DApp pel control i gestió de la temperatura de medicaments termolàbils

Josep Genovard Oliver

Sistemes de blockchain

Màster en Ciberseguretat i Privadesa

Nom del Tutor/a del TF:
Friman Sanchez Castaño

Nom del de/la PRA:
Victor Garcia Font

9 de gener de 2024

Universitat Oberta de Catalunya

Fitxa del Treball Final

Títol del treball:	DApp pel control i gestió de la temperatura de medicaments termolàbils
Nom de l'autor/a:	Josep Genovard Oliver
Nom del Tutor/a del TF:	Friman Sanchez Castaño
Nom del de/la PRA:	Victor Garcia Font
Data de lliurament:	9 de gener de 2024
Titulació o programa:	Màster en Ciberseguretat i Privadesa
Àrea del treball final:	Sistemes de blockchain
Idioma del treball:	Català
Paraules clau:	Blockchain, DApp, medicaments termolàbils

Resum del treball

Els medicaments termolàbils, els que són sensibles a les fluctuacions de temperatura, poden perdre eficàcia si es conserven en males condicions, i augmentar el nombre d'afectes secundaris. A causa de les evidències existents de la mala conservació dels medicaments, en aquest treball es dissenya i implementa un sistema pel control i la gestió descentralitzada de medicaments termolàbils. Partint de la idea d'un projecte en desenvolupament, que registra paràmetres com la temperatura, aquest treball proposa una descentralització del sistema mitjançant tecnologia blockchain, amb l'objectiu de millorar la integritat i traçabilitat de les dades, i afegir una millor participació de l'usuari. En el document s'explica que la proposta distingeix diferents actors com farmacèutics, usuaris i sensors, els quals poden executar les funcions necessàries per al bon funcionament del sistema. Mentre que els farmacèutics podran registrar nous usuaris o medicaments al sistema, i assignar quin sensor controlarà un medicament, els sensors només podran registrar temperatures periòdicament i els usuaris consultar certes dades. Quan un medicament deixi de ser subministrable, s'enviarà una notificació a l'usuari corresponent.

Abstract

Thermolabile drugs, those sensitive to temperature fluctuations, may lose effectiveness if stored improperly, leading to an increase in side effects. Due to existing evidence of inadequate drug storage, this work designs and implements a system for the decentralized control and management of thermolabile drugs. Stemming from the concept of an ongoing project that records parameters such as temperature, this study proposes a decentralization of the system using blockchain technology, aiming to enhance data integrity, traceability, and user engagement. The document explains that the proposal distinguishes between various actors such as pharmacists, users, and sensors, each responsible for specific functions in the system's proper operation. Pharmacists can register new users or drugs in the system, and assign which sensor will monitor a drug, while sensors can only periodically record temperatures, and users can access specific data. Notifications will be sent to the respective user when a drug becomes non-dispensable.

Índex

Acrònims

DApp Decentralized Application. 12–14, 25

GDP Good Distribution Practice. 15

IPFS InterPlanetary File System. 20

IUM Identificador Únic de Medicament. 27, 44, 46

SC Smart Contract. 7, 10, 16, 20, 21, 23, 24, 27, 28, 30, 32, 34, 35, 37–42, 46, 48

TFM Treball de Fi de Màster. 16

WBAN Wireless Body Area Network. 16

WHO World Health Organization. 15

Capítol 1

Introducció

1.1 Context i justificació del treball

Els medicaments tenen un paper fonamental en la vida quotidiana, ja que contribueixen a millorar la salut i el benestar de les persones. La seva capacitat de prevenir la propagació de malalties, tractar malalties agudes i atendre pacients crònics fa que els medicaments siguin instruments essencials en l'atenció mèdica d'avui en dia. Aquests, poden alleujar símptomes, controlar el dolor, reduir la inflamació, curar infeccions i proporcionar tractaments per a innumerables afeccions. En molts casos, els medicaments permeten a les persones viure més temps, de manera més saludable i ser més productives.

En concret, els medicaments termolàbils són fàrmacs sensibles, principalment, a les fluctuacions de temperatura i poden perdre la seva eficàcia si s'esposen a temperatures fora del seu rang de conservació específic. A més, els medicaments termolàbils també poden ser sensibles a altres factors com humitat, llum i vibració. Per tant, aquests fàrmacs requereixen condicions adequades de transport i emmagatzematge per garantir que mantinguin la seva seguretat i eficàcia al llarg de la seva vida útil.

Segons les regulacions actuals, la cadena de fred ha de ser controlada durant el transport des de la fàbrica fins a l'hospital, així com durant l'emmagatzematge a les instal·lacions del mateix hospital. Per tant, quan els medicaments són transportats, emmagatzemats i dispensats al domicili del pacient, no es regulen els paràmetres i, així com ja s'ha demostrat en anteriors articles [1], els medicaments no estan ben conservats en aquests casos. De fet, existeixen estudis relacionats que arriben a la conclusió que els refrigeradors domèstics no són vàlids per a la conservació dels medicaments termolàbils [2].

El projecte QChainMED, de l'Institut d'Investigació Sanitària de les Illes Balears (IdISBa) i la Universitat de les Illes Balears (UIB), té com a objectiu el registre de diversos paràmetres de medicaments termolàbils que han de ser emmagatzemats a la nevera dels pacients que els prenen. Aquests paràmetres inclouen la temperatura, la humitat, el moviment i l'acceleració, encara que es presta una atenció principal a la temperatura. Es cerca gestionar i controlar aquesta temperatura mitjançant alertes automàtiques dirigides als pacients.

Al projecte mencionat, s'ha dissenyat un sensor que recull aquestes dades i les transmet mitjançant la tecnologia Bluetooth a un dispositiu proper, que actua com a pont amb el servidor. Aquest darrer dispositiu envia paquets del resum de dades, mitjançant les tecnologies LoRa o WiFi, al servidor. A més, està dissenyat per emetre alarmes audibles i visuals si la temperatura

es desvia del rang desitjat. Quan les dades han arribat al servidor, aquest les guarda i, si cal, envia notificacions a la plataforma d'usuari. La plataforma d'usuari permet registrar pacients, sensors i gateways, i assignar gateways a pacients o sensors a gateways segons sigui necessari [3].

No obstant això, és important destacar que aquest projecte té l'inconvenient de ser totalment centralitzat; un sol servidor guarda i gestiona totes les dades. El propòsit d'aquest treball de fi de màster és abordar aquesta centralització i descentralitzar el sistema, aconseguint una major integritat i traçabilitat de les dades. Per això, es partirà de la idea que ja es disposa de sensors que, periòdicament, llegeixen i envien la temperatura a un Smart Contract (SC) de la blockchain. Aprofitant això, també es permetrà que els usuaris puguin visualitzar les dades dels seus propis medicaments, donant més eines per millorar la conservació d'aquests.

1.2 Objectius del treball

Aquest treball té l'objectiu d'adaptar la proposta del projecte QChainMED [3] a un entorn descentralitzat. Es partirà des del supòsit que ja es disposa dels sensors amb capacitat de lectura de temperatura i que la guarden periòdicament al contracte de la blockchain que es crearà. Per tant, el projecte es focalitzarà amb el desenvolupament d'un conjunt d'SCs per a l'emmagatzematge i la gestió de les dades de temperatura dels medicaments. Hi haurà diferents tipus d'actors: farmacèutics, usuaris, sensors i l'autoritat, amb funcionalitats específiques per cada un d'ells. El contracte ha de permetre no només registrar les dades, sinó també gestionar als mateixos actors i permetre realitzar-los les funcions.

Principalment, els farmacèutics i sensors seran registrats per una autoritat principal, els farmacèutics han de poder registrar usuaris i associacions amb els sensors, i els usuaris podran visualitzar el registre de paràmetres dels seus propis fàrmacs. Com es suposat, els sensors podran registrar les temperatures i, a partir d'aquestes, es podran crear alertes per avisar als usuaris.

El llistat d'objectius és el següent:

- Estudi de l'estat de l'art de sistemes de gestió de les dades de medicaments, o similars.
- Disseny del protocol de gestió de les dades que es vol implementar, segons els requisits definits.
- Implementació d'un sistema que permeti gestionar els sensors i les seves dades.
- Anàlisi del cost de la implementació i viabilitat.

1.3 Impacte en sostenibilitat, ètic-social i de diversitat

Aquesta secció reflecteix els impactes positius que el projecte pot aportar a la societat actual, fent referència a les tres dimensions de la competència transversal UOC "Compromís ètic i global".

1.3.1 Dimensió de sostenibilitat

El primer aspecte, la sostenibilitat, és el més discutible per aquest projecte; fet que radica en l'ús de la blockchain. Avui en dia, és perfectament conegut que aquesta tecnologia pot tenir un impacte ecològic molt negatiu, però hi ha estratègies per reduir la petjada que causa.

La primera blockchain va ser Bitcoin, que va utilitzar un protocol de consens conegut com a *Proof of Work*. Aquest protocol, tot i proporcionar una alta seguretat descentralitzada, requereix una quantitat significativa d'energia per validar transaccions i minar nous blocs a la xarxa. Aquesta alta demanda energètica és el resultat d'una competició per ser el primer a resoldre un complicat càlcul, i com més ràpids siguin els miners, més difícil esdevé la tasca, augmentant així el consum energètic.

La majoria de les altres plataformes varen utilitzar el mateix protocol, però gràcies a una creixent consciència sobre el canvi climàtic, s'han cercat alternatives més sostenibles a aquest protocol, i moltes altres xarxes blockchain, com Ethereum, han optat per canviar a protocols com *Proof of Stake* amb l'objectiu de reduir el seu impacte ecològic i fomentar la sostenibilitat.

Aquest treball té per objectiu la sostenibilitat pel que, a l'hora d'implementar el protocol, es prioritzaria un entorn amb poca petjada; s'utilitzaran xarxes sostenibles com: Ethereum 2.0.

1.3.2 Dimensió de comportament ètic i de responsabilitat social

El treball té un clar comportament ètic, pretén oferir una eina per millorar la conservació dels medicaments que es tradueix en una millor salut dels pacients.

Tot i que ja s'ha abordat la idea de controlar els paràmetres crítics relacionats amb la salut dels pacients i la conservació dels medicaments en altres projectes similars [3], aquest treball va més enllà de la simple recopilació de dades. En lloc de limitar-se al registre de dades, aquest projecte ofereix serveis addicionals als usuaris, col·locant-los al centre del procés.

La proposta feta vol oferir un sistema transparent en la informació. Els usuaris tindran accés a les seves pròpies dades i es podrà verificar fàcilment la validesa de la informació. Aquestes propietats, com la transparència i la integritat, es basen en els principis fonamentals de la tecnologia blockchain. A més, a la idea de *registre i gestió* de dades, s'afegeix una important dimensió ètica, la participació activa de l'usuari, ja que, tenint ara accés a les dades, aquests tindran un paper essencial en la supervisió i la millora de la conservació dels seus propis medicaments.

Aquest enfocament reflecteix un compromís amb els valors ètics i la responsabilitat social, així com una preocupació pel benestar dels pacients i la millora de la salut. El projecte no només es tradueix en una millor conservació dels medicaments, sinó que també en l'empoderament dels pacients i fomenta la seva participació activa en el procés de cura de la salut.

1.3.3 Dimensió de diversitat, gènere i drets humans

La dimensió de diversitat també es té en compte pel projecte. L'actual document presenta un protocol amb l'objectiu de donar eines a tots els usuaris que necessiten guardar medicaments amb requisits de temperatura crítics.

La proposta està pensada per afegir-se en el sistema hospitalari públic, on els farmacèutics seran els encarregats de gestionar totes les dades. Per tant, quan els pacients vagin a recollir el medicament, aquest ja durà un sensor incorporat. Amb aquest sistema, tots els pacients de

l'hospital podrien gaudir d'aquest servei, amb independència del gènere, classe social, creences, origen, etc.

Hi hauria una altra possibilitat de funcionament del sistema i és que les persones que ho necessitin puguin adquirir-lo externament. Això fomentaria la separació de classes socials, ja que només podrien permetre-s'ho les persones amb major nivell adquisitiu. Per això, no es té previst aquest cas d'ús.

En resum, es poden mencionar les següents propietats del projecte, referent a la diversitat:

- **Accés i equitat**: És essencial assegurar que l'accés al sistema i als seus beneficis sigui igualitari per a tothom, sense importar el gènere, l'origen ètnic o altres característiques personals. S'ha de garantir que tothom tingui la mateixa oportunitat de gestionar les seves dades de salut i beneficiar-se del control de temperatura dels medicaments.
- **Diversitat i inclusió**: En el desenvolupament i la implementació del projecte, s'han de tenir en compte les diferents necessitats i contextos dels usuaris. S'ha de treballar perquè les persones amb pocs recursos puguin aprofitar, tant com les altres, el sistema proposat.
- **Drets humans**: El projecte respecta i promou els drets humans, com el dret a la salut. A més, per assegurar el dret a la privacitat de les dades, es reservaran les dades privades com noms o identificadors personals. Si es volguessin utilitzar, s'hauria d'utilitzar el xifratge de les dades.
- **Participació activa**: En el treball s'ha considerat important el fet d'involucrar als usuaris en el procés de conservació dels seus propis medicaments, i per això es permet l'accés a totes les seves dades.

L'aplicació d'aquestes consideracions a la dimensió de diversitat, gènere i drets humans assegurarà que el treball de fi de màster no només millora la conservació dels medicaments, sinó que també contribueixi a la construcció d'una societat més justa, equitativa i inclusiva en l'àmbit de la salut.

