UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Disseny d'un *Cloud Service* per a l'eina *OpenNAC*

Autor: Adrià Sala Fernández

Director:	Jaume Timoneda
Realitzat a:	Departament de Matemàtiques i Informàtica

Barcelona, 14 de setembre de 2020

Abstract

The following document explains the development process followed to design a *cloud* service for *openNAC*, a NAC solution made by *OpenCloudFactory*. The main goal of this project is to establish the foundations for the creation of an internal tool that enhances and optimizes tasks performed by the technical team of the company as well as its main product. The project uses as many modern tools as possible (such as Docker or newer versions of frameworks and such) with a main focus on the "backend" of the designed service. Modularity is prioritized, since the ultimate goal is to build the foundation for a new area of activity inside the company.

Resumen

El documento siguiente muestra el proceso de desarrollo seguido para el diseño de un servicio *cloud* para la herramienta openNAC de la empresa *OpenCloudFactory*. El proyecto tiene como objetivo establecer los fundamentos para la creación de una herramienta interna que facilite las tareas tanto del equipo técnico de la empresa como del principal producto de esta. En el proyecto se usan el máximo de herramientas modernas (como por ejemplo Docker o versiones recientes de frameworks u otros) con un enfoque principal en el "backend" de este nuevo servicio. El trabajo prioriza la modularidad, ya que el objetivo final es construir el fundamento para una nueva área de actividad en la empresa

Resum

El següent document mostra el procés de desenvolupament seguit per al disseny d'un servei *cloud* per a l'eina openNAC de l'empresa *OpenCloudFactory*. El projecte té com a objectiu establir els fonaments per a la creació d'una eina interna que faciliti les tasques tant de l'equip tècnic de la companyia com del seu principal producte. En el projecte es fan servir el màxim d'eines modernes (com ara l'ús de Docker o versions molt recents de frameworks o programari) amb un focus principal en el "backend" del nou servei. El treball prioritza la modularitat, ja que l'objectiu final és construir el fonament per una nova àrea d'activitat a l'empresa.

Agraïments

Aquest document marca el final d'una etapa molt important de la meva vida, i a part d'agrair el suport rebut al llarg d'aquest projecte, vull donar també les gràcies per tot el suport que he rebut durant aquest grau.

En primer lloc als meus pares, que sempre han estat allà quan més els he necessitat i sense ells no hauria sigut possible estudiar un grau sobre un món que m'ha apassionat des de molt petit.

També vull agrair el suport dels meus companys de carrera, en especial a Sergio Montoya i Marc Urgel, que han aconseguit que els pitjors moments de la carrera siguin també memorables.

Voldria agrair també tota l'ajuda dels meus amics de sempre, que m'han donat ànims en tot moment, en especial a José Sánchez i Xavi Charles que sempre m'han ajudat a desconnectar en aquells moments en què era necessari.

També m'agradaria incloure en aquesta secció a la Claudia Rodríguez, que també m'ha recolzat molt al llarg de tot aquest projecte i m'ha fet agafar embranzida en aquells moments en els quals la motivació començava a escassejar.

Per últim, m'agradaria agrair tot el suport a l'equip tècnic de l'empresa, sense el qual no podria haver començat el disseny d'aquest projecte a causa del meu desconeixement sobre el producte *openNAC*. Els hi estaré molt agraït per tot.

Índex

Índex de figures

Índex de taules

1 Introducció

El projecte

A la present memòria es presenta un nou servei al núvol d'ús empresarial per a donar suport a un producte extern de ciberseguretat ja existent (*openNAC*) i tot el procés seguit pel seu disseny. S'explica l'anàlisi previ per definir les funcionalitats que haurà d'implementar el nou servei, la sel·lecció d'eines tecnològiques útils pel seu desenvolupament i el disseny i la seva implementació.

Aquest projecte ha estat realitzat en col·laboració amb *OpenCloudFactory*, empresa fabricant de software referent en el món de la ciberseguretat pel seu principal producte, anomenat *openNAC*, que és un software NAC que ofreix gran modularitat i permet una visibilitat completa sobre tots els components que formen la xarxa.

L'aplicació dissenyada treballarà braç a braç amb *openNAC* i té el potencial de convertir-se en una eina molt important per al funcionament d'aquesta. L'objectiu d'aquest projecte és establir els fonaments per a la creació d'aquesta nova eina. És per això també que un dels focus principals de la nova eina és la modularitat, permetent que es puguin afegir nous mòduls amb gran facilitat.

El projecte s'ha basat en el concepte de "cloud service", pel que en un futur podria ser instal·lada en un servidor centralitzat. Està dividida en dos apartats que es podran ampliar fàcilment: el portal d'administració, que ofereix eines útils per a l'optimització de tasques comercials i tècniques de l'empresa, com ara oferir un registre d'instàncies i el seu estat actual, un altre de les llicències atorgades...; i l'API, que serà l'apartat que està obert a la xarxa externa i es connecta amb totes les instàncies del producte, per recollir el seu estat actual o retornar informació útil per al seu funcionament (llistes de MAC Vendors, blacklist d'IP...).

Per tal de fer el projecte el més interessant possible, s'han fet servir tecnologies no observades al llarg del grau o observades breument, com ara l'ús d'un framework en PHP (*Symfony*) i el desplegament del projecte i entorn de desenvolupament en *Docker*.

En el projecte s'han fet servir conceptes apresos al llarg de la carrera a les assignatures de *Software Distribuït*, *Disseny de Software*, *Bases de Dades Avançades*, *Sistemes Operatius*, *Enginyeria de Software*, *Fonaments de la Ciberseguretat*, *Xarxes* i altres coneixements adquirits a *Practiques a Empresa I*, gràcies a haver fet el conveni de pràctiques a la mateixa.

L'empresa

OpenCloudFactory és un "fabricant" de ciberseguretat amb clients arreu del món. Va ser fundada el 2012 i es va basar principalment en l'estudi de les tecnologies

NAC.

Amb presència a Espanya, Mèxic i Brasil, el 2016 l'empresa va començar a desplegar-se comercialment com a "vendor" de ciberseguretat, i el 2017 va ser reconeguda per la consultora tecnològica *Gartner* com únic fabricant europeu inclòs en el seu **Market Guide** de tecnologia Network Access Control (NAC). Durant els dos anys següents, l'empresa va tornar a ser considerada com a empresa de tecnologia avantguardista i alineada amb les tendències dels experts tecnològics, i segueix sent l'únic fabricant europeu dins de l'informe.

El 2018, l'empresa fòu reconeguda pel *MINECO* (Ministerio de Economía y Competitividad), obtenint el segell d'empresa innovadora; títol atorgat a aquelles empreses amb un ADN innovador. Avui en dia, l'empresa forma part del programa d'acceleració europea promogut per *EIT Digital* i està inclosa dins el catàleg de productes STIC, del **CCN-CERT** (Centro Criptológico Nacional). La solució que ofereix OCF dins del catàleg de solucions de ciberseguretat del CCN-CERT és l'eina EMMA, de visibilitat i control sobre la xarxa.

L'empresa ha treballat amb gran quantitat de grans i petites empreses, institucions i entitats públiques, d'entre les quals destaquen la UB, CaixaBank, Santander, FGC, Telefònica...

Aquest any, l'empresa ha començat un procés de reestructuració molt important i ha començat a treballar amb empreses externes o "partners", amb l'objectiu de poder augmentar els recursos disponibles per permetre's ampliar el nombre de projectes en funcionament sense haver de reduir la qualitat de l'atenció a aquests.

A part d'altres productes i projectes en fase de desenvolupament, l'empresa té 3 productes principals:

- **openNAC**: eina sobre la qual es basa el projecte i l'única amb la qual he treballat fins ara i que explicaré en major detall a la següent secció. Bàsicament és una solució NAC modular que ofereix una gran varietat de mòduls, com ara un de visibilitat de la xarxa corporativa (anomenat EMMA), un de control d'accés basat en polítiques, un altre basat en una sol·lució d'accés remot...
- **Viapps**: eina que permet crear, instal·lar, configurar, actualitzar, gestionar, monitorar i auditar serveis instal·lats a xarxes corporatives com ara firewalls, servidors DNS/DHCP/NTP, proxies HTTP, SMTP gateways i loadbalance de IPs.
- **PCI Mail-Vault**: solució que permet que tots els correus rebuts de l'exterior amb dades de targetes de crèdit/dèbit siguin emmagatzemats a una base de dades segura, assegurant que l'entorn PCI-DSS del client no es vegi afectat per possibles correus amb informació sensible. Aquests correus tan sols poden ser gestionats per personal autoritzat que tingui la necessitat de fer servir aquestes dades.

El producte: openNAC

L'eina openNAC és una plataforma d'accés a la xarxa per a entorns LAN/WAN, que permet a les empreses que instal·lin el servei autenticar, autoritzar i auditar tots els accessos basats en un conjunt de regles o polítiques. L'objectiu d'aquesta eina és augmentar la seguretat de la xarxa, oferint a les organitzacions la visibilitat del 100% dels elements de la xarxa i el control d'aquests actius (dispositius i altres) connectats mitjançant tant Wi-Fi com cable o accessos VPN.

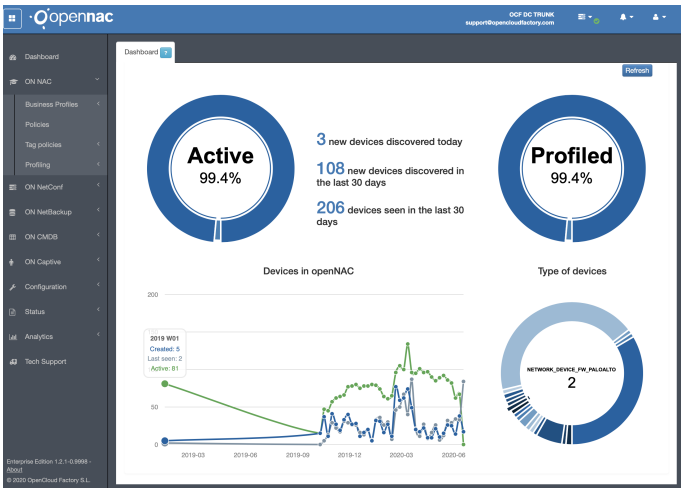

Figura 1: Portal de l'eina openNAC

L'eina està dividida en mòduls, perquè cada client pugui contractar els que consideri més adequats pel seu cas d'ús. Els principals serien els següents:

- **Autenticació**: permet comprovar la identitat de les entitats que es connecten a la xarxa corporativa. La identitat es pot validar mitjançant diversos repositoris (com ara un Active Directory, certificats digitals, adreça MAC...).
- **Autorització**: ofereix la capacitat d'assignar privilegis a cadascuna de les entitats que accedeixen a la xarxa corporativa, com ara assignar-los una VLAN determinada.
- **Auditoria**: permet recollir, agrupar i avaluar els accessos o intents d'accés a la xarxa corporativa i tenir un historial de tota l'activitat a la xarxa.
- **Inventari**: l'eina té una base de dades de configuració (anomenada CMDB) que emmagatzema els detalls de cadascuna de les identitats que es connecta a la xarxa.
- **Perfilat**: permet establir i comprovar un perfil per a un estat concret d'una identitat, que pot ser configurada com a obligatòria per a accedir a la xarxa; un exemple en seria una versió concreta d'un sistema operatiu, la presència d'un antivirus...

- **Posturació**: permet avaluar en temps real el comportament de cadascun dels dispositius de la xarxa i determinar si el comportament d'aquests està dins d'uns paràmetres de normalitat esperats, prenent mesures correctives de no ser així.

- **Remediació**: relacionat amb el punt anterior, ofereix la possibilitat d'executar les accions necessàries per a resoldre o minimitzar l'amenaça detectada. Exemples de remediació inclourien l'aïllament de certes entitats compromeses de la xarxa corporativa.

- **Autenticació 2FA**: funcionalitat dissenyada per a complementar l'accés de VPNs a la xarxa amb un segon factor d'autenticació. Les tecnologies compatibles són Mobile Connect i Google Authenticator.

- **Agent Natiu**: instal·lat al dispositiu que s'hagi de connectar a la xarxa, permet extraure informació útil (software instal·lat, estat de l'antivirus...) per a poder respondre amb major rapidesa a amenaces i anomalies detectades.

- **Integració amb SIEM**: permet integrar l'eina amb un sistema de gestió d'informació i esdeveniments de seguretat de tercers per a poder concentrar els l·logs"enregistrats amb la resta de sistemes d'informació de l'empresa.

- **Orquestració**: permet l'ús de l'eina com a element de gestió de la seguretat, gràcies a la seva capacitat de comunicar-se amb altres elements de la xarxa (firewalls, antivirus, IDS...); la comunicació amb aquests és bidireccional, així que openNAC pot enviar informació enregistrada a la seva base de dades (CMDB) per a permetre a aquests altres elements prendre decisions amb major seguretat.

- **Sensor**: mòdul que permet complementar la visualització del comportament de les entitats de la xarxa mitjançant la inspecció i classificació del tràfic de xarxa de la capa d'aplicació (capa 7 del model OSI). Permet també la visualització de segments de la xarxa no controlats pel NAC.

- **Network Device Compliance**: permet validar paràmetres de configuració específics per a qualsevol dispositiu de xarxa, com ara encaminadors, switches... Es pot fer servir per auditoria, validació de la seguretat o configuració general dels dispositius. Es poden extraure còpies de seguretat de les configuracions i crear configuracions predeterminades que realitzar de manera automatitzada a conjunts de switch de diverses marques i diversos entorns.

L'eina està dividida en tres components principals, i cadascuna d'aquestes té una funció diferent:

- **Core**: component crucial i principal, té el rol prioritari i actua com a "cervell" d'openNAC; proporciona els serveis AAA (*Authentication, Authorization & Accounting*) mitjançant **FreeRadius**, entre altres. També s'encarrega de mantenir la base de dades (CMDB) i el funcionament del portal administratiu.

- **Sensor**: component amb un rol essencialment tecnològic; recull i descodifica els protocols utilitzats a la xarxa corporativa i els envia a l'Analytics. El funcionament d'aquest procés de descodificació es basa principalment en **Bro IDS/IPS**.
- **Analytics**: component basada en el stack ELK (**ElasticSearch**, **Logstash** & **Kibana**), que recull totes les comunicacions interceptades pel Sensor i les emmagatzema assignant-los índex per tal de facilitar el procés de cerca i filtrat, per a poder generar *dashboards* per a la visualització de dades.

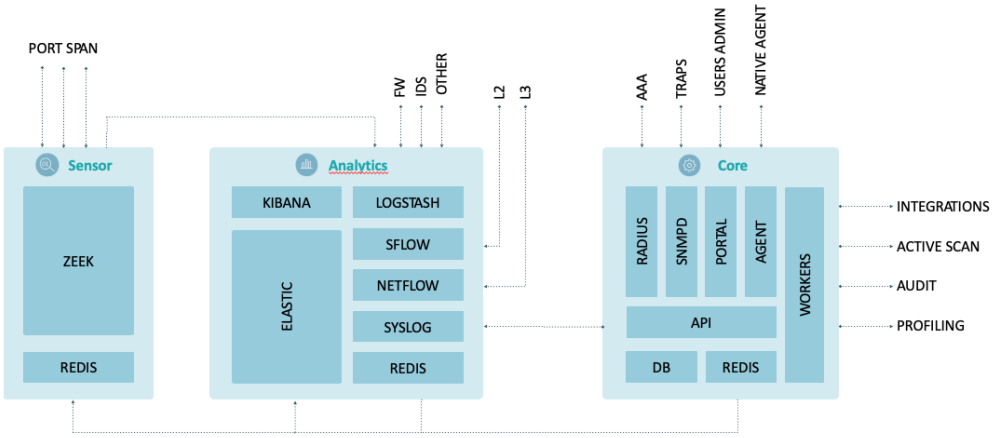

Figura 2: Estructura dels components d'openNAC

A continuació, podem observar un exemple d'una xarxa corporativa amb openNAC instal·lat.

Podem observar que el Sensor estarà, junt amb el firewall, escoltant tot el tràfic que va sortint i entrant de la xarxa al punt més crític de la infraestructura, mentre que el Core i Analytics es poden executar a un centre de dades extern.

Aquests centres externs solen ser habitualment entorns *AWS* o *Azure*; l'eina ja està dissenyada amb això en ment i la instal·lació en aquests tipus d'entorns es pot realitzar amb gran facilitat.

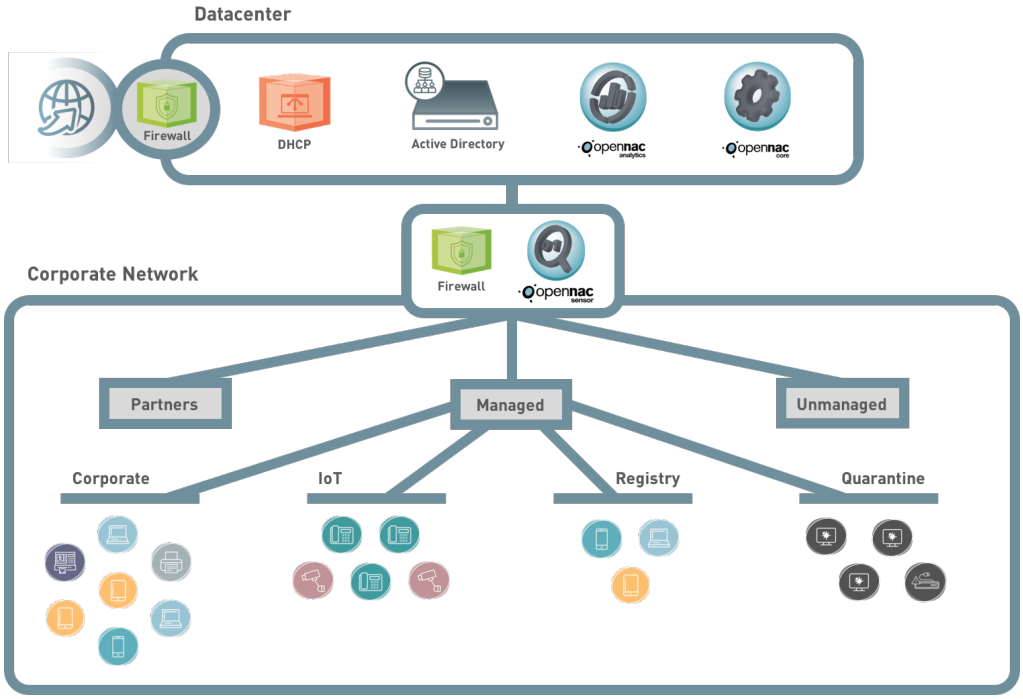

Figura 3: Exemple d'arquitectura amb una instalació d'openNAC

El més destacable del funcionament d'openNAC respecte als competidors és l'ús de **tags** i **polítiques**.

Els **tags** són etiquetes que serveixen per a perfilar la identitat d'un element de la xarxa; el seu sistema operatiu, l'estat del seu antivirus, les aplicacions instal·lades... La majoria es generen automàticament mitjançant processos interns predefinits (pel desenvolupador o pel client en les seccions definides), però també se'n poden afegir manualment una vegada ja s'ha enregistrat el dispositiu a la CMDB.

