Desenvolupament d'una arquitectura client/servidor per tal de fer una anàlisi exhaustiva dels logs de l'empresa

Carles Ribera Sànchez

Resum—Aquest projecte, va tenir el propòsit de desenvolupar una arquitectura client/servidor per tal de poder fer una anàlisi exhaustiva dels esdeveniments que tenien lloc als servidors de l'empresa, alhora que s'utilitzaven eines de desenvolupament modernes i que cobrissin les necessitats del projecte. El principal objectiu que va motivar aquest treball és tenir un millor control sobre els esdeveniments dels servidors, i poder reduir el temps d'actuació davant una incidència. A més, tot el projecte ha estat desenvolupat centrant el desenvolupament en l'arquitectura client/servidor, i en la seguretat de la informació i les comunicacions a aquesta. S'ha comprovat que les tecnologies utilitzades, encara que són tecnologies noves, s'han comportat adecuadament al desenvolupament del treball. Durant l'article es tractaran aspectes del projecte com els components de l'arquitectura, la seguretat en ells, o la seva comunicació.

Paraules clau—Anàlisi exhaustiu, AngularJS, Arquitectura client/servidor, Back-end, Flask, Front-end, Logs, Servidors Windows.

Abstract— This project had the purpose of developing a client/server architecture in order to be able to make an exhaustive analysis of the events that took place on the servers of the company, while using modern development tools that would cover the needs of the project . The main motivation for this work is to have a better control over the events of the servers, and to reduce the response time to unexpected incidences. In addition, the whole project has been developed focusing in a client/server architecture, and in the security of the information and the communication too. The choosen technologies, despite being relatively new, have proven to be fairly adecuate for the development of our work. In this article we will review several aspects of the project related to the development architecture, such as its inner components, their communication exchanges and its security.

Index Terms— AngularJS, Back-end, Client/server architecture, Exhaustive analysis, Flask, Front-end, Logs, Windows Servers.

1 Introducció

Aquest projecte es centra en el desenvolupament d'una eina, amb la capacitat de poder interactuar amb els servidors de l'empresa, i de manera autònoma o sota demanda d'un usuari, poder fer una anàlisi exhaustiva dels esdeveniments del servidor que es vulgui consultar, per tenir així un control de *Logs* centralitzat i poder actuar de manera àgil davant una possible incidència en un servidor.

Aquest projecte està motivat, per l'empresa *Sanitas Mayores,* que va proposar el projecte de cara a millorar el rendiment en l'anàlisi dels *Logs* dels seus servidors, tenir un millor control sobre ells, poder explotar-los, per una part de cara a fer-ne controls i actuar amb rapidesa davant d'incidències, però també, per altra banda, per millorar un aspecte de l'empresa que no estava explotat i on hi havia una possibilitat de millora per part de l'empresa de cara a futures auditories, on es demanés informació que, fins a la introducció del projecte, no es podia aconseguir o eren molt difícils de recuperar.

Aquest projecte té dues parts molt diferenciades, en primer lloc el back-end, aquest, desenvolupat en el framework de python Flask[1], que servirà les peticions dels clients, mantindrà el contacte amb els servidors de l'empresa i realitzarà l'anàlisi de les dades recollides i el seu posterior desat a la base de dades, amb un sistema gestor de base de dades en PostgreSQL[2].

En segon lloc, trobem el client o front-end web, aquest estarà desenvolupat bàsicament amb dues tecnologies relativament noves com AngularJS[3], un framework de JavaScript, o Bootstrap[4], juntament amb Google Charts[5], a més d'HTML i CSS. Aquest client, fa peticions REST al back-end, i el back-end respon amb les dades sol·licitades pel client. Per finalitzar, el client, amb les dades rebudes, omple les diferents vistes de l'aplicació.