1.4 Enfocament i mètode seguit

Tot i partir de la idea d'un projecte que ja està en desenvolupament, aquest treball té com a objectiu crear un producte nou. Això és degut al fet que la descentralització de la plataforma impedeix la reutilització del codi existent. A més, els diferents codis i l'abast diferent del projecte fan que aquesta sigui una tasca complexa i amb reptes importants.

El desenvolupament de la Decentralized Application (DApp) seguirà una metodologia àgil. Aquesta elecció es basa en la comprensió que la primera versió del producte pot no ser la definitiva i que, segurament, s'hauran de realitzar canvis per perfeccionar-la. La metodologia àgil implica que el producte es desenvolupi en petites etapes successives per, si la versió final no és l'esperada, tornar a començar el procés. En cada iteració es poden fer millores, correccions i adaptacions basades en els errors o requisits emergents. Aquesta flexibilitat és fonamental en un projecte com aquest, on l'objectiu és crear una plataforma descentralitzada única i efectiva que satisfaci les necessitats dels usuaris i garanteixi la seguretat de les dades i les transaccions.

1.5 Planificació del treball

Els recursos necessaris per a aquesta pràctica són mínims. La documentació es realitza utilitzant l'editor de Làtex *Overleaf*; per la creació de diagrames, s'usarà l'eina *Draw.io*, i per desenvolupar i provar el codi, es farà servir *Remix* juntament amb *Visual Studio Code (VSC)*. Les tres primeres eines es troben disponibles en línia i són gratuïtes, mentre que la darrera és d'accés obert.

Aquest treball se centra principalment en la codificació del *backend*, la part lògica del sistema. Així i tot, l'estructura estarà orientada a facilitar una posterior interfície d'usuari (*frontend*). Després, s'hauria de desplegar la DApp en un entorn de prova, com podria ser una de les xarxes de proves d'Ethereum, i finalment decidir la cadena a utilitzar.

En resum, els materials necessaris són accessibles per a qualsevol persona amb un ordinador amb connexió a Internet.

Respecte a la planificació de cada una de les dates, les feines de desenvolupar codi sempre són complicades de planificar. Això és degut a que a l'hora de fer el codi poden sorgir errors mals de resoldre o pot ser difícil implementar la funcionalitat desitjada, sobretot quan un no és expert. Per això, en aquest projecte s'han aproximat les dates de la Fig. 1.1. La imatge, per poder-se veure en major detall, també es troba disponible a [4].

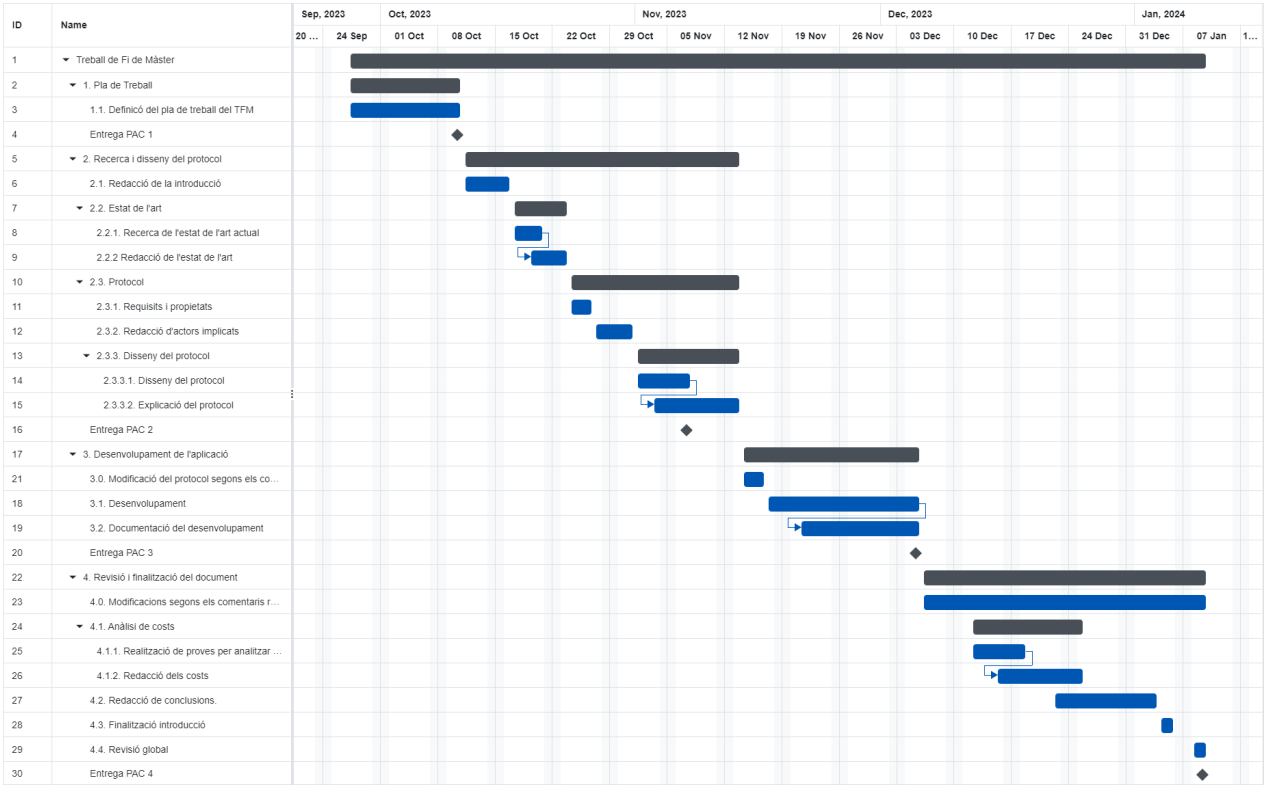

Figura 1.1: Diagrama de Gantt de la distribució de tasques.

1.6 Breu sumari de productes obtinguts

El producte obtingut és un sistema per al control i la gestió de medicaments termolàbils.

Aquest projecte permet als usuaris monitoritzar i registrar els paràmetres crítics de temperatura dels seus medicaments, assegurant-ne la correcta conservació. La DApp està dissenyada amb una metodologia àgil que permet adaptar-la a les necessitats dels usuaris i garantir la seguretat de les dades. A més, busca promoure la transparència i la participació activa dels usuaris en la gestió de la seva pròpia salut. El producte es basa en tecnologia blockchain per a la immutabilitat i la integritat de les dades, oferint un enfocament innovador per millorar la conservació de medicaments i empoderar els pacients.

1.7 Breu descripció dels altres capítols de la memòria

En els següents capítols s'anirà desenvolupament cadascun dels objectius del projecte. Es començarà analitzant l'estat de l'art de la gestió de medicaments termolàbils i els projectes de medicina descentralitzada; es farà una proposta de disseny, començant amb una visió general i seguint amb una explicació molt detallada; es mostrarà i explicarà la implementació feta, basada en el disseny i utilitzant el llenguatge *Solidity*; s'analitzarà el cost i la viabilitat del projecte, i s'acabarà fent una conclusió general, incloent possibles futures millores.

Capítol 2

Estat de l'art

Els darrers anys s'han començat a observar els avantatges que aporta la tecnologia blockchain, o cadena de blocs, entre els quals destaquen la integritat, la transparència i el no rebuig en origen i destí. S'ha pres consciència que aquesta tecnologia pot oferir millores que van més enllà de les àrees inicialment considerades, com el sector financer. De fet, el sector de la salut és un dels que està adquirint importància, a causa de la seva rellevància en la societat.

En el mateix sector, la World Health Organization (WHO), publica informes periòdics sobre qüestions de salut globals, aporta recomanacions o directrius, i en general, s'encarrega de promoure la salut en l'àmbit mundial. En concret, en relació amb aquest treball, la WHO contribueix al desenvolupament d'un suplement tècnic [5] anomenat Good Distribution Practice (GDP) [6] que pretén descriure els requisits mínims necessaris en la gestió de l'emmagatzematge i el transport de productes farmacèutics sensibles a la temperatura, assegurant la qualitat i la integritat dels fàrmacs a la cadena de transport, a més de mantenir-ne l'eficàcia.

A l'informe [7], del *WHO Technical Report Series*, es detalla com hauria de ser el monitoratge d'aquests medicaments en àrees d'emmagatzematge fixes, com ara hospitals, laboratoris o farmàcies. Entre els requisits hi ha la definició d'un sistema de monitoratge sense fil i automàtic, amb un sistema d'alarmes, escalable i segur. Aquest treball aconsegueix el compliment d'aquests requisits fins a l'usuari final. És a dir, s'obtindrà un sistema de monitoratge automàtic de temperatura a casa dels pacients, reduint i controlant les possibles desviacions de les recomanacions d'emmagatzematge, derivades de l'ús d'una nevera no pensada per emmagatzemar medicaments.

El sistema proposat utilitza la tecnologia de blockchain per assegurar la integritat de les dades. A més, aquest és un sistema a temps real per controlar les dades dels fàrmacs, possibilitant l'ús de gràfics gràcies a la marca de temps de les dades, juntament amb la definició d'alarmes, o notificacions, adaptades a les especificacions de cada medicament. Finalment, cal mencionar que el sistema suggerit en aquest treball també és escalable i adaptable, ja que es podrien monitorar altres paràmetres, com el sacseig o la humitat, amb canvis senzills a l'arquitectura.

Sense anar més enllà, durant el passat curs 22/23, l'alumne Paula Otero va realitzar el treball final, titulat *Trazabilidad de las cadenas de frío de vacunas en Ethereum* [8], com a part del Màster Universitari en Ciberseguretat i Privadesa, a la UOC. Aquest treball es va veure motivat per la importància de les vacunes contra la COVID-19 i en ell es proposa un protocol per fer el seguiment del transport de medicaments, fent ús de la immutabilitat, la transparència i, per tant, la traçabilitat que proporciona la tecnologia blockchain per si mateixa.

El document descriu una solució interessant, tot i que senzilla. Per altra banda, l'objectiu en el present document va més enllà del registre del transport de lots; es pretén gestionar l'estat de conservació en tot moment, a través del monitoratge de la temperatura. Aquesta gestió no es durà a terme durant tot el procés, ja que la cadena de fred està regulada des de la fàbrica fins a l'hospital [9]. En canvi, la gestió començarà en el moment en què el medicament surti de l'hospital, quan els fàrmacs són realment vulnerables.

En l'àmbit de la gestió remota i descentralitzada, també hi ha feina feta. Per exemple, l'article [10] aborda el creixent ús de Wireless Body Area Network (WBAN) per al monitoratge remot de pacients a través de dispositius portàtils, destacant la seva naturalesa descentralitzada. La seva proposta, mostrada en la Fig. 2.1, se centra en la integració de sistemes WBAN amb SCs en una blockchain de consorci, per proporcionar un servei de processament de dades distribuït que manté la privadesa dels pacients i redueix la necessitat d'intermediaris centralitzats. Precisament, del tema de la privadesa ja se n'ha parlat al capítol d'introducció 1 del present document. Per tant, el protocol desenvolupat en aquest Treball de Fi de Màster (TFM) es podria executar en un entorn privat si les dades que s'han de publicar es consideren confidencials.

La proposta [10] permet notificacions en temps real sobre esdeveniments de salut i promou la descentralització en la gestió de dades de salut; les dades recollides es registrarien periòdicament a la blockchain, basada en el protocol Ethereum. Així i tot, en la proposa de Griggs, es genera una alerta al metge i a l'hospital, quan es recullen dades crítiques, i s'augmenta el preu del gas per accelerar la finalització de la transacció. Per altra banda, en el protocol de gestió de fàrmacs, d'aquest treball, no es considera necessari aquest augment de gas en cap moment, ja que les variacions de temperatura són progressives. Així i tot, per evitar els costos associats a les transaccions, i donada la gran quantitat de dades que s'arriben a recopilades, és ineficient emmagatzemar totes les mesures a la xarxa de blocs. Llavors, el protocol dissenyat només emmagatzemarà les dades de temperatura quan aquestes es trobin fora del rang desitjat.

Per acabar, cal mencionar que ja existeixen diversos treballs que usen la tecnologia blockchain al sector farmacèutic, centrant-se en altres aspectes importants de la medicina. Entre ells, la *start-up* nord-americana BlockMedx [11] presentava una solució electrònica per a les receptes mèdiques, aconseguint un seguiment més exhaustiu i una major seguretat. Tot i això, entre els treballs estudiats cap es basa en la conservació eficaç dels medicaments a casa dels pacients. Aquest factor és important, ja que el descontrol dels paràmetres d'aquests medicaments afecta directament la salut i el benestar dels pacients.

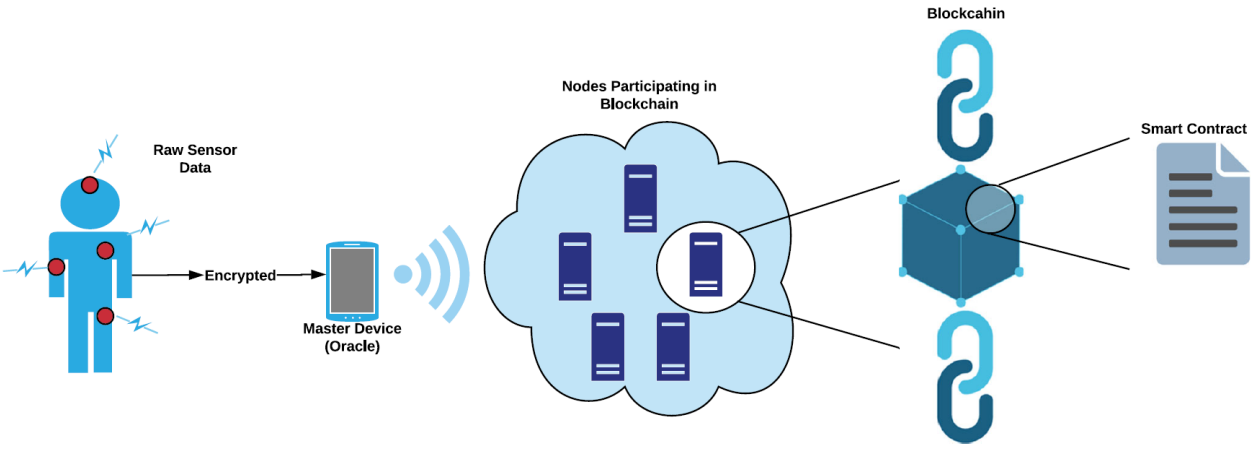

Figura 2.1: Funcionament del sistema que proposa l'article [10].

