacción a enviar terá un **gasPrice de 0**, e non así o **gasLimit**, o cal deberá ser obrigatoriamente maior que cero xa que seguirá a consumir gas, só que dito gas non terá valor, será 0, polo que o custo total da transacción tamén. Esta foi unha das tarefas máis complexas e custosas de deseñar e implementar, polo que cabe mencionar a axuda atopada en [26] e [65]. Parte do código final será da forma de:

```
1 val gasProvider = StaticGasProvider(Convert.toWei("0", Convert.Unit.WEI).toBigInteger(),
   BigInteger("3000000"))
2 val transactionReceipt0: TransactionReceipt? = faucet.giveMeEther().sendAsync().get()
```

Con isto, quedan vistos todos os conceptos importantes sobre a lóxica implementada no proxecto. Unha vez a temos, as tarefas de implementación restantes non serán máis que variantes do amosado ou ben sinxelo código que faga uso do mesmo. Quedará por tanto centrarse nos *layouts* ou interface gráfica, e comezar probar dende as aplicacións móbiles as diversas funcionalidades implementadas. Ambas cousas, vémolos na seguinte sección.

6.2 Probas

Imos agora ca parte referente ás probas realizadas sobre a nosa aplicación móbil. Hai que mencionar que os tipos de tests realizados serán os chamados **Tests de Aceptación**, os cales consisten en que ou ben os desenvolvedores ou os usuarios, comproben que a aplicación cumpre cas especificacións realizadas na fase de Análise. No noso caso, isto levouse a cabo co despregue da app en diferentes dispositivos móbiles físicos e emuladores, realizando así

tamén uns **Tests de Compatibilidade**, para dende eles, facer diferentes fluxos de execución na aplicación e ver que o comportamento é o esperado. No que respecta a isto, outra división das nosas probas pode ser en tanto de **Caixa Negra**, referidas á simple comprobación de que o resultado dunha funcionalidade é o esperado tras unha entrada coñecida, como de **Caixa Branca**, onde a maiores, se foi analizando o código e *debuggeando* as diferentes tarefas para ver que o funcionamento interno do implementado foi o esperado. Debido á súa extensión non podemos ver todas estas probas, así e todo imos mostrar capturas e explicar algunhas delas, as máis esenciais, tanto dende unha perspectiva orientada ás de caixa negra, amosando as diferentes **pantallas** da app referentes á chamada de certas funcionalidades, como de tipo caixa branca, amosando **logs** internos tanto da app como da Blockchain. Seguindo o procedemento que estamos a levar na memoria, todas aquelas probas non detalladas, ou ben non acompañadas de capturas, poderán atoparse integramente no Apéndice F correspondente.

6.2.1 Fluxo de execución A: Primeiros pasos e App CDC

Xestión de Usuarios e Faucet

As primeiras funcionalidades a probar serán a **xestión de usuarios** e o noso **Faucet**. Para isto, imos **crear unha nova conta**, **verificar a mesma via email**, e **conseguir Ether** usando o noso Faucet. Na Figura 6.1a, vemos como creamos un novo usuario na nosa app introducindo email e contrasinal. Amósase na Figura 6.1a a data e hora na que se realizou dita acción para dar veracidade ás seguintes probas, xa que no Apéndice F podemos confirmar na Figura F.2 que na consola Firebase aparece o novo usuario creado ca mesma data. Con isto, teremos a nosa conta lista para empezar usar a app móbil, polo que comezarán aparecer os **Welcome Sliders** que comentamos na sección de deseño 3.3.2. Imos detallar o segundo deles, o cal servirá para probar unhas cantas funcionalidades como: a **verificación** da conta da app *via email*, a **creación** exitosa dunha *wallet* na Blockchain e o correcto funcionamento do noso **Faucet** personalizado, probando así xa as primeiras funcionalidades referentes á Blockchain e as primeiras interaccións cun Smart Contract (de tipo gratuito, como explicamos na anterior sección). Importante **diferenciar entre as contas da aplicación e da Blockchain**, xa que a primeira delas é creada con Firebase e servirá para poder usar a aplicación, e a segunda mediante interaccións ca nosa cadea privada para poder operar na mesma.

Na Figura 6.1b vemos dito *Welcome Slider*. Nel, teremos que centrarnos no botón **Dame Saldo**, o cal outorgará esas 100 unidades de Ether gratis á conta que o solicite facendo a chamada á Blockchain para comezar traballar en dita cadea, iso si, sempre que dita conta se confirmara previamente *via email*. Para facer isto, *clickaremos* na frase onde o indica para que se nos envíe un email de confirmación, o cal conterá un *link* no que pincharemos para verificar a nosa conta. Dita funcionalidade, será a aportada por Firebase por defecto, polo que se pode

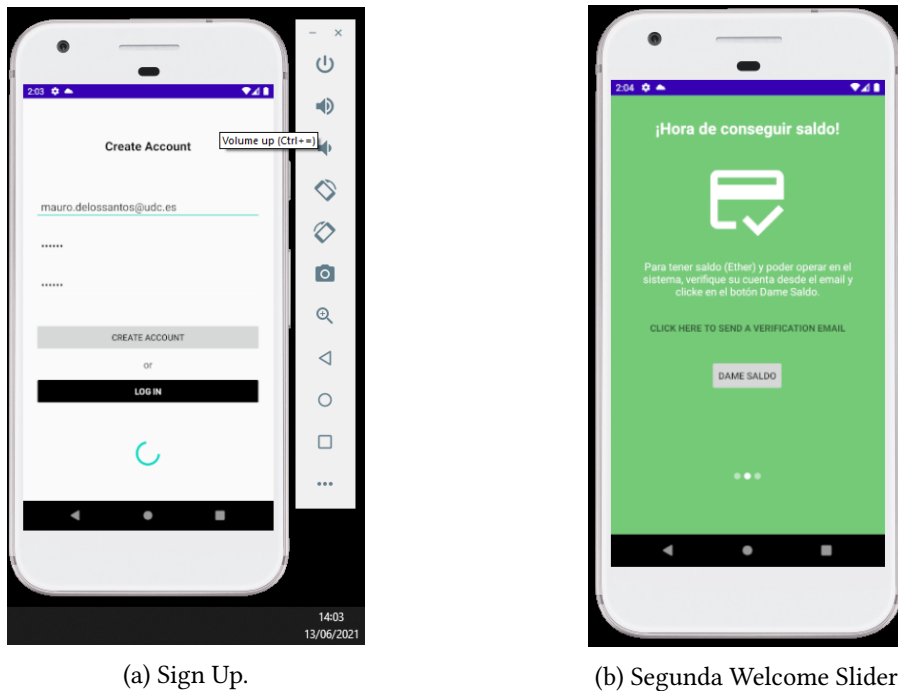


Figura 6.1: Primeiras pantallas da app: Sign Up e Welcome Slider 2.

consultar en detalle na súa documentación [49]. Unha vez confirmada, poderemos corroborar que todo foi ben volvendo solicitar Ether e analizando os *logs* da app e da Blockchain, probando de paso tamén o posible *bug* de pedir veces ilimitadas Ether.

```
Created new Wallet!
address: 0x02ec282320de1e339dc2831f4c6efef1bde8c450
private key: 737030895998196632192222195460134263616412
public key: 1004329930084699955736162323251441068096993
Balance of Ether: 0
```

Figura 6.2: Logcat ca creación da nova conta na Blockchain.

```
Getting Ether for account: 0x02ec282320de1e339dc2831f4c6efef1bde8c450
Ether transferred to newly created account: 0x02ec282320de1e339dc2831f4c6efef1bde8c450
Successful transaction -> Hash: 0x2c982eb44574d105f718f50ae77f10267c3fcd1a7787db8006433b6698e1f913. Block number: 6644. Gas used: 28689.
```

Figura 6.3: Logcat ca chamada ao Faucet.

Nos anteriores logs, obtidos do Logcat de Android Studio durante a execución da app e do log da nosa Blockchain (probas de caixa branca), vemos na Figura 6.2 como se creou satisfactoriamente a nosa *wallet* na Blockchain, imprimindo a súa dirección e par de claves, as cales aparecen cortadas pero que foron ofrecidas ao usuario no primeiro *Welcome Slider* para que as gardara. Na Figura 6.3, corroboramos que o Ether foi transferido correctamente

contando ademais ca información referente á propia transacción na Blockchain, a cal podemos verificar a maiores visualizando os logs da propia cadea como facemos na Figura F.3 no Apéndice F, onde tamén observamos a dirección do noso Smart Contract referente ao Faucet (*recipient*). Antes de seguir cas seguintes probas reproducíronse na app as seguintes accións: o usuario recentemente creado encheu os seus **datos persoais** na app paciente e **xerou o seu QR**, así como tamén acudiu a un centro autorizado para que os seus datos foran **engadidos ao sistema**.

Lectura de pasaporte e engadir dose

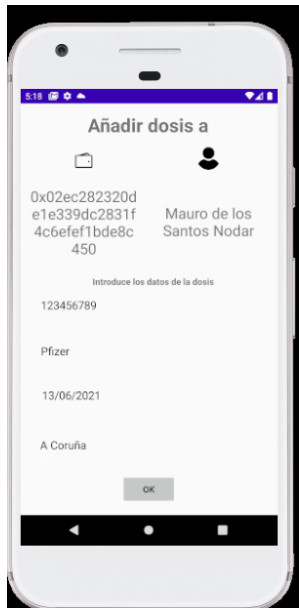
Unha vez feito isto, imos ver un **fluxo completo da app versión CDC**, o cal consistirá en: **ler o pasaporte** do usuario recentemente engadido, para posteriormente engadir ao seu rexistro médico unha **dose**. Decidimos probar este caso de uso porque, primeiro, os procesos de engadir unha proba ou unha reacción adversa serán practicamente idénticos, polo que ca proba dun dos tres casos será suficiente. E segundo, porque durante o proceso no que se levan a cabo ditas tarefas imos ver outras funcións como as referentes ás principais pantallas do CDC: o seu *Home* ou a confirmación das escrituras na Blockchain. Como veremos posteriormente na Figura 6.6b, as contas autorizadas contarán ca funcionalidade principal de escanear un pasaporte. Ao seleccionala, abrírase a cámara co lector QR para así acceder á información de dito paciente, o cal se xa foi engadido ao sistema, sería mostrado como na Figura 6.6a. Nella, clickaremos no botón para engadir unha nova dose para introducir os datos necesarios, os cales posteriormente corroboraremos antes de escribilos na cadea de bloques. Estas accións, as cales recordamos, estarán realizadas dende a app do CDC ao cal acude o paciente do que acabamos de crear a conta, amósanse nas Figuras 6.5a e 6.5b.

```
INFO [06-13|17:19:33.645] Submitted transaction hash=0xbcb17565f7f49c0cc031
0ff410fdb06e16b09d45954e6a5672fc8cebe3aebb9 from=0x02EC282320dE1e339Dc2831F4C6EFf1bDE8c450 n
once=3 recipient=0x0c8E392EF799F3EADBAe7B4d780716533ad03347 value=0
```

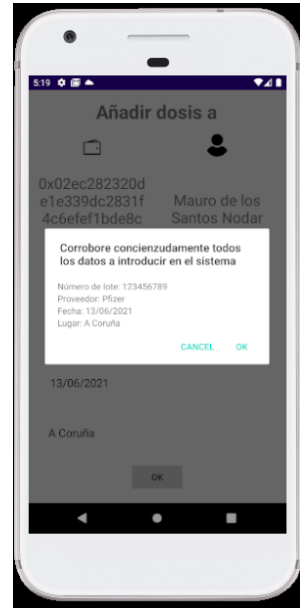
Figura 6.4: Log da Blockchain ca transacción a cal engade a dose.

Tras confirmar os datos, a transacción será enviada e escrita correctamente na Blockchain (Figura 6.4), e a app do CDC volverá ao *Home* acompañando de verde (ou vermello en caso contrario) a información da transacción na Blockchain como vemos na Figura 6.6b.

A maiores da información exitosa da transacción, a cal, de novo, podemos corroborar no log da Blockchain vendo que dita transacción foi enviada, minada e procesada de forma correcta, como vemos na Figura 6.4, vemos como dita pantalla inicial do CDC da cal falabamos antes conta cas opcións de: **Perfil**, para almacenar a súa información máis básica como especificabamos cos obxectos CDC no deseño 3.2.2, **Escáner**, a cal servirá para co lector QR ler os pasaportes pertinentes, e **CDC Info**, funcionalidade extra, a cal xorde ao longo da realización do proxecto, e a cal servirá para consultar a información detallada de contas CDC.



(a) Engadindo Dose dende CDC.

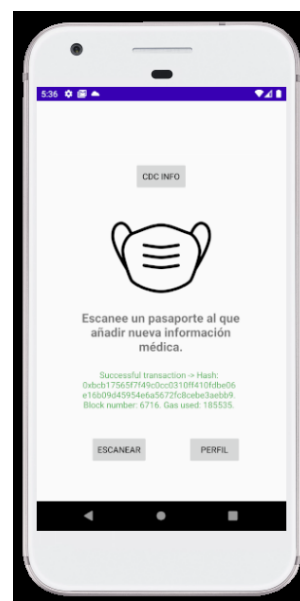


(b) Confirmando a info. da dose a engadir.

Figura 6.5: Engadir dose a un paciente dende app CDC.



(a) Lectura dun pasaporte dende CDC.



(b) Home CDC tras engadir dose.

Figura 6.6: Lectura do pasaporte do usuario creado, e Home CDC tras modificar o rexistro médico.

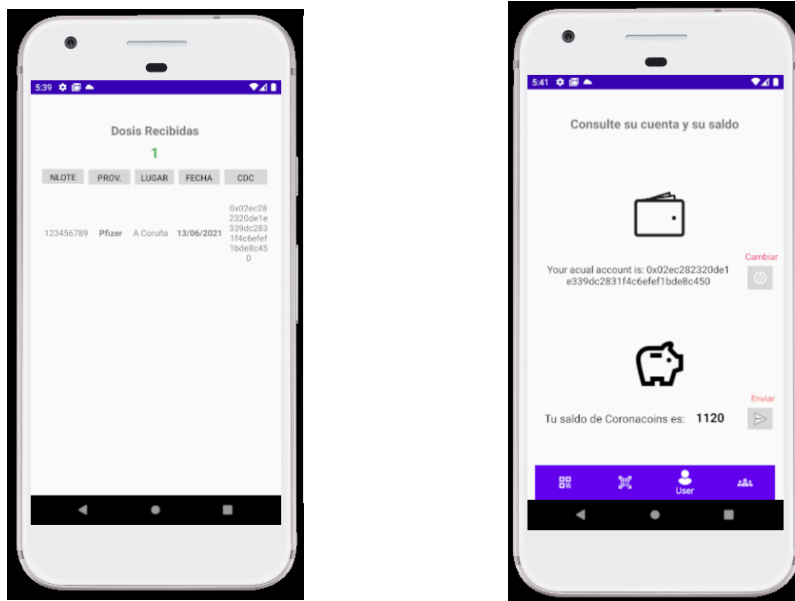
6.2.2 Fluxo de execución B: App paciente e Funcionalidades Extra

Agora, para finalizar as probas, imos realizar outro **fluxo de execución**, neste caso **ca versión da app paciente**, co cal testaremos de novo o **escáner** dun QR, o cal conterá a información do pasaporte do paciente ao cal lle acabamos de engadir a dose e a cal ademais veremos en **detalle**, e por outro lado, probaremos tamén as funcionalidades referentes á criptomoeda, ao primeiro, comprobar o noso **saldo** para corroborar que recibimos os incentivos acordados ás respectivas lecturas e escrituras feitas, así como segundo, **comerciar** ca mesma enviando unha porción dela a outra conta da Blockchain. Imos velo por orde.

Doses en detalle e criptomoeda

Na Figura 6.10 vemos a pantalla principal do paciente (tamén o facemos no Apéndice F na Figura F.5), é dicir, volvemos á conta a cal creamos ao principio. Nela, teremos un QR coa nosa dirección de *wallet* e cos nosos datos persoais que como dixemos antes, engadiriámoslos manualmente antes de ser introducidos no sistema clickando no **botón flotante** situado na parte inferior dereita. Por outro lado, teremos tamén esa **mensaxe esclarecedora** da que falabamos na sección de deseño 3.2, a cal nos dirá o estado do noso pasaporte en cada momento, indicándonos se non estamos logueados en ningunha *wallet* (é dicir falta a chave privada da mesma), se a información da mesma consta ou non no sistema (está escrita na Blockchain, como é o caso despois das probas anteriores) ou se nos falta algún dato persoal por introducir no noso código QR (ca funcionalidade do botón flotante). Imos agora, ca funcionalidade de escanear un pasaporte, dispoñible no segundo botón da *Bottom Navigation Bar*, ler dende outra conta dito QR, simulando a **lectura do pasaporte** do paciente, e, tras como no caso anterior, utilizar a cámara do dispositivo a modo de escáner QR e ver a mesma pantalla que amosamos na Figura 6.6a ca información do pasaporte (exceptuando que agora non aparecerán os botóns de engadir doses, probas ou reaccións), veremos como aparece unha dose engadida, polo que clickando nela poderemos vela en detalle, accedendo á pantalla da Figura 6.7a.

Vemos polo tanto, como as anteriores escrituras e a actual lectura do pasaporte, así como a visualización en detalle dos datos médicos referentes ás doses (para as probas e reaccións, como dixemos antes, será idéntico), funciona da forma esperada. Polo que, imos agora ver a parte do paciente que fai referencia á nosa *wallet* e á nosa criptomoeda. A pantalla inicial, á cal se chega clickando no terceiro botón da *Bottom Navigation Bar*, ca icona dun usuario, vémosla na Figura 6.7b. Nesta Figura, temos a nosa dirección de *wallet* ca que estamos logueados neste momento, podendo cambiala introducindo unha nova chave privada, como vemos na Figura 6.8b. Tamén temos o noso saldo actual de criptomoeda, o cal cobra sentido ao termos **1000 unidades** por ser engadidos ao rexistro, **100 unidades.** por rexistrar a inoculación dunha dose, e **20 unidades** (2 x 10) polas dúas lecturas de dito pasaporte efectuadas. Vemos polo



(a) Doses en detalle dun paciente.