- *E-mail de contacte: carles.ribera@e-campus.uab.cat*
- *Menció realitzada: Tecnologies de la Informació.*
- *Treball tutoritzat per: Rubén Martinez Vidal (DEIC)*
- *Curs 2018/19*

A continuació es presenta una imatge introductòria de l'arquitectura i el disseny de l'aplicació, on es mostren els

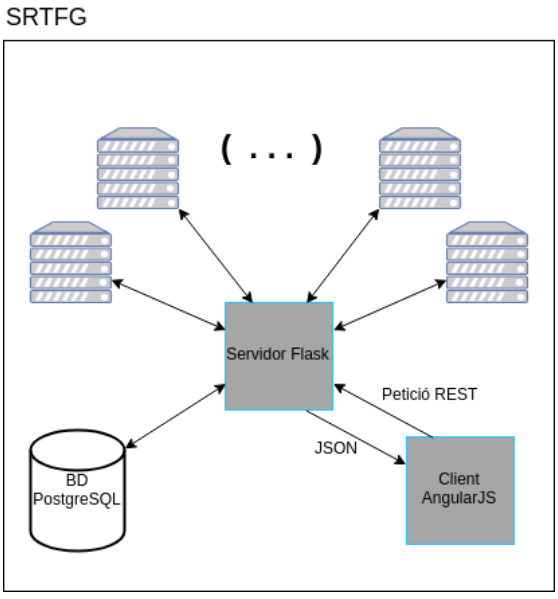

Fig. 1. Arquitectura, components del projecte, i comunicació entre ells.

diferents components d'aquesta, i el mode d'interactuar a més del pas de missatges entre les diferents entitats implicades.

El projecte està enfocat en millorar la seguretat de l'empresa, per això, el projecte, tracta diversos aspectes relacionats amb la seguretat de la informació i les comunicacions, centrant-se a assegurar una comunicació segura entre el *back-end* i el *front-end,* o assegurant un control d'accés sobre les dades, aquest i altres aspectes seran desenvolupats amb detall durant l'informe.

Aquest informe estarà dividit en cinc seccions, en primer lloc, es detallaran els objectius del projecte al llarg del seu desenvolupament, seguidament, es parlarà del context o estat de l'art on podem situar el treball, amb aquesta secció acabaria la secció introductòria al projecte.

Un cop acabades aquestes seccions introductòries, aniríem a les seccions d'exposició del desenvolupament del projecte, on es parlarà principalment de la metodologia emprada i el desenvolupament fet en ambdues parts de l'aplicació. Tot seguit s'exposaran els resultats obtinguts de la realització del projecte, on es detallarà que hem obtingut, que s'esperava.

Per tancar el document, trobarem les conclusions i quines línies de futur té el projecte, la bibliografia de l'article i els agraïments.

2 Objectius

El principal objectiu del projecte, i com ja s'ha comentat a la introducció de l'article, és proporcionar una eina d'anàlisi exhaustiva dels *Logs* dels servidors de l'empresa, que sigui capaç de fer una anàlisi proactiu i que sigui segura tant amb les dades com en les comunicacions, fiable i escalable per poder ampliar el projecte un cop tancat el Treball de Fi de Grau, per introduir-hi millores de forma fàcil.

En l'inici del projecte també es van definir més objectius, que es volien assolir a l'hora de tancar el projecte, aquests són:

- Aconseguir una arquitectura client/servidor, fent el servidor transparent pel que fa al client, mitjançant peticions REST per part del front-end per recollir informació del servidor.
- Millorar la seguretat dels servidors de l'empresa al fer un control actiu dels *Logs,* tasca que, com hem dit, no està definida a l'empresa.
- Implementar una base de dades amb un sistema gestor PostgreSQL, per desar mètriques, informació relacionada amb els servidors, o per emmagatzemar un historial d'extraccions de dades.
- Implementar un control d'accés a l'aplicació i al servidor, per evitar, per exemple, atacs de suplantació d'identitat entre d'altres i no comprometre la seguretat de l'empresa.
- Aplicació de tècniques de desenvolupament web, per optimitzar el pas de missatges entre el front-end i el back-end, i fer les comunicacions el més lleugeres possibles, mitjançant el pas d'informació en format JSON i no un fitxer HTML sencer.
- Documentar tot el projecte perquè sigui mantenible en el temps, i assegurar que tot usuari rep la mateixa informació.
- Assegurar el secret de les comunicacions xifrant amb un certificat HTTPS, les comunicacions entre client i servidor.
- Fer un servidor robust a peticions simultànies en el temps per diferents usuaris.
- Poder generar Panell de control (*Dashboard)* des de la mateixa aplicació, per explotar les dades que hi ha desades a la base de dades de forma visual.
- Poder generar informes amb la informació guardada, per a la seva impressió o desat.
- Permetre a l'aplicació una comunicació entre el servidor, i els diferents servidors a analitzar per l'aplicació.
- Poder, mitjançant una anàlisi dels *Logs* dels servidors, discernir entre *Logs* "normals", i "Anòmals", per guardar els *Logs* que no entren en la normalitat.
- Poder programar el servidor, perquè faci un anàlisi proactiu dels *Logs* dels servidors, sense interacció de l'usuari.

