Miriam Gertrudix Pedrola

DISSENY I IMPLEMENTACIÓ D'UNA API REST PER CONNECTAR UN SISTEMA EMBEDDED AL NÚVOL.

TREBALL DE FI DE GRAU

dirigit per Marc Sánchez Artigas

Grau d'Enginyeria Informàtica/Telemàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2023

Resum.

Durant els darrers anys, la quantitat de dades i informació que utilitzen diàriament les empreses en el sector tecnològic ha augmentat de manera exponencial. Això ha causat que moltes d'aquestes empreses s'hagin hagut d'adaptar, ja que els sistemes d'emmagatzematge que usaven fins al moment eren poc escalables o s'estaven quedant obsolets. Una de les solucions més usades és el Cloud Computing, és a dir, fer servir un servei virtual per a emmagatzemar aquestes dades. Aquest era un dels objectius que tenia l'empresa Quercus Technologies quan vaig començar a formar-hi part.

En aquest treball s'ha dissenyat i implementat una Api REST per tal de connectar el producte LPR5, una càmera de detecció de matrícules, al núvol. Per fer-ho he fet un petit estudi previ per demostrar els avantatges que ens suposaria usar Google Cloud Platform per emmagatzemar les dades generades. També he explicat quins serveis d'aquesta plataforma necessitem que interactuï la nostra API i com ho ha de fer tenint en compte el tipus de dades que genera i consumeix el producte.

Per últim, he fet una comparació d'aquest nou sistema amb el què ja estava implementat anteriorment en aquest producte, per poder confirmar-ne la millora.

Resumen.

Durante los últimos años, la cantidad de datos e información que utilizan diariamente las empresas del sector tecnológico ha aumentado de forma exponencial. Esto ha causado que muchas de estas empresas se hayan tenido que adaptar, ya que los sistemas de almacenamiento que utilizaban hasta este momento eran poco escalables o se estaban quedando obsoletos. Una de las soluciones más usadas ha sido el Cloud Computing, es decir, utilizar un servicio virtual para almacenar estos datos. Este era uno de los objetivos que tenía la empresa Quercus Technologies cuando empecé a formar parte de ella.

En este trabajo se ha diseñado e implementado una Api REST para conectar el producto LPR 5, una cámara de detección de matrículas, a la nube. Para hacerlo he hecho un pequeño estudio previo para demostrar las ventajas que nos supondría usar Google Cloud Platform para almacenar los datos generados. También he explicado que servicios de esta plataforma necesitamos que interactúe nuestra API y como lo tiene que hacer teniendo en cuenta el tipo de datos que genera y consume el producto.

Por último, he hecho una comparación de este sistema nuevo con el que ya estaba implementado previamente en este producto, para demostrar la mejora.

Abstract.

In the last few years, the amount of data and information that technology enterprises use everyday has increased exponentially. As a consequence a lot of these enterprises have had to adapt, because their storage systems were poorly scalable or have become out of date.One of the most used solutions in this scenario is the Cloud Computing, in other words, to use a virtual service to store all this data. That was one of the goals of the Quercus Technologies team before I became part of it.

In this paper I have designed and implemented an API REST to connect the LPR5 product, a license detection camera, to the cloud. To do that I have done previous research to demonstrate the advantages of using Google Cloud Platform to store the data generated. I also explained which platform services I used in order to work with the type of data that LPR5 generates and consumes.

Finally, I made a comparison between the new and the old system, to prove its improvement.

Índex de taules

Agraïments **5**
1. Introducció **6**
1.1 Descripció general situació previa 6
1.2 Definició del problema 10
1.3 Proposta de solució 11
1.4 Objectius 12
2. Disseny i Arquitectura **13**
2.1 Anàlisi dels requisits 13
2.1.1 Requisits funcionals 13
2.1.2 Requisits no funcionals 14
2.1.3 Casos d'ús 14
2.2 Arquitectura proposada 19
2.3 Decisions de disseny 22
2.3.1 Adreces Macs 22
2.3.2 Unitats LPR5 23
2.3.3 Producció 24
2.3.4 QC 25
3. Estudi de l'infrastructura **27**
3.1 LLenguatges Implementació 27
3.1.1 Java[1] 27
3.1.2 Node JS[2] 27
3.1.3 Resultat 27
3.1.4 Llibreries utils 28
3.2 Cloud Computing 28
3.2.1 Oferta de serveis 29
3.2.2 Preu 29
3.2.3 Regions disponibles 29
3.2.4 Suport i documentació 29
3.3 Google Cloud Platform 30
3.3.1 Entorns de desenvolupament 30
La Configuració d'aquests entorns s'explicarà en l'apartat d'implementació 31
3.3.2 Serveis utilitzats 31
4. Implementació **33**
4.1 Configuració de l'entorn de GCP 33
4.1.1 Authentication 33
4.1.2 Cloud Run 35
4.1.3 API Gateway 37
4.1.4 Firestore i Storage 37
4.2 Implementació del codi de la API 40
4.2.1 Estructura 40

4.2.2 GCP middleware 48
4.2.3 API Endpoints 64
4.3 Interfície gràfica. 75
5. Avaluació **78**
5.1 Servei actual 78
5.1.1 Proves 78
5.1.2 Posada a producció 79
5.2 Servei actual en comparació de l'antic 80
6. Conclusions **81**
6.1 Treball a futur i coses a millorar 81
7. Annex A **83**
7.1 API Gateway fitxer de configuració 83
7.2 Fitxer de configuració de la base de dades 91
7.3 Fitxer de configuració de la base de Storage 96
7.4 Fitxer controllers/productions.js 102
7.5 Fitxer controllers/qcs.js 110
8. Referències **117**

Índex de figures

Figura 2: Camara Smart LPR 5 7
Figura 3: Esquema d'un punt de producció LPR 5 9
Figures 4 i 5: Zona habilitada per al control de qualitat a Reus. 10
Figura 6: Esquema de la solució 11
Figura 7: Diagrama de casos d'ús del projecte 15
Figura 8: Disseny del projecte 20
Figura 9: Diagrama de classes de l'aplicació 22
Figura 10: Esquema dels endpoints per a gestionar el rang d'adreces MAC. 23
Figura 11: Esquema dels endpoints per a gestionar les unitats LPR5.. 24
Figura 12: Esquema dels endpoints per a gestionar les produccions. 25
Figura 13: Esquema dels endpoints per a gestionar els QC. 26
Figura 14: Esquema dels entorns de Google Cloud Platform. 31
Figura 15: Esquema dels serveis de Google Cloud Platform utilitzats. 32
Figura 16: Estructura de l'API gateway 34
Figura 17: Página de configuració de la service account 35
Figura 18: Pàgina de configuració de CloudRun 36
Figura 19: Exemple de definició del endpoint al fitxer de configuració de l'API Gateway. 37
Figura 20: Exemple de document LPR_PS_MACRANGE 38
Figura 21: Exemple de document LPR_PS_SN 39
Figura 22: Exemple de document LPR_PS_PRODUCTIONS 39
Figura 23: Exemple de document LPR_PS_QC 40
Figura 24: Exemple de Cloud Storage 40
Figura 25: Estructura fitxers de l'aplicació 41
Figura 26: Fragment del fitxer de configuració de la API 43
Figura 27:Fitxer de configuració de les rutes routes.js 45
Figura 28:Fitxer routes/macRanges.js 46
Figura 29:Fitxer routes/LPR5.js 48
Figura 30: Fragment del fitxer db.js metode getDB 48
Figura 31: Fragment del fitxer db.js mètodes getAllDocumentsDB i getAllDocumentsDBWithoutPagination 51
Figura 32: Fragment del fitxer db.js mètodes getAllDocumentsByQueryDB i getAllDocumentsByQueryDBWithoutPagination 53
Figura 33: Fragment del fitxer db.js mètode setDB 54
Figura 34: Fragment del fitxer db.js mètode setNoIdDB 54
Figura 35: Fragment del fitxer db.js mètode updateDB 55
Figura 36: Fragment del fitxer db.js mètode getCountDocumentsDB 55
Figura 37: Fragment del fitxer storage.js mètode uploadProduction 58
Figura 38: Fragment del fitxer storage.js mètode uploadQC 59
Figura 39: Fragment del fitxer storage.js mètode downloadProduction 62

Figura 40: Fragment del fitxer storage.js mètode downloadQC 64
Figura 41: Exemple de dades del body que passem per publicar una producció 65
Figura 42: Fragment del fitxer controllers/productions.js mètode createProduction 69
Figura 43: Fragment del fitxer controllers/qcs.js mètode getQc 73
Figura 44: Fragment del fitxer controllers/productions.js mètode addProdLog 74
Figura 45: Fragment del fitxer controllers/qcs.js mètode downloadQcImages 75
Figura 46: Página de login QuercusApp 75
Figura 47: Página de Productions QuercusApp 76
Figura 48: Página de QC QuercusApp 76
Figura 49: Página de MACs QuercusApp 77
Figura 50: Prova de crear un rang d'adreces MAC 78
Figura 51: Prova de crear una producció LPR5 78
Figura 52: Prova de crear un QC LPR5 79
Figura 53: Prova gestió d'adreces MAC lliures 79

Agraïments

M'agradaria agrair al meu tutor d'aquest treball en Marc Sánchez per tot l'assessorament que he obtingut. Als meus companys de la carrera, perquè m'han ajudat a fer el camí una mica més planer durant aquests anys i els estaré eternament agraïda. Als companys de Quercus, per ajudar-me a créixer en el món laboral i per tota l'ajuda rebuda en aquest treball, sobretot al Santi Pàmies, el Sergi Fibla, l'Stefano Zuin i el Marc Roca. A l'Adrià Martínez per ser el millor mentor que em podria trobar i per agrair-li també tot el seu suport en aquest projecte. I finalment als meus pares i parella per creure en mi i per donar-me suport des del primer dia en aquesta carrera.