Les **polítiques** es fan servir a l'eina openNAC per a determinar si un dispositiu podrà accedir a la xarxa i a quina part d'aquesta podrà accedir-hi; es poden utilitzar com a filtres de les polítiques l'hora a la qual es realitza la primera connexió, l'usuari amb el qual s'intenta accedir, els *tags* del dispositiu, el tipus de connexió que vol realitzar (8021X, MAB, VPN...). En cas de passar aquests filtres, es pot escollir a quina VLAN se li dona accés. El motor encarregat de gestionar tot el funcionament de l'assignació de polítiques el trobarem al Core.

1.1 Motivació

El projecte sorgeix de l'interès en concepte de servei al núvol i la idea darrera d'aquest. Sempre m'ha agradat la idea de sistema que "pensa" mentre un usuari es connecta des d'un simple navegador. A part d'això, una altra part del projecte que també em va interessar molt era el poder fer servir eines que fins ara no m'havia proposat fer servir, tant pel meu propi compte com per part de la universitat. L'eina

Docker, per exemple, és un tema del que he sentit molt a parlar pero mai he tingut l'oportunitat de posar-me a fer proves i posar-la en pràctica. També a l'empresa s'havia comentat les oportunitats de cara al món de la *Continuous Integration*[1] i *Quality Assurance*[2] que ofereix l'eina.

L'interés per part de l'empresa en controlar amb més precisió certs aspectes d'*openNAC* també va ser un dels principals impulsors del projecte. Podríem destacar-ne el desig de voler mantenir un control més correcte i adeqüat de quines instàncies del producte hi ha en funcionament avui en dia, de quines estan llicenciades actualment, quan caduquen les seves llicències, etc.

Avui en dia el problema que hi ha amb això és que la majoria d'aquests temes depenen de fulles de càlcul d'Excel i, en el pitjor dels casos, de correus i missatges de Skype. L'empresa està creixent molt i aviat arribarà a un punt on comença a ser necessari el desenvolupament d'una eina que ajudi a la realització d'aquestes tasques i a mantenir un tracking sobre aquestes.

Un altre motiu pel qual ha nascut el projecte és per l'interès per part de l'empresa en centralitzar el funcionament de certs mòduls d'*openNAC*; comença a tenir projectes amb empreses amb menys recursos o simplement que tenen el desitg d'instal·lar l'eina a infraestructures amb menor capacitat de processament.

És per això que una de les idees principals del projecte és la modularitat, ja que la idea és que a mesura que es vagin definint quins mòduls de l'eina poden ser centralitzats es puguin anar afegint al servei sense gaire complexitat, i la capacitat de reduïr esforços a les instàncies per a permetre'ns realitzar instal·lacions amb menor necessitat de recursos.

2 Objectius

D'acord amb la introducció, podem extreure els següents objectius relacionats amb el projecte:

1. Dissenyar un possible esquema del servei al núvol i definir les eines que seran necessàries.

2. Adquirir coneixements de totes les eines que seràn necessaries per a la realtització del projecte, com ara *Docker*, *Symfony*, *NGINX*, el llenguatge de programació *PHP*, connexions *LDAP* i altres conceptes d'*openNAC* amb els quals no estic familiaritzat (ús de llicències, *healthcheck...*).

[1] Pràctica d'enginyeria de software que intenta fer el màxim d'integracions d'un projecte el més sovint possible per tal de detectar els errors amb la major rapidesa possible

[2] Conjunt d'activitats que s'apliquen a un producte per tal de verificar que els requisits de qualitat d'aquest siguin satisfets

3. Creació d'un entorn de desenvolupament en *Docker* amb gran diversitat de components.

4. Intentar implementar el màxim d'eines al servei al núvol, entre les següents:

    - Sistema de gestió de claus API per definir l'accés al servei, amb la capacitat de poder renovar o revocar-les.
    - Repositori de recursos comuns a les instàncies del software, obtingudes de fonts oficials (anàlisi de "CVE", MAC vendors, blacklist d'IP...).
    - Sistema de gestió de llicències, amb la possibilitat de renovar o revocar-les o d'implementar llicències amb rols diferents.
    - Emmagatzematge de l'estat de les diverses instàncies del programa amb detall per cadascun dels serveis que tingui i la capacitat de poder saber d'una manera immediata què falla.

5. Creació d'un projecte amb un focus principal en la modularitat.

6. Creació d'un entorn de desenvolupament el més semblant a l'entorn de producció possible

3 Planificació

Per a l'explicació de com s'ha planificat el projecte s'ha fet un diagrama Gantt amb les diverses tasques realitzades al llarg del desenvolupament d'aquest.

La planificació inicial del projecte s'ha vist afectada notablement a causa de la crisi sanitària del *SARS-CoV2*: això ha implicat que la totalitat de l'aprenentatge, disseny i implementació s'han realitzat des del teletreball.

Una gran part del projecte, llavors, s'ha hagut de dur a terme en solitari, a causa del gran volum de feina que ha hagut de fer front l'empresa per tal de mantenir una bona situació econòmica.

Aquest fet ha afegit una complicació nova: el procés d'adaptació a l'ús de VPN i eines de videoconferència a l'empresa ha estat força complex i ha implicat una reestructuració en la manera de fer les coses.

Aquest fet va implicar ajornar la realització del TFG pels voltants d'un mes, tal com es podrà observar al diagrama. Afortunadament, gràcies al període d'extensió ofert per la UB s'ha pogut completar el projecte.

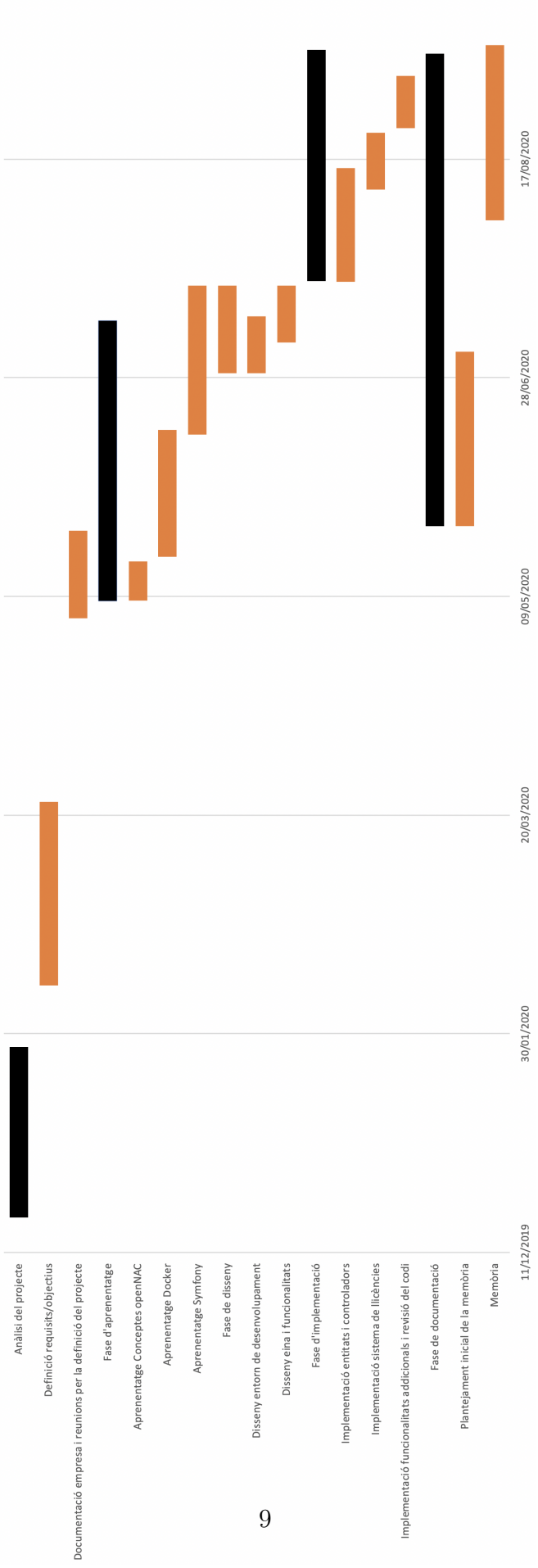

Figura 4: Diagrama Gantt del projecte

4 Anàlisi

la idea del projecte comença partint dels següents requeriments acordats amb l'empresa:

- L'entorn de programació ha de ser PHP; es el llenguatge de programació utilitzat a tots els projectes i productes de l'empresa i, per tant, el projecte ha de seguir els mateixos patrons.
- L'ús d'un framework serà essencial; per tal d'estalviar-se tasques de més baix nivell i de facilitar notablement la feina necessitada per a la implementació.
- El projecte s'haurà de poder desplegar automàticament amb *Docker* ja que l'objectiu, extern al projecte que s'està desenvolupant, és poder desplegar l'aplicació mitjançant l'ús d'eines d'integració contínua com ara *Jenkins*.
- L'aplicació hauria de ser el més modular possible per tal de poder anar afegint noves eines a la plataforma dissenyada sense gaires maldecaps.
- El projecte haurà d'autenticar els usuaris del portal administratiu mitjançant l'*Active Directory* de l'empresa, pel que per fer el registre de nous usuaris s'haurà de fer les modificacions adients des d'allà i no a l'eina a dissenyar.

Pel que fa als requeriments del propi projecte, cal que la plataforma a dissenyar pugui:

- Crear un sistema per a generar claus API que permetin l'accés a l'eina des de l'exterior amb un gran focus en la seguretat de l'eina.
- Ser capaç d'emmagatzemar una gran quantitat de dades, servint-les de la manera més ràpida possible, i poder processar peticions a l'API de diverses instàncies alhora.
- Fer servir el màxim d'eines el més modernes possible.

El sistema d'usuaris no serà gaire important perquè dependrà totalment de l'AD de l'empresa. El sistema de claus API, però, haurà de ser el més robust possible i caldrà que el mètode d'autenticació sigui el més segur possible. L'API, però, tindrà un ús molt limitat i no podrà accedir a recursos de l'aplicació ni a cap dada que pugui comprometre el sistema.

Un altre fet a destacar és que el projecte en un primer lloc es va considerar fer-lo amb una API REST per definir tota la comunicació amb la base de dades; a causa que l'eina tan sols es vol fer servir mitjançant el portal administratiu que es crearà, es va descartar l'opció de dissenyar l'aplicació mitjançant un "API-First approach". El projecte es limitarà a dissenyar les dues seccions del servei al núvol,

que seran l'apartat d'administració controlada mitjançant el portal administratiu i l'apartat de comunicació amb les instàncies definides.

Abans de començar a decidir quines eines fer servir per al disseny, i seguint els punts definits als Objectius del projecte tant com l'ordre de necessitats principals de cara al producte i l'entorn de l'empresa, s'ha imposat un ordre i una prioritat pels elements a dissenyar de l'eina al núvol:

1. **Sistema de gestió de claus API**: aquest és l'element més prioritari perquè d'ell depèn tota interacció entre el servei al núvol i les instàncies del producte amb les quals s'haurà d'interactuar.

2. **Sistema de gestió de llicències**: aquest passarà a ser el segon element amb major prioritat a causa del requeriment per part de l'empresa de començar a definir un sistema de seguiment de les llicències aplicades i el seu estat, per tenir coneixement de quan s'han de prendre mesures per renovar-les, mesures com posar-se en contacte amb els clients o altres possibles accions.

3. **Seguiment de l'estat de cadascuna de les instàncies**: aquest passa a ser la tercera secció de l'eina amb major prioritat, però podria ser perfectament la segona. Avui en dia, l'empresa depèn dels correus per a poder anar fent un seguiment de l'estat de les instàncies i clústers definits; l'objectiu d'aquesta secció seria recollir totes les dades de les instàncies i mostrar amb facilitat quines tenen problemes i què hi falla, tot això fent servir el *healthcheck*, funcionalitat d'*openNAC* que s'introdueix a posteriori.

4. **Funcionalitats extra de les eines esmentades prèviament**: d'entre elles s'hauria de destacar la possibilitat de renovar les claus API d'una manera automàtica, la possibilitat de renovar i revocar llicències o d'implementar-les amb rols diferents.

5. **Repositori de recursos comuns**: aquest punt junt amb l'anterior, seria fàcil d'implementar des de la banda del servei al núvol, però requeriria d'una gran modificació del producte i un notable esforç per part de l'equip tècnic de l'empresa, pel que per tal de reduïr costos s'ha decidit optar per prioritzar les tasques aïllades del producte de l'empresa i deixar aquelles que requereixin modificacions del producte pel final del projecte o com a futures tasques posteriors a aquest.

4.1 Mesures de confidencialitat a l'empresa i de cara al client

A diferència d'un projecte fet per un alumne de manera independent, el fet de fer el projecte de TFG amb una empresa té les seves implicacions. Afortunadament, però, l'empresa ha sigut molt flexible amb tot allò que puc mostrar del producte i l'únic aspecte de l'eina que no podré mostrar amb gran detall és, evidentment, com es generen les llicències del producte. Per la resta, com el projecte s'enfoca

principalment en el servei al núvol i no en realitzar canvis al programari principal de l'empresa, no caldrà mostrar el funcionament intern del software amb gran detall. El codi font contindrà, llavors, la implementació del servei al núvol al complet excepte l'arxiu necessari per generar llicències.

Pel que fa a l'aspecte de treballar amb dades de clients, com ara on estan instal·lades les instàncies o altres dades que podrien ser interessants per mostrar exemples de funcionament, amb les màquines de l'empresa i crear conjunts de dades ficticis n'hi haurà prou per mostrar totes les funcionalitats, pel que no serà necessari ometre cap dada.

De cara al disseny de l'eina, però, sí que s'hauran de tenir en compte els següents aspectes:

- Molts clients no volen que es pugui accedir a les seves instàncies des de l'exterior, per temes de seguretat, pel que s'haurà de tenir en compte que es vol evitar haver de realitzar connexions des del servei al núvol cap a la instància en si; per tant, tan sols treballarem amb respostes a peticions realitzades des de les instàncies.
- Caldrà tenir en compte també que hi ha clients que tenen instal·lacions que són completament offline; s'haurà d'oferir un mètode per a poder fer traçabilitat d'aquestes instàncies tot i no poder fer servir aspectes com recollir l'estat dels components d'aquestes, o altres funcionalitats.
- També és interessant recollir dades més internes sobre les instàncies i clústers, però sempre s'haurà de respectar la privacitat del client i la recol·lecta de dades del caràcter més general possible.

4.2 Conceptes d'*openNAC*

Sobre *openNAC* caldrà analitzar dos conceptes que encara no s'han introduït. Un d'ells és el *healthcheck*, el mòdul de monitorització del producte, que realitza comprovacions constants per assegurar que tots els serveis que requereix estiguin funcionant adeqüadament. També serveix per a sistemes clusteritzats, realitzant comprovacions a cadascun dels servidors dels *openNAC*s.

L'estat del servidor, llavors, es pot visualitzar accedint al portal administratiu de l'eina i visitant el panell superior:

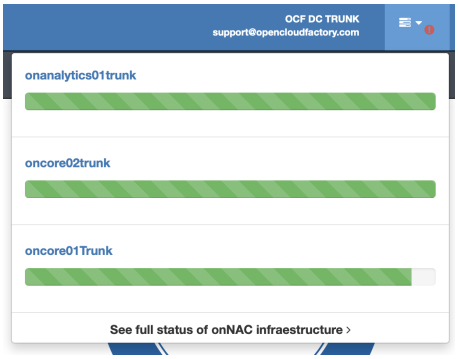

Figura 5: Menú desplegable del mòdul *healthcheck*

Si cliquem al botó inferior del desplegable, on hi posa "See full status of onNAC infrastructure", podrem veure una vista ampliada de la informació que ofereix el menú desplegable. Tota aquesta informació es pot recollir en un JSON que es retorna fent una crida mitjançant l'API d'*openNAC*, amb un `GET` a `/admin/rest/healthcheck`.

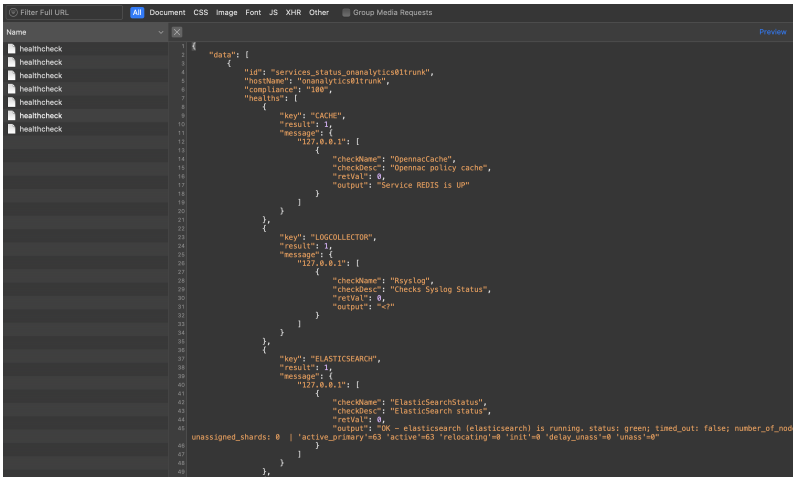

Figura 6: Visualització del JSON retornat per `/admin/rest/healthcheck`

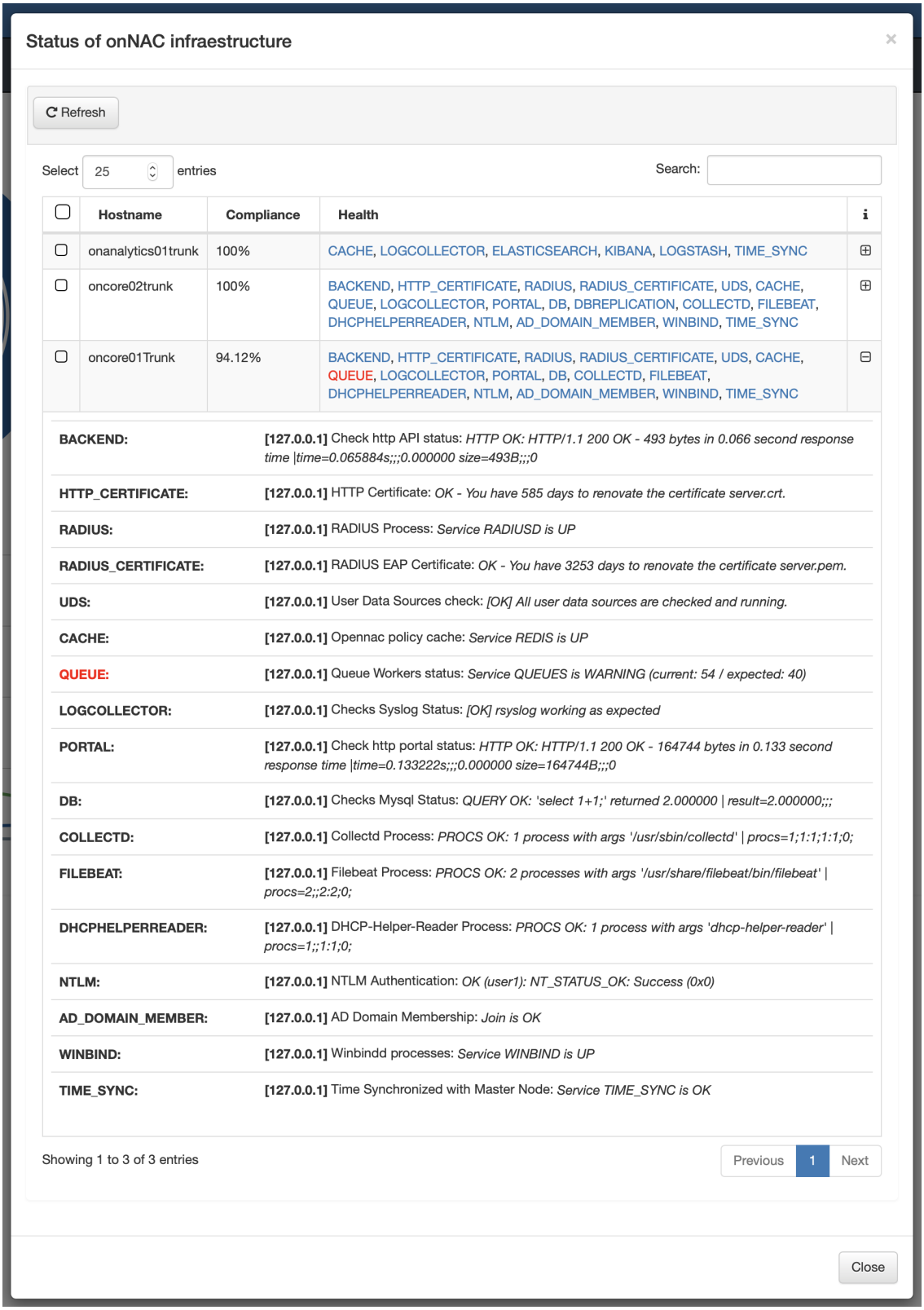

Figura 7: Menú desplegable de *healthcheck* del portal administratiu d'*openNAC*

Les dades recollides al fitxer JSON, més concretament les que es troben sota la clau `data`, seran les que s'enviaran al servei al núvol amb certa freqüència; encara s'està debatent, però es calcula que s'enviïn les dades cada cop que es calcula l'estat dels serveis per tal d'aprofitar el `cron`[3] ja definit per fer aquesta tasca.