(b) Perfil cos datos referentes á criptomoneda.

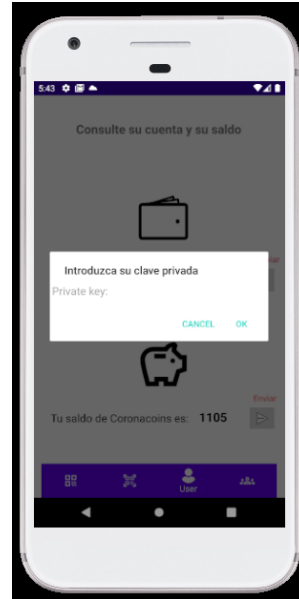
Figura 6.7: Detalles das doses recibidas, da wallet actual e do saldo de criptomoneda.

tanto que a parte dos incentivos referente á criptodivisa funciona correctamente, polo que para verificar todas as súas funcionalidades imos comerciar con ela, enviando **15 unidades á conta 0x17b...**, para despois cambiar de *wallet* testeando dita funcionalidade e corroborar que se recibiron. Vemos o proceso na Figura 6.8.

Ao confirmar a transacción clickando en *OK*, enviaremos dita cantidade de criptomoneda á *wallet* indicada. Para verificar dita proba, podemos ou acceder aos *logs* tanto de Logcat (Figura 6.9) como da Blockchain (ca Figura F.4 do Apéndice F), para verificar a transacción, como ver o saldo actualizado na propia pantalla, como se aprecia no fondo da Figura 6.8b, onde pasou de **1120 a 1105**, sendo restadas as 15 unidades enviadas. No referente aos logs, obtemos a información da conta á cal llo enviamos na Figura 6.9, onde vemos que a cifra da criptodivisa acaba en 5, e isto só pode ser pola implementación correcta da funcionalidade de *trading* (xa que os incentivos só son múltiplos de 10). Por outro lado, vemos tamén un concepto moi interesante a mencionar, que é a cantidade restante de Ether da conta, que tras facer todas as transaccións amosadas neste capítulo, aínda segue a ser **99,99 unidades** de Ether, o que nos serve para probar dende a correcta configuración da cadea ao demostrar a **baixa dificultade de minado** e en consecuencia o seu **baixo custo de transaccións** e **alta velocidade de resolución** das mesmas, ata a **correcta decisión de outorgar 100 unidades de Ether** co noso Faucet, ao seren estes suficientes fondos para facer milleiros de transaccións.



(a) Comerciendo ca criptomoeda.



(b) Cambiando de Wallet na app.

Figura 6.8: Probando as funcionalidades referentes á criptomoeda.

```
Getting Balances of account 0x17b25a5d178aa4e40bd987d39ec982f78ad53c27
Balance of Ether: 99994221219262160000
Balance of Coronacoins: 1165
```

Figura 6.9: Logcat ca información referente á transacción de coins.

Conclusión de probas

Chegados até aquí, probáronse **todas** as funcionalidades do sistema, as primeiras dende **Remix** e directamente contra a Blockchain, e as restantes dende a **App**. Amosáronse tamén todos os tipos de interaccións ca Blockchain, así como **transaccións** a ambos Smart Contracts, á vez que mostrou gran parte do **aspecto** da app para corroborar que se seguiu o deseño dos *mockups* do apartado 3.3. Aínda así, hai unha serie de funcionalidades, as cales, por límites de extensión desta memoria, non poden ser amosadas en detalle, pero que foron implementadas correctamente e as cales lle dan un gran potencial ao sistema. Ditas tarefas, consideradas como secundarias, serán accedidas dende as opcións no **menú superior dereito** da app versión pacientes, e podémolas ver na Figura 6.10 para comentalas brevemente.

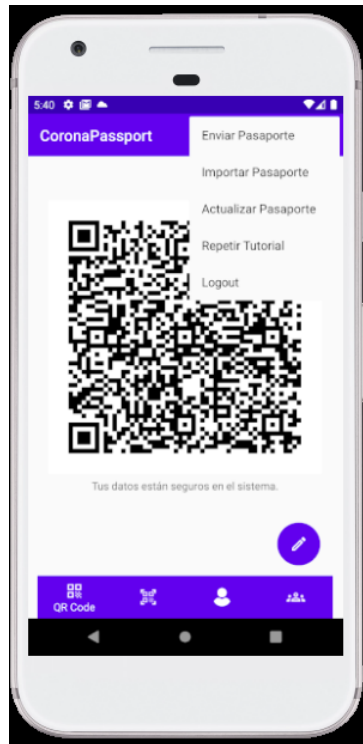


Figura 6.10: Funcionalidades extra non detalladas.

Vemos como cada usuario poderá realizar funcións básicas que van dende **desloguearse** da súa conta da app ou **repetir o tutorial** inicial cos *Welcome Sliders* (por se necesitara máis Ether ou volver gardar a súa chave privada), até **actualizar a información do seu QR** (*Actualizar Pasaporte*) para ver o estado do seu pasaporte. Dende este menú despregable tamén se poderá acceder a unha das funcionalidades máis potentes da aplicación, o **envío e recepción do pasaporte cifrado** por un algoritmo de criptografía simétrica (*AES*) cunha chave pre-acordada, a cal é introducida tanto por orixe como por destino na propia app. Dita funcionalidade brinda ao sistema dun maior potencial, ao por exemplo poder demostrar inmunidade a distancia e ademais, facéndoo dun xeito secreto e seguro. Non se mostra a súa proba en detalle porque sería dunha gran extensión, xa que é unha funcionalidade longa e complexa, pero precisamente por ditos motivos cabía mencionala e ensinar o acceso a ela.

Queda polo tanto finalizado en primeiro lugar, o **capítulo referente á Dapp**, ao amosar os seus detalles de implementación e probas, como en segunda instancia, o **ciclo de desenvolvemento software incremental**, ao rematar con todas as tarefas referentes ás probas da implementación, tanto da app como da propia Blockchain. Chégase entón así a unha primeira versión final do proxecto, polo que agora quedará facer unha recapitulación do mesmo para despois sacar conclusións. Abordamos dito traballo no seguinte e último capítulo da memoria.

Conclusións, Incidencias e Traballo Futuro

Este capítulo fai un resumo do traballo feito así como unha comparación cos obxectivos marcados no momento inicial do proxecto, para sacar así conclusións e repasar as leccións aprendidas. Por outra banda, enumeraranse as incidencias atopadas durante a realización do sistema acompañadas dunha breve explicación da súa solución, así como as diferentes tarefas a realizar para levar dito traballo a un entorno real de produción.

7.1 Conclusións

7.1.1 Cumprimento dos obxectivos principais

Imos entón enumerar os obxectivos iniciais acompañándoos da solución implementada que os satisfizo, para concluír en primeiro lugar que o resultado final cumpre ca totalidade de obxectivos marcados inicialmente, ao ter conseguido:

- **Crear un sistema descentralizado con intelixencia automatizable** para incorporar todos os datos médicos referentes á [COVID-19](#) asociados de forma unívoca ao cidadán. Dito obxectivo foi conseguido a través da **creación da nosa Blockchain privada**, rede a cal será a infraestrutura principal do proxecto e na cal será almacenada dita información. Ao implementar esta solución, creamos pois un sistema distribuído, intelixente e seguro, necesario para o tratamento adecuado dos datos a engadir no sistema así como pilar fundamental da infraestrutura do noso modelo de pasaporte dixital [COVID-19](#). Por outro lado, dita solución tamén arranxa os problemas detectados no estado da arte e expostos na xustificación do proxecto, como poden ser a precariedade tecnolóxica do sistema sanitario ou mesmo a falta dun sistema unificado para o tratamento de todos os datos relacionados ca [COVID-19](#).

- **Crear unha criptomoeda propia** grazas ao **Token ERC-20 sobre a nosa Blockchain privada**. Ter elaborado dita criptodivisa permitiunos, a parte de adentrarnos na temática das criptomoedas a fondo, a cal cada vez ten un maior peso e importancia na sociedade actual e futura, tamén crear unha ferramenta a cal puido outorgar funcionalidades ao noso sistema que van dende o **comercio** con ela, de igual maneira que con calquera outra criptodivisa no mercado, podendo enviala e recibila sen limitacións, até poder utilizala a modo de **incentivo** económico de cara ao cidadán presente no noso sistema de pasaporte **COVID-19**, punto crítico no proxecto e o cal foi cumprido grazas a implementación desta solución. Con dita criptomoeda poderemos fomentar o emprego do sistema creado e beneficiar ás diferentes partes involucradas no seu uso, así como dotar dunha aínda maior innovación e orixinalidade ao proxecto proposto.
- **Crear unha app móbil, descentralizada e multiplataforma** grazas á **elaboración da nosa Dapp en KMM**. Dita aplicación, que en realidade como ben se explicou foi posteriormente dividida en dúas versións, a parte de facilitar un doado despregue tanto en Android como en iOS grazas ao seu carácter multiplataforma, actuará como principal medio para a comunicación do usuario ca nosa Blockchain, outorgándolle así tamén á propia aplicación un carácter descentralizado e distribuído, á vez que unha escalabilidade e usabilidade enorme ao sistema proposto, achegándoa ao peto da maioría da poboación. Por outro lado, será a través dela onde o usuario poida xa non só explotar as funcionalidades e requisitos detectados e satisfeitos no traballo, se non ver melloradas certas situacións cotiás actuais xeradas por mor da pandemia, que con dita aplicación móbil poderían verse eliminadas, como poden ser parte das restricións sociais derivadas da **COVID-19**.

7.1.2 Cumprimento dos requisitos técnicos

En canto ás funcionalidades máis técnicas e concretas, foron todas satisfeitas na súa totalidade grazas ao resultado da consecución destes tres obxectivos. E é que mediante unha das dúas **Dapps** desenvoltas, a versión para os pacientes, poderemos **ler o pasaporte** dixital ca información médica **COVID-19** despregado sobre a Blockchain, así como **comerciar ca nosa criptomoeda** recibida a modo de incentivo. Por outro lado, poderemos tanto **amosar o noso pasaporte** mediante un código QR ca información para facer as consultas axeitadas á Blockchain, como tamén **enviar cifrado dito QR** a aquela **Dapp** da persoa ou entidade ca cal queiramos compartir a nosa información. En canto á outra versión da **Dapp**, aquela dispoñible para as autoridades, poderemos **escribir na cadea de bloques** tanto a información de novos pacientes, **engadindo** seu rexistro médico ao sistema, como complementar dita información actualizándoa con **doses, probas ou reaccións adversas**. Todos estes datos, así como as mencionadas escrituras, terán sempre un carácter seguro ao ser información verificada por

profesionais, corroborada polos mesmos e escritas só dende contas autorizadas e controladas individualmente, as cales terán á súa vez un control e auditoría constante feitos tanto polo equipo do proxecto como polas autoridades pertinentes.

7.1.3 Cumprimento doutros obxectivos

A parte do cumprimento tanto dos obxectivos como das funcionalidades, os fundamentos nos que se basea o sistema foron todos eles acadados con éxito ao, como ben expuxemos antes: crear un **sistema para almacenar a información** ca nosa Blockchain e unha **app que faga uso da mesma**, cas nosas **Dapp** multiplataforma. A maiores disto, foi posible tamén a creación da funcionalidade de **Faucet personalizado** ao crear un sistema propio de outorgamento de Ether baixo demanda, para que así as contas que o soliciten, poidan facer uso de todas as funcionalidades expostas e mellorar considerablemente a utilidade do proxecto.

7.1.4 Cumprimento dos requisitos de TFG de dobre mención

Unha vez finalizado o traballo, podemos observar como de igual forma que anticipabamos no informe xustificativo de dobre mención, o proxecto cubriu amplamente os requisitos dun *Traballo de Final de Grao* deste tipo, neste caso concreto, das especializacións en **Tecnoloxías da Información (TI)** e **Enxeñaría de Computadores (EC)**.

Como ben se especificaba en dito informe, o proxecto xa non só abordou **conceptos técnicos de ambas ramas**, senón que tamén absorbeu o **carácter e pensamento crítico** aportado polas mesmas. O traballo tentou anticiparse aos cambios de dirección das **TI**, implementando unha solución novidosa a un problema sen arranxar o cal está á orde do día. Á súa vez, o proxecto nace ca finalidade de mellorar a vida xa non só do usuario, senón da cidadanía en xeral, adaptándose ás novas necesidades sociais e empregando tecnoloxías modernas e disruptivas para crear dita solución. Todos estes obxectivos propios xa non só de ambas mencións, senón propósitos xerais de calquera **enxeñaría**. Por outro lado, botando unha visual xenérica aos conceptos técnicos, afondou profundamente en materias de elevada complexidade, a maioría delas nin sequera vistas no grao, como poden ser **Blockchain** ou o **Desenvolvemento Móbil Multiplataforma**, todas elas baixo un mesmo fío condutor como foi a **Ciberseguridade**, campo de estudo transversal a ambas ramas e polo cal se decidiu facer dita dobre mención.

Polo que tras realizar o proxecto, confirmamos a utilización daqueles **conceptos técnicos** enumerados no informe xustificativo da dobre mención e que facían referencia a ambas mencións: exemplos disto son o desenvolvemento xa non dunha, senón de dúas aplicacións móbiles usando **KMM** e introducíndose así no desenvolvemento multiplataforma e en conceptos que van máis aló do amosado xa non só na **mención de EC**, senón no grao en xeral. Por outro lado, grazas á extensa parte de Blockchain (**mención de TI**) e os seus múltiples compoñentes, seguíronse abordando conceptos inherentes a **EC**, como son a programación

eficiente, a inmutabilidade do código e o prezo das escrituras, á vez que se profundizou como base en conceptos de **TI** como a criptografía, a seguridade da información, os despregues e a administración de sistemas ou redes (Blockchain), entre outros moitos. Por riba, todos estes conceptos, aínda que divididos de forma xeral nestas dúas grandes partes, cada unha máis relacionada con cada mención, nunca estiveron illados uns dos outros, se non que pola contra, intentáronse **combinar** para crear un sistema máis potente e máis completo, grazas á **mestura das competencias** obtidas de ambas ramas ca finalidade de crear unha solución única e homoxénea.

Concluímos entón, que o proxecto cubriu amplamente as expectativas que se poden esperar dun traballo deste tipo, xa non só pola **dobre carga de traballo e horas** invertidas, senón tamén pola parte de **investigación, aplicación e ampliación** de conceptos referentes a ambas mencións, ca finalidade de crear un sistema novidoso e disruptivo que os combine para en conxunto aportar unha **solución efectiva para un problema actual**.

7.1.5 Conclusión final

Polo que para concluír, vemos como ao rematar dito proxecto conseguíronse **cumprir todos os requisitos e obxectivos propostos** nun primeiro momento, tanto principais (cos obxectivos), técnicos (cas funcionalidades) e teóricos (cos fundamentos). Conseguiuse tamén elaborar un novo e innovador sistema, totalmente completo e funcional, que potencialmente pode ser a base dun novo tratamento e xestión da información referente á **COVID-19**, elaborando así unha ferramenta, que cos recursos e apoios necesarios, podería axudar a cambiar e mellorar gran parte das situacións actuais derivadas da pandemia. Polo que realizando este traballo de final de grao, xa non só se creou unha **complexa solución tecnolóxica** que aborda e profundiza en diferentes campos da informática (como poden ser a ciberseguridade, o desenvolvemento móbil ou a Blockchain), se non que tamén se propuxo unha **solución de carácter xeral** a múltiples problemas que están a día de hoxe presentes nas nosas vidas e para os que aínda se están buscando solucións.

7.2 Incidencias

Exponse a continuación unha enumeración que indica os principais problemas atopados durante o desenvolvemento do traballo acompañados dunha breve explicación da súa solución.

- **CoronaFaucet:** Todas as tarefas que xiraron ao redor da implementación desta funcionalidade levaron asociadas unha gran dificultade e unha serie de problemas. Primeiro de todo, porque a información dispoñible referente ás Blockchains privadas xa non é moi extensa, e moito menos combinada ca creación de servizos de retribución de Ether

baixo demanda nelas, estilo Faucet. O habitual, é a utilización dun Faucet xa creado por *Testnets* existentes (e.g., Ropsten [37]) ou a non utilización do mesmo e usar no *Genesis Block* o comando *alloc* para profundar con Ether ás contas que indiquemos. Ambas solucións non son válidas para a aproximación proposta neste TFG, polo que, despois de falar persoalmente co autor do TFM pola Universidade de Basilea en [26], así como colgando no foro máis importante de desenvolvedores Ethereum unha consulta con dita problemática [65], e tras facer un gran traballo de investigación e deseño por conta propia, implementouse dita funcionalidade, a cal foi detallada na respectiva sección de implementación da Blockchain, onde grazas á existencia de nodos que minen transaccións con *gas price* de cero, á configuración adecuada da Blockchain para soportalas, e á correcta elaboración das transaccións para facer uso das mesmas, conseguiuase solucionar o problema implementando un sistema propio de retribución de Ether baixo demanda de forma gratuíta.

- **Non soporte da combinación eventos-lista de tipo complexo:** A aproximación desexada de devolver a información dun pasaporte, é dicir, de ler o mesmo cun só método á vez que devolvera a información e escribira na Blockchain os datos acordes aos incentivos, non foi posible. A ferramenta *Web3j*, ca súa funcionalidade de crear *wrappers* tipo Java a partir dos Smart Contracts de *Solidity*, os cales no permiten facer un adecuado tratamento da información dos eventos dende o código, só soporta aqueles tipos de datos que pertencen á clase *TypeReference*, requisito non cumprido por aquelas coleccións de datos de tipos complexos creados no proxecto, como foi no noso caso, cos datos nos *arrays* de *Dosis*, os cales serán necesarios para poder xestionar os datos devoltos polos eventos de ditas lecturas. A solución adoptada, como xa se explicou na memoria, foi a da división de dito método en dous: nun que se encargara de escribir a información referente aos incentivos na cadea, e noutro de tipo *view* que devolvera a información do pasaporte dispoñible na cadea sen facer uso da información dos eventos, pero si disparándoos igualmente.