1. Introducció

Durant els darrers anys, la quantitat de dades i informació que utilitzen diàriament les empreses en el sector tecnològic ha augmentat de manera exponencial. Això ha causat que moltes d'aquestes empreses s'han hagut d'adaptar, ja que els sistemes d'emmagatzematge que usaven fins al moment eren poc escalables o s'estaven quedant obsolets.

És per això que fa una mica més d'una dècada es va començar a parlar de Computació al núvol, tot i que no ha sigut fins als darrers anys que ha pres un paper molt important en el sector tecnològic.

Quercus Technologies és una de les empreses punteres en el món de sistemes de guiatge en pàrquings i els seus dispositius estan desplegats arreu del món. El seu principal producte es diu Smart LPR 5, és una càmera de detecció i reconeixement de matrícules. Aquest producte es produeix principalment en la seva seu central de producció a Reus, tot i que a vegades és necessari produir dispositius a la resta de seus. Arriba un punt en què el seu sistema de producció queda antiquat i neix una problemàtica que necessita una renovació per tal de garantir un correcte emmagatzematge de les dades de producció, i en diferents punts de producció.

Aquest sistema de producció ha de garantir que es puguin muntar diferents punts de producció arreu del món i que, tot i això, totes les dades quedin enregistrades en un mateix lloc.

1.1 Descripció general situació previa

Per donar una mica de context al lector, en el següent punt es dóna més informació sobre producte LPR5 i com es produeix.

1.1.1 Smart LPR 5

Smart LPR 5 o simplement LPR 5 és un dels productes més produïts a Quercus Technologies. Es tracta d'una càmera de reconeixement de matrícules *All-in-one*. Això vol dir que tot l'equipament necessari perquè pugui reconèixer i processar la matrícula està integrat en la mateixa unitat. El reconeixement de les matrícules és una de les parts més importants en sistemes de pàrquing dissenyats per a tenir un control dels vehicles dintre de les instal·lacions.

Figura 2: Camara Smart LPR 5

El procés de producció d'una unitat LPR 5 es divideix en tres parts, arrancar el servidor DHCP, arrancar el LPR 5 Flasher i per últim passar el QC test.

El primer pas és arrancar el servidor DHCP. Aquest servidor simplement facilita una IP a la unitat.

Després comença el procés de Flashejar la unitat, aquest procés és el que s'encarrega d'instal·lar el Firmware a la unitat.

Un cop s'ha creat una producció LPR 5 ha de passar un test de control de qualitat abans d'enviar aquesta unitat al client. Aquest control de qualitat o QC[1] comprova aquests àmbits:

- Reconigtion: Controla que detecti i llegueixi bé la matricula.
- Lamp: Comprova que la camera funciona tant en RGB[2] com en Blanc i negre.

[1] QC: Quality Control.
[2] RGB: Red Green Blue, fa referència a imatges en color.

- Zoom: Comprova que la camera tingui el zoom adequat per a detectar les matricules.
- Focus: Comproba que la camera estigui ben enfocada.
- Iris: Comproba que la brillantor sigui l'adequada.
- Imputs/Ouputs: es comprova que les entrades i sortides de la unitat funcionen bé.
- Wiegand[3]: es comprova que la conexió es correcta.

Aquest procés es duu a terme a les instal·lacions situades a Reus, on es troba el punt de producció principal i totes les unitats es produeixen allí. Tot i això, en el moment de la instal·lació es poden generar problemes que requereixin tornar a flashejar la unitat. Si aquesta situació és donada, depenent de la localització del pàrquing, enviar la unitat a les instal·lacions de Reus per a flashejar la unitat i tornar-la a enviar al pàrquing podria tardar dies, fins i tot mesos, a part que és una despesa innecessària. És per això que existeixen les RMA o punts de reparació. Són com petits punts de producció donats a les diferents seus que Quercus té fora d'Espanya, que únicament s'utilitzen per a aquests casos.

1.1.2 Procés de producció

1.1.2.1 Entorn de desenvolupament

Per a poder produir unitats LPR 5, es necessita un ordinador amb un sistema operatiu específic. En aquest ordinador és colona un repositori Git LFS, aquests tipus de repositoris estan preparats per a treballar amb fitxers molt grans, i el que fan és utilitzar referències al fitxer en comptes del mateix fitxer. En aquest repositori trobem tots els fitxers necessaris per a la producció d'una unitat.

Entre aquests fitxers tenim scripts, fitxers en format cfg que ens proporcionen dades de configuració, i fitxers de text on registrar dades generades durant el procés de producció.

1.1.2.2 Configuració d'una unitat LPR 5

Per a la configuració d'una unitat LPR 5 tenim diferents scripts:
El primer que s'executa és **launchDhcp.sh.**

Des del moment que s'inicia la unitat per primer cop, fins que no supera el QC la unitat no té una IP fixa. Aquest script s'encarrega d'aixecar el servidor DHCP a l'ordinador local sobre el port ethernet on es connecta la unitat per tal d'assignar-li una adreça IP.

Després tenim el **launchLPR5Flasher.sh.** Aquest script instal·la el Firmware a la unitat seguint els següents passos:
S'ha d'indicar el nom del departament que està produint. Aquest quedarà registrat com a producer. Després també haurem de definir la llicència, el

[3] Wiegan: L'interficie Wiegand es un estandar de cablejat utilitzat per interconectar periferics.

LPRCountries, el número de sèrie, model de la unitat i quantitat d'unitats a produir.

Un cop definides aquestes dades, es connecta la unitat i es produeix l'adquisició del Firmware.
Agafa una adreça MAC disponible del fitxer macs.cfg i s'envia a la unitat i llavors aquesta ja està llesta per instal·lar el firmware.
Una vegada es finalitza el procés, escriu les dades especificades en el fitxer serialsLPR5.txt amb el següent format:

yyyy-mm-dd hh:mm:ss | LPRCountries | Versió de Firmware | Número de serie | Adreça MAC

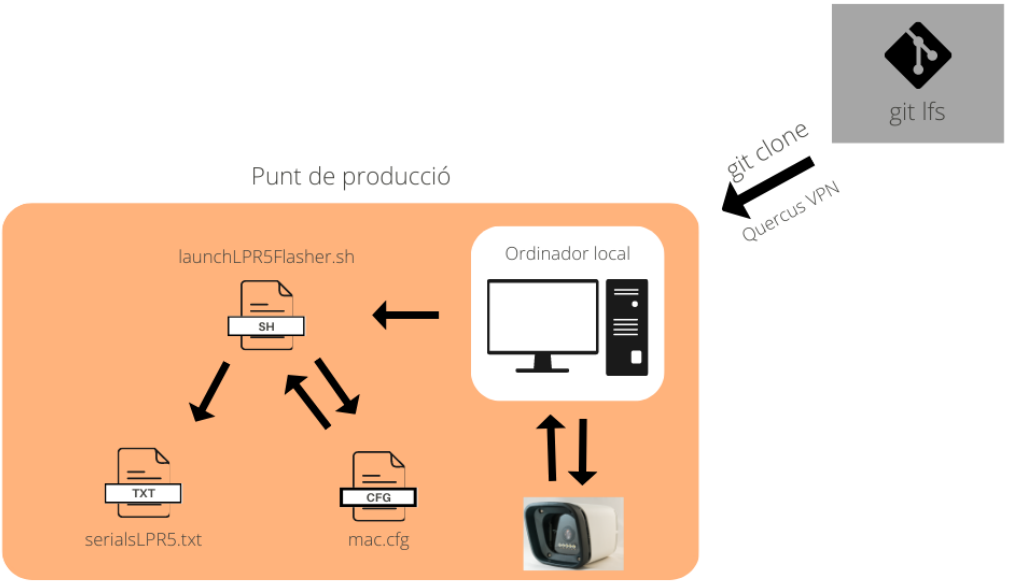

Figura 3: Esquema d'un punt de producció LPR 5

Després la unitat es reinicia i entra en mode QC i, per tant, ja està llesta per passar el control de qualitat.

1.1.2.3Control de Qualitat

Aquest procés requereix fer-se en un espai habilitat en la zona de producció preparada amb unes condicions ideals per tal de poder fer proves de reconeixement.


Figures 4 i 5: Zona habilitada per al control de qualitat a Reus.

Per a passar el test de QC executem **launchLPR5QC.sh**, que implementa el model de test de l'ISTQB[4]. Primer de tot ens demana el port de xarxa on està connectada la unitat, i ens tornarà a demanar el nom del departament que està passant el test. També caldrà passar-li la matrícula amb la qual es realitzen les proves i una vegada té totes aquestes dades podem començar amb el test.

El test consisteix a validar que fa el reconeixement d'una matrícula correctament, que la lent està ben enfocada amb un zoom adequat. Comprova que la càmera funciona tant en RGB com en Blanc i negre i també en comprova la brillantor de la imatge. Finalment, també comprova que les entrades i sortides de la unitat són correctes i que el protocol Wiegan funcioni correctament.