Pel que fa al sistema de llicències, dins de l'apartat de configuració del portal administratiu de l'eina podem trobar un apartat dedicat a l'activació de la llicència. En aquest apartat podem generar el fitxer de petició de llicència que s'ha d'enviar a l'empresa per a la seva activació, que requerirà el nom de l'empresa client i el correu de contacte d'aquesta:

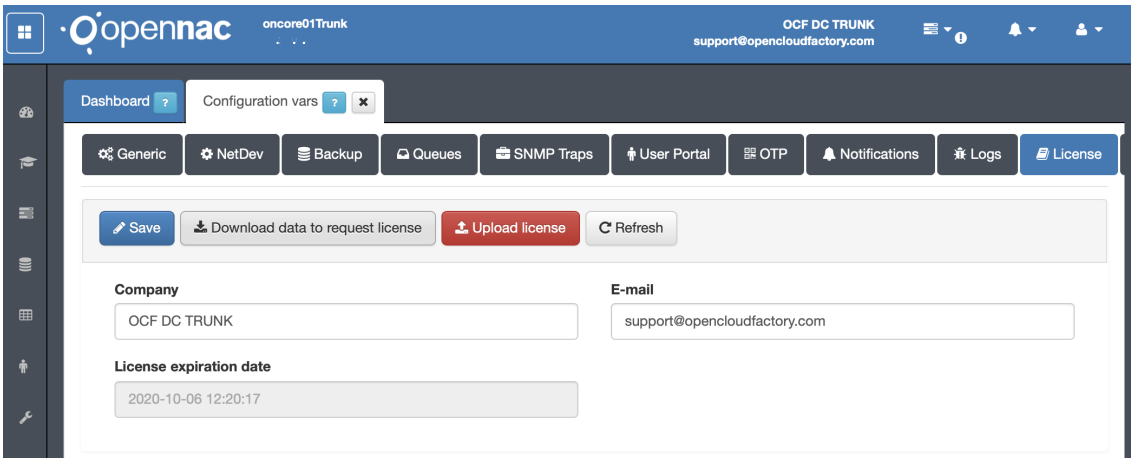

Figura 8: Secció de llicència del portal administratiu

Si cliquem el botó "Download data to request license", se'ns descarrega un fitxer JSON amb un contingut similar al següent:

```
{
    "company":"OCF DC TRUNK",
    "email":"support@opencloudfactory.com",
    "serialNumber":"0x00012900",
    "interfaces":{
        "lo":{
            "ip":"127.0.0.1\/8",
            "mac":"1\/128"
        },
        "eth0":{
            "ip":"X.X.X.X\/X",
            "mac":"fe80a04e95fffe4b3612\/64"
        }
    }
}
```

Figura 9: JSON amb contingut de petició de llicència

[3]Servei d'Unix per a la planificació de tasques que s'utilitza per a definir l'execució de tasques repetitives en els intervals temporals que un desitgi

Aquest fitxer, per activar-lo, caldrà passar-lo pel script d'activació de llicències, al qual li haurem de passar el fitxer amb la informació generada pel portal, el nombre de dies que durarà la llicència del producte, els certificats privats i públics de l'empresa per a signar la llicència i, finalment, una adreça on desar el fitxer de llicència generat:

```
adrisala@Sala-MBP license_php_testing % php create_license.php
Usage: create_license.php -i {inputPath}
OPTIONS:
    -t    License TTL in days (default 365).
    -pub  Public key path
    -priv Private key path
    -out License output path
```

```
adrisala@Sala-MBP license_php_testing % php create_license.php -i test_data.json
 -t 1 -pub public.pem -priv private.pem -out license.zip
File to encrypt:
{
    "company": "OCF DC TRUNK",
    "email": "support@opencloudfactory.com",
    "serialNumber": "0x00012900",
    "interfaces": {
        "lo": {
            "ip": "127.0.0.1\/8",
            "mac": "1\/128"
        },
        "eth0": {
            "ip": "1.1.1.1\/24",
            "mac": "fe80a04e95fffe4b3612\/64"
        }
    },
    "ttl": "2020-09-13 10:01:29"
}
Encrypted text:
tJIU7MCnejY1g6nk4fScC3TmCu1qXd1z5ZHOEwfE9bNzEni4c4UECU49+K6ACKeMeoumhhGBjdqnIhMF
45q+YfI7OpacGoxZ0QPoBO3dMnBdFhUuNhP1+kRJhFaVJ6JOU+4kdlpxap4iEKDW3Dx0kzrj7Y55FBuJ
WTXacjDNboMcib+KLFjvRefvGu3kKJowOxKge/WvTV555S5Hk/8Japj42T4m4aCtpx6dU7evhnwTUJjY
Ulx7KvEPVGov+RCLcYsHq/+Z35VdL+x12qLD7HVCa/LOZgD/Fs4lumQAxxGm1L1/mC53DFBNKHafCE2U
8GWCotEGYtci01PyuE0oJQ==
zip -j license.zip /tmp/opennac.license
MD5 public key: 17741b5e189a50c0d1e4d62ab0de6658

License created on license.zip
```

Figura 10: Opció d'ajuda del script de creació de llicències i output resultant de l'execució d'aquest

Finalment, se'ns genera un fitxer `.zip` que serà el que s'enviarà al client i que ell haurà de pujar a la secció del portal esmentada anteriorment amb el botó "Upload License". El contingut del fitxer segueix una estructura similar a la d'un certificat, tal com es pot observar a continuació:

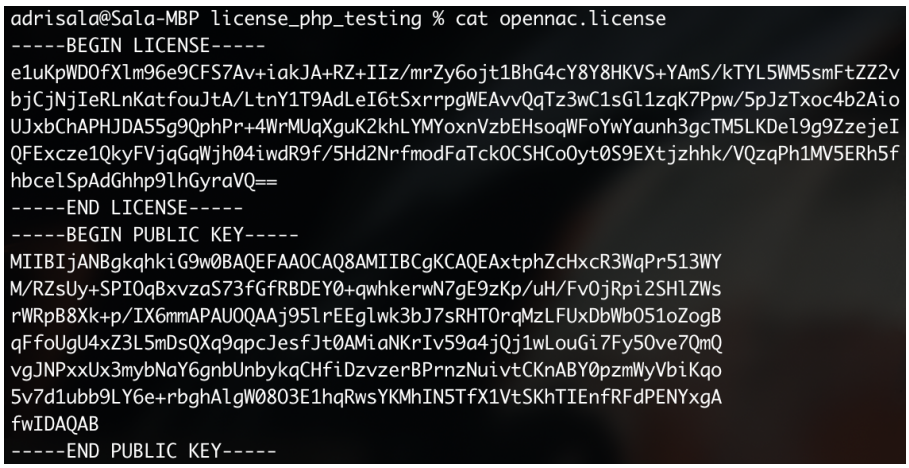

Figura 11: Opció d'ajuda del script de creació de llicències i output resultant de l'execució d'aquest

Una de les primeres idees que va sorgir al llarg de l'anàlisi del projecte va ser la idea d'automatitzar completament aquest procés pel client; ja que el fitxer que conté la informació de la petició de llicència és un JSON, es pot enviar fàcilment al servei al núvol amb clicar un botó i des del servei mateix processar aquesta petició quan es desitgi, o fins i tot gestionar la totalitat del procés des del portal administratiu del servei, extraient des d'aquest la petició de llicència del client i retornant-li la llicència activada sense necessitat de que el client hagi de fer res.

Malauradament, seguint amb els punts descrits al subapartat anterior, la major part dels clients no desitgen que es tingui contacte amb els seus sistemes des de l'exterior (és a dir, que el servei al núvol pugui contactar amb les seves instàncies lliurement) i, en alguns casos, s'evita que aquestes tinguin accés a internet, pel que s'haurà de mantenir el contacte amb aquestes en un sol sentit, de la instància cap al servei i, en tot cas, aprofitar les respostes que s'hagin de retornar a les peticions per a enviar les dades requerides.

5 Disseny

Tenint en compte els punts explicats a la secció anterior, passarem a l'apartat del disseny del projecte. Com a primer punt del disseny, tenint en compte que el projecte estava limitat a l'ús de PHP com a llenguatge de programació principal, s'ha près la decisió de per quin framework optar. A la secció següent s'explicarà amb més detall el procès d'elecció.

5.1 Framework

Degut a la gran quantitat de frameworks de PHP que es poden trobar a la xarxa, es va decidir acotar la llista a les quatre següents opcions, que no per casualitat són els més coneguts avui en dia:

- **Laravel**: degut a la seva gran popularitat, l'habilitat de crear aplicacions seguint el model MVC (Model Vista Controlador) i la seva àmplia documentació, s'ha inclòs a la llista com una de les principals opcions. També cal destacar la senzillesa del codi, que és molt expressiu per si sol, i el seu rendiment.
- **Symfony**: framework que conté gran quantitat de "components" i "bundles" que es poden aprofitar pel projecte, a part d'estar també basat en el model MVC i disposar d'una àmplia documentació. També és destacat per la capacitat que té de crear codi modular, pel que es fa servir per a la creació de projectes a gran escala. És complex de fer servir en la seva totalitat, degut a la gran quantitat d'eines que ofereix, però la documentació oferida a la xarxa facilita molt el seu aprenentatge.
- **Zend**: és el framework principal de l'empresa, u tilitzat per a tots els productes de l'empresa, pel que a l'equip tècnic hi ha molt coneixement dels millors usos del framework que podria ser útil pel projecte i es podria aprofitar molt codi. També és molt utilitzat en la creació de projectes a gran escala degut a la gran quantitat de llibreries que es poden trobar. Malauradament, però, la seva última versió estable té més de 4 anys i és complex d'aprendre.
- **CodeIgniter**: conegut per ser un framework amb molt poc requeriment en memòria, un dels seus punts forts és el rendiment i la seva velocitat pel que fa a la comunicació amb la base de dades. Segueix l'arquitectura MVC tot i que no necessàriament la segueix, ja que podem tenir controladors sense necessitat de trobar models o vista. També és relativament senzill d'instal·lar.

Característiques	**CodeIgniter**	**Zend**	**Symfony**	**Laravel**
Alta modularitat		X	X	
Enfocat al núvol			X	X
Modern	X		X	X
Alt rendiment	X			X
Documentació de qualitat			X	X
Fàcil comunicació amb la BBDD	X		X	X
Codi senzill	X		-	X
Capacitat de reutilitzar codi		X	X	

Taula 1: Resum de la comparativa entre els diversos frameworks

Després d'un anàlisi de les diverses opcions que ofereix el mercat, s'ha près la decisió de fer servir el framework **Symfony**, degut a la seva gran capacitat

modular, el potent mapejador d'objectes que ofereix Doctrine ORM, la seva estreta vinculació amb el model MVC (treballat en gran profunditat a la universitat) i les recomanacions per part de l'equip tècnic de l'empresa.

Seguint l'objectiu del projecte, es farà servir la versió 5.0 del framework, la mes moderna fins al moment i llançada el novembre del 2019. Al llarg del desenvolupament és probable que calgui fer un canvi de versió ja que la versió tindrà, pel que s'indica a la pàgina web oficial [6], una vida útil que durarà aproximadament fins al juliol del 2020.

Per tal d'aprendre el funcionament d'aquesta eina, se seguiran uns cursos fets per la mà d'un membre de l'equip tècnic de Symfony [8], on s'explica Symfony des d'un punt de vista molt superficial i també, al final, s'acaba introduïnt a conceptes més complexos, com ara el funcionament del fitxer `Kernel.php`, que és el "core" del framework.

5.2 Model Relacional

Tenint en compte que es farà servir un framework basat en el model MVC, i després d'investigar superficialment el funcionament de Symfony, caldrà dissenyar un petit diagrama de classes a partir del qual basar la implementació.

Abans de presentar el diagrama, caldrà presentar certs conceptes sobre Symfony:

- **Controladors**: seran els encarregats de processar les crides a cadascuna de les adreçes definides o, tal com es coneixen a Symfony, *routes*. Cada crida al controlador demanarà la vista necessària, en cas de ser requerida, i la tornarà al client. La classe *AbstractController* de la qual hereten tots els controladors, conté una gran quantitat de funcionalitats que ajudaran a la implementació del projecte.
- **Vista**: l'encarregat de generar aquest apartat serà el *Twig Template Engine*, un sistema que donades unes plantilles o *templates* generades en un llenguatge que extén HTML amb funcionalitats com l'herència, té la capacitat d'assignar variables o executar algunes funcions senzilles, o fins i tot la capacitat executar funcions d'objectes que se li passin per paràmetre.
- **Model**: juntament amb Doctrine, un sistema de mapeig relacional d'objectes (ORM), caldrà definir una classe que representi cadascuna de les entitats del programa i posteriorment una classe que actuarà com a repositori per a fer peticions o accedir a cadascun dels conjunts d'aquestes dades.

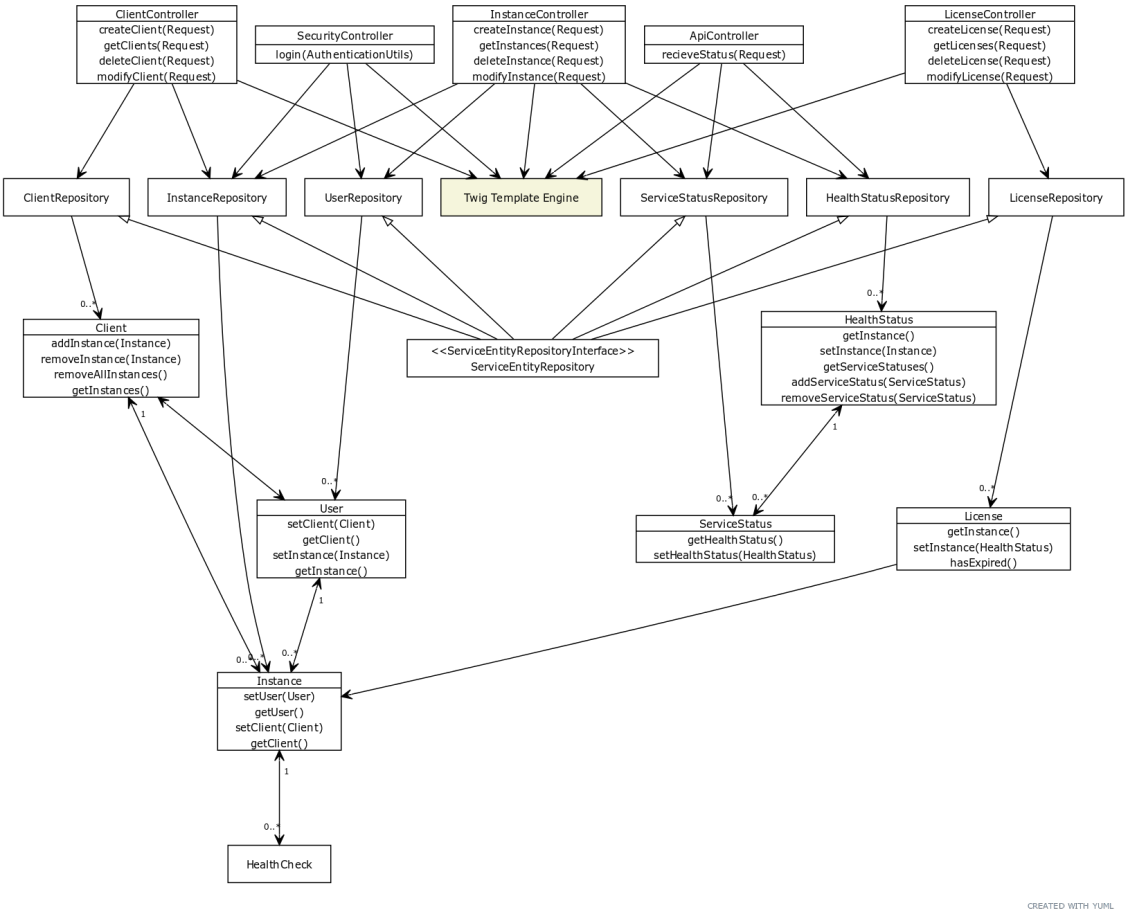

Figura 12: Diagrama de classes inicials

5.3 Arquitectura

Un cop s'ha fet l'elecció de quin framework es farà servir, cal decidir-se per les components que formaran l'arquitectura del projecte.

5.3.1 Servidor HTTP

La opció senzilla hauria sigut optar per un servidor web que ja ho deixès tot llest, com ara XAMPP que és un sistema que ja deixa preparats un Apache, una base de dades MySQL i PHP per tan sols haver de fer les configuracions adequades per a que aquests mòduls funcionin adequadament.

Degut al desig de cercar una major modularitat i nivell de complexitat fent servir *Docker*, es vol optar per anar montant els diversos mòduls per separat. En primer lloc, llavors, cal prendre la decisió de quin servidor web fer servir. Tant Symfony com PHP ofereixen també una sol·lució de servidor web que podria funcionar per crear un ràpid entorn de desenvolupament, però això causaria conflicte amb un

dels principals objectius del projecte: crear un entorn de desenvolupament el més semblant a l'entorn de producció possible.