Capítol 3

Protocol dissenyat

Les seccions 1.1 i 1.2 ja introdueixen la proposta de disseny d'aquest treball. Es vol crear un sistema descentralitzat que permeti gestionar la conservació dels medicaments termolàbils. A més, aquest sistema no només ha de permetre guardar els paràmetres dels fàrmacs, sinó que també ha de permetre que els diferents actors, sobretot farmacèutics i usuaris, puguin ser gestionats i executar les funcions desitjades.

3.1 Requisits i propietats

Una de les passes prèvies més importants per fer un protocol és establir els requisits i les propietats que s'han d'assolir. Així, en aquest treball, alguns requisits són necessaris per satisfer la funcionalitat bàsica que es proposa, però d'altres han sortit de l'anàlisi de l'estat de l'art actual, a la secció 2.

A continuació es mostra un llistat de les propietats de seguretat que el protocol ha de complir. Cada una d'elles està acompanyada d'un requisit del sistema.

- **Autenticació.**

  Propietat: L'autenticitat és la propietat que assegura que les entitats digitals són genuïnes i representen amb precisió la seva identitat reclamada.

  Requeriment: Tots els participants del protocol han de ser autenticats. Això inclou als farmacèutics i als usuaris, però també al sensor que enviï temperatures, cap actor extern ha de ser poder modificar una acció en nom d'un actor legal.

- **Integritat de les dades.**

  Propietat: La integritat de les dades es refereix a l'exactitud, coherència i fiabilitat de la informació emmagatzemada i processada en un sistema o base de dades. Garanteix que les dades romanen intactes i sense canvis durant els processos d'emmagatzematge, recuperació i transmissió.

  Requeriment: La integritat de les dades és fonamental en un sistema perquè garanteix que la informació sigui de confiança i fiable. Totes les dades generades pels diferents participants del protocol han de ser rebudes i emmagatzemades sense cap alteració. Llavors, les modificacions deshonestes de les dades han de ser impossibles.

- **Confidencialitat.**

  Propietat: La confidencialitat es refereix a la protecció de la informació delicada o privada contra l'accés no autoritzat, la divulgació o l'exposició. Garanteix que només les persones o entitats autoritzades tinguin accés a les dades i impedeix que parts no autoritzades obtinguin o utilitzin la informació amb fins no autoritzats.

  Requeriment: La confidencialitat és especialment important per a les dades delicades com les relacionades amb la salut. Protegint la confidencialitat, les organitzacions i les persones poden mantenir la confiança, protegir la privadesa i reduir els riscos associats a l'accés no autoritzat o la divulgació de dades confidencials. En aquest sistema, les dades recopilades no contenen dades confidencials sobre la salut dels pacients, sinó que s'emmagatzemaran de manera anònima i només els farmacèutics podran saber qui es troba darrere cada cartera, si disposen d'un registre extern.

- **Marcatge de temps.**

  Propietat: El marcatge de temps és el procés d'enregistrament de manera segura de la data i l'hora en què ocorre un esdeveniment o transacció en particular. Proporciona un mecanisme fiable i a prova de manipulacions per establir l'ordre cronològic i la integritat dels esdeveniments o dades.

  Requeriment: A l'hora de registrar de paràmetres, es registrarà els temps. Així, serà més fàcil fer un seguiment de les dades i veure en quin moment hi ha hagut incidències, i fer un futur anàlisi per detectar períodes més vulnerables.

- **Immutabilitat de les dades.**

  Propietat: La immutabilitat de les dades fa referència a la propietat de les dades que no poden canviar o alterar-se una vegada que s'han creat o emmagatzemat. Garanteix que una vegada que les dades s'han escrit, no es poden modificar, manipular o eliminar sense deixar rastre o registre dels canvis.

  Requeriment: Quan un sistema d'emmagatzematge proporciona immutabilitat de les dades, aquestes no es poden eliminar ni modificar. Aquesta propietat és diferent de la integritat de les dades, ja que la immutabilitat garanteix l'emmagatzematge permanent de les dades. L'ús de la blockchain en el sistema assegura aquesta propietat.

- **Disponibilitat.**

  Propietat: La disponibilitat fa referència a l'accessibilitat i usabilitat de les dades, sistemes o serveis quan els necessiten els usuaris autoritzats. Garanteix que la informació i els recursos estiguin sempre accessibles, siguin fiables i funcionin com s'espera.

  Requeriment: La descentralització que s'usarà en el sistema, assegura la disponibilitat de les dades emmagatzemades i la capacitat per guardar-ne de noves, mentre no sigui el mateix usuari que ha perdut la connexió.

A més de totes aquestes propietats de seguretat, el sistema ha de permetre la funcionalitat bàsica que exigeix el mateix projecte. Aquestes s'aniran explicant en seccions posteriors, així com es vagi definint el protocol.

3.2 Actors implicats

Hi ha diferents tipus d'actors. Tots ells, però, tenen accés al sistema a través d'una cartera virtual pròpia. A més, no tots els actors són persones reals que interactuen amb el sistema, sinó que també hi ha un actor que executa les funcions preparades de manera autònoma, els sensors. A continuació s'expliquen cada un d'ells.

- **Autoritat**: És l'encarregat de gestionar la resta d'actors. Les seves funcions es basaran a registrar nous farmacèutics, usuaris i sensors, i donar-los de baixa quan sigui necessari. El rol d'aquest actor, per tant, l'assumirà una única persona de confiança, com el responsable de farmàcia del mateix hospital.
- **Farmacèutics**: Són els membres de la plantilla de farmacèutics de l'hospital. Seran els encarregats d'entregar el medicament amb un sensor, el qual també hauran d'assignar al pacient.
- **Usuaris**: Són els pacients que necessitin guardar medicaments termolàbils a casa seva, els qui únicament podran consultar les alertes dels seus fàrmacs. Amb aquesta capacitat d'observació, l'usuari pot millorar la conserva. Per tal de respectar la privacitat dels usuaris, no es registrarà cap informació personal, només el número de la seva cartera virtual.
- **Sensors**: Aquests no són persones, sinó dispositius programats que accediran periòdicament a la funció de registre de dades. Tal i que, com s'explicarà a la posterior secció 3.3.3, no es registraran totes les dades, només les que surtin del rang previst.

3.3 Disseny del protocol

El protocol es divideix en quatre grups de funcions:

- *gestió d'actors*
- *assignació de sensor a usuari*
- *registres de temperatura*
- *visualització de les dades*

Així com mostra la Fig. 3.1, es crearà un SC per cada tipus d'actor i els actors només podran accedir a les funcions del contracte corresponent. A més, els contractes interactuaran entre ells per complir la funcionalitat prevista. Finalment, és important mencionar que hi haurà un cinquè SC, el *Processador*, que gestionarà i guardarà els registres de les temperatures, els medicaments i les associacions entre sensors i usuaris. Tot i que per tal de limitar l'abast del projecte no serà així, per l'emmagatzematge d'algunes d'aquestes dades es podria utilitzar un servidor InterPlanetary File System (IPFS), reduint el cost. IPFS és un sistema de fitxers distribuït, dissenyat per millorar l'eficiència dels sistemes descentralitzats.

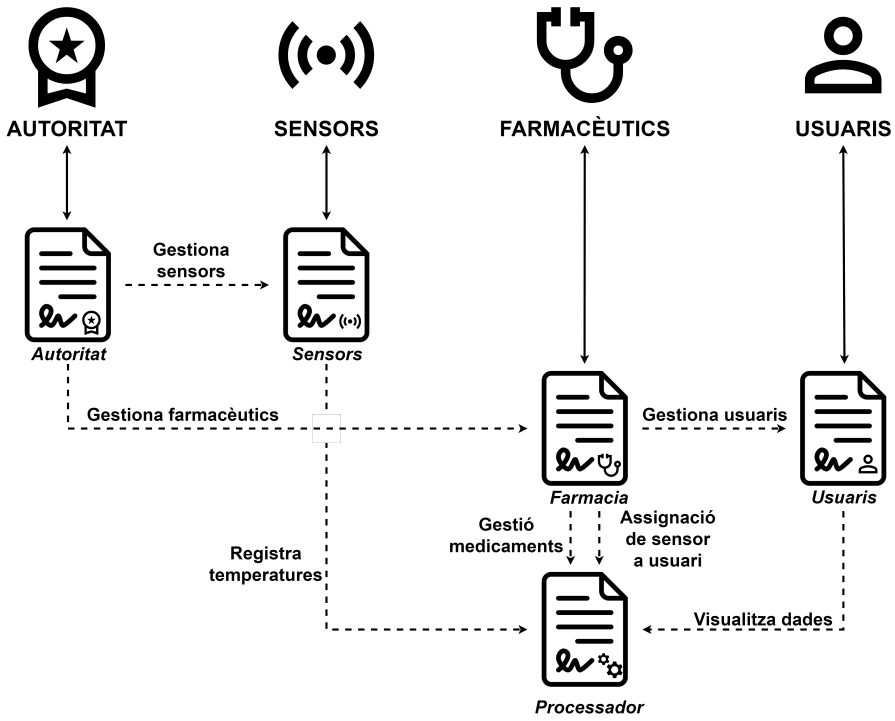

Figura 3.1: Diagrama de la comunicació entre SCs dins del mateix protocol.

L'estratègia de crear un SC per cada tipus d'actor facilitarà la implementació d'una interfície d'usuari, ja que la pàgina podrà filtrar segons l'adreça de la cartera virtual de l'usuari i només mostrar les funcions de l'SC corresponent, si és que l'adreça pertany a qualque grup.

En les següents seccions s'explica cada un dels grups de funcions mencionats, que ja es poden observar en la interacció entre contractes de la Fig. 3.1, on el contracte de la part inferior és el *Processador*.

Per una major facilitat en la comprensió dels següent algoritmes, es recomana consultar la taula de variables, adjunta a l'apèndix A.

3.3.1 Gestió d'actors

Aquesta secció descriu la gestió dels diferents tipus d'actors, mencionats i explicats a la secció 3.2. No tota la funcionalitat recaurà en un sol actor; es distribuirà entre l'*Autoritat* i els *Farmacèutics*. A continuació s'explica en detall la gestió d'actors.

Gestió de farmacèutics i sensors

Per un correcte funcionament del sistema és necessari que hi hagi un actor privilegiat, amb la capacitat de crear la resta d'actors i, si fos necessari, ajustar algun paràmetre global. En el sistema actual, l'actor *Autoritat* s'encarregarà de registrar nous *Sensors* i *Farmacèutics*. Com s'ha comentat a la secció 3.2, en un cas real, el paper d'autoritat el podria tenir el responsable de farmàcia, o qualsevol persona de confiança dins de l'hospital, ja que la seva feina serà registrar els nous sensors i farmacèutics que es vulguin introduir al sistema, o donar-los de baixa quan

sigui el moment.

Com es vorà en les següent seccions, els actors necessiten estar registrats per interactuar amb el sistema. Per tant, la tasca de l'autoritat és senzilla però important, en el sentit de que és un punt crític dins del sistema; si l'autoritat és maligna i dona de baixa als actors, el sistema deixarà de funcionar.

La gestió de farmacèutics i de sensors es fa de manera similar. A l'hora del registre, l'autoritat executarà una funció introduint els paràmetres necessaris. Com que un dels requisits del sistema és assegurar la confidencialitat de les dades, s'han reduït al màxim les dades emmagatzemades. L'estructura de dades dels farmacèutics es mostra a les taula 3.1; la dels sensors és idèntica. A més, per identificar-los no s'utilitzarà cap identificador extra, sinó que s'utilitzarà la seva mateixa adreça de la cartera virtual, d'aquesta manera es redueix l'ús de dades innecessàries, que es tradueixen en un major cost final del sistema. Així i tot, a l'estructura es poden afegir tants de paràmetres com es considerin necessaris.

Nom del paràmetre	**Tipus de variable**	**Explicació**
estat	enum	L'estat de l'actor pot ser *alta* o *baixa*. Aquestes opcions seran les dues possibles de la variable.

Taula 3.1: Estructura de farmacèutics.

La funció principal d'aquest bloc és registrar als nous actors. Per això, l'autoritat haurà d'executar la funció *registraFarmaceutic* o *registraSensor*, del seu propi contracte; les dues funcions són similars. L'algoritme 1 mostra la lògica de la funció per registrar farmacèutics. Aquesta funció crearà una nova variable de farmacèutic, f; hi assignarà els valors corresponents, l'estat d'alta, i la mapejarà amb la seva adreça. És important destacar que, igual que la resta de funcions del sistema, només certes adreces podran executar la funció, les associades a un tipus d'actor o un contracte determinat. En aquest cas concret, només l'adreça de l'autoritat podrà crear-la.

