Joan Jara Bosch

MONITORITZACIÓ D'UN SERVIDOR PARAL·LEL DISTRIBUÏT BASSAT EN PLAQUES ODROID

TREBALL DE FI DE GRAU

dirigit per Carles Aliagas Castell

Grau d'Enginyeria Informàtica/Telemàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2020

Agraïments

M'agradaria donar gràcies a tots els professors que he tingut al llarg de la meva vida pel que m'han ensenyat i per formar-me com a persona, en especial m'agradaria donar gràcies a la meva professora de Física de l'institut per tindre tanta paciència amb mi quan era un adolescent i per guiar-me per a triar aquesta carrera. També m'agradaria donar gràcies als meus professors de la carrera perquè alguns d'ells m'han fet decidir que m'agradaria fer en un futur. Moltes gràcies als professors Carles Aliagas, Pere Millán i Carlos Molina per donar-me aquesta oportunitat de fer el treball de fi de grau sobre el seu projecte.

Gràcies a la meva família per educar-me, alimentar-me, fer-me posar les piles i el que ve a ser donar suport durant tota la meva vida. Moltes gràcies als meus amics per aguantar-me i donar-me ànims i fer-me costat durant aquests anys tan durs i durant la pandèmia.

Finalment, donar gràcies a la meva gosseta que tot i ser un animal, es fa estimar i dóna molts ànims en els pitjors moments. En general, gràcies a tothom que ha estat al meu costat per educar-me, fer-me créixer com a persona i per animar-me. Gràcies a tots.

Resum.

Aquest projecte neix amb la intenció d'ajudar en l'àmbit docent així com trobar un sistema de baix cost i d'alta fiabilitat com pot ser un clúster. La idea principal és ajudar a automatitzar aquest clúster perquè pugui ser utilitzat en assignatures de la carrera d'informàtica i alhora, en altres carreres si fa falta. Per això es necessita un sistema capaç d'automatar la instal·lació per al funcionament d'aquest clúster i un sistema fàcil i intuïtiu per a monitorar-lo com és en aquest cas una aplicació web. A més, es triava crear una aplicació web per a així emprar també tasques de publicitat o màrqueting per a donar a conèixer el projecte de manera internacional. Com a metodologia de treball es va començar parlant amb el company sobre les necessitats i configuracions necessàries per al sistema, seguidament es van plantejar models de base de dades per a mantenir la informació d'usuaris i es van fer alguns sketches per a així decidir que tipus de disseny aplicar en les vistes de l'aplicació. A continuació, es va triar un framework i es van analitzar les seves dependències i necessitats per al seu funcionament, com poden ser dos servidors web i un proxy invers. Una vegada conegudes les necessitats, es va començar el procés de creació de l'aplicació web i una vegada acabat, es va afegir la part de monitoratge del sistema de plaques. Per a fer aquest monitoratge, es va crear una pàgina que rep un JSON d'un servidor local creat en Node.js per a l'obtenció de dades. D'aquesta manera es van cobrir les funcions principals i després es van afegir d'addicionals com és la internacionalització de l'idioma. Es van obtenir uns bons rendiments i un consum baix, a més es va comprovar el correcte funcionament del monitoratge amb comandes com el *stress* per a així corroborar la sortida esperada.

Resumen.

Este proyecto nace con la intención de ayudar en el ámbito docente así como encontrar un sistema de bajo coste y de alta fiabilidad como puede ser un clúster. La idea principal es ayudar a automatizar dicho clúster para que pueda ser utilizado en asignaturas de la carrera de informática y a la vez, en otras carreras si hace falta. Por eso se necesita un sistema capaz de automatizar la instalación para el funcionamiento de dicho clúster y un sistema fácil y intuitivo para monitorear como es en este caso una aplicación web. Además, se escogía el crear una aplicación web para así emplear también tareas de publicidad o márquetin para dar a conocer el proyecto de manera internacional. Como metodología de trabajo se empezó por hablar con el compañero sobre las necesidades y configuraciones necesarias para el sistema, seguidamente se plantearon modelos de base de datos para persistir información de usuarios y se hicieron algunos sketches para así decidir que tipo de diseño aplicar en las vistas de la aplicación. A continuación, se escogió un framework y se analizaron sus dependencias y necesidades para su funcionamiento, como pueden ser dos servidores web y un proxy inverso. Una vez conocidas las necesidades, se empezó el proceso de creación de l'aplicación web y una vez terminado, se añadió la parte de monitorización del sistema de placas. Para hacer dicha monitorización, se creó una página que recibe un JSON de un servidor local creado en Node.js para la

obtención de datos. De esta manera se cubrieron las funciones principales y luego se añadieron de adicionales como es la internacionalización del idioma. Se obtuvieron unos buenos rendimientos y un consumo bajo, además se comprobó el correcto funcionamiento del monitoreo con comandos como el *stress* para así corroborar la salida esperada.

Abstract.

This project was born with the intention of helping in the teaching field as well as finding a low cost and highly reliable system such as cluster. The main idea is to help automate this cluster so that it can be used in computer science subjects and at the same time, in other careers if needed. This requires a system capable of automating the installation for the operation of this cluster and an easy and intuitive system to monitor like a web application. In addition, it was chosen to create a web application to also use advertising or marketing tasks to publicize the project internationally. As a working methodology, we first talked to the partner about the needs and configurations needed for the system, then database models were proposed to save the user information and some sketches were made to decide what kind of design to apply in the application views. A framework was then chosen and its dependencies and needs for its operation were analyzed, such as two web servers and a reverse proxy. Once the needs were known, the process of creating the web application began and once it was finished, the monitoring part of the plate system was added. To do this monitoring, a page was created that receives a JSON from a local server created in Node.js for data collection. In this way the main functions were covered and then additional ones were added such as language to make it international. Good yields and low consumption were obtained, in addition the correct operation of the monitoring with commands like the stress was verified to thus corroborate the awaited exit.

Índex

Índex de taules

Índex de figures

1 Introducció

Aquest treball és un treball de fi de grau d'Enginyeria Informàtica de la Universitat Rovira i Virgili on es veuen reflectits els coneixements adquirits durant tota la carrera, més concretament, els coneixements adquirits en assignatures com SOB[1], SCE[2], SD[3], FSO[4]...

El treball es basa en la creació i implementació d'una de les fases d'un projecte més gran. En aquest projecte, es busca millorar el rendiment i disponibilitat que ens ofereix un sol ordinador amb l'ajuda d'un sistema més econòmic i fiable. En el projecte es va escollir la utilització d'un clúster de plaques Odroid les quals són molt econòmiques en comparació a altres plaques similars. Gràcies a aquest sistema proposat en el projecte, s'obtenen resultats molt competents a preus assequibles per la utilització d'aquests en qualsevol àmbit. Perquè el sistema funcioni correctament, es busca l'automatització d'aquest i la possibilitat de monitoratge des d'una pàgina web. És aquí on entra la finalitat del treball de fi de grau, ja que, els nodes o plaques del clúster no disposaran de monitors perquè puguin ser monitorades i per tant, és necessari un sistema que en una xarxa local, permeti el monitoratge i manteniment del clúster. Per dur a terme la finalitat d'aquest, s'ha implementat i creat de manera automàtica una aplicació web feta amb Django, que permet el monitoratge i manteniment d'un sistema paral·lel basat en plaques Odroid. L'aplicació web té diversos usos com pot ser el de donar els medis d'obtenció dels fitxers necessaris per a l'automatització de la configuració de les plaques, des de permetre el mateix monitoratge i manteniment del sistema i fins a aportar un manual o guia per a permetre a l'usuari saber com instal·lar i configurar el sistema.

El monitoratge dóna l'oportunitat a l'usuari de saber l'estat del sistema en tot moment sempre que estigui a la xarxa local. A més a més, amb el manteniment es poden llençar comandes al sistema per a arreglar errors. D'aquesta manera, l'usuari pot controlar el sistema i mantenir-lo sota control sempre.

1.1 Motivació

Després de parlar amb una gran varietat de professors, s'ha escollit aquest treball per tres grans motius.

El primer motiu és el fet de ser un treball de més d'una persona, és a dir, el treball és tan gran que es pot dividir, com ha estat el cas, i per tant guanyes un segon punt de vista del company. A més a més, en treballar en el mateix entorn i tindre necessitats similars, hi ha la possibilitat de donar un cop de mà al company o rebre'l de part seva i per tant fer un treball més sòlid.

El segon motiu és el fet que el projecte intenti resoldre un problema en l'àmbit educatiu com pot ser facilitar la creació d'un entorn de treball en paral·lel per a fer execucions i proves. Es troba interessant perquè dones oportunitats a centres educatius o persones que no tenen molts recursos i d'aquesta manera obtenim satisfacció personal en ajudar a persones.

Finalment, l'últim motiu és el fet de tocar i posar en pràctica els coneixements que s'han obtingut durant el llarg de la carrera sobre assignatures que es consideren molt importants. De la mateixa manera, sorgeix la curiositat de treballar amb dispositius hardware

[1] Sistemes Oberts, assignatura d'últim any d'enginyeria informàtica.

[2] Sistemes de Comerç Electrònic, assignatura d'últim any d'enginyeria informàtica.

[3] Sistemes Distribuits, assignatura de tercer any d'enginyeria informàtica.

[4] Fonaments dels Sistemes Operatius, assignatura de segon any d'enginyeria informàtica.

que no s'han tractat de manera pràctica i això dóna una visió més gran del que és capaç de fer una placa de la mida d'una targeta de crèdit.

Resumin, la motivació ha vingut pels motius esmentats anteriorment i per altres com poden ser l'àmbit del treball, tracte dels professors, confiança que ha sigut transmesa durant aquests anys pels professors darrere aquest projecte, per curiositat i ganes de solucionar un problema.

1.2 Objectius

Per a poder obtenir les competències d'una o moltes assignatures de la carrera d'Enginyeria Informàtica, és necessari disposar i poder accedir a sistemes distribuïts on poder fer paral·lelització. Generalment aquests sistemes tenen un valor elevat i molts centres docents no tenen l'oportunitat d'obtenir-ne un. Per altra banda, els centres docents que disposen dels recursos necessaris, es veuen afectats perquè no disposen de tants nuclis com voldrien.

Per això busquem crear un sistema distribuït, que se sap que no serà el millor però que busca reduir el consum i preu, i a la vegada augmentar la fiabilitat. Així doncs, per un preu per sota de la mitja, es busca cobrir la falta de nuclis per a la paral·lelització en molts centres docents i la creació d'oportunitats en centres que no disposen ni d'un sistema per a treballar la paral·lelització.

1.3 Aplicació web

En aquesta fase del projecte, l'aplicació web per a dur a terme el monitoratge, és una fusió de dos webs. Per una banda tenim els apartats destinats a fer màrqueting o publicitat. És a dir, seccions on s'explica la idea del projecte, qui som, com descarregar el projecte, com fer-lo servir...

I per altra banda, disposa dels apartats encarregats de fer el monitoratge i manteniment de les plaques Odroid. D'aquesta manera, el que vindrien a ser dues aplicacions web, s'ajusten i es fusionen en una sola aplicació web local.

Els apartats complets són la portada, qui som, monitoratge, manteniment, manual i descàrregues. A més a més, té una base de dades on guarda informació sobre les galetes, privacitat, professors, alumnes i usuari encarregat de fer el monitoratge i les tasques d'administrador.

Finalment, hi ha l'opció de canvi d'idioma, on es pot escollir entre el català, el castellà o espanyol i l'anglès.

1.4 Necessitats

Per acabar, s'exposaran totes les necessitats per a la creació de l'aplicació web, l'obtenció de les dades i com realitzar el monitoratge.

Primer de tot, per l'aplicació web, serà necessari escollir un servidor web que pugui servir les peticions HTTP[5], els fitxers estàtics i les configuracions de seguretat. A continuació, serà necessària una base de dades per persistir els usuaris de manera local i en aquest cas, els professors i alumnes implicats en el projecte per tal de fer un codi dinàmic. A més a més, en ser un projecte d'àmbit docent i obert per a tothom, seràn necessaris diversos

[5] Hyper-Text Transfer Protocol, protocol per enviar text amb peticions.

idiomes a l'hora de crear l'aplicació web per a cobrir el màxim d'usuaris possibles. Finalment, és necessari un segon servidor web per a manegar peticions per a l'aplicació web.

Pel que fa a l'obtenció de dades, serà necessari un servidor al node *master* per a poder fer consultes no bloquejants per a l'aplicació web. A més a més, el servidor haurà d'estar sempre obert per poder rebre els missatges i connexions i s'haurà de buscar una manera per tal d'aconseguir aquestes característiques.

Per acabar, perquè l'usuari no tingui complicacions, serà necessari afegir un sistema d'automatització perquè de manera senzilla i intuïtiva l'usuari pugui instal·lar totes les necessitats i el sistema de monitoratge.

Per acabar, a continuació es mostra una imatge per mostrar l'esquema de les necessitats.

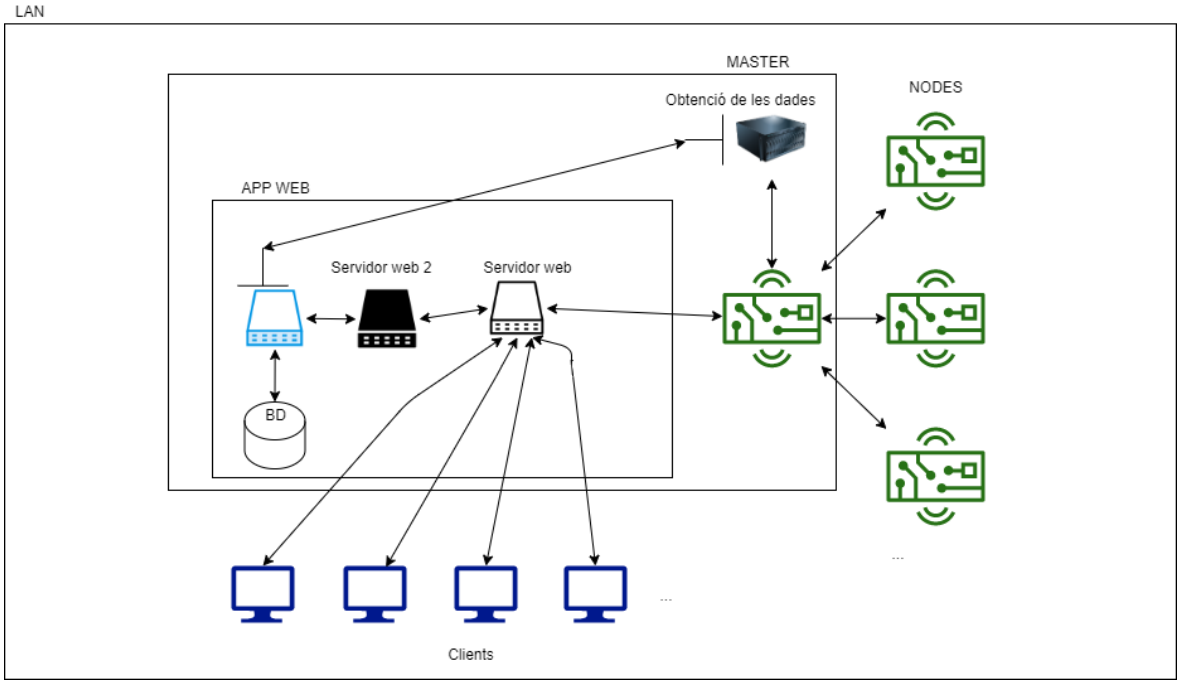

Figura 1. Diagrama de l'aplicació web (1)

2 Eines de treball

Aquesta secció de la memòria serveix per entendre les eines utilitzades per cobrir les necessitats i el perquè de la utilització d'aquestes eines.

2.1 Aplicació web

Aquesta secció serveix per explicar amb més detall les eines necessàries pel desenvolupament de l'aplicació web.

2.1.1 Framework

És necessari d'algun tipus de programari que incorpori reutilització de codi així com estàndards de treball i eines per comunicar-se amb la base de dades. Un *framework* seria ideal, ja que aquesta eina aporta rapidesa en el procés de creació i desenvolupament de l'aplicació així com a eines ja implementades com és el registre d'usuaris.

Per altra banda, al reutilitzar codi i portar-ne d'implementat, evita un dels errors comuns en la programació que és la repetició i redundància de codi. D'aquesta manera treu molta feina de sobre ja que el desenvolupador no tindrà que escriure molt de codi repetit i a la vegada evita errors per aquest i per tant que tingui codi redundant.

Per acabar, normalment els *frameworks,* treballen amb una estructura similar de directoris, la qual el desenvolupador només ha d'introduir el seu codi en llocs específics i per tan permet al desenvolupador el fàcil manteniment i actualització de l'aplicació.

2.1.2 Servidor web

Com s'ha comentat anteriorment a la introducció, per la creació d'aplicacions web és necessari tindre eines que ens permetin la comunicació i manteniment de l'aplicació per tal de poder enviar i rebre peticions per a que l'usuari final pugui treballar i utilitzar l'aplicació.

Un servidor web, ens permetrà manegar les pàgines i fitxers estàtics. D'aquesta manera podrem allotjar l'aplicació a un lloc per poder ser trobada i comunicada.

En ocasions, s'utilitzen diversos servidors web per a repartir la feina entre ells i aprofitar característiques i funcionaments que manquen en l'altre servidor.

2.1.3 Internacionalització de l'idioma

Com s'ha comentat, és necessari arribar al màxim de gent possible perquè el propòsit del treball no és només permetre el monitoratge d'un clúster de plaques Odroid, sinó que també es busca ajudar al màxim de persones i per tant necessitem un *software* o una manera de fer que l'aplicació web tradueixi les vistes per tal de permetre la seva difusió al màxim de gent possible.

2.2 Monitoratge

Aquesta secció serveix per explicar quines eines es creuen necessàries per a dur a terme el monitoratge.

2.2.1 Gràfics

Es necessari de la utilització de gràfic per tal de mostrar l'estat del sistema d'alguna manera. La millor manera per mostrar l'estat serà utilitzant gràfics ja que són intuïtius, fàcils d'interpretar i serà més fàcil que no pas anar mostrant missatges d'error. D'aquesta manera, necessitarem buscar una manera o eina que ens generi gràfics o que els porti incorporats per

tal de tindre a la secció de monitoratge els gràfics necessaris per mostrar la informació essencial pel correcte funcionament del sistema de places Odroid.

2.2.2 Servidor per obtenir les dades

Tal com s'ha vist al subapartat anterior, si necessitem una eina per mostrar gràfics, necessitarem una eina per tal d'aconseguir les dades necessàries per a generar els gràfics i una eina capaç de connectar-se a l'aplicació per enviar les dades.

Per tant, un servidor a banda seria una bona idea i una eina necessària, ja que ens permetria establir una connexió amb l'aplicació web i a la vegada podria dedicar-se a obtenir dades per tal d'enviar-les al detectar una connexió o comunicació.

Per aquets motius s'ha pensat en la utilització d'un servidor, d'aquesta manera també seguim la idea de derivar la feina amb feines unitàries o casi unitàries per permetre el correcte joc de proves i a la vegada la correcta escalabilitat.

2.3 Automatització

Aquesta secció serveix per explicar quines eines seran necessàries per tal de realitzar una automatització.

2.3.1 Scripting

S'ha considerat que és important i imprescindible la utilització de *scripts* per tal d'aconseguir realitzar una automatització, ja que els scripts són molts útils en un gran nombre de situacions. Els scripts permetran el manegament d'execució per al superusuari i per tant controlarem que no pugui modificar el sistema ningú més que l'usuari amb privilegis.

A la vegada, els scripts són molt configurables gràcies a que permeten opcions i paràmetres i per tan podríem fer que un script tingues dos comportaments diferents o més d'un gràcies a les opcions i la capacitat de manegament d'aquestes. Per una altra banda, també ens ofereixen la capacitat d'executar comandes per a realitzar la feina de manera ràpida.

Per aquests motius, es considera necessari l'ús de *scripts* en el projecte per tal d'automatitzar.

2.4 Resum

Resumint tots els apartats anteriors, s'ha arribat a la conclusió que serà necessari utilitzar moltes eines per les diferents tasques que volem dur a terme com pot ser la utilització d'un *framework* o la necessitat de tindre una eina capaç de generar gràfics i actualitzar-los.

3 Requisits

Els requisits són una manera d'acordar com ha de ser un programari. És a dir, gràcies als requisits, s'arriba a un acord de quines característiques, disseny i funcions ha de cobrir el programari. Normalment, s'utilitza en l'Enginyeria del Software i en projectes de creació d'aplicacions siguin web o mòbil.

Els requisits es poden dividir en dos grans grups, els requisits funcionals i els no funcionals que s'expliquen a continuació.

3.1 Requisits no funcionals

Els requisits no funcionals són requisits que com bé indica el nom, no serveixen per indicar les funcions o comportament del programari, sinó que s'utilitzen més aviat per indicar com guardar les dades, acordar el disseny de l'aplicació... Són restriccions que imposa el client pel programa que necessita.En aquest cas, els requisits no funcionals són els següents:

Requeriments no funcionals
Utilitzar la família de la font Optima.
Utilitzar els colors de pàgines web de la URV[6].
Utilitzar diverses carpetes per classificar la informació.
Utilitzar una base de dades local per a guardar la informació.
Implementar la web de manera que funciona en mòbils i navegadors.
Implementar la web en diversos idiomes.
Seguir el patró MVC
Proporcionar un manual per al client.
Implementar multi idioma

Taula 1. Requisits no funcionals

3.2 Requisits funcionals

Els requisits funcionals són tots aquells que estableixen el funcionament de l'aplicació per tal que funcioni correctament. Normalment, s'indiquen els requisits funcionals amb l'ajuda de diagrames de casos d'ús textuals per així indicar les funcions que ha de complir el programari i quin és el comportament esperat de l'aplicació. Un exemple d'un requisit funcional pot ser la creació d'un fitxer i en cas de no tenir permisos mostrar un missatge d'error. Els requisits funcionals del sistema de monitoratge són els següents:

Requeriments funcionals
Persistir els usuaris, professors, alumnes i pàgines legals.
Poder monitorar la utilització de la CPU i la RAM, temperatura, entropia, disc i càrrega del sistema.
Saber la quantitat de temps que el sistema està obert.
Mantenir el sistema.
Enviar comandes als nodes.
Modificar, afegir o eliminar dades.
Iniciar sessió i tancar-la.
Tindre un servidor web per manegar les peticions HTTP.
Tindre un servidor web per comunicar-se amb l'aplicació.
Obtenir dades pel monitoratge.

Taula 2. Requisits funcionals

3.3 Casos d'ús

Un cas d'ús és la descripció d'una acció o activitat on intervenen diferents elements. Normalment els elements que apareixen en els diagrames de casos d'ús són:

Elements	Funció
Actor	Un actor és una entitat ja sigui persona o maquinaria, que utilitza funcions del programari i per tant és relaciona amb els casos d'ús. Hi ha actors ficticis que genera el sistema quan per exemple hi ha una acció automatitzada.
Casos d'ús	Són accions iniciades per altres casos d'ús o un actor i els quals representen funcions del programari. En ocasions hi ha casos d'ús que tenen prerequisits com per exemple haver accionat la funció de inici de sessió per poder accionar la funció veure dades.
Relació	La relació indica quins actors poden o no poden realitzar accions. És a dir, quins treballs poden executar, o no, els actors amb els casos d'ús. Les relacions sempre seran binaries.

Taula 3. Elements diagrames d'ús

3.3.1 Diagrama de casos d'us

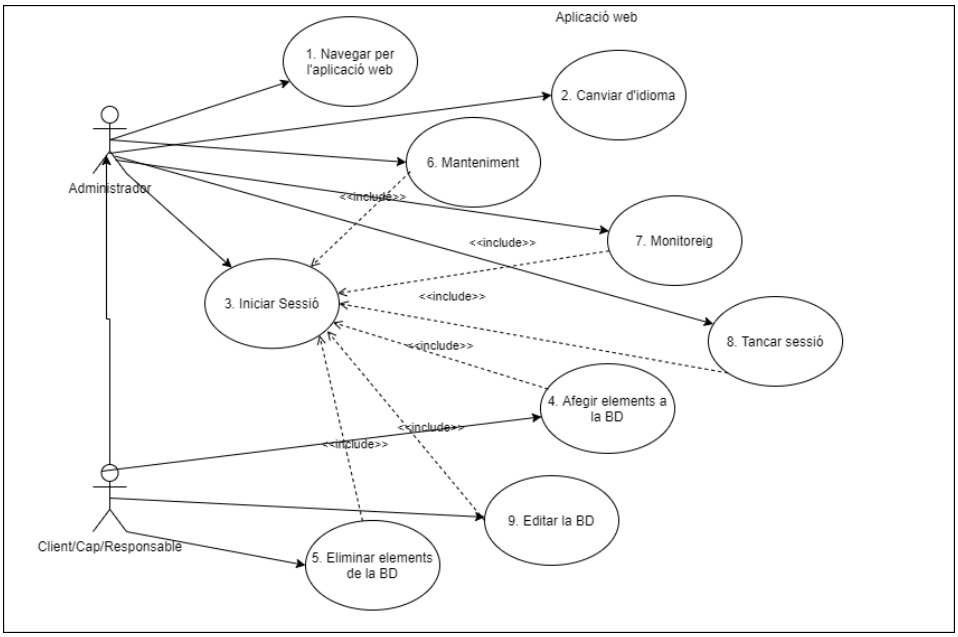

Figura 2. Diagrama de casos d'ús

Com s'observa en el diagrama de casos d'ús anterior, hi ha dos actors on principalment un hereta de l'altre per així indicar que hi ha casos d'ús generals pels dos actors.

El primer actor és l'encarregat a dur a terme el monitoratge i el manteniment, i per tant, necessita poder accedir a la plana web i navegar-hi, canviar d'idioma si s'escau, iniciar sessió, tancar sessió i el més important un cop iniciada la sessió permetre la funcionalitat de monitorar i mantenir el sistema.

Pel que fa al segon actor, es tractaria del cap de l'empresa o centre docent, el qual hauria de poder realitzar les funcions de l'encarregat de monitoratge i a més a més, poder modificar la base de dades per afegir, eliminar o modificar-hi elements. En tenir guardats a la base de dades els usuaris encarregats del monitoratge, aquest segon actor podrà manar sobre l'altre, ja que té la capacitat de crear o eliminar usuaris autoritzats pel monitoratge.