És per aquesta raó que es va decidir investigar sobre quins són els principals servidors web i quin d'ells seria el més adeqüat pel projecte. D'entre tots els oferts al mercat, es va acotar la llista a les següents dues opcions, que també són les més conegudes, utilitzades i més fàcils d'integrar amb Symfony [7]:

- **Apache HTTP**: projecte iniciat al 1995, és el més popular des de llavors; conté una documentació molt extensa i una llarga llibreria de mòduls que permeten que es pugui connectar amb molts llenguatges de programació sense necessitat d'haver-hi d'instal·lar software extern. El mateix servidor web s'encarrega de processar les peticions i retornar-les.
- **NGINX**: projecte iniciat al 2002, d'aquest se'n pot destacar la seva arquitectura asíncrona i basada en events. Ha crescut la demanda a causa del seu consum reduït de recursos i l'habilitat d'escalar amb facilitat fins i tot en hardware poc potent. Envia contingut estàtic amb gran facilitat i està dissenyat per a delegar les peticions que requereixin dinamisme a altre software, possiblement ubicat a una altra màquina, dissenyat per a aquesta tasca. Se sol fer servir gràcies a la seva eficiència sota entorns que requereixin alta disponibilitat i eficiència.

A causa de les recomanacions per part de l'equip tècnic de l'empresa i el fet d'haver fet servir Apache prèviament al llarg de la carrera, s'ha optat per fer servir **NGINX** per a poder afegir un coneixement més a la llista de tecnologies apreses al llarg d'aquest treball de fi de grau.

El fet de fer servir NGINX implicarà la instal·lació d'una component addicional per a poder processar les crides al codi en PHP; es farà servir **PHP-FPM** per tal de resoldre aquest dubte aparegut per l'ús d'aquest servidor web: aquesta implementació de PHP està molt enfocada al rendiment i té un consum de recursos moderat, però aquests beneficis tenen el cost d'un procés d'instal·lació més complex.

5.3.2 Base de Dades

Un cop escollits el servidor web i el framework que es farà servir pel projecte, caldrà investigar sobre el sistema de bases de dades que es farà servir per aquest. En primer lloc, cal decidir-se entre fer servir una BBDD que segueixi un model relacional o no relacional:

- **SQL**: base de dades basada en el model relacional, on abans d'introduir a la base de dades la informació que desitgem, li indicarem com serà aquesta informació tindrà cada dada nova que es vagi afegint i separarem les dades per conjunts, on cada dada del mateix conjunt contindrà la informació del mateix

tipus. Un benefici d'aquest tipus de base de dades és que ofereixen eines molt efectives per evitar la duplicitat de les dades, i també permet eliminar amb facilitat dades que siguin dependents d'altres i relacionades entre si. Per tot allò que sigui informació gràfica, multimèdia i altres tipus de dades semblants, però, no és gens efectiu i és recomanable optar per un sistema no relacional. Modela les dades basant-se en el concepte de "taula", i és complex d'escalar en entorns de big data, però per mides més reduïts de dades és més efectiu i senzill de fer servir.

- **No-SQL**: base de dades contrària al sistema relacional; es basa en el concepte de "document" i està molt enfocat a l'emmagatzematge de grans quantitats de dades. El seu punt fort és que no cal desenvolupar un model de bases de dades previ al seu ús i és molt flexible. La seva quota de mercat és molt més reduïda que les bases de dades basades en SQL (que domina un $60,48\%$ del mercat [9]), però cada cop va tenint més presència al mercat i gaudeix d'una àmplia documentació. A diferència de SQL, però, no disposa d'un llenguatge estàndard per a realitzar peticions a la base de dades, aquest llenguatge dependrà del sistema gestor de BBDD que fem servir.

A causa de la naturalesa orientada a objectes del projecte i les dades que caldrà emmagatzemar, que són fàcilment representables a una taula i tampoc serà una quantitat massa notable (el nombre d'instàncies estarà bastant per sota del miler), s'optarà per una base de dades basada en **SQL**. L'únic conjunt de dades que podria ser problemàtic és l'emmagatzematge de dades del *healthcheck*, de les quals sí que n'hi haurà una quantitat molt notable (al voltant de 1-4 objectes de tipus `HealthStatus` per instància, amb 10-20 objectes de tipus `ServiceStatus` per cada `HealthStatus`), però per la mesura del projecte, que tan sols farà servir conjunts de dades de proves reduïts, no s'ha considerat necessari implementar cap mesura per controlar aquest escenari.

El que seria més adeqüat, de cara al futur del projecte, seria afegir dos sistemes de bases de dades més; una memòria cau per optimitzar l'accès a dades d'ús freqüent, implementada amb molta probabilitat en un sistema com *Redis* (ja que ja es fa servir a *openNAC* i la resta de productes de l'empresa), i una base de dades No-SQL per emmagatzemar la gran quantitat de dades recollides de cadascun dels *healthchecks* rebuts de les diferents instàncies del producte, per tal de no carregar tant la BBDD principal que contindria les dades bàsiques del programa (sobre clients, instàncies, llicències...) que seria implementada en *ElasticSearch*, que també és utilitzat per l'empresa.

Un cop decidit el tipus de base de dades que es farà servir, cal prendre la decisió de quin sistema gestor de BBDD es farà servir. A continuació s'esmenten les tres principals opcions que s'han valorat per al projecte:

- **MySQL**: és conegut per tenir el millor llenguatge per a escriure i fer peticions complexes sense gaire dificultat. Disposa també d'una llarga varietat

de motors d'emmagatzematge (ón *InnoDB* és el motor utilitzat per defecte) i té una velocitat de lectura excel·lent. El major problema que té és la velocitat d'escriptura i també té problemes de concurrència quan es barreja amb operacions d'escriptura.

- **MariaDB**: es podria considerar com el fill de MySQL, ja que el van crear treballadors d'aquest després de la compra d'Oracle [11]. Té actualitzacions de seguretat molt freqüents, demostrant un gran interès en l'eina per part de l'equip tècnic, i és molt compatible amb gran quantitat de llenguatges. També és destacable pel seu gran rendiment en conjunts de dades reduïts o d'un mida menor al big data i per fer servir el mateix llenguatge per les peticions que MySQL. També ofereix un sistema de sharding molt efectiu per a la creació de clústers. Un altre punt a favor és què és el sistema gestor de BBDD, utilitzat a totes les eines de l'empresa, pel que resulta molt familiar.

- **PostgreSQL**: d'entre els tres esmentats, aquest seria el més òptim en conjunts de dades molt grans (tot i que amb poca diferència respecte a l'anterior). La seva documentació no és massa bona, però gràcies a haver-la fet servir prèviament a la carrera és un sistema que resulta familiar.

Per raons de cost i pel fet de ja haver escollit altres eines amb les quals no hi havia gaire familiarització, s'ha optat per fer servir **MariaDB**, ja que també es l'eina per defecte utilitzada a l'empresa per tots els seus projectes.

5.3.3 Directori d'Usuaris

Pel que fa al *Active Directory*, i per no voler fer servir el directori de l'empresa per raons de seguretat, caldrà crear un directori actiu senzill que contingui únicament un usuari amb rol d'administrador, ja que amb un usuari en tindrem prou per demostrar l'eficàcia de l'aplicació.

Per tal d'utilitzar el mínim de recursos per aquest aspecte no crucial del projecte, es farà servir la implementació d'un servidor LDAP senzilla basada en *ApacheDS* [12], amb un repositori smiple fet a mà.

6 Implementació

Començarem amb la implementació del projecte explicant breument l'entorn de desenvolupament que es farà servir, i posteriorment ens endinsarem a la implementació del projecte en si i problemes trobats al llarg de la implementació d'aquest.

6.1 Entorn de desenvolupament

Tots els fitxers que s'aniran descrivint es troben al directori `/docker` del projecte, i totes les variables que es vagin utilitzant als fitxers es podran trobar al fitxer ocult `.env`. Lògicament, les contrassenyes trobades als fitxers no estarien accessibles a un repostori públic, pel que totes les dades sensibles que es puguin trobar al projecte son de prova.

La descripció del sistema sencer es podrà trobar al fitxer `docker-compose.yml`, on es defineixen les carpetes que contenen el "context" de cadascun dels contenidors (també conegudes com a `Dockerfile`), les variables que es faràn servir, els ports que caldrà exposar (a la resta de contenidors) o publicar (a la xarxa), dependències entre contenidors i volums que farà servir.

Abans de començar a presentar el contingut de cadascun dels contenidors, però, caldrà introduïr a la funcionalitat "volumes" de *Docker*. Aquesta funcionalitat permet la creació directoris absoluts personalitzats dins del contenidor en qüestió [13]. Es permet la creació de directoris "clonats" que agafen un directori de l'amfitrió (el sistema que executarà *Docker*) i el copien al contenidor. Aquest directori, però, no tan sols està clonat, sinó que està també "vinculat", ja que si modifiquem un fitxer a l'amfitrió, el fitxer es veurà també modificat al contenidor que contingui el directori.

En general es desaconsella l'ús d'aquest tipus de volums als contenidors de bases de dades, ja que aquests directoris no són gestionats per *Docker* i, per tant, es redueix la portabilitat de l'eina. Pel projecte com no haurem d'anar canviant de màquina amb facilitat, ja ens anirà bé per a poder mantenir un control de les dades emmagatzemades des del sistema amfitrió.

També existeixen configuracions pels requeriments de consistència de dades, on la configuració per defecte és que tant contenidor com amfitrió disposin dels mateixos elements en tot moment, o la configuració `cached`, utilitzada als contenidors encarregats de carregar el projecte trobat a la carpeta `/src`, que permet que les dades modificades a l'amfitrió arribin amb un petit delay al volum del contenidor.

El canvi d'aquesta configuració és molt notable als sistemes amfitrions que funcionin amb un sistema operatiu Mac OSX [14]. Amb el canvi d'aquesta configuració, s'ha aconseguit que el temps de càrrega de les pàgines del portal administratiu passés d'una mitja de 10 segons a mig segon, més o menys.

Una altre opció que ofereix Docker al fitxer de descripció del sistema a arrencar és la d'indicar en quin ordre s'han d'anar executant cadascun dels contenidors, mitjançant l'opció de configuració `depends_on`. El problema que té això és que espera a que s'hagi executat el contenidor, no els serveis què conté el contenidor, pel que si volem executar alguna funció inicial des del contenidor que conté el motor PHP que afecti al contenidor amb la BBDD, caldrà cercar alguna alternativa per evitar que s'executi la comanda abans de que s'hagi obert.

Gràcies al script `wait-for-it.sh` podem comprovar la disponibilitat d'un host i un port TCP. Amb l'execució d'aquesta comanda, a la qual li passaríem el contenidor i port del qual esperem alguna resposta abans de fer res, i l'execució de tot el sistema es paralitza fins que no s'hagi activat el port en qüestió, per a poder executar posteriorment la comanda desitjada sense problemes.

Cal destacar també les configuracions per permetre comunicacions amb els contenidors, mitjançant `expose` obrim el port dins el mateix sistema docker, però no a l'exterior, i amb la configuració `ports`, permetrem que es connectin dispositius al contenidor des de l'exterior, mapejant el port del contenidor amb un de la maquina amfitriona (per exemple, si fem `80:81`, exposarem el port 80 del contenidor al port 81 de l'amfitrió).

A continuació s'explica la funció de cadascun dels contenidors i quin es el procés que se segueix per a que Docker pugui executar-los:

- **Contenidor** `ldap`: contindrà la implementació senzilla d'un servidor LDAP que serà executada amb un fitxer de configuració `.ldif` que descriurà el contingut del directori actiu, que serà un usuari amb rol d'administrador, nom d'usuari `admin` i la contrassenya `opennac`. D'aquest n'exposarem el port 10389, per a que es pugui tenir accés des del contenidor `php-fpm`.

- **Contenidor** `database`: contindrà la base de dades de l'aplicació utilitzant la versió més recent de *MariaDB*. Exposarà el port 3306 per permetre connexions del motor PHP a la BBDD. Tot el contingut de la base de dades es clonarà a l'amfitrió.

- **Contenidor** `php-fpm`: contenidor encarregat d'executar tot el codi, i que per tant tindrà una copia del directori amb el codi font, aquest contenidor haurà d'esperar a que s'hagin executat els dos contenidors anteriors abans d'executar-se i per a esperar a l'execució de la base de dades farem servir el script `wait-for-it.sh`. Abans d'executar la darrera comanda es descarregaran totes les llibreries necessàries per poder llegir fitxers ZIP, contactar amb el directori actiu i altres extensions de PHP necessàries (amb la comanda `docker-php-ext-install`), com ara l'extensió per a poder establir comunicació amb la base de dades. També necessitarem instal·lar `composer`, és a dir el gestor de dependències de Symfony i `yarn` i `nodejs`, que serviran per a processar els fitxers `.js` i `.css` del projecte. Si l'execució del script funciona correctament, es passa a executar una migració de la base de dades per tal de comprovar que es puguin carregar correctament i que s'estigui fent servir una versió de la BBDD que estigui al dia amb la versió del projecte. Finalment exposem el port 9000.

- **Contenidor** `nginx`: contindrà el servidor web i farem públics els ports 443 i 80, permetent l'accés a ells des de l'exterior. Pel que fa a la configuració del servidor NGINX, no farem res destacable a part de configurar un proxy per redirigir les peticions al contenidor `php-fpm`. Això ho aconseguirem amb la

configuració del fitxer `default.conf` que hi ha a la carpeta `nginx/conf.d`. Per la resta de fitxers, ens basarem en les configuracions bàsiques esmentades a la documentació de Symfony [7] i alguns fitxers d'exemple als quals he aconseguit accés gràcies a l'equip tècnic de l'empresa. Aquest directori també necessitarà accès al directori del codi font.

Tant el contenidor `nginx` com el contenidor `php-fpm` necessitaran accés al codi font perquè `nginx` serà l'encarregat de servir tot el contingut estàtic (com imatges, JavaScript, CSS...) al client i `php-fpm` s'encarregarà d'executar tot el codi PHP necessari.

Com a resum, aqui tenim un petit diagrama que mostra la topologia final del sistema que s'ha definit com a entorn de desenvolupament. Totes les components s'han dividit el màxim possible per tal de satisfer el desig d'establir una gran similaritat entre l'entorn de desenvolupament i l'entorn de producció.

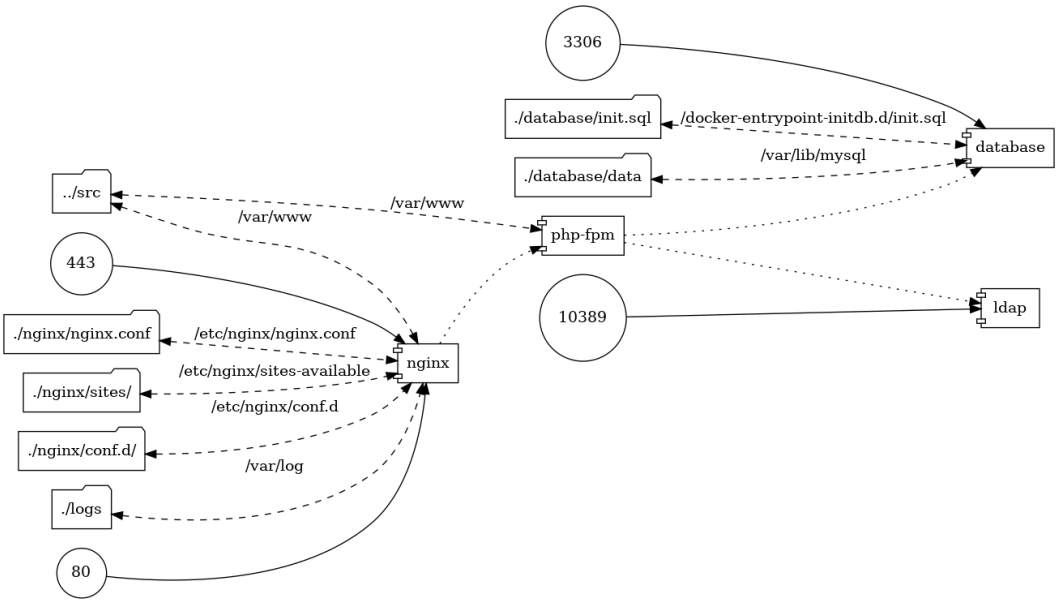

Figura 13: Topologia del sistema de desenvolupament definit

6.2 Base de dades

Per a la definició de la base de dades caldrà llistar les diferents entitats del servei i establir les relacions que necessitarem entre elles. Totes les classes les crearem amb l'eina de creació d'entitats de Symfony, que afegeix les relacions necessàries per cadascuna de les entitats i s'encarrega, fent servir Doctrine, de traduïr aquestes a taules a la base de dades. Aquesta eina la podrem executar amb la comanda `php bin/console make:entity`, al directori del projecte al contenidor `php-fpm`:

- **Usuari**: classe per defecte de Symfony. Com l'accés al portal administratiu es farà mitjançant LDAP, tan sols farem servir la classe per a crear usuaris per a permetre l'autenticació al sistema d'API fent servir un usuari comú per a totes les instàncies d'un client.

- **Client**: taula que definirà el perfil de cadascun dels clients registrats a l'eina. Cada client tindrà un usuari únic que farà servir per a l'accés a l'API del servei al núvol.

- **Instància**: taula principal que defineix cadascuna de les instal·lacions del producte a l'infraestructura d'un client. El concepte d'instància pot fer referència a sistemes muntats en clúster, pel que l'objecte creat sempre apuntarà al node "principal" de l'infraestrucutra. Aquest objecte contindrà un "token", que és el que haurà de fer servir per a poder accedir al servei al núvol i una data d'expiració, per indicar quan caduca aquest token. En cas d'haver caducat, no se li permetrà accès al servei. També s'inclourà una referència a la llicència que tindrà assignada; en cas de no tenir-ne, aquest camp tindrà valor nul. Altres camps que contindrà inclouen l'adreça IP de la instància, una curta descripció (que normalment serà la localització de la instància) i informació addicional que pugui servir a l'equip de *Professional Services*, com ara la manera d'accedir a la instància o altres detalls destacables.

- **Llicència**: taula que definirà la llicència generada per a una instància; aquesta contindrà un identificador que la relacionarà amb la instància la qual s'està activant, la data de caducitat d'aquesta llicència (diferent a la caducitat de la instància), un camp amb l'identificador del fitxer generat de llicència i un altre amb l'identificador de la petició de llicència, que es generarà des de l'*openNAC* (que serà un número de sèrie).

- **Dades del Healthcheck**: aquestes dades les dividirem en dues taules, descrites a continuació:

  - Health status: que farà referència a l'estat d'un node de l'infraestructura de la instància; això vol dir que, si per exemple tenim un sistema en clúster amb 3 *openNAC*, es crearan 3 files a la taula, una per cada node. Totes aquests estats, però, apuntaran al mateix identificador d'instància, per indicar que totes formen part del mateix sistema. Contindrà els camps principals que conté el JSON de healthcheck per a cada node del sistema, tal com s'ha descrit anteriorment.
  - Service status: que farà referencia a l'estat d'un servei d'un node; per cada node de l'infraestructura es crearan diversos objectes d'aquest tipus, i tots apuntaran al mateix estat de node (Health status). Contindrà tots els camps referents a cadascun dels serveis que conté el JSON de healthcheck per cada node del sistema, tal com s'ha descrit anteriorment, d'entre ells tindriem l'output de l'execució de la comanda per a comprovar l'estat del servei o el nom i una breu descripció del "check" que s'ha realitzat.