Algorithm 1 Algoritme per registrar un farmacèutic ($registraFarmaceutic$)

Input: $aFarmaceutic$

Require: $msg.sender = owner$

$\# f$

$f_{estat} \leftarrow alta$

$farmaceuticsMap[aFarmaceutic] \leftarrow f$

Per altre banda, les funcions de donar de baixa un actor, *baixaFarmaceutic* i *baixaSensor*, es basen en canviar l'estat de l'actor. L'algoritme 2 mostra el cas dels farmacèutics. Com es pot veure, es comprova que l'actor estigui prèviament donat d'alta, per evitar modificacions innecessàries.

Algorithm 2 Algoritme per donar de baixa un farmacèutic (*baixaFarmacia*)

Input: $aFarmaceutic$

Require: $msg.sender = owner$

Require: $farmaceuticsMap[aFarmaceutic]_{estat} = alta$

$farmaceuticsMap[aFarmaceutic]_{estat} \leftarrow baixa$

Gestió d'usuaris

La gestió dels *Usuaris* estarà a càrrec dels *Farmacèutics*. Aquesta assignació de tasques té l'objectiu de: quan els usuaris vagin a la consulta per primera vegada, no estaran registrats al sistema i, si el mateix farmacèutic pot donar-lo d'alta, l'usuari podrà ser donat d'alta i endur-se'n un sensor en sortir de la mateixa consulta, optimitzant la gestió del temps de tot el personal i dels propis usuaris.

La confidencialitat en les dades de l'usuari és inclús més crítica que la dels altres actors. Això és degut al fet que els farmacèutics podrien estar identificats de manera pública sense gaire inconvenient, però la identificació d'un pacient al sistema involucre saber que està en tractament d'un medicament i, a més, no serà difícil saber de quin, ja que es pot identificar el medicament que pren l'usuari. Per aquest motiu, l'estructura de dades dels usuaris és simplifica en el seu estat, igual que la dels farmacèutics, mostrada a la taula 3.1.

Les funcions *registraUsuari* i *baixaUsuari* són similars a les mencionades en l'anterior apartat 3.3.1. En concret, els algoritmes 3 i 4 mostren el funcionament final.

En aquest cas, les funcions només podran ser executades per un farmacèutic validat. Per això, s'haurà de validar que l'adreça que vol executar la funció pertany al conjunt de farmacèutics registrats, abans d'iniciar el procés de registre.

Algorithm 3 Algoritme per registrar un usuari (*registraUsuari*)

Input: $aUsuari$

Require: $farmaceuticValid(msg.sender)$

$\# u$

$u_{estat} \leftarrow alta$

$UsuarisMap[aUsuari] \leftarrow u$

Algorithm 4 Algoritme per donar de baixa un usuari (baixaUsuari)

Input: $aUsuari$

Require: $farmaceuticValid(msg.sender)$

Require: $UsuarisMap[aUsuari]_{estat} = alta$

$UsuarisMap[aUsuari]_{estat} \leftarrow baixa$

Els algoritmes 3 i 4 mostren una validació de l'adreça que crida la funció amb $farmaceuticValid()$. Aquesta funció està dins del contracte de l'autoritat, on s'emmagatzemen els registres dels farmacèutics. La funció mencionada, que es mostra a l'algoritme 5, cercarà l'estat de l'adreça i retornarà un booleà indicant si l'estat és d'alta o no. L'SC de l'autoritat tendrà una funció similar per la validació de sensors, $sensorValid()$.

Algorithm 5 Algoritme per validar el registre d'un farmacèutic ($farmaceuticValid$)

Input: $account$

Require: $msg.sender = aContractes$

```
if farmaceuticsMap[account]_estat = alta then
    Output: true
else
    Output: false
end if
```

En aquest cas, a la funció hi accedirà directament el contracte dels farmacèutics. Per aquest motiu, la primera passa serà comprovar que l'adreça d'accés és una de les adreces esperades; la dels contractes.

3.3.2 Assignació de sensor a usuari

L'assignació de sensor a usuari és el procés de registrar que un sensor s'ha entregat a un usuari, perquè les dades que aquest registri es guardin amb l'etiqueta de l'usuari corresponent.

Aquesta associació la farà un farmacèutic en el moment d'entregar el sensor. Per tant, hi ha d'haver una funció dins de l'SC de farmacèutics. Tot i això, com que tot el processament relacionat amb els sensors i els seus registres es farà a dins de l'SC *Processador*, mencionat al principi de la secció 3.3, la funció del registre de l'SC dels farmacèutics només cridarà a aquest segon SC, facilitant i abaratint el tractament posterior.

La funció *registraAssociacio* de l'SC dels farmacèutics cridarà, per tant, la funció *registraAssociacio* del *Processador*. Això sí, després d'haver validat que l'adreça de l'usuari que s'ha indicat a la funció està registrada com a tal. Aquest procés serà el mateix que la validació dels farmacèutics, explicada a l'anterior apartat 3.3.1, tot i que en aquest cas l'SC de farmacèutics podrà validar els usuaris sense haver de cridar a altres.

Llavors, la funció principal farà el que indica l'algoritme 6. Es pot veure que entre els paràmetres d'entrada d'aquesta funció, es troba l'Identificador Únic de Medicament (IUM) corresponent. Per entendre això, s'ha de tenir en compte el fet que el sensor estarà aferrat a un medicament que s'endurà l'usuari. Els diferents medicaments tenen diferents requisits i, com es menciona a la pròxima secció 3.3.3, només es registraran les temperatures fora de rang. Llavors, és necessari que el sistema sàpiga quin medicament controla un sensor, per saber si registrar, o no, les temperatures que vagin arribant del sensor.

Les associacions són una estructura que es registrarà de la mateixa manera que un sensor o un usuari. Les seves dades, explicades a la taula 3.2, són necessàries durant el procés de conservació del medicament.

Nom del paràmetre	Tipus de variable	Explicació
dataInici	uint	Timestamp al que s'entrega el medicament a un pacient i s'inicia el procés de conservació d'aquest.

dataFi	uint	Timestamp al que s'ha de subministrar el medicament, i s'acaba el procés de conservació del medicament.
aUsuari	address	Adreça de l'usuari al que s'entrega el medicament per conservar. És important, perquè això permetrà que els usuaris pugui veure l'estat de conservació, de només els seus medicaments.
aSensor	address	Adreça del sensor al que s'entrega el medicament per conservar. És important, perquè això permetrà que els sensors només puguin afegir dades a les seves associacions.
ium	uint	Identificador intern del medicament que es conserva. Amb aquest identificador es podran consultar els paràmetres de conservació del medicament que s'està conservant, per optimitzar el procés i només registrar les dades necessàries.
revocada	bool	Boleà que indica si una associació ha estat revocada. Això permet aturar els processos en que el medicament s'ha hagut de tirar o han sorgit altres inconvenients.

Taula 3.2: Estructura d'associacions.

Les passes que es realitzen a la funció de registrar una associació (algoritme 6) són:

1. Comprova que l'adreça d'accés és la d'un contracte de la mateixa DApp.
2. Comprova que la data de fi introduïda és major que l'actual: no té sentit seleccionar una data de subministrament (quan s'ha de prendre el medicament) anterior.
3. Comprova que l'usuari està registrat.
4. Comprova que el sensor està registrat.
5. Comprova que, si hi ha una associació prèvia al sensor, ha caducat o ha estat revocada.
6. Comprova que el medicament existeix, a través de comprovar la llargària del nom.
7. Augmenta el valor de la variable *totalAssociacions*, que serveix d'identificador de les associacions.
8. Crea una variable d'associació i hi guarda tots els valors corresponents.
9. Mapetja la variable amb el seu identificador, corresponent a *totalAssociacions*.
10. Mapetja l'identificador amb l'adreça del sensor. Així, es podrà saber quina associació té un sensor, en tot moment.
11. Mapetja l'identificador amb l'adreça de l'usuari. Així es podrà saber quines associacions té un usuari.
12. Retorna l'identificador de l'associació.

Algorithm 6 Algoritme per registrar una associació entre pacient i sensor ($registraAssociacio$)

Input: $aUsuari, aSensor, ium, dataFi$
Require: $msg.sender = aContractes$
Require: $now < dataFi$
Require: $usuariValid(aUsuari)$
Require: $sensorValid(aSensor)$
  if $associacionsMap[associacioDeSensor[_aSensor]]_{dataFi} > 0$ **then**
    Require: $associacioDeSensor[aSensor]).dataFi > now$ ò $associacionsMap(associacioDeSensor[aSensor]).revocada$
  end if
Require: $medicamentsMap[ium]_{nom}.lenght > 1$
  $totalAssociacions \leftarrow totalAssociacions + 1$
  $\# \ a$
  $a_{dataInici} \leftarrow now$
  $a_{dataFi} \leftarrow dataFi$
  $a_{aUsuari} \leftarrow aUsuari$
  $a_{aSensor} \leftarrow aSensor$
  $a_{ium} \leftarrow ium$
  $a_{revocada} \leftarrow false$
  $associacionsMap[totalAssociacions] \leftarrow a$
  $associacioDeSensor[_Sensor] \leftarrow totalAssociacions$
  $associacionsDeUsuari[_aUsuari] \leftarrow totalAssociacions$
  Output: $totalAssociacions$

Gestió de medicaments

Perquè es puguin tenir presents els paràmetres del medicament, hi ha diferents opcions. Aquestes podrien ser: consultar un servidor extern central, consultar arxius d'un IPFS o registrar-los en el mateix sistema. Per tal de no rompre la descentralització, un objectiu important del treball, i per la poca càrrega de dades que tendran els medicaments en si, s'ha optat per la darrera opció mencionada. Per tant, els medicaments es registraran dins d'una estructura dissenyada per aquest projecte i implementada en l'SC *Processador*.

Els encarregats del registre seran els farmacèutics, com en l'assignació de sensor-usuari, però, de la mateixa manera que en la funció mencionada, la funció dels farmacèutics cridarà a una segona funció de l'SC *Processador*. Aquesta funció es mostra a l'algoritme 7 i, bàsicament, comprovarà que el medicament no està registrat i que les dades són vàlides, crearà una nova variable *medicina*, hi assignarà les dades corresponents i la mapetjarà amb l'IUM indicat. En concret, les dades de l'estructura de la medicina són les presents en la taula 3.3.

Nom del paràmetre	**Tipus de variable**	**Explicació**
nom	string	Nom comercial del medicament.
minTemp	int16	Temperatura mínima del rang de conservació del medicament.
maxTemp	int16	Temperatura màxima del rang de conservació del medicament.
numMaxRegistres	uint16	Numero de registres fora de rang que pot assumir el medicament sense perdre eficàcia.

Taula 3.3: Estructura del medicament.

Algorithm 7 Algoritme per registrar un medicament (*registraMedicament*)

Input: $ium, nom, minTemp, maxTemp, numMaxRegistres$
Require: $msg.sender = aContractes$
Require: $medicamentsMap[ium]_{numMaxRegistres} = 0$
Require: $numMaxRegistres > 0$
Require: $nom.lenght > 2$
$\# \ m$
$m_{nom} \leftarrow nom$
$m_{minTemp} \leftarrow minTemp$
$m_{maxTemp} \leftarrow maxTemp$
$m_{numMaxRegistres} \leftarrow numMaxRegistres$
$medicamentsMap[ium] \leftarrow m$

És important destacar que gràcies a aquest sistema de registre de medicaments, i a que les dades d'associacions sensor-usuari inclouen l'identificador de la medicina, no serà necessari emmagatzemar la totalitat de les temperatures que mesurin els sensors. La funcionalitat clau d'aquest sistema recau en la capacitat de determinar quan un medicament deixa de ser apte i,

per tant, necessita ser substituït. Per aconseguir aquest objectiu, és suficient que es registrin les dades que han sortit del rang establert. Llavors, només es registraran les temperatures per damunt del màxim o per sotal mínim permès pel medicament en concret. Aquest enfocament contribuirà significativament a la reducció i optimització de les dades emmagatzemades.

3.3.3 Registre de temperatura

El registre de temperatures és tasca dels sensors. Per aquest treball es suposa que es disposa d'un conjunt de sensors amb capacitat d'executar la funció *registraTemperaturaSensor* periòdicament, indicant un valor de temperatura mesurat. Aquest valor podria ser una mitja del darrer període, sent una dada més representativa de la totalitat del temps entre registres, o el valor mesurat en el moment. Es pretén que aquestes mesures permetin detectar quan un medicament deixa de ser subministrable, gràcies a la variable $numMaxRegistres$ de l'estructura d'un medicament. A més, aquest registre permetrà crear un gràfic de la conservació del medicament, ja que per cada mesura es guardarà la data i hora, amb un *timestamp*.

La funció mencionada, que ha de ser executada pels sensors, estarà dins del seu SC. Aquesta però, com en el cas de la secció 3.3.2, cridarà a un funció de *Processador*, que implementarà la funcionalitat desitjada.

La funció encarregada de processar les temperatures que arriben i, si cal, registrar-les, s'anomenarà *registraTemperatura*. Aquesta funció rebrà el valor de temperatura i l'adreça del sensor que la valor registrar. Llavors, a partir del valor i els paràmetres de conservació del medicament, haurà de determinat si la temperatura s'ha de registrar i registrar-la, si és el cas, en una estructura *registre*, que tindrà els paràmetres de la taula 3.4. L'algoritme 8 mostra la logística completa de la funció i l'algoritme 9 explica la funció *necessitatRegistrar*, que es crida dins l'anterior funció per saber si les dades estan dins o fora del rang permès.

Nom del paràmetre	**Tipus de variable**	**Explicació**
data	uint	Data de registre de la temperatura.
aSensor	address	Adreça del sensor que ha enviat la temperatura.
valorTemp	int16	Valor mesurat i enviat pel sensor. Aquest valor està fora del rang desitjat ja que, si no fos així, no es registraria..

Taula 3.4: Estructura del registre.