Si el test ha estat validat correctament, es genera un fitxer de log on la unitat informa que ha estat validada i ja és operativa.

1.2 Definició del problema

Aquesta organització planteja una sèrie de problemes que s'havien de resoldre:

- **Únic punt de producció registrat:** Que es treballi amb fitxers locals dificulta el fet de crear un punt de muntatge nou com podria ser un RMA. Quan hi ha la necessitat de muntar un RMA a una seu diferent de la de Reus

[4] ISTQB: International Software Testing Qualifications Board.

es torna una feina molt complicada i sovint pot arribar a crear conflictes, com per exemple unitats que han passat flasher, però no s'han registrat.

- **Descentralització de la informació:** En cas de buscar una solució al punt anterior, quan es necessita un RMA en una altra seu es crea un fitxer serials.txt i un mac.cfg nous per aquell punt de producció. Això crea una descentralització les dades, que podria ser fatal si no hi ha una bona comunicació entre els diferents punts.
- **Seguretat de les dades:** Si tot el sistema de producció depèn d'un sol ordinador i aquest ordinador en algun punt falla, no sols significaria que s'han perdut dades de producció, sinó que també significa la parada de producció.

1.3 Proposta de solució

La proposta al problema que hem definit anteriorment es tracta de crear una API REST que connecti el producte amb una plataforma al núvol. D'aquesta manera el personal de l'empresa tindrà un fàcil accés a les dades de producció indiferentment de la localització.

Aquesta API podrà publicar dades de producció en una base de dades al núvol així com fitxers si escau. Però també permetrà llegir aquestes dades, és a dir, totes les accions que fins ara es feien en local ara passaran a fer-se al núvol.

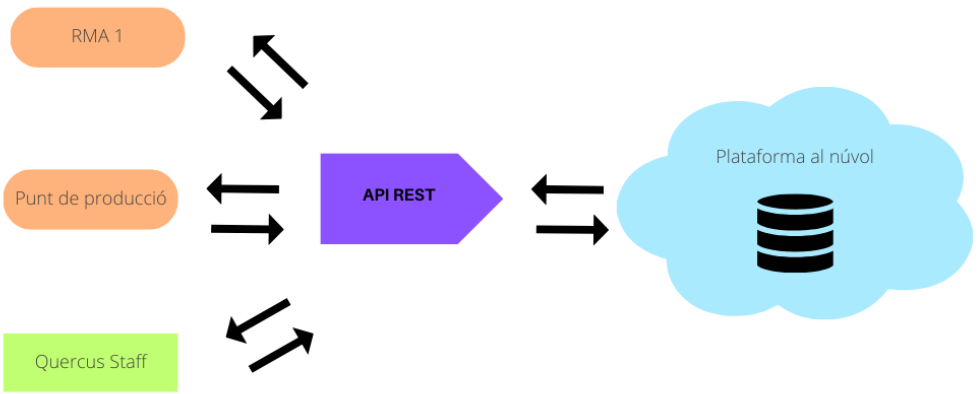

Figura 6: Esquema de la solució

1.4 Objectius

Un cop donat el context, els objectius principals en aquest projecte són els següents:

- **Centralitzar les dades:** Hem de trobar una manera de tenir totes les dades en un únic lloc i que sigui accessible per als membres de l'empresa que ho necessitin. D'aquesta manera tenir dades reals sobre la producció i unitats refetes.
- **Automatitzar processos que abans s'havien de fer manualment:** D'aquesta manera a part d'estalviar-nos errors humans que hi pugui haver en crear un punt de producció RMA nou, també evitem tenir una persona fent aquesta tasca.
  Per fer-ho cal trobar quina plataforma al núvol s'adapta millor al nostre producte, tenint en compte quines dades i de quin tipus haurem d'emmagatzemar.
  També haurem de dissenyar com ha de ser l'API REST perquè l'accés a aquestes dades sigui ràpid i efectiu. Tindrem diferents tipus d'endpoints:
  - Pujada: per a penjar dades en el format que vulguem.
  - Consulta: per consultar dades.
  - Baixada: per a descarregar dades o fitxers.

Finalment, caldrà demostrar que hi ha hagut una millora en consideració amb el sistema anterior. Busquem millores en rapidesa i en cost.

El desenvolupament d'aquest treball el dividirem en tres parts:

- Estudi i explicació de les tecnologies utilitzades.
- Disseny i implementació de l'API
- Test de demostració

2. Disseny i Arquitectura

En aquesta secció del projecte es detalla quin ha sigut el plantejament proposat per a assolir els objectius proposats.

En aquest sistema tindrem dos actors claus que seran els que executaran les accions. Per tal d'entendre bé el funcionament del sistema explicaré breument els dos actors:

- **Production System:** Aquest sistema és el que es llança des del punt de producció i el que executa els scripts explicats anteriorment. Totes les accions que es duguin a terme durant el procés de producció, les executa el production system.
- **Quercus Staff:** Aquest actor és qualsevol treballador de Quercus, per tant, podrà realitzar en gran part accions de lectura de dades.

Un cop definits aquests actors veurem quines accions necessiten dur a terme per tal de fer un disseny de la nostra API.

2.1 Anàlisi dels requisits

Una vegada estudiat el sistema de producció del LPR 5, fem una anàlisi dels requisits que ha de complir l'API que hem de dissenyar. Per fer-ho he dividit els requisits en dos tipus, els funcionals i els no funcionals.

2.1.1 Requisits funcionals

R1	Un usuari de Quercus ha de poder crear un rang d'adreces MAC.
R2	Un usuari de Quercus ha de poder llistar els rangs d'adreces Mac indistintament de si queden adreces lliures o no.
R3	Un usuari de Quercus 'ha de poder veure quantes adreces lliures queden al sistema.
R4	El sistema de producció ha de poder reservar una adreça MAC per a assignar-li a una unitat.
R5	El sistema de producció ha de poder registrar produccions de cada unitat. I també ha de poder emmagatzemar els fitxers de log que genera la unitat.
R6	Un usuari de Quercus ha de poder llistar les produccions creades i poder descarregar els fitxers de log.
R7	El sistema de producció ha de poder registrar que una unitat s'ha sotmès al test de QC, l'hagi passat o no. També ha de poder

	emmagatzemar els fitxers de log que genera el QC, així com les imatges que ha fet durant el test.
R8	Un usuari de Quercus ha de poder llistar els QCs i s'ha de poder descarregar els fitxers de logs i les imatges.

2.1.2 Requisits no funcionals

R1	Les peticions de l'API REST han de ser gairebé immediates, així que hi ha d'haver una latencia molt baixa.
R2	Qualsevol dada que es modifiqui a la base de dades s'ha de fer en menys de dos segons per evitar conflictes entre unitats.
R3	El sistema s'ha d'assegurar que només els membres de quercus tinguin accés API REST.

2.1.3 Casos d'ús

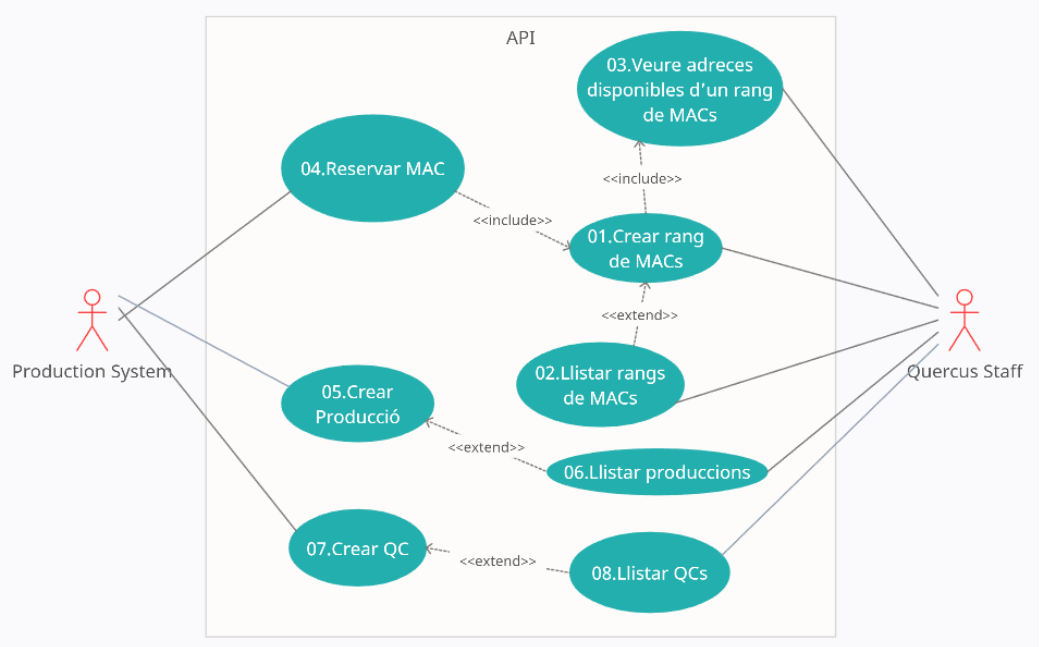

Figura 7: Diagrama de casos d'ús del projecte

Cas d'ús 01. Crear un rang de MACs.

Resum de la funcionalitat: Es crea un rang amb les adreces MAC que hem comprat.