A continuació es pot observar un diagrama amb la definició que s'ha generat de cadascuna de les taules:

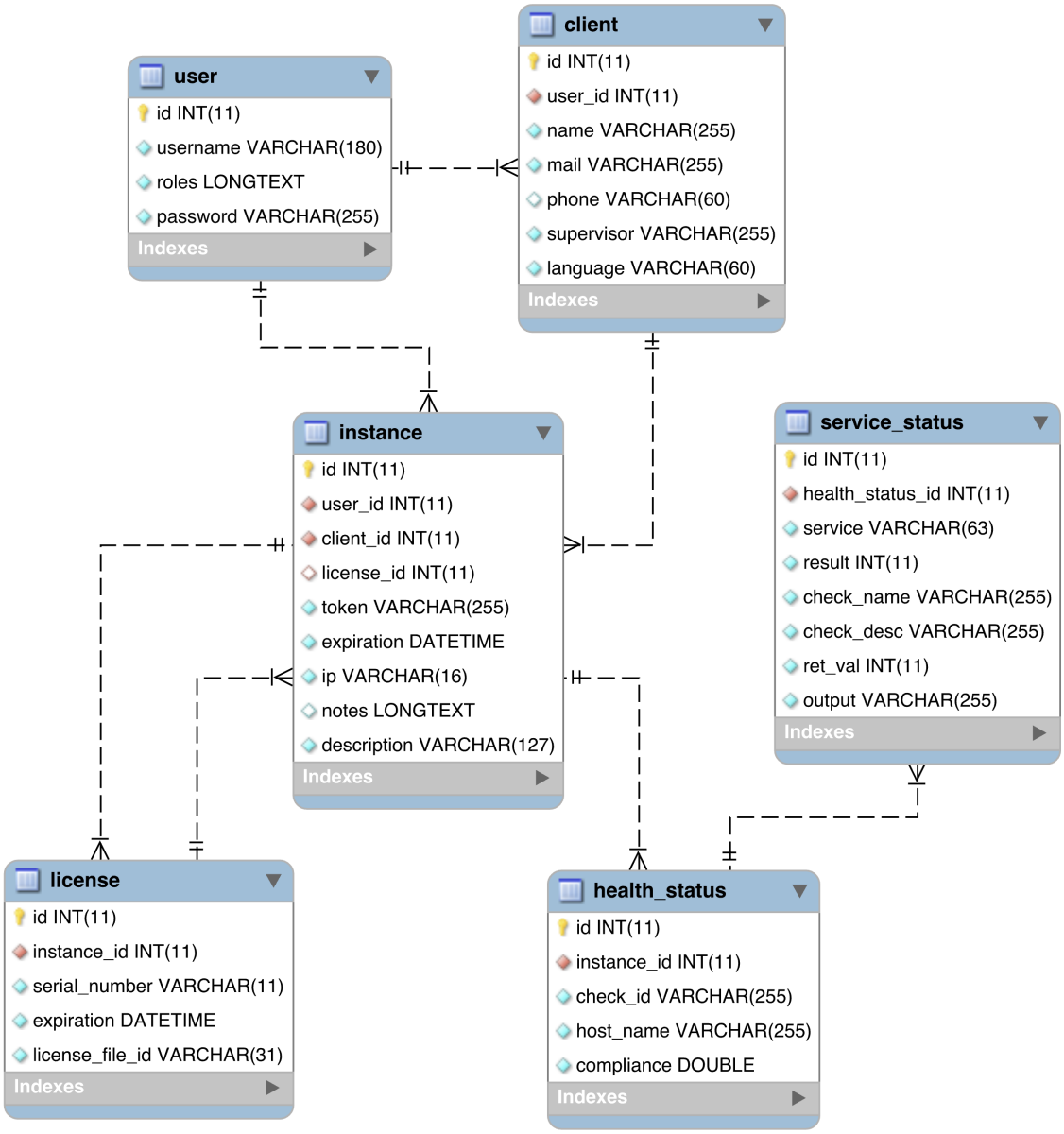

Figura 14: Topologia de l'estructura de la DDBB definida

Una opció que s'ha valorat és la d'eliminar l'expiració d'una instància i fer servir l'expiració de la llicència, per tal de facilitar més l'esquema i perquè, en principi, si a un client li caduca la llicència, no hauria de tenir l'accés permès al servei al núvol.

Això, però, entra en conflicte amb la idea d'evitar el mínim contacte amb les instàncies del client, ja que per a poder revocar una llicència caldria connectar-se a la instància i eliminar la llicència que estigui en ús o si es volgués modificar la caducitat d'aquesta, sigui per ampliar-la o escurçar-la, ens passaria el mateix.

De manera que així, en cas de ser necessari, se li pot revocar fàcilment l'accés al servei al núvol al client que es desitgi sense problema i s'evita haver de connectar amb les màquines dels clients.

Per a obtenir els objectes relacionats en el cas de les relacions d'un a més d'un, tan sols caldrà filtrar la taula adient pel nombre d'identificador de l'objecte principal; si volem llistar les instàncies d'un client, tan sols caldrà filtrar les instàncies que tinguin l'identificador del client determinat. De tota aquesta tasca, però, s'encarrega Symfony i el seu sistema de "repositoris" amb l'eina Doctrine de manera automàtica.

6.3 Directori del projecte

Un cop s'ha parlat sobre l'entorn de desenvolupament i la base de dades del projecte, es passa a fer-ho sobre l'estructura del projecte, seguint els models per defecte de Symfony. Tots els fitxers esmentats a continuació es poden trobar dins de la carpeta `/src` del projecte principal. Seguint l'ordre alfabètic de les carpetes del projecte, cadascuna d'elles conté:

- `assets`: amb les carpetes `css` i `js` dins, trobarem els fitxers `.scss` i `.js` sense compilar. Aquests es compilen amb la comanda `yarn encore dev`, executada al contenidor `php-fpm`, i els resultats de la compilació es trobaran a la carpeta `public`, esmentada posteriorment.

- `bin`: aquí podem trobar l'eina més útil del framework, el fitxer `console`, encarregat d'executar des de les crides esmentades als diversos punts (com ara `make:entity`), fins a eines de depuració per veure les rutes definides al nostre sistema. La gràcia d'aquest fitxer és que, amb l'ús de les *recipes*[4], podem instal·lar totes les eines que desitgem per tal de facilitar-nos el procés de desenvolupament. El fitxer de testeig `phpunit` no s'ha fet servir per aquest projecte.

- `config`: aquí s'emmagatzemen les configuracions generals de Symfony i dels plugins que es facin servir. Aquí estarà definida la connexió amb el servei LDAP (al fitxer `services.yaml`) i la configuració dels proveïdors d'usuaris, la pàgina de registre i els autenticadors que fer servir en funció de les rutes visitades (al fitxer `security.yaml`).

- `migrations`: carpeta on s'aniran emmagatzemant tots els canvis que va fent a la base de dades el sistema Doctrine.

[4]Sistema encarregat d'afegir noves comandes a `bin/console` i nous "bundles" (o plugins), creant els fitxers necessaris, modificant i/o afegint noves línies a fitxers ja existents. Aquestes eines faciliten moltes tasques de cara a la implementació i disseny de projectes, i la seva desinstal·lació també és relativament senzilla.

- `node_modules`: carpeta que descarrega les dependències necessàries pel funcionament de `webpack-encore`, el sistema encarregat de compilar els fitxers de la carpeta `assets`.

- `public`: carpeta on es poden trobar els recursos estàtics de l'eina; els fitxers compilats per `webpack-encore` i les rutes públiques de l'eina per a poder executar les peticions AJAX. La carpeta també conte el fitxer `index.php`, que serà per on passaran totes les crides al portal o API i el punt d'entrada al servei al núvol.

- `src`: carpeta amb el codi font del servei al núvol, que s'explicarà a la següent subsecció.

- `templates`: carpeta amb les plantilles Twig de tot el portal (tots els fitxers `.html.twig`) que Symfony s'encarregarà de compilar en temps d'execució i retornarà a l'usuari.

- `vendor`: carpeta amb totes les dependències generades pel gestor de paquets Composer, necessàries pel correcte funcionament de Symfony.

- `var`: carpeta amb logs i memòria cau de Symfony. També s'emmagatzemen dins d'aquest directori, a la carpeta `uploads`, totes les dades relacionades amb les llicències, emmagatzemant les dades de petició de cada llicència (al directori `uploads/data`) i les dades de cadascuna de les llicències generades (a `uploads/licenses`).

- **Fitxers** `composer.json`**,** `package.json` **i** `yarn.lock`: fitxers amb les dependències necessàries pel funcionament de Symfony i els altres mòduls i plugins relacionats.

- **Fitxer** `webpack.config.js`: fitxer amb la configuració de `webpack-encore`, on s'indica a quin directori pertany cada fitxer `.js` de la carpeta `assets` i altres configuracions per a la correcta compilació dels fitxers `.scss` i `.js`.

Dins de la carpeta `config` també es poden configurar les rutes (o *paths*) del projecte, però és recomanable anar-les definint als controladors amb l'ús de les *annotations* de Symfony, que permeten definir amb detall en comentaris de PHP cadascuna de les rutes, permetent assignar noms, mètodes HTTP acceptats i altres opcions sobre la funció que li pertoqui.

6.4 Estructura del projecte

Caldrà dividir la presentació de l'estructura del projecte en l'apartat de "back-end", que ha sigut el focus prioritari del projecte, i en una breu explicació de la implementació del "front-end".

6.4.1 Backend

El projecte s'ha estructurat seguint el model MVC que segueix Symfony. Cal indicar però, que realment Symfony ofereix les eines per a facilitar la tasca del Controlador, però per banda del Model i de la Vista l'elecció és de l'usuari.

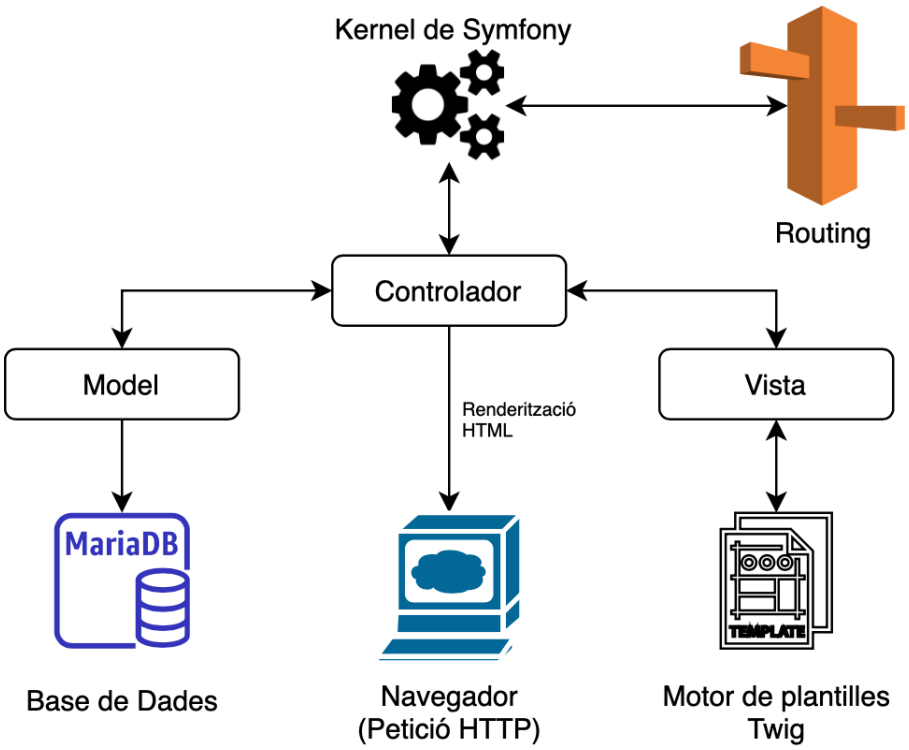

Figura 15: Diagrama resum del model MVC de Symfony

Per defecte, s'ofereix fer servir Doctrine ORM per la banda del Model, que funciona molt adeqüadament i facilita molt la integració del codi amb la base de dades, i Twig per a la vista, que genera els documents HTML per a cadascuna de les crides amb un sistema basat en plantilles molt eficiens, però Symfony en si no depèn de que fem servir Doctrine o Twig. Es podria haver basat el codi en connexions PDO i en HTML bàsic i funcionaria exactament igual [16].

Seguint l'ordre de la subsecció anterior, separarem l'explicació de l'estructura del projecte per directoris, en ordre alfabètic. Tots els directoris i fitxers esmentats es poden trobar a la carpeta `src/src/` del projecte.

`Controller`

Seguint un dels objectius del projecte, s'ha volgut aprofitar al màxim el fet de fer servir Symfony, explotant al màxim totes les utilitats oferides per aquest framework amb l'objectiu de facilitar la implementació d'aquest.

Degut a fer servir el model amb Doctrine, al llarg del codi dels controladors es veuran crides a mètodes com ara `$this->getDoctrine()->getManager()` o `$this->getDoctrine()->getRepository(X::class)`, que serviran per a fer servir el controlador d'entitats de Doctrine, encarregat de persistir i emmagatzemar tots els canvis de les taules de base de dades i per a obtenir les entitats que pertanyen a una taula, respectivament.

També, a causa de fer servir Twig, es podran observar crides també a `$this->render('...', [...])`, que és el mètode encarregat de generar la plantilla indicada amb els paràmetres que li passem i retornar la resposta amb el HTML generat a l'usuari.

Com podem observar l'ús de `$this` en Symfony, als controladors (que extenen `AbstractController`), ofereix moltes comoditats a l'hora d'implementar l'aplicació. Aquests "helpers" també serveixen per a fer respostes; un exemple és el que s'ha esmentat prèviament per a la generació dels documents HTML, però també disposem de "helpers" per a retornar JSON (`$this->json(...)`), fitxers (`$this->file(...)`).

L'altre fet destacable és que a cada mètode del controlador afegirem el mètode `$this->denyAccessUnlessGranted('IS_AUTHENTICATED_FULLY')`, que ens servirà per a verificar que l'usuari que està realitzant la petició estigui correctament registrat. En cas de no ser així, se'l redirigirà a la pantalla d'inici de sessió.

Una altra utilitat que es pot observar és el fet de passar paràmetres als mètodes, com ara l'opció de rebre per paràmetre entitats definides pel projecte: si la petició conté un identificador, es pot extraure directament l'objecte de la base de dades que tingui l'`id` indicat, sense haver de fer les corresponents crides al repositori.

Una altra opció de paràmetre que es fa servir és la `SerializerInterface`, utilitzada per a poder definir els JSON de resposta còmodament al nostre gust.

Dins el directori `Controller` trobarem els diversos controladors del programa. S'ha decidit separar els controladors per secció del portal, per a poder afegir noves seccions al portal amb la major facilitat possible.

- `ApiController`: classe encarregada de gestionar les crides rebudes a la ruta `/api/*`. A dins hi trobem el mètode `recieveStatus`, encarregat de rebre la informació del healthcheck d'una instància d'openNAC i processar-la. El mètode recorrerà el JSON i crearà un objecte del tipus `HealthStatus` per cada node de la infraestructura i relacionarà amb aquest els respectius objectes del tipus `StatusService`.
- `ClientController`: classe encarregada de gestionar les crides rebudes a la ruta `/client/*` i `/data/clients`. Dins hi trobem mètodes senzills per a la creació d'un client en cas de rebre una petició `POST`, creant un usuari amb el nom definit, l'obtenció del llistat de clients en format JSON o la plantilla en HTML i l'opció d'esborrar clients en funció del seu indicador.

- `InstanceController`: aquesta classe s'encarrega de gestionar les crides rebudes a `/instance/*` i `/data/instance/*`. També disposem de les mateixes crides que al controlador previ, per a crear instàncies, assignant a aquestes l'usuari que li pertoca, el llistat en JSON o la plantilla en HTML, d'obtenir la informació de la qual es disposa sobre aquest i d'esborrar-ne una amb l'identificador, esborrant també els fitxers de llicència que tingui assignats, en cas d'haver-hi.

- `LicenseController`: al igual que les classes anteriors, es permeten les accions de llistar en JSON o HTML, crear i esborrar, tot i que no es permet modificar en aquest cas ja que, tal com s'ha comentat fins ara, per a renovar una llicència cal generar una nova. També s'ofereix l'opció de descarregar llicències en fitxers `.zip`.

- `PanelController`: aquesta és la classe més senzilla, simplement retorna la pàgina principal en cas de trobar-se a la ruta `/`.

- `SecurityController`: controlador generat per la instal·lació del mòdul de seguretat de Symfony, amb l'execució de la comanda `make:auth` mitjançant `bin/console`. Aquest controlador genera la vista de l'inici de sessió en cas de rebre una petició a `/login`. El procés d'autenticació dependrà del que tinguem configurat a `../config/packages/services.yaml`, que com s'ha comentat previament, serà mitjançant un LDAP. Definirem la ruta `/logout` al fitxer `../config/routes`, per a que quan es faci un logout es cridi a la ruta i se'ns redirigeixi a la pàgina d'inici de sessió.

`DataFixtures`

Directori amb una classe, `AppFixtures`, que servirà per a generar una falsa base de dades cada cop que s'executi la comanda `./fixtures` des del directori `/docker` del projecte o s'executi `doctrine:fixtures:load` al contenidor `php-fpm`, mitjançant `bin/console`.

`Entity`

Una funcionalitat molt còmoda de Doctrine és el fet que, si per exemple tenim un objecte que té un array d'altres entitats, les entitats relacionades, si així s'ha definit a les "annotations" amb l'opció `orphanRemoval`, que serveix per a representar atributs privats que es faràn servir únicament per a l'entitat que les tingui assignades, i per tant esborrar l'objecte pare implicarà que s'esborri la resta de les entitats incloses dins d'aquest atribut (que es un array) automàticament.

Aquesta és la carpeta on trobarem definides totes les classes utilitzades al projecte, i totes s'han anat definint fent servir la funcionalitat `make:entity` de `bin/console`. Aquesta funcionalitat, quan s'executa, genera un formulari on es va

preguntant el nom de l'entitat a crear i ajuda a definir també els atributs d'aquestes i les relacions amb altres entitats (de tipus `OneToMany` o `OneToOne`), si s'escau.

L'altra opció de la qual disposem és `cascade`, que permet configurar efectes en "cascada" sobre entitats relacionades. Si fem servir, per exemple, la configuració `persist`, amb persistir l'objecte que tingui la relació principal, persistirem els canvis també sobre les entitats que hagi modificat.

Totes les classes faran servir el generador d'identificadors de Doctrine, que els genera en forma d'enter de manera ascendent.

A continuació explicaré els punts destacables de cada definició de les entitats més destacables, sense entrar en detall en cadascun dels atributs, punt que ja s'ha exposat al subapartat de la definició de la base de dades.