La funció *registraTemperatura* crida a la funció *necessitatRegistrar* i, si aquesta retorna *true*:

1. Augmenta el valor de la variable *totalRegistres*, que serveix d'identificador dels registres.
2. Crea una variable de *registre* de temperatura.
3. Guarda els valors corresponents en la variable.
4. Mapetja la variable amb el seu identificador, corresponent a *totalRegistres*.
5. Mapetja l'identificador amb l'adreça del sensors. Així, es podrà fer el seguiment de registres d'un sensor.

6. Crida a *processaTemperatura*(), que determinarà si el medicament continua sent subministrable i, si no ho és, revocarà l'associació i notificarà a l'usuari.

Algorithm 8 Algoritme per registrar una mostra de temperatura (*registraTemperatura*)

Input: $aSensor, valor$
Require: $msg.sender = aContractes$
Require: $necessitaRegistrar(aSensor, valor)$
$totalRegistres \leftarrow totalRegistres + 1$
$\# \ r$
$r_{data} \leftarrow now$
$r_{aSensor} \leftarrow aSensor$
$r_{valor} \leftarrow valor$
$registresMap[totalRegistres] \leftarrow r$
$registresDeSensor[aSensor] \leftarrow totalRegistres$
processaTemperatura(aSensor)

La funció *necessitatRegistrar* cerca els requisits del medicament associat al sensor i els compararà amb el valor de temperatura enviat. D'aquesta manera només es registren els valors que es troben fora de rang, i que poden provocar una pèrdua d'eficàcia del medicament. Tot i que s'ha optat per utilitzar aquesta logística: sempre es crida a la funció *registraTemperatura* i, a través d'aquesta, es determina si s'ha de registrar o no; es podria optar per primer cridar a la funció *necessitatRegistrar* i només si el resultat d'aquesta és fals, cridar a la següent. Amb aquesta segona opció, s'aconsegueix tenir una despesa zero en els casos que la temperatura està dins del rang permès del medicament, ja que la que seria la primera funció és de només visualització i, per tant, té cost zero. El sistema de cridar sempre a una funció, s'usa perquè aporta facilitat de simulació en entorns sense aplicació d'usuari, però suposarà que en cada crida a la funció es gasti un poc de gas.

Tal com està plantejat el protocol, la funció *necessitatRegistrar*, detallada en l'algoritme 9, farà el següent:

1. Cercarà l'associació actual del sensor.
2. Si l'associació no ha caducat ni està revocada, cerca l'identificador de la medicina corresponent a l'associació.
3. Amb aquest identificador, es cerquen les temperatures màxima i mínima del medicament i es comparen amb el *valor* d'entrada.
4. Si el valor rebut està fora del rang permès, s'ha de registrar el valor; retorna *true*.
5. En qualsevol altre cas, retorna $false$.

El final del procés de registre de la funció privada *registraTemperatura* es crida a una altre funció *processaTemperatura*. Aquesta funció, mostrada en l'algoritme 10, processarà totes les dades que s'han registrat per una mateixa associació, és a dir, per un període de conservació, i determinarà si el medicament segueix amb bones condicions.

El primer pas que mostra l'algoritme 10 és la crida a la funció *medicamentEnMalEstat*. Aquesta funció, mostrada a l'algoritme 11, compararà el numero de registres de l'associació

Algorithm 9 Algoritme per consultar si és necessari registrar un valor de de temperatura (*necessitatRegistrar*)

Input: $aSensor, valor$

Require: $msg.sender = aContractes$

$idAssociacio \leftarrow associacioDeSensor[aSensor]$

if no $associacionsMap[idAssociacio]_{revocada}$ i $associacionsMap[idAssociacio]_{dataFi} < now$ **then** $idMedicament \leftarrow associacionsMap[idAssociacio]_{idMedicament}$

if $valor < medicamentsMap[idMedicament]_{minTemp}$ ò $valor > medicamentsMap[idMedicament]_{maxTemp}$ **then**

Output: $true$

end if

end if

Output: $false$

actual amb la variable *numMaxRegistres* del medicament que es conserva: si s'han registrat més temperatures fora de rang de les permeses, la conservació haurà fallat i s'haurà de revocar l'associació i notificar-ho a l'usuari; si no s'han registrat prou mesures fora de rang, segueix el procés de conservació. Aquesta funció només retorna un booleà per saber si cal actuar; la funció *processaTemperatura* s'encarregarà del procés de notificar i revocar.

Continuant amb la funció *processaTemperatura*, l'algoritme 10 mostra que, si el medicament està en mal estat: es crearà una notificació per l'usuari, mitjançant una estructura del pròpi SC, i es revocarà l'associació. La notificació tendrà els camps indicats a la taula 3.5. A l'algoritme 10 s'observen les passes per crear una notificació i revocar l'associació.

Nom del paràmetre	Tipus de variable	Explicació
data	uint	Data de creació de la notificació.
vista	bool	Indica si l'usuari ha llegit la notificació, o no.
idAssociacio	uint	Identificador de l'associació a la que es refereix.

Taula 3.5: Estructura de la notificació.

Respecte la funció *getRegistres* de l'algoritme 11, és una funció de visualització de dades que s'explica en major detall al pròxim apartat 3.3.4. En concret, la seva logística es mostra a l'algoritme 16.

3.3.4 Visualització de dades

La visualització de les dades és una part fonamental del sistema. Una de les propietats que aportarà el disseny actual és que dona eines als usuaris per poder millorar el procés de conservació. Per això s'han dissenyat quatre funcions que engloben tota la funcionalitat que necessita l'usuari. A més, també s'han dissenyat dues funcions de visualització pels farmacèutics, per tal de facilitar les seves tasques.

Algorithm 10 Algoritme per processar l'estat de conservació del medicament i revocar-lo i avisar a l'usuari, si el medicament és insubministrable ($processaTemperatura$)

Input: $aSensor$

if $medicamentEnMalEstat(_aSensor)$ **then**
$totalNotificacions \leftarrow totalNotificacions + 1$
$\#$ n
$n_{data} \leftarrow now$
$n_{vista} \leftarrow false$
$n_{idAssociacio} \leftarrow associacioDeSensor[aSensor]$
$notificacionsMap[totalNotificacions] \leftarrow n$
$notificacionsDeUsuari[associacionsMap[associacioDeSensor[aSensor]]_{aUsuaris}] \leftarrow totalNotificacions$
$associacionsMap[associacioDeSensor[aSensor]]_{revocada} \leftarrow true$
end if

Algorithm 11 Algoritme per determinar si un medicament està en mal estat de conservació ($medicamentEnMalEstat$)

Input: $aSensor$

$\#auxdataInici \leftarrow associacionsMap[associacioDeSensor[aSensor]]_{dataInici}$
$\#auxdataFi \leftarrow associacionsMap[associacioDeSensor[aSensor]]_{dataFi}$
$(\#auxRegistresAssociacio, ,) \leftarrow getRegistres(aSensor, auxdataInici, auxdataFi)$
$\#auxIum \leftarrow associacionsMap[associacioDeSensor[_aSensor]]_{ium}$
if $auxRegistresAssociacio > medicamentsMap[auxIum]_{numMaxRegistres}$ **then**
Output: $true$
end if
Output: $false$

Visualitzacions d'usuaris

- *elsMeusSensors()*. Aquesta funció de visualització retorna tots els sensors associats a un usuari, de la manera que mostra l'algoritme 12.

Algorithm 12 Algoritme de visualització per retornar tots els sensors associats a un usuari ($elsMeusSensors$)

$\#auxIdAssociacions \leftarrow visualitzaAssociacionsUsuari(msg.sender)$
$\#auxSensors$
for $(i \leftarrow 0; i < auxIdAssociacions.length; i++)$ **do**
  $auxSensors[i] \leftarrow sensorAssociacio(auxIdAssociacions[i])$
end for
Output: $auxSensors$

La funció *sensorAssociacio* que es mostra, tan sols retorna el sensor que s'ha assignat a una associació en concret, seleccions el paràmetre corresponent, *aSensor*, de la seva estructura, mostrada a la taula 3.2.

- *lesMevesAssociacionsActuals()*. Aquesta funció de visualització retorna totes les associacions amb sensor que té un usuari en el moment de cridar-la. És a dir, l'identificador de conservació dels medicaments que té en estat de conservació en aquell mateix moment. A més, també retorna la data de finalització de l'associació en si. La funció de l'SC dels usuaris cridarà a l'SC *Processador*, el qual compte amb la funció que mostra l'algoritme 13. Aquesta funció està programada amb dos bucles: el primer filtra les associacions, deixant només els identificadors de les que són actuals i no revocades, i el segon elimina els possibles espais buits i omple una segona cadena amb la data de finalització de cada associació.

- *visualitzaNotificacioNoVista()*. Aquesta funció no és de només visualització, ja que després de retornar la primera notificació no vista d'un usuari, la marcarà com a *ja visualitzada*. La funció de l'SC dels usuaris cridarà a *visualitzaNotificacioNoVista* de l'SC *Processador*. Aquesta, mostrada a l'algoritme 14, comprovarà si té notificacions pendents amb la funció *usuariTeNotificacions* i, si és el cas, la marcarà com a vista i la retornarà. Per altre banda, la funció *usuariTeNotificacions* recorrerà la cadena de notificacions d'usuari, cercant si hi ha notificacions noves, com indica l'algoritme 15.

- *visualitzaUnCicle()*. Aquesta funció de visualització pretén, sobretot, complementar la visualització de notificacions i la de visualitzar les pròpies associacions. L'anterior funció mencionada, $visualitzaNotificacioNoVista$, retorna la data d'una notificació i l'identificador de l'associació que ha fet saltar les alarmes. En aquest moment, l'usuari sap que el medicament no es troba amb condicions de ser subministrat. Així i tot, la funció $visualitzaUnCicle$ retornarà totes les mesures que s'han pres fora de rang. Així, l'usuari podrà saber quines temperatures fora de rang s'han registrat per un medicament en concret, i en quin instat, ja que també retorna les dates. Per tant, l'usuari podrà saber l'estat de conservació de, no només els medicaments que hagin estat rebutjats, sinó de tots els medicaments que se li han assignat. En concret, pot saber totes les seves associacions del moment amb la funció $lesMevesAssociacionsActuals$, prèviament mencionada.

Algorithm 13 Algoritme de visualització per retornar totes les associacions d'un usuari ($visualitzaAssociacionsUsuari$)

Input: $aUsuari$

$\#auxIdAssociacions$
$\#auxPosicio \leftarrow 0$
for $(i \leftarrow 0; i < associacionsDeUsuari[aUsuari].length; i++)$ **do**
  if $associacionsMap[associacionsDeUsuari[aUsuari][i]]_{dataFi} > now$ i no $associacionsMap[associacionsDeUsuari[aUsuari][i]]_{revocada}$ **then**
    $auxIdAssociacions[auxPosicio] \leftarrow associacionsDeUsuari[aUsuari][i]$
    $auxPosicio \leftarrow auxPosicio + 1$
  end if
end for
$\#auxFinalidAssociacions$
$\#auxDataAssociacoins$
for $(i \leftarrow 0; i < auxPosicio; i++)$ **do**
  $auxFinalidAssociacions[i] \leftarrow auxIdAssociacions[i]$
  $auxDataAssociacoins[i] \leftarrow associacionsMap[auxIdAssociacions[i]]_{dataFi}$
end for
Output: $(auxFinalidAssociacions, auxDataAssociacoins)$

Algorithm 14 Algoritme per retornar una notificació, encara no vista per l'usuari ($visualitzaNotificacioNoVista$)

Input: $aUsuari$

$(\#auxNotificacions, \#auxPosicio) \leftarrow visualitzaAssociacionsUsuari(msg.sender)$
Require: $auxNotificacions = true$
$notificacionsMap[auxPosicio]_{vista} \leftarrow true$
Output: $notificacionsMap[auxPosicio]_{data}, notificacionsMap[auxPosicio]_{idAssociacio}$

Algorithm 15 Algoritme de visualització per retornar si existeixen notificacions noves, indicant la posició si és el cas ($usuariTeNotificacions$)

Input: $aUsuari$

$\#auxIdNotificacions \leftarrow notificacionsDeUsuari[_aUsuari]$
$\#auxSensors$
for $(i \leftarrow 0; i < auxIdNotificacions.length; i++)$ **do**
  if no $notificacionsMap[auxIdNotificacions[i]]_{vista}$ **then**
    Output: $(true, auxIdNotificacions[i])$
  end if
end for
Output: $(false, 0)$

visualitzaUnCicle cridarà a la funció *getRegistresUsuari*, de l'SC *Processador*. La funció s'encarregarà de seleccionar el sensor i les dates d'inici i fi de l'assignació per, posteriorment, cridar a la funció general *getRegistres*. Aquesta darrera, retorna tots els valors registrats per un sensor entre dues dades, tota la informació que s'ha seleccionat prèviament. El funcionament de *getRegistres* es mostra a l'algoritme 16. Es tracta de començar amb tots els registres d'un sensor, filtrar només els que estan entre les dades compreses i acabar retornant el nombre de registres, i dues cadenes: una pels valors de temperatura i un per la data de registre d'aquestes.