Paràmetres d'entrada:

Paràmetres de sortida: cap

Usuari: Quercus staff.

Precondició: cap

Procés normal principal:

1. Comprovar si ja existeix a la base de dades el rang que pretenem guardar.

Alternatives de procés i excepcions:

1a. El rang proposat no existeix a la base de dades, per tant, el podem guardar.

1b. El rang suggerit ja existeix a la base de dades, mostrem un missatge informant-ho.

Cas d'ús 02. Llistar rangs de MACs.

Resum de la funcionalitat: Llista tots els rangs de MACs que s'han registrat a la base de dades.

Paràmetres d'entrada: cap.

Paràmetres de sortida: Llista de tots els rangs de macs en format JSON.

Usuari: Quercus staff.

Precondició:

Procés normal principal:

1. Agafa tots els rangs d'adreces MAC del sistema.

Alternatives de procés i excepcions:

1a. Si existeixen rangs de MAC, els retorna.

1b. Si no existeixen, retorna una llista buida.

Cas d'ús 03. Veure Adreces disponibles.

Resum de la funcionalitat: Retorna la quantitat d'adreces MAC lliures hi ha al sistema.

Paràmetres d'entrada: Cap

Paràmetres de sortida: Quantitat d'adreces lliures

Usuari: Quercus staff.

Precondició: Cap

Procés normal principal:

1. Agafem totes les adreces lliures de la base de dades.
2. Les contem.
3. Retornem la quantitat obtinguda.

Alternatives de procés i excepcions:

Cas d'ús 04. Reservar MACs.

Resum de la funcionalitat: reserva una adreça MAC per a assignar-la a una unitat LPR5.

Paràmetres d'entrada: cap

Paràmetres de sortida: adreça MAC lliure.

Usuari: Production system.

Precondició: cap

Procés normal principal:

1. Comprovar si hi ha cap adreça MAC lliure

Alternatives de procés i excepcions:

1a. Si no hi ha adreces lliures, mostrem un missatge informant-ho i acabem el procés.

1b. Si hi ha adreces disponibles, actualitzem la nova adreça lliure i l'actual la retornem com a paràmetre de sortida.

Cas d'ús 05. Crear una Producció.

Resum de la funcionalitat: Registrar una nova producció d'una unitat al sistema per guardar la seva informació a la base de dades

Paràmetres d'entrada: producer, data, license, lprCountries, MAC address, result, Firmware version.

Paràmetres de sortida: cap

Usuari: Production system.

Precondició: cap

Procés normal principal:

1. Comprovar que els paràmetres d'entrada no estan buits i son el format correcte.

2. Comprovar que l'adreça MAC estigui dins d'un rang correcte i no estigui utilitzada

3. Guardem les dades a la base de dades.

4. Comprovar si aquesta unitat ja existeix a la base de dades.

Alternatives de procés i excepcions:

1a. Si estan buits o amb un format erroni, informem l'usuari i acabem el procés.

1b. Si les dades són correctes, continuem amb el procés.

2a. Si no és dins d'un rang correcte o no està lliure, informem l'usuari i acabem el procés.

2b. Si és dins del rang i és lliure continuem amb el procés.

4a. Si la unitat ja existeix, afegim la id de la producció a la seva entrada de la base de dades.

4b. Si la unitat no existeix, la creem i afegim la id de la producció.

Cas d'ús 06. Llistar produccions.

Resum de la funcionalitat: Llistar totes les produccions que s'han registrat a la base de dades.

Paràmetres d'entrada: filtres (opcional)

Paràmetres de sortida: Llista de produccions.

Usuari: Quercus staff.

Precondició: cap

Procés normal principal:

1. Comprovar si ens han passat filtres per paràmetre.
2. Comprovar si hi ha produccions registrades a la taula de produccions de la base de dades.

Alternatives de procés i excepcions:

1a. Si ens han passat filtres, els apliquem per a agafar les dades de la taula de produccions de la base de dades.

1b. Si no ens han passat cap filtre, agafem totes les dades de la taula de produccions de la base de dades.

2a. Si està buida, retornem una llista buida.

2b. Si hi han dades, les posem en una llista i les retornem.

Cas d'ús 07. Crear un QC.

Resum de la funcionalitat: Registrar quan una unitat ha llençat un test QC al sistema per guardar la seva informació a la base de dades

Paràmetres d'entrada: producer, data, resultat i número de sèrie.

Paràmetres de sortida: cap

Usuari: Production system.

Precondició:

Ha d'existir una producció amb el mateix número de sèrie a la base de dades.

Procés normal principal:

1. Comprovar que els paràmetres d'entrada no estan buits i són el format correcte.
2. Comprovar que existeix una unitat produïda amb el mateix número de sèrie.
3. Guardar les dades a la base de dades.
4. Afegim la id del QC a l'entrada de la base de dades de la unitat.

Alternatives de procés i excepcions:

1a. Si estan buits o amb un format erroni, informem l'usuari i acabem el procés.

1b. Si les dades són correctes, continuem amb el procés.

2a. Si no existeix, informem l'usuari i acabem el procés.

2b. Si existeix, continuem amb el procés.

Cas d'ús 08. Llistar QCs.

Resum de la funcionalitat: Llistar tots els QCs que s'han registrat a la base de dades.

Paràmetres d'entrada: filtres (opcional)

Paràmetres de sortida: Llista de QCs

Usuari: Quercus staff.

Precondició: cap

Procés normal principal:

1. Comprovar si ens han passat filtres per paràmetre.

2. Comprovar si hi ha produccions registrades a la taula de produccions de la base de dades.

Alternatives de procés i excepcions:

1a. Si ens han passat filtres, els apliquem per a agafar les dades de la taula de QCs de la base de dades.

1b. Si no ens han passat cap filtre, agafem totes les dades de la taula de QCs de la base de dades.

2a. Si està buida, retornem una llista buida.

2b. Si hi han dades, les posem en una llista i les retornem.

2.2 Arquitectura proposada

A continuació es mostra un dibuix i explicaré quina ha sigut l'arquitectura que hem utilitzat. La fletxa marcada amb el número 1 defineix la comunicació entre el sistema de producció i la nostra aplicació. Aquesta comunicació estarà centrada en les peticions més importants. La gran majoria seran d'escriptura, ja que seran peticions més crítiques i estaran reservades a un grup concret de Quercus.

Amb el número 2 podem veure la comunicació de la resta de l'empresa, per mitjà d'una interfície gràfica amb l'aplicació que he implementat. Aquesta comunicació donarà pas majoritàriament a peticions de lectura de dades o petites configuracions. Tothom que treballa a Quercus hi tindrà accés, ja que seran operacions de consulta d'informació.

Finalment, la comunicació número 3 és la que es dóna entre l'aplicació i el núvol. Aquesta comunicació s'emprarà per emmagatzemar o demanar dades o fitxers al núvol.

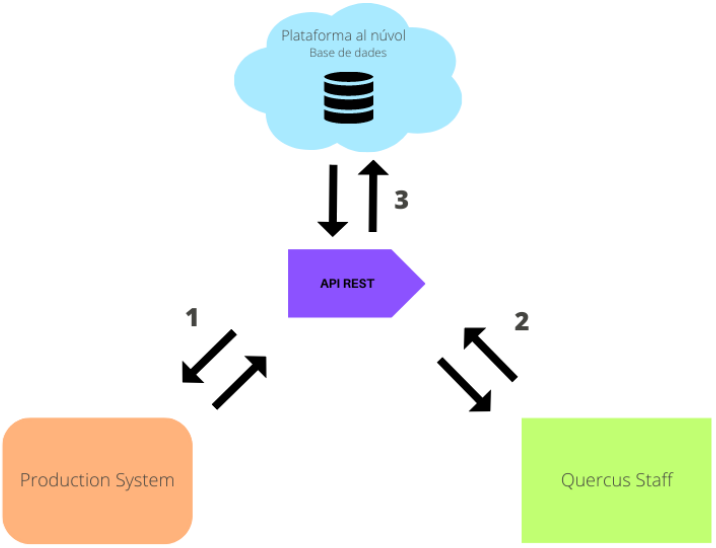

Figura 8: Disseny del projecte

Després de fer un estudi dels casos d'ús i de veure quines accions ha de complir la nostra aplicació, podem dividir totes aquestes accions en 4 grups clars:

- **LPR5**: És la divisió més general. Aquesta divisió s'encarrega del tractament de la informació de les mateixes unitats LPR5.

- **MACS**: Darrere de la producció d'una unitat LPR5 hi ha una adreça MAC que és necessària de tenir en compte, això provoca que existeixi una gestió de rangs i reserves d'adreces MAC. Aquesta gestió es tracta en aquesta divisió.

- **Producció:** En aquesta secció ens cuidem de totes les accions relacionades amb la producció d'una unitat, exceptuant el tractament de MACS que fa la secció anteriorment esmentada.

  Quan es crea una producció s'han d'emmagatzemar les dades següents:

  - L'identificador de la producció (ha de ser autogenerada i no es pot repetir).
  - La llicència de la unitat.

- La llibreria de reconeixement.
- L'adreça MAC.
- El departament que ha creat la producció (per saber si sa creat en un punt de producció o en un RMA).
- Resultat del procés de creació.
- El número de sèrie.
- La versió del Firmware.
- La data en què s'ha produït.