3.3.2 Casos d'ús textuals

Els casos d'ús textual són els requeriments i/o funcions del diagrama d'usos, explicades de manera entenedora i per realitzar l'acord entre el client i el programador.

3.3.2.1 Actor usuari

Cd'ús 01. Navegar per l'aplicació

Resum de la funcionalitat: deixar a l'usuari navegar lliurement per l'aplicació web i obtenir informació sobre qui som i el projecte.

Paràmetres d'entrada: cap.

Paràmetres de sortida: cap.

Actors: cap o client, administrador.

Precondició: cap.

Postcondició: s'ha identificat la petició de l'usuari a la plana web i se li ha mostrat el contingut adequadament.

Procés normal principal:

1. L'actor que està a la plana web, pitja una opció del menú de navegació.
2. L'aplicació web agafar la petició de l'usuari i el redirigeix.
3. L'usuari pot veure el contingut adequadament.

Alternatives de procés i excepcions:

3a. L'usuari no pot veure el contingut degut que ha pitjat monitoratge o manteniment:

3a1. El sistema el redirigeix a la plana d'inici de sessió.

3a2. Si és un usuari autoritzat, accedirà amb les credencials.

3a3. Veu la informació adequadament.

Cd'ús 02. Canviar d'idioma

Resum de la funcionalitat: permetre a l'usuari, canviar l'idioma per el de la seva preferència.

Paràmetres d'entrada: cap.

Paràmetres de sortida: cap.

Actors: cap o client, administrador.

Precondició: cap.

Postcondició: s'ha identificat la petició de l'usuari a la plana web i se li ha mostrat el contingut adequadament amb l'idioma canviat.

Procés normal principal:

1. L'actor que està a la plana web, pitja el menú desplegable d'idiomes.
2. L'actor o usuari, escull l'idioma de preferència.
3. L'usuari pot veure el contingut adequadament amb l'idioma que ha escollit.

Cd'ús 03. Iniciar Sessió

Resum de la funcionalitat: permetre a l'usuari iniciar sessió per a poder realitzar més accions que les generals.

Paràmetres d'entrada: usuari i contrasenya.

Paràmetres de sortida: cap.

Actors: cap o client, administrador.

Precondició: Haver navegat a la pestanya d'inici de sessió.

Postcondició: L'usuari ha iniciat la sessió satisfactòriament.

Procés normal principal:

1. L'actor introdueix l'usuari i la contrasenya.
2. Pitja el botó d'iniciar sessió.
3. L'usuari veu que ha iniciat sessió correctament.

Alternatives de procés i excepcions:

2a. L'usuari ha introduït els paràmetres d'inici incorrectament:

2a1. Se li mostra un missatge d'error al client, administrador o usuari.

2a2. El redirigim al inici de sessió un altre cop.

2a3. Estem al punt 1, un altre cop.

Cd'ús 06. Manteniment

Resum de la funcionalitat: permetre a l'administrador o cap realitzar comandes de manteniment a tots els nodes del sistema a través de l'aplicació web.

Paràmetres d'entrada: cap.

Paràmetres de sortida: cap.

Actors: administrador, cap o client.

Precondició: Haver iniciat sessió i haver navegat a la pestanya de manteniment.

Postcondició: poder realitzar comandes als nodes.

Procés normal principal:

1. L'administrador o cap, tria a qui enviar-li les comandes.
2. L'administrador o cap, prem el botó de connexió al servidor node.
3. Escriu comandes per a ser enviades.
4. L'administrador pot veure el output de les comandes.
5. Tornar al punt 3.

Alternatives de procés i excepcions:

2a. Error perquè el servidor Node.js li ha succeït algun problema:

2a1. Missatge indicant que el servidor està caigut i que necessita ser arreglat per línia de comandes.

2a2. Un cop arreglat el servidor, tornar al punt 2.

5a. L'administrador te massa output innecessari:

5a1. Tria el output del node que no vol.

5a2. Prem el botó d'eliminar output.

5a3. S'actualitzen les terminals.

5a4. Tornar al punt 3.

5b. L'administrador vol desconnectar-se:

5b1. Clicar el botó de tancar.

5b2. I clicar el botó de tancat sessió.

5c. L'administrador vol connectar-se a un altre:

5c1. Clicar el botó de tancar.

5c2. Tornar al punt 1 del procés principal.

Cd'ús 07. Monitoratge

Resum de la funcionalitat: permetre a l'administrador o cap realitzar les accions de monitoratge a tots els nodes del sistema a través de l'aplicació web.

Paràmetres d'entrada: cap.

Paràmetres de sortida: cap.

Actors: administrador, cap o client.

Precondició: Haver iniciat sessió.

Postcondició: poder desconnectar-se i tancar sessió.

Procés normal principal:

1. L'administrador prem el botó tancar sessió..
2. L'administrador o cap, a tancat la sessió i ara es com un usuari normal.

Cd'ús 08. Tancar Sessió

Resum de la funcionalitat: permetre a l'usuari desconnectar-se.

Paràmetres d'entrada: cap.

Paràmetres de sortida: cap.

Actors: cap o client, administrador.

Precondició: Haver navegat a la pestanya d'inici de sessió.

Postcondició: L'usuari ha iniciat la sessió satisfactòriament.

Procés normal principal:

1. L'actor introdueix l'usuari i la contrasenya.
2. Pitja el botó d'iniciar sessió.
3. L'usuari veu que ha iniciat sessió correctament.

Alternatives de procés i excepcions:

2a. L'usuari ha introduït els paràmetres d'inici incorrectament:

2a1. Se li mostra un missatge d'error al client, administrador o usuari.

2a2. El redirigim al inici de sessió un altre cop.

2a3. Estem al punt 1, un altre cop.

3.3.2.2 Actor administrador o cap de l'empresa o centre docent

Cd'ús 04. Afegir elements a la BD

Resum de la funcionalitat: permetre afegir objectes/elements a la base de dades com pot ser afegir un usuari.

Paràmetres d'entrada: els necessaris per a crear l'element.

Paràmetres de sortida: cap.

Actors: cap o client.

Precondició: Haver iniciat sessió i haver navegat a https://ip/admin.

Postcondició: s'ha afegit correctament l'element i es pot veure a la base de dades si fem una crida.

Procés normal principal:

1. L'administrador ha d'escollir que vol afegir i clicar el tipus (usuari, Student, teacher...).
2. L'usuari veurà tots els elements de la taula i un botó per afegir un nou element a la taula.
3. Clicar el botó.
4. Ha d'introduir els paràmetres per a la creació.
5. Prémer el botó per afegir.
6. Missatge indicant l'addició de l'element a la base de dades.

Alternatives de procés i excepcions:

5a. L'usuari no introdueix tots els paràmetres obligatoris:

5a1. Missatge d'error en l'afegiment de l'element, indicant el paràmetre que falta.

5a2. Introduir el paràmetre o paràmetres que falten.

5a3. Tornar a prémer el botó per afegir l'element.

Cd'ús 05. Eliminar elements a la BD

Resum de la funcionalitat: permetre eliminar objectes/elements a la base de dades com pot ser eliminar a un usuari.

Paràmetres d'entrada: cap.

Paràmetres de sortida: cap.

Actors: cap o client.

Precondició: Haver iniciat sessió i haver navegat a https://ip/admin.

Postcondició: s'ha eliminat correctament l'element i ja no apareix a la base de dades si fem una crida.

Procés normal principal:

1. L'administrador ha d'escollir que vol eliminar i clicar el tipus (usuari, Student, teacher...).
2. Seguidament veurà tots els elements per a ser eliminats o modificats.
3. Escollir un element.
4. Clicar el boto d'eliminar.

Cd'ús 09. Modificar la BD

Resum de la funcionalitat: permetre modificar objectes/elements a la base de dades com pot ser modificar el nom d'un usuari.

Paràmetres d'entrada: cap.

Paràmetres de sortida: cap.

Actors: cap o client.

Precondició: Haver iniciat sessió i haver navegat a https://ip/admin.

Postcondició: s'ha afegit correctament l'element i es pot veure a la base de dades si fem una crida.

Procés normal principal:

1. L'administrador ha d'escollir que vol modificar i clicar el tipus (usuari, Student, teacher...).
2. Seguidament veurà tots els elements per a ser eliminats o modificats.
3. Escollir un element.
4. L'usuari escull els paràmetres que vol o ha de modificar.

5. Pitja el botó de guardar.

Alternatives de procés i excepcions:

5a. L'usuari s'ha deixat un paràmetre obligatori en blanc:

5a1. Missatge d'error en la modificació de l'element, indicant el paràmetre que està buit.

5a2. Introduir el paràmetre o paràmetres que falten.

5a3. Tornar a prémer el botó per guardar.

4 Decisions de disseny

Aquesta secció de la memòria està destinada a informar sobre les decisions de disseny de l'aplicació web com poden ser decisions per a la creació de la base de dades, per inserir uns elements o uns altres a les planes web...

4.1 Aplicació web

En aquesta secció s'explica quines eines s'han escollit finalment per a treballar i desenvolupar l'aplicació web.

4.1.1 Django

Django és un *Framework* per a desenvolupar aplicacions web el qual està escrit i creat amb Python i a la vegada és gratuït i obert per a tothom. És a dir, qualsevol persona pot utilitzar-lo. S'ha escollit aquest *Framework* dintre d'un ventall molt gran com podria ser Ruby on Rails, Spring, Vue.js... Perquè es buscava l'aprenentatge d'un *Framework* no utilitzat a la carrera com pot ser Ruby on Rails i a la vegada escrit amb un llenguatge de programació poc usat en la carrera.

A més a més, els avantatges principals d'un *Framework* són el gran nombre de components que porten perquè no s'hagi de reinventar la roda i per tant que processos o problemes que són molt habituals per a un programador web, es puguin fer ràpidament i no des de zero com és el cas de l'acció registrar-se o iniciar sessió.

Finalment, Django té una sèrie de característiques que el fan molt interessant i suculent a l'hora d'escollir-lo i treballar amb ell. Utilitza el patró MVC[6] el qual és molt útil, ja que separa el codi en tres capes diferents i cada capa té les seves funcionalitats. D'aquesta manera fem un software que és més fàcil de reutilitzar, aplicar, mantenir i escalar.

Una altra característica és que en ser codi obert, la comunitat que té al darrere ha creat una gran varietat de mòduls i llibreries les quals donen un gran nombre d'opcions a l'hora de realitzar implementacions i així cobrir les necessitats sorgides. Per altra banda, és molt ràpid de fer servir i és molt escalable, ja que compta amb una gran reutilització de codi gràcies a les seves Apps, les quals es poden reutilitzar en altres projectes o importar-ne de fetes al teu projecte nou.

Per últim, és bastant segur contra atacs informàtics com poden ser el SQL Injection o el CSRF[7]. Així doncs, pels motius comentats en aquest apartat, s'ha utilitzat Django per a la creació de l'aplicació web.

4.1.2 Gunicorn

Gunicorn, o Green Unicorn, és un servidor WSGI[8] que és compatible amb Django. Django, ja té el seu propi servidor WSGI però la utilització d'aquest és per a ús de proves i desenvolupament. Per tant és necessari un servidor compatible amb Django capaç de rebre peticions simultànies i respondre-les. Gunicorn, és un dels servidors més usats per

[6] Model, Vista i Controlador. És un patró del Software, usat per escalar i reutilitzar codi en aplicacions.

[7] Cross-site request forgery, o falsificació de petició, es un exploit que permet la execució de comandes no autoritzades.

[8] Web Server Gateway, és una convenció per a que un servidor pugui reenviar peticions a aplicacions web.

aplicacions web fetes amb Django, com és el cas d'Instagram. S'utilitza perquè és estable, freqüentment utilitzat i fàcil de fer servir. A més a més, seguint la filosofia Unix, se centra a fer poca feina però ben feta i per tant no és necessari preocupar-se de les peticions, de la comunicació amb el servidor web, de tindre múltiples processos per a l'aplicació web i haver de balancejar la càrrega amb aquests processos. Per tant, Gunicorn és el nostre servidor web per a comunicar-se amb l'aplicació web.

4.1.3 Nginx

Nginx és un servidor web molt lleuger i a la vegada també és un Proxy invers per a contingut web. És a dir, el Nginx, ens serveix per a servidor web per a l'aplicació web i a la vegada per servir els fitxers estàtics de la plana web.

Com s'ha dit anteriorment, seguint la filosofia Unix, és millor que cada cosa s'encarregui d'una feina però que la feina que fa, la faci correctament. A més a més, Nginx és molt més lleuger i eficient que la gran majoria de servidors web, per no parlar de la configuració. Per exemple, s'ha afegit a la configuració, l'obtenció de certificats per a establir una connexió segura i encriptada. Per aquests motius, i perquè normalment Nginx i Gunicorn van de la mà[12], s'ha escollit usar Nginx, perquè es busca eficiència i rapidesa sense un gran consum de recursos i aquest servidor web és exactament el que fa. Així doncs, Nginx cobrirà la necessitat de servidor web per a rebre peticions HTTP i servir fitxers estàtics.

4.1.4 Multillenguatge

És una manera d'aplicar internacionalització i localització a aplicacions. Gràcies al I18n de Django, s'ha pogut cobrir la meitat del manegament d'idioma perquè no cobreix el canvi d'idioma d'elements de la base de dades. S'han creat fitxers per als diferents idiomes perquè la pàgina sigui multi idioma. Els idiomes admesos són el català, el castellà i l'anglès a causa dels idiomes coneguts.

4.1.5 Modeltranslation

El modeltranslation[1] [10] és una aplicació pública que permet la traducció de la base de dades. Utilitza el patró decorator, i d'aquesta manera, afegeix columnes addicionals a la base de dades amb el sufix de l'idioma. Així l'usuari és capaç de traduir els objectes de la base de dades i és capaç de manegar el canvi d'idioma no manegat pel I18n.

4.2 Monitoratge

En aquesta secció, es comentarà quines eines s'han utilitzat i quines no per a realitzar el monitoratge.

4.2.1 Gràfics

4.2.1.1 Monitorix

En un principi, s'anava a utilitzar un projecte de codi obert anomenat Monitorix[2], el qual té molt bona pinta i no requereix la utilització de gràfics, ja que el mateix projecte ja els crea en un servidor muntat al port 8080. El projecte Monitorix, té una guia on simplement segueixes els passos i pots instal·lar automàticament el servidor amb els gràfics. Els problemes que es van trobar van ser els següents.

Primer de tot, els objectius i motivacions de l'alumne d'aprendre i posar en marxa els seus coneixements no seria dut a terme i per tant es perdria una mica el sentit de què l'alumne

realitzes la plana web local perquè seguint una guia simple, es pot obtenir el monitoratge del sistema.

Segon problema, per monitorar tots els nodes, s'hauria d'instal·lar el projecte a tots els nodes i per tant s'obtindria un consum major el qual no és desitjat. A més a més, ocuparia molt més tràfic a la xarxa que el desitjat, ja que s'intenta aconseguir la rapidesa més gran, no seria una bona aproximació.

Tercer problema, per a fer la comunicació amb cada node pel monitoratge amb Monitorix, hauríem de saber l'adreça IP[9] de cada node i si per exemple tenim un sistema amb 100 plaques, hauríem de saber les 100 adreces.

Per tant, el Monitorix no és una bona decisió de treball en el cas del sistema creat, tot i això s'ha fet menció a la memòria perquè se sàpiga de l'existència d'aquest projecte i perquè se sàpiga tots els passos i proves que realitzades fins a arribar al resultat final.

4.2.1.2 Chart.js

Chart.js[3], és un projecte de codi obert i gratuït que permet la fàcil creació i configuració de gràfics amb l'ajuda del llenguatge de programació Javascript. No s'ha de confondre amb el Chart del Canvasjs[4], ja que aquest si és pagant i no gratuït. Es menciona aquesta confusió perquè va ser un problema que es va tindre i els gràfics sortien amb marques d'aigua.

Es va optar per utilitzar Chart.js, perquè com es pot observar, tot el nostre projecte té la intenció de ser de codi obert, i per tant s'ha utilitzat tot de manera gratuïta i oberta. Per altra banda, amb els problemes que es van produir amb el Monitorix i l'Ansible, veure punts **4.2.1.1 i 4.3.1** respectivament, es va pensar en com crear els gràfics per a l'aplicació web i pels motius esmentats anteriorment, es va arribar a la conclusió de què és la millor solució per a la creació de gràfics.

4.2.2 Servidor

4.2.2.1 Websocket

Un cop es compren com s'ha realitzat i com està formada l'aplicació web, es necessita una manera d'obtenir la informació. Per obtenir-la, s'ha optat per a la utilització del Javascript amb l'ajuda dels websockets. Els websockets, són sockets web que funcionen de manera asíncrona per tal de no bloquejar el navegador i d'aquesta manera permetre el correcte funcionament de la plana. Tenen uns mètodes que funcionen com a callbacks o events per tal de poder enviar missatges de la pàgina a un servidor i d'aquesta manera no bloquejar la pàgina. Quan rebin el missatge del servidor, s'activarà l'event o callback i es rebrà la informació.

4.2.2.2 Node.js

S'ha usat Node.js per a realitzar la implementació d'un servidor local per l'obtenció de dades necessàries per a crear els gràfics. S'ha optat a la implementació d'aquest servidor addicional perquè des de Django no s'ha trobat manera de fer-ho ràpidament i sense haver de refrescar la pàgina.

[9] IP, Internet Protocol, és un número que identifica una interfície d'un dispositiu.

Aquest servidor ha passat per dues versions. La primera, com ja s'havia comentat, era un simple servidor on, amb l'ajuda d'Ansible, s'instal·lava i mantenia per tots els nodes. Per problemes amb recursos contra el CORS[10], s'ha passat a la segona versió.

La segona versió, només s'instal·la al node central on rep el hostname del node que monitorara i segons el node, s'estableix una connexió segura utilitzant el SSH a una terminal remota del node escollit, per així rebre les dades. Aquí va sorgir un problema amb la connexió SSH, en un principi s'utilitza identificació per claus per així fer una connexió segura i més ràpida, ja que no s'ha d'introduir contrasenya. Però tot i això, es va realitzar una prova i es va observar un retard en l'obtenció de les dades i per tant pèrdua d'històric. En mirar de trobar l'error, es va obtenir un valor d'un segon i mig per cada connexió als nodes. Es va decidir aplicar una configuració al SSH per tal de crear un socket[11] que es mantindrà obert mentre que hi hagi connexions SSH en un interval de dos minuts[9]. Per tant, gràcies a aquest socket, la primera connexió tarda el segon i mig però després en deixar la connexió oberta, les següents connexions del servidor Node.js, als nodes, tarda 170 milisegons.

Finalment, aquest servidor, també utilitza certificats per així fer la connexió de segura entre l'aplicació web i ell mateix. D'aquesta manera es cobreix la necessitat d'obtenció de dades d'una manera segura i ràpida.

4.3 Automatització

4.3.1 Ansible

Ansible[5] és un programari destinat a l'automatització de servidors per tal de mantenir-los i administrar-los. Treballa amb Python i utilitza fitxers del tipus YAML[12] per a descriure les accions, als hosts, als usuaris i moltes més opcions per tal de fer la feina més fàcil per als programadors i així tindre una gran varietat de combinacions per administrar el servidor.

S'anava a utilitzar l'Ansible per tal de fer una instal·lació d'un servidor Node.js, el servidor Node.js va ser la mesura alternativa del Monitorix juntament amb Chart.js i els websockets que s'han vist als apartats **4.2.1.2, 2.2.2.1 i 2.2.2.2**, a cada node per tal d'obtenir les dades per a fer els gràfics i el monitoratge. El problema va venir amb un error del Javascript. Com s'ha comentat anteriorment al punt **4.1.1**, Django ja crea i té algunes mesures de seguretat per evitar atacs informàtics.

Una de les mesures és contra el CORS, i per tant no es podia obtenir la informació dels servidors de cada node. A partir d'aquí es va intentar dues solucions, una és afegir a la petició, l'acceptació del CORS. Va ser descartada a causa de problemes amb la utilització de la CPU[13], ja que augmentava d'un ús normal del 5% a un ús aproximat del 80%, el qual no és desitjable.

[10] Cross-origin resource sharing, mecanisme per permetre l'obtenció de dades o documents d'altres dominis.

[11] Un socket és un programa destinat a connecatar un client i un servidor.

[12] YAML, format de serialització de dades legible per a l'èsser humà.

[13] Central Processing Unit, és la unitat central de processament o cervell d'un ordinador.

La segona solució és enviar la IP al servidor del node central i així, a l'estar al mateix domini s'evita el problema. S'enviala IP o hostname[14] del node a l'enviament al servidor per així comparar i saber si s'ha de fer SSH[15] o no per obtenir les dades.

4.3.2 Pm2

Pm2 és un dels projectes de Node.js que s'ha descarregat i instal·lat amb el NPM[16] i el qual té la feina d'executar mantenir els servidors creats en Node.js. De tal manera que simularà el comportament d'un daemon[17] de Linux i si algun servidor deix de funcionar, el reiniciarà perquè torni a respondre. Es podria dir que funciona com un Watchdog[18]. Així doncs, gràcies al pm2, el servidor node és capaç de mantenir-se sempre obert.

4.3.3 Scripts

S'utilitza el scripting[19] de Bash[20], per a la creació i execució del sistema de monitoratge. S'han creat un total de set scripts per a dur a terme l'automatització del sistema amb generació de seguretat, instal·lació de paquets necessaris, possibilitat d'eliminació del sistema de monitoratge, instal·lació de dependències per a l'usuari Odroid i scripts per a l'obtenció de les dades que fan possible el monitoratge.

4.4 Base de dades

Per entendre i explicar com s'ha creat i quines taules té la base de dades, s'ha creat un diagrama de classes on cada classe és una taula de la base de dades.

[14] Hostname, nom que se li dona a una màquina per a poder saber de manera més fàcil qui és sense tindre que saber la IP.

[15] SSH, Secure Shell, nom d'un protocol i manera d'accedir remotament a una terminal d'un servidor o ordinador.

[16] NPM, node package manager, és la eina dedicada a descarregar i tractar els projectes fets per la comunitat de Node.js

[17] Programa que s'executa en segon pla sempre que es compleixin les condicions d'inici.

[18] Mecanisme de seguretat per mantenir els servidors sempre encesos.

[19] Creació de scripts, que és un fitxer d'ordres per a ser executat per automatitzar i integrar la comunicació amb altres llenguatges de programació.

[20] Bourne-again Shell, és un llenguatge d'ordres i Shell d' Unix.

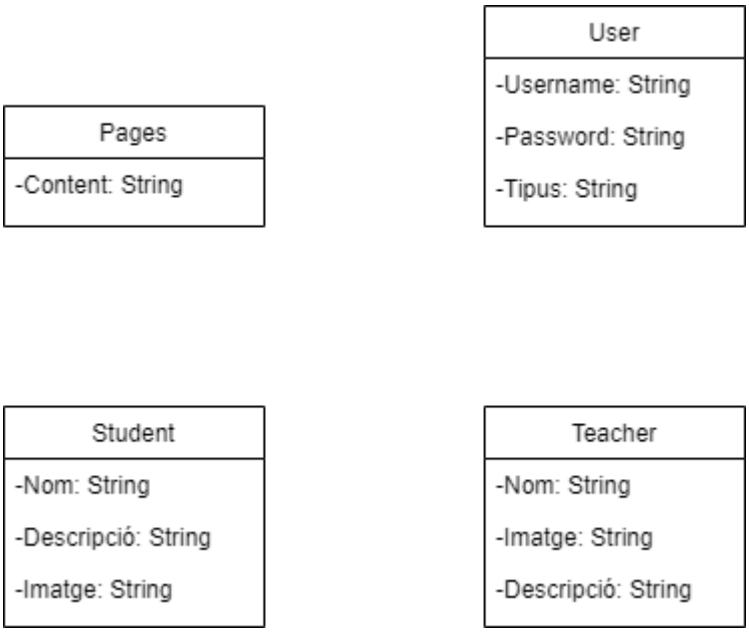

Figura 3. Diagrama de classes

Com es pot observar, el contingut de la base de dades consta de quatre taules diferents. Tot i que inicialment, només existia una taula. S'ha pensat que no és necessari guardar informació de les plaques com podria ser el hostname, la IP o la direcció MAC[21] gràcies al fet que amb la configuració i automatització realitzada per l'estudiant David Ferrer, ens permet accedir a un fitxer amb tota aquesta informació i per tant és redundant.

Per altra banda, en un principi, com ja havíem exposat, només existia una taula, la taula User o Usuari. El motiu va ser el següent, només es volia guardar informació de l'usuari, ja que no teníem cap altre element per a persistir. A més a més, d'aquesta manera es podia manegar qui accedia o qui no a les planes de monitoratge i manteniment. Addicionalment, la taula disposava d'un mètode, el mètode per registrar-se, però va acabar sent suprimit perquè feia insegura l'aplicació web pel fet que qualsevol persona tenia la possibilitat de registrar-se i per tant d'iniciar sessió i veure el monitoratge del sistema. Pels motius esmentats anteriorment, es va prendre la decisió d'eliminar el registre de l'usuari.

Finalment, es va decidir afegir les taules teacher, student i pages per tal de generar HTML[22] dinàmic. És a dir, inicialment amb el primer disseny de la base de dades, les planes de polítiques de privacitat i la plana sobre nosaltres, partien de codi estàtic i per tant si volíem modificar les polítiques de privacitat o si el projecte es modificava i s'afegia personal, el HTML havia de modificar-se. D'aquesta manera, gràcies a Django, hi ha la possibilitat d'utilitzar tags[7] que ens permeten consultar variables, fer bucles, afegir blocs de contingut, canviar d'idioma... D'aquesta manera, és possible l'obtenció dels elements de la base de dades com una variable, i fer un bucle per a tindre el codi HTML dinàmic i per tant fer l'aplicació web molt més escalable i amb menys codi, per tant, més senzilla.