7.3 Próximos pasos e liñas futuras de cara a produción

Nesta sección enumeraranse aquelas liñas a seguir nun futuro para mellorar o sistema proposto. Por outro lado, mencionaranse tamén certos cambios que, por limitacións ben de tempo, ou de recursos, non foron posibles de implementar pero que dotarían ao proxecto dun maior valor. A finalidade será facer unha selección daqueles conceptos máis importantes que habería que realizar, ou polo menos ter en conta, para un hipotético despregue nun entorno real de produción do sistema proposto no proxecto.

- **Xestión das conexións:** Habería que mellorar certos aspectos referentes á toma de decisións en canto ás conexións dende a [Dapp](#) móbil até a Blockchain. Actualmente está modelado de forma que mediante un *socket* cunha *Http connection* á dirección IP estática do nodo, conseguimos conectarnos ao mesmo para comezar facer as transaccións. Unha aproximación mellor, máis escalable e que seguira os principios das [DLT](#), sería que dita conexión se probara a facer de forma iterativa e aleatoria sobre un listado de direccións IPs de nodos anotadas nun ficheiro de texto, editable polo usuario, para así engadir os nodos que considerara necesarios. Desta maneira, a conexión sería distribuída, xa que recaería sobre unha lista editable (poderíanse engadir novos nodos) de *endpoints* públicos, para ir probando aleatoriamente a conexión con eles. Por outro lado, quizais habería que buscar aproximacións máis realistas e seguras para facer esas conexións en vez de cunha conexión *Http*, usando algún protocolo con maior seguridade como *Https*, ou algún outro tipo de *socket*, por exemplo. O motivo da non implementación de ditos requisitos foi, en primeiro lugar, porque para facelo necesitaríamos ter varios nodos correndo despregados e minando, o que requiriría unha gran potencia computacional, a cal non se tivo ao realizar o TFG, así como que o tipo de conexión, aínda que non sexa a máis segura, é totalmente válida e funcional e máis para un caso xenérico como o exposto. Por último, a non óptima xestión de ditas conexións pode dar lugar a certo *freezing* da app cas conexións máis duradeiras cara a Blockchain, debido ao tratamento non asíncrono das conexións ca mesma. Poderíase solucionar ca xestión de procesos en *background* ou ca xestión de peticións asíncronas. De novo, por extensión e limitación temporal, a maiores de que non é un problema, xa que funciona correctamente, senón unha optimización, optouse por deixalo desta maneira.
- **Parte de Apple:** Para sacarlle o máximo partido ás ferramentas de desenvolvemento multiplataforma habería que implementar a parte específica referente ao sistema operativo da mazá. Para elo, teríamos que desenvolver as interfaces gráficas de dito [SO](#), así como aquelas clases que requirisen una interacción con elementos nativos do sistema, coma por exemplo a cámara co lector QR, a importación e envío do ficheiro co QR cifrado ou o uso de librerías externas non compatibles de igual maneira con ambos sistemas. Por outro lado, habería que prestar unha maior atención á organización do código do proxecto [KMM](#), facendo unha análise moito máis concreta da colocación de cada clase, implementando a compartición do mesmo por ambas partes co uso do módulo *shared* e espremendo as funcionalidades de [KMM](#) referentes á optimización do código híbrido. O motivo principal para non facelo, ademais das limitacións de tempo e de que non son cambios que afecten á lóxica principal do sistema, a cal por riba será común a ambos sistemas, foi a necesidade de ter un ordenador macOS para poder compilar e executar o código, requisito que non se satisfizo debido ás limitacións económicas.

- **Mellorar o tratamento das datas:** Debido a que en *Solidity* non hai tipos referentes a datas, decidiuse xestionalas mediante *strings*, pero quizais nun entorno real onde a usabilidade e a estética é importante, podería ser unha aproximación máis acertada representalas con algún tipo específico ou formato estándar a cumprir en toda a *Dapp*. Isto, podería estar acompañado de funcionalidades gráficas para a elección das datas dende a app, con, por exemplo, calendarios para escoller as datas nos casos de uso que sexan necesarios (inoculación de dose ou data de nacemento).
- **Posibilidade de gardar a KPriv en firebase de forma online:** Sería interesante ademais de ofrecerlle ao usuario a posibilidade de que garde pola súa conta a chave privada da súa *wallet*, deseñar un sistema que lle axude a isto para, a parte de almacenala cifrada na app (por exemplo, gardala de forma segura e online xunto coas súas credenciais de usuario cun mecanismo de persistencia de Firebase). Xa que moitos *frameworks* existentes hoxe en día, como *Binance* [66], nin sequera o teñen implementado, e debido a que ofertamos unha solución alternativa almacenándoa cifrada na propia app para que a poida volver consultar de xeito ilimitado, decidiuse non facelo, aínda que sería un campo interesante de estudo.
- **Auditorías aos procesos referentes ás apps das autoridades:** Habería que ter un potente sistema de control de cara ao outorgamento de ditas apps, xa que ao seren as únicas que poden escribir na Blockchain, o seu adecuado manexo é crítico para o correcto funcionamento de todo o sistema. Será de vital importancia ter unha ou varias entidades externas, en conxunto co equipo da *Dapp*, controlando e supervisando en todo momento as decisións sobre o outorgamento de ditas contas. Nun entorno de produción sería habitual que varias autoridades externas como a policía, algunha autoridade sanitaria e o propio equipo da *Dapp*, revisaran individualmente ditas accións para ter unha maior barreira fronte posibles usos fraudulentos da versión da aplicación con privilexios. Un exemplo sería tratalas da mesma forma que se tratan aquelas contas de traballadores en sectores críticos e con accesos privilexiados como sistemas tributarios, policiais, etcétera.

A maiores destas liñas xenéricas (e algo máis específicas, como vimos nos últimos puntos), existen un gran número de posibles cambios mínimos a realizar que mellorarían o sistema, pero debido a que ningún deles goza dun carácter crítico, así como ningún deles afecta tampouco ás funcionalidades, senón que tratan conceptos estéticos, de usabilidade ou de optimización, non se comenta ningún a maiores. Con isto vemos que, sen nin sequera verse obrigados a implementar as liñas futuras (aínda que sería aconsellable facelo) e cas incidencias solucionadas como se detallou, conseguiuase un potente sistema, completamente listo para ser despregado sen practicamente cambios nun entorno de produción real.

Apéndices

Planificación e avaliación de custos

A.1 Planificación do proxecto

A continuación amósase a planificación mediante un **Diagrama de Gantt**, o cal consta cas diferentes **fases do proxecto**, así como a súa data de inicio, fin e duración.

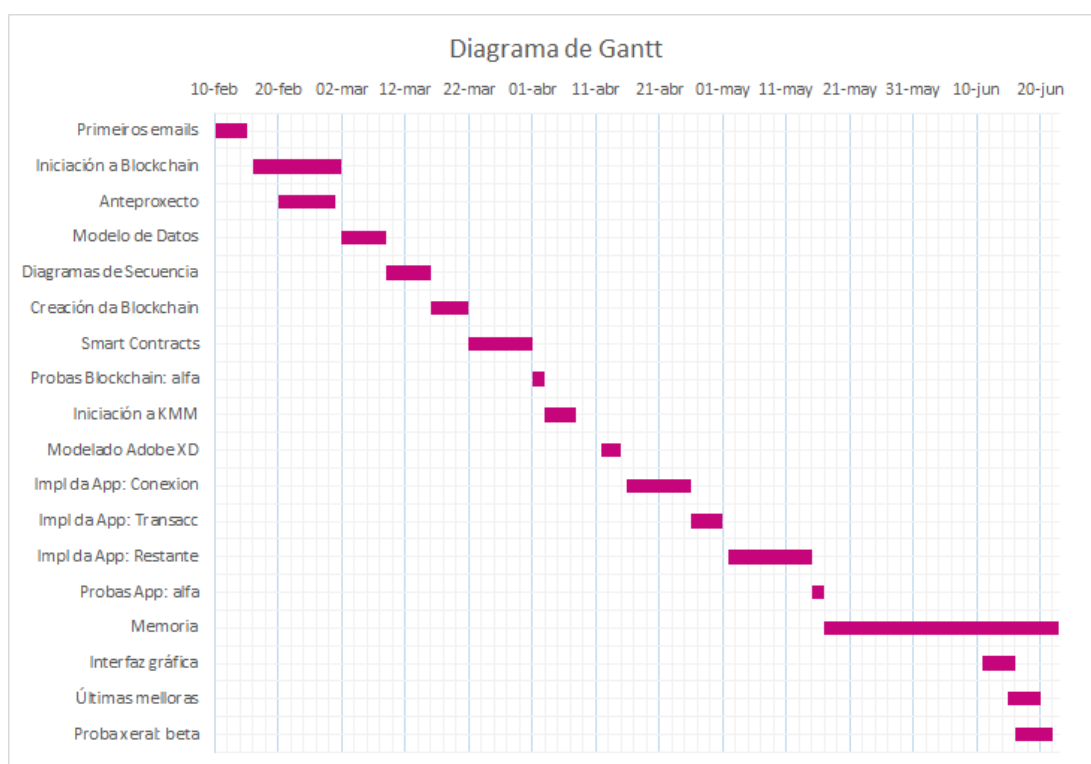


Figura A.1: Diagrama de Gantt do Proxecto.

En canto ás diferentes fases, aclarar de forma breve en que consistiu cada unha:

- **Primeiros emails:** Foron as primeiras mensaxes co titor para falar da idea, acordar a tutorización do TFG e onde me comezou indicar os primeiros traballos e artigos similares a ler para ir familiarizándome ca temática.
- **Iniciación a Blockchain:** Constou dunha parte teórica, onde lin e consultei múltiples fontes para adentrarme a fondo no mundo de Blockchain, en concreto de Ethereum e nas súas posibilidades de programación, e tamén unha parte, realizando un curso *online* gratuito de iniciación á escritura de Smart Contracts en *Solidity* chamado **CryptoZombies** [58]. Ao final, houbo unha parte práctica ao deseñar a infraestrutura do proxecto acorde ao visto.
- **Anteproxecto:** Nesta fase redactáronse os tres documentos necesarios a entregar para que o TFG fora aprobado, cas diferentes iteracións alumno-titor necesarias até chegar ás versións finais e firmadas de todos eles.
- **Modelo de Datos:** Primeira tarefa puramente práctica do TFG a cal constou da enumeración e organización dos datos a almacenar na Blockchain, os seus tipos, relacións, etcétera.
- **Diagramas de secuencia:** Tarefa onde se realizaron os diferentes diagramas de secuencia de todos os casos de uso existentes no proxecto.
- **Creación da Blockchain:** Investigación e desenvolvemento da Blockchain privada para finalizar co seu despregue e proba.
- **Smart Contracts:** Implementación, despregue e proba dos Smart Contracts en *Solidity* na plataforma Remix, para crear tanto as funcionalidades do rexistro médico como da criptomoeda personalizada.
- **Blockchain - alfa:** Esta tarefa finaliza cando se chega á primeira versión estable da parte de Blockchain do traballo onde temos a nosa Blockchain privada despregada e correndo ca configuración inicial correcta e unha serie de Smart Contracts despregados nela, os cales dende Remix foron testeados ca realización de diferentes transaccións a modo de proba.
- **Iniciación a KMM:** Consistiu na familiarización con este *framework*, lendo a documentación e buscando información teórica, así como creando o primeiro proxecto a modo de *Ola Mundo*.
- **Modelado en Adobe XD:** Realización de todos os *mockups* e fluxos entre eles que tería a *Dapp* final para imitar e deseñar o seu aspecto e funcionamento. Esta etapa tamén incluíu a familiarización co *framework* de Adobe XD.

- **Implementación da Dapp - conexión:** Primeira etapa de implementación da Dapp con KMM que consistiu en xestionar a conexión e as lecturas na Blockchain privada.
- **Implementación da Dapp - transaccións:** Mellora da fase anterior ca realización de transaccións dende a Dapp ca finalidade de escribir na Blockchain.
- **Implementación da Dapp - restante:** Depuración das funcionalidades anteriores, mellora da interface gráfica e incorporación de funcionalidades extra. Fase encargada tamén de dotar ao sistema dunha maior seguridade con tarefas como: a persistencia segura de datos ou o envío cifrado do pasaporte.
- **Dapp - alfa:** Esta tarefa finaliza, ao igual ca da parte de Blockchain, cando se dispón dunha Dapp, sen interface gráfica depurada pero con todo o conxunto de funcionalidades necesarias implementadas e probadas nunha primeira instancia.
- **Memoria:** Consiste na redacción da memoria final do TFG.
- **Interfaz gráfica:** Fase dedicada á mellora da estética da app, mellorando os *layouts* de todas as *activities* en xeral, así como do aspecto en conxunto da app móbil.
- **Últimas melloras:** Nesta etapa pulíranse os últimos detalles do traballo, mellorando a estrutura e organización do código, eliminando *boilerplating*, optimizando funcionalidades e corrixindo erros menores cuxa realización da memoria axudou a detectar.
- **Proba xeral - beta:** Última fase de probas onde de forma intensiva se probaron todos os casos de uso dende varios emuladores e dispositivos físicos executando a app de forma simultánea. Será a proba máis similar a un entorno de produción, salvando as limitacións de recursos informáticos (nodos mineiros e dispositivos *Android*), así como tamén será a fase que axude a describir con maior exactitude o capítulo de liñas futuras de cara ao paso a produción. Unha vez finalizada esta fase, estaría lista a segunda versión final funcional da app e do sistema en xeral, a nomeada versión *beta*.

A.2 Avaliación de custos do proxecto

Debido a que non foi necesaria a utilización hardware nin de *royalties* de ningún tipo para a realización do proxecto, o custo do mesmo consistirá na remuneración das horas invertidas na súa elaboración. Aínda que non é exacto, de acordo á táboa escrita a medida que se foi realizando o TFG e na que se indica por cada fase das anteriores a súa duración en días e, asumindo un tempo medio de dedicación por día traballado de 5 horas, obtemos que as horas totais invertidas foron: **735 horas**. A estas horas, haberá que restarlle as de aqueles días que aínda dentro das diferentes fases, non se traballou, os cales ao longo dos catro meses e medio foron 17. Polo que, $17 \cdot 5 = 85$, e $735 - 85 =$ **650 horas** totais invertidas no TFG.

Debido a que non existe ningún convenio que se adecúe de forma exacta ao prezo/hora das tarefas de informática en xeral, fanse os seguintes supostos á hora de calcular o custo total do traballo. Leváronse a cabo tres roles á hora de realizar o traballo, cada un deles cunha serie de horas asociadas e un prezo/hora diferente. Expóñense a continuación:

- **Xefe de proxecto:** Ao ter que planificar a elaboración do mesmo, asignar recursos (neste caso só un), horas por día, tarefas, fitos, etcétera. Estímanse unhas 50 horas das 650 nesta tarefa, polo que asumindo un prezo de 50 €/hora (baseándose nas tendencias actuais das empresas informáticas) teríamos un total de: $50 \text{ horas} \cdot 50 \text{ €/hora} =$ **2.500 €**.
- **Analista - Diseñador:** Ao ter que identificar os requisitos e funcionalidades do sistema, así como deseñar a arquitectura, casos de uso, modelado de datos, etcétera. Estímanse unhas 250 horas invertidas nestas tarefas a asúmese un prezo por hora en torno aos 40 €/hora baseándose de igual maneira que no caso anterior nas tendencias actuais dos traballos informáticos en Galicia. Temos polo tanto que: $250 \text{ horas} \cdot 40 \text{ €/hora} =$ **10.000 €**.
- **Programador:** Ao ter que encargarse da implementación de todos os submódulos (Blockchain, Scripts, Java, Kotlin) así como da súa proba, xunto cos despregues de ambas partes. Estímanse unhas 350 horas dedicadas a estas tarefas, polo que asumindo un prezo xenérico para todas elas duns 30 €/hora de acordo aos mesmos criterios que nos casos anteriores, teríamos que: $350 \text{ horas} \cdot 30 \text{ €/hora} =$ **10.500 €**.

Sumando as tres cifras, obteríamos a modo de aproximación que o custo real para unha empresa á hora de realizar dito proxecto aproximaríase aos **23.000 €**.

Alternativas ao deseño da arquitectura do proxecto

O seguinte apéndice terá a función de comentar as opcións de deseño da infraestrutura do proxecto baralladas en dita fase. Encargarase de enumerar, exemplificar e detallar as mesmas, así como xustificar a elección comentada no capítulo 3 fronte a elas. Expóñense polo tanto nas Figuras B.1 e B.2 ditas alternativas baralladas pero non escollidas.

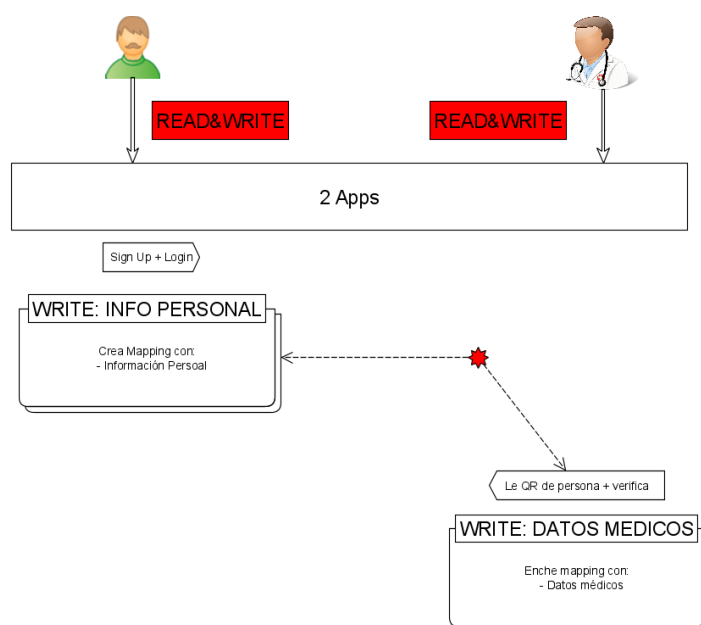


Figura B.1: Modelo 1 non elixido de escritura na Blockchain.