- **QC:** Tant si ha passat el Quality Control com si no , ha de quedar registrat que s'ha realitzat el test. La informació sobre aquest test de control de qualitat la tracta aquesta divisió.

  Quan es registra un QC s'han d'emmagatzemar les següents dades:

  - Identificador de QC (ha de ser autogenerada i no es pot repetir).
  - Departament que ha executat el QC.
  - Resultat del QC.
  - Número de sèrie.
  - Data en la qual s'ha passat el QC.

D'aquestes divisions també en podem treure una relació entre elles. La divisió de LPR5 ha de guardar una llista amb les instàncies de produccions i de QC que es creen de cada unitat. Aquí tenim el diagrama de classes per entendre una mica millor quina informació ha de guardar cada divisió.

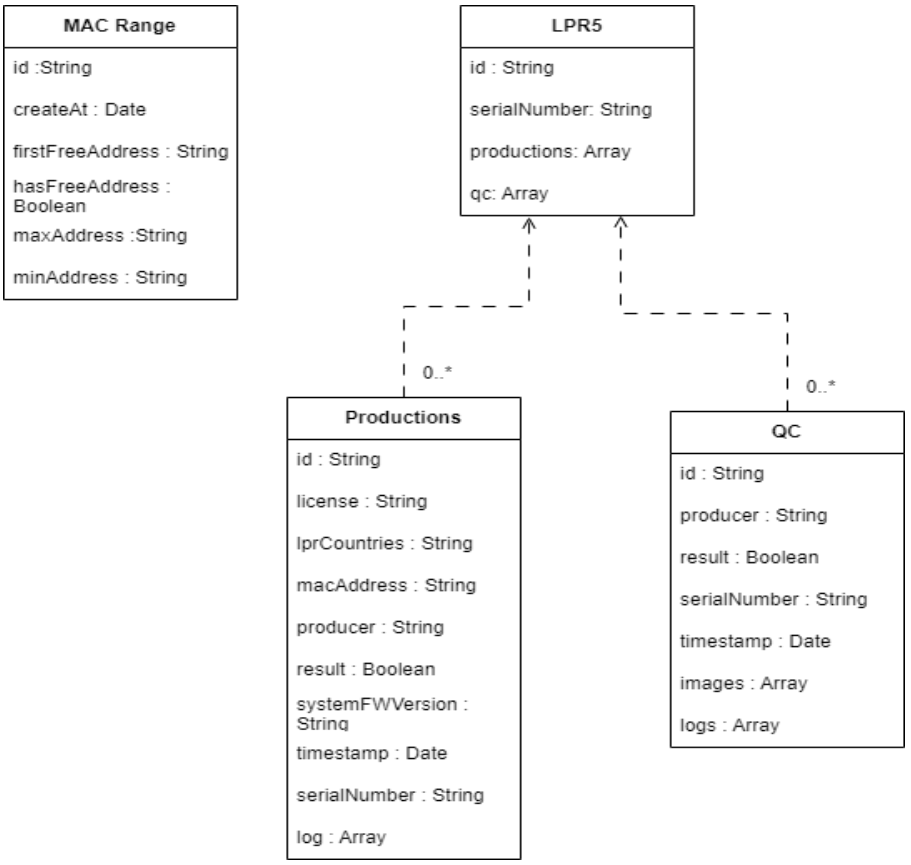

Figura 9: Diagrama de classes de l'aplicació

2.3 Decisions de disseny

En aquest apartat explicaré quins són els endpoints proposats per a l'API i que ha de fer cada un d'ells.

2.3.1 Adreces Macs

Per a gestionar les adreces MAC, tenim un recurs amb la ruta /macRanges.

Si fem un GET a aquesta ruta, ens retorna tots els rangs d'adreces MAC que hem creat al sistema, tant si n'hem gastat totes les adreces com si encara en té de lliures. Ho podrem fer de dues maneres, si li passem un paràmetre a la query indicant el nom d'un camp de la taula Mac a la base de dades, ens filtrarà la llista de rangs de MACs per aquest camp, mentre que si no li proporcionem aquest paràmetre, simplement ens agafarà totes les entrades de la base de dades. També li podem especificar quantes entrades volem que ens retorni i a partir de quina entrada volem que les comenci a llistar.

Si fem POST en aquesta mateixa ruta, ens permet crear un rang d'adreces nou. Per fer-ho haurem d'enviar un body amb una adreça màxima i una adreça mínima en

format JSON. Primer farà una validació si s'han passat les dades esperades i si estan en el format correcte. Després es fa una altra validació per assegurar-nos que no existeix un rang que contingui aquestes adreces a la base de dades, i finalment es valida que l'adreça mínima sigui més petita que la màxima. Una vegada ha passat totes les validacions guardarà el nou rang a la base de dades.

Un altre recurs amb la ruta /macRanges/macs/count el qual només es pot fer un GET, ens retornarà quantes adreces disponibles ens queden al sistema. Per fer-ho agafa tots els rangs existents i els recorre un a un. Si la variable *hasFreeAddress* és igual a false, passa al següent rang. Si és true, resta de l'adreça màxima la següent adreça disponible i això li dona la quantitats d'adreces disponibles d'aquell rang. Finalment, es fa un sumatori de les adreces disponibles de tots els rangs i es retorna aquest valor.

Finalment, el recurs amb la ruta /macs/reserve, fent un PATCH ens permetrà reservar una adreça MAC lliure per a la nostra unitat. Buscarà a la base de dades el primer rang amb adreces lliures. Agafa l'adreça lliure per retornar-la i calcula la nova adreça lliure per actualitzar la base de dades. Si es tracta de l'última adreça lliure del rang, posarà la variable *hasFreeAddress* com a falsa. En el cas que totes les adreces estiguin esgotades retornarà un error.

Figura 10: Esquema dels endpoints per a gestionar el rang d'adreces MAC.

2.3.2 Unitats LPR5

Per a obtenir informació de les unitats LPR5 tenim dos recursos els quals es pot fer GET.
El primer amb la ruta /lpr5/{serialNumber} ens retorna la informació sobre la unitat que busquem passant-li com a paràmetre el número de sèrie d'aquesta.

Primer valida que li ha passat el número de sèrie, i després el va a buscar a la base de dades. Una vegada trobada la unitat, agafa els identificadors i va a buscar les produccions i els qc a la base de dades i retorna tota la informació en format JSON. Si no existeix cap unitat amb aquest número de sèrie, retorna un error.

L'altre amb la ruta /lpr5/{serialNumber}/verify ens verifica que el número de sèrie és correcte. Els tres primers dígits del número de sèrie de les unitats LPR5 ens diu

quines característiques té la unitat. Això és una manera interna que tenim a Quercus de verificar que el producte és correcte. Aquests tipus els posarà manualment a la base de dades l'equip d'Enginyeria de Quercus.

A aquest recurs li passem com a paràmetre de la query la llicència i la llibreria de reconeixement. Primer valida que li han passat aquests dos camps i en el format correcte i després busca si existeix cap tipus que contingui el codi i aquests dos paràmetres. En cas d'existir enviarà un missatge OK i en cas que no un error perquè voldria dir que el número de sèrie no és correcte.

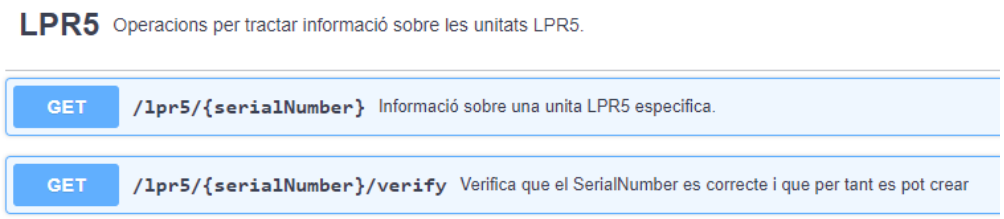

Figura 11: Esquema dels endpoints per a gestionar les unitats LPR5..

2.3.3 Producció

Per a la gestió d'informació de les produccions, hem dissenyat quatre recursos. Si fas un POST a la ruta /lpr5/{serialNumber}/productions, es crea una producció nova amb el número de sèrie que li passis com a paràmetre a la ruta, passant totes les dades que hem especificat en l'apartat anterior al body. Primer valida que s'han passat totes les dades requerides al body i que estan en el format correcte. Un cop validades les dades guarda la producció a la base de dades. Després busca la unitat que tingui el número de sèrie que hem passat. Si ja existeix guarda l'identificador de producció i si no, abans la crea.

I si fas un GET a aquesta mateixa ruta, et llista totes les produccions del sistema. De la mateixa manera que ho fa amb els rangs, li pots passar per query un camp per filtrar o simplement pots filtrar-les totes de cop. També pots posar un límit i un offset com amb les MACs.

I per últim tenim dos recursos per publicar o descarregar un fitxer de log. Fem un POST a la ruta /lpr5/productions/{productionId} passant per paràmetre l'Identificador de la producció i proporcionant el fitxer, per publicar-lo. Primer valida que s'ha proporcionat totes les dades i després comprova que el fitxer pesa menys de 2MB. Per acabar penja el fitxer a l'Storage i guarda el nom del fitxer a la producció.

Si fem un GET, ens descarrega el fitxer de log de la producció en cas que existeixi. Li passem com a paràmetre l'identificador de la producció i el descarrega de l'Storage.