- `Client`: cada element d'aquesta entitat pot tenir diverses instàncies assignades, pel que dins hi tenim definida una relació del tipus `OneToMany` amb la classe `Instance`, amb l'opció `orphanRemoval=true` per a eliminar totes les instàncies relacionades en cas de ser eliminat. El mateix, però en una relació `OneToOne`, ens passarà amb la classe `User`.
- `HealthStatus`: aqui hi definirem la relació `OneToMany` amb cadascun dels `ServiceStatus` que formin part d'aquest node, configurat a `orphanRemoval=true`, i tindrem una relació inversa `ManyToOne` amb la classe `Instance`.
- `User`: aqui hi definirem una relació `OneToOne` amb el client que se li hagi assignat. El camp password serà buit, ja que utilitzarem el token de les instàncies per a iniciar sessió a l'API del servei al núvol, i al fer servir el sistema d'autenticació mitjançant LDAP, no es registraran els usuaris que hagin iniciat sessió en aquesta taula.
- `License`: aqui establirem una relació `OneToOne` amb la classe `Instance`, amb l'opció `cascade` configurada a `persist`.

`Form`

En aquest directori, sense entrar en gaire detall, trobarem les classes encarregades de crear cadascun dels formularis corresponents a cadascun dels controladors de les diverses entitats. Cadascuna estarà personalitzada per satisfer els requeriments del controlador: el formulari `ClientType`, per exemple, contindrà un camp extra que representarà el nom d'usuari que crearem per a fer el login mitjançat l'API. El formulari `LicenseType` està configurat per a generar un camp per a pujar fitxers amb les seves corresponents limitacions, com ara que el fitxer hagi de ser JSON o que no tingui una mida massa gran.

Tots extenen la classe `AbstractType`, que és la que ofereix, amb els mètodes abstractes predefinits, totes aquestes funcionalitats.

`Migrations`

Aquesta carpeta contindrà els arxius PHP que representaran cadascun dels canvis realitzats a la base de dades.

`Repository`

Aquest directori contindrà cadascuna de les classes generades per la comanda `make:entity` de `bin/console`, que faran referència als repositoris per a accedir a les dades de les taules. Obtindrem aquests repositoris cridant a `$this->getDoctrine()->getRepository(X::class)`, on `X` serà el nom de l'entitat de la qual volem el repositori.

`Security`

Dins aquest directori trobarem dues classes: l'encarregada d'autenticar els intents de login si l'accés s'efectua per base de dades (`LoginFormAuthenticator`), que està desactivat temporalment ja que farem servir l'autenticació mitjançant LDAP i l'encarregada de realitzar l'autenticació de l'API mitjançant el "token" d'una instància (`InstanceAuthenticator`).

a causa que el primer s'ha generat de manera automàtica amb l'execució de la comanda `make:auth`, és més interessant parlar sobre la classe encarregada de l'autenticació de l'API.

La classe `InstanceAuthenticator`, llavors, comprovarà que la petició rebuda contingui les capçaleres HTTP `X-Cloudnac-Username` i `X-Cloudnac-Token`. En cas de tenir aquestes capçaleres, es passaria a verificar, en ordre, que la instància amb el token o token previ indicat existeixi, i en cas de ser així es comprova que l'usuari encaixi amb l'usuari emmagatzemat a la instància que encaixa amb el token.

Si encaixen les dues capçaleres, es passa a comprovar si el token ha expirat, i en cas de no ser així se li retorna l'usuari al controlador i se li assigna a l'objecte de tipus `Request`, que es pot observar que es fa servir a la major part de mètodes dels controladors.

Si el procès d'autenticació falla per algun motiu, s'impedirà l'execució del mètode sol·licitat i es retornarà un missatge HTTP tipus `401`. El procés es resumeix en el diagrama presentat a continuació:

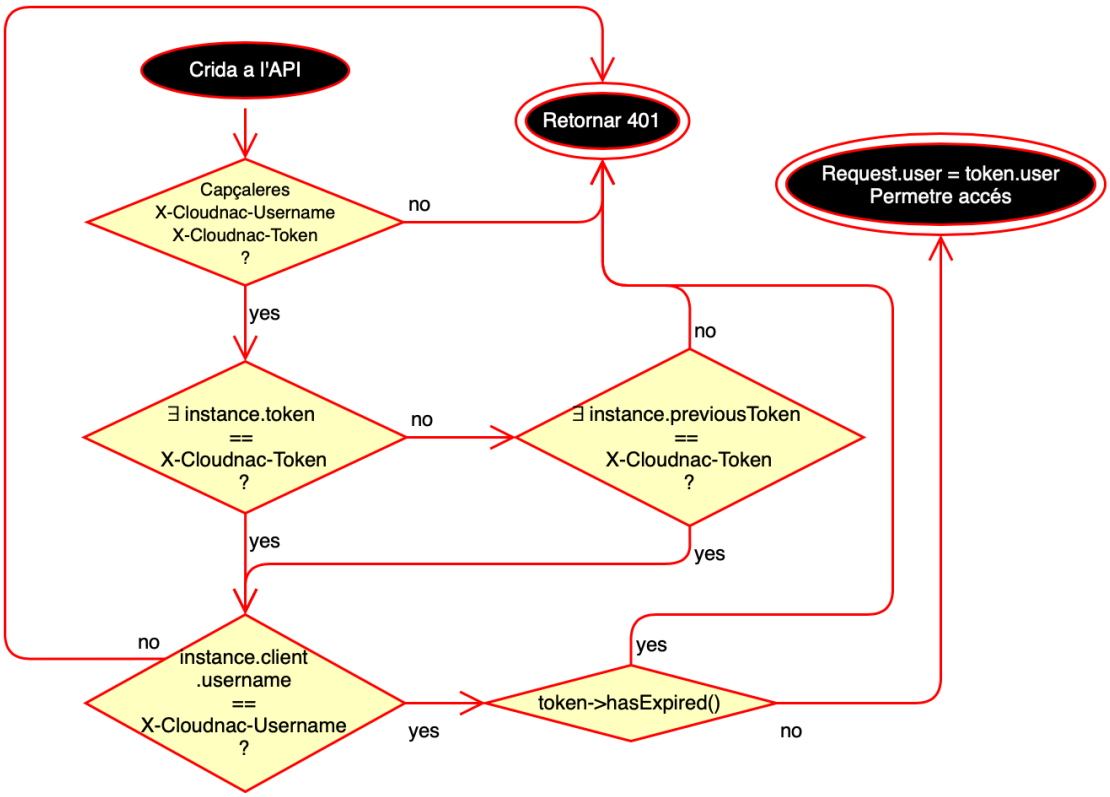

Figura 16: Diagrama resum model d'autenticació a l'API

La idea que hi ha darrera de l'atribut `previousToken` és la de poder actualitzar o canviar el token de la instància de la màquina del client de manera automàtica. Des del controlador `ApiController` es gestionaran les crides dels clients i, en cas de rebre una que s'hagi autenticat fent servir el token previ li enviarem un missatge d'alerta indicant que el token rebut està obsolet i que s'ha canviat per un de nou, inclòs al missatge.

Un detall que es mostra al diagrama és que la comprovació d'aquests es realitza successivament, una primera que l'altre. Això no és cert, a causa que la comprovació es realitza mitjançant una "query" a la base de dades que utilitza la funció `OR`; la petició quedarà com que es volen extraure les entitats que encaixin amb una o les dues condicions. No es podrà encaixar amb les dues condicions, però, ja que aquests atributs sempre tindran valors diferents.

El JSON rebut per la instància serà de l'estil mostrat a l'imatge següent. La tasca de substituir el token configurat per el nou quedarà delegada a l'*openNAC*. Aquest token sera l'únic que es podrà fer servir de cara a les futures peticions:

Figura 17: Resposta de l'API a l'autenticació amb un token obsolet

Fitxer `Kernel.php`

Aquest fitxer, que no té directori per ell mateix, és l'encarregat de configurar tots els "bundles" utilitzats per l'aplicació i de proveïr-los amb la configuració de l'aplicació. Crea un "contenidor" abans de servir les peticions amb el mètode `handle()`. Aquest fitxer, també, serà l'encarregat de carregar totes les rutes definides i de realitzar les crides als mètodes pertinents en funció de la petició que s'hagi rebut, al igual que activar els "bundles" necessaris per a l'execució d'aquest.

Diagrama

A continuació es mostra un diagrama de classes que resumeix tot els punts esmentats fins ara.

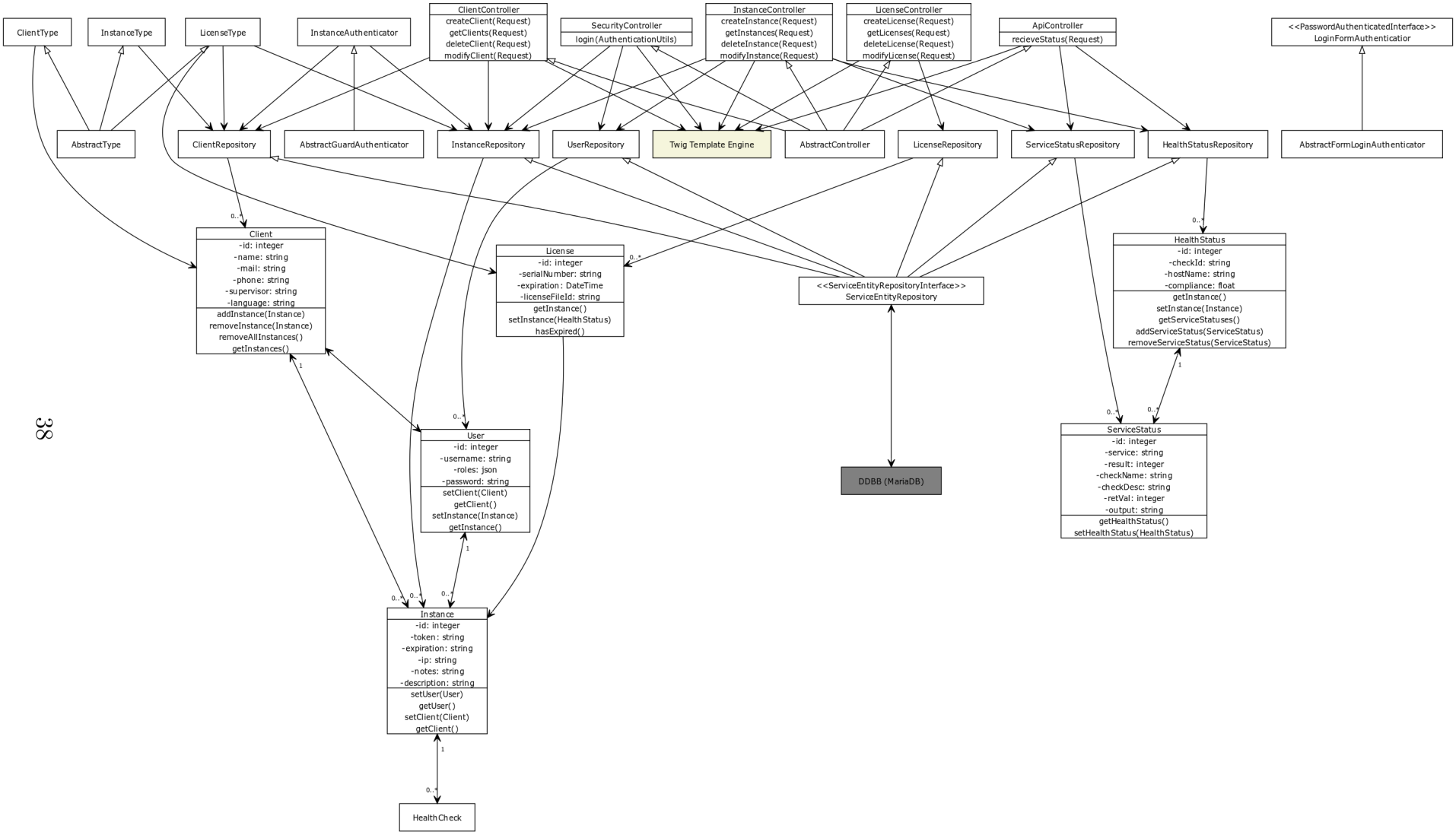

Figura 18: Diagrama de clases del model del backend

6.4.2 Frontend

Tot i que el principal focus del projecte és el "backend", cal introduïr breument a tot allò que s'ha fet per a la definició del front-end del projecte.

Per a tots els HTML generats farem servir Twig, que és un sistema de templates molt semblant al framework Django a causa que la sintàxi està basada en el seu sistema de templates [17], el qual s'ha fet servir a l'assignatura de Software Distribuït de la carrera, fet que va ajudar al procés de familiarització amb aquest motor.

Dins el directori `/src/templates/` hi trobem el fitxer `base.html.twig`, que serà la plantilla principal del portal, que tindrà a la capçalera el vincle amb el fitxer `app.css` generat per `webpack-encore`. En cas d'haver-hi usuari es dibuixarà també el nav-bar superior que tindrà el logotip de l'aplicació i el botó de logout, junt amb la barra de navegació lateral que contindrà l'accés a totes les funcionalitats del portal.

Quan es diu "en cas d'haver-hi usuari", vol dir que si l'usuari no està amb la sessió iniciada no se li mostraran les nav-bar esmentades i tan sols s'inclourà el `.css`. Això serà útil per a la definició de la plantilla de login, que està dins del directori `security/`, al fitxer `login.html.twig`. Com l'usuari encara no haurà iniciat sessió, aquest no existirà i se li mostrarà la pantalla d'inici de sessió.

Pel que fa a la resta de plantilles, no hi ha gran cosa a destacar, ja que totes segueixen el mateix patró: tenen una taula buida, amb un botó de crear i un formulari per a crear una entitat que rebrà del controlador que li pertoqui.

Totes les entitats que es poden visualitzar tenen la seva plantilla (és a dir `Client`, `Instance` i `License`) on l'entitat del tipus `Instance` contindrà un modal amb una taula buida que servirà per a mostrar l'estat dels nodes de la instància, és a dir, les entitats de tipus `HealthStatus` i `ServiceStatus`.

Cadascuna de les entitats s'anira llistant amb una crida AJAX al mètode que retornarà el JSON amb el contingut de la taula sencera. Aquesta crida es processarà dins dels fitxers `.js` que hi ha definits per a cadascuna de les entitats visualitzades del portal. En tots els apartats es generen botons d'esborrar i modificar per cada línia de la taula (excepte en el de llicència, que tan sols es fa servir el d'esborrar) les crides AJAX per cada botó que es vagi afegint. A la visualització de les instàncies es generaran les barres de l'estat de cada instància i els "tooltips" amb les dades extra, en cas de ser necesari.

Els fitxers que hi ha a la carpeta `assets/css` i `assets/js`, però, no seràn els utilitzats en temps d'execució, caldrà compilar-los com s'ha esmentat prèviament, mitjançant la comanda `yarn encore dev` al contenidor `php-fpm`.

Un altre fitxer important és el de les rutes, que es genera gràcies al plugin de Symfony *FOS JsRouting*, que recull les rutes que estan exposades (amb la configu-

ració `"expose"= true`) i les recull en un JSON que donarem a la classe `Routing` dels fitxers JavaScript. Aquesta classe crearà les URL adients per a cadascuna de les rutes, amb fins i tot els paràmetres que siguin necessaris. Utilitzarem aquest plugin per a qualsevol crida AJAX o URL que calgui generar.

Com a últim detall del frontend, per a poder modificar objectes o eliminar-los, s'ha afegit un botó per a cada tasca a cada fila de la taula. En cas de voler eliminar un objecte amb clicar l'objecte ja es fa la crida AJAX pertinent, però en cas de voler-lo modificar, s'omplen els camps del modal d'edició de l'objecte amb els valors de la fila on s'ha clicat i un altre valor ocult amb l'identificador de l'objecte. D'aquesta manera, quan es faci clic al botó per a realitzar la modificació, es farà la crida AJAX pertinent per a modificar l'entitat amb l'identificador indicat al camp ocult.

7 Resultats

Per a la presentació dels resultats caldrà dividir les presentacions en la part de l'entorn Docker, el portal administratiu i l'API de l'eina.

7.1 Entorn Docker

Per a l'execució del projecte caldrà accedir al directori `/docker` d'aquest i executar l'entorn docker mitjançant la comanda `docker-compose up -build`.

Gràcies a l'ús de Composer i Yarn veurem com amb l'execució d'aquesta comanda ja es realitza la instal·lació de tots els plugins necessaris per al correcte funcionament del projecte.

Un cop fet això, caldrà carregar l'esquema d'entitats del projecte Symfony a la base de dades. Això s'aconsegueix executant `docker-compose run php-fpm bin/console doctrine:schema:create` des del directori `docker`.

Un cop carregat l'esquema, podem executar `./migrate` per assegurar-nos que tot hagi funcionat correctament i visitant l'adreça `http://localhost/` podrem accedir al portal.

L'altre "script" del qual disposem per al funcionament de l'eina és la comanda `./fixtures`, que es pot executar si es desitja generar una base de dades de proves amb clients, instàncies i llicències generats.

7.2 Portal d'administració

Un cop hem executat el projecte, si visitem l'adreça `http://localhost/`, observarem la següent pantalla d'inici de sessió:

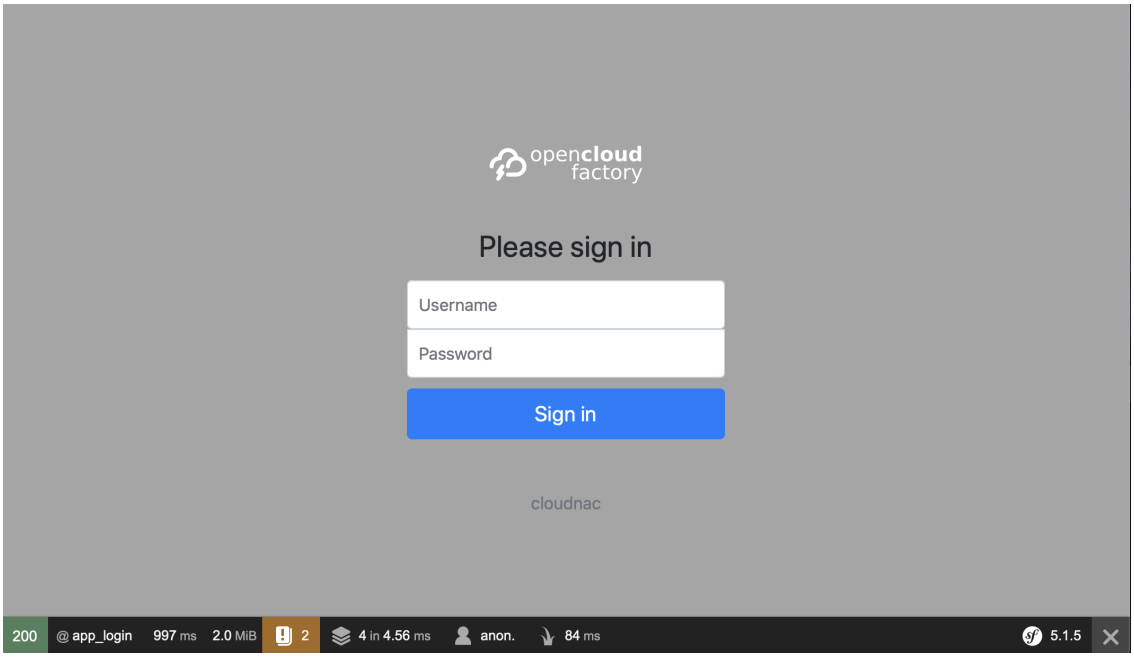

Figura 19: Pantalla d'inici de sessió del portal

La barra inferior és el *profiler* de Symfony. Es deixarà activat per a poder verificar que el portal va realitzant totes les accions desitjades, com ara l'inici de sessió per LDAP, que es pot verificar que s'ha realitzat introduïnt l'usuari `admin` i `opennac` i accedint al portal.