Aquests són els principals objectius de l'aplicació, i per tant del projecte, tots els objectius abans descrits comprenen l'abast del projecte.

3 Estat de l'art

A l'inici d'aquest projecte, es van recopilar informacions sobre estàndards d'anàlisi de *Logs* que ja existissin al mercat, encara que fossin de codi obert, i implementar només el front-end sobre aquest estàndard, i només haver d'ajustar el servidor a les necessitats del projecte.

Aquesta alternativa es va rebutjar al poc temps de començar l'estudi de la infraestructura de l'empresa, ja que

la majoria d'estàndards d'anàlisi, eren orientats a Linux, i els servidors de l'empresa no compta amb servidors Linux, sinó amb servidors Windows, però al cercar nous back-ends compatibles amb Windows en els que basar el front-end, les solucions proposades no eren adients pel projecte, i es va passar a desenvolupar un *software* propi per a l'aplicació.

Aquests estàndards de *Logs* que s'han comentat són, per exemple, Fail2Ban[6] o DenyHosts[7], aquestes solucions només compatibles per a Linux. Posteriorment es van trobar, eines compatibles amb Windows com Win2Ban[8] que és una adaptació de Fail2Ban per a Windows, però no complia els requisits mínims per entrar al projecte.

En conclusió, l'estat de l'art en aquest camp, per a Windows, i de codi obert, no hi ha una gran varietat, i com el projecte, no s'ha trobat cap solució semblant.

4 METODOLOGIA I PROCÉS DE DESENVOLUPAMENT

4.1 Metodologia

En aquesta subsecció, es pretén presentar la metodologia emprada en el desenvolupament del projecte. Aquesta metodologia, com ara veurem, s'ha centrat en tècniques de desenvolupament de software per dur a terme el projecte sencer.

Pel desenvolupament del software, tant del front-end com del back-end, es va escollir la metodologia *Rapid application development* (RAD)[9], que a més d'incloure el desenvolupament iteratiu, inclou el *prototyping*. Com diu l'article que parla sobre RAD citat de *Riffat Naz* i de *M.N.A. Khan, "es necessita menys temps per produir un software de qualitat, encara que de vegades només funcioni en projectes petits i mitjans*".

S'ha escollit aquesta metodologia, ja que amb les dues metodologies que inclou, s'ha considerat que, al ser un únic desenvolupador, al fer el prototip al final de cada fase, es pot fer una petita demostració a l'empresa. A més, es podrà assolir un objectiu relacionat amb la documentació, ja que aquesta metodologia permetrà, al final de cada iteració, documentar els canvis i millores introduïts al *software*.

Amb aquesta metodologia escollida, podrem tenir un gran feedback per part de l'empresa, i poder canviar aspectes del desenvolupament al final de cada iteració, gràcies a d'inclusió del *prototyping*, El prototipatge ens ajudarà a poder dividir el projecte en parts i a poder construir una aplicació i refinar-la perquè s'ajusti més als objectius de l'usuari final [10].

Pel que fa al desenvolupament de l'aplicació, es dividirà en iteracions, on s'implementaran funcionalitats, de més bàsica a més complexa, amb el desenvolupament en espiral podem implantar una metodologia de treball, on, fins que la funcionalitat de la iteració no sigui implementada, no es tancarà i es passarà a la següent.

Les iteracions que es van definir per seguir durant la planificació i que per tant són les fases que s'han anat seguint en el desenvolupament del projecte són:

- Primera iteració: Implementació de la descàrrega de *Logs* dels servidors.
- Segona iteració: Tractament de la informació, extracció de mètriques dels *Logs* i desat a la base de dades.
- Tercera iteració: Primera implementació del front-end i representació de la informació en el front-end d'AngularJS.
- Quarta Iteració: Implementació del control d'accés al front-end i al back.-end.
- Cinquena iteració: Implementació de l'anàlisi proactiu al servidor, i implementació de la generació de reports.
- Sisena iteració: Nou paradigma en l'extracció de mètriques.
- Setena iteració: Implementació de recapte de *Logs* "especials" per desar-los a la base de dades.
- Vuitena iteració: Implementació de funcionalitats addicional de funcionalitat a les ja implementades.
- Novena iteració: Implementació del certificat HTTPS.