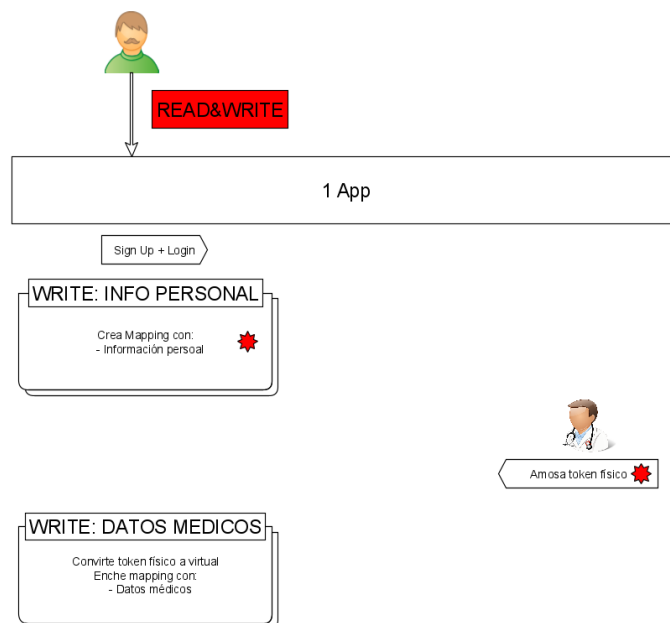


Figura B.2: Modelo 2 non elixido de escritura na Blockchain.

Os problemas polos cales foron descartados os anteriores modelos veñen marcados polos asteriscos vermellos e son:

- No primeiro deles, ilustrado na figura B.1, no cal existen dúas apps diferentes pero ambas teñen permisos de escritura, o problema ven determinado pola primeira escritura do paciente. Aínda que non poida escribir información médica, o paciente sería o encargado da primeira escritura na Blockchain onde se inicializaría o *mapping* da información persoal do paciente, traendo consigo o problema a escritura sen verificación, podendo causar serios problemas se á hora de escribir se producira algún erro tipográfico, por exemplo, nalgún campo crítico como son os identificadores nacionais ou médicos. Un dos beneficios da Blockchain é que é inalterable e inmodificable, o cal neste caso se transformaría no gran inconveniente da solución, xa que o rexistro médico da *wallet* dun paciente quedaría corrupto ao ter información falsa ou errónea, todo debido ao permiso outorgado ao cidadán para escribir na Blockchain sen a verificación dunha segunda parte. É por isto polo que se descartou esta opción.
- No segundo modelo, ilustrado na figura B.2, onde só existiría unha app manexada integramente polo paciente/cidadán, o cal sería o único encargado de escribir na Blockchain, con axuda dunha segunda entidade externa ao tratarse de información médica, modelo moi similar ao existente en EEUU en [13], trae consigo unha serie de problemas. O primeiro deles, é idéntico ao comentado no modelo anterior: as escrituras erróneas por diversos motivos, e a maiores, teríamos a problemática de como modelar o caso

de uso de que a entidade médica, tras vacinar/facer unha proba/diagnosticar unha reacción adversa, tería que dalgunha maneira ensinar un *token* ao cidadán, que dende a app ao recibilo, recoller a información a escribir e a engadir á Blockchain. Isto, a parte dos problemas de estandarización (xa que debería ser exactamente o mesmo *token* e método para todas as autoridades, en calquera lugar onde se use a app), traería tamén problemas de seguridade pola posible creación ou alteración deses *tokens*, en caso de ser por exemplo como en EEUU, simples QRs ou webs centralizadas. Pola contra, se se realizara cun sistema de *tokens* seguro, mediante Blockchain por exemplo, implicaría o deseño completo dun novo sistema, o cal sería un gasto de recursos enorme para a sinxeleza ca que se pode modelar isto ca primeira solución exposta. Polo que aínda que podería chegar a non ter problemas de seguridade ou funcionamento se se modelara dunha forma correcta, por motivos de recursos, tempo e existencia de solucións igual de válidas como a primeira, esta opción tamén foi descartada.

Deseño: elementos restantes

O seguinte apéndice actuará como complemento ao explicado no capítulo 3 da memoria, referente ás fases de Análise e Deseño. En concreto, detallaranse aqueles conceptos que polo seu carácter menos esencial e a limitación de extensión da memoria non puideron ser incluídos na mesma, pero que aínda así merece a pena mencionar. Atoparemos a continuación imaxes, anacos de código e explicacións que complementen á información exposta na memoria, incluso referenciando dende a mesma ditos conceptos.

C.1 Casos de uso e diagramas de secuencia

Consulta de saldo

No referente aos casos de uso, simplemente amosar o diagrama de secuencia da consulta de saldo da criptomoeda cara a nosa Blockchain. Nel, vemos como Alice só terá que dende a app, chamar directamente a un método da Blockchain sen parámetros nin coste, para poder obter así o seu balance de criptomoeda, obtida ben por incentivos ou ben polo comercio *intra-app* con ela. Todo isto vémosto nesta Figura C.1.

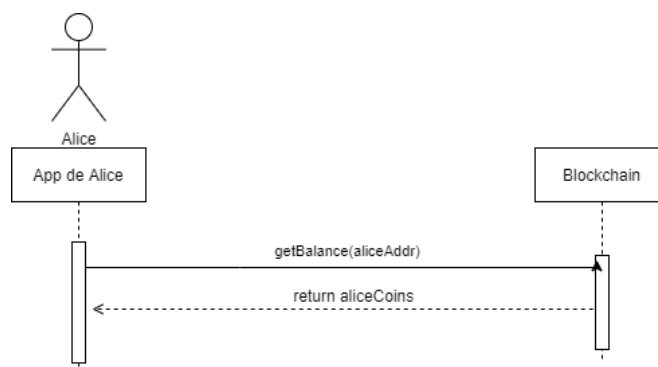


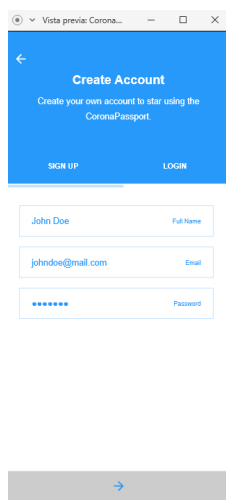
Figura C.1: Alice consulta seu saldo de Coronacoins.

C.2 Deseño da app

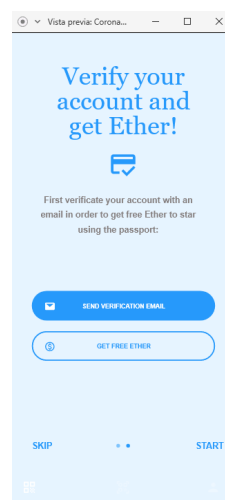
No que respecta ao deseño da app mediante *mockups*, este apéndice tentará incluír, seguindo a mesma estrutura que a sección 3.3, todas aquelas imaxes ou explicacións complementarias ao exposto na memoria. Imos véndoas apartado por apartado.

C.2.1 Parte compartida

Xestión de usuarios e Welcome Sliders



(a) Mockup: Sign Up.



(b) Mockup: Welcome Slider 2.

Figura C.2: Mockups de Sign Up e da segunda Welcome Slider.

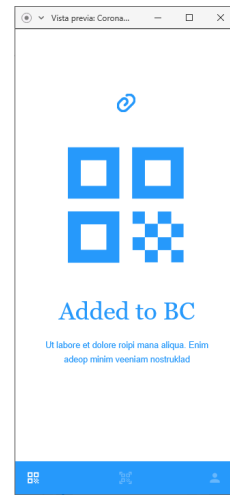
C.2.2 Parte paciente

Patient Home Screen

Nas dúas imaxes posteriores (Figuras C.3a e C.3b) vemos a pantalla principal da app versión paciente nos seus dous posibles estados. Por outro lado, a continuación vemos os dous tipos de JSON que poden aparecer no **Código QR** de ditas pantallas: un cos campos da información persoal inicializados co valor X e co campo *Wallet* ca conta logueada (situación por defecto ao comezar ca app), e outro cos mesmos datos pero cambiando as X por valores, obtidos da información persoal engadida polo usuario.



(a) Pantalla inicial. Os datos non constan na Blockchain.



(b) Pantalla inicial. Os datos están almacenados na Blockchain.

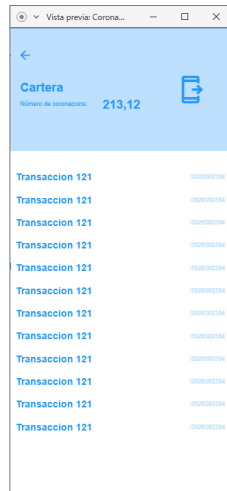
Figura C.3: Mockups das pantallas iniciais posibles na app paciente.

```
1 { "Wallet": "0x6ae5a56f6d89cb76cec4eb689babfed576c0b0e1",
2   "IDNac": "X",
3   "IDMed": "X",
4   "Nombre": "X",
5   "Ap1": "X",
6   "Ap2": "X",
7   "Genero": "X",
8   "FechaNac": "X",
9   "Pais": "X",
10  "Contacto": "X" }
```

```
1 { "Wallet": "0x6ae5a56f6d89cb76cec4eb689babfed576c0b0e1",
2   "IDNac": "1111111",
3   "IDMed": "2222222",
4   "Nombre": "Mauro",
5   "Ap1": "de los Santos",
6   "Ap2": "Nodar",
7   "Genero": "Masc",
8   "FechaNac": "10111998",
9   "Pais": "ESP",
10  "Contacto": "69696969" }
```

Parte criptomoeda

Neste apartado podemos consultar os *mockups* deseñados en primeira instancia do comportamento da app cas accións referentes á criptomoeda.



(a) Mockup: Wallet.

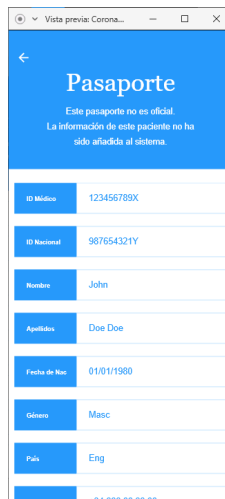


(b) Mockup: Envío de Criptos.

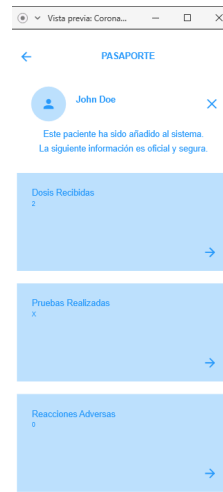
Figura C.4: Mockups das pantallas referentes á criptomoeda.

Ler pasaporte e mostrar información

Mostramos as Figuras C.5a e C.5b para ilustrar respectivamente as dúas posibilidades de lectura de pasaporte: a visualización do QR dunha *wallet* non incorporada ao sistema e a de alguén que si que está dado de alta na cadea.



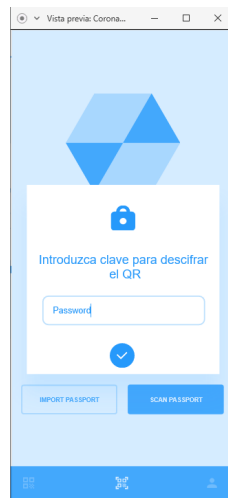
(a) Mockup: Ler Pasaporte non engadido ao sistema.



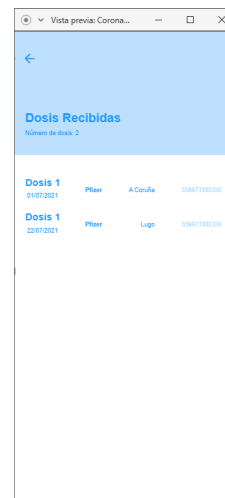
(b) Mockup: Ler pasaporte oficial.

Figura C.5: Mockups da lectura do pasaporte.

Por outro lado, veremos a continuación na Figura C.6a o outro tipo de lectura dun pasaporte, é dicir, a pantalla para a importación dun QR, en concreto, aquela que sae en primeira instancia ao *clickar* en dita funcionalidade, que nos pedirá o contrasinal acordado con orixe co que foi cifrado o pasaporte a importar. Para acabar, vemos na Figura C.6b o deseño da pantalla que mostrará as doses inoculadas en detalle ao paciente do que acabamos de ler o pasaporte, pantalla á que se accederá *clickando* como ben dixemos na memoria, sobre a información referente ás doses amosada na primeira pantalla tras a lectura do pasaporte, é dicir, na representada na Figura C.5b.



(a) Mockup: Ler Pasaporte. Importar QR.



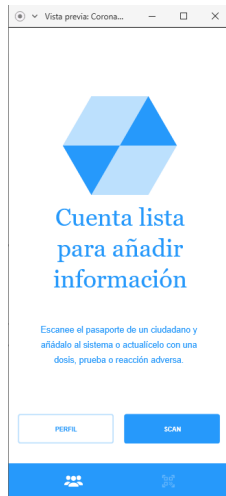
(b) Mockup: Doses en detalle.

Figura C.6: Mockups da importación do pasaporte e das doses en detalle.

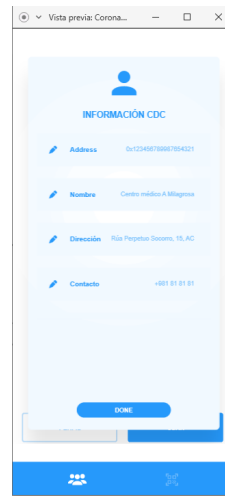
C.2.3 Parte CDC

Perfil

No que respecta á app para os CDCs ou autoridades sanitarias, mostramos de igual forma os *mockups* referenciados dende a memoria ou obviados na mesma polos motivos xa expostos. Por un lado, a pantalla *Home* desta versión da app na Figura C.7a. Por outro lado, o perfil CDC na Figura C.7b, xa que a versión CDC tamén terá dita pantalla na que engadir información manualmente da mesma forma que tiñamos na app paciente, aínda que neste caso non terá unha finalidade concreta no sistema, simplemente será informativa.



(a) Mockup: Home CDC.

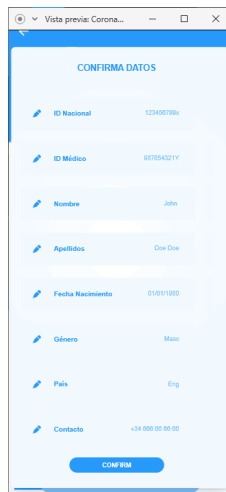


(b) Mockup: Perfil CDC.

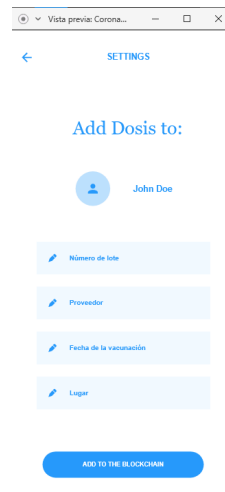
Figura C.7: Mockups da pantalla inicial e de perfil da app CDC.

Escribir no rexistro médico escaneado

Por último, neste apartado veremos os *mockups* referentes á confirmación dos datos a engadir á Blockchain no caso de *clickar* na pantalla de lectura dun pasaporte non engadido (Figura C.5a) no botón para facelo, acción amosada na Figura C.8a. Por outro lado tamén veremos a pantalla posterior a pulsar o botón de engadir unha dose, onde se nos pedirán os datos da mesma como vemos na Figura C.8b.



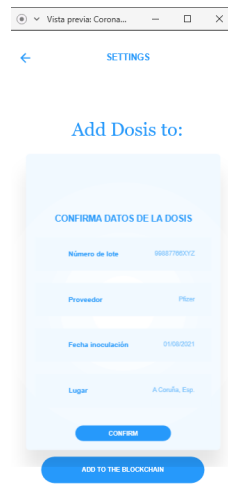
(a) Engadir datos ao sistema dende CDC.



(b) Engadir dose ao pasaporte dende CDC.

Figura C.8: Mockups de engadir datos de perfil e unha dose dende a app CDC.

Por último, veremos a situación modelada de confirmar os datos da dose a engadir por segunda vez antes de escribir na cadea, como amosamos na Figura C.9a, cumprindo así a norma de que para facer calquera escritura no rexistro médico de alguén hai que corroborar dúas veces como mínimo a información a engadir. Por outro lado, vemos a pantalla inicial do CDC tras facer a transacción que escribe dita dose, a cal terá un *pop-up* ca información de dita interacción ca nosa cadea de bloques, como vemos na Figura C.9b.



(a) Mockup: Perfil CDC.



(b) Mockup: Home CDC con mensaxe de transacción realizada correctamente.

Figura C.9: Perfil CDC e Home CDC despois de transacción realizada correctamente.

Selección Tecnolóxica Restante

Expóñense a continuación todas aquelas ferramentas escollidas para o traballo pero non comentadas na sección 4.2 correspondente polo seu carácter menos crítico e a limitación de extensión. Dividímolas de igual maneira en aquelas de ámbito **global**, referentes á **Blockchain**, onde veremos *Ethereum*, a cal foi explicada en detalle na sección da memoria correspondente e para non ser redundante trasladouse a este apéndice, e aquelas referentes á **Dapp**.

D.1 Selección tecnolóxica

D.1.1 Globais

Windows 10

Actual sistema operativo mundialmente coñecido e desenvolvido por Microsoft como parte da familia de sistemas operativos *Windows NT* [67]. Todas as funcionalidades do TFG foron realizadas sobre este **SO**, concretamente sobre a versión *Windows 10 Home de 64 bits*, a cal é un requisito fundamental para poder minar con Geth. Todas as ferramentas expostas neste capítulo teñen a súa última versión para este sistema operativo e algunhas como Metamask están aconselladas para ser usadas e explotadas ao 100% especialmente sobre el. Por estes múltiples motivos foi o sistema operativo elixido para a realización do proxecto.