Algorithm 16 Algoritme de visualització per retornar els registres de temperatura d'un sensor entre dos instants de temps (*getRegistres*)

Input: $aSensor, start, end$

$\#auxIdRegistres \leftarrow registresDeSensor[aSensor]$
$\#auxRegistres$
$\#auxPosicio \leftarrow registresMap[auxIdRegistres[i]]_{data}$
for $(i \leftarrow 0; i < auxIdRegistres.length; i++)$ **do** $\#auxData \leftarrow 0$
  if $auxData >= start$ i $auxData <= end$ **then**
    $auxRegistres[auxPosicio] \leftarrow auxIdRegistres[i]$
    $auxPosicio \leftarrow auxPosicio + 1$
  end if
end for
$\#auxDataArray$
$\#temperaturaArray$
for $(i \leftarrow 0; i < auxPosicio; i++)$ **do**
  $auxDataArray[i] \leftarrow registresMap[auxRegistres[i]]_{data}$
  $temperaturaArray[i] \leftarrow registresMap[auxRegistres[i]]_{valorTemp}$
end for
Output: $(auxPosicio, auxDataArray, temperaturaArray)$

Visualitzacions de farmaceutics

- *visualitzaUnCicle(idAssociacio)*. Aquesta funció de visualització fa la mateixa funció que l'anteriorment mencionada $lesMevesAssociacionsActuals()$, pels usuaris. En aquest cas, però, serà necessari que el farmacèutic introdueixi l'adreça de l'usuari del qual vol saber les associacions ja que els farmacèutics tindràn accés a les dades de tots els usuaris. Llavors, es cridarà la mateixa funció de l'SC *Processador*, mostrada a l'algoritme 13.

- *visualitzaMedicament(ium)*. La darrera funció de visualització, com es pot suposar amb el mateix títol, permetrà visualitzar les dades del medicaments ja registrats, als farmacèutics. La part important d'aquest procés es desenvolupa a l'SC *Processador*, que cercarà els paràmetres del medicament que identifica l'*ium* introduït, i els retornarà.

Capítol 4

Implementació

En aquest capítol es mostra i s'explica el codi que permetrà gestionar i controlar els medicaments durant el període de conservació desitjat, utilitzant la tecnologia blockchain. Tot el codi s'ha desenvolupat amb el llenguatge *Solidity*, i està pensat perquè sigui fàcil fer una interfície d'usuari, i el desplegament en una cadena de blocs.

Per la programació, s'han seguit els protocols explicats a la secció 3.3. Per això, si s'ha entès el funcionament dels diferents algoritmes, serà senzill entendre el codi final. A més, s'ha intentat respectar el nom de les variables que s'expliquen en la taula A.1 de l'annex A.

També és important mencionar que el sistema programat és completament funcional però, a més, està programat per permetre una fàcil incorporació de variables o funcions de visualització, en cas de ser necessari. Per exemple, en el futur es pot considerar que l'autoritat hagi de poder retornar un llistat dels farmacèutics donats d'alta al sistema. Per aconseguir-ho, es pot crear una funció de només visualització que consulti el *mapping farmaceuticsMap* i retorni els que estan d'alta.

Per explicar el codi de manera ordenada, es començarà explicant les funcions més representatives de cada un dels contractes d'actor:

- *Autoritat.sol*
- *Farmacia.sol*
- *Sensors.sol*
- *Usuaris.sol*

Així com s'ha comentat, reiterades vegades, en la secció 3.3, gran part de les funcions d'aquests contractes no implementen tota la funcionalitat, sinó que s'ajuden del contracte *Processador.sol*. Per tant, llavors s'explicarà part d'aquest contracte, que complementarà l'explicació anterior.

El motiu de la creació d'un contracte *Processador.sol* es basa en reduir el cost del sistema, facilitar els processos generals i aconseguir un codi ordenat. Hi ha estructures de dades que no pertanyen directament a cap actor però han de ser accessibles per la majoria d'ells, com és el cas dels registres de temperatura. Llavors, no té sentit implementar la seva funcionalitat en el contracte d'un actor i, en cas de fer-ho, podria provocar un augment del cost, ja que s'haurien de fer moltes consultes i escriptures externes al contracte.

En conclusió, el *Processador.sol* és el que s'encarrega de gestionar la major part de les dades i el que aplica la major part de la funcionalitat. A més, com es vorà més endavant, aquest és l'encarregat de desplegar tots els altres contractes. Així, s'assegura un desplegament segur i senzill.

En aquest apartat només s'aportarà el codi de les funcions més destacables. Així i tot, al repositori [13] de GitHub es pot consultar tot el codi.

4.1 Autoritat.sol

Aquest contracte inclou tota la funcionalitat que necessita l'autoritat. A més, com que l'actor registra i dona de baixa a farmacèutics i sensors, al mateix contracte s'emmagatzemen les variables de farmacèutic i sensor. També s'han creat dues funcion perquè els altres contractes puguin validar si un farmacèutic o sensor està donat d'alta.

4.1.1 Declaracions inicials

La majoria de les variables del contracte s'expliquen a la taula A.1. Així i tot, cal mencionar que s'afageix l'ús d'una la llista enumerada *estatActor*: s'utilitzarà per assignar l'estat d'*alta* o *baixa* als actors.

El *constructor* es cridarà per desplegar el contracte; aquest ha d'inicialitzar totes les dades necessàries. En concret, com que *Processador.sol* serà l'encarregat de desplegar el contracte, només hi indicarà: l'adreça de l'autoritat, l'única que podrà donar d'alta i baixa als actors, i l'adreça del contracte *Processador.sol*, que es guardaran a les variables corresponents.

Després es definiran les estructures de *Farmaceutic* i *Sensor*, i els *mapping* necessaris per guardar-los.

4.1.2 *Modifiers*

En aquest SC, les funcions de gestió de farmacèutics i sensors només són accessibles per l'adreça de l'*autoritat*, mentre que les funcions de validar els farmacèutics i sensors només ho seran pels altres SCs del sistema. Llavors, s'han dissenyat dos *modifiers* generals per controlar l'accés a les funcions: *onlyByAutoritat* i *onlyByContractes*.

4.1.3 Funcions

Com ja s'ha mencionat, les funcions de l'*Autoritat.sol*, es basen en: registrar, donar de baixa i validar, a farmacèutics i sensors.

Les funcions de registrar i donar de baixa a farmacèutics, explicades en els algoritmes 1 i 2, respectivament, s'han codificat així com mostra el llistat 4.1, seguint exactament els dissenys anteriors: es comprova l'adreça d'accés i es crea i guarda una variable, o es canvia l'estat. El procés de gestió de sensors és similar.

A més de les funcions de gestió de farmacèutics i sensors, s'han programat funcions perquè la resta de contractes puguin validar, de manera ràpida, si una adreça pertany a un farmacèutic, o sensor, registrat al sistema. En concret, el llistat 4.2 mostra la funció que permetrà validar

```
function registraFarmaceutic(address _aFarmaceutic) public onlyByAutoritat(
    msg.sender){
     Farmaceutic memory f;
     f.estat = estatActor.alta;
     farmaceuticsMap[_aFarmaceutic] = f;
}

function baixaFarmaceutic(address _aFarmaceutic) public onlyByAutoritat(msg.
    sender){
     require(farmaceuticsMap[_aFarmaceutic].estat == estatActor.alta, "
         Farmaceutic no te estat \"alta\"");
     farmaceuticsMap[_aFarmaceutic].estat = estatActor.baixa;
}
```

Listing 4.1: Funcions de registrar i donar de baixa a farmacèutics al sistema.

els farmacèutics. Aquesta funció l'utilitzarà, per exemple, el contracte *Farmacia.sol* per validar que l'adreça que vol executar una funció sí que pertany a un farmacèutic registrat.

```
function farmaceuticValid(address _account) public view onlyByContractes(msg
    .sender) returns(bool farmaceuticIsValid){
     if (farmaceuticsMap[_account].estat == estatActor.alta) {
         return true;
     } else {
         return false;
     }
}
```

Listing 4.2: Funció per validar que una adreça pertany a un farmacèutic.

4.2 Farmacia.sol

El contracte *Farmacia.sol* inclou la funcionalitat dels farmacèutics. És a dir que, principalment, inclourà funcions per gestionar als usuaris, registrant-los i donant-los de baixa; comprovar la validesa d'usuaris, com fa el contracte *Autoritat.sol* amb farmacèutics i sensors; registrar medicaments; i gestionar associacions entre pacient i sensors, activar o desactivar els períodes de conservació d'un medicament per un pacient. A més de tot això, hi ha dues funcions de només visualització que permetran que els farmacèutics puguin desenvolupar la seva tasca de manera senzilla i sabent que, en cas de dubte, tenen l'opció de consultar les dades.

4.2.1 Declaracions inicials

Les declaracions de l'SC es basen en la estructura d'*Usuari*, amb el *mapping* corresponent i la variable *estatActor*, i variables d'adreça o contracte per cridar o poder ser cridat per altres SC del sistema. A més, el constructor inicialitzarà les adreces de contractes, passades per paràmetre.

4.2.2 *Modifiers*

De la mateixa manera que, en l'*Autoritat.sol*, les funcions només són accessibles per l'adreça de l'*autoritat* i els altres contractes del sistema; en aquest cas, les funcions només seran accessibles per farmacèutics registrat i els altres contractes del sistema. Per tant, es compte amb els *modifiers*: *onlyByFarmecautics* i *onlyByContractes*. El funcionament del primer es basa en cridar la funció *farmaceuticValid* d'*Autoritat.sol* 4.1.

4.2.3 Funcions

Les primeres funcions del contracte, *registraUsuari*, *baixaUsuari* i *usuariValid*, serveixen per registrar, donar de baixa o validar el registre d'un usuari. La simplicitat d'aquestes funcions permet aconseguir que unes de les funcions més utilitzades del sistema es redueixin en poc més de 10 línies de codi, assegurant un cost petit de desplegament i funcionament.

Llavors, al contracte es defineix la gestió del medicaments. Els medicaments, en aquest cas, únicament es podran crear i visualitzar. Tanmateix, les funcions *registraMedicament* i *visualitzaMedicament* tan sols cridaran a les corresponents funcions de l'SC *Processador.sol*, on s'ha desenvolupat tota la lògica necessària.

Farmacia.sol acaba definint un apartat de gestió d'associacions. Per tant, al contracte es defineixen les funcions *registraAssociacio*, *baixaAssociacio* i *visualitzaAssociacionsUsuari*, que s'encarreguen de registrar noves associacions, revocar associacions existents i visualitzar les associacions d'un usuari en contret, respectivament. En concret, la funció de registre haurà de començar validant que l'usuari encara no estigui registrat, i el farmacèutic sí. Les tres funcions acaben cridant al *Processador.sol*, on es processaran les associacions.

4.3 Sensors.sol

La funcionalitat de l'SC *Sensors.sol* es basa la capacitat de rebre temperatures dels sensors, per posteriorment registrar-se si és necessari.

4.3.1 Declaracions inicials

Les declaracions del contracte es simplifiquen a les variables dels SC *Autoritat.sol* i *Processador.sol*, utilitzades per accedir a les funcions *sensorValid* i *registraTemperatura*, respectivament. Les dues variables s'inicialitzen al constructor.

4.3.2 *Modifiers*

Així com s'explica en l'*Autoritat.sol*, 4.1, o el *Farmacia.sol*, 4.2, també es requereix la exclusiva accessibilitat als sensors registrats. Això es validarà a través del *modifiers*: *onlyBySensors*, que cridar la funció *sensorValid* d'*Autoritat.sol* 4.1.

4.3.3 Funcions

Aquest contracte s'ha simplificat al màxim, ja que els sensors no necessiten funcions especial; únicament estan programats per registrar la temperatura cada cert període. En concret, només hi ha una funció *registraTemperaturaSensor* que valida que l'executor sigui un sensor registrat i *registraTemperatura* de *Processador.sol*. Com s'explica en la corresponent secció 4.5, la funció està optimitzada per suposar el menor cost possible.

4.4 Usuaris.sol

El contracte *Usuaris.sol* inclou les quatre funcions necessàries pels usuaris del sistema. Aquestes funcions permetran que un usuari visualitzi els seus sensors o associacions, o consulti les dades d'una associació sense cap cost. A més, també podrà visualitzar les notificacions, però amb un petit cost.

4.4.1 Declaracions inicials

Les úniques declaracions inicials són les variables dels SC *Farmacia.sol* i *Processador.sol*, que s'utilitzaran per cridar a funcions d'aquests contractes. Les dues variables s'inicialitzen al constructor.

4.4.2 *Modifiers*

De la mateixa manera que en els altres contractes d'actor, l'SC *Usuaris.sol* necessita comprovar que a les seves funcions hi accedeixi un usuari registrat al sistema. Aquesta validació es farà amb el *modifier onlyByUsuaris*, que cridarà a la funció *usuariValid* de *Farmacia.sol*, 4.2.

4.4.3 Funcions

Aquest contracte compte amb tres funcions de visualització, sense cost, que permetrà una funcionalitat senzilla i àgil a l'usuari. Aquestes són:

- *elsMeusSensors*: Aquesta funció, com indica el nom, retornarà les adreces dels sensors associats a un usuari. Així, un usuari podrà saber fàcilment quins sensors té, o hauria de tenir, dins la gelera. Per fer-ho, la funció cerca les associacions de l'usuari que l'executa, cridant a *visualitzaAssociacionsUsuari* de *Processador.sol*, i llavors cerca el sensor de cada associació, també a través de *Processador.sol*. El procediment es pot veure al llistat 4.3.

- *lesMevesAssociacionsActuals*: La funció retorna els identificadors de les associacions de l'usuari que executa la funció. Això s'aconsegeuix a través de la crida a la funció *visualitzaAssociacionsUsuari* de *Processador.sol*.