21 Media access controller, és un identificador únic assignat al controlador de la targeta de xarxa d'un dispositiu.

22 HyperText Markup Language, s'utilitza per al desenvolupament de pàgines web.

4.5 Aplicació web

Pel que fa a l'aplicació web, es van fer tres dissenys i es va escollir un d'ells. A continuació podeu veure'ls.

4.5.1 Primer prototip

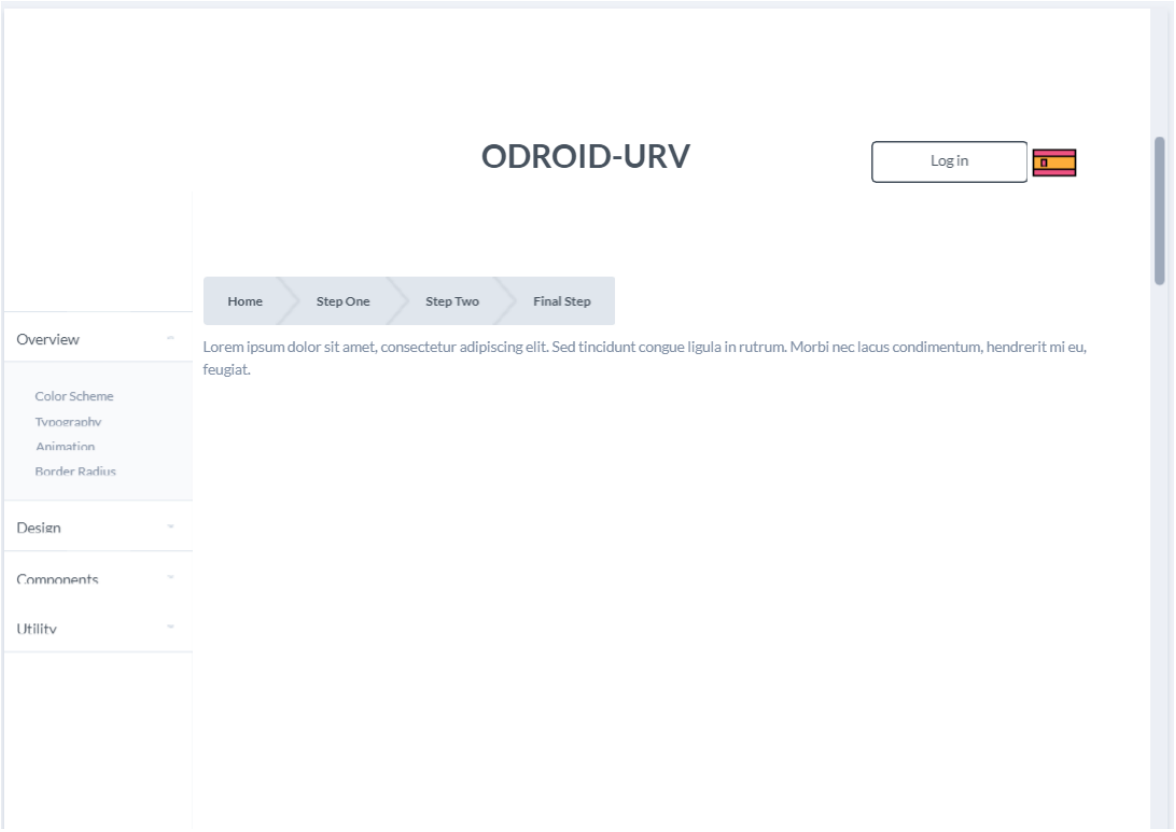

Figura 4. Prototip 1

Com es pot observar el primer prototip, disposava d'un menú vertical per navegar per l'aplicació web. A continuació, es va optar per a posar un "camí de motlles de pa", que serveix per saber en quina part de l'aplicació es troba l'usuari. Finalment, disposava d'un botó per escollir l'idioma on l'idioma venia representat per una bandera i un botó d'inici de sessió.

D'aquest primer prototip es va descartar el "camí de motlles de pa" a causa de la falta de profunditat de l'aplicació web. És a dir, no existia una profunditat d'apartats dintre de les pàgines i per tant no feia falta saber d'on venies, ja que tots o quasi tots els nivells de profunditat són de nivell 1. Per altra banda, es va descartar el menú vertical perquè es va trobar més atractiu la barra de navegació per les pàgines. Així doncs, pel segon disseny es va mantenir el botó i la bandera per l'idioma.

4.5.2 Segon prototip

Figura 5. Prototip 2

Com es pot apreciar, aquest segon disseny manté el botó de la bandera del país per canviar d'idioma, i s'ha modificat el menú vertical per una barra de navegació horitzontal. Per altra banda, com ja es tenia decidit en més profunditat el model general de l'aplicació web, es va afegir el model per escollir la presentació dels gràfics on primerament va ser un carrusel, element web per canviar d'objecte en clicar a dreta o esquerra de l'objecte.

D'aquest segon prototip es varen descartar el botó de la bandera, ja que segons W3C[23] no és recomanable posar imatges de banderes perquè un país no representa l'idioma com que molts països tenen més d'un idioma oficial[8]. Per aquest motiu en el següent prototip es va canviar. Finalment, es va eliminar el carrusel perquè la gràcia dels gràfics és poder veure'ls tots a la vegada per obtenir la informació i no pas haver de canviar cada cop que es vol veure un gràfic.

[23] World Wide Web Consortium, s'encarrega de generar i estandarditzar processos i feines per a la creació de webs.

4.5.3 Tercer prototip

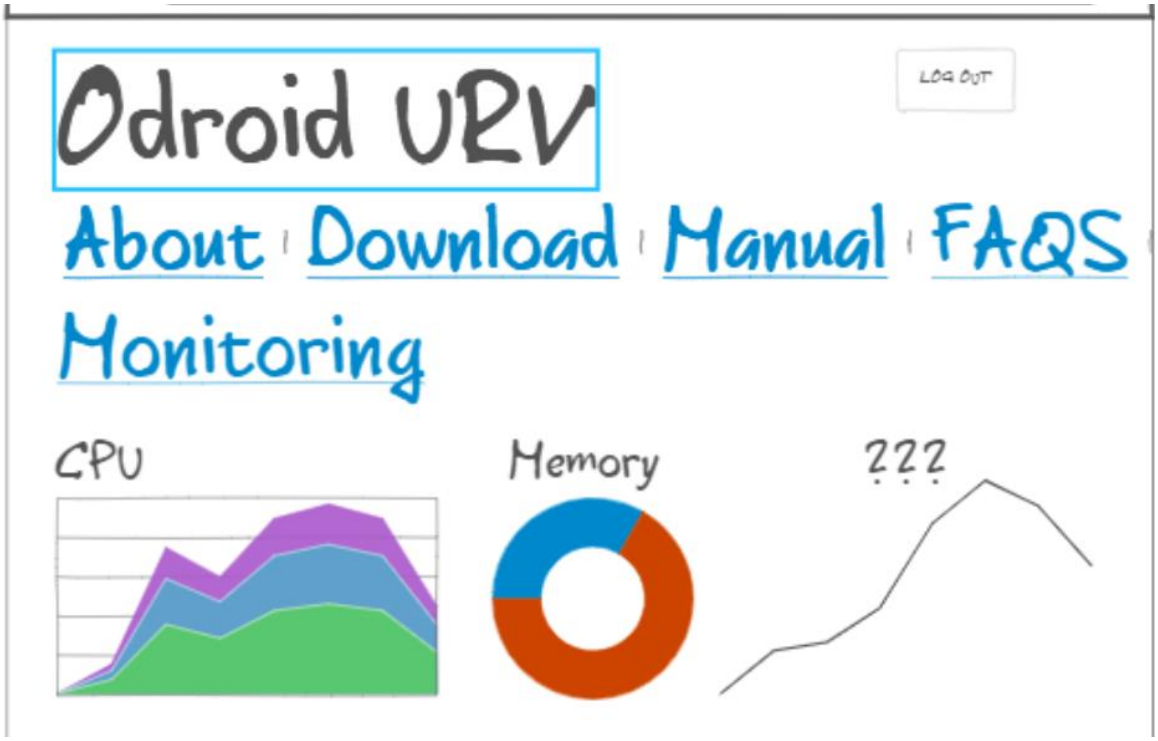

Figura 6. Prototip 3

En aquest tercer prototip es va utilitzar una eina diferent de la utilitzada anteriorment per a fer el disseny. En aquest model ja s'ha passat a l'observació total dels gràfics i gràcies als canvis dels altres dos models, del tercer prototip s'ha arribat a l'actual aplicació web.

4.6 Resum

Resumint tots els apartats anteriors, s'han descartat solucions com el Monitorix i l'Ansible per problemes que han succeït. Per altra banda, gràcies als scripts, s'ha creat una manera d'automatitzar la creació i automatització dels servidors Node.js, Nginx i Gunicorn juntament amb la instal·lació del Django. Un cop tot instal·lat, es pot accedir a l'aplicació web on el servidor Nginx rebrà la petició, i respondrà a les necessitats que ell mateix pugui solucionar com és el cas de fitxers estàtics.

En cas contrari, desperta el servidor Gunicorn el qual és l'encarregat de servir les peticions i aquest executa codi de Django. Django amb l'ajuda del patró MVC, refresca la pàgina o executa codi i respon a Gunicorn.

Finalment, quan es vol monitorar, es creen els gràfics amb Chart.js i s'obre un websocket que estableix una connexió segura amb el servidor Node.js. D'aquesta manera, l'aplicació web rep la resposta del servidor en format JSON[24] i aquesta la interpreta per a actualitzar els gràfics del Chart.js. A continuació es pot veure una il·lustració de la figura 1 completada amb les eines utilitzades i escollides.

[24] Creació de scripts, que és un fitxer d'ordres per a ser executat per automatitzar i integrar la comunicació amb altres llenguatges de programació.

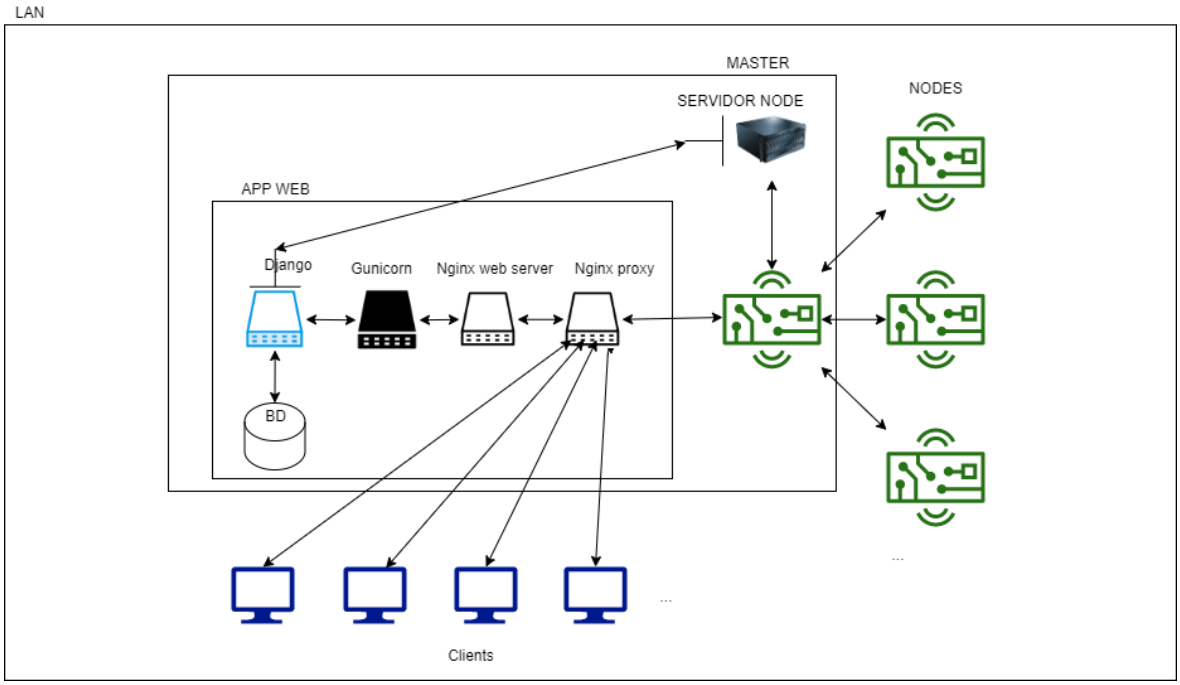

Figura 7. Diagrama de l'aplicació web (2)

5 Implementació

Aquesta secció de la memòria està destinada a entendre millor com s'ha implementat el sistema de monitoratge així com estendre el coneixement del funcionament d'aquest. La secció està dividida en quatre parts. La primera serveix per entendre el procés d'automatització del monitoratge. La segona part, mostra el servidor Node.js i els scripts que aquest executa per obtenir les dades. La tercera part parlarem sobre la implementació de l'aplicació web, així com els fitxers pel Nginx i el Gunicorn. Finalment la última part, té la intenció d'arribar a profunditzar en la comunicació.

5.1 Automatització

L'automatització del muntatge de l'aplicació web s'ha dut a terme amb el script general *start-monitoring.sh* el qual crida a més scripts per tal d'encapsular el codi i fer-lo més escalable i poder reutilitzar-lo.

5.1.1 Gen-cer.sh

Aquest script rep per paràmetre el domini o IP que té la placa central, on està muntat tot el sistema de monitoratge. D'aquesta manera genera uns certificats i claus per a fer la comunicació segura i encriptada entre servidor Node.js i aplicació web. A més a més, el script serà executat per l'usuari amb privilegis o per un usuari *sudoer*[25].

Els paràmetres addicionals per a la generació de les claus i el certificat, s'agafen per defecte perquè aquest script és executat per altres que tenen la intenció d'automatitzar i que l'usuari només hagi de fer un clic. Els paràmetres són els següents:

- Country = ES
- State = Tarragona
- Locality = Reus
- Organization = URV
- Organizationunit = URV
- Email = joan.jara@estudiants.urv.cat

S'han posat aquests valors per defecte, ja que el projecte és de la Universitat Rovira i Virgili. A més a més, s'ha posat de localitat i email els de l'estudiant Joan Jara perquè ell va ser el creador del script i així no implicava a gent no vinculada en el projecte.

Per entendre millor la generació dels certificats, es mostraran les comandes utilitzades, en aquest script. Primer de tot s'ha utilitzat la comanda **openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/nginx/ssl/nginx.key -subj "/C=$country/ST=$state/L=$locality/O=$organization/OU=$organizationalunit/CN=$commonname/emailAddress=$email" -out /etc/nginx/ssl/nginx.crt** on s'utilitza openssl una eina que serveix per a crear i manegar claus i certificats per a generar seguretat. A més a més, s'utilitza com a eina criptogràfica. En aquest cas, s'utilitza per generar una clau i un certificat, amb el paràmetre re -x509, es genera una petició d'auto signatura que dura el nombre de dies passat a l'opció -days. Per tant el certificat durarà un any. A continuació, es demana una nova clau on la guardarem al fitxer *nginx.key* i que no serà encriptada per l'opció

[25] Sudoer, és un usuari que pertany a aquest grup i per tant pot utiltzar la eina sudo per executar com usuari amb privilegis.

-nodes ja que serà d'ús local. Per a la creació del certificat, es passaran totes les dades mostrades anteriorment amb l'opció -subj i així l'usuari no s'ha d'esperar a omplir totes les dades perquè serà generat per defecte. La sortida del certificat es guarda a *nginx.crt*.

Un cop es té el certificat i la clau, es necessari el DHParam[26], per tal de poder enviar informació encriptada.

Per altra banda, aquests certificats només serveixen de manera local i són auto signats, per tant, no tenen una validació i el primer cop que s'intenta entrar a l'aplicació web, es mostrarà un missatge d'advertència.

Finalment, en ser un projecte destinat a la docència i a l'ús públic, l'usuari pot modificar el script i posar els valors que vulgui. Un cop modificats els valors simplement ha de fer una execució del script passant per paràmetre la IP o domini vinculat a la IP.

5.1.2 Pm2.sh

Aquest script té el propòsit d'instal·lar els mòduls necessaris per al funcionament del servidor Node, així com l'automatització i execució del servidor.js per a l'usuari Odroid. Un cop estan instal·lats els mòduls, crea un fitxer de configuració pel SSH de l'usuari per tal d'agilitzar i incrementar el temps de comunicació entre l'aplicació web i el servidor. La creació del fitxer de configuració es realitza amb un *echo* redirigit cap al fitxer **~/.ssh/config** on el fitxer té de dades *els Host, el ControlMaster, ControlPath i el ControlPersist.*

- *Host **,* ens indica que sigui la configuració per defecte, ja que admet tots els Host.
- *ControlMaster auto,* per tant es managerà automàticament.
- *ControlPath **~/.ssh/sockets/%r@%h-%p***, serveix per indicar on es guardaran els sockets per a reutilitzar la connexió i que així la connexió es faci més ràpida. La %r ens indica el nom d'usuari remot per fer l'inici de sessió, la %p ens indica el port i la %h ens indica el hostname del servidor. D'aquesta manera cada socket serà únic i no trencarem connexions d'altres usuaris.
- *ControlPersist 2000,* ens indica el temps que el socket estarà viu des de l'última connexió establerta. D'aquesta manera, es controla que el socket s'acabi destruint.

La instal·lació dels paquets es realitza amb la comanda *npm install -g* on l'opció -g significa de manera global i per tant anteriorment, s'han modificat els permisos i el propietari dels directoris globals del node, */usr/local/bin i /usr/local/lib* [19][20].

5.1.3 Start-monitoring.sh

Finalment, el script principal per a l'automatització del sistema de monitoratge i manteniment. Aquest script és executat per usuaris *sudoers* o amb privilegis. Comença l'execució amb una actualització i instal·lació dels components necessaris. A continuació, parteix de dos bucles, un pel *apt-get install i l'altre pel pip3 install*. El propòsit dels bucles és fer menys codi i reutilitzar un condicional per comprovar si el paquet està instal·lat o no. Si està instal·lat, no entra dintre el *if* i per tant no s'intenta reinstal·lar. El codi seria com el següent:

[26] Diffie-Hellman Parameters, són paràmetres per a la realització d'intercanvi d'informacio entre servidor i client utilitzant TLS o SSL.

```
Variable=”packet1 packet2 packet3 ...”
Per paquet a $variable fer
       Si !paquet_instalat llavors
             Apt-get install $paquet / pip3 install $paquet
       Fsi;
mentre;
```

La comprovació de la instal·lació de *l'apt-get install* es realitza amb **$(dpkg-query -W -f='${Status}' $package 2>/dev/null | grep -c "ok installed") -eq 0**, on la comanda *dpkg-query* és una eina per fer quèries de la base de dades dels paquets, amb l'opció -W, es llistaran tots els paquets de la base de dades que coincideixin amb el patró. Pel que fa a l'opció -f, s'utilitza per indicar el format de la sortida i per tant en passar el nom del paquet que es vol saber si està o no instal·lat, això s'obté de sortida el paquet i el seu estatus. Passarem la sortida a la comanda *grep* que amb l'opció -c, contarà el nombre de vegades que surt un cert patró. En aquest cas conta si surt el patró *ok installed*. Per tant si està instal·lat, sortirà un valor d'1 o superior i en cas contrari, el valor serà 0. Per aquest motiu s'utilitza el comparador numèric -eq per la condició del *if*. En el cas del pip3, es realitza un procés semblant, tot i que es modifica el *dpkg-query* pel *pip3 list –format=columns*, d'aquesta manera es llisten tots els paquets instal·lats pel *pip3* i amb una redirecció cap al *grep*, se sabrà si està instal·lat o no.

A continuació, es creen directoris per a copiar els fitxers clonats del git. S'usa la comanda *mkdir -p <directori o directoris>*. Gràcies a l'opció -p, es pot crear el directori dintre d'un directori en cas de no existir i així ens estalviem fer comprovacions. Seguidament, es copien els arxius clonats als directoris corresponents i els directoris creats. S'utilitza comanda *cp -p* per mantenir els permisos i copiar-los. A continuació, com que el script *start-monitoring.sh* és executat per l'usuari amb privilegis, s'ha de canviar el propietari perquè els directoris i els fitxers han de ser accessibles per l'usuari Odroid. A la vegada, tots aquests fitxers i directoris s'ubiquen al directori home perquè els nodes el comparteixen i així tots tindran accés.

Es crea un soft link[27], amb la comanda *ln -s*, del següent arxiu del dnsmasq **/etc/dnsmasq.d/dnsmasq_hosts.conf**. Aquest arxiu conté informació dels nodes necessària per al monitoratge com és la IP, la MAC i el hostname. En fer un soft link, quan es connecti una nova placa i es modifiqui el fitxer, l'aplicació web que llegeix el soft link, veurà els canvis.

Ja per acabar, s'executa el script *gen-cer.sh* i el script *pm2.sh*. El script *pm2.sh*, com que és necessari per a l'usuari Odroid, s'executa de la següent manera, *su odroid -c "./pm2.sh"*. Així s'aconsegueix executar com a usuari Odroid. Per acabar, es fa un reinici del dimoni **SSHD**, del dimoni **Nginx** i del dimoni **Gunicorn**.

5.1.4 Stop-monitoring.sh

Aquest script és la contrapart del *start-monitoring.sh*. La seva finalitat és la possibilitat d'eliminar tota la instal·lació realitzada. Així es proporciona una eina a l'usuari, per a poder eliminar el sistema de monitoratge en cas de no necessitar-lo més. El script simplement utilitza les comandes *rmkdir, rm, apt-get remove, pip3 uninstall i npm uninstall* per eliminar els directoris, els fitxers i per a desinstal·lar els paquets respectivament.

[27] El soft link és un enllaç simbòlic que simbolitza l’accés directe de Windows

5.2 Servidor node

En aquesta segona part, s'explica com funciona i quines són les funcions principals del *servidor.js*, que és el servidor amb Node.js d'on s'obtenen les dades necessàries per a la creació i actualització dels gràfics.

5.2.1 Monitoring.sh i json-server.sh

Els scripts *monitoring.sh* i *json-server.sh* serveixen per a l'obtenció de dades per a la creació dels gràfics. El servidor Node.js, té un *websocket*, el qual està escoltant al port 3000. Un cop es connecta al *socket*, es comença la comunicació amb l'aplicació web i demana les dades per fer el monitoratge gràcies al script *monitoring.sh*. Aquest script simplement serveix per a no duplicar codi i s'encarrega de determinar si el script *json-server.sh* s'executa en local o per SSH a un altre node. Tot seguit, es pot apreciar el codi dels scripts en pseudocodi per tal d'entendre millor les opcions i l'obtenció de dades.

5.2.1.1 Codi script monitoirng.sh

```
monitoring(whoami)
begin
     Si whoami = master llavors
       Json-server();
     Sinó
       Ssh -o StrickHostKeyChecking=no whoami json-server()
     Fsi;
end;
```

Com es pot observar i com s'ha comentat, aquest script serveix per determinar si el script *json-server.sh* s'executa en local o no. En compartir el directori principal per tots els nodes, el script estarà per tots els nodes al mateix lloc. A més a més, s'afegeix l'opció per a no mirar la contrasenya del node, ja que en el seu moment, es van crear claus per fer la connexió més segura i còmoda.

5.2.1.2 Codi json-server.sh

```
Json-server()
Begin
       dades = Obtenir_dades();

       dades = tractar_dades(dades);

       retorn dades;
end;
```

Com es pot veure, s'agafen totes les dades per a fer els gràfics. S'han utilitzat comandes com ps, free -m i moltes més per obtenir dades però també s'han llegit molts fitxers per obtenir moltes altres dades com és el cas de l'entropia, el temps...

Ara que més o menys se sap com s'obtenen les dades pels gràfics, es passarà a un nivell més i s'explicarà dada a dada les opcions i fitxers que s'han tocat.

Per la cpu s'ha utilitzat la comanda **ps -A -o pcpu | tail -n+2 | paste -sd+ | bc**. La comanda ps, fa una còpia instantània dels processos del sistema. Amb l'opció -A, s'indica que sigui de tots els processos i amb l'opció -o indiquem nosaltres l'output de la comanda on

indiquem que sigui el percentatge de CPU utilitzada amb pcpu. D'aquesta manera obtenim una columna instantània de la utilització de la CPU. A continuació, s'envia la sortida de la comanda a la comanda tail que ajuda a eliminar les primeres dues línies per a poder fer la suma de la utilització. Arribant ja al final de la comanda, s'envia l'output a la comanda paste que transformarà la columna amb una fila on entre cada element de la columna passa a ser element més el símbol de suma i aquest ouput s'envia a la comanda bc que ens ajuda a fer operacions matemàtiques i per tant sumarà tots els % d'utilització de CPU.

Per saber la RAM[28], s'ha executat dins el script la comanda **free -m | awk '{print $7}' | head -2 | tail -1**. La comanda free, ensenya la utilització de la memòria, en escollir l'opció -m, la sortida de la comanda passa a ser en Megabytes. La sortida l'enviem a la comanda awk on escollirem la columna número set per a ser mostrada. La columna número set és la columna referent a memòria RAM disponible. Finalment, amb l'ajuda de les comandes tail i head, s'obté el valor en Megabytes de la memòria RAM, ja que amb la comanda head s'elimina les files no desitjades i amb la comanda tail, s'elimina la fila inicial també no desitjada.

Seguidament per obtenir l'espai ocupat i disponible del disc, s'utilitza la comanda **df -t ext4 --output=used,avail | head -2 | tail -1**. La comanda df, reporta la utilització de l'espai a disc, en escollir l'opció -t ext4, limitem la sortida al sistema de fitxers tipus ext4. Les plaques Odroid per defecte utilitzen el ext4 i per aquest motiu s'ha escollit la sortida ext4. Finalment, limitem la sortida per memòria disponible i usada amb l'opció –output=used,avail. Com l'obtenció de dades anteriors, s'utilitza la comanda head i tail per a descartar files no desitjades i així obtenir un sol resultat.