Si cliquem al símbol de l'usuari de la barra i observem la pagina mostrada, veurem com es descriu com s'ha iniciat la sessió de l'usuari:

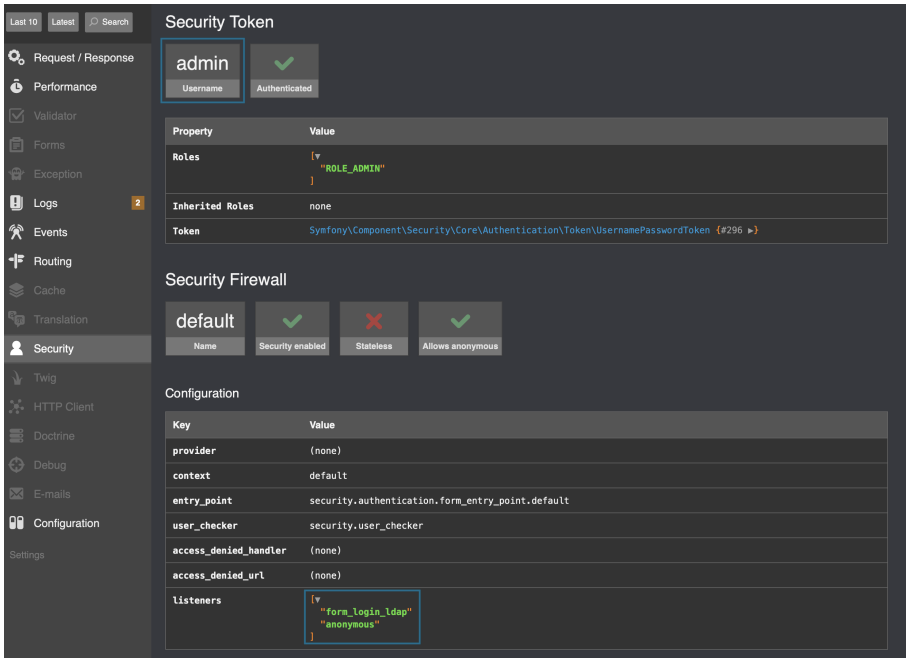

Figura 20: Pantalla del *profiler* per a `/login`

Podem desactivar el profiler canviant la variable `APP_ENV=prod` del fitxer de variables del projecte `.env` de la carpeta `/src/src`. Tornant al portal, si l'inici de sessió és el correcte se'ns mostrarà la pantalla d'inici del portal:

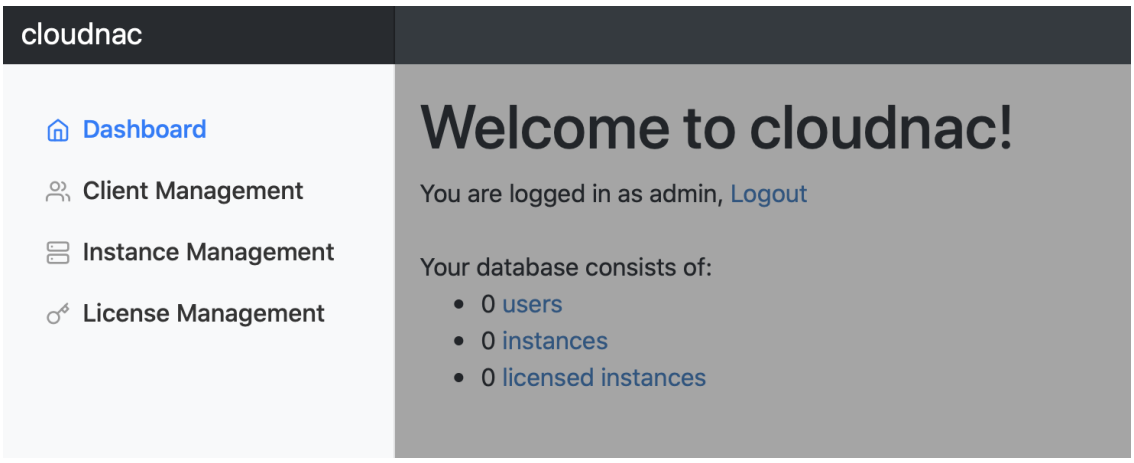

Figura 21: *Homepage* del portal

Clicant a les opcions de l'esquerra podem passar a les diverses seccions del portal, descrites a continuació.

7.2.1 *Client Management*

Un cop s'entri en aquesta secció, podrem observar una taula com la següent, que mostrarà la descripció de cadascun dels clients.

Figura 22: Pantalla de la secció d'administració de clients

Aqui tindrem dos modals, un per a crear un nou client i un altre per a modificar-ne un d'existent, prement el botó groc de la fila de la taula que vulguem modificar.

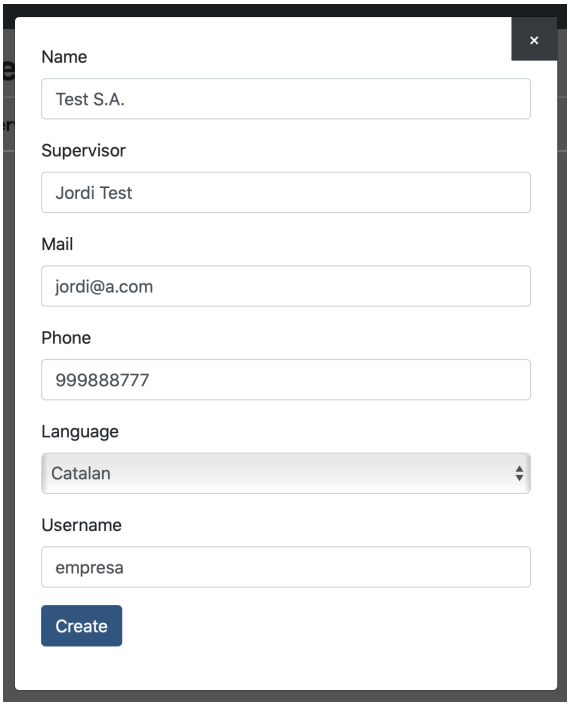

Figura 23: Modal de creació d'un client

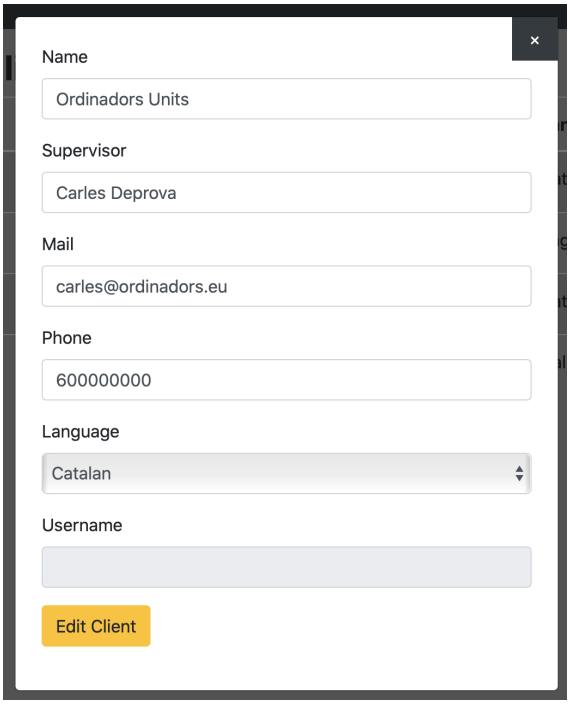

Figura 24: Modal de modificació d'un client

Si esborrem un client des d'aquesta secció, s'esborrarà de la base de dades el client i totes les instàncies i llicències relacionades amb aquest.

7.2.2 *Instance Management*

Aquesta secció s'encarrega de descriure cadascuna de les instàncies registrades al servei al núvol, amb el seu corresponent token. Aquesta és la vista inicial.

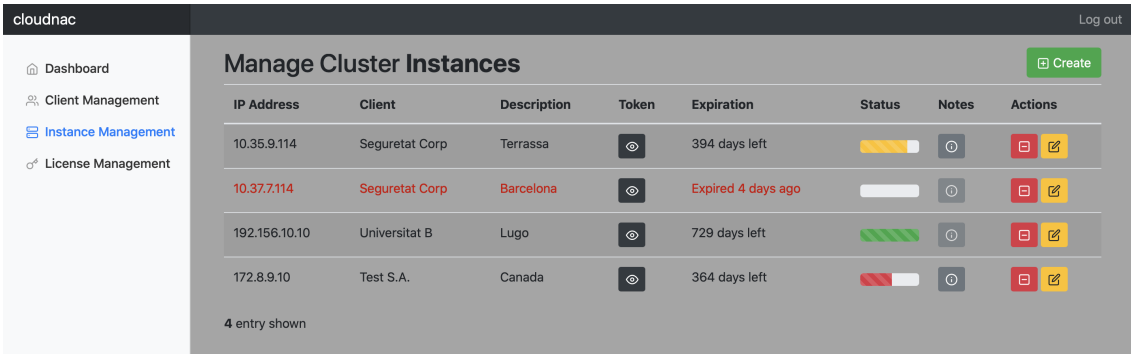

Figura 25: Pantalla d'administració d'instàncies

Tal com podem observar, les instàncies caducades es mostraran en vermell. A la dreta trobem les barres d'estat de cadascuna de les instàncies, que es defineix calculant la mitja entre l'estat de "compliance" de cadascun dels nodes de la instància

en qüestió. Si cliquem la barra, se'ns mostrarà un modal amb més informació sobre l'estat del "healthcheck". Podem observar també que els serveis que no funcionen adeqüadament es mostren en vermell.

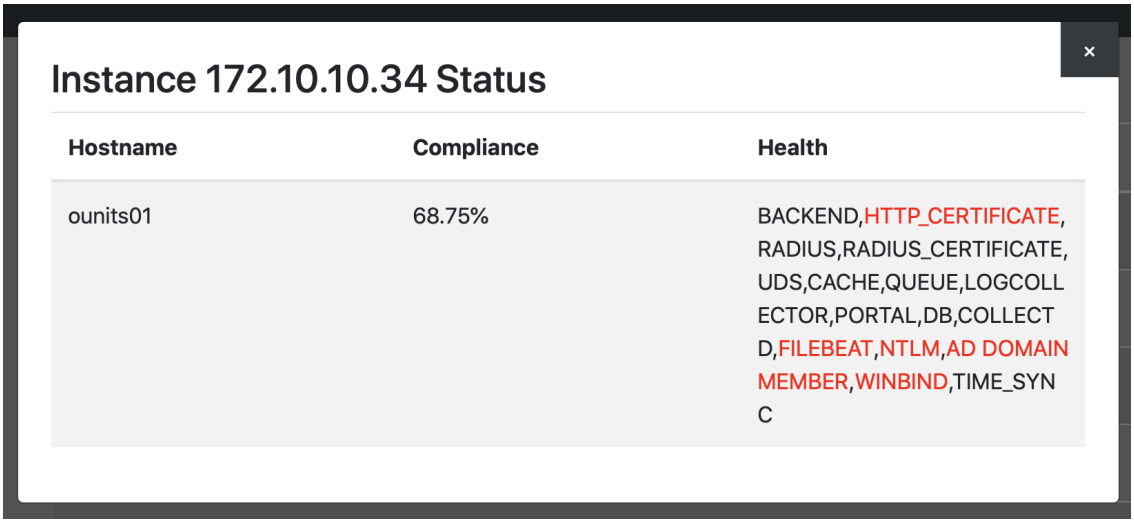

Figura 26: Modal d'estat d'una instància amb un node

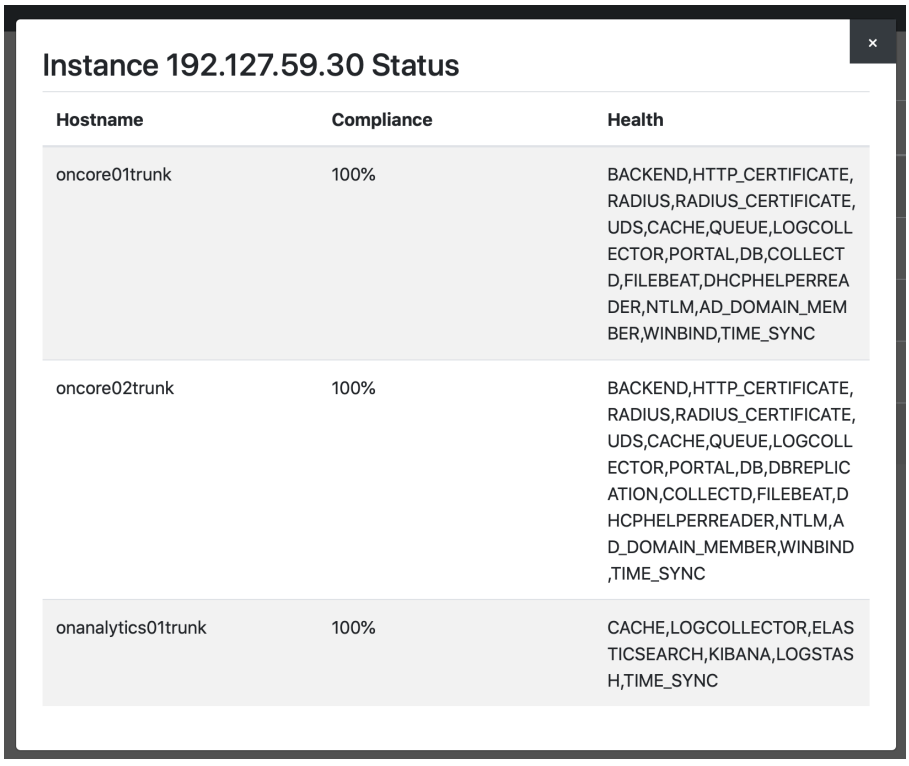

Figura 27: Modal d'estat d'una instància amb múltiples node

També tenim un "tooltip" que mostra informació extra en cas d'haver sigut introduïda. L'altre botó de la taula es el del token, que oculta el valor d'aquest camp fins que s'ha clicat.

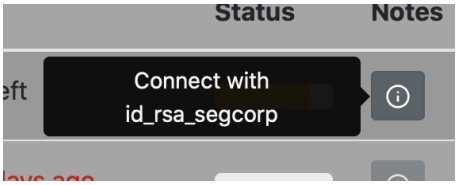

Figura 28: Tooltip amb informació addicional sobre la instància

IP Address	Client	Description	Token	Expiration	Status	Notes	Actions
10.35.9.114	Seguretat Corp	Terrassa	leprbxpcobkg0oosskok0k8k8o8wo0k	394 days left			
10.37.7.114	Seguretat Corp	Barcelona		Expired 4 days ago			

Figura 29: Fila amb el botó de token clicat

Finalment, tindrem els modals de creació i modificació d'instàncies. En el de modificació, se'ns mostra el token i s'avisa a l'usuari que el fet d'actualitzar qualsevol dels camps de la instància engegarà el procés de canvi de tokens, explicat al punt de l'API. Al modal de creació podem observar com es mostra un llistat generat al servidor amb els clients disponibles.

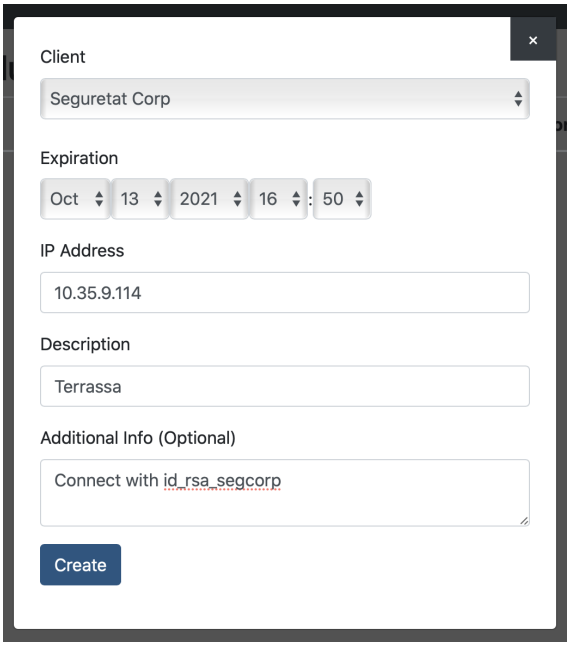

Figura 30: Modal de creació d'una instància

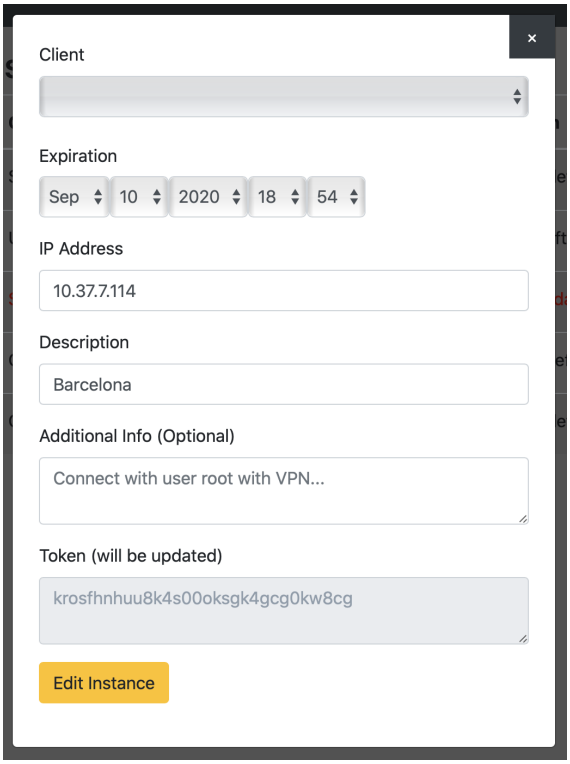

Figura 31: Modal de modificació d'una instància

7.2.3 *License Management*

Per últim, tindrem la secció encarregada de les llicències. Aquesta, a diferència de les anteriors, tan sols oferirà un modal per a la creació d'una llicència i l'habilitat de poder descarregar les que haguem generat. La llicència es descarrega en format `.zip`, però al contenir tan sols el fitxer de llicència, el navegador Safari l'extrau directament, pel que el resultat de la descàrrega és el fitxer de llicència extret.