Git

Sistema distribuído de control de versións gratuíto e *open-source* [68]. Facilita a xestión de proxectos de forma rápida e eficiente. Neste caso, foi utilizado para levar un histórico claro de versións do proxecto, aínda que só foi empregado de forma intensiva para o desenvolvemento de código da **Dapp** móbil, co plugin de Android Studio. Para facer uso del, empregouse **Git Bash** [69], aplicación para Windows que ofrece unha capa de emulación da liña de comandos Git a través dunha *shell*.

yEd

Potente ferramenta gratuita e *open-source* con versión de escritorio para a realización de diagramas de alta calidade [70]. É compatible ca maioría de sistemas operativos, e permite tanto crear diagramas manualmente dende cero como importar datos externos para a súa análise, resultando así nunha das mellores ferramentas *open-source* do mercado para este cometido.

Draw.io

Ferramenta de creación e edición libre de diagramas que permite a integración con diversas plataformas [71]. Como vantaxe recalcar que ten unha moi potente versión web dende a cal se poden crear diagramas e imaxes personalizadas e feitas a medida para cada proxecto diferente, á vez que permite crear todo tipo de diagramas profesionais cunha estética excelente.

D.1.2 Blockchain

Ethereum

Plataforma descentralizada e *open-source* para o traballo ca tecnoloxía Blockchain e a programación de Smart Contracts [72]. A súa gran vantaxe para ser a elixida recae en que é programable, o que encaixa á perfección para a creación de **Dapps** personalizadas e como é neste caso, a creación dunha Blockchain privada baseada en todos os seus principios. Foi explicada en detalle no capítulo anterior (no apartado 4.1).

D.1.3 Dapp

Adobe XD

Editor de gráficos vectoriais desenvolvido e publicado por Adobe co fin de deseñar e crear prototipos da experiencia de usuarios (*mockups*) para aplicacións webs e móbiles [73]. Aínda que é relativamente recente (2015) e bastante complexa, a gran variedade de funcionalidades que ofrece e a súa presenza cada vez maior no mundo empresarial fai que sexa a ferramenta utilizada para o deseño da experiencia de usuario (UX) da **Dapp** móbil.

Android Studio

IDE baseado no software de *IntelliJ IDEA* de *JetBrains* punteiro a nivel mundial que provee das mellores e máis rápidas ferramentas para o desenvolvemento de aplicacións en calquera dispositivo Android [74]. Consta de centos de plugins que facilitan o uso doutros frameworks como poden ser Firebase ou Kotlin, utilizados neste proxecto. Facilita á súa vez a creación de layouts e interfaces, a obtencións de APKs (*Android Package*) finais firmadas á vez que é

un potente editor de código intelixente, entre outras características a destacar. Todas estas funcionalidades, entre outra moitas, sumado a que é o IDE oficial para o desenvolvemento de apps Android, fixo que fora o elixido para a realización deste proxecto.

Blockchain: elementos restantes

A finalidade deste apéndice será complementar e completar a información detallada no Capítulo 5 aportando tanto anacos de código da implementación como imaxes das probas realizadas. De novo, polo seu carácter menos crítico e a limitación de extensión do documento, dita información non puido ser incorporada na memoria, pero pagaba a pena mencionala para reflexar de mellor maneira o traballo realizado á vez que para ofrecer diferentes apoios ás explicacións dadas no capítulo correspondente.

E.1 Implementación

E.1.1 Creación da Blockchain privada

Geth

Para descargala, só haberá que ir á súa páxina web cas *releases* [75] para seleccionar a última versión e instalala como calquera outra, dunha forma rápida e sinxela co instalador gráfico de Windows.

O seguinte será **sincronizarnos** ca rede pública de Ethereum. Esta sincronización consistirá na descarga da cadea de bloques pública de Ethereum dende outros *peers*. Levará certo tempo e conseguirase simplemente ca execución do comando:

```
1 $ geth
```

Genesis Block

Ofrecemos a continuación o código íntegro do JSON do **Genesis Block** do noso proxecto así como unha posterior explicación daqueles parámetros non detallados na memoria:

```
1 {
2   "coinbase"   : "0x8c3fcd01f1d256cf932e8cb25060599191cd4ef3",
3   "difficulty" : "0x20000",
4   "gasLimit"   : "0x2fef8",
```

```

5  "extraData" : "",
6  "nonce"      : "0xc0b1d19",
7  "mixhash"    : "0x0000000000000000000000000000000000000000000000000000000000000000",
8  "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
9  "timestamp"  : "0x60c112a0",
10 "alloc": {
11   "0x8c3fcd01f1d256cf932e8cb25060599191cd4ef3": {
12    "balance": "7000000000000000000000000000000000000000000000000000000000000000"
13   },
14   "0x14f275bfc0a527cf88c7e64f3edbdb1279637257": {
15    "balance": "0"
16   }
17 },
18 "config": {
19   "chainId": 15032020,
20   "homesteadBlock": 0,
21   "eip150Block": 0,
22   "eip155Block": 0,
23   "eip158Block": 0,
24   "byzantiumBlock": 0,
25   "constantinopleBlock": 0,
26   "petersburgBlock": 0,
27   "istanbulBlock": 0
28 }
29 }
30 }
31 }

```

Configuración Bloque Xénese

- **extraData:** Espazo gratuíto e opcional de 32 bytes para engadir a información que consideremos necesaria. No noso caso deixamos o campo baleiro xa que non foi de utilidade engadir nada.
- **nonce & mixhash:** Hashes de 64 bits e 256-bits respectivamente, que combinados demostran que se realizou unha cantidade de cálculos suficientes para cada bloque minado (PoW). É a proba de traballo de minería criptograficamente segura que demostra que se gastou unha cantidade concreta de cálculo na determinación de dito valor. É dicir, verifica ou valida que un bloque foi minado criptograficamente de maneira correcta. Neste primeiro bloque, carece de importancia.
- **parentHash:** Hash de 256-bits que completa ao bloque pai, incluíndo o seu *nonce* e *mixhash*. No bloque xénese é 0 debido a que é o punteiro ao bloque pai, o cal este primeiro bloque non ten. No resto, é un valor diferente a 0.
- **timestamp:** Valor escalar igual á saída da función *time()* de UNIX no inicio deste bloque. Ten varios obxectivos na cadea, como poden ser verificar a orde dos bloques dentro da mesma, ou, combinado co nivel de dificultade de minado, xestionar as diferencias temporais entre bloques. O valor amosado é o número de segundos dende o 1 de Xaneiro de 1970, en valor hexadecimal, á hora de escribir esta memoria. Nun caso real,

habería que recalculalo ao despregalo de novo, aínda que a súa importancia aparece despois deste primeiro bloque.

- **config**: Campo que indica o comezo da configuración da cadea de bloques personalizada en si.

Geth terminal

Detállase de novo o comando de despregue da Blockchain e dun nodo, xunto da explicación daqueles parámetros que non foron explicados na memoria.

```
1 $ geth --datadir ./chaindata --networkid 15032020 --miner.gasprice 0 --txpool.pricelimit 0 --nat any
  --identity private_chain --nodiscover --port=30342 --http --http.port 9545 --http.api
  'admin,shh,txpool,debug,db,eth,net,web3,personal,miner' --http.corsdomain '*' --rpc
  --rpc.allow-unprotected-txs=true
  --rpcapi='admin,shh,txpool,debug,db,eth,net,web3,personal,miner' --rpcport 9545 --rpcaddr
  '0.0.0.0' --rpc.corsdomain "*" --allow-insecure-unlock --unlock 1 console 2>> myEth.log
```

- **--nat**: Poderíase especificar unha dirección IP estática concreta para nomear ao nodo. No noso caso, vale calquera, aínda que nun futuro cun hipotético despregue real a produción de dito proxecto, sería obrigatorio utilizalo para poder acceder de forma pública e descentralizada aos nodos como se explicou na sección de liñas futuras.
- **--identity**: Serve para indicar o tipo da cadea de bloques, no noso proxecto, privada.
- **--nodiscover**: Desactivación da busca automática de *peers*. Habería que configurala manualmente se a quixeramos. Neste primeiro intre, non está configurada debido a que só temos un nodo na cadea para a realización do proxecto.
- **--allow-insecure-unlock & --unlock 1**: Permite o desbloqueo de contas da Blockchain, así como especifica cal desbloquear, neste caso a segunda conta (xa que a 0 sería a primeira). Válido para firmar transaccións ca súa chave privada de forma externa. Aínda que non é unha funcionalidade utilizada no proxecto, en certos casos podería ser útil e podería ser interesante tela en certos nodos con acceso (IP) non público ou privado.

E.1.2 Smart Contracts: rexistro médico e criptomoeda

Estruturas e modifiers do rexistro Médico

Amósase a continuación parte do ficheiro do noso primeiro contrato, **MedicalRecord.sol**, en concreto o código referente aos **tipos de datos complexos** (*structs*):

```
1 struct Dosis {
2     uint256 nLote;
3     string proveedor;
4     string lugar;
```

```

5     string timestamp;
6     address cdc;
7 }
8
9 struct Prueba {
10     uint256 idPrueba;
11     string tipo;
12     string resultado;
13     string timestamp;
14     string lugar;
15     address cdc;
16 }
17
18 struct Record {
19     uint32 idNacional;
20     uint32 idMedico;
21     address direccionPaciente;
22     string nombre;
23     string apellido1;
24     string apellido2;
25     string fechaNacimiento;
26     string genero;
27     string pais;
28     string contacto;
29     Dosis[] dosisRecibidas;
30     Prueba[] pruebasRealizadas;
31     string[] reaccionesAdversas;
32 }
33
34 struct Cdc{
35     string nombre;
36     string direccion;
37     string contacto;
38 }

```

Vemos tamén a continuación aquel código que definíamos na memoria como referente ou complementario aos dous *mappings* principais, así como a chamada ao *modifier* exposto na memoria e a implementación do *modifier dosisRecibida()*, o cal mencionamos na memoria:

```

1 mapping (address => uint256) dirToRecord;
2
3 //mapping (address => Record) records; //opción tamén válida
4 mapping (uint256 => mapping (address => Record)) records;
5 mapping (address => Cdc) cdc;
6
7 mapping (address => bool) public isPatient;
8 mapping (address => bool) public isCdc;

```

```

1 modifier patientExist(address patient) {
2     require(isPatient[patient]);
3     _;
4 }
5
6 function addDosis(address _direccionPaciente, uint256 _nLote, string memory _proveedor, string
7     memory _lugar, string memory _timestamp)
8     public
9     patientExist(_direccionPaciente)

```

```

1 modifier dosisRecibida(address _direccionPaciente) {
2     uint256 recordID = dirToRecord[_direccionPaciente];
3     require(records[recordID][_direccionPaciente].dosisRecibidas
4         .length > 0);
5     _;
6 }

```

Funcións de escritura no rexistro médico

Detállanse a continuación dous exemplos de funcións de escritura na Blockchain co seu código completo, a primeira delas complementando ao amosado na memoria, e a segunda brindando un novo exemplo.

```

1  function addRecord (
2      uint32 _idNacional, uint32 _idMedico,
3      address _direccionPaciente, string memory _nombre,
4      string memory _apellido1, string memory _apellido2,
5      string memory _fechaNacimiento, string memory _genero,
6      string memory _pais, string memory _contacto)
7      public
8      patientNotExist(_direccionPaciente)
9      cdcExist(msg.sender)
10     {
11         records[recordCount][_direccionPaciente].idNacional = _idNacional;
12         records[recordCount][_direccionPaciente].idMedico = _idMedico;
13         records[recordCount][_direccionPaciente].direccionPaciente = _direccionPaciente;
14         records[recordCount][_direccionPaciente].nombre = _nombre;
15         records[recordCount][_direccionPaciente].apellido1 = _apellido1;
16         records[recordCount][_direccionPaciente].apellido2 = _apellido2;
17         records[recordCount][_direccionPaciente].fechaNacimiento = _fechaNacimiento;
18         records[recordCount][_direccionPaciente].genero = _genero;
19         records[recordCount][_direccionPaciente].pais = _pais;
20         records[recordCount][_direccionPaciente].contacto = _contacto;
21
22
23         isPatient[_direccionPaciente] = true;
24         dirToRecord[_direccionPaciente] = recordCount;
25
26         emit PatientRecordAdded(recordCount, _direccionPaciente);
27
28         patientCount += 1;
29         recordCount += 1;
30
31         payReward(_direccionPaciente, 3);
32     }

```

```

1  function addDosis(
2      address _direccionPaciente, uint256 _nLote,
3      string memory _proveedor, string memory _lugar,
4      string memory _timestamp)
5      public
6      patientExist(_direccionPaciente)
7      cdcExist(msg.sender)
8      {
9          uint256 recordID = dirToRecord[_direccionPaciente];
10         uint lengthDoses = records[recordID][_direccionPaciente].dosisRecibidas.length;
11
12         Dosis memory tmp;
13         tmp.nLote = _nLote;
14         tmp.proveedor = _proveedor;
15         tmp.lugar = _lugar;
16         tmp.timestamp = _timestamp;
17         tmp.cdc = msg.sender;
18
19         records[recordID][_direccionPaciente].dosisRecibidas.push(tmp);
20
21         payReward(_direccionPaciente, 2);
22         emit DoseAdded(recordID, _direccionPaciente,
23             records[recordID][_direccionPaciente].dosisRecibidas[lengthDoses]);
24     }

```

Funcións de lectura do rexistro médico

De igual forma temos as dúas funcións de consulta á Blockchain das que falabamos en dito subapartado na memoria, amosando o seu código íntegro a continuación:

```

1  function getRecord(address _direccionPaciente)
2      public returns (
3          uint32 _idNacional, uint32 _idMedico,
4          string memory _nombre,
5          string memory _apellido1, string memory _apellido2,
6          string memory _fechaNacimiento, string memory _genero,
7          string memory _pais, string memory _contacto,
8          Dosis[] memory _dosisRecibidas,
9          Prueba[] memory _pruebasRealizadas,
10         string[] memory _reaccionesAdversas)
11  {
12      uint256 recordID = dirToRecord[_direccionPaciente];
13
14      _idNacional = records[recordID][_direccionPaciente].idNacional;
15      _idMedico = records[recordID][_direccionPaciente].idMedico;
16      _nombre = records[recordID][_direccionPaciente].nombre;
17      _apellido1 = records[recordID][_direccionPaciente].apellido1;
18      _apellido2 = records[recordID][_direccionPaciente].apellido2;
19      _fechaNacimiento = records[recordID][_direccionPaciente].fechaNacimiento;
20      _genero = records[recordID][_direccionPaciente].genero;
21      _pais = records[recordID][_direccionPaciente].pais;
22      _contacto = records[recordID][_direccionPaciente].contacto;
23      _dosisRecibidas = records[recordID][_direccionPaciente].dosisRecibidas;
24      _pruebasRealizadas = records[recordID][_direccionPaciente].pruebasRealizadas;
25      _reaccionesAdversas = records[recordID][_direccionPaciente].reaccionesAdversas;
26
27      emit PatientRecordAccesed(_idNacional, _idMedico, _nombre, _apellido1, _apellido2,
28      _fechaNacimiento, _genero, _pais, _contacto, _dosisRecibidas , _pruebasRealizadas ,
29      _reaccionesAdversas);
30
31      payReward(_direccionPaciente, 1);
32  }
33
34  function getFreeRecord(address _direccionPaciente)
35      public
36      view
37      returns (
38          uint32 _idNacional, uint32 _idMedico,
39          string memory _nombre,
40          string memory _apellido1, string memory _apellido2,
41          string memory _fechaNacimiento, string memory _genero,
42          string memory _pais, string memory _contacto,
43          Dosis[] memory _dosisRecibidas,
44          Prueba[] memory _pruebasRealizadas,
45          string[] memory _reaccionesAdversas)
46  {
47      uint256 recordID = dirToRecord[_direccionPaciente];
48      _idNacional = records[recordID][_direccionPaciente].idNacional;
49      _idMedico = records[recordID][_direccionPaciente].idMedico;
50      _nombre = records[recordID][_direccionPaciente].nombre;
51      _apellido1 = records[recordID][_direccionPaciente].apellido1;
52      _apellido2 = records[recordID][_direccionPaciente].apellido2;
53      _fechaNacimiento = records[recordID][_direccionPaciente].fechaNacimiento;
54      _genero = records[recordID][_direccionPaciente].genero;
55      _pais = records[recordID][_direccionPaciente].pais;
56      _contacto = records[recordID][_direccionPaciente].contacto;
57      _dosisRecibidas = records[recordID][_direccionPaciente].dosisRecibidas;
58      _pruebasRealizadas = records[recordID][_direccionPaciente].pruebasRealizadas;
59      _reaccionesAdversas = records[recordID][_direccionPaciente].reaccionesAdversas;
60  }

```

Criptomoeda

Na parte referente á Criptomoeda, comezaremos expoñendo o código do contrato do *Token*, o que implementa a criptomoeda:

```

1  contract Coronacoin is StandardToken, Ownable {
2      string public constant name = "Coronacoin";
3      string public constant symbol = "CVD";
4      uint8 public constant decimals = 32;
5
6      uint256 public constant INITIAL_SUPPLY = 16e10;
7
8      constructor() public {
9          totalSupply_ = INITIAL_SUPPLY;
10         balances[msg.sender] = INITIAL_SUPPLY;
11         emit Transfer(address(0x0), msg.sender, INITIAL_SUPPLY);
12     }
13 }

```

Por outro lado vemos esas variables das que falabamos na sección análoga da memoria situadas nunha parte do contrato do rexistro médico onde se referencia por primeira vez e se fai uso de dita criptomoeda:

```

1  Coronacoin public coronacoin;
2  address public tokenAddress;
3  uint256 public tokenRewardBigAmount;
4  uint256 public tokenRewardMediumAmount;
5  uint256 public tokenRewardSmallAmount;
6  }

```

E por último, detallaremos a inicialización de dita criptomoeda no construtor do contrato do rexistro médico.

```

1  constructor()
2      public
3  {
4      setCoronacoin(address(new Coronacoin()));
5      uint256 initialSmallReward = 10;
6      uint256 initialMediumReward = 100;
7      uint256 initialBigReward = 1000;
8      setCoronacoinRewards(initialSmallReward, initialMediumReward, initialBigReward);
9  }
10 }

```

E como non, todas aquelas funcións encargadas das funcionalidades referentes á mesma, que van dende creala ou cambiar o valor dos incentivos até pagar os mesmos ou consultar o saldo, estas dúas últimas xa detalladas na memoria pero amosadas completas a continuación.

```

1  function setCoronacoin(address _newCoronacoin)
2      internal
3      onlyOwner
4      notNull(_newCoronacoin)
5  {
6      coronacoin = Coronacoin(_newCoronacoin);
7      tokenAddress = address(coronacoin);
8  }
9
10 function setCoronacoinRewards(uint256 _tokenRewardSmall, uint256 _tokenRewardMedium,
11                               uint256 _tokenRewardBig)
12     public
13     onlyOwner
14     higherThanZero(_tokenRewardSmall)
15     higherThanZero(_tokenRewardMedium)

```



```

16     higherThanZero(_tokenRewardBig)
17     {
18         tokenRewardSmallAmount = _tokenRewardSmall;
19         tokenRewardMediumAmount = _tokenRewardMedium;
20         tokenRewardBigAmount = _tokenRewardBig;
21         emit TokenRewardSet(_tokenRewardSmall, _tokenRewardMedium, _tokenRewardBig);
22     }
23
24     function payReward(address _patientAddress, int number)
25     private
26     notNull(_patientAddress)
27     {
28         if(number == 3){
29             coronacoin.transfer(_patientAddress, tokenRewardBigAmount);
30         } else if(number == 2){
31             coronacoin.transfer(_patientAddress, tokenRewardMediumAmount);
32         } else{
33             coronacoin.transfer(_patientAddress, tokenRewardSmallAmount);
34         }
35         emit RewardPaid(_patientAddress);
36     }
37
38     function balanceOfCoronacoins(address _patientAddress)
39     public
40     view
41     notNull(_patientAddress)
42     returns (uint256)
43     {
44         return coronacoin.balanceOf(_patientAddress);
45     }

```

A maiores, ensinar nesta Figura E.1, a organización final dos Smart Contracts unha vez acabado o proxecto, da cal falabamos na memoria e onde podemos observar como que aínda que o noso sistema estea centrado en tres contratos, **MedicalRecord**, **Coronacoin** e **CoronaFaucet**, farán falta un bo número extra deles para conseguir todas as funcionalidades que desexamos.