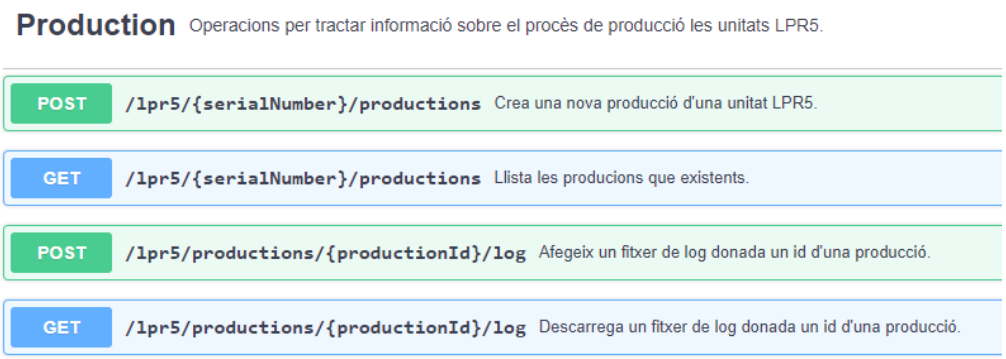

Figura 12: Esquema dels endpoints per a gestionar les produccions.

2.3.4 QC

Els recursos per gestionar la informació sobre el control de qualitat o QC és molt similar als de les produccions.

Fent un POST a /lpr5/{serialNumber}/qcs creem un QC definint el número de sèrie com a paràmetre de la ruta i proporcionant les dades de QC al body. Segueix la mateixa dinàmica que amb la producció.

Fent un GET a la ruta /lpr/qcs ens llista tots els QCs registrats al sistema. Per a publicar un log fem un POST a la ruta /lpr5/qc/{qcId}/log passant el fitxer, mentre que fent un GET a la mateixa ruta ens el podem descarregar.
El mateix per a les imatges que genera el test de QC, si fem POST a la ruta /lpr5/qc/{qcId}/images i proporcionant la imatge la publicarem, amb un GET ens la descarregarem.

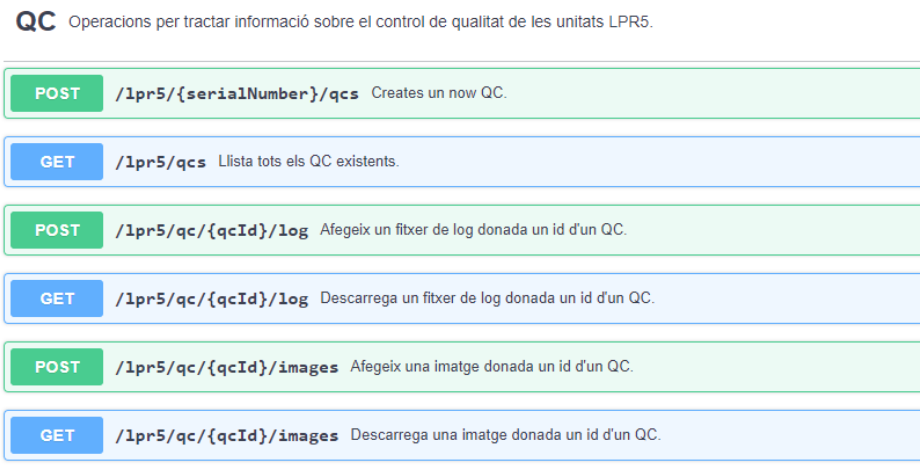

Figura 13: Esquema dels endpoints per a gestionar els QC.

3. Estudi de l'infrastructura

Un cop definit el problema, l'equip de Software decidim que la millor opció per a solucionar el problema és crear una API REST que es connecti al núvol, per tant, hem de determinar quins llenguatges utilitzarem i perquè, així com quina plataforma cloud.

3.1 LLenguatges Implementació

Un cop va quedar decidida la idea de preparar una API REST, vam començar a fer recerca sobre quina tecnologia i llenguatge de programació eren els més adequats. Teníem dues tecnologies web clares, les quals calia comparar per decidir quina de les dues s'adequava millor a la nostra aplicació, Java o Node.js. Així que vam començar a buscar que ens proporcionava cada una d'elles:

3.1.1 Java[1]

Tot i no ser un llenguatge pensat per a desenvolupament web com a propòsit principal, és bastant utilitzat en les empreses de tecnologia per a crear API. Aquest llenguatge té molta antiguitat, això fa que tingui eines molt potents així com IDE (Entorns de desenvolupament integrat) o moltes llibreries. Això fa que treballar amb Java sigui molt còmode. Per altra banda, Java permet executar el codi en qualsevol plataforma independentment del sistema operatiu i del seu origen sempre que aquest tingui la màquina virtual JVM instal·lada, ja que el codi s'executa en aquesta màquina virtual i s'adapta a l'entorn.

3.1.2 Node JS[2]

Pel que fa a Node JS, és un entorn de programació pensat per al desenvolupament d'aplicacions web. Els programes estan escrits en JavaScript i fa servir el motor JavaScript V8 de Google. Aquesta tecnologia es caracteritza per tenir una fàcil escalabilitat, ja que usa un model d'entrada i sortida sense bloqueig i creat per mitjà d'esdeveniments.

A part també es caracteritza per a tenir un "desenvolupament rapider", és a dir, la seva estructura és molt senzilla d'entendre i això fa que tingui un desenvolupament molt ràpid.

3.1.3 Resultat

Després d'analitzar les dues tecnologies l'equip de Software vam decidir utilitzar Node JS, en aquest apartat us explico més detall de per què:

- **Velocitat i Rendiment:** Com hem dit abans Node JS utilitza el motor V8 de Google que es un motor molt eficient. Un dels requisits proposats era que les peticions a l'API havien de ser gairebé immediates. Les aplicacions Java solen tardar més a arrancar, i això pot fer que el Node JS tinguem una resposta més ràpida.
- **Gran quantitat de llibreries i gestor de paquets:** Node JS té una quantitat enorme de llibreries *Open-Source* que ens poden ajudar a fer certes tasques. A part

fent servir el gestor de paquets NPM[5] és molt més senzill gestionar aquestes llibreries o frameworks.

- **Concurrència:** Volem tenir la possibilitat de crear més d'un punt de producció. Això vol dir que en algun moment podrem estar fent peticions a l'API des de dos llocs a la vegada. Node JS treballa amb asincronisme el que ens ajuda a gestionar les connexions concurrents de manera més eficient.
- **Experiència:** Finalment podem dir que els dos llenguatges estan ben preparats per al desenvolupament d'API REST. Tots els punts anteriors també es poden aconseguir si utilitzessim un entorn Java ben preparat utilitzant frameworks com Spring Boot. La raó que va acabar fent que escollíssim Node JS va ser l'experiència que el mànager del projecte ja tenia amb Node JS.

3.1.4 Llibreries utils

En aquest apartat m'agradaria parlar de paquets que m'han sigut molt útils per desenvolupar l'API:

- **Express:** Express és una llibreria de NodeJS que està dissenyada per simplificar el procés de construir aplicacions web i API. Aquesta proporciona una sèrie de característiques i eines per fer algunes tasques de desenvolupament web molt més fàcils, així com tasques d'encaminament, middleware o d'encaminament RESTful.
- **qs:** Aquesta llibreria s'utilitza per treballar amb query Strings, ja que passa el format d'aquests paràmetres a un objecte fàcil de tractar.
- **fs:** Fs és una llibreria que proporciona funcionalitats per treballar amb sistemes de fitxers. Et permet fer diverses operacions relacionades amb fitxers, així com escriure i llegir, crear directoris o eliminar fitxers entre d'altres.
- **macAddr:** Aquesta llibreria et permet passar canviar de format string a MAC i viceversa.

3.2 Cloud Computing

Com hem explicat al principi, aquesta aplicació ens ajudarà a connectar el sistema de producció al núvol, actualment només comptem amb un punt de producció, no contarem les RMAs perquè són peticions puntuals. Per tant, en l'actualitat no tindrem molta càrrega de peticions, però ens agradaria que el nostre sistema fos escalable, per facilitar el procés si en algun moment es volguessin crear més punts de producció. Per altra banda, el nostre equip de desenvolupadors no ha treballat mai amb plataformes al núvol, doncs, també busquem que sigui fàcil de configurar. Avui en dia hi ha moltes plataformes de cloud computing.

Nosaltres decidim analitzar les dues plataformes més grans avui al mercat, Amazon web services[3] i Google Cloud Platform[4].

Per començar a comparar les dues plataformes, el primer que necessitem és entendre que busquem. Tot i que la nostra aplicació haurà d'emmagatzemar moltes dades, estem parlant d'unes 4000 unitats a l'any, la nostra aplicació és petita.

[5] NPM: Node Package Manager

Busquem un Paas. És a dir una plataforma on podem desenvolupar i executar la nostra aplicació i que ens doni d'eines com ara emmagatzematge de dades o arxius. La nostra preocupació és el codi i el funcionament de l'aplicació, la resta de configuració ha de ser controlada per la plataforma. Aquests són els aspectes que vam analitzar:

3.2.1 Oferta de serveis

Pel que fa a l'oferta de serveis, AWS[6] té un ventall de serveis molt més ampli que GCP, gairebé el doble. Tot i que amb els serveis de GCP[7] són suficients per a poder manejar la nostra aplicació.