Per la recaptació d'informació sobre la temperatura, s'utilitza la comanda **sensors | tail -2 | head -1 | cut -d " " -f 9-**. La comanda sensors, mostra tota la informació obtinguda pels sensors de la placa. Així doncs, amb aquesta informació, i com amb les altres operacions, s'utilitza el tail i el head per descartar informació no desitjada. Un cop es té la fila que es vol, es redirigeix la sortida cap a la comanda cut, la qual serveix per retallar informació segons un separador passat per l'opció -d i els paràmetres que es volen obtenir amb l'opció -f. Així és com se sap la temperatura crítica i actual de la placa. A continuació, s'aplica una substitució de caràcters no desitjats amb la línia següent: **echo "${temp//[+,),(,=,a-z,C,°]/}"**. Ara ja si tenim la temperatura crítica i actual llestes per a ser enviades a l'aplicació web.

En el cas de la càrrega del sistema[18], es llegeix el fitxer **/proc/loadavg** que conté la mitja d'espera dels processos per un minut, cinc minuts i quinze minuts. Així doncs, llegirem el fitxer amb un cat i la sortida la passarem a la comanda awk per tal de retornar els tres paràmetres desitjats. Finalment, queda una línia com la següent, **cat /proc/loadavg | awk '{print $1" "$2" "$3}'**.

Tot i això, encara no sabem si l'estat de la comanda anterior és crític o no perquè és necessari saber el nombre total de cores per saber si el sistema té una càrrega excessiva o no. Per tant, executarem la comanda **lscpu | head -3 | tail -1 | cut -d " " -f 15-**. Aquesta serveix per a obtenir informació sobre l'arquitectura de l'ordinador o sistema. Un cop tenim aquesta informació, amb el head i el tail, s'elimina la informació no usada i finalment, amb el cut, s'agafa el paràmetre 15 fins al final.

A continuació, per obtenir l'entropia disponible al sistema, llegirem el fitxer **/proc/sys/kernel/random/entropy_avail** [17] amb la comanda cat, i així de simple és obtenir l'entropia del sistema. L'entropia del sistema, ens indica el nombre de valors aleatoris

[28] RAM, memòria d'accés aleatori. És memòria volàtil i que s'utilitza en l'ús dels ordinadors.

capaços de generar en un instant. Si l'entropia és molt baixa, es té un problema de seguretat perquè no pots generar claus segures i no es pot encriptar degudament.

Ja acabant, per obtenir de manera fàcil el valor de temps per saber des de quan està obert el sistema, s'ha utilitzat tres cops el mateix procediment per obtenir dies, hores i minuts. Simplement s'ha llegit el fitxer **/proc/uptime**[16], i amb la comanda awk, s'ha obtingut el valor amb segons. Seguidament, amb la comanda echo, s'ha replicat el valor en segons i s'ha afegit operacions matemàtiques com són divisions i mòduls, com s'ha fet amb la comanda per obtenir el valor de la CPU, per tal d'obtenir el valor numèric en dies, minuts i segons. I finalment amb la comanda bc, s'han realitzat els càlculs matemàtics.

Pel que fa a la memòria escrita o llegida, s'hi ha empleat la comanda iostat, la qual donà informació d'entrada i sortida pels dispositius del sistema. Seguidament, s'ha emprat la comanda tail per eliminar files d'informació no desitjades i finalment, amb l'ajuda de la comanda awk, s'han sumat els valors de la columna cinc i sis les quals donen informació sobre la lectura i l'escriptura a memòria. Al ser acumulatiu, a l'hora d'enviar el primer cop es controla perquè no hi hagi errors en la realització del gràfic.

Per últim, per aconseguir les dades pel gràfic de xarxa, s'ha anat al directori **/sys/class/net/** el qual té de subdirectoris les interfícies disponibles a la placa. En una línia es realitza un bucle per mirar cada subdirectori i agafar el nom de la interfície de xarxa[14]. A continuació dintre el mateix bucle, es llegeix el contingut dels fitxers tx_bytes i rx_bytes del subdirectori i així obtenir els bytes acumulats d'enviament (tx_bytes o transite bytes)[15] i els bytes acumulats rebuts (rx_bytes o bytes recieved)[15]. Finalment, com el cas anterior, al ser acumulat es té en compte a l'hora de realitzar el gràfic i en una sola línia obtindríem tota la informació de xarxa. **for i in /sys/class/net/*; do RX=$(cat $i/statistics/tx_bytes); TX=$(cat $i/statistics/rx_bytes); network=$(echo "_"$(basename $i)": "$RX" "$TX" "$network) ; done**

Per acabar, és realitzar un *echo* per retornar el resultat de totes les comandes anteriors en format JSON[25].

5.2.2 Maintenance.sh

Aquest script, a diferència dels scripts de monitoratge, no és necessari d'un segon script per a comprovar quin node tractem, ja que s'envia com a paràmetre juntament amb la llista de cadenes de caràcters per a muntar la comanda a executar. Un cop la comanda està muntada, es comprova a quin node es vol executar per a fer un SSH o no.

5.2.3 Servidor.js

El servidor.js, és un script realitzat en el llenguatge de programació Javascript i el qual s'utilitza per obtenir les dades pel monitoratge o enviar comandes als nodes. El servidor crea un websocket que llegeix els certificats generats en l'automatització del sistema. Un cop té els certificats, escoltarà al port 3000. El websocket té un event per detectar si hi ha una connexió i la mateixa connexió té un event per detectar el missatge o missatges que pot rebre de l'aplicació web. Dintre l'event de la connexió, mira si el missatge que rep és de monitoratge, manteniment o indica primerament de qui vol les dades perquè a continuació el servidor li pugui demanar als scripts.

El servidor per a fer les seves feines, genera dos fills o processos que s'encarreguen d'executar el script de *monitoring.sh* o el de *maintenance.sh* i no bloquejar-se per així seguir rebent peticions i connexions.

5.3 Aplicació web

5.3.1 Nginx

Pel que fa al servidor web i Proxy invers *Nginx,* s'ha creat un fitxer anomenat **odroid_site_ssl**, el qual conté tota la configuració per l'aplicació web. En una ocasió, van sorgir errors que van ser solucionats gràcies a les següents referències[21][22][23]. Aquest fitxer s'ha descarregat juntament amb tots els fitxers de l'aplicació web i un cop executat el script *start-monitoring.sh* es copia l'arxiu al directori **/etc/nginx/sites-available/** el qual és el directori que utilitza *Nginx* per a saber els llocs disponibles. Però tot i això, encara no ens funcionaria el lloc web, ja que també és necessari copiar l'arxiu al directori **/etc/nginx/sites-enabled/**. En aquest cas, s'ha generat un *soft link* i un cop generat el enllaç, se reinicia el dimoni *Nginx*.

Pel que fa al fitxer **odroid_site_ssl**, conte un seguit de línies i opcions per a configurar l'aplicació, primer es mostraran les opcions de connexió segura i a continuació les d'obtenció de fitxers statics.

- *Listen 443 ssl http2,* d'aquesta manera s'indica al servidor quins ports i protocols escoltar i utilitzar. S'ha escollit el ssl i el http2 perquè són protocols segurs que encripten la comunicació.
- *Server_name _,* aquesta opció serveix per a indicar el nom del servidor i per tant com accedir pel navegador. Al passar el valor _, el servidor agafa la IP de manera dinàmica i així en cas de que canvies la IP de la placa *master,* la pàgina no cauria.
- *client_body_buffer_size 1k,* amb aquesta línia és capa la mida del buffer del cos per tal d'evitar atacs.
- *client_header_buffer_size 1k*, de la mateixa manera, també es capa el buffer del header.
- *client_max_body_size 100M*, capem la mida màxima del cos del client.
- *large_client_header_buffers 2 1k*, capem els buffers dels clients a 1K per evitar atacs de buffer overflow.
- *add_header X-Frame-Options "SAMEORIGIN"*, d'aquesta manera només acceptarem dades que provinguin del mateix origen. En aquest cas, el servidor Node.js, està en el mateix domini.
- *add_header Strict-Transport-Security "max-age=31536000; includeSubdomains; preload"*, indiquem al header que volem transport segur.
- *add_header X-XSS-Protection "1; mode=block"*, modifiquem el header per tal d'afegir seguretat contra *cross-site scripting* i que ens injectin codi a l'aplicació web.
- *add_header Content-Security-Policy "default-src 'self' http: https: data: blob: 'unsafe-inline' ws:; connect-src ws:" always,* s'indica la política de seguretat del contingut i s'afegeix a les opcions per connectar-se amb websockets.
- *ssl_prefer_server_ciphers on,* indiquem que volem encriptació.
- *ssl_certificate /etc/nginx/ssl/nginx.crt,* indiquem on trobar el certificat per a fer la connexió segura.
- *ssl_certificate_key /etc/nginx/ssl/nginx.key,* s'indica on trobar la clau per la connexió segura.

- *ssl_dhparam /etc/nginx/ssl/dhparam.pem,* s'indica on trobar el DHparam per a establir una connexió segura.
- *location /static/ { root /home/odroid/.django-monitor/odroid;},* així *Nginx* sap on trobar els fitxers estàtics .

Gràcies a aquesta configuració, estem segurs i el servidor sap on es troben els fitxers estàtics per a poder-los servir.

5.3.2 Gunicorn

Pel que fa al servidor web *Gunicorn*, s'ha creat un fitxer per a crear un dimoni per aquest servidor. Un cop clonat el projecte de monitoratge, i a l'executar el script d'automatització *start-monitoring.sh,* es copiarà el fitxer ***gunicorn.service*** al directori **/etc/systemd/system/** el qual conté els dimonis que seran executats. Un cop copiat, és necessari realitzar una recàrrega dels dimonis amb la comanda **systemctl daemon-reload**. A continuació, s'ha d'activar amb la comanda **systemctl enable gunicorn**. Ara ja tenim el nostre dimoni per al servidor web. Per al correcte funcionament d'aquest, s'indica que s'executi i reinici sempre que després de detectar connexió a internet per així mantenir-lo sempre encés i evitar caigudes del servidor.

5.3.3 Websocket

Pel que fa a l'aplicació web, per les seccions monitoratge i manteniment, disposa de la creació d'un websocket per tal de connectar-se al servidor Node.js. En el cas del monitoratge, primer envia de qui vol la informació i el servidor Node.js es guarda la informació, a continuació demana dades pel monitoratge i així gràcies a les funcions que té el websocket, pot rebre i enviar informació sense bloquejar l'aplicació web[24]. S'han utilitzat les següents funcions.

- *Onopen,* per tal d'executar codi en detectar una connexió oberta.
- *Onclose,* per executar codi al detectar que la connexió s'ha tancat.
- *Onmessage,* per detectar quan rep un missatge i executar codi o no per portar a terme unes accions o unes altres.
- *Send,* per enviar missatges al websocket connectat.

D'aquesta manera podem realitzar una connexió segura, no bloquejant i rebre i enviar informació.

5.3.4 Django

Pel que fa a Django, gràcies al fet que segueix el patró MVC, com s'havia comentat anteriorment, podem fer codi escalable, reutilitzable i senzill. A continuació es pot observar una imatge de l'arbre de treball que utilitza Django i explicarem els aspectes més importants de la seva implementació.

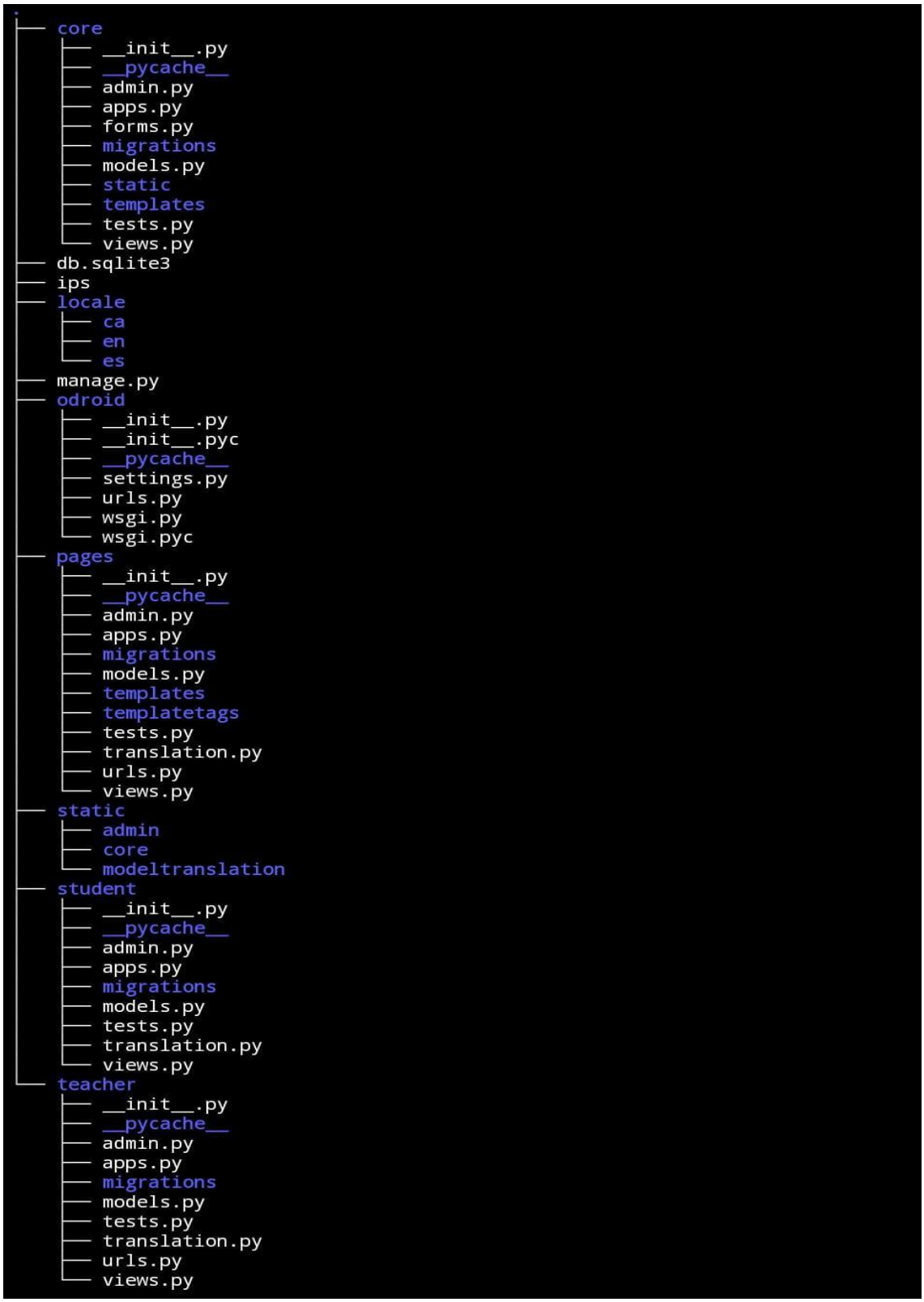

Figura 8. Arbre de treball de Django

Com es pot observar compta amb set directoris dintre el directori arrel del projecte. El directori *static* serveix per a guardar i poder localitzar els fitxers estàtics de l'aplicació web. Guardar fitxers com poden ser imatges, fitxers Javascript i d'estils. Per altra banda, el directori *locale*, serveix per guardar els fitxers generats pel **I18n**, per tant, els fitxers encarregats de manegar el canvi d'idioma.

Pel que fa als altres directoris, es pot observar que el directori *odroid*, és el subdirectori del projecte el qual conté els fitxers de configuració i les direccions permeses. Els altres directoris són Apps generades per la comanda *python3 manage.py startapp <nom>*, les quals s'utilitzen per reutilitzar codi en altres projectes, ja que s'importen al fitxer Settings del directori odroid. Aquestes Apps tenen la mateixa estructura i tenen uns fitxers encarregats de fer unes funcions predeterminades. Els fitxers models.py, views.py i el directori *templates* són els encarregats de dur a terme el patró MVC.

- *Models.py*, s'encarrega de fer de model i per tant de manegar la informació de la base de dades i de permetre la interacció d'aquesta. En aquesta aplicació web, els únics models operatius són els del *teacher, student, user i pages*.
- *Views.py*, s'encarrega de fer de controlador i per tant manega el flux de la presentació per la vista o actualitzar la base de dades, obtenir dades, eliminar-les...
- Finalment, dintre el directori *templates*, hi ha els fitxers html que són els encarregats de donar la vista de l'aplicació web. En aquesta aplicació web, només tenim vistes per l'aplicació *core* i l'aplicació *pages*.

Un cop hi ha més coneixement sobre l'estructura de treball de Django i sobre la utilització dels seus fitxers, s'acabarà l'explicació indicant que les úniques Apps que tenen models, tenen un fitxer addicional anomenat *translation.py*, el qual s'encarrega d'aplicar el patró decorator i afegir a la base de dades les columnes addicionals per a la traducció i manegament de l'aplicació web.

5.4 Comunicació

En aquest apartat, s'explicarà com s'ha comunicat tot el que s'ha implementat fins ara. És a dir, es busca explicar com s'han relacionat les part d'automatització, aplicació web i servidor. Per una altra banda, també es busca explicar la comunicació del client amb el sistema.

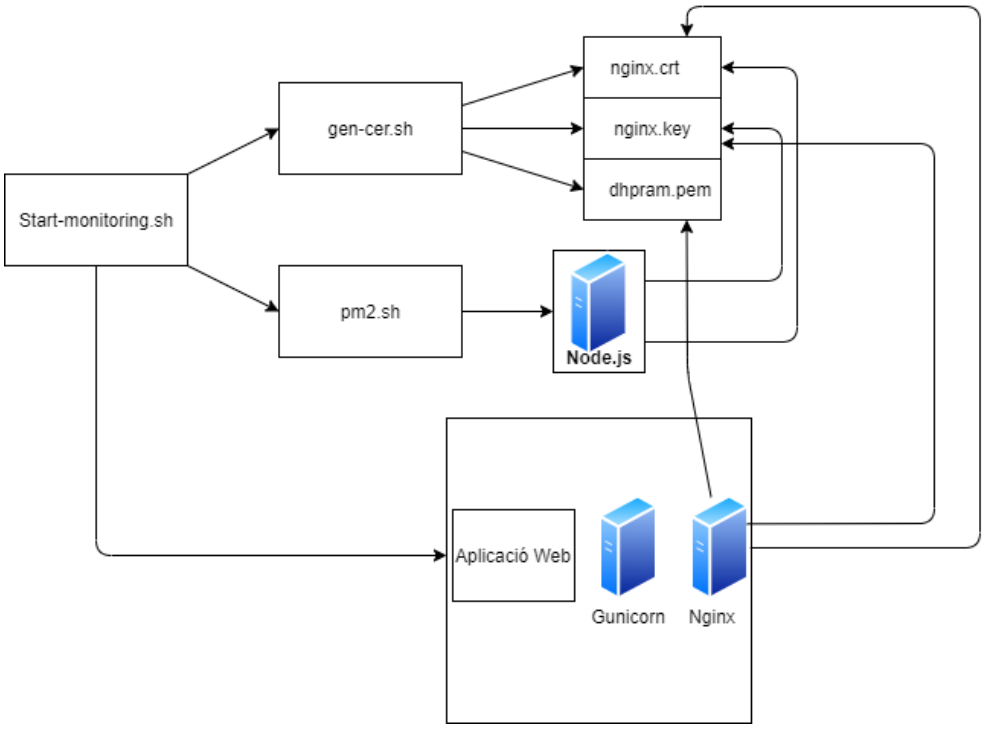

Figura 9. Procés de creació

Com es pot observar, primerament es necessita l'execució del script principal *start-monitoring.sh*, aquest amb l'ajuda dels scripts *gen-cer.sh i pm2.sh*, generarà un certificat, una clau i un DHparam. A continuació, el script haurà creat i posat en marxa els servidors web Gunicorn i Nginx. El servidor Nginx, anirà a buscar els fitxers creats pel script *gen-cer.sh* per tal de poder realitzar connexions segures. Mentre, el script *pm2.sh* posarà en marxa el servidor Node.js el qual també anirà a buscar els fitxers per a establir una connexió segura. Arribats aquest punt l'apartat d'automatització ja ha complert la seva feina que ha estat donar vida als apartats d'aplicació web i a l'apartat del servidor Node.js.

A continuació, s'explicarà la comunicació client aplicació web i aplicació web amb el servidor Node.js.

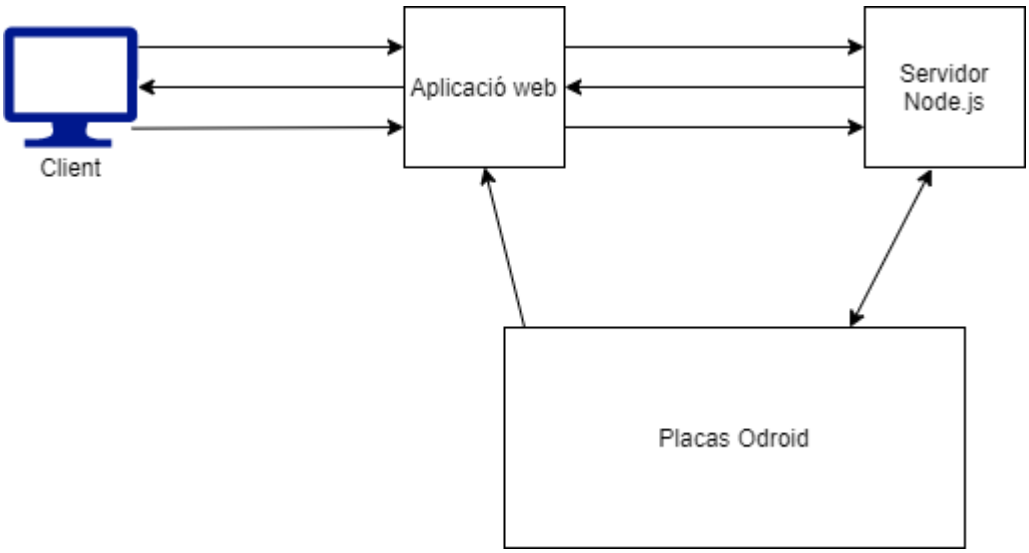

Figura 10. Comunicació

Com es pot veure, el client es comunica amb l'aplicació web. Aquest ho fa mitjançant un navegador i llavors és quan comença la comunicació client-servidor. Com s'ha vist abans, s'aplica comunicació segura a través del HTTPS[29] i dels certificats generats i per tan les tres fletxes de la imatge que hi ha entre el client i l'aplicació web, representen com es posen d'acord client i servidor per tal d'encriptar la informació. Primer el navegador voldrà connectar-se al servidor i aquest li enviarà el certificat amb la clau. Com que el certificat és auto signat, no té cap CA[30] que el validi i per tant mostrarà un missatge d'advertència. Com que la comunicació és només local, no hi ha problema, ja que només és necessari per a la encriptació. Llavors, al acceptar el certificat i tirar endavant, nosaltres enviem una clau al servidor. D'aquesta manera és podrà iniciar la comunicació segura i encriptada a partir de la tercera fletxa. Aquest procés es repeteix per l'aplicació web i el servidor Node.js i per tant un cop acceptats els certificats, el client farà una petició que serà encriptada, l'aplicació web la desencriptarà i llegirà. En cas de necessitar del servidor Node.js, l'aplicació l'encriptarà i l'enviarà al servidor i aquest la desencriptarà i realitzarà les operacions adients.

D'aquesta manera tenim una comunicació fàcil i segura entre clients i servidors i s'ha vist la relació amb les parts.

[29] Hyper-Text Transfer Protocol Secure, és la versió segura del HTTP.

[30] Certification Authority, és una unitat que vàlida els certificats i els fa confiables.

6 Avaluació

Aquesta part de la memòria tractarà de testejar el sistema de monitoratge a partir dels casos d'ús. S'ha aplicat en cada cas d'ús textual un joc de proves de manera que hi ha nou jocs de proves on cadascun d'ells té diverses proves per determinar si passen o no els casos esperats i així determinar errors no esperats per arreglar en un futur i a la vegada confirmar que el sistema funciona com s'espera. Per una altra banda, amb l'avaluació s'intenta descobrir si s'aconsegueixen complir o no els requisits funcionals.

6.1 Navegació per l'aplicació web

Un dels objectius, era aconseguir fer publicitat i per tant, fer que l'aplicació web fos intuïtiva, navegable i que controles errors. Per aquest cas, no hi ha misteri, s'ha realitzat una navegació a tots els llocs possibles de l'aplicació web obtenint els resultats esperats. En el cas de posar una secció que no té vista, salta l'error 404, pàgina no trobada.

Figura 11. Error 404

En cas contrari, es pot navegar com s'espera i veure informació sobre nosaltres, descarregar paquets per la instal·lació, veure la guia o manual.

Figura 12. Barra de navegació

Com es pot veure a la imatge anterior, també es compleixen els requeriments no funcionals amb el tipus de lletra, Optima, i els colors permesos per la Universitat a les planes web.

6.2 Canvi d'idioma

En aquest apartat, tampoc hi ha hagut molt de misteri, simplement s'han fet una sèrie de proves per veure que es complien els requisits del multi llenguatge. En el cas del **I18n**, s'ha anat a totes les seccions de l'aplicació web que no tenen elements de la base de dades i s'ha comprovat el canvi d'idioma observant el correcte funcionament i per tant l'assoliment d'arribar el màxim de gent possible i permetre una aplicació més global i internacional.

Seguidament, i de la mateixa manera, s'ha anat a les vistes que contenen elements de la base de dades per veure el correcte funcionament del **Modeltranslation**. S'ha seguit el mateix procediment de prova que el **I18n**. Com el cas anterior, s'han assolit els objectius i funciona correctament.

6.3 Iniciar sessió

Per verificar el control dels dos actors i l'inici de sessió, s'han creat dos usuaris a la plana web. Un te permisos per afegir, modificar i eliminar elements a la base de dades. Mentre que el segon no pot realitzar canvis a la base de dades. Seguidament s'ha comprovat que els dos usuaris puguin realitzar les accions comunes de monitoratge i manteniment.

A la vegada s'han realitzat proves en la introducció de la contrasenya i s'ha observat el correcte funcionament. Amb això s'aconsegueix privacitat i que el sistema de monitoratge no sigui accessible per tothom. Com s'ha dit, realment l'aplicació web estaria dividida per a fer publicitat i una segona aplicació per a fer el manteniment i monitoratge.

A continuació, es mostren imatges demostrant el correcte funcionament de l'inici de sessió.