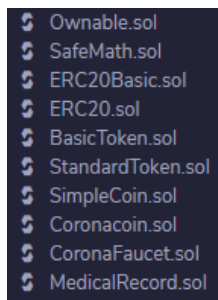


Figura E.1: Smart Contracts en Remix.

Por último, e para acabar ca parte referente aos contratos relacionados ca criptomoeda, teremos o código completo daquel contrato que implementa ao noso **Faucet personalizado**:

```

1 contract CoronaFaucet {
2
3     [...]
4
5     receive() external payable {
6         // react to receiving ether
7         emit etherReceived(msg.value);
8     }

```

```

9
10 [...]
11
12 function giveMeEther() public{
13     uint balance = address(this).balance;
14     if (balance < 1 ether){ //if there is no Ether
15         emit etherOutOfStock(balance);
16     }else if (balance < 100000 ether){ //if SC balance lower than 100k
17         uint percentage = balance * 0.001 ether; //send the 0.1% of Ether
18         msg.sender.transfer(percentage);
19         emit etherGiven(msg.sender);
20         emit etherMinimum(balance);
21     }else{
22         msg.sender.transfer(100 ether); //else send 100 Ether
23         emit etherGiven(msg.sender);
24     }
25 }
26 }
    
```

Eventos

Para acabar con esta sección, amosaremos como ben dicíamos na memoria, o código íntegro da **definición** (*event*) da maioría dos **eventos** presentes nos nosos Smart Contracts, xa que a **chamada** aos mesmos (ca función de *Solidity emit*) pode atoparse nos anteriores exemplos de código. Primeiro, os de *MedicalRecord.sol*:

```

1 event PatientRecordAccesed(uint32 _idNacional,
2     uint32 _idMedico, string _nombre, string _apellido1, string _apellido2,
3     string _fechaNacimiento, string _genero, string _pais, string _contacto,
4     Dosis[] _dosisRecibidas, Prueba[] _pruebasRealizadas, string[] _reaccionesAdversas);
5 event PatientRecordAdded(uint256 recordID, address patientAddress);
6 event CdcAdded(address cdcAddress);
7 event DoseAdded(uint256 recordID, address patientAddress, Dosis newDose);
8 event TestAdded(uint256 recordID, address patientAddress, Prueba newTest);
9 event ReactionAdded(uint256 recordID, address patientAddress, string reaccion);
10 event RewardPaid(address patientAddress);
11 event TokenRewardSet(uint256 _tokenRewardSmall,
12     uint256 _tokenRewardMedium, uint256 _tokenRewardBig);
    
```

E por último, os do noso Faucet (*CoronaFaucet.sol*):

```

1 event etherMinimum(uint etherRemaining);
2 event etherGiven(address patientAddress);
3 event etherReceived(uint amount);
4 event etherOutOfStock(uint etherRemaining);
    
```

E.2 Probas

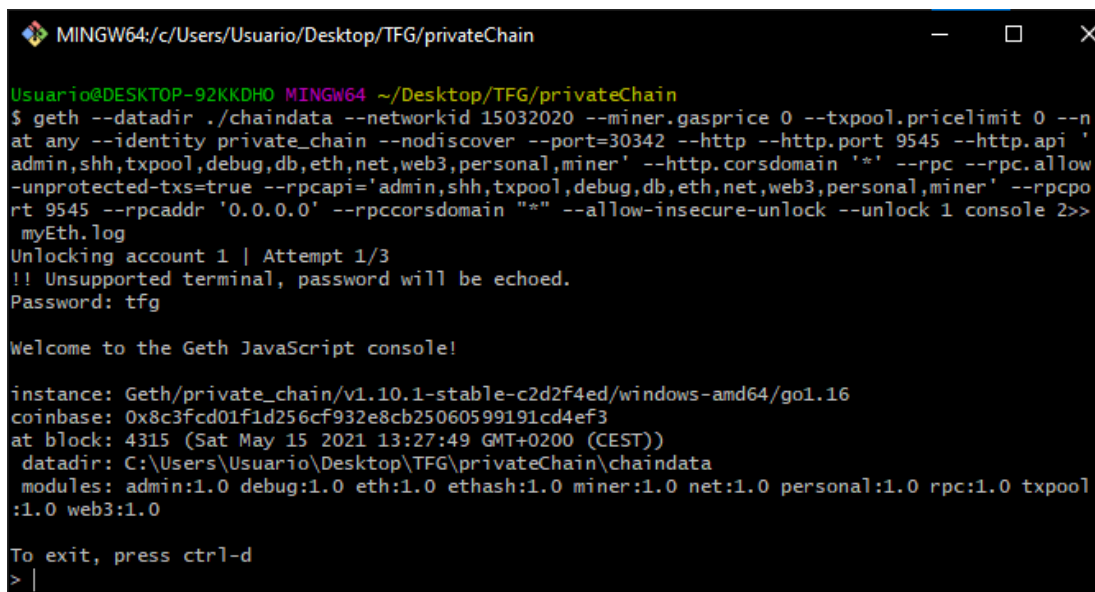
E.2.1 Blockchain Privada

Post-execución dos comandos

Comando para observar o ficheiro ao que enviamos os logs:

```
1 $ tail -f myEth.log
```

Información do comando que inicia o nodo da Blockchain (Figura E.2) e ficheiro de logs da cadea tras iniciala (Figura E.3):



```

MINGW64/c/Users/Usuario/Desktop/TFG/privateChain
Usuario@DESKTOP-92KKDHO MINGW64 ~/Desktop/TFG/privateChain
$ geth --datadir ./chaindata --networkid 15032020 --miner.gasprice 0 --txpool.pricelimit 0 --n
at any --identity private_chain --nodiscover --port=30342 --http --http.port 9545 --http.api '
admin,shh,txpool,debug,db,eth,net,web3,personal,miner' --http.corsdomain '*' --rpc --rpc.allow
-unprotected-txs=true --rpcapi='admin,shh,txpool,debug,db,eth,net,web3,personal,miner' --rpcpo
rt 9545 --rpcaddr '0.0.0.0' --rpccorsdomain "*" --allow-insecure-unlock --unlock 1 console 2>>
myEth.log
Unlocking account 1 | Attempt 1/3
!! Unsupported terminal, password will be echoed.
Password: tfg

Welcome to the Geth JavaScript console!

instance: Geth/private_chain/v1.10.1-stable-c2d2f4ed/windows-amd64/go1.16
coinbase: 0x8c3fcd01f1d256cf932e8cb25060599191cd4ef3
at block: 4315 (Sat May 15 2021 13:27:49 GMT+0200 (CEST))
 datadir: C:\Users\Usuario\Desktop\TFG\privateChain\chaindata
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool
:1.0 web3:1.0

To exit, press ctrl-d
> |

```

Figura E.2: Inicio dun nodo na nosa Blockchain privada.



```

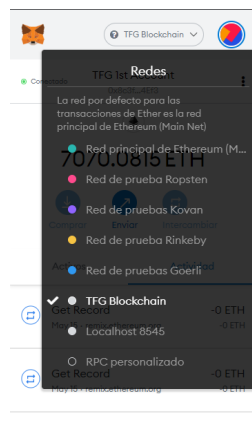
MINGW64/c/Users/Usuario/Desktop/TFG/privateChain
Usuario@DESKTOP-92KKDHO MINGW64 ~/Desktop/TFG/privateChain
$ tail -f myEth.log
INFO [06-01|18:37:58.326] IPC endpoint opened           url=\\.pipe\geth.ipc
ERROR [06-01|18:37:58.327] Unavailable modules in HTTP API list  unavailable="[shh db]" available="[admin debug web3 eth txpool pe
rsonal ethash miner net]"
INFO [06-01|18:37:58.328] HTTP server started           endpoint=[::]:9545 prefix= cors= vhosts=localhost
WARN [06-01|18:37:58.328] -----
WARN [06-01|18:37:58.328] Referring to accounts by order in the keystore folder is dangerous!
WARN [06-01|18:37:58.328] This functionality is deprecated and will be removed in the future!
WARN [06-01|18:37:58.328] Please use explicit addresses! (can search via 'geth account list')
WARN [06-01|18:37:58.328] -----
INFO [06-01|18:38:02.216] Unlocked account             address=0x14f275bfc0a527cf88c7e64f3e6db1279637257
INFO [06-01|18:38:02.267] Etherbase automatically configured  address=0x8c3fcd01f1d256cf932e8cb25060599191cd4ef3

```

Figura E.3: Logs do inicio dun nodo na nosa Blockchain privada.

E.2.2 Remix + Metamask: Cadea privada e primeiros contratos

Primeiro, deberemos instalar MetaMask como extensión de Google Chrome de maneira moi sinxela e como fariamos con calquera outra. Unha vez que a temos, creamos unha conta, abrimos Remix e pulsamos sobre a icona da mesma para loguearnos. Unha vez logueados, teremos que conectarnos a unha Blockchain. Por defecto, MetaMask ofrécenos a conexión ca rede pública e cas *testnets* máis coñecidas de Ethereum, como ben vemos na Figura E.4a, pero nós queremos conectarnos ca Blockchain que acabamos de despregar. Polo que, como sabemos que temos un *end-point* RPC, teremos que clicar en engadir RPC personalizado e encher os ocos ca información amosada na Figura E.4b, a cal tamén foi ilustrada na memoria.



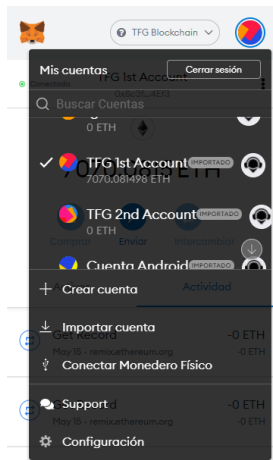
(a) Metamask: Elixir Blockchain.

Nombre de la Red	TFG Blockchain
Nueva URL RPC	http://127.0.0.1:9545
ID de Cadena	15032020
Símbolo (opcional)	ETH
URL del Explorador de bloques (opcional)	
<input type="button" value="Cancelar"/> <input type="button" value="Guardar"/>	

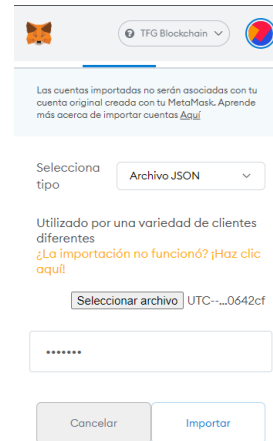
(b) Detalles da Blockchain á cal conectarse.

Figura E.4: Conectándonos á nosa cadea con Metamask.

Tras escribir o seu identificador (importante para o envío de transaccións, o cal ten que ser o mesmo que o especificado no *Genesis Block* para evitar ataques *replay* e securizar a cadea, como ben explicamos anteriormente nesta memoria) e a súa URL, onde introduciremos o *endpoint* que nos marcou o log ao despregala, como tamén amosamos anteriormente, tras poder establecer opcionalmente un símbolo e un explorador de bloques, o cal non temos ao ser unha Blockchain privada, poderemos conectarnos a dita cadea de bloques. Para confirmar que todo foi ben, importamos o ficheiro da conta creada e ver seu saldo, así como enviar unha transacción á cadea. Tanto dende a extensión, como *clickando* en *Ampliar Vista* para facelo dende pantalla completa, vemos na Figura E.5a en primeiro lugar as contas importadas (que por diversas probas feitas durante a memoria xa nos deixan diferenciar as dúas contas da Blockchain co mesmo saldo que nos aparecía na *Geth Console* anteriormente), pero para confirmar na súa totalidade que o funcionamento foi correcto, *clickaremos* en *Import Account* para importar a conta que acabamos de crear. Para elo, poderemos facelo ou ben ca súa chave privada, ou cun ficheiro JSON, o cal será aquel con nome UTC almacenado na carpeta *keystore* do cal xa falamos. Eliximos e importamos dita conta, como se mostra na Figura E.5b.



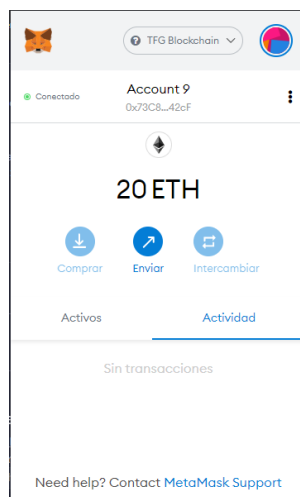
(a) Accounts de Metamask.



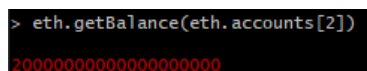
(b) Importando dende MetaMask conta da Blockchain.

Figura E.5: Probando as contas da cadea con Metamask.

Por último, para corroborar que todo foi ben e tras ver o Ether dispoñible, o cal concorda co que viamos dende a *Geth Console* tras executar os primeiros comandos da sección de proba na Figura 5.2, a última proba consistirá en enviar unha transacción. Como aínda non temos contratos para enviar información personalizada, imos enviar Ether dende esta conta que acabamos de crear até a segunda conta que tiñamos, a cal tiña a carteira baleira de Ether, como viamos na Figura E.5a e especificabamos no bloque xénese. Tamén veremos nos logs da nosa cadea que ocorre ao enviar esta transacción, e se tras minar, a información da mesma cambia. Vémolos todo en conxunto nas seguintes Figuras E.6a, E.6b, E.7a, E.7b e E.8.



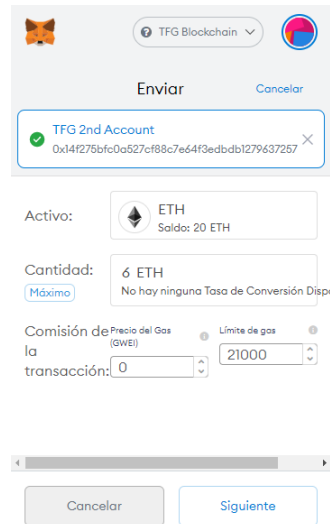
(a) Conta importada satisfatoriamente.



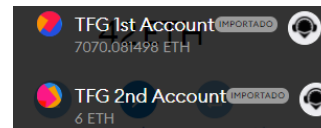
(b) Saldo da conta importada.

Figura E.6: Probando a importar conta com Metamask e verificar saldo.

Apuntar que como en vez deses previos 18 de Ether, temos 20 (como vemos na Figura E.6a), debido a que eses dous extra foron engadidos por uns segundos de minado. Para corroboralo, se volvemos consultar o saldo de dita conta, aparécenos que efectivamente é 20, feito amosado na Figura E.6b e o cal confirma que a información brindada por MetaMask é a axeitada e polo tanto que a conta foi creada e importada correctamente. Imos agora dende ela enviar 6 de Ether á segunda conta, a cal tiña 0 de saldo como viamos na Figura E.5a.



(a) Enviando transacción dende Metamask.



(b) Comprobando os saldos das contas da Blockchain tras o envío da transacción.

Figura E.7: Enviando transacción dende Metamask e comprobando resultado.

```
INFO [06-01|19:19:19.612] Submitted transaction hash=0x337a3a14c9fae8eb5835d1f5598c6fc950588bf21175715e0fda2bed5a
bbf26d From=0x73C8B0e928823C597b4CD79Fdb480dbd220642cF nonce=0 recipient=0xd4f275bfc0a527cf88c7e64f3edbdb1279637257 value=6000000000
0000000000
```

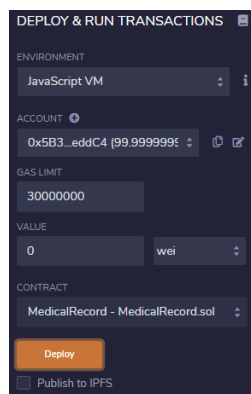
Figura E.8: Comprobando os logs da Blockchain tras o envío da transacción.