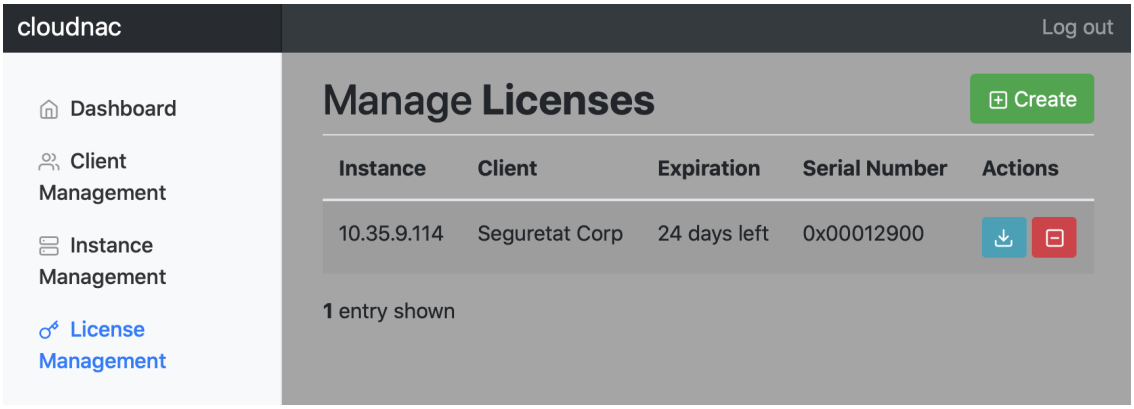

Figura 32: Pantalla d'administració de llicències

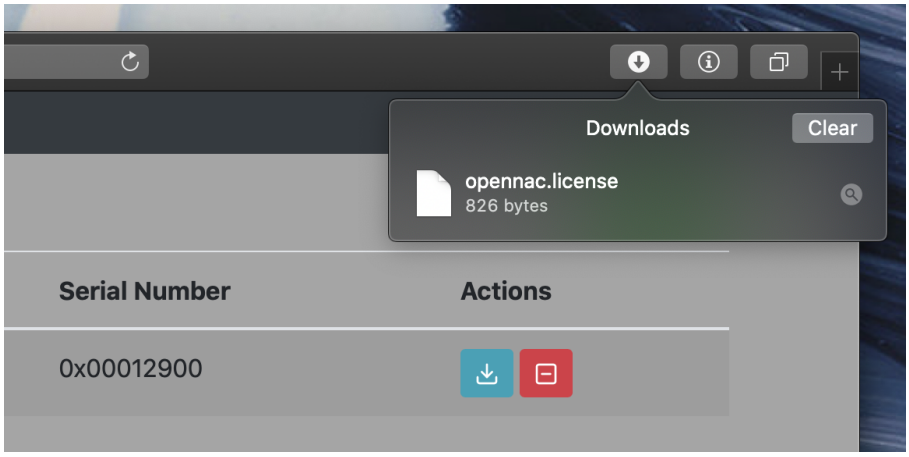

Figura 33: Exemple de escàrrega d'una llicència

Podem observar com el modal de llicències mostra un llistat de les instàncies disponibles que no s'han llicenciat encara.

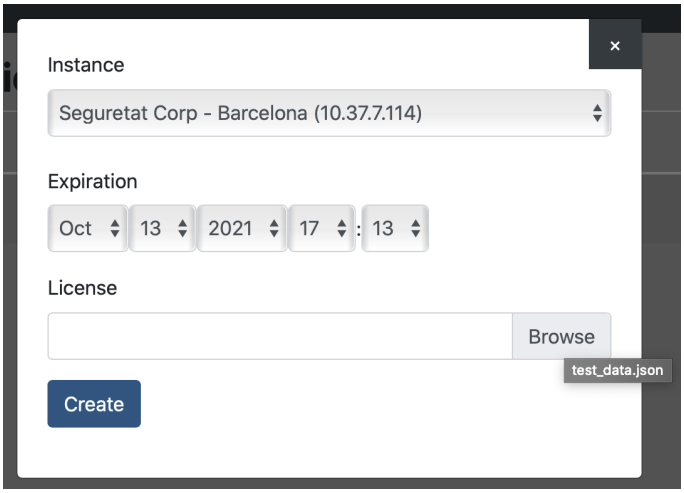

Figura 34: Modal de creació d'una llicència

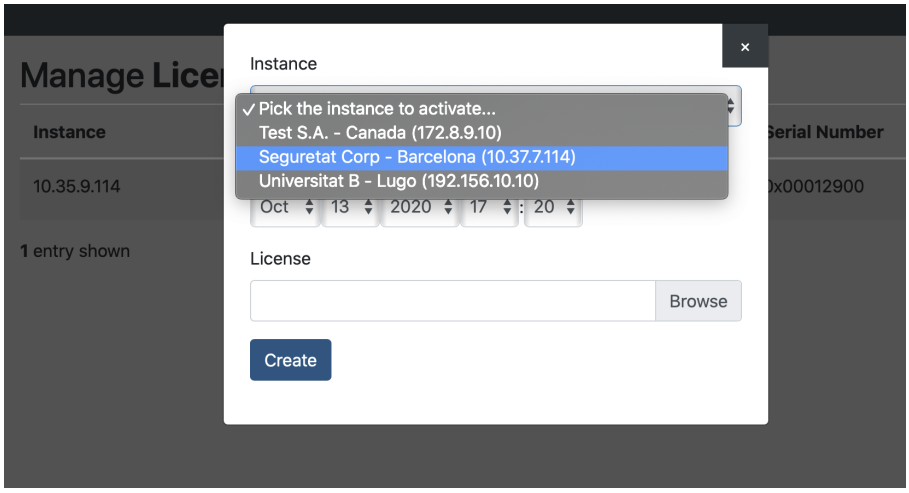

Figura 35: Llistat d'instàncies disponibles del modal

A l'hora de testar aquest apartat, s'ha de tenir en compte que la versió pujada del codi no té el fitxer `.php` encarregat de generar els fitxers de llicència, pel que no serà possible crear noves entitats; s'ha eliminat de l'entrega final a causa de ser un fitxer confidencial de l'empresa.

7.2.4 Servei API

L'API de l'aplicació tan sols permetrà una funció, la de rebre les dades del *healthcheck* d'una instància amb l'accés al servei al núvol configurat. L'adreça d'aquesta petició estarà a `/api/status`, i tan sols admetrà peticions de tipus `POST`. Per a representar la crida des de l'openNAC, farem servir *Postman*, una aplicació per al testeig de crides a APIs [18].

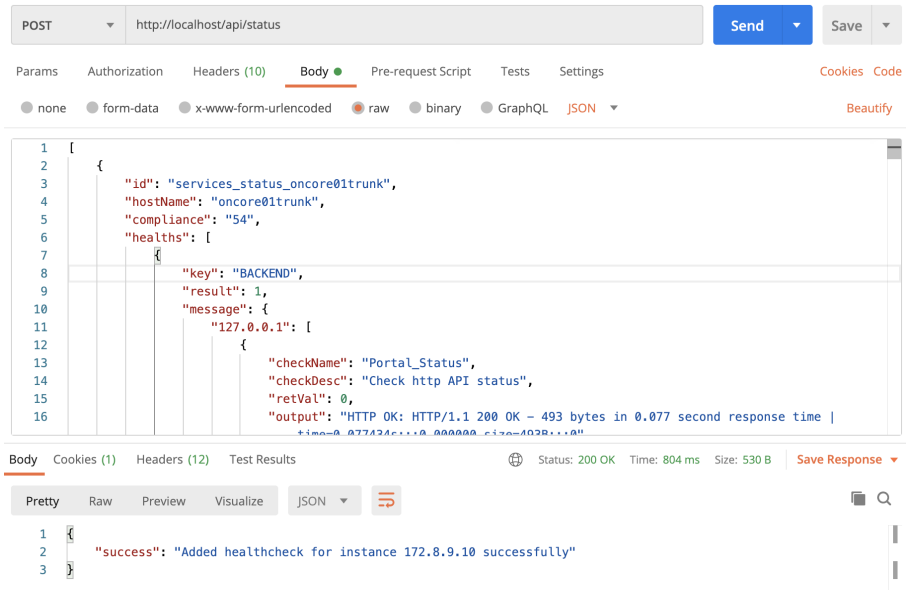

Figura 36: Exemple de petició d'una instància al servei al núvol, amb el format `JSON` adient

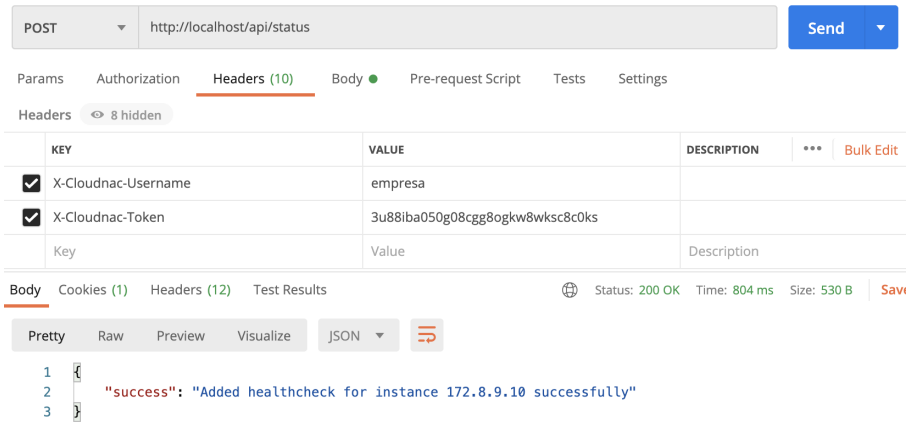

Figura 37: Exemple que mostra el token i usuari utilitzats per a la petició

Si tot funciona correctament, se'ns avisarà de quina instància s'ha vist modificada. En cas d'haver modificat una instància cal tenir en compte que el valor del token es veurà actualitzat, i aquest valor no estarà accessible per la instància. Per a resoldre això, si fem una crida amb el token previ, se'ns alertarà de que s'està fent servir un token obsolet i es permetrà l'accés a l'API per un últim cop amb aquell token. Si provem a fer servir el mateix, no se'ns deixarà utilitzar l'API per al token obsolet, tal com es pot observar a continuació.

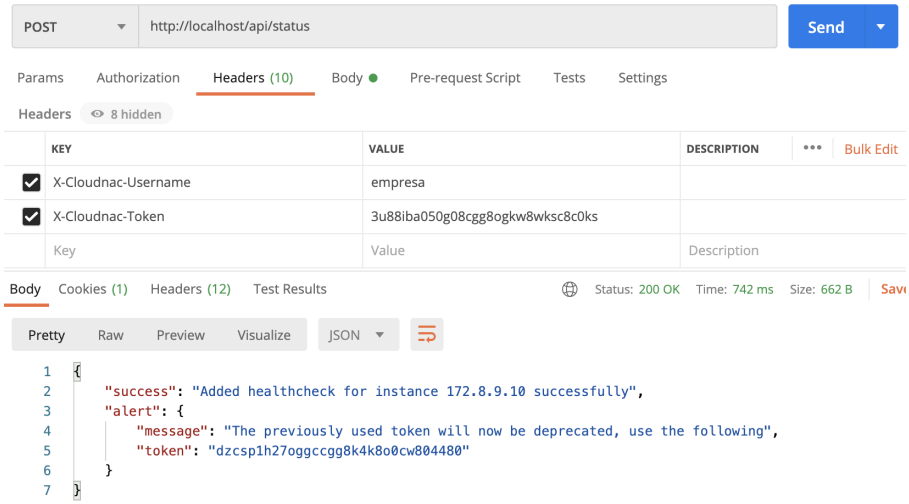

Figura 38: Exemple de petició d'una instància a l'API amb token obsolet

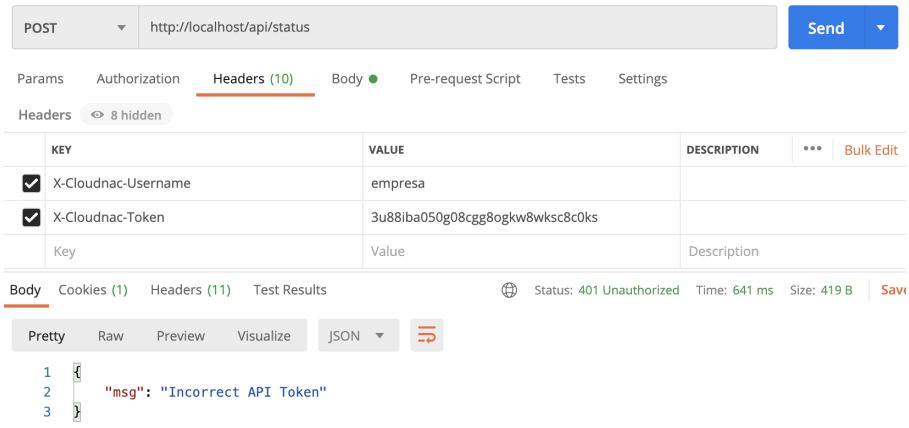

Figura 39: Exemple de petició d'una instància a l'API amb un token obsolet del qual ja se'ns ha avisat

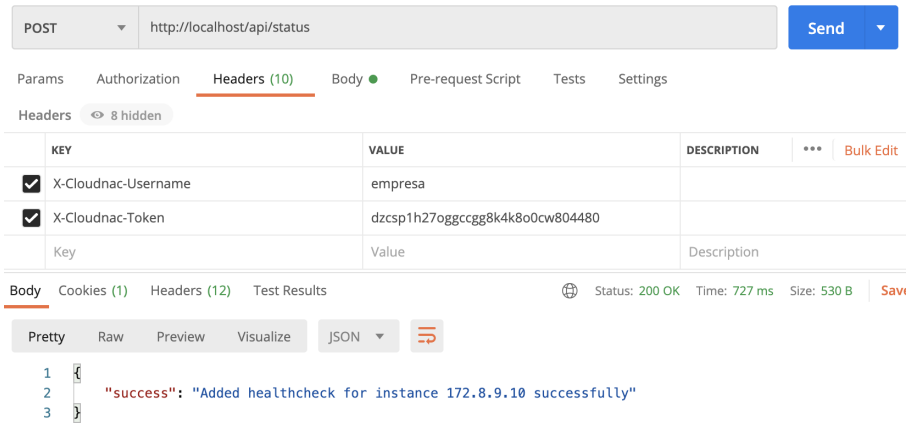

Figura 40: Exemple de petició d'una instància a l'API amb el token renovat

8 Conclusions

En acabar el projecte, ja s'estan començant a preparar tots els mecanismes per a la instal·lació del projecte a la infraestructura de l'empresa. Gràcies a haver sigut dissenyat en Docker, la instal·lació als servidors de l'empresa serà relativament senzilla. Tot allò que només era teoria i explicacions de com anava el projecte a les reunions dels divendres a l'empresa, passarà a fer-se realitat molt aviat.

Pel que fa als objectius, trobo que s'ha pogut realitzar una explicació molt extensa de cadascuna de les fases del projecte i els motius que han guiat les decisions dins de cadascuna d'aquestes.

Un altre objectiu que s'ha satisfet és haver fet servir i adquirir coneixements de moltes tecnologies desconegudes per mi. Docker sempre m'havia semblat una de les tecnologies més complicades que es podia fer servir avui en dia. Després del projecte em segueix semblant una tecnologia molt complicada però molt important de conèixer i saber fer servir. El coneixement que tinc sobre aquesta eina ara em servirà per a futurs projectes; molts d'ells molt probablement al llarg de la meva estada en l'empresa en la qual em trobo avui en dia.

Un altre coneixement molt interessant ha sigut el descobriment del framework PHP Symfony. Mai havia programat en PHP i en poques ocasions he fet servir frameworks per a programar, fora de les assignatures del grau que ho ensenyaven, és clar. Aquesta eina m'ha fet descobrir tot un món i aprendre conceptes molt més avançats sobre la programació web. També m'ha semblat molt interessant utilitzar Doctrine per establir totes les comunicacions amb la base de dades.

Una de les majors complexitats del projecte ha sigut el fet de compaginar-lo amb el treball a l'empresa i la situació excepcional de crisi sanitària en la qual ens trobem avui en dia. Havent de fer classes en línia i teletreballar implicava passar-se el dia davant de la pantalla a la mateixa habitació quasi tot el dia; això implicava una càrrega psicològica molt important. Però trobo que gràcies a això surto més preparat de cara al futur.

Pel que fa a l'objectiu d'implementar el màxim d'eines al servei, n'estic bastant satisfet, ja que s'han pogut implementar aquelles més importants, deixant com a futura tasca aquelles més secundàries que ampliaran l'API, basant-se en el sistema d'autenticació ja definit. També m'ha sabut greu deixar de banda aquelles tasques que requerien modificacions a openNAC, però la situació és la que és i s'han hagut de prioritzar aquelles tasques que eren independents del producte.

També he descobert, tot i no ser el focus prioritari del projecte, molts conceptes sobre el front-end i el seu funcionament. Trobo que cal molta feina en aquest apartat, però el fonament bàsic és prou funcional.

La modularitat també s'ha aconseguit, ja que afegir noves entitats i relacions entre aquestes és molt senzill gràcies a Doctrine. Si es desitja crear una nova ruta pel portal, tan sols cal anar al controlador relacionat amb l'entitat o crear-lo i afegir

la funció que calgui.

Ara tan sols queda mirar; l'entrega d'aquest projecte marca l'inici de la formalització d'aquest a l'estructura de l'empresa i a partir d'ara començaré a treballar a ampliar-lo, junt un altre membre de l'equip tècnic. La principal missió ara serà realitzar les modificacions necessàries a openNAC per a poder sincronitzar-lo amb aquest servei al núvol, acabant d'afegir en aquest una secció per afegir usuari i token.

Referències

[1] David Stuart & Kevin Beaver *Next-Generation IPS for Dummies.* 2013.

[2] OpenCloudFactory *openNAC Enterprise 1.2.0 Documentation.* 2019: doc-opennac.opencloudfactory.com

[3] OpenCloudFactory *Software NAC, visibilidad 100% en la red.* 2020: www.opencloudfactory.com

[4] CCN-CERT *EMMA.* 2020: www.ccn-cert.cni.es/soluciones-seguridad/emma.html

[5] CentOS *CentOS Documentation Home* 2020: docs.centos.org

[6] Symfony SAS: 5.0 Release, `https://symfony.com/releases/5.0`

[7] Symfony SAS: Configuring a Web Server, `https://symfony.com/doc/current/setup/web_server_configuration`

[8] KnpUniversity: Tutorial Track for Symfony 5 | Symfony Casts, `https://symfonycasts.com/tracks/symfony`

[9] ScaleGrid: 2019 Database Trends - SQL vs. NoSQL, Top Databases, Single vs. Multiple Database Use, `https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/`

[10] Educba: MySQL vs NoSQL | Top Comparison to Learn With Infographics, `https://www.educba.com/mysql-vs-nosql/`

[11] TecMint: The Story Behind Acquisition of "MySQL" by Sun Microsystem and the Rise of "MariaDB", `https://www.tecmint.com/the-story-behind-acquisition-of-mysql-and-the-rise-of-mariadb/`

[12] GitHub Repository: kwart/ldap-server: ApacheDS LDAP Playground, `https://github.com/kwart/ldap-server`

[13] Docker Docs: Composer file version 3 reference; Short Syntax, `https://docs.docker.com/compose/compose-file/#short-syntax-3`

[14] Docker Docs: File system sharing; Performance issues, solutions and roadmap, `https://docs.docker.com/docker-for-mac/osxfs/#performance-issues-solutions-and-roadmap`

[15] GitHub Repository: vihnubob/wait-for-it: Pure bash script to test and wait on the availability of a TCP host and port, `https://github.com/vishnubob/wait-for-it`

[16] Symfony SAS: Introduction - Why would you Like to create your Own Framework?,
https://symfony.com/doc/current/create_framework/introduction.html#why-would-you-like-to-create-your-own-framework

[17] Wikipedia, The Free Encyclopedia: Twig (template engine) - Wikipedia,
https://en.wikipedia.org/wiki/Twig_(template_engine)

[18] Postman, INC: Postman | The Collaboration Platform for API Development,
https://www.postman.com