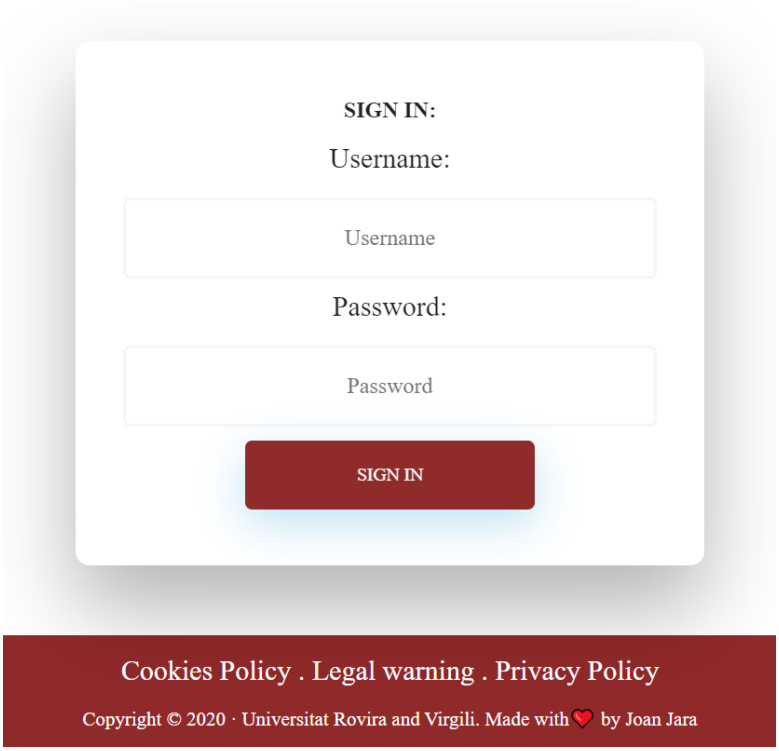

Figura 13. Pestanya d'inici de sessió

Com es pot observar, partim d'un formulari senzill per introduir l'usuari i la contrasenya. En cas d'introduir les credencials correctament es veurà la pàgina d'inici amb

el menú de navegació mostrat a la **figura 10**. En cas contrari, el servidor ens redirigirà a una pàgina d'error com la següent.

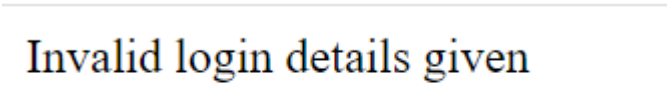

Figura 14. Credencials invàlides

Així doncs, confirmem el correcte funcionament del cas d'inici de sessió.

6.4 Tancar sessió

Aquest cas és el més senzill de comprovar ja que només apareix la opció per a tancar sessió en cas d'haver-la iniciat anteriorment. Un cop s'ha tancat la sessió, s'ha comprovat no tindre accés al monitoratge, manteniment i/o base de dades.

6.5 Afegir, eliminar o modificar un element a la base de dades

En aquest cas, s'ha comprovat la separació de privilegis entre els actors i aprofitant que ja hi ha creats dos usuaris de proves, s'ha entrat primerament a la pàgina d'administració com a usuari encarregat només del monitoratge. El resultat ha estat l'esperat ja que no te els suficients privilegis per a modificar la base de dades.

Figura 15. Permisos insuficients

En canvi, si s'inicia sessió amb les credencials del superusuari, es mostren les taules de la base de dades. Llavors pots escollir una taula i fer les operacions adients.

Figura 16. Pàgina d'administració

En aquestes proves, s'ha modificat, eliminat i afegit a l'estudiant Joan Jara Bosch. S'afegeix una imatge per veure el procés general de totes les proves a la base de dades però amb l'exemple comentat de l'estudiant Joan Jara.

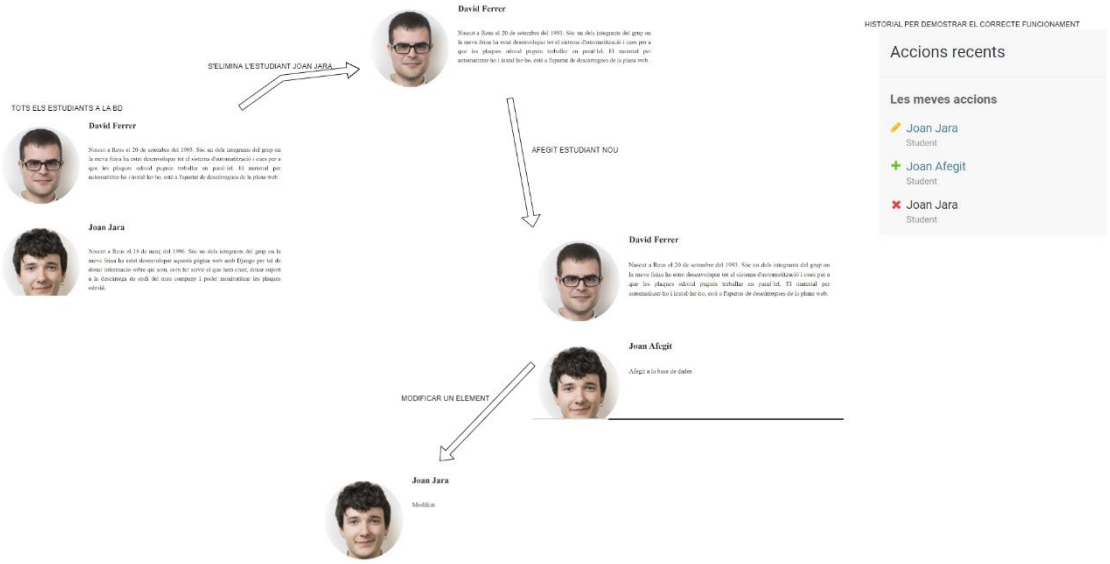

Figura 17. Test per la base de dades

Finalment, per acabar amb el joc de proves i avaluació de la base de dades, es comprovarà quin és el comportament en el cas d'oblidar-nos d'omplir un atribut de l'element a la hora de crear-lo o modificar-lo. En aquest cas, la descripció de l'estudiant.

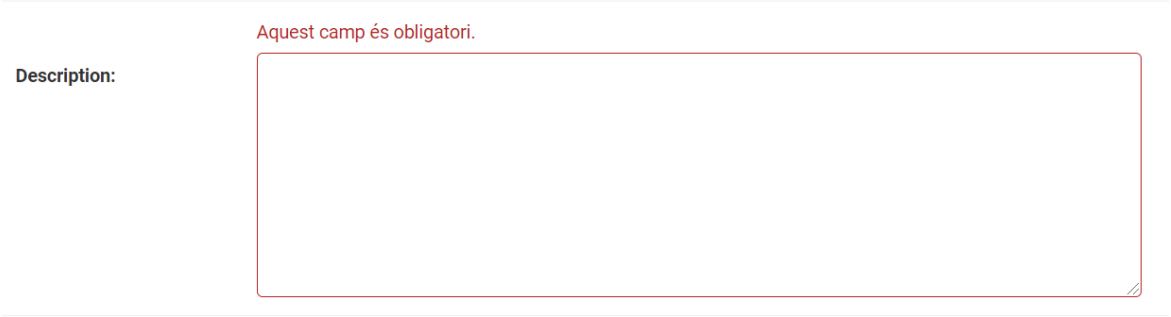

Figura 18. Camps obligatoris

Amb aquestes proves, es comprova i afirmar el comportament esperat. A més a més, compleix amb els requisits de persistir dades i poder manegar usuaris, estudiants, professors...

6.6 Monitoratge

Pel que fa al sistema de monitoratge, s'ha comprovat el funcionament d'aquest amb l'ajuda dels gràfics i la comanda *stress* per tal de posar més carrega o menys al sistema de plaques Odroid.

6.6.1 CPU

Per a fer proves per veure la utilització de la CPU, s'ha usat la comanda *stress -c <nombre treballs>*. D'aquesta manera podem veure com funciona el sistema amb pocs i molts treballs encarregats de fer operacions amb arrels quadrades.

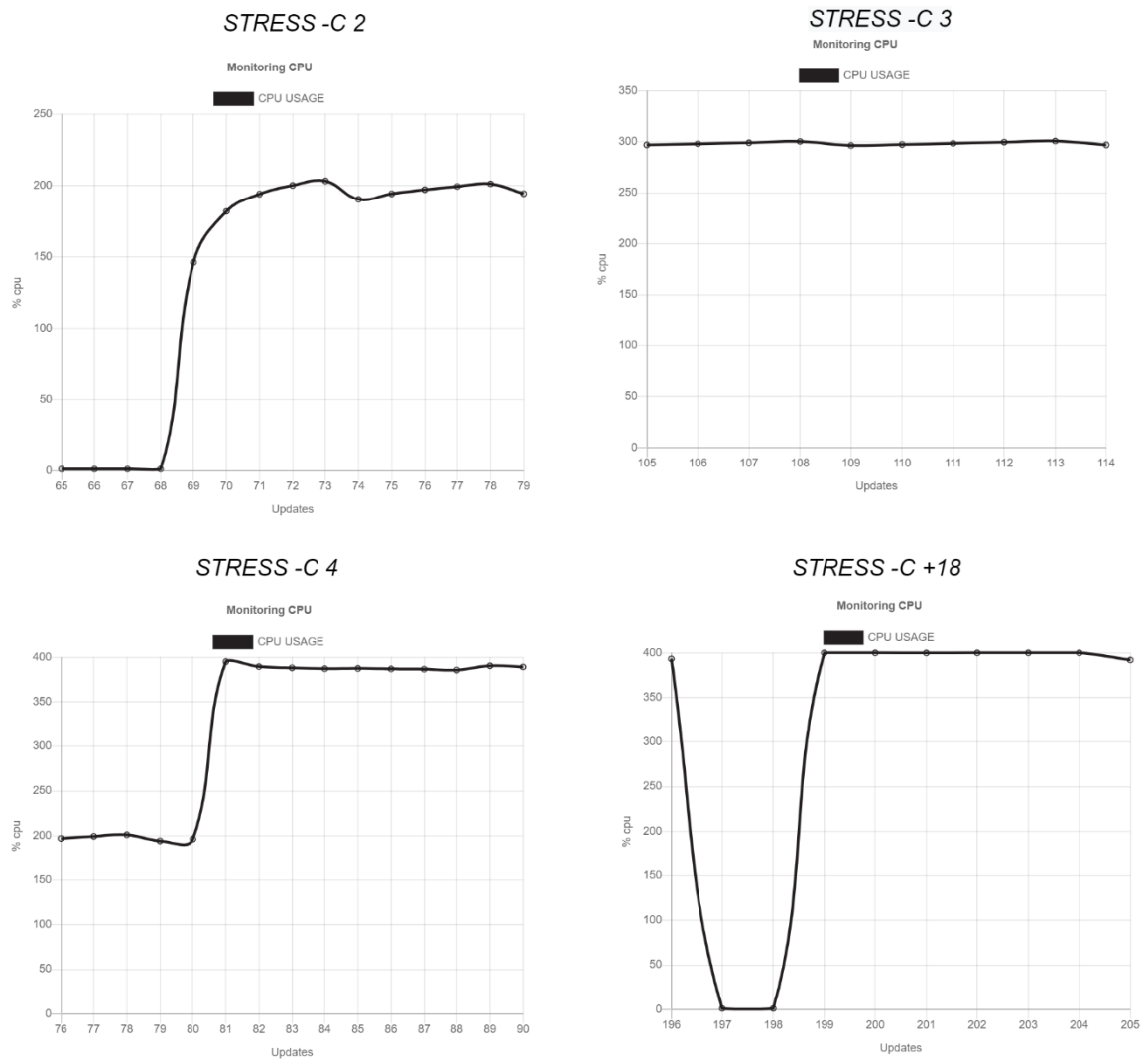

Figura 19. Monitoratge de la CPU

S'han realitzat diferents proves amb la comanda *stress,* com es pot analitzar en la imatge anterior. Primerament s'han posat dos treballadors a un node. Com a resultat la utilització de la CPU a augmentat en un 200%. A continuació, s'ha passat de dos treballadors a quatre on s'observa que el percentatge oscil·la entre el 400% i el 375%. Després, s'ha deixat una estona amb tres treballadors i ha disminuït la utilització a l 300%. Per acabar, s'han realitzat proves augmentant i disminuint el nombre de treballadors. Tots els resultats semblen normals. Aquets valors tan grans és degut a que l'obtenció de les dades es realitza sobre el total dels cores que tenen les plaques, al tindre 4 cores, hi ha de màxim un 400% d'utilització.

6.6.2 RAM

En aquest cas, s'utilitza la opció --vm per tal d'estressar a la RAM. La opció --vm, que s'utilitza per crear tants processos com s'indiquin, encarregats per a fer feines d'ocupació de RAM. Cada treballador per defecte, treballarà amb 256MB de RAM.

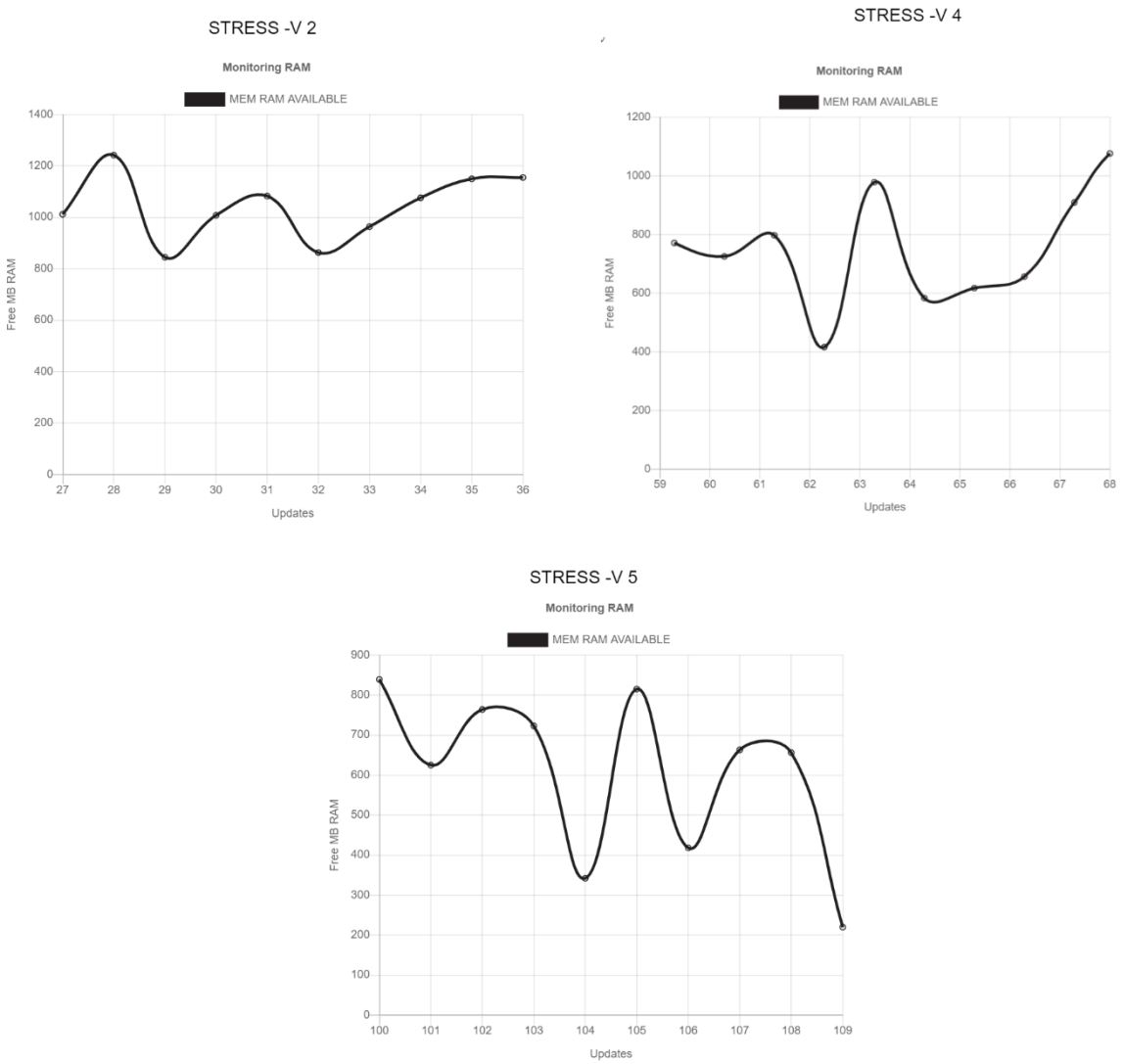

Figura 20. Monitoratge de la RAM

Com es pot observar, per cada treballador afegit, el node ocupa i allibera més RAM. Com s'havia comentat, cada treballador tracta 256MB de RAM, en el gràfic podem veure que en un principi la memòria RAM disponible era d'uns 1200 MB i per tant, el gràfic presenta un comportament adequat. A la imatge, es presenten tres gràfics amb els diferents valors passats, en el primer al tindre dos treballadors la RAM oscil·la de 1200MB a uns 800MB, en els següents gràfics, el comportament es replica com s'ha explicat.

6.6.3 DISK

Per veure el funcionament del gràfic de disc, s'han fet diverses proves d'ocupació d'espai i alliberació d'espai ocupat. Finalment, per mostrar el correcte funcionament i que dur a terme el seu propòsit, es podrà veure una imatge amb dos gràfics de disc.

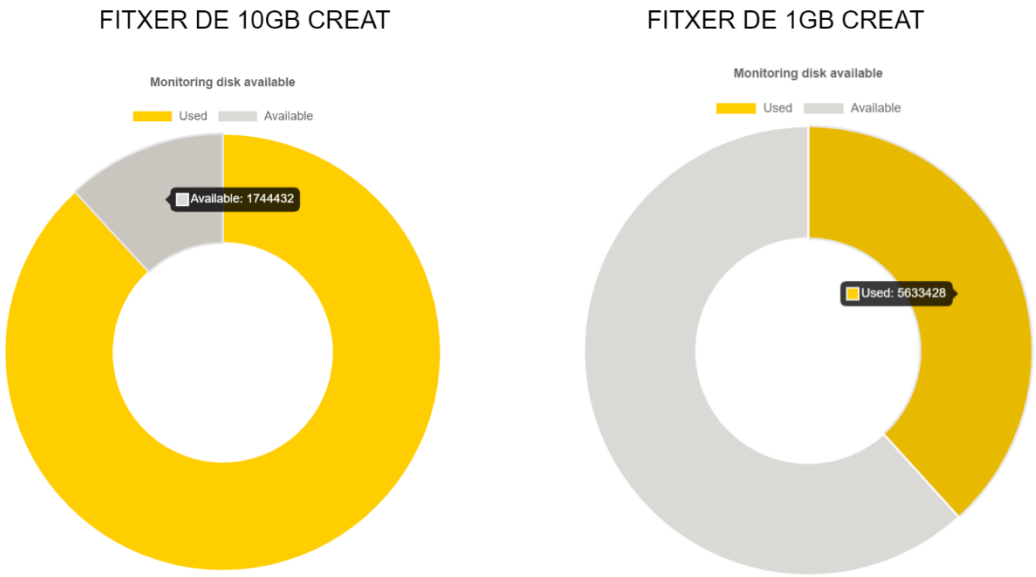

Figura 21. Monitoratge del disc

Com es pot observar, s'ha ocupat i alliberat espai de disc. S'ha usat la comanda **fallocate -l 1G test.img,** la qual col·loca 1GB en un fitxer anomenat test.img. Com s'ha pogut comprovar el gràfic per monitorar el disc, s'ha comportat de manera adequada i esperada.

6.6.4 Memòria I/O

Pel monitoratge de la lectura i escriptura a memòria, s'ha usat la comanda *stress -d <número de treballadors> i cat /dev/file*, on anteriorment, s'han copiat els fitxers creats de 10GB i 1GB a l'apartat **6.6.3**. D'aquesta manera s'ha observat el correcte funcionament del gràfic.

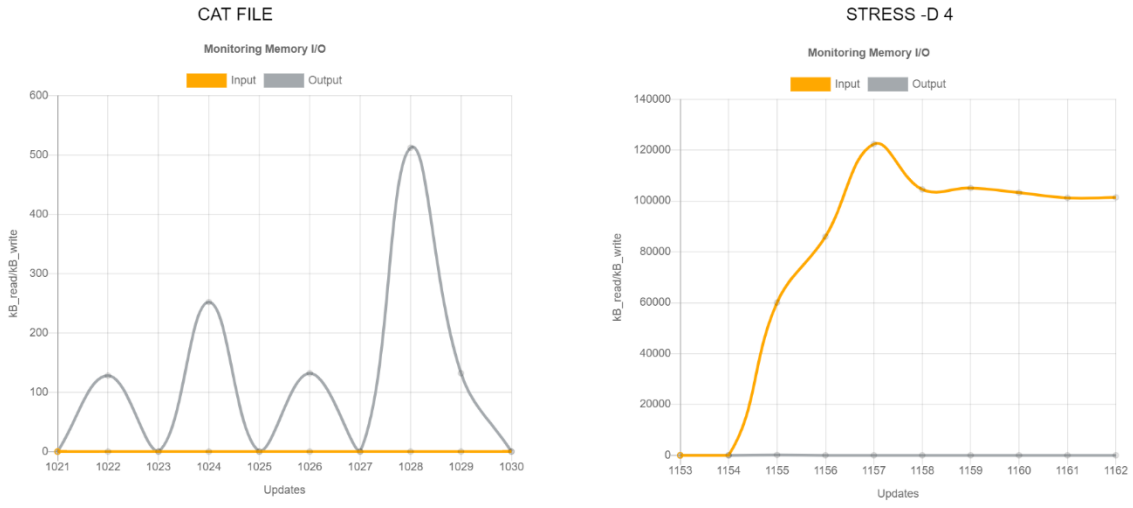

Figura 22. Monitoratge I/O

Com es visible en els gràfics, s'observa l'increment al executar les comandes esmentades anteriorment i per tant l'usuari serà capaç de monitorar correctament els KB escrits i llegits.

6.6.5 Entropia

Pel que fa a l'entropia, no s'ha pogut estressar ja que no s'ha trobat cap manera de disminuir els nombres aleatoris disponibles. Per aquest motiu, s'ha testejat el comportament normal.

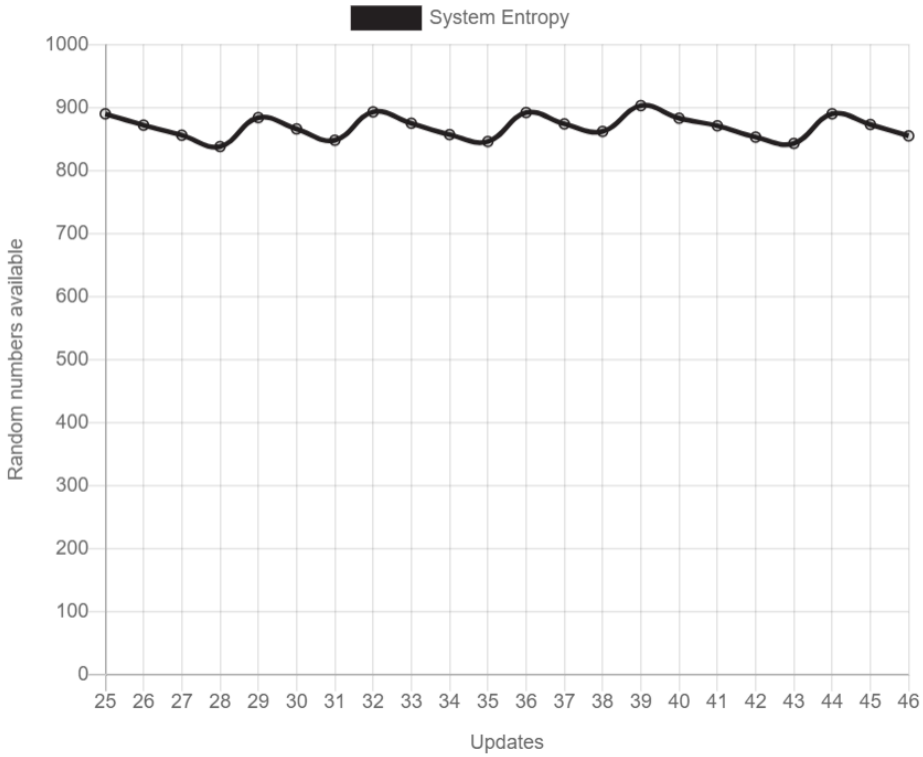

Figura 23. Monitoratge entropia

Com es pot observar, el sistema funciona correctament ja que uns valors perjudicials serien per sota de 200 ja que no podríem utilitzar comandes d'encriptació per mantenir el sistema segur.

6.6.6 Carregament del sistema

Com amb l'avaluació del gràfic de la CPU i el seu monitoratge, podem observar que el gràfic del carregament del sistema, funciona de manera similar i per tant poc a poc incrementa el carregament. En un inici augmenta el carregament d'un minut i de manera incremental, s'afegeixen els carregaments dels cinc i quinze minuts, fins arribat a carregament màxim.

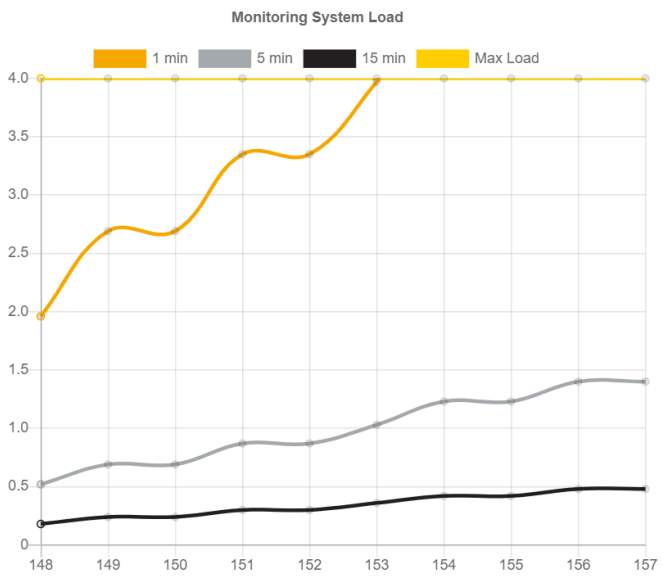

Figura 24. Monitoratge del carregament del sistema

D'aquesta manera s'obtenen resultats i conclusions com la utilització de la CPU.