- *visualitzaUnCicle*: La darrera funció de visualització retorna els registres de temperatura d'una associació, acompanyades de la corresponent marca temporal. D'aquesta manera, es senzill fer una gràfic temporal de totes les temperatures que s'han registrat fora de rang, per a un medicament.

```
function elsMeusSensors() public view onlyByUsuaris(msg.sender) returns(
    address[] memory adrecesSensors){
     (uint[] memory auxIdAssociacions, ) = process.
         visualitzaAssociacionsUsuari(msg.sender);
     address[] memory auxSensors = new address[](auxIdAssociacions.length);
     for(uint i = 0; i < auxIdAssociacions.length; i++){
         auxSensors[i] = process.sensorAssociacio(auxIdAssociacions[i]);
     }

     return auxSensors;
}
```

Listing 4.3: Funció per retornar les adreces dels sensors associats al mateix usuari.

Llavors, hi ha una quarta funció. Aquesta darrera no és de visualització, ja que té un cost associat, però retornarà uns valors. La funció és *visualitzaNotificacioNoVista* i retornarà la primera notificació no vist d'un usuari, si en té. No pot ser de visualització ja que, abans de retornar la notificació la marca com a vista. De totes maneres, en aquest contracte només es crida a una segona funció *visualitzaNotificacioNoVista* del *Processador.sol*; en el corresponent apartat 4.5, s'explicarà en detall.

4.5 Processador.sol

El *Processador.sol* és l'SC que s'encarrega de gestionar i processar la major part d'execucions del sistema. En ell es guarden els medicaments, registres de temperatura, associacions, notificacions i totes les variables i *mappings* corresponents.

4.5.1 Declaracions inicials

Aquest contracte es cridat per tots els altres del sistema i, a més, s'encarrega de desplegar la resta de contractes. Per tant, necessitarà tenir les variables d'adreça de tots els SC i un booleà per assegurar que els contractes només es despleguen una vegada, *contractesDesplegats*. També cridarà, en algún moment, a funcions d'*Autoritat.sol*, *Sensors.sol* i *Farmacia.sol*, pel que haurà de tenir les variables de contracte.

Com s'ha comentat anteriorment, el contracte guarda els medicaments, registres de temperatura, associacions i notificacions, pel que haurà de definir les estructures i *mappings* corresponents.

4.5.2 *Modifiers*

L'apartat de *Modifiers* és senzill, comparat amb la totalitat del contracte, ja que només es necessita un *modifier* pel desplegament, *onlyByAutoritat*, i un per a la resta de funcions, *onlyByContractes*.

4.5.3 Funcions

Tot i que s'ha intentat simplificar el contracte, per la tasca que abraça, aquest té moltes funcions i, algunes, molt carregades. El present apartat s'ha dividit entre els diferents grups de funcionalitat, per poder entendre millor la complexa funcionalitat.

Desplegament de contractes

La primera funció és *desplegaTotsElsSC*. Aquesta funció no només s'encarrega de desplegar tots els contractes del sistema, sinó que també intercanviarà les adreces de contracte, entre aquests, perquè tots puguin aconseguir la funcionalitat prevista. A més, només permetrà un desplegament de contractes. La funció acabarà retornant les adreces de tots els contractes desplegats.

Gestió de temperatures

La funció principal d'aquest bloc és *registraTemperatura*, que es mostra al llistat 4.4. Aquesta, serà cridada indirectament cridada pels sensors, ja que aquests es comunicaran amb el seu contracte, explicat a la secció 4.3, i serà el mateix SC qui cridarà la funció *registraTemperatura*, indicant l'adreça del sensor i la temperatura registrada.

```
function registraTemperatura(address _aSensor, int16 _valor) public
    onlyByContractes(msg.sender){
     require(necessitatRegistrar(_aSensor, _valor), "La temperatura esta dins
         del rang, o l'associacio ha expirat.");

     totalRegistres++;

     Registre memory r;
     r.data = block.timestamp;
     r.aSensor = _aSensor;
     r.valorTemp = _valor;
     registresMap[totalRegistres] = r;

     registresDeSensor[_aSensor].push(totalRegistres);

     processaTemperatura(_aSensor);
}
```

Listing 4.4: Funció per registrar les temperatures llegides fora de rang.

El primer pas serà consultar, a la funció de visualització *necessitatRegistrar*, si és necessari registrar les temperatures. La funció mencionada, consultarà la validesa de l'associació, i els valors màxim i mínim de temperatura, del medicament que controla el sensor, per determinà si la temperatura s'ha de registrar o no, retornant *vertader* o *fals*, respectivament.

Si s'ha rebut *fals* de la funció *necessitatRegistrar*, es sortirà de la funció, retornant el missatge *La temperatura esta dins del rang, o l'associacio ha expirat.*; altrament: s'augmentarà el valor de registres totals del sistema, es crearà un registre i s'hi guardaran les variables, es mappetjarà el registre, i es cridarà a la funció *processaTemperatura*.

La funció *processaTemperatura*, mostrada al llistat 4.5 serveix per determinar si el medicament segueix sent vàlid o no, i s'ha de revocar l'associació i enviar una notificació. Com que l'objectiu és consumir el menor gas possible, la primera passa és cridar a una funció de visualització, *medicamentEnMalEstat*, que determinarà si el medicament segueix sent consumible o no, depenent del nombre de registres obtinguts i permesos. Llavors, si el medicament segueix en bon estat sortirà, però sinó: augmentarà el valor de les notificacions totals del sistema, crearà una notificació i s'omplirà, es mappetjarà la notificació, i es revocarà l'associació.

```
function processaTemperatura(address _aSensor) private {
    if(medicamentEnMalEstat(_aSensor)) {
        totalNotificacions++;

        Notificacio memory n;
        n.data = block.timestamp;
        n.vista = false;
        n.idAssociacio = associacioDeSensor[_aSensor];
        notificacionsMap[totalNotificacions] = n;

        notificacionsDeUsuari[associacionsMap[associacioDeSensor[_aSensor]].
           aUsuari].push(totalNotificacions);

        associacionsMap[associacioDeSensor[_aSensor]].revocada = true;
    }
}
```

Listing 4.5: Funció per determinar si el medicament segueix sent vàlid.

A més del conjunt de funcions de registre de temperatura, el bloc compte amb dues altres funcions que els diferents contractes podran cridar. Aquests són:

- *visualitzaNotificacioNoVista*, llistat 4.6: La crida el contracte *Usuaris.sol*, explicat a la secció 4.4, i s'encarrega de retornar la notificació no llegida més antiga de l'usuari, si en té. Per optimitzar el procés, el primer pas és cridar una funció de visualització, *usuariTeNotificacions*, que s'encarregarà de determinar si l'usuari té notificacions pendents i, si en té, retornar la posició dins de la cadena de notificacions. Llavors, la funció només haurà de retornar *No hi ha notificacions.* o les dades de la notificació no llegida, i marcar-la com a llegida.

- *getRegistresUsuari*: Aquesta funció retorna la cadena de tots els registres d'una associació. El funcionament es basa en cercar el sensor i les dates de principi i fi, de l'associació, per llavor cridar a la funció privada de visualització *getRegistres*. La funció *getRegistres*, que es es mostra al llistat 4.8, retorna totes les temperatures registres per un sensor, entre dos instants de temps, amb la marca de temps de cada una i el total. Com es pot observar, la funció utilitza dos bucles: el primer selecciona tots els registres del sensor dins de l'interval temporal i el segon selecciona els valors de temperatura i temps, de cada un dels registres seleccionats.

```
function visualitzaNotificacioNoVista(address _aUsuari) public
    onlyByContractes(msg.sender) returns (uint dataNotificacio, uint
    idAssociacio) {
     (bool auxNotificacions, uint auxPosicio) = usuariTeNotificacions(
        _aUsuari);
     require (auxNotificacions, "No hi ha notificacions.");
     notificacionsMap[auxPosicio].vista = true;
     return (notificacionsMap[auxPosicio].data, notificacionsMap[auxPosicio].
        idAssociacio);
}
```

Listing 4.6: Funció per retornar notificacions no llegides d'un usuari.

```
function getRegistres(address _aSensor, uint _start, uint _end) private view
     returns (uint totalTemperatures, uint[] memory dataTemperatura, int16[]
    memory valorTemperatura) {
     uint[] memory auxIdRegistres = registresDeSensor[_aSensor];
     uint[] memory auxRegistres = new uint[](auxIdRegistres.length);
     uint auxPosicio = 0;

     for (uint i = 0; i < auxIdRegistres.length; i++) {
         uint auxData = registresMap[auxIdRegistres[i]].data;
         if (auxData >= _start && auxData <= _end) {
             auxRegistres[auxPosicio] = auxIdRegistres[i];
             auxPosicio++;
         }
     }

     uint[] memory auxDataArray = new uint[](auxPosicio);
     int16[] memory temperaturaArray = new int16[](auxPosicio);
     for (uint i = 0; i < auxPosicio; i++) {
         auxDataArray[i] = registresMap[auxRegistres[i]].data;
         temperaturaArray[i] = registresMap[auxRegistres[i]].valorTemp;
     }

     return (auxPosicio, auxDataArray, temperaturaArray);
}
```

Listing 4.7: Funció per registrar les temperatures llegides fora de rang.

Gestió d'associacions

La gestió d'associacions s'encarrega de registrar, rebutjar i visualitzar les dades de les associacions entre sensors i usuaris, del sistema.

La funció més extensa és *registraAssociacio*, del llistat **??**, que comprova que:

1. La data de finalització, proposada, no sigui del passat.
2. L'usuari estigui correctament registrat.
3. El sensor estigui correctament registrat..
4. Si el sensor ja ha tingut alguna associació, aquesta estigui caducada o rebutjada.
5. L'IUM indicat sigui vàlid.

Si totes les comprovacions són correctes, la funció crearà, omplirà i mapetjarà l'associació, retornant l'identificador al final.

```
function registraAssociacio(address _aUsuari, address _aSensor, uint _ium,
    uint _dataFi) public onlyByContractes(msg.sender) returns(uint
    idAssociacio){
     require(block.timestamp < _dataFi, "Data finalitzacio incorrecte");
     require(farm.usuariValid(_aUsuari), "Usuari no registrat");
     require(autor.sensorValid(_aSensor), "Sensor no registrat");
     if(associacionsMap[associacioDeSensor[_aSensor]].dataFi > 0) {
         require(((associacionsMap[associacioDeSensor[_aSensor]].dataFi >
            block.timestamp) || (associacionsMap[associacioDeSensor[_aSensor
            ]].revocada)), "Sensor actualment utilitzat");
     }
     require(bytes(medicamentsMap[_ium].nom).length > 1, "El medicament no
        existeix");

     totalAssociacions++;

     Associacio memory a;
     a.dataInici = block.timestamp;
     a.dataFi = _dataFi;
     a.aUsuari = _aUsuari;
     a.aSensor = _aSensor;
     a.ium = _ium;
     a.revocada = false;
     associacionsMap[totalAssociacions] = a;

     associacioDeSensor[_aSensor] = totalAssociacions;

     associacionsDeUsuari[_aUsuari].push(totalAssociacions);

     return totalAssociacions;
}
```

Listing 4.8: Funció per registrar les temperatures llegides fora de rang.

La segons funció, *baixaAssociacio*, canviarà l'estat de l'associació que es vulgui rebutjar, després de comprovar que aquesta està registrada i no revocada.

Finalment, les funcions de visualització són:

- *sensorAssociacio*: Aquesta, és cridada per la funció *elsMeusSensors* dels usuari, explicada en la secció 4.4. *sensorAssociacio* retorna l'adreça del sensor d'una associació.

- *visualitzaAssociacionsUsuari*: La darrera funció retorna tots els identificadors d'associació que estan relacionats amb un mateix usuari. Aquesta, és cridada per les funcions *elsMeusSensors* i *lesMevesAssociacionsActuals* dels usuaris. El procediment de *visualitzaAssociacionsUsuari* es basa en seleccionar, de les associacions de l'usuari, les que no estan caducades ni revocades.

Gestió de medicaments

El darrer bloc de funcionalitat del *Processador.sol* s'encarrega de gestionar els registres de medicament. Els farmacèutics han de poder registrar nous medicaments i visualitzar-los, el que ja s'ha comentat a la corresponent secció 4.2. Llavors, en aquest SC, s'implementa la funcionalitat necessària per registrar nous medicaments, amb la funció *regitraMedicament*, que: crearà un nou medicament, li assignarà les dades i el guardarà, després d'haver comprovat que els camps introduits pel farmacèutic són correctes.

Per acabar, es troba la funció de visualització *visualitzaMedicament*, que retornarà les dades del medicament del qual s'indiqui l'IUM.

Capítol 5

Costs del sistema

Aquest projecte no està pensat per un ús lliure, sinó per un hospital o conjunts d'hospitals que gestionin i abonin el cost d'ús. Si el sistema és usat per un hospital molt gran, o compartit per un bon nombre d'hospitals, podria ser pràctic el fet de crear una cadena de blocs privada: tot i que el procés és més complicat, el cost del sistema és pràcticament zero, després d'haver creat la cadena. A més, el fet de fer servir una blockchain privada, afavoreix la confidencialitat de les dades, i es podrien afegir atributs extres.

Si per altre banda es decideix utilitzar una blockchain pública, com per exemple *Ethereum*, l'organització ha de tenir present que totes les accions de modificació tenen un cost, i que el mateix hospital haurà de fer-se càrrec del cost de les transaccions dels diferents actors. Per tant, haurà d'abonar una certa quantitat de monedes virtuals a cada actor, perquè puguin pagar el *gas* de les transaccions.

El *gas*, en context de la tecnologia blockchain, es refereix a una mesura de cost computacional necessària per executar operacions a la plataforma blockchain. Cada instrucció o operació que es realitza a un contracte intel·ligent consumeix una quantitat específica de *gas*. Les unitats del *gas* són *Gwei* i, a la plataforma d'Ethereum, es paga amb *Ether* ($1Ether = 10^9 Gwei$). El preu d'*Ether* a pagar es calcula de la manera següent: $Ether = gas \cdot taxaGas$, on la *taxaGas* influirà en el preu, però també a la velocitat de minat (a major taxa, més ràpid serà el minat).