Como vemos na Figura E.7b, agora a conta ten 6 Ether, os cales foron enviados dende a conta recentemente creada ca transacción de Metamask (Figura E.7a), e por riba, dita transacción pódese ver en detalle nos logs ca Figura E.8, onde nos amosa que conta envía dita transacción, con que valor (*value*, neste caso o Ether transferido) e até que conta. Comportamento exactamente igual ao desexado, polo que grazas a esta serie de probas realizadas, podemos concluír que creamos adecuadamente a nosa cadea de bloques, a cal está completamente lista para comezar despregar Smart Contracts nela e mediante transaccións, interactuar con eles. Imos pois, ca seguinte fase de probas. Cabe mencionar que as imaxes E.7a e E.8 foron tamén amosadas na memoria, pero repítense nesta sección para dar unha maior claridade ás probas expostas e ao resto de figuras.

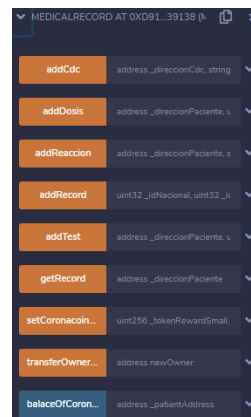
E.2.3 Smart Contracts: rexistro médico e criptomoneda

Despregue e proba mock

Para facer estas probas, as cales non foron amosadas na memoria polo seu carácter *mock*, non necesitaremos nada máis que despregar nas máquinas virtuais de Remix, cas contas pre-establecidas que xa nos brinda esta ferramenta, ditos Smart Contracts e probar as súas funcionalidades grazas aos recursos gráficos que nos da. Os pasos serían os amosados nas figuras E.9a, E.9b e E.10.



(a) Despregue e proba mock dende Remix dos primeiros Smart Contracts (1).



(b) Despregue e proba mock dende Remix dos primeiros Smart Contracts (2).

Figura E.9: Primeiras probas mocks no despregue de Smart Contracts dende Remix.



Figura E.10: Despregue e proba mock dende Remix dos primeiros Smart Contracts.

Como vemos, dito proceso será moi sinxelo e teremos que primeiro, como se amosa na Figura E.9a, elixir o tipo de despregue de máquina virtual Java (**JavaScript VM**), para tras engadir un 0 ao valor máximo do límite de gas, xa que como imos despregar contratos longos e facer escrituras pesadas será necesario un **alto límite de gas** como ben explicamos na sección de configuración do bloque xénese 5.1.1, bastará con darlle ao botón **Deploy** co Smart Contract desexado para ver como a transacción foi correctamente executada, até vemos a dirección do contrato despregado na Figura E.10. Por último, na Figura E.9b, vemos parte das funcións do contrato cas cales interactuar e facer probas. Aínda que non as imos probar todas, xa que iremos repartindo chamadas ás mesmas entre as diferentes seccións de probas da memoria e deste apéndice, imos neste caso de despregue *mock*, probar a engadir un novo rexistro médico dun paciente ao sistema e acceder á súa información co método de lectura de pasaporte gratuito dende o CDC engadido na proba amosada na memoria.

(a) Engadir rexistro de paciente á Blockchain.

(b) Chamada á lectura gratuita do rexistro engadido.

Figura E.11: Probas mock de engadir rexistro e ler gratuitamente o mesmo.

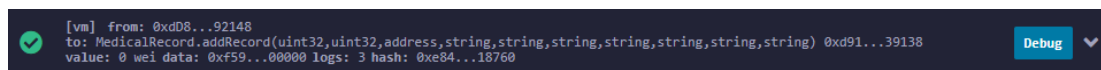


Figura E.12: Resultado da proba mock de engadir rexistro de paciente á Blockchain.

A maiores, para comprobar que podemos acceder aos datos de dito rexistro e paciente, facendo unha chamada á función de lectura gratuita do pasaporte, vemos como a información incorporada á Blockchain é a mesma que a devolta polo método, o cal nos confirma o correcto funcionamento do contrato intelixente. Vémolo na Figura E.11b.

Despregue e proba na Blockchain privada

O único non amosado destas probas na memoria foi a mensaxe de confirmación de Remix á transacción que despregou o noso Smart Contract de *MedicalRecord.sol* na nosa Blockchain privada, o cal vemos na Figura E.13.

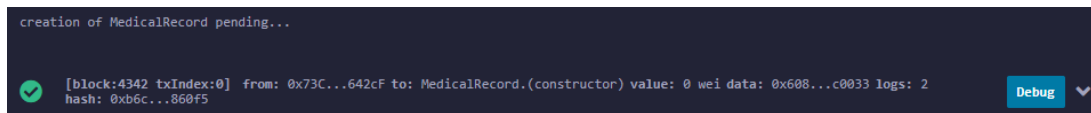


Figura E.13: Log de Remix do despregue do contrato na nosa cadea.

Elementos restantes da Dapp

Este apéndice será análogo ao anterior, pero desta vez en lugar de para a parte Blockchain, para aquela referente á aplicación móbil distribuída. Tentará aportar aquel código mencionado ou resumido no Capítulo 6 da memoria, así como ofrecer imaxes e explicacións complementarias ás probas expostas na mesma.

F.1 Implementación

F.1.1 Lóxica destacada

Conexión da Blockchain

```

1  try{
2      Log.d(TAG, "Connecting to the Blockchain...")
3      var web3: Web3j =
4          Web3j.build(HttpService("http://192.168.0.14:9545"))
5      try {
6          val clientVersion: Web3ClientVersion =
7              web3.web3ClientVersion().sendAsync().get()
8          if (!clientVersion.hasError()) {
9              Log.d(TAG, "Connected to the Blockchain!")
10             } else {
11                 Log.d(TAG, "Error occurred during the connection attempt.")
12             }
13         } catch (e: Exception) {
14             Log.d(TAG, "Exception thrown in connection attempt: " + e.message)
15         }
16         Log.d(TAG, "Setting up Bouncy Castle")
17         setupBouncyCastle();

```

Escrituras básicas e Encrypted Shared Preferences

```

1  try {
2      walletDir = File("$walletPath/$fileName")
3      var credentials: Credentials = WalletUtils.loadCredentials(pwd, walletDir)
4      val masterKeyAlias: String = MasterKeys.getOrCreate(MasterKeys.AES256_GCM_SPEC)
5      val sharedPreference: SharedPreferences = EncryptedSharedPreferences.create(
6          "pasaporte",
7          masterKeyAlias,
8          applicationContext,

```

```

9      EncryptedSharedPreferences.PrefKeyEncryptionScheme.AES256_SIV,
10     EncryptedSharedPreferences.PrefValueEncryptionScheme.AES256_GCM
11   )
12   with(sharedPref.edit()) {
13     putString("key", credentials.ecKeyPair.privateKey.toString())
14     putString("account", credentials.address)
15     putString("pubkey", credentials.ecKeyPair.publicKey.toString())
16     commit()
17   }
18   return credentials
19 } catch (e: java.lang.Exception) {
20   Log.d(TAG, "Error during wallet creation: " + e.message)
21   return null
22 }

```

Miscelánea: Firebase, Algoritmos Criptográficos e Manexo de códigos QR

```

1 private fun signIn() {
2     if (identicalPassword()) {
3         spinner.visibility = View.VISIBLE
4         userEmail = etEmail.text.toString().trim()
5         userPassword = etPassword.text.toString().trim()
6         /*create a user*/
7         firebaseAuth.createUserWithEmailAndPassword(userEmail, userPassword)
8             .addOnCompleteListener { task ->
9                 if (task.isSuccessful) {
10                     toast("created account successfully !")
11                     val credentialsReturned = createWallet(userPassword)
12                     val intent = Intent(this, IntroSliderActivity::class.java)
13                     intent.putExtra("privateKey",
14                         credentialsReturned?.ecKeyPair?.privateKey.toString())
15                     intent.putExtra("account", credentialsReturned?.address)
16                     startActivity(intent)
17                     finish()
18                 } else {
19                     toast(task.exception.toString())
20                     spinner.visibility = View.INVISIBLE
21                 }
22             }
23     }
24 }
25 private fun sendEmailVerification() {
26     firebaseAuth.currentUser?.let {
27         it.sendEmailVerification().addOnCompleteListener { task ->
28             if (task.isSuccessful) {
29                 toast("email sent to $userEmail")
30             } else {
31                 toast(task.exception.toString())
32             }
33         }
34     }
35 }

```

A continuación amósanse ditos fragmentos de código referentes ás funcionalidades do **escáner QR**, do **xerador de códigos QR** e do **cifrado simétrico** de mensaxes, dos cales se fala na memoria pero non se chegan a exemplificar.

```

1 R.id.navigation_qrScanner -> {
2     /* Create a ZXing IntentIntegrator and start the QR code scan */
3     val integrator = IntentIntegrator(this)
4     integrator.setRequestCode(REQUEST_CODE)
5     integrator.setOrientationLocked(true)
6     integrator.initiateScan()
7     return@OnNavigationItemSelectedListener true
8 }

```

```

9
10 [...]
11
12 private fun generateQRCode(text: String): Bitmap {
13     val width = 500
14     val height = 500
15     val bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888)
16     val codeWriter = MultiFormatWriter()
17     try {
18         val bitMatrix = codeWriter.encode(text, BarcodeFormat.QR_CODE, width, height)
19         for (x in 0 until width) {
20             for (y in 0 until height) {
21                 bitmap.setPixel(x, y, if (bitMatrix[x, y]) Color.BLACK else Color.WHITE)
22             }
23         }
24     } catch (e: WriterException) {
25         Log.d("Debugging", "generateQRCode: ${e.message}")
26     }
27     return bitmap
28 }
29
30 [...]
31
32 val spec = PBEKeySpec(pwd.toCharArray(), salt, 1000, 32 * 8)
33 val key: SecretKey =
34     SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1")
35         .generateSecret(spec)
36
37 val directory: File = filesDir
38
39 val file = File(directory, "qr.txt")
40
41 var fos: FileOutputStream? = null
42 try {
43     fos = FileOutputStream(file)
44     fos.write(encryptMsg(message, key))
45     fos.close()
46
47 //CÓDIGO JAVA PARA CIFRAR CON AES
48 public static byte[] encryptMsg(
49     [...]
50     /* Encrypt the message. */
51     Cipher cipher = null;
52     cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
53     cipher.init(Cipher.ENCRYPT_MODE, secret);
54     byte[] cipherText = cipher.doFinal(message.getBytes("UTF-8"));
55     return cipherText;
56 }

```

Transaccións ca Blockchain

Primeiro de todo, vemos a captura e o comando do proceso de creación de *wrappers* Java a partir dos nosos Smart Contracts escritos en *Solidity*, necesarios para efectuar correctamente as transaccións cara eles.

- Copiar o **ABI** do Smart Contract a un ficheiro.

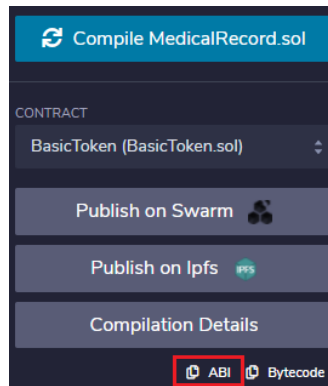


Figura F.1: Descargar ABI dende Remix.

- Xerar os wrappers con Web3j.

```
1 $ web3j generate solidity --abiFile=MedicalRecord5.abi --package=smartcontracts
  --outputDir=src/main/java/tfg/delossantos/coronapass/androidApp/
```

A continuación, vemos a lóxica completa dunha transacción cara a nosa cadea de bloques:

```
1 val credentials: Credentials = Credentials.create(ECKeypair.create(BigInteger(key)))
2 val contractAddress = "0x0c8E392EF799F3EADBAe7B4d780716533ad03347"
3 val gasProvider = StaticGasProvider(
4     Convert.toWei("20000", Convert.Unit.WEI).toBigInteger(),
5     BigInteger("3000000")
6 )
7 val medicalRecord = MedicalRecord3.load(
8     contractAddress,
9     web3,
10    credentials,
11    gasProvider
12 )
13 val transactionReceipt: Tuple12<BigInteger, BigInteger, String, String, String, String, String,
14     String, String, MutableList<MedicalRecord.Dosis>, MutableList<MedicalRecord.Prueba>,
15     MutableList<String>>? = medicalRecord.getFreeRecord(
16         walletJson
17     ).sendAsync().get()
18 [...]
19 val transactionReceipt0: TransactionReceipt? = medicalRecord.addDosis(acc, BigInteger(nLote),
20     provider, lugar, times).sendAsync().get()
21 [...]
22 val balance: EthGetBalance = web3.ethGetBalance(
23     credentials.address,
24     DefaultBlockParameterName.LATEST
25 ).sendAsync().get()
```

F.2 Probas

F.2.1 Fluxo de execución A: Primeiros pasos e App CDC

Xestión de Usuarios e Faucet

Confirmamos ca seguinte Figura F.2 que na consola Firebase aparece o usuario creado nas probas detalladas na memoria ca mesma data.


Identificador	Provedores	Fecha de creación
mauro.delossantos@udc.es		13 jun. 2021

Figura F.2: Consola Firebase: confirmación da correcta creación da nova conta.

Por outro lado, podemos verificar a chamada ao noso Faucet visualizando os logs da propia cadea como facemos na Figura F.3, onde observamos a dirección do noso Smart Contract *CoronaFaucet.sol* (*recipient*).

```
INFO [06-13|14:08:54.716] Submitted transaction hash=0x2c982eb44574d105f718
f50ae77f10267c3fcd1a7787db8006433b6698e1f913 from=0x02EC282320dE1e339Dc2831F4C6EFf1bDE8c450 n
once=0 recipient=0x62227410914108DBD53E94A4D4AF6e759d1AE3DF value=0
```

Figura F.3: Log da Blockchain ao chamar ao Faucet.

F.2.2 Fluxo de execución B: App paciente e Funcionalidades Extra

Na Figura F.4 podemos corroborar a transacción de criptomoedas amosadas na memoria consultando dita transacción directamente no log da nosa Blockchain Privada.

```
INFO [06-13|17:43:06.789] Submitted transaction hash=0xd2a6ba5cbb32cad29890
2b8937b8d2bc0740aaf9a9fc4fe86468abdb6c1c9ed8 from=0x02EC282320dE1e339Dc2831F4C6EFf1bDE8c450 n
once=5 recipient=0xb21deab7E32502DaC657340ED82bD89d63024A23 value=0
```

Figura F.4: Log da Blockchain ca transacción de envío de Criptomoedas.

E por último, na Figura F.5 vemos a pantalla principal do paciente.



Figura F.5: Home da app paciente tras ser vacunado por primeira vez.

Proxecto de código aberto

Este apéndice servirá para confirmar o carácter *Open Source* do proxecto, ao especificarse a continuación o *link* ca dirección do **contedor Git** onde poderemos atopar **todo** o código realizado no proxecto, dende aquel referente aos **Smart Contracts** desenvolto en *Solidity* dende *Remix*, até o código *Kotlin* do noso proxecto *Kotlin Multiplatform Mobile* para crear a **app móbil distribuída (Dapp)**. Todo este código estará baixo unha **licenza Open Source (GNU GPLv3)** co fin de que, como ben dixemos no Capítulo 2, poida ser verificado, depurado e mellorado por unha comunidade, eliminando así toda traza de opacidade e contribuíndo á idea principal pola que naceu o proxecto: axudar da mellor maneira posible á sociedade actual e satisfacer as súas necesidades. Por outro lado, daralle ao *Traballo de Final de Grao* unha maior transparencia, ao ensinar todo o traballo feito sen filtros nin limitacións, para que todo aquel que o consulte poida verificar o traballo e as horas dedicadas ao mesmo.