6.6.7 Temperatura

En general la temperatura s'ha avaluat amb les avaluacions dels altres gràfics ja que al estressar-los en general o individualment, s'ha observat l'increment de la temperatura o el decrement d'aquesta. A la vegada, s'ha observat una temperatura màxima de 70 graus i en cap moment ha arribat a la temperatura crítica de 85 graus. La temperatura més baixa, ha estat quan no s'executava cap comanda al node i a arribat a baixar als 40 graus.

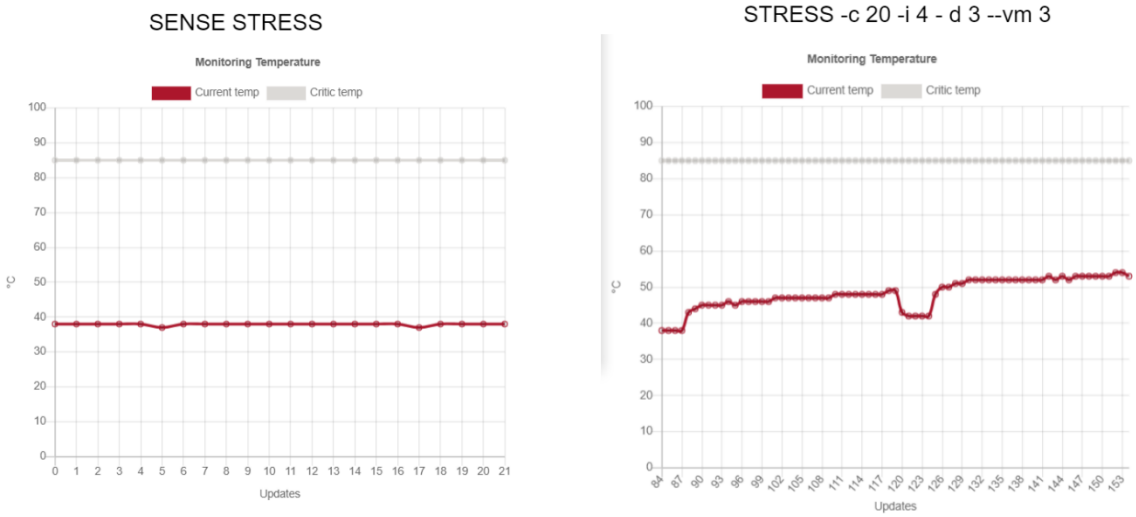

Figura 25. Monitoratge temperatura

6.6.8 Xarxa

Pel que fa a la xarxa, s'han realitzat diverses avaluacions amb la utilització de la comanda *ping*, la qual server per enviar i rebre paquets de diferents mides.

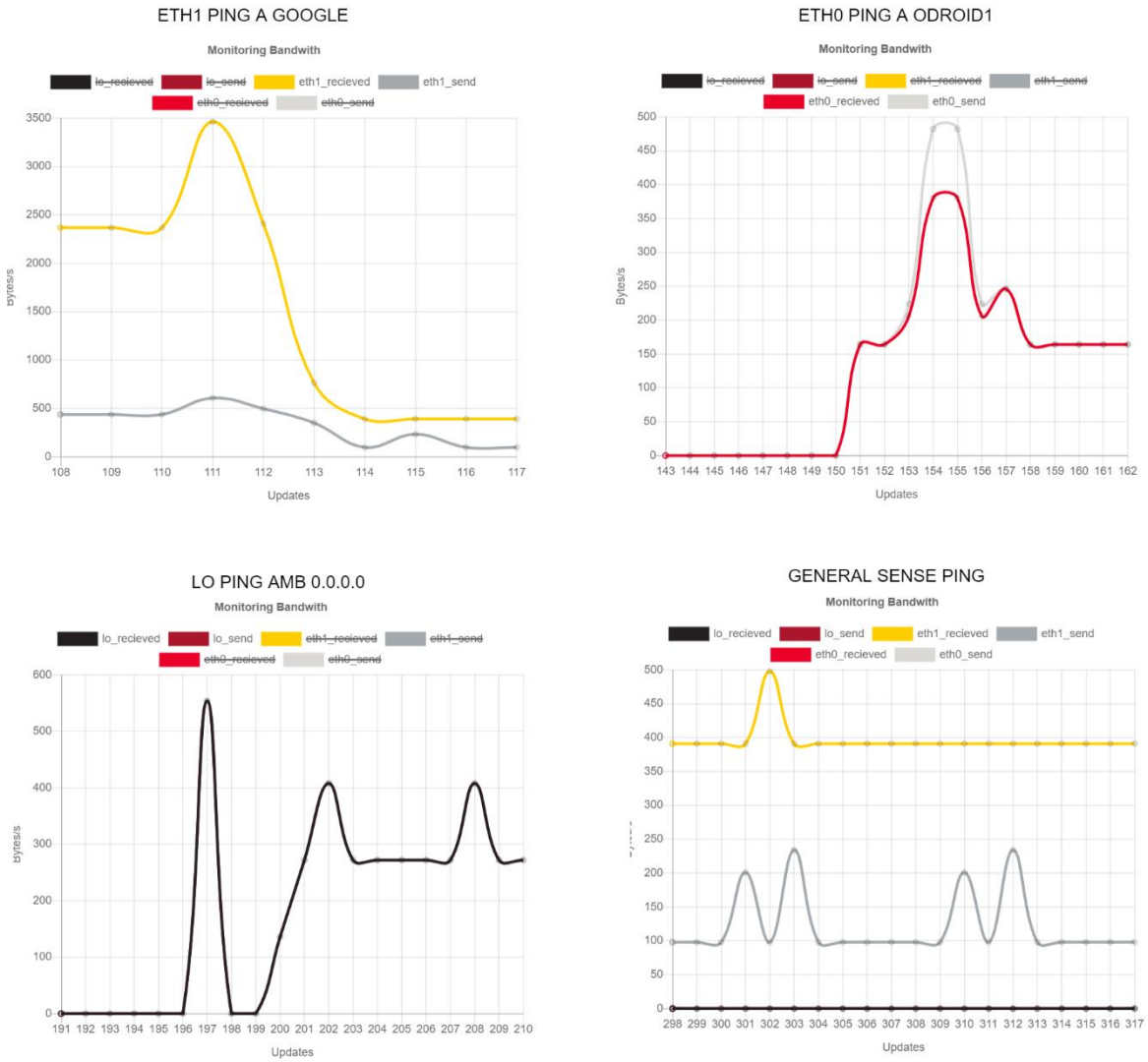

Figura 26. Monitoratge xarxa (1)

Com es pot veure a la imatge anterior, primerament, s'han tret les interfícies *lo i eth0.* D'aquesta manera es poden prendre mesures individuals per la interfície connectada al *router.* Com es pot observar, quan es porta una estona enviant paquets, al parar d'enviar-los, el tràfic disminueix. A continuació, es segueix el mateix procediment i s'avalua la interfície connectada a la LAN de les plaques. En un principi, no existia enviament d'informació entre elles i per tant el tràfic era nul però al executar la comanda *ping*, es pot veure l'increment del tràfic. En el tercer gràfic, s'ha avaluat finalment la interfície *lo,* la qual envia i rep informació amb ella mateixa, el comportament ha estat el mateix que el segon gràfic. Es partia d'un tràfic nul i al executar la comanda *ping,* el tràfic ha augmentat.

Finalment, s'observa l'últim gràfic de la imatge, el qual representa un estat sense enviament de paquets a través de la comanda *ping.* Com es pot observar el tràfic de les interfícies *eth0 i lo*, és nul però en canvi el tràfic amb la interfície *eth1* no ho és. Això es degut a la connexió SSH per tal d'executar les comandes per estressar el sistema.

Ara que s'han realitzat les proves individuals, es faran dos proves més per veure el comportament general del gràfic amb totes les interfícies juntes.

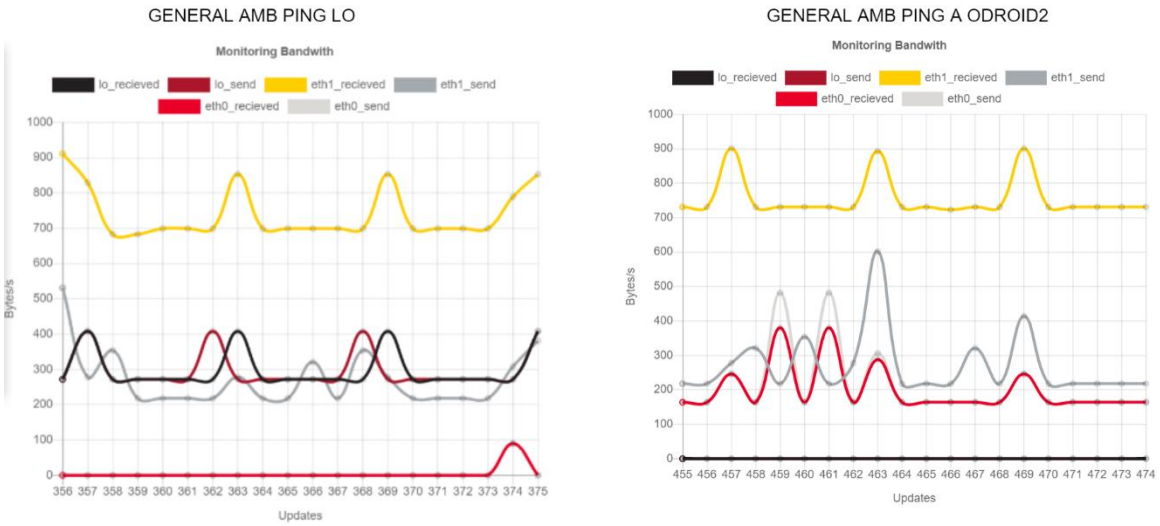

Figura 27. Monitoratge xarxa (2)

Les dades obtingudes de la imatge anterior, indiquen un correcte funcionament del gràfic de xarxa ja que en el primer gràfic de la imatge s'observa el tràfic de la connexió SSH, i el tràfic del *ping* amb ell mateix, en canvi no hi ha tràfic entre el node central i un altre node. En el segon gràfic, s'observa un comportament similar però s'han canviat els papers la inertície *lo i eth0*, ja que ara el *ping* s'ha realitzat pel node odroid2 i per tant no hi ha tràfic amb ell mateix, hi ha el tràfic de la connexió SSH i el tràfic del *ping*.

6.6.9 Resum

Com a avaluació general, es veu un correcte funcionament de tots els gràfics i del sistema ja que les proves han estat realitzades per tots els nodes. La comunicació ha estat ràpida i s'ha pogut configurar. Per tant, s'obtenen els objectius de permetre a l'usuari aconseguir un sistema de monitoratge del sistema de plaques Odroid.

A més a més, cada gràfic compleix amb el seu propòsit d'informar a l'usuari i segueix els casos d'ús. Per tant aquesta avaluació ha estat tot un èxit.

6.7 Manteniment

Pel que fa al sistema de manteniment, es va crear una secció a la plana web la qual és necessari haver iniciat sessió com un dels dos tipus d'actors.

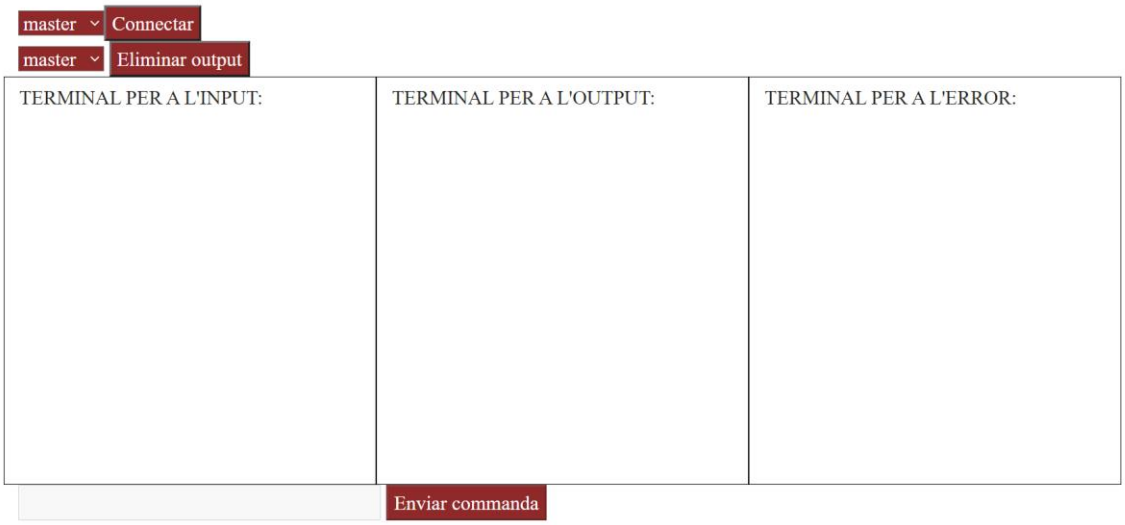

Figura 28. Manteniment

Un cop dintre s'observa que hi ha tres terminals, una destinada a fer de *echo* de la comanda enviada per tal de tindre un històric mentre es fa el manteniment. La segona, serveix pe a la sortida estàndard i l'última ens indica la sortida d'errors. A continuació, segons les proves realitzades es dividirà en tres subapartats per veure el funcionament en la placa *master*, en un node i en tots els nodes a la vegada.

6.7.1 Manteniment placa master

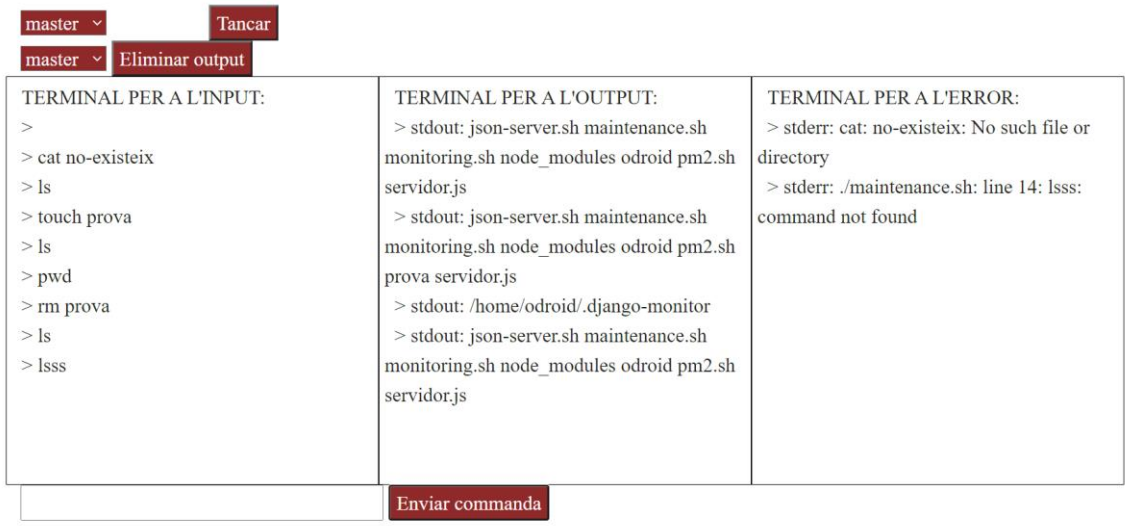

Figura 29. Manteniment amb node local

Com es pot observar amb la imatge anterior, primerament no s'ha enviat cap comanda, per tant ha fet no hi ha cap tipus de sortida. Seguidament s'ha recreat intencionadament un error per veure la sortida a la tercera terminal. Com es pot veure no existeix el fitxer o directori. Per anar acabant, s'han executat una sèrie de comandes per tal de veure la creació i eliminació d'un arxiu. Finalment, s'ha reproduït l'error de comanda no trobada.

En general el funcionament ha estat l'esperat però no s'esperava la última sortida d'error. Ja que es volia amagar de l'usuari la utilització del script *maintenance.sh*. Es volia amagar ja que si es segueix reproduint aquest error, l'usuari pot trobar estrany o molest

l'error degut a que no és el veritable error, l'error és que la comanda no existeix o està mal escrita. Degut a aquest comportament, s'ha seguit fent més proves i s'ha observat que no té la capacitat de redirecció de la sortida entre comandes. És a dir, no es poden executar comandes del tipus *ls | grep -c "hola"* o *echo "prova" > file1_prova.txt*. I per tant s'ha de buscar una solució[11].

6.7.2 Manteniment a un node remot

Seguint amb les proves per avaluar el programari, s'ha provat la funcionalitat d'enviar una comanda de manera remota a una placa.

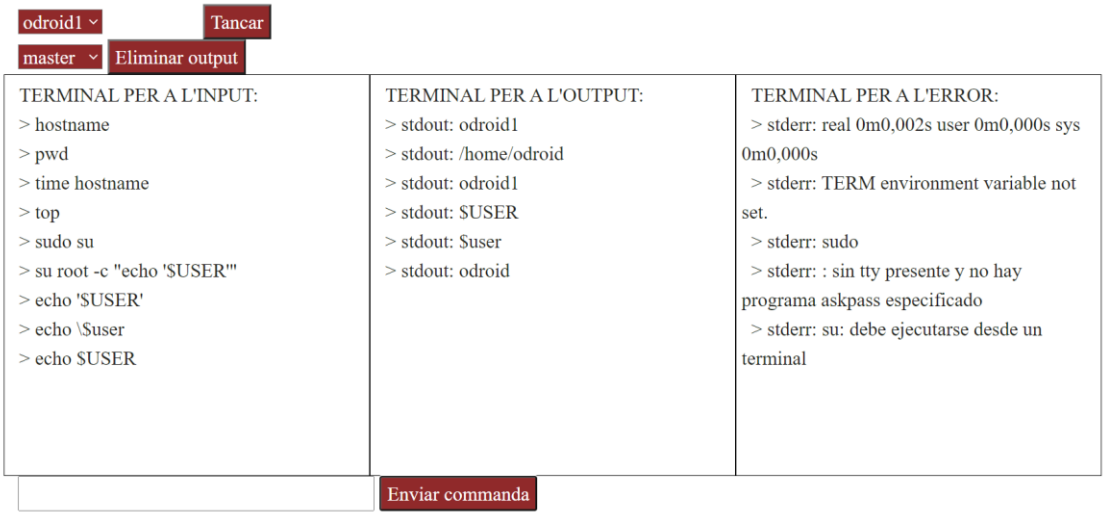

Figura 30. Manteniment node remot

Com es pot observar, hi ha el correcte funcionament de execució de comandes de manera remota amb els seus errors i les seves sortides estàndards. S'ha intentat fer una prova per a simular l'autenticació com a superusuari i s'ha mostrat l'error *sin tty presente*. Això es a causa d'executar la comanda com a node remot i per tant és necessari realitzar un SSH per a executar la comanda, al estar a una plana web, no hi ha terminal i per tant s'origina aquest error.

6.7.3 Cas general per a tots els nodes

Finalment, s'ha testejat el funcionament d'execució de comandes per a tots els nodes i aprofitant aquesta prova, s'ha afegit el funcionament de la eliminació de sortides estàndard no desitjades. Així es permet a l'usuari una major llibertat per veure el output desitjat.

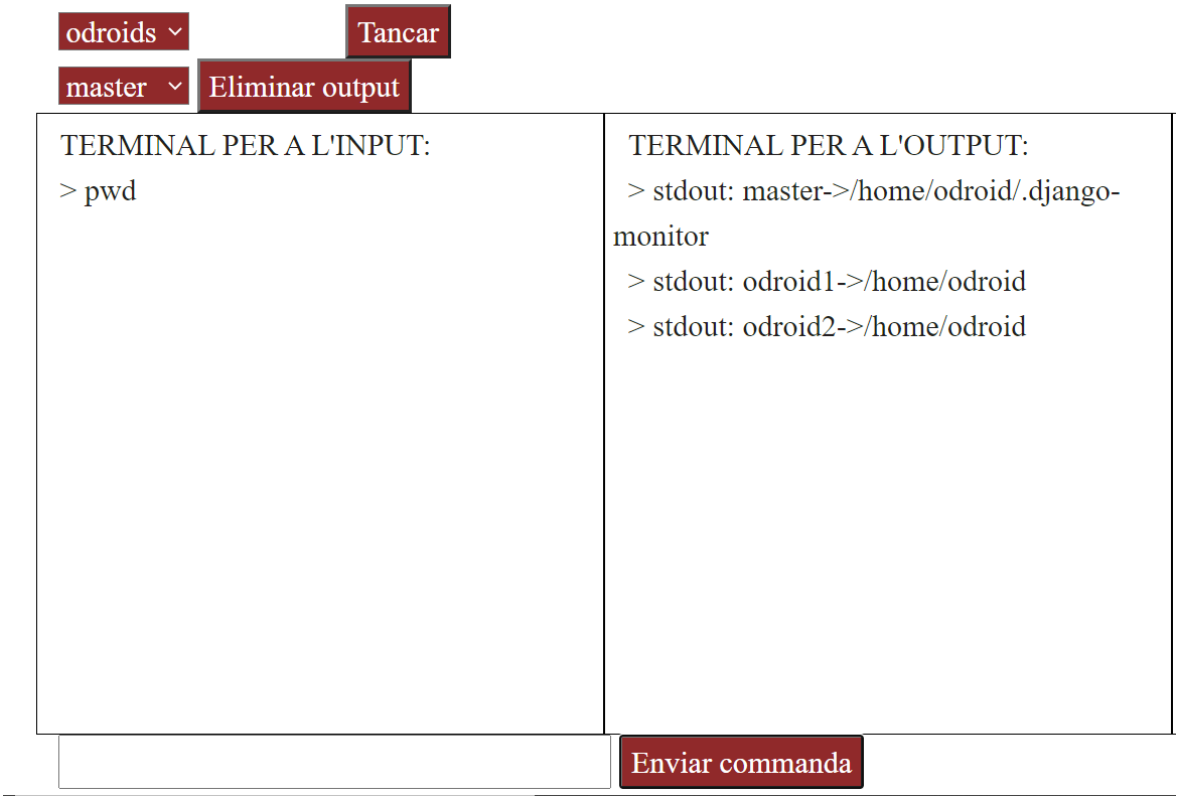

Figura 31. Manteniment global

A la imatge anterior es veu l'execució global d'una comanda per a tots els nodes.

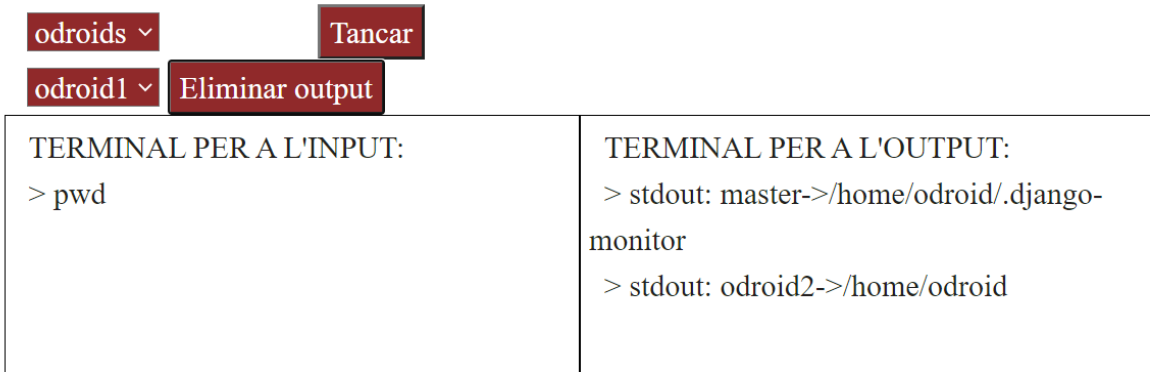

Figura 32. Manteniment, eliminar output no desitjat

A la imatge anterior, s'ha eliminat la sortida del node odroid1. Així doncs, gràcies a aquests jocs de proves, s'ha pogut comprovar la funcionalitat parcial del s requisits i objectius de manteniment ja que s'aconsegueix executar comandes de manera remota i en global però s'hauria de poder executar amb redirecció de sortida i eliminar l'error del script *maintenance.sh.*

7 Previsió de futur

En un futur si tingués més temps, m'agradaria posar més gràfics i millorar el funcionament d'aquests permetent a l'usuari l'opció d'una configuració més amplia. Per una altra banda, m'agradaria millorar bastant la part de manteniment, afegint unes terminals més reals i les quals permetin redirecció de l'output no com les que hi ha ara. Seguidament, dividiria el projecte en dos per tindre l'apartat de manteniment local i l'apartat de publicitat i donar-nos a conèixer en una altra aplicació web. Així podríem posar ja en marxa el projecte mentre llencem i millorem versions del monitoratge local i el manteniment.

Per acabar, afegiria botons a l'apartat de manteniment per poder tancar, obrir, reiniciar i actualitzar els nodes gràcies a funcionalitats com *wakeonlan* que permeten despertar i obrir sistemes de la mateixa LAN a través del port Ethernet.

Finalment, per testejar d'una millor manera, provaria el sistema amb més nodes per veure que el que s'ha implementat, pot escalar de manera adequada i funciona correctament per tal d'afegir més canvis o no.

8 Conclusions

Finalment, com a conclusions s'han obtingut que encara que una aplicació web sembli poca feina, realment és un procés llarg el qual s'han de tindre en compte tots els elements, com comunicar-los, quines funcions ha de realitzar l'aplicació entre altres. Totes aquestes coses que s'han de tindre en compte, fan que el procés de creació sigui llarg.

Personalment, m'ha ajudat a entendre una mica més la feina que pot desenvolupar un administrador, ja que a vegades s'han hagut de realitzar manteniments i canvis a causa d'actualitzacions d'Ubuntu que feien que el codi o la sortida d'algunes comandes es modifiqués. A la vegada, he entès que es necessita una mica de tot perquè jo en un principi pensava que les pàgines web i les aplicacions web no eren difícils de crear i que la gent cobrava molt per aquesta feina. Però un cop experimentada tota la feina que porta la creació d'una aplicació, m'ha fet reflexionar que totes les parts de la programació són necessàries per a algun motiu com pot ser publicitat, mantenir dades, creació i automatització i el més important, ajudar a persones.