Utilitzant una blockchain pública, el cost d'ús del sistema variarà de la cadena de blocs que s'utilitzi. A més, l'organització ha de tenir present que el preu de les monedes virtuals fluctua molt sovint.

Per tenir una aproximació del cost d'ús del sistema, s'ha suposat la implementació d'aquest a la plataforma Polygon, tenint en compte el cost mitjà de dia 25 de desembre de 2023, que era de 0,7923€/*Matic*, on *Matic* és la moneda usada a la plataforma, i una taxa de 120 Gwei, corresponent a una espera d'entre 10 i 30 segons per transacció. Polygon és una plataforma que té com a objectiu augmentar l'escalabilitat d'Ethereum, gràcies a una gran quantitat de cadenes laterals, o *sidechains*.

La taula 5.1 mostra el *gas* aproximat que s'utilitza per cada una de les accions, i el cost d'aquestes si es considera el preu de *Matic* de dia 25 de desembre i una taxa ràpida, de 120 Gwei. Així i tot, el cost pot variar una mica depenent de l'entrada de la funció o del processament necessari.

De la taula 5.1 es pot fer una estimació del cost. Amb aquest preu i taxa, el desplegament de tots els contractes tan sols costaria 1,13 euros a l'organització que implantés el sistema.

Llavors, suposant que: els medicaments es guarden durant una mitja de 30 dies, els sensors registren una temperatura cada 30 minuts i només el 4% de les mostres de temperatura estan fora de rang, es pot aproximar que el cost que l'hospital hauria d'assumir per cada medicament és d'aproximadament 8,63€, segons els valors expressats i sense tenir en compte les possibles consultes de notificació per part dels usuaris, que costaran 0,4 i 0,7 cèntims d'euro, depenent de si hi ha notificació o no.

Funció	**Gas (weis)**	**EUR (120 GWei)**
Desplegament de tots els SC	11903907	1,1318
Registrar farmacèutic	46574	0,0044
Baixa farmacèutic	24777	0,0024
Registrar usuari	57121	0,0054
Baixa usuari	35352	0,0034
Registrar medicament	92664	0,0088
Registrar associació	269099	0,0256
Baixa associació	64602	0,0061
Intentar registrar valor dins del rang	58591	0,0056
Registrar valor fora rang	156884	0,0149
Registrar valor fora del rang i enviar notificació	353941	0,0337
Visualitzar notificació (no n'hi ha)	46670	0,0044
Visualitzar notificació (sí n'hi ha)	76605	0.0073

Taula 5.1: Cost de les principals funcions del contracte a la blockchain de Polygon.

Tot i que el cost de 8,63€ pot parèixer excessiu, tenint en compte el nombre de medicaments que es conserven simultàniament, cal mencionar que el cost dels medicaments mes utilitzats sol oscil·lar els 200 i 500 euros per dosi, com en el cas de [14], [15] i [16]. Per tant, un control dels medicaments suposarà una gran disminució del cost, ja que els medicaments en mal estat són menys efectius i requereixen major nombre de subministraments.

Capítol 6

Conclusions i treballs futurs

6.1 Conclusions

En aquest TFM, s'ha desenvolupat un sistema de control de medicaments termolàbils, basat en la tecnologia blockchain. El disseny s'ha abordat amb una atenció especial a les diverses necessitats i requeriments del projecte, proporcionant una estructura que permeti una implementació i modificació eficients en el futur. Així doncs, es pot afirmar haver complert els diferents objectius del projecte, que eren:

- *Estudi de l'estat de l'art*: S'ha analitzat un conjunt de sistemes existents, relacionats amb l'ús de la blockchain pel control dels medicament, i recomanacions en el transport de medicaments. Així i tot, s'ha diferenciat la intenció del projecte amb els projectes existents, demostrant la millora d'algun apartat, i s'ha demostrat el compliment de les necessitats del procés, més enllà de les recomanacions de les organitzacions internacionals.
- *Disseny del protocol*: El protocol, que s'explica amb un profund detall, ha tingut en compte els requisits específics del projecte.
- *Implementació del sistema*: La implementació del sistema, segons el protocol, demostra la correcte funcionalitat.
- *Anàlisi del cost*: L'anàlisi del sistema, mostra la viabilitat del sistema en termes econòmics. El cost que s'hauria d'assumir, es veuria recompensat en l'augment d'eficàcia dels medicaments.

El producte resultant ha demostrat ser tant necessari com assequible. Tot i que la importància de controlar els medicaments termolàbils ja era coneguda i treballada, l'ús innovador de la tecnologia blockchain en aquest context específic no té precedents documentats. Malgrat les clares avantatges que aquesta tecnologia aporta, s'ha identificat un repte potencial: l'ús d'aquesta tecnologia tan moderna podria limitar l'accessibilitat als usuaris que no disposin dels coneixements o recursos tecnològics necessaris, impedint-los la consulta de les seves dades i la informació relativa a la caducitat dels seus medicaments.

Afrontant aquesta consideració, el sistema es podria modificar per no requerir l'atenció dels usuaris, sent així accessible per a tothom. Aquestes adaptacions i millores es detallen a la secció de línies futures (Secció 6.2).

6.2 Línies de futur

L'abast d'un TFM està fitat, principalment pel temps. Tot i que s'han aconseguit els objectius prevists, el sistema té molt de marge de millora.

Primer de tot, és necessari crear el *frontend* del sistema, perquè es pugui implementar en un entorn real. La implementació es pot fer en diferents entorns, com React, i sense gaire esforç es pot proporcionar una pàgina de visualització senzilla i còmoda.

Tot i que s'ha afirmat la viabilitat del sistema, un dels objectius del TFM era el d'afavorir el comportament ètic i la diversitat. Per assegurar la diversitat, és important assegurar la usabilitat de les persones sense recursos tecnològics, que permeten el seguiment dels registres. Llavors, s'haurien d'incorporar sistemes per poder avisar als usuaris del mal estat del seu medicament, a través de canals tradicionals, com una telefonada.

Està clar que es vol evitar que els usuaris s'autosubministrin fàrmacs negativament alterats. Així i tot, l'objectiu primordial és el de salvar el fàrmac, ja que aquest suposa molts de doblers. Per això, es considera implantar mesures de prevenció. Les dues principals consideracions són:

- Modificar el disseny perquè avisi a l'usuari abans d'arribar a l'estat de degradat. Així el mateix usuari pot prendre mesures per salvar el medicament, observant el gràfic de registres.
- Proporcionar geleres intel·ligents, connectades al sistema, amb capacitat d'augmentar o disminuir la temperatura, segons els registres.

Tot i que el fet de proporcionar geleres intel·ligents pareix un cost inassumible, es podria optar per petites geleres (de retorn), que només hauria de salvar un únic medicament per assegurar la seva rendibilitat.

Bibliografia

[1] do Pazo-Oubiña, F., Alorda-Ladaria, B., Gomez-Lobon, A. et al. Thermolabile drug storage in an ambulatory setting. Sci Rep 11, 5959 (2021). https://doi.org/10.1038/s41598-021-85413-0.

[2] Braune K, Kraemer LA, Weinstein J, Zayani A, Heinemann L. Storage Conditions of Insulin in Domestic Refrigerators and When Carried by Patients: Often Outside Recommended Temperature Range. Diabetes Technol Ther. 2019 May;21(5):238-244. https://doi.org/10.1089/dia.2019.0046.

[3] Genovard-Oliver, J., Frontera-Bergas, M., Vinaixa-Fernandez, M. et al. Monitoring Medicine Quality Conservation: An IoT-Based Platform Design. 2023 International Symposium on Networks, Computers and Communications (ISNCC), Doha, Qatar, 2023, pp. 1-8, doi: 10.1109/ISNCC58260.2023.10323904.

[4] Fitxer del diagrama GANTT. https://drive.google.com/file/d/1orF8BFhF-M3m_g8G9oU81DB-KyOmhs9w/view?usp=sharing.

[5] World Health Organization. WHO Technical Report Series, No.961, 2011. Annex 9 Model guidance for the storage and transport of time- and temperature-sensitive pharmaceutical products (2011).

[6] European Medicines Agency. Good distribution practice. https://www.ema.europa.eu/en/compliance-post-authorisation/good-distribution-practic.

[7] World Health Organization. TRS 961 - Annex 9, Supplement 6: Temperature and humidity monitoring systems for fixed storage areas Annex 9. WHO Technical Report Series, No. 961, 2011.

[8] Paula Otero Colmenar. Trazabilidad de las cadenas de frío de vacunas en Ethereum. TFM en Sistemas de Blockchain. UOC.

[9] Real Decreto 782/2013, de 11 de octubre, sobre distribución de medicamentos de uso humano, Spanish Government, 2013, BOE.

[10] Griggs, K.N., Ossipova, O., Kohlios, C.P. et al. Healthcare Blockchain System Using Smart Contracts for Secure Automated Remote Patient Monitoring. J Med Syst 42, 130 (2018). https://doi.org/10.1007/s10916-018-0982-x.

[11] Cbinsights, BlockMedx. https://www.cbinsights.com/company/blockmedx.

[12] Taralunga, D.D.; Florea, B.C. A Blockchain-Enabled Framework for mHealth Systems. Sensors 2021, 21, 2828. https://doi.org/10.3390/s21082828.

[13] Smart Contracts TFM control i gestio de medicaments. GitHub. https://github.com/josepgenovard/Smart-Contracts-TFM-control-i-gestio-de-medicaments.

[14] Agencia española de medicamentos y productos sanitarios. PROSPECTO Benepali 50mg solucion inyectable en pluma precargada. https://cima.aemps.es/cima/dochtml/p/1151074002/P_1151074002.html#5.

[15] Agencia española de medicamentos y productos sanitarios. FICHA TECNICA HYRIMOZ 40 MG SOLUCION INYECTABLE EN PLUMA PRECARGADA. https://cima.aemps.es/cima/dochtml/ft/1181286004/FT_1181286004.html.

[16] Agencia española de medicamentos y productos sanitarios. PROSPECTO XOLAIR 150 MG SOLUCION INYECTABLE https://cima.aemps.es/cima/dochtml/p/05319008/P_05319008.html.

Apèndix A

Taula de variables

La taula A.1 per entendre la notació i les diferents variables utilitzades.

Abans de mirar aquesta taula, és important aclarir algunes qüestions:

- Les variables que fan referència a paràmetres d'una estructura, no s'expliquen. Per exemple, es pot observar com les variables *ium* i *dataFi* de l'algoritme 6, fan referència als paràmetres de l'estructura *Associacio*, representada a la taula 3.2.
- Els *mappings* són cadenes de dades que s'utilitzen per trobar-ne d'altres. Per exemple, es pot guardar tota la variable medicament, i per accedir-hi el cercarem pel seu identificador. En el corresponent apartat s'indica que es cerca i que s'ha d'indicar per trobar la variables cercada.
- Les variables començades amb *aux*, fan referència a variables auxiliars d'una funció però no tenen una importància clau. Per això, tot i que no s'expliquen, es pot entendre la seva funció només amb el nom.

Nom / Símbol	**Descripció**
Símbols	
$a \leftarrow b$	El valor de b es guarda a la variable a.
a_b	Es selecciona el paràmetre b de la variables a.
$\# \ a$	Es crea la variable a.
$a[b]$	Es selecciona la posició b de l'array/mapping a.
$funcioExemple(a, b)$	Crida a la funció $funcioExemple$, passant-li per paràmetres les variables a i b.
$variable.lenght$	Retorna la llargària de la cadena de caràcters *variable*.
Variables comunes	
$account$	Adreça d'un actor o contracte. S'utilitza quan no es sap a qui pertany una adreça.
$aContractes$	Adreça d'algun contracte desplegat. Pot fer referència a un conjunt de contractes.
$aFarmaceuitc$	Adreça d'un farmacèutic.

aFarmacia	Adreça del contracte *Farmacia.sol*
aProcessador	Adreça del contracte *Processador.sol*
aSensor	Adreça d'un sensor.
aSensors	Adreça del contracte *Sensors.sol*
aUsuari	Adreça d'un usuari.
idAssociacio	Identificador d'una associació.
owner	Adreça de l'autoritat.
msg.sender	Adreça que crida la funció. Pot ser un actor o un contracte.
now	Data en format *timestamp* de l'instant en que s'executa el codi.
totalAssociacions	Número d'associacions totals. Aquesta variables serveix d'identificador incrementador.
totalNotificacions	Número de notificacions totals. Aquesta variables també serveix d'identificador incrementador.
totalRegistres	Número de registres de temperatura totals. Aquesta variables també serveix d'identificador incrementador.
Estructures de dades	
a	Associació. Taula 3.2.
f	Farmàcia. Taula 3.1.
n	Notificació. Taula 3.5.
r	Registre. Taula 3.4.
s	Sensor. Igual que taula 3.2.
u	Usuari. Igual que taula 3.2.
Mappings	
associacioDeSensor	Identificador de la darrera associació d'un sensors, amb la seva adreça.
associacionsDeUsuari	Identificadors de totes les associacions que ha tingut un usuari, amb la seva adreça.
assocacionsMap	Associació amb el seu identificador.
farmaceuticsMap	Farmacèutics amb la seva adreça.
sensorsMap	Sensors amb la seva adreça.
medicamentsMap	Medicament amb el seu identificador, *ium*.
notificacionsDeUsuari	Identificadors de totes les notificacions a un mateix usuari, amb la seva adreça.
notificacionsMap	Notificació amb el seu identificador.

Taula A.1: Notació dels algoritmes i explicació de les variables.