3.2.2 Preu

Un dels objectius d'aquest projecte era només pagar pel que s'està utilitzant. Les dues plataformes ofereixen plans que compleixin amb aquest objectiu. AWS té una facturació per hora fent servir els seus serveis, mentre que GCP té una facturació per segons usats. Els preus són molt similars

3.2.3 Regions disponibles

Les regions que ens ofereixen les plataformes cloud també ens va semblar un tema important. Les dues avantatges de tenir una plataforma cloud amb servidors a una regió propera on es vulgui instal·lar el punt de producció són:

- Alta disponibilitat: Això ens permet una connexió ininterrompuda.
- Baixa latència: Ens assegura minimitzar el retard des que una petició s'ha fet fins que arriba la resposta.

En el nostre cas, ens centrem principalment en la nostra seu de producció a Reus. Si comparéssim GCP amb AWS, ens adonem que Google té més presència europea que Amazon, tot i que la diferència és mínima.

3.2.4 Suport i documentació

Estem parlant de les dues grans competidores en el món de la computació al núvol. Les dos plataformes tenen els seus serveis ben documentats amb informació de fàcil accés. A part que totes dues són molt populars i, per tant, és molt fàcil trobar-ne informació en fòrums en línia.

Després de l'estudi veiem que les dues plataformes són molt similars, i que, doncs, qualsevol de les dues és una bona opció per a la nostra decisió. El motiu per què s'acaba escollint GCP és perquè tot i que l'equip de Software de Quercus no té experiència amb plataformes al núvol, un membre d'IT sí que té experiència prèvia amb Google i això ens aporta una ajuda en temes de configuració.

[6] AWS: Amazon Web Services
[7] GCP: Google Cloud Platform

3.3 Google Cloud Platform

3.3.1 Entorns de desenvolupament

GCP et permet crear entorns diferents dins d'una mateixa organització. Nosaltres hem creat 3 entorns iguals, el de development, stage i production. D'aquesta manera ens assegurem que si es trenqués l'entorn de desenvolupament, no afectaria producció.

- **Development:** Aquest entorn és el que utilitzem l'equip de software. És un entorn perquè l'equip pugui fer proves.
- **Stage:** Una vegada tenim una versió del producte estable la passarem a l'entorn d'Stage. Aquest entorn té dues funcionalitats diferents. La primera és que permet a l'equip de QA[8], que son els encarregats de testejar el producte, pugui fer proves en un entorn el més real possible. I l'altra funcionalitat, perquè un altre equip del departament, l'equip d'Embedded és l'encarregat de desenvolupar el punt de producció, i aquests companys necessiten un entorn on poder fer proves per incorporar l'API al sistema de producció.
- **Production:** Una vegada QA ha aprovat la nova versió a Stage es passa a production, que com ve el seu nom indica és producció. Una vegada a l'entorn de production automàticament passa a ser la versió que utilitza el punt de producció real de Quercus.

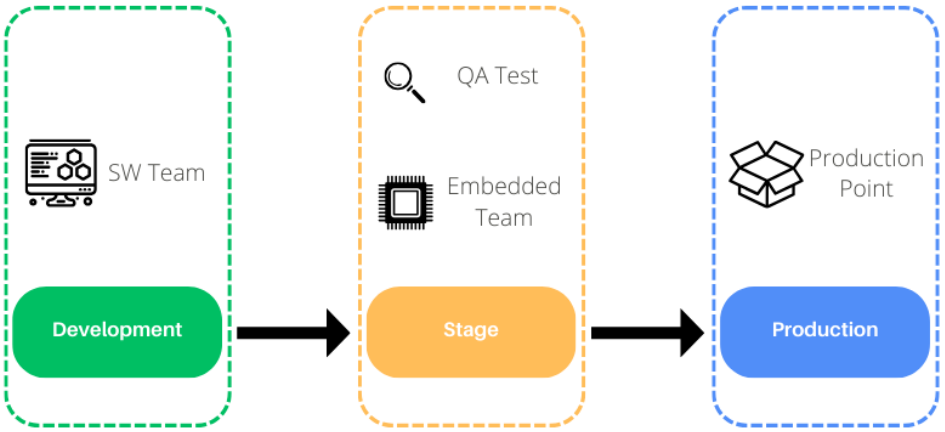

Figura 14: Esquema dels entorns de Google Cloud Platform.

La Configuració d'aquests entorns s'explicarà en l'apartat d'implementació

3.3.2 Serveis utilitzats

- **Firestore**: Firestore és una base de dades no relacional, que utilitza un model document-based per emmagatzemar la informació. Les dades es guarden en col·leccions i aquestes contenen documents. Cada document és en format JSON i fa servir un esquema flexible. Això vol dir que no et força a seguir una estructura

[8] QA: Quality Acceptance

rígida per als documents i aquesta es pot modificar en el cas que es vulgui afegir un camp nou sense afectació a les dades ja guardades. Firestore també et permet filtrar i ordenar els resultats de les cerques. Està molt preparat per treballar amb Node JS, i facilita la interacció amb el teu codi amb funcions bàsiques.

Et permet fer operacions de lectura, que inclouen des d'obtenir documents fins a filtrar-los dins d'una col·lecció. També té operacions d'escriptura, en les quals pots crear documents nous, actualitzar-ne d'existents o eliminar-los. Aquestes últimes operacions no s'escriuen a la base de dades fins que tota la transacció s'ha executat amb èxit.

- **Storage**: És un servei que ens permet emmagatzemar objectes, en el nostre cas, ens serveix per emmagatzemar fitxers. És organitzat per containers anomenats "buckets". Un bucket és similar a una carpeta. Aquests han de tenir un nom únic per a tot GCP. I cada bucket es pot configurar de forma diferent en l'àmbit de control d'accés i retenció de dades.

- **Cloud run**:Servei d'execució d'aplicacions. Un dels principals avantatges d'aquest servei és que executa les aplicacions en contenidors, això vol dir que es crea una imatge amb tots els elements necessaris, així com codi i llibreries per exemple, i aquest és executat. Cloud run té un registre d'imatges, d'aquesta manera si en desplegar una imatge nova ens trobem amb problemes, sempre podrem tornar a l'anterior.

  Un altre avantatge de Cloud run és que és un sistema serverless. És a dir, no ens hem de preocupar de manegar i configurar un servidor, ell ho fa automàticament depenent del tràfic de l'aplicació.

- **Api Gateway**: Servei d'encaminament i seguretat de les peticions. És un servei que et permet crear, desplegar i manegar API. Actua com una porta principal i ajuda a centralitzar les entrades i sortides en un sol punt, d'aquesta manera es pot tenir un control més exhaustiu dels accessos de la teva aplicació.

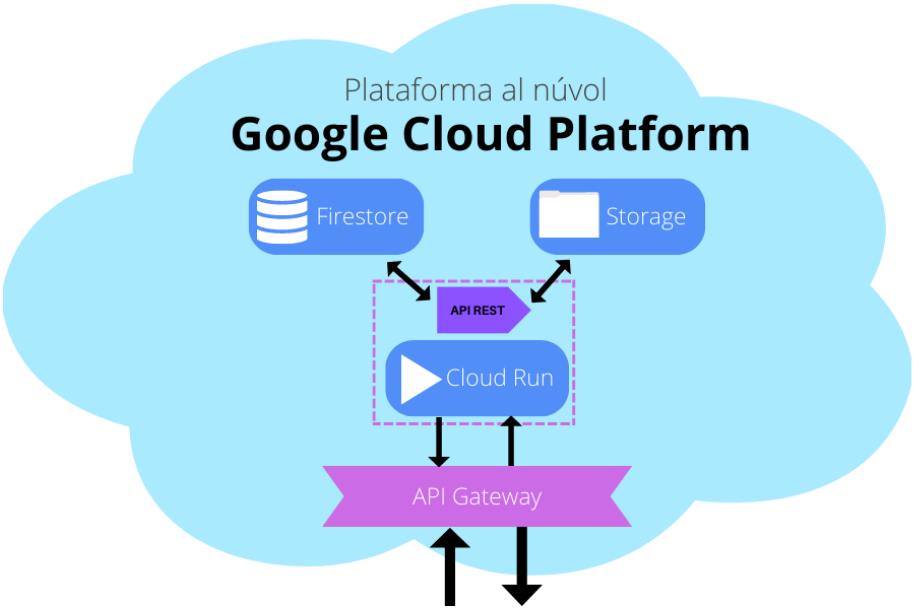

Figura 15: Esquema dels serveis de Google Cloud Platform utilitzats.

4. Implementació

Aquest apartat conté informació confidencial.

5. Avaluació

Una vegada acabada d'implementar l'API es va posar al sistema de producció de Quercus. En aquest apartat us explico quina ha sigut l'avaluació del servei i quins avantatges ha portat en comparació al sistema antic.

5.1 Servei actual

5.1.1 Proves

Per a testejar el sistema actual hem fet una prova intentant flashejar una unitat des de zero.

He creat un rang de MAC nou des de l'adreça 9C:CC:6A:84:82:29 a l'adreça 9C:CC:6A:84:82:30. S'ha creat sense problema.