Gràcies a aquest treball he crescut com a persona i he après a muntar una aplicació amb un framework que desconeixia però que és molt utilitzat. A muntar i configurar els servidors web necessaris per a fer funcionar aquest framework i així poder persistir les dades. A la vegada, he aplicat conceptes ensenyats durant la carrera com són el patró MVC vist a l'assignatura de programació avançada de dispositius mòbils.

Finalment, he après perquè és necessari tindre grups de treball formats per moltes persones, ja que durant la carrera he tingut bastants problemes amb els meus companys de pràctiques. He pogut veure que és necessari per dividir la feina correctament i així fer-la més ràpida. A la vegada, en dividir la feina, es poden aplicar jocs de proves més unitaris per a després passar a proves més generals pel sistema.

Així doncs, considero que la carrera i el treball de fi de grau, m'han fet créixer com a persona i a la vegada m'han format per poder afrontar les competències i reptes per a tindre en un futur.

9 Annex

9.1 Manual d'instal·lació

En aquest apart del annex, s'explicarà com s'ha de dur a terme la instal·lació amb uns senzills passos.

Primer de tot necessitem una sèrie de materials per a la instal·lació hardware, on a continuació es mostrarà una llista d'aquests materials.

- Plaques Odroid
- Switch amb tants ports com plaques Odroid, o diversos Swicth
- 1xHdmi (opcional)
- Tants clables ethernet com plaques +1 per el master cap al router
- Un router amb connexió a internet.
- Un adaptador USB a ethernet
- Ratolí (opcional)
- Teclat (opcional)

Un cop tenim el material, el primer pas serà muntar les plaques en alguna estructura per permetre la correcta ventilació. En aquest projecte disposem de tres plaques les quals s'han apilat de la següent manera.

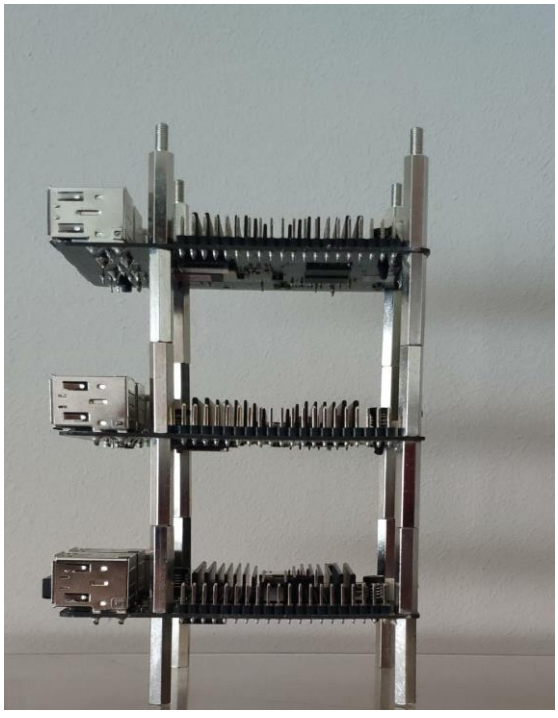

Figura 33. Manual instal·lació (1)

Un cop estan muntades, el següent pas serà el d'endollar els cables Ethernet des de el port Ethernet al port Ethernet del Switch. Seguidament, es s'endollarà l'adaptador USB a Ethernet a la placa escollida com a node central o *master.*

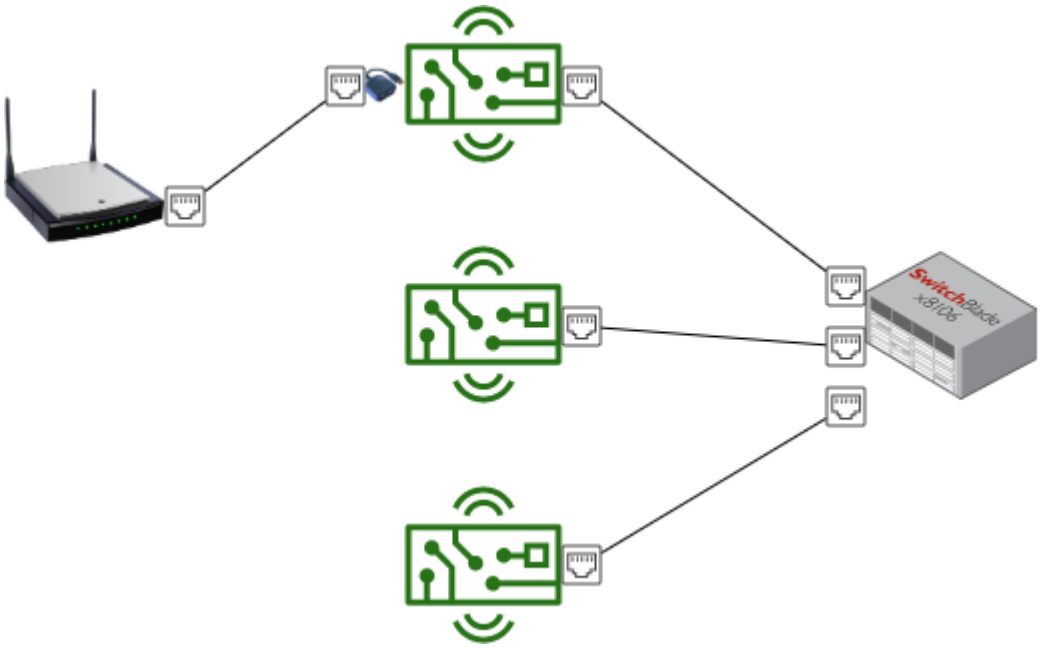

Figura 34. Manual instal·lació (2)

Finalment, només faltarà endollar les plaques i el Switch a la corrent elèctrica. Abans d'això, ens descarregarem el programa *balenaEtcher* i la imatge desitjada per a la instal·lació de cada placa. En aquest cas, tractàvem amb plaques **Odroid C2** i s'ha utilitzat la imatge d'Ubuntu. Procedim a instal·lar amb el programa *balenaEtcher,* la imatge per a cada placa. Un cop instal·lada, ja es pot endollar a la corrent elèctrica.

Un cop feta la instal·lació del *Hardware*, es procedeix a la instal·lació del *Software*. Primer de tot s'haurà d'establir una connexió SSH al node central. Per aconseguir aquesta connexió, haurem de saber la IP d'aquesta la qual es pot aconseguir a la pàgina del router. Simplement s'ha d'obrir un navegador i dirigir-nos a la pàgina **192.168.1.1**. Un cop dintre, haurem d'introduir un usuari i una contrasenya on la mateixa pàgina ja et diu com aconseguir-los, en cas contrari es recomana trucar a la companyia telefònica.

Dintre de la pàgina s'hauria de poder navegar fins trobar una cosa similar a la següent.

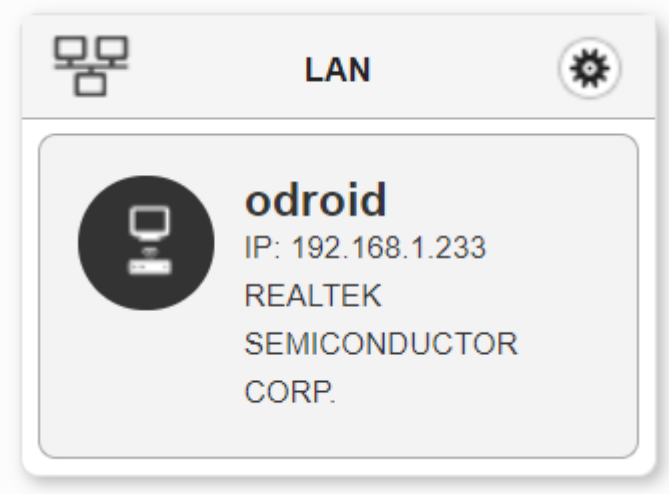

Figura 35. Manual instal·lació (3)

Com es pot observar, d'aquí s'obté la adreça IP. Un cop coneguda, ja podem establir la connexió amb la placa *master*, utilitzant la comanda **ssh nom@IP**, on el nom serà odroid i la IP serà el valor que cadascú li surti a la imatge anterior. Per tant en aquest cas seria *ssh odroid@192.168.1.233*, a continuació es demanarà la contrasenya, al ser el primer cop que entrem al sistema, la contrasenya serà la que ve per defecte per tant s'introduirà la contrasenya *odroid*.

Un cop dintre a la placa central o *master*, ens dirigim amb la comanda *cd* a la carpeta Documentos o en aquest cas *Documents* de la següent manera, *cd Documents/*, i un cop dintre s'ha d'executar la comanda *sudo apt update -y*, la qual demanarà la contrasenya que per defecte segueix sent *odroid* i a continuació llegirà els repositoris i mostrarà un missatge d'error. Per ara no cal preocupar-se del missatge d'error ja que el problema es solucionarà amb la següent instal·lació.

Un cop llegits els repositoris, s'introdueix la comanda *sudo apt-get install git -y*, i un cop s'ha instal·lat el *git*, executarem la comanda ***git clone https://github.com/davidf2/odroid-cluster.git,*** així descarregarem el treball de fi de grau del David Ferrer, el qual s'encarrega de la instal·lació automàtica del sistema. Un cop clonat, entrarem dintre el directori generat per la clonació amb la comanda *cd odroid-cluster/*, i finalment executarem la comanda *sudo ./init_master.sh* per tal d'iniciar la instal·lació i dintre d'aquesta instal·lació, es clonarà i s'instal·larà el sistema de monitoratge, ja que els dos treballs de fi de grau van junts. D'aquesta manera s'instal·larà el *Software* necessari per al funcionament del clúster i a la vegada el sistema per monitorar-lo.

9.2 Manual d'utilització

Un cop seguits els passos anterior i finalitzada la instal·lació, es podrà accedir al sistema de monitoratge a través de qualsevol ordinador que estigui a la mateixa connexió local del router. Per a utilitzar-lo, s'ha d'obrir un navegador i dirigir-se a la pàgina https://IP, on la IP serà el valor que s'hagi obtingut de la Figura 35. Manual instal·lació (3). En aquest cas ens dirigiríem a la pàgina https://192.168.1.233, la qual mostrarà una advertència. Aquesta advertència es generada a causa dels certificats auto signats i per tant que el navegador no pot confirmar que siguin vàlids. Ens dirigim a opcions avançades i acceptem. Ara ja es podrà utilitzar l'aplicació web trobada a https://IP.

En cas d'utilitzar *Firefox o* un navegador que no sigui *Chrome*, es recomana dirigir-se a la plana https://IP:3000, per acceptar els certificats del servidor Node.js, els quals s'haurà de seguir el mateix procediment anterior, ja que sinó no es podrà utilitzar els apartats de monitoratge i manteniment. Un cop acceptats tos els certificats, ja es podrà navegar per l'aplicació web i els apartats monitoratge i manteniment. S'hauria de veure una pàgina com la següent.

Figura 36. Manual d'utilització (1)

Com es pot observar, tenim set opcions a la barra de navegació. D'entre totes elles, ens centrarem a explicar la plana de l'administrador, i les pàgines *Monitoreig i Manteniment* perquè les altres planes són les de publicitat i màrqueting i per tant la navegació per elles és intuïtiva i l'usuari ja tindrà experiència en la navegació de pàgines web.

Pel que fa al monitoreig, primer s'haurà d'iniciar sessió i per tant si anem a monitoreig ens redirigirà a la pàgina d'inici de sessió. L'usuari i contrasenya per defecte és odroid i odroid. Es recomana que un cop iniciada la sessió, es canviï la contrasenya, s'explicarà més endavant. Un cop iniciada la sessió, ens dirigirem a la pàgina monitoreig.

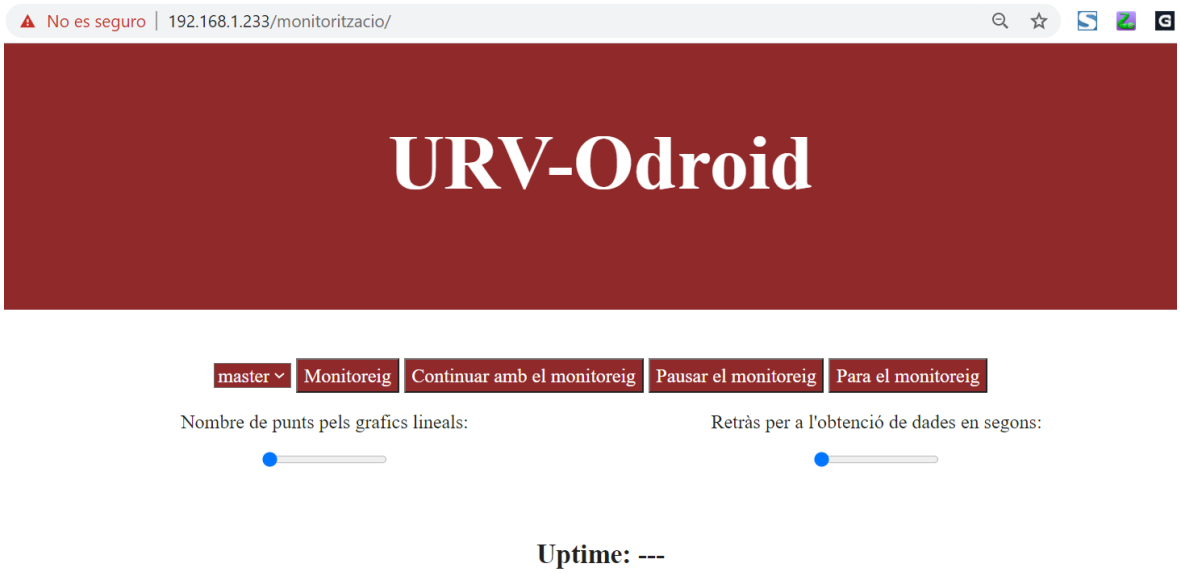

Figura 37. Manual d'utilització (2)

Com es pot observar, tenim un desplegable per escollir el node o placa per a monitorar. Un cop escollida la placa, només cal clicar el botó **Monitoreig** i ens apareixeran els gràfics. Un cop iniciat el monitoreig, podem escollir si pausar-lo amb el botó **Pausar el monitoreig** o parar-lo amb el botó **Para el monitoreig** per iniciar-ne un de nou[13]. S'ha tingut en consideració el cas que un usuari no pares el els gràfics abans d'iniciar-ne uns de nous. El botó **Continuar amb el monitoreig**, serveix per si s'ha pausat anteriorment els gràfics, reprendre la execució.

A més a més, hi ha dos sliders els quals tenen la utilitat de modificar el nombre de punts per gràfic per tindre més o menys històric i el nombre de segons de retard per la obtenció de dades, en cas de voler dades cada 20 segons per exemple. És pot observar un exemple d'utilització en les següents imatges.

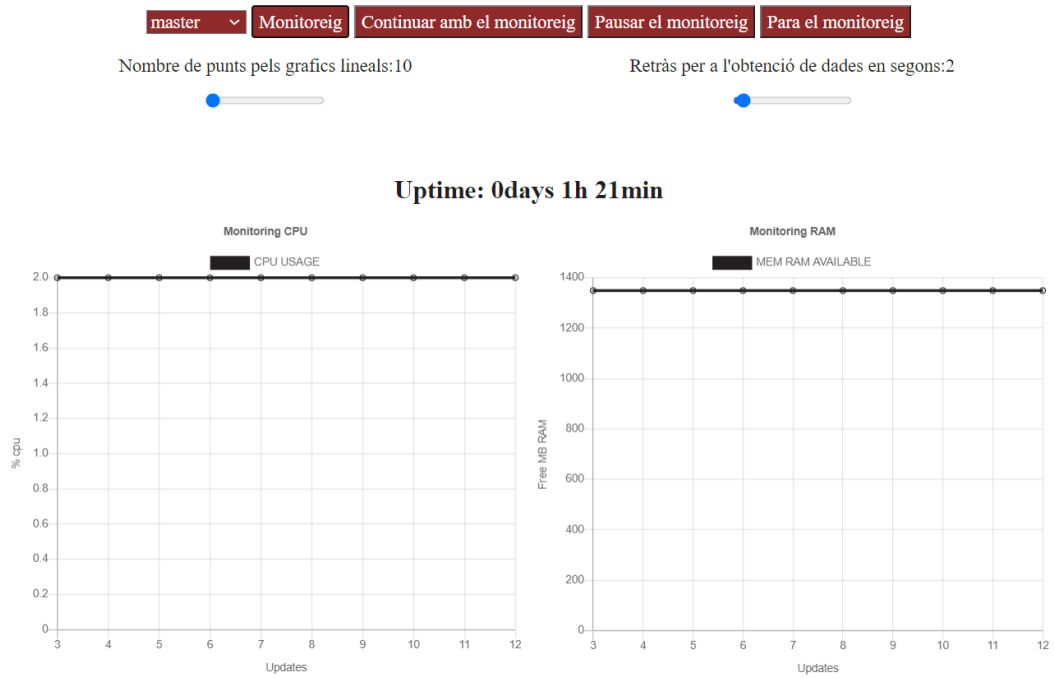

Figura 38. Manual d'utilització (3)

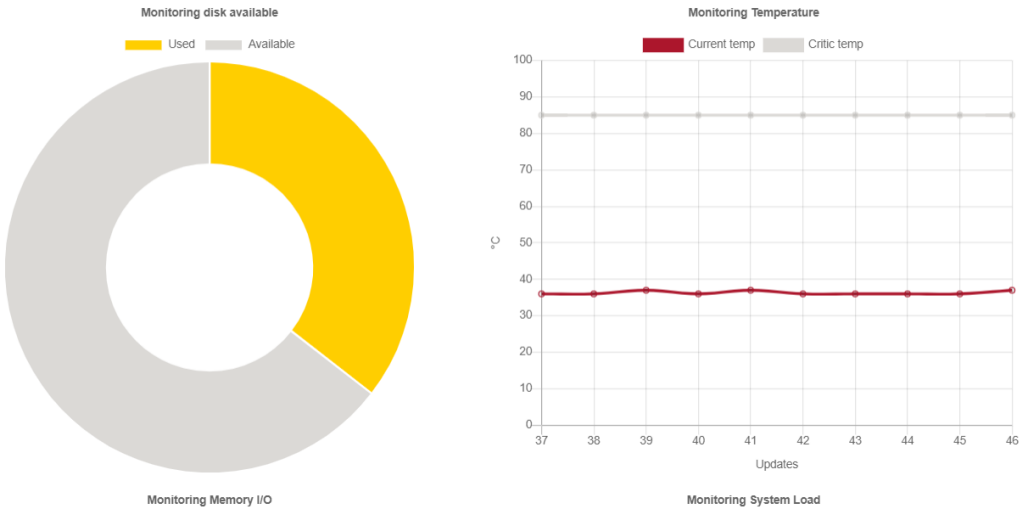

Figura 39. Manual d'utilització (4)

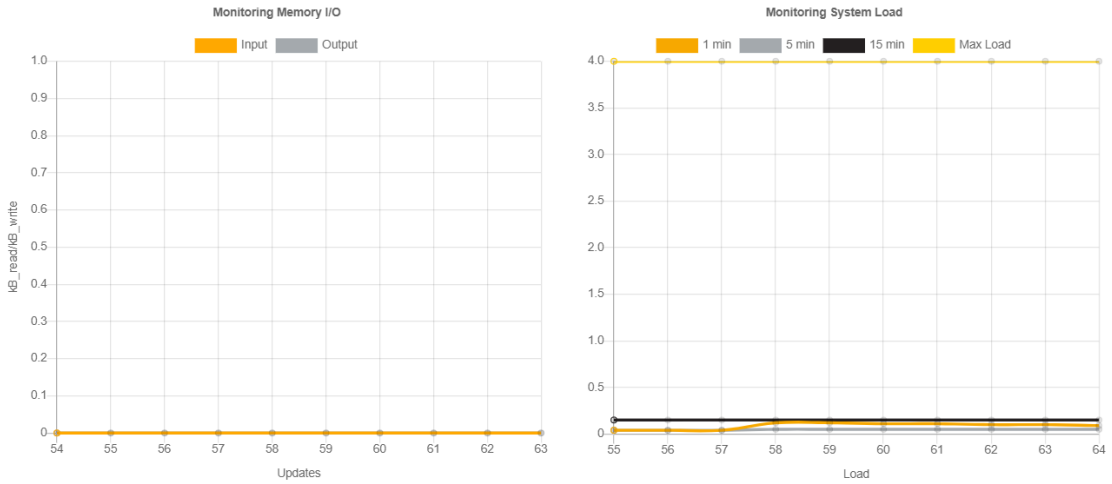

Figura 40. Manual d'utilització (5)

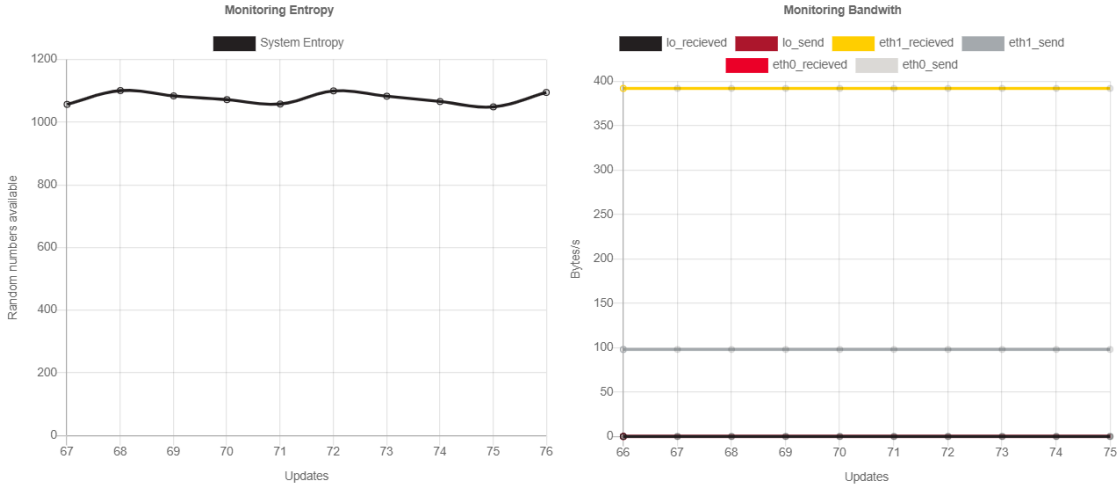

Figura 41. Manual d'utilització (6)

En el cas del *manteniment*, tindrem una pàgina inicial com la següent.

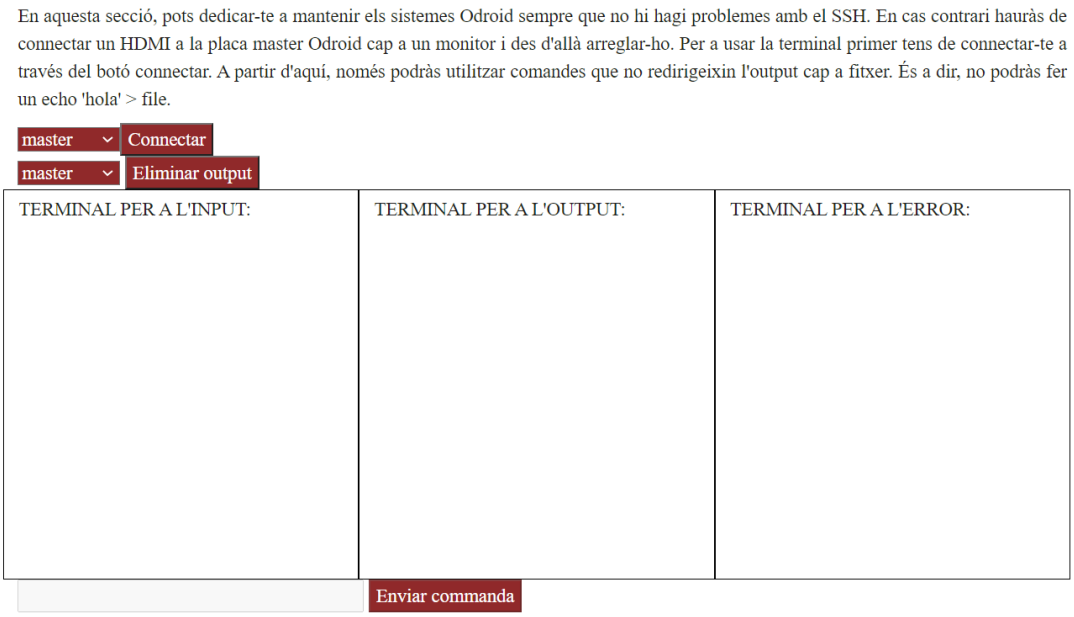

Figura 42. Manual d'utilització (7)

Com es pot observar, es disposa d'un desplegable per a escollir amb qui connectar-se i seguidament al clicar sobre el botó **Connectar**, es realitzarà la connexió amb el servidor Node.js. Un cop connectats, es podran executar algunes comandes escrites al text input i enviada clicant sobre el botó **Enviar commanda**. Un cop enviada la comanda, es realitzarà un *echo* a la primera terminal per mantenir l'històric i es mostrarà l'output estàndard i d'error si n'hi ha. En cas de voler eliminar algun output, amb el segon desplegable es podrà escollir el output de la placa que es vol eliminar i seguidament al clicar **Eliminar output**, s'eliminarà el output de les terminals de sortida estàndard i d'error.