Link do repositorio

`https://git.fic.udc.es/mauro.delossantos/coronapassport-tfg.git`

Relación de Acrónimos

ABI	Contract Application Binary Interface. 62 , 63 , 122
AES	Advanced Encryption Standard. 41 , 61 , 72
API	Application Programming Interfaces. 39 , 40 , 47 , 63
CDC	Centros para o Control e Prevención de Enfermidades. 15 , 19 , 22 , 23 , 25 , 26 , 30 , 32 , 33
COVID-19	Coronavirus Disease 2019. 1–5 , 12–14 , 17 , 18 , 21 , 26 , 37 , 45 , 47 , 73 , 74 , 76
Dapp	Decentralized application. 5 , 10 , 11 , 14 , 18 , 19 , 22 , 31 , 34 , 42 , 47 , 57 , 62 , 72 , 74 , 75 , 78 , 79 , 84 , 85 , 99 , 100 , 125
DLT	Distributed Ledger Techonology. 10 , 35 , 78
EVM	Ethereum Virtual Machine. 41 , 55
IDE	Integrated Development Environment. 40 , 41 , 100 , 101
IoT	Internet of Things. 12
KMM	Kotlin Multiplatform Mobile. 11 , 14 , 39 , 40 , 59 , 74 , 75 , 78 , 84 , 85 , 125
NSA	National Security Agency. 41
OMS	Organización Mundial da Saúde. 1
PCR	Polymerase Chain Reaction. 3
PoW	Proof of Work. 104 , 129
RPC	Remote Procedure Call. 46 , 47 , 53 , 113

SDK Software Development Kit. [39](#)

SO Sistema Operativo. [31](#), [78](#), [99](#)

SPOF Single Point of Failure. [15](#)

Glosario

AES 256 GCM Modo de operación de AES na súa versión de 256 bits. Dito modo GCM, (*Galois/Counter Mode*), dota a AES tanto de confidencialidade, como de autenticidade dos datos (integridade) na mesma operación, sen facer uso de algoritmos adicionais. [42](#)

Blockchain Tecnoloxía distribuída para o almacenamento de información en bloques mediante transaccións dunha forma segura e inmutable. [1](#)

Criptografía Simétrica Tamén chamada criptografía de chave secreta, é un método criptográfico no cal se usa unha mesma chave para cifrar e descifrar mensaxes no emisor e receptor. [14](#)

Faucet Smart Contracts, aplicacións ou sitios web moi utilizados naquelas Blockchains cuxa criptodivisa carece de valor, como nas privadas ou nas *testnets*. A súa finalidade é transferir de forma gratuíta unha porción da criptomoeda da Blockchain á *wallet* do usuario que faga uso del. [18](#), [22](#), [29](#)

GNU GPLv3 A *GNU General Public License version 3* é unha licenza de dereitos de autor amplamente usada no mundo do software libre e código aberto. Garante aos usuarios finais a liberdade de usar, estudar, compartir e modificar o software. [125](#)

nodos mineiros Nodos que resollen complexos problemas criptográficos nun proceso denominado *minería*. Cada nodo mineiro ten como obxectivo ser o primeiro nodo en crear un novo bloque na Blockchain e demostrar que é o que realizou o traballo requirido (de aí o [PoW](#)). Una vez que toda a rede verifica a transacción, engádese un novo bloque á cadea de bloques existente e o nodo mineiro recibe una recompensa. [10](#), [36–38](#), [42](#)

Smart Contract Código que se almacena na Blockchain o cal consiste en funcións ou eventos que permiten aos usuarios interactuar ca cadea de bloques. [11](#)

Bibliografía

- [1] OMS, “COVID-19 Public Health Emergency of International Concern (PHE-IC) Global research and innovation forum,” 2020, consultado o 19 de Maio de 2021. [En liña]. Disponible en: [https://www.who.int/publications/m/item/covid-19-public-health-emergency-of-international-concern-\(pheic\)-global-research-and-innovation-forum](https://www.who.int/publications/m/item/covid-19-public-health-emergency-of-international-concern-(pheic)-global-research-and-innovation-forum)
- [2] “Solicitud de Prueba Diagnóstica de Infección Activa (PDIA) para entrar en España,” *Ministerio de Sanidad*, consultado o 19 de Maio de 2021. [En liña]. Disponible en: <https://www.spth.gob.es/info-pcr>
- [3] “Getting a vaccine certificate for international travel,” *Welsh Government*, consultado o 19 de Maio de 2021. [En liña]. Disponible en: <https://gov.wales/getting-vaccine-certificate-international-travel>
- [4] “Interim Public Health recommendations for fully vaccinated people,” *USA Government*, consultado o 19 de Maio de 2021. [En liña]. Disponible en: <https://www.cdc.gov/coronavirus/2019-ncov/vaccines/fully-vaccinated-guidance.html>
- [5] RTVE, “5.000 personas vibran con Love of Lesbian en un concierto piloto sin distancias en Barcelona,” 2021, consultado o 19 de Maio de 2021. [En liña]. Disponible en: <https://www.rtve.es/noticias/20210327/5000-personas-vibran-love-of-lesbian-concierto-piloto-sin-distancias-barcelona/2083963.shtml>
- [6] “Germany to loosen Covid restrictions for vaccinated people,” *Financial Times*, consultado o 19 de Maio de 2021. [En liña]. Disponible en: <https://www.ft.com/content/86877a16-437a-41a5-bf47-2f2d84219809>
- [7] “Ethical implementation of immunity passports during the COVID-19 pandemic,” *US National Institute of Health*, consultado o 19 de Maio de 2021. [En liña]. Disponible en: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7337820/>

- [8] M. A. Hall and D. M. Studdert, ““Vaccine passport” certification — policy and ethical considerations,” 2021, consultado o 19 de Maio de 2021. [En liña]. Dispoñible en: <https://www.nejm.org/doi/full/10.1056/NEJMp2104289>
- [9] D. Z. Emanuel, “The ethics of ‘vaccine passports’. At All Things Considered, hosted by Ailsa Chang,” 2021, consultado o 19 de Maio de 2021. [En liña]. Dispoñible en: <https://www.npr.org/2021/04/06/984829952/the-ethics-of-vaccine-passports?t=1621425605986>
- [10] “Así será la cartilla de vacunación de todos los españoles,” *La Razón*, consultado o 2 de Marzo de 2021. [En liña]. Dispoñible en: <https://www.larazon.es/salud/20201222/shdsrvh3fjbsrambrsspj43p2e.html>
- [11] “Así es el certificado de vacunación contra el covid de la xunta,” *La Voz de Galicia*, consultado o 21 de Abril de 2021. [En liña]. Dispoñible en: https://www.lavozdeg Galicia.es/noticia/sociedad/2021/04/20/certificado-vacunacion-xunta/00031618916530153673366.htm?utm_source=lavoz&utm_medium=notif_push&utm_campaign=web
- [12] “Coronavirus: Commission proposes a digital green certificate,” *European Commission*, consultado o 19 de Maio de 2021. [En liña]. Dispoñible en: https://ec.europa.eu/commission/presscorner/detail/en/IP_21_1181
- [13] Vxpass, “Vxpass,” 2021, consultado o 16 de Febreiro de 2021. [En liña]. Dispoñible en: <http://www.vxpass.com/>
- [14] “COVID-19 vaccine passports - re-emergence of global mobility?” *Harvey Law Group*, 2021, consultado o 19 de Maio de 2021. [En liña]. Dispoñible en: <https://www.lexology.com/library/detail.aspx?g=636dc7f0-1a6d-4e07-8ef1-c88c62f4a314>
- [15] “In coronavirus fight, China gives citizens a color code, with red flags,” *New York Times*, consultado o 19 de Maio de 2021. [En liña]. Dispoñible en: <https://www.nytimes.com/2020/03/01/business/china-coronavirus-surveillance.html>
- [16] “Green Pass, vaccination certificate and certificate of recovery,” *Ministry of Health of Israel*, 2021, consultado o 19 de Maio de 2021. [En liña]. Dispoñible en: <https://www.gov.il/en/Departments/General/corona-certificates>
- [17] “Coronapas - COVID-19 test,” *European Commission*, 2021, consultado o 19 de Maio de 2021. [En liña]. Dispoñible en: <https://www.sundhed.dk/borger/min-side/corona/covidpas/>

- [18] L. Ricci, D. di Francesco Maesa, A. Favenza, and E. Ferro, "Blockchains for COVID-19 contact tracing and vaccine support: a systematic review," 2021, consultado o 17 de Maio de 2021. [En liña]. Disponible en: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9366740>
- [19] T. F. Caramés, I. F. Míguez, O. B. Nóvoa, and P. F. Lamas, "Enabling the internet of mobile crowdsourcing health things: A mobile fog computing, Blockchain and IoT based continuous glucose monitoring system for diabetes mellitus research and care." 2019, consultado o 21 de Abril de 2021. [En liña]. Disponible en: <https://europepmc.org/article/pmc/6696348>
- [20] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2021, consultado o 19 de Maio de 2021. [En liña]. Disponible en: <https://bitcoin.org/bitcoin.pdf>
- [21] V. Buterin, "Ethereum whitepaper," 2021, consultado o 19 de Maio de 2021. [En liña]. Disponible en: <https://ethereum.org/en/whitepaper/>
- [22] A. Fatrah, S. E. Kafhali, A. Haqiq, and K. Salah, "Proof of concept Blockchain-based voting system," 2019, consultado o 19 de Fevereiro de 2021. [En liña]. Disponible en: https://www.researchgate.net/publication/338450750_Proof_of_Concept_Blockchain-based_Voting_System
- [23] X. Ma, J. Zhou, X. Yang, and G. Li, "A Blockchain voting system based on the feedback mechanism and Wilson score," 2020, consultado o 20 de Fevereiro de 2021. [En liña]. Disponible en: <https://www.mdpi.com/2078-2489/11/12/552/pdf>
- [24] G. G. Dagher, P. B. Marella, M. Milojkovic, and J. Mohler, "Broncovote: Secure voting system using Ethereum's Blockchain," 2018, consultado o 21 de Fevereiro de 2021. [En liña]. Disponible en: <https://www.scitepress.org/Papers/2018/66097/66097.pdf>
- [25] S. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, "Blockchain technology and its relationships to sustainable supply chain management," 2018, consultado o 19 de Maio de 2021. [En liña]. Disponible en: https://farapaper.com/wp-content/uploads/2019/04/Fardapaper-Blockchain-technology-and-its-relationships-to-sustainable-supply_chain_management.pdf
- [26] F. Bitterli, "Building a private Ethereum Blockchain in a box," 2020, consultado o 15 de Marzo de 2021. [En liña]. Disponible en: https://wwz.unibas.ch/fileadmin/user_upload/wwz/00_Professuren/Schaer_DLTFintech/Lehre/Bitterli_2020.pdf
- [27] M. Sutanto, "Different approaches in consuming KMM modules in iOS," 2019, consultado o 19 de Maio de 2021.

- [En línea]. Disponible en: <https://medium.com/wantedly-engineering/different-approaches-in-consuming-kmm-modules-in-ios-7957c722b114>
- [28] M. de los Santos Nodar, "Paradigmas de desarrollo Android - iOS," 2021, consultado o 28 de Maio de 2021. [En línea]. Disponible en: <https://mauropi.ddns.net/docs/PARADIGMAS.pdf>
- [29] "KMM documentation," *Kotlin Lang*, consultado o 16 de Maio de 2021. [En línea]. Disponible en: <https://kotlinlang.org/docs/mobile/home.html>
- [30] B. Yong, J. Shen, X. Liu, F. Li, H. Chen, and Q. Zhou, "An intelligent blockchain-based system for safe vaccine supply and supervision," 2019, consultado o 2 de Marzo de 2021. [En línea]. Disponible en: <https://sci-hub.st/https://www.sciencedirect.com/science/article/abs/pii/S0268401219304505>
- [31] "Introducing vechain's drug and vaccine traceability solution," *Vechain Foundation*, consultado o 16 de Febreiro de 2021. [En línea]. Disponible en: <https://vechainofficial.medium.com/introducing-vechains-drug-and-vaccine-traceability-solution-c17fb869e003>
- [32] NFhbar, "Ethereum-medical-records," 2017, consultado o 23 de Marzo de 2021. [En línea]. Disponible en: <https://github.com/NFhbar/Ethereum-Medical-Records/tree/master/contracts>
- [33] C. M. Angelopoulos and V. Katos, "Dhp framework: Digital health passports using blockchain—use case on international tourism during the COVID-19 pandemic," 2020, consultado o 18 de Maio de 2021. [En línea]. Disponible en: <https://arxiv.org/abs/2005.08922>
- [34] A. Bansal, C. Garg, and R. P. Padappayil, *Optimizing the implementation of COVID-19 'immunity certificates' using blockchain*, 2020.
- [35] "Certificado COVID-19 digital de la UE," *Comisión Europea*, 2020, consultado o 14 de Xuño de 2021. [En línea]. Disponible en: https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/safe-covid-19-vaccines-europeans/eu-digital-covid-certificate_es#qu-es-el-certificado-covid-digital-de-la-ue
- [36] "Ethereum testnet faucets," consultado o 25 de Maio de 2021. [En línea]. Disponible en: <https://ethereum.org/en/developers/docs/networks/#testnet-faucets>
- [37] "Ropsten faucet," consultado o 25 de Maio de 2021. [En línea]. Disponible en: <https://faucet.ropsten.be/>

- [38] “Rinkeby faucet,” consultado o 25 de Maio de 2021. [En línea]. Disponible en: <https://faucet.rinkeby.io/>
- [39] “COVID-19 vaccination record card,” *USA Government*, 2021, consultado o 2 de Marzo de 2021. [En línea]. Disponible en: https://www.tn.gov/content/dam/tn/health/documents/cedep/novel-coronavirus/Shot_Card_Spanish.pdf
- [40] “How does a transaction get into the blockchain?” *EuroMoney Learning*, 2020, consultado o 28 de Maio de 2021. [En línea]. Disponible en: <https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain>
- [41] G. Hartmann, G. Stead, and A. DeGani, “Cross-platform mobile development,” 2011, consultado o 28 de Maio de 2021. [En línea]. Disponible en: <https://wss.apan.org/jko/mole/Shared%20Documents/Cross-Platform%20Mobile%20Development.pdf>
- [42] A. Biørn-Hansen, T.-M. Grønli, G. Ghinea, and S. Alouneh, “An empirical study of cross-platform mobile development in industry,” 2019, consultado o 28 de Maio de 2021. [En línea]. Disponible en: <https://downloads.hindawi.com/journals/wcmc/2019/5743892.pdf>
- [43] E. Hjort, “Evaluation of React Native and Flutter for cross-platform mobile application development,” 2020, consultado o 28 de Maio de 2021. [En línea]. Disponible en: https://www.doria.fi/bitstream/handle/10024/180002/hjort_elin.pdf?sequence=2&isAllowed=y
- [44] “AES,” consultado o 24 de Maio de 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Advanced_Encryption_Standard
- [45] “Solidity,” consultado o 20 de Maio de 2021. [En línea]. Disponible en: <https://solidity-es.readthedocs.io/es/latest/>
- [46] “Geth,” consultado o 20 de Maio de 2021. [En línea]. Disponible en: <https://geth.ethereum.org/>
- [47] “Remix,” consultado o 20 de Maio de 2021. [En línea]. Disponible en: <https://remix-project.org/>
- [48] “Metamask,” consultado o 20 de Maio de 2021. [En línea]. Disponible en: <https://metamask.io/>
- [49] “Firebase,” consultado o 20 de Maio de 2021. [En línea]. Disponible en: <https://firebase.google.com/?hl=es>

- [50] “Shared Preferences,” consultado o 24 de Maio de 2021. [En liña]. Disponible en: <https://developer.android.com/reference/android/content/SharedPreferences>
- [51] “Encrypted Shared Preferences,” consultado o 24 de Maio de 2021. [En liña]. Disponible en: <https://developer.android.com/topic/security/data>
- [52] “Web3j,” consultado o 20 de Maio de 2021. [En liña]. Disponible en: <http://docs.web3j.io/latest/>
- [53] “How do I set up my own Ethereum Testnet?” *CoinBundle Team*, 2018, consultado o 8 de Marzo de 2021. [En liña]. Disponible en: <https://medium.com/coinbundle/how-do-i-set-up-my-own-ethereum-testnet-cebab790c696>
- [54] “How to: Create your own private Ethereum Blockchain,” *Mercury Protocol*, 2017, consultado o 15 de Marzo de 2021. [En liña]. Disponible en: <https://medium.com/mercuryprotocol/how-to-create-your-own-private-ethereum-blockchain-dad6af82fc9f>
- [55] P. Humiston, “Build your first Ethereum Smart Contract with Solidity - tutorial,” 2017, consultado o 16 de Marzo de 2021. [En liña]. Disponible en: <https://codeburst.io/build-your-first-ethereum-smart-contract-with-solidity-tutorial-94171d6b1c4b>
- [56] Niskmac, “Genesis Block explanation,” 2016, consultado o 31 de Maio de 2021. [En liña]. Disponible en: <https://ethereum.stackexchange.com/questions/2376/what-does-each-genesis-json-parameter-mean>
- [57] Skozin, “Ethereum dev network,” 2017, consultado o 22 de Marzo de 2021. [En liña]. Disponible en: <https://github.com/skozin/ethereum-dev-net>
- [58] Loom, “Solidity tutorial and Ethereum Blockchain programming course,” 2021, consultado o 18 de Febreiro de 2021. [En liña]. Disponible en: <https://cryptozombies.io/en/course>
- [59] B. Mmopelwa, “How to implement Ethereum blockchain in Android using Web3j,” 2020, consultado o 18 de Abril de 2021. [En liña]. Disponible en: <https://boemo1mmopelwa.medium.com/implementing-ethereum-blockchain-in-android-with-web3j-485ea0747088>
- [60] E. Ackon, “Communicating with an Ethereum Smart contract via Android,” 2021, consultado o 20 de Abril de 2021. [En liña]. Disponible en: <https://medium.com/interfacing-with-a-blockchain/communicating-with-an-ethereum-smart-contract-via-android-24ee0dd2c115>

- [61] E. Gacoki, "Firebase email and password authentication in Android using Kotlin," 2021, consultado o 28 de Abril de 2021. [En línea]. Disponible en: <https://www.section.io/engineering-education/firebase-email-and-password-authentication-in-android-using-kotlin/>
- [62] S. Kargopolov, "Firebase authentication example in Kotlin," 2021, consultado o 28 de Abril de 2021. [En línea]. Disponible en: <https://www.appsdeveloperblog.com/firebase-authentication-example-kotlin/>
- [63] N. Hour, "Building a simple Android QR code reader," 2020, consultado o 5 de Abril de 2021. [En línea]. Disponible en: <https://www.nighthour.sg/articles/2020/building-a-simple-android-qr-code-reader.html>
- [64] U. Adil, "Encrypting images using AES in Android with Kotlin + Glide + RxJava2," 2019, consultado o 13 de Mayo de 2021. [En línea]. Disponible en: <https://www.linkedin.com/pulse/encrypting-images-using-aes-android-kotlin-glide-rxjava2-umair-adil/>
- [65] M. de los Santos Nodar, "How to get Ether in private Blockchain?" consultado o 14 de Maio de 2021. [En línea]. Disponible en: <https://ethereum.stackexchange.com/questions/98046/how-to-get-ether-in-private-blockchain>
- [66] "Binance," consultado o 25 de Maio de 2021. [En línea]. Disponible en: <https://www.binance.com/es>
- [67] "Windows 10," consultado o 20 de Maio de 2021. [En línea]. Disponible en: <https://www.microsoft.com/es-es/software-download/windows10>
- [68] "Git," consultado o 20 de Maio de 2021. [En línea]. Disponible en: <https://git-scm.com/>
- [69] "Git Bash," consultado o 24 de Maio de 2021. [En línea]. Disponible en: <https://gitforwindows.org/>
- [70] "Yed," consultado o 20 de Maio de 2021. [En línea]. Disponible en: <https://www.yworks.com/products/yed>
- [71] "Draw.io," consultado o 20 de Maio de 2021. [En línea]. Disponible en: <https://app.diagrams.net/>
- [72] "Ethereum," consultado o 20 de Maio de 2021. [En línea]. Disponible en: <https://ethereum.org/en/>
- [73] "Adobe XD," consultado o 20 de Maio de 2021. [En línea]. Disponible en: <https://www.adobe.com/es/products/xd.html>

- [74] “Android Studio,” consultado o 20 de Maio de 2021. [En línea]. Disponible en:
<https://developer.android.com/studio>
- [75] “Geth releases,” consultado o 31 de Maio de 2021. [En línea]. Disponible en:
<https://geth.ethereum.org/downloads/>