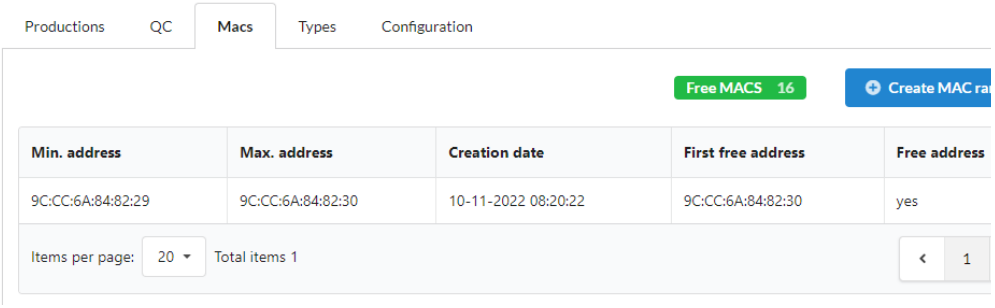

Figura 50: Prova de crear un rang d'adreces MAC

Llavors he començat tot el procés de flashejar. S'ha creat una producció i ha passat el QC. Tant a la producció com al QC s'han publicat els fitxers sense problema.

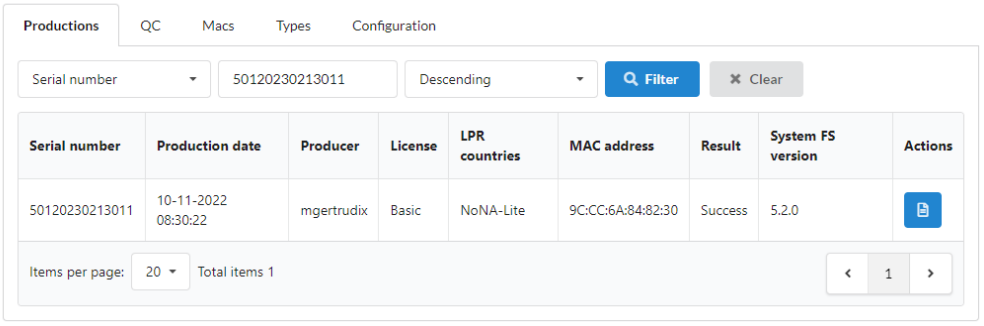

Figura 51: Prova de crear una producció LPR5

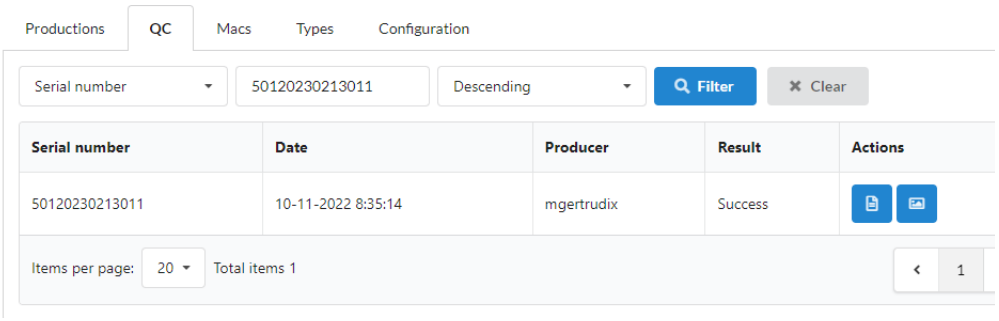

Figura 52: Prova de crear un QC LPR5

Una vegada que la unitat s'ha creat perfectament i ens hem assegurat que ha quedat registrada a GCP, ens assegurem si la gestió de macs funciona correctament, ja que he gastat totes les MAC del rang, per tant, m'hauria de sortir que no té adreces disponibles. Veiem que funciona correctament.

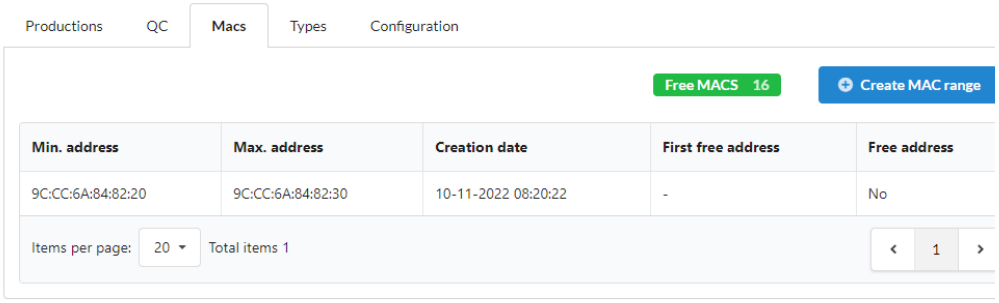

Figura 53: Prova gestió d'adreces MAC lliures

5.1.2 Posada a producció

S'ha desenvolupat una API Rest capaç de connectar el sistema de producció d'un dispositiu embedded al Cloud. L'API permet diferents operacions tant de lectura com d'escriptura de dades, així com gestió d'aquestes.

L'equip embedded ha realitzat una interfície gràfica per al servidor de producció, fent així molt més fàcil el procés de producció. En aquest programa hi ha integrat l'API REST que he desenvolupat.

L'equip de QA ha sigut l'encarregat de testejar l'aplicació. L'aplicació no només ha funcionat amb èxit, sinó que després de fer diverses proves per a comprovar com reacciona davant de fallades. En tot moment retorna les excepcions i els missatges donant la informació necessària.

Actualment, l'API que he desenvolupat està funcionant en el sistema de producció de LPR5 i després d'estar una temporada en actiu he volgut ressaltar les millores que ha aportat envers el sistema antic en el següent punt.

5.2 Servei actual en comparació de l'antic

Podem dir que el gran valor que aporta el sistema nou amb l'API és qualitat humana. El que em refereixo amb qualitat humana és que s'han automatitzat la majoria dels processos que abans eren manuals, com per exemple la gestió de les MACs. Abans requeríem una persona que fes tot el procés d'escriptura dels fitxers indicats per tal de crear adreces MAC, mentre que ara mateix aquest procés es pot fer des de la web.

També cal destacar que amb la implementació dels recursos de lectura i la web, qualsevol treballador de Quercuspot veure quantes unitats LPR 5 s'han produït i veure'n les dades des de qualsevol ordinador quan abans només es podien accedir des de l'ordinador de producció.

Una altra gran millora ha sigut permetre tenir més d'un punt de producció. Actualment, a Reus s'està produint des de dos punts diferents a la vegada. Això permet a producció anar molt més ràpid quan tenen un volum de feina molt elevat.

6. Conclusions

Aquest projecte m'ha ajudat a entendre molt millor com funciona el sistema de producció de Quercus, i també ha aportat el meu granet de sorra per a millorar-lo. També m'ha permès ampliar els meus coneixements en plataformes Cloud que fins ara eren totalment desconegudes. He agafat molt més context sobre les API, ja que ja havia utilitzat API anteriorment però no n'havia dissenyat ni desenvolupat cap.

Al principi del projecte ens vam posar dos objectius que s'han complert. El primer va ser **Centralitzar les dades**. Ara tenim totes les dades en el servidor de GCP, d'aquesta manera per molts punts de producció que creem, la base de les dades és la mateixa. A banda d'això, el sistema està preparat per a la concurrència que múltiples punts de producció puguin desencadenar.

El segon objectiu que vam proposar va ser **Automatitzar processos que abans s'havien de fer manualment**. Aquest objectiu també s'ha assolit. Al principi comentàvem que quan es volia crear un punt de producció nou s'havia de duplicar el sistema de fitxers que teníem per a registrar unitats produïdes, i que s'havia de controlar la repartició de dades perquè no s'estiguessin produint unitats amb dades repetides. Això ho havia de fer una persona responsable, que necessàriament havia de tenir un nivell alt de coneixement d'aquests fitxers. Tot això ha canviat amb la nostra API. Ara aquests processos o bé han desaparegut perquè ja no hi ha necessitat de duplicar el sistema, o bé s'han automatitzat i ja no és necessari que hi hagi una persona responsable per fer-ho.

A banda dels objectius proposats, també s'ha notat una millora en temps de producció, ja que en poder crear més d'un punt de producció, ara es poden produir el doble d'unitats amb el mateix temps.

6.1 Treball a futur i coses a millorar

Encara que l'API ha tingut molt bona rebuda i l'acceptació ha sigut bona, també he detectat certes coses que es poden millorar en un futur:

- **Evitar el cold-start:** Cloud run té configurat per defecte, que quan fa una estona que no s'utilitza, atura el servei. Això pot provocar que quan anem a fer una petició tardi una mica més, ja que ha d'aixecar el servidor i executar el servei. Podríem fer ús del servei Cloud Scheduler de GCP, que és un servei per manegar tasques periòdiques i configurar-lo perquè no aturi el servidor en horari laboral. Això augmentarà una mica el cos de GCP, però crec que l'experiència serà molt més agradable. És un canvi que sols aplica a l'entorn de producció.
- **Optimització del codi:** Com hem pogut anar veient al llarg del projecte les operacions de producció i les de QC són molt similars. Buscaria una manera de fer operacions més genèriques per tal d'optimitzar codi.

7. Annex A

Aquest apartat conté informació confidencial.

8. Referències

[1] Java article de la viquipedia: https://ca.wikipedia.org/wiki/Java_(llenguatge_de_programaci%C3%B3)

[2] Node JS pàgina oficial: https://nodejs.org/es/about

[3] Amazon Web Services pàgina oficial: https://aws.amazon.com/es/?nc2=h_lg

[4] Google Cloud Platform pàgina oficial: https://cloud.google.com/?hl=es