Finalment, per a navegar a la plana d'administració de la base de dades, s'haurà de posar al navegador https:IP/admin. S'hauria de veure una pàgina similar o igual a la següent:

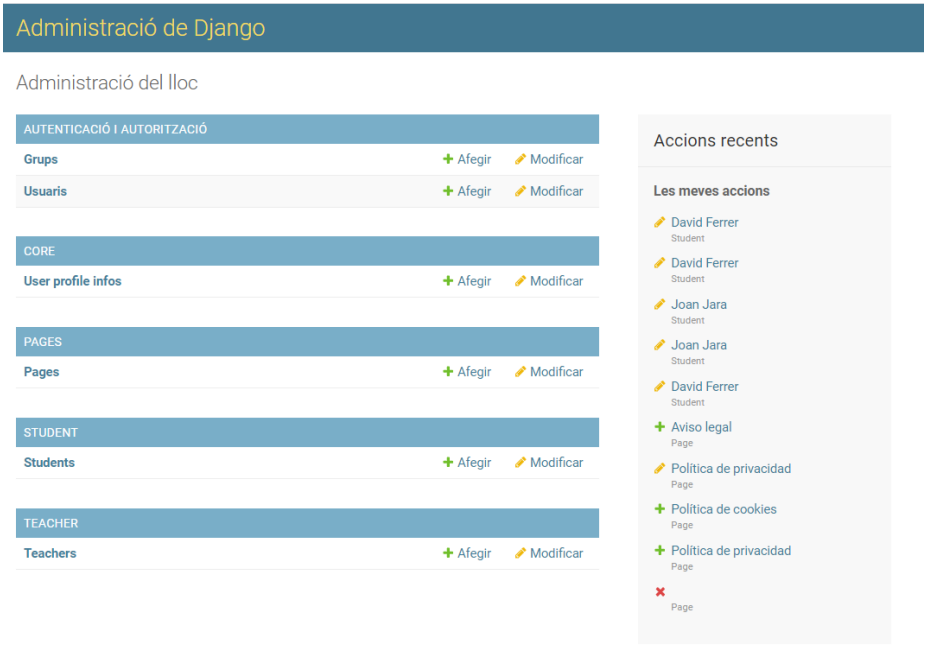

Figura 43. Manual d'utilització (8)

Com es pot observar, es poden diferenciar les taules que hi ha a la base de dades, on a cada taula hi ha la opció d'afegir o modificar. D'aquesta manera, es podrà manegar la base de dades i canviar la contrasenya d'usuari. A continuació es mostraran exemples d'utilització a la taula *student.*

Primer clicarem sobre la taula on volem afegir, eliminar o modificar un element o més d'un element.

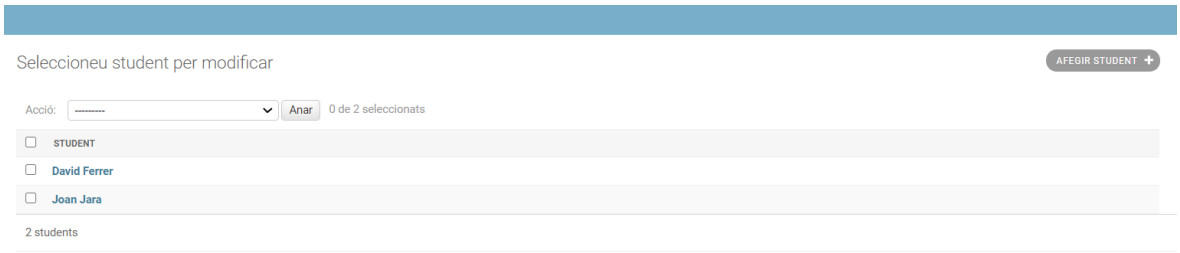

Figura 44. Manual d'utilització (9)

Com es pot observar, hi ha un botó per afegir un element o clicant sobre un dels elements de la base de dades, es podrà modificar o eliminar. Si cliquem a l'estudiant Joan Jara, és veurà una pàgina com la següent:

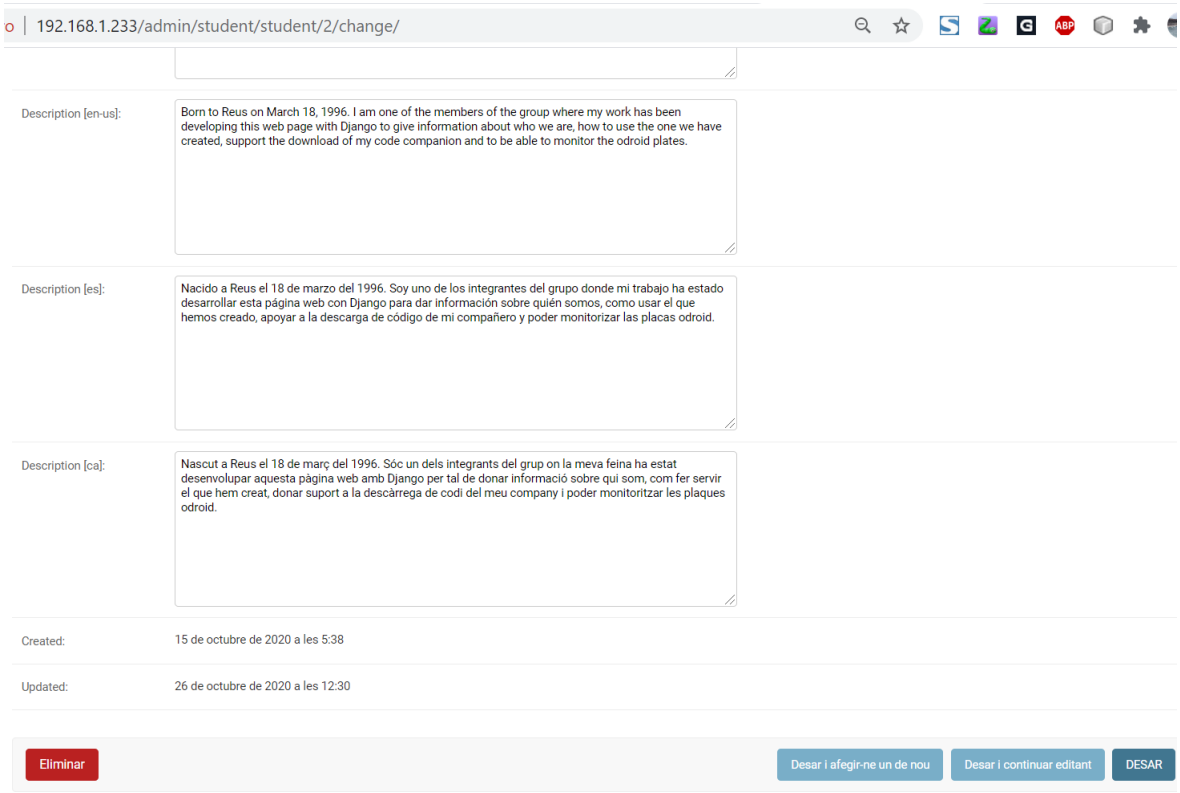

Figura 45. Manual d'utilització (10)

Com es pot observar, es poden modificar els valors dels atributs i clicar un dels botons de la cantonada inferior dreta per a desar el contingut modificat o per altra banda clicar el botó **Eliminar** situat a la cantonada inferior esquerra.

Per a canviar la contrasenya i afegir usuaris, només s'han de seguir els passos demostrats a la taula *Student* però a la taula *User*.

9.3 Fitxers de codi

En aquest subapartat del Annex, mostrarem els scripts més curts comentats anteriorment a l'aparat d'Implementació.

9.3.1 Codi Start-monitoring.sh

```
#!/bin/bash
#Author: Joan Jara
#Description: Script to download, and put the configuration files for our
webapp
# in django in order to monitor the system made of odroid. This script
will download
# and install all we need and will make up the node server too.
#Version: 1
#Date: 10/7/2020

#We need root permissions.
if [[ $EUID -ne 0 ]]; then
    echo "You must be root or sudo to do this."
    exit
fi

#Upgrades and instalations needed.
apt update -y
update="python3-pip python3 gunicorn nginx python3-django gettext nload
npm sysstat libjpeg8-dev zlib1g-dev"
for package in $update; do
      if [ $(dpkg-query -W -f='${Status}' $package 2>/dev/null | grep -c
"ok installed") -eq 0 ]; then   apt-get install $package -y; fi
done

pip3 install --upgrade setuptools
pip3 install --upgrade django
install="virtualenv wheel gunicorn requests pillow django-modeltransla-
tion"
for package in $install;do
      if [ $(pip3 list --format=columns | grep -c $package) -eq 0 ];then
pip3 install $package; fi
done
#Change permisions and config of npm for odroid user.
mkdir -p /home/odroid/.django-monitor/
mkdir -p /home/odroid/.pm2/

chown -R odroid:odroid /home/odroid/.pm2/
chown -R odroid:odroid /home/odroid/.django-monitor/

#Copy scripts and webapp to the default folder
cp -rp odroid/ /home/odroid/.django-monitor/
cp -p servidor.js /home/odroid/.django-monitor/
cp -p monitoring.sh /home/odroid/.django-monitor/
cp -p maintenance.sh /home/odroid/.django-monitor/
cp -p json-server.sh /home/odroid/.django-monitor/
cp -p pm2.sh /home/odroid/.django-monitor/
#We made a variable to know where we are.
descarregues=$(pwd)

#If this file/link already exist because of previous test,
#we gonna delete it. If not we gonna create the hard link.
if [ -f /home/odroid/.django-monitor/odroid/ips ]; then
      rm /home/odroid/.django-monitor/odroid/ips
```

```
fi

ln -s /etc/dnsmasq.d/dnsmasq_hosts.conf /home/odroid/.django-moni-
tor/odroid/ips


#Copy files to their directori to make the deamons.
cp -p $descarregues/gunicorn.service /etc/systemd/system/gunicorn.service
cp -p $descarregues/odroid_site_ssl /etc/nginx/sites-availa-
ble/odroid_site
rm -r /etc/nginx/sites-available/default
rm -r /etc/nginx/sites-enabled/default
ln -s /etc/nginx/sites-available/odroid_site /etc/nginx/sites-enabled

#We generate the keys and certificate to make our webapp lan more secure.
IP=$(hostname -I | awk '{print $1}')
$descarregues/gen-cer.sh $IP

#We reload, enable and start the deamons
systemctl daemon-reload
systemctl start gunicorn
systemctl enable gunicorn
systemctl enable nginx
systemctl start nginx

#Installing node
chown -R odroid: /usr/local/lib
chown -R odroid: /usr/local/bin

su odroid -c "./pm2.sh"

#Finally the instalation is completed.
sudo env PATH=$PATH:/usr/bin /usr/local/lib/node_modules/pm2/bin/pm2
startup systemd -u odroid --hp /home/odroid
su odroid -c "pm2 save"

echo "-------------------INSTALATION FINISHED ENJOY-------------------"

#We restart the deamons because there are cases were the instalation
needs a second restart
systemctl restart nginx
systemctl restart gunicorn

systemctl restart sshd
```

9.3.2 Codi stop-monitoring.sh

```
#!/bin/bash
#Author: Joan Jara
#Description: Script to delete and stop and delete all the monitoring.
#Version: 1
#Date: 10/7/2020

#We need root permissions.
if [[ $EUID -ne 0 ]]; then
    echo "You must be root or sudo to do this."
    exit
fi

#Uinstalling all we installed.
apt remove python3-pip python3 nginx python3-django gettext nload npm -y
```

```
pip3 uninstall --yes virtualenv wheel
pip3 uninstall --yes gunicorn
npm uninstall os -g
npm uninstall websocket -g
npm uninstall child_process -g
npm uninstall http -g
npm uninstall pm2 -g

#Deleteing the files and documents created.
rm -r /home/odroid/.npm-global
rm -r /home/odroid/.django-monitor
rm -r /etc/nginx/ssl/

#Stoping monitorix deamon and uninstalling it.
systemctl disable monitorix
systemctl stop monitorix
apt-get remove monitorix -y

descarregues=$(pwd)

#Stoping deamons and uninstalling it.
systemctl disable gunicorn
systemctl stop gunicorn
systemctl disable nginx
systemctl stop nginx
systemctl remove nginx
apt autoremove nginx -y
rm -r /etc/systemd/system/gunicorn.service
rm -r /etc/nginx/sites-available/*
rm -r /etc/nginx/sites-enabled/*

#Stoping node server
env PATH=$PATH:/usr/bin /home/odroid/.npm-global/lib/node_modu-
les/pm2/bin/pm2 unstartup systemd -u odroid --hp /home/odroid

#Finally the instalation is completed.
echo "-------------------ALL UNINSTALLED-----------------------------"


reboot
```

9.3.3 Codi Monitoring.sh

```
#!/bin/bash
#Author: Joan Jara Bosch
#Description: In this script we use the json-server script in order to
achieve the information for the graphics. We opted to do it in this way
because the json script is in a local place where
#           can be accessed for every odroid. So we only need to execute
the json script with ssh or without.
#Version: 2
#Date: 1/8/2020

if [ "$1" == "master" ]; then
      json=$(./json-server.sh )
else
      json=$(ssh -o StrictHostKeyChecking=no $1 "/home/odroid/.django-mo-
nitor/json-server.sh")
fi

sleep $2
```

```
echo $json
```

9.3.4 Codi json-server.sh

```
#!/bin/bash
#Author: Joan Jara Bosch
#Description: Script that retunrs a json with the information needed to
make the graphics in javascript. We put this script in a general place in
order to achieve the json for all the odroid cards.
#Version: 1
#Date: 26/9/2020

cpu=$(ps -A -o pcpu | tail -n+2 | paste -sd+ | bc)
mem=$(free -m | awk '{print $7}' | head -2 | tail -1)
mem_avail=$(df -t ext4 --output=used,avail | head -2 | tail -1)
temp=$(sensors | tail -2 | head -1 | cut -d " " -f 9-)
load=$(cat /proc/loadavg | awk '{print $1" "$2" "$3}')
cpunum=$(lscpu | head -3 | tail -1 | cut -d " " -f 15-)
entropy=$(cat /proc/sys/kernel/random/entropy_avail)
days=$(echo "$(awk '{print $1}' /proc/uptime) /86400" | bc)
min=$(echo "($(awk '{print $1}' /proc/uptime) %86400) /3600" | bc)
sec=$(echo "(($(awk '{print $1}' /proc/uptime) %86400)%3600)/60" | bc)
writeread=$(iostat | tail -n +7 | awk '{r+=$5}{w+=$6}END{print w" "r}')

for i in /sys/class/net/*; do RX=$(cat $i/statistics/tx_bytes); TX=$(cat
$i/statistics/rx_bytes); network=$(echo "_"$(basename $i)": "$RX" "$TX"
"$network) ; done

temp=$(echo "${temp//[+,),(,=,C,a-z,°]/}")
temp=$(echo $temp | tr -s " ")

load=$(echo $load" "$cpunum)
up=$(echo $days"-"$min"-"$sec)

echo '{"cpu": "'"$cpu"'", "mem":"'"$mem"'", "mem_avail":"'"$mem_avail"'",
"temp": "'"$temp"'", "load": "'"$load"'", "entropy": "'"$entropy"'",
"up": "'"$up"'", "net": "'"$network"'", "write_read": "'"$writeread"'"}'
```

9.3.5 Codi Pm2.sh

```
#!/bin/bash
#Author: Joan Jara Bosch
#Descritption: Script to start the node server automatically even after
reboot.
#Version: 1
#Date: 14/9/2020


#Installing npm modules for my project
cd /home/odroid/.django-monitor
update="utf-8-validate bufferutil pm2"
for package in $update; do
        if [ $(npm list -g | grep -c $package) -eq 0 ]; then   npm ins-
tall -g $package; fi
done

update="utf-8-validate bufferutil os websocket http child_process"
for package in $update; do
        if [ $(npm list | grep -c $package) -eq 0 ]; then   npm install
$package; fi
done
```

```
pm2 start /home/odroid/.django-monitor/servidor.js
pm2 startup
echo "-------------------INSTALATION FINISHED ENJOY-------------------"

mkdir -p ~/.ssh/sockets
echo -e "Host *\n\tControlMaster auto\n\tControlPath
~/.ssh/sockets/%r@%h-%p\n\tControlPersist 2000" > ~/.ssh/config
```

9.3.6 Codi gen-cer.sh

```
#!/bin/bash
#Author: Joan Jara Bosch
#Description: Script to generate the key and certificates (autosigned) to
make the websocket connection and webapp more secure.
#Version: 1
#Date: 16/8/2020

#Required
domain=$1
commonname=$domain

#We create the directory where the key will be
mkdir -p /etc/nginx/ssl/

#Parameters to generate the key
country=ES
state=Tarragona
locality=Reus
organization=URV
organizationalunit=URV
email=joan.jara@estudiants.urv.cat

#Optional
password=urvodroid

#If the domain don't exist, then we finish the script with an error
if [ -z "$domain" ]
then
    echo "Argument not present."
    echo "Useage $0 [common name]"

    exit 99
fi

#Generate the key and the crt
echo "Generating key request for $domain"
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/nginx/ssl/nginx.key -subj "/C=$country/ST=$state/L=$locality/O=$or-
ganization/OU=$organizationalunit/CN=$commonname/emailAddress=$email" -
out /etc/nginx/ssl/nginx.crt

echo "----------------------------"
echo "-----Below is your CRT-----"
echo "----------------------------"
echo
cat /etc/nginx/ssl/nginx.crt

echo
echo "----------------------------"
echo "-----Below is your Key-----"
echo "----------------------------"
```

```
echo
cat /etc/nginx/ssl/nginx.key

#Generate the pem
openssl dhparam -dsaparam -out /etc/nginx/ssl/dhparam.pem 4096

#Changing the owner and group
chown odroid:odroid /etc/nginx/ssl/nginx.key
chown odroid:odroid /etc/nginx/ssl/nginx.crt
```

9.3.7 Codi maintenance.sh

```
#!/bin/bash
#Author: Joan Jara Bosch
#Description: THis script only wants to execute commands and return their
error and output
#Version: 1
#Date: 9/10/2020

aux=""

for val in "${@:1:$#-1}"; do
      aux=$(echo "$aux$val ")
done

if [ "${@: -1}" == "master" ]; then
      $aux
else
      if [ "${@: -1}" == "odroids" ]; then
                if [ -f /home/odroid/.django-monitor/odroid/ips ]; then
                  aux2=$($aux)
                  echo "master->"$aux2

                        for odroid in $(cat /home/odroid/.django-moni-
tor/odroid/ips ); do
                                val=$(echo $odroid | tr "," " " | awk
'{print $2}')
                                aux2=$(ssh -o StrictHostKeyChecking=no
$val "$aux")
                                echo $val"->"$aux2
                        done
                fi
        else
                aux=$(echo "ssh -o StrictHostKeyChecking=no ${@: -1}
"$aux"")
                $aux
        fi
fi

exit 0
```

9.3.8 Codi servidor.js

```
//variables needed to create the server
var WebSocketServer = require('websocket').server;
var https = require('https');
var os = require('os');
var fs = require('fs');
const procesFill = require('child_process');

//getting the certificates to do the connection secure
const server = https.createServer({
```

```
  cert: fs.readFileSync('/etc/nginx/ssl/nginx.crt'),
  key: fs.readFileSync('/etc/nginx/ssl/nginx.key'),
  passphrase: 'odroid'
});

//listen the port
server.listen(3000, '0.0.0.0');
const wsServer = new WebSocketServer({
   httpServer: server
});

var whoami;
//the server gets a requets, he reads it and then responds it
wsServer.on('request', function (request) {
   //get the request
   const connection = request.accept(null, request.origin);
   //responds it
   connection.on('message', function(message){
      //if we want the json for monitoring
      if(message.utf8Data.includes("monitor-")){
            var child = procesFill.spawn('bash', ['./monitoring.sh', who-
ami, message.utf8Data.split("-")[1]])
            child.stdout.on('data', (data) => {
                  connection.send(data);
            });
      }
      else{
            //if we want to send commands to make manteniment
            if(message.utf8Data.includes("manteniment")){
                  console.log(message.utf8Data);
                  text = message.utf8Data.substring(12, mes-
sage.utf8Data.length);
                  console.log(text);
                  text= text.split(" ");
                  try{
                        var child = procesFill.spawn('./maintenance.sh',
text);

                        child.stdout.on('data', (data) => {
                              console.log('stdout: '+data.toString());
                                  connection.send("stdout: "+data.toS-
tring());
                        });

                        child.stderr.on('data', (data) => {
                              console.log('stderr: ' + data.toString());
                              connection.send("stderr: "+data.toS-
tring());

                        });

                        child.on('exit', (code) => {
                              console.log('child process exited with code
' + code.toString());
                        });
                  }
                  catch(error){
                        connection.send(" ");
                  }
            }else{
                  //or our first message to tell who we are
```

```
                whoami=message.utf8Data;
            }
        }
    });
    //close the connection
    connection.on('close', function(reasonCode, description) {

    });

});
```

9.3.9 Codi Nginx

```
#We need gunicorn because django uses WSGI (a standar). So in order to
have a server with django, we need something that can handle the WSGI and
this is the purpouse of gunicorn.
[Unit]
#We want that the deamon works always when we have network.
Description=gunicorn daemon
Wants=network-online.target
After=network-online.target

[Service]
Environment=SECRET_KEY=secret
User=odroid
Group=www-data
WorkingDirectory=/home/odroid/.django-monitor/odroid
ExecStart=/usr/local/bin/gunicorn --workers 3 --bind
unix:/home/odroid/.django-monitor/odroid/odroid.sock --bind
unix:/home/odroid/.django-monitor/odroid/odroid_pub.sock odroid.wsgi:ap-
plication

[Install]
WantedBy=multi-user.target
```

9.3.10 Codi Gunicorn

```
#We will use this file to config and use nginx for media files, reverse
proxy ....
server {
#In orderr to do a connection secure
    listen 443 ssl http2;
    error_page 497 https://$host:$server_port$request_uri;
    server_name _;
#headers to evite attacks
    server_tokens off;
    client_body_buffer_size 1k;
    client_header_buffer_size 1k;
    client_max_body_size 100M;
    large_client_header_buffers 2 1k;
    add_header X-Frame-Options "SAMEORIGIN";
    add_header Strict-Transport-Security "max-age=31536000; includeSubdo-
mains; preload";
    add_header X-XSS-Protection "1; mode=block";
    add_header Content-Security-Policy "default-src 'self' http: https:
data: blob: 'unsafe-inline' ws:; connect-src ws:" always;

    ssl off;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
```

```
#cerificates that will be used
    ssl_certificate /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key /etc/nginx/ssl/nginx.key;
    ssl_dhparam /etc/nginx/ssl/dhparam.pem;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:EC-
DHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384;
    ssl_ecdh_curve secp384r1; # Requires nginx >= 1.1.0
    ssl_session_timeout  10m;
    ssl_session_cache shared:SSL:10m;
    ssl_session_tickets off; # Requires nginx >= 1.5.9
    resolver 8.8.8.8 8.8.4.4 valid=300s;
    resolver_timeout 5s;

#our media files
    location /static/ {
        root /home/odroid/.django-monitor/odroid;
    }

    location /media/ {
        root /home/odroid/.django-monitor/odroid;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/home/odroid/.django-moni-
tor/odroid/odroid.sock;
        limit_except GET HEAD POST { deny all; }
    }

}
```

10 Referències

[1] Pàgina Web https://django-modeltranslation.readthedocs.io/en/latest/.[consulta] 23/10/2020

[2] Pàgina Web https://www.monitorix.org/.[consulta] 10/08/2020

[3] Pàgina Web https://www.chartjs.org/.[consulta] 24/09/2020

[4] Pàgina Web https://canvasjs.com/javascript-charts/.[consulta] 30/08/2020

[5] Pàgina Web https://docs.ansible.com/ansible/latest/user_guide/intro_getting_started.html.[consulta] 10/07/2020

[6] Pàgina Web http://wwwa.urv.cat/la_urv/3_organs_govern/secretaria_general/legislacio/2_propia/altres/manual_identitat_institucional.pdf/.[consulta] 20/07/2020

[7] Pàgina Web https://docs.djangoproject.com/en/3.1/ref/templates/builtins/.[consulta] 4/07/2020

[8] Pàgina Web https://www.w3.org/TR/i18n-html-tech-lang/#ri20040808.173208643/.[consulta] 23/10/2020

[9] Pàgina Web https://www.cyberciti.biz/faq/linux-unix-reuse-openssh-connection/.[consulta] 31/10/2020

[10] Pàgina Web https://django-modeltranslation.readthedocs.io/en/latest/registration.html/.[consulta] 23/10/2020

[11] Pàgina Web https://stackoverflow.com/questions/16155932/redirecting-output-to-a-log-file-using-node-js/.[consulta] 20/10/2020

[12] Pàgina Web https://stackoverflow.com/questions/43044659/what-is-the-purpose-of-using-nginx-with-gunicorn/.[consulta] 27/06/2020

[13] Pàgina Web https://stackoverflow.com/questions/24815851/how-to-clear-a-chart-from-a-canvas-so-that-hover-events-cannot-be-triggered/.[consulta] 17/10/2020

[14] Pàgina Web https://stackoverflow.com/questions/3294072/get-last-dirname-filename-in-a-file-path-argument-in-bash/.[consulta] 22/20/2020

[15] Pàgina Web https://copperegg.zendesk.com/hc/en-us/articles/214633883-What-do-TX-and-RX-refer-to-in-the-Network-Charts-/.[consulta] 22/20/2020

[16] Pàgina Web https://www.cyberciti.biz/faq/how-do-you-find-the-uptime-of-a-linux-server/.[consulta] 27/10/2020

[17] Pàgina Web https://major.io/2007/07/01/check-available-entropy-in-linux/.[consulta] 19/10/2020

[18] Pàgina Web https://askubuntu.com/questions/532845/what-is-system-load/.[consulta] 19/10/2020

[19] Pàgina Web https://stackoverflow.com/questions/9023672/how-do-i-resolve-cannot-find-module-error-using-node-js/.[consulta] 28/10/2020

[20] Pàgina Web https://stackoverflow.com/questions/14152635/node-js-websocket-module-installed-but-wont-work-in-scripts/.[consulta] 28/10/2020

[21] Pàgina Web https://askubuntu.com/questions/917147/nginx-wont-start-since-ubuntu-upgrade-16-04/.[consulta] 30/06/2020

[22] Pàgina Web https://stackoverflow.com/questions/861792/nginx-fails-to-start/.[consulta] 30/06/20202

[23] Pàgina Web https://askubuntu.com/questions/361902/how-to-install-nginx-after-removed-it-manually/.[consulta] 3/07/2020

[24] Pàgina Web https://www.tutorialspoint.com/websockets/websockets_send_receive_messages.htm/.[consulta] 15/10/2020

[25] Pàgina Web https://stackoverflow.com/questions/12524437/output-json-from-bash-script/12524510/.[consulta] 15/10/2020