Respecte a la metodologia de testeig que s'utilitzarà, es farà un test de l'aplicació al final de cada iteració, que validaran o no el tancament de la fase i l'avançament a la següent.

Al testeig tractarem a l'aplicació com si fos una caixa negra i generant entrades controlades, per concloure el test s'establiran uns llindars d'èxit, per sobre del qual s'avança a la següent iteració, això s'anomena testeig funcional[11].

Les proves o els casos de prova, estaran basats en els requeriments detallats en el Dossier del projecte, i basats també en els objectius plantejats a l'inici.

No s'utilitzaran eines de testeig, ja que els tests es basaran en entrades controlades de dades i fitxers, per comprovar que el comportament de l'aplicació és l'adequat pel desenvolupament fet.

Per tot això farem servir la tècnica d'Smoke Testing[12], o proves de fum, que consisteixen a fer proves del *software* ràpides per comprovar que funciona i que compleix amb les funcionalitats bàsiques pel bon funcionament de l'aplicació. Aquesta metodologia de testeig s'ha escollit, ja que consisteix en proves ràpides que pot fer una sola persona, i poder avançar ràpidament a la següent fase.

Per últim s'ha volgut explicar la metodologia de planificació que s'ha escollit, s'ha escollit, Kanban[13], un mètode per a la gestió del treball que permet tenir una visualització àgil de les tasques a fer, les que s'estan fent i les fetes, en una pissarra i la classificació de targetes amb les tasques al llarg de les columnes.

Per gestionar la pissarra Kanban, s'ha escollit l'aplicació Trello[14], una aplicació gratuïta que permet administrar les tasques i projectes a través d'una interfície web.

S'ha escollit aquesta metodologia, ja que és una tècnica senzilla i potent per a una bona organització, i permet tenir una visió global del projecte amb un cop d'ull al tenir totes les activitats d'aquest, sobre la pissarra Kanban.

4.2 Desenvolupament

En aquesta secció, es pretén presentar el desenvolu-

pament realitzat durant el projecte. En concret, en detallarem tres apartats.

En el primer apartat explicarem les funcions realitzades al back-end o servidor del projecte. En el segon apartat serà homòleg al primer, però amb la diferència que es parlarà sobre el front-end o client de l'aplicació.

Per acabar, l'últim apartat serà sobre funcionalitats addicionals, detalls d'implementació i decisions de disseny.

4.2.1 Back-End

En aquesta secció, es proposa una explicació detallada de la part del back-end de l'arquitectura que s'ha desenvolupat en aquest projecte. Aquí, s'explicaran les tècniques utilitzades, les decisions de disseny que s'han pres, els problemes trobats, entre d'altres.

Introducció

El back-end ha estat desenvolupat, íntegrament en python, concretament en el framework de python Flask, aquest framework, ha permès realitzar un back-end per al projecte que compleix totes les funcions desitjades, ja que s'han aconseguit assolir tots els objectius plantejats al començament del projecte, respecte a la funcionalitat bàsica que havia de tenir el back-end, perquè s'ha vist que és un framework molt robust, i amb una corba d'aprenentatge suau.

El back-end ha acabat sent, pràcticament (a excepció d'un aspecte que es tractarà com a línia de millora de l'aplicació) el que es volia des del plantejament del projecte, una entitat capaç de servir peticions REST, enviant en aquestes peticions, fragments d'informació en forma de JSON, i que tingués un paper molt important en l'anàlisi de *logs* dels servidors de l'empresa.

Descripció del back-end

L'entitat de l'aplicació que representa el Back-end, en l'arquitectura del desenvolupament del projecte, és l'entitat sobre la qual recauran les funcions d'anàlisi, estudi, desat de dades i servei de dades a les aplicacions client que consumeixin l'API REST que proveeix el servidor.

En primera instància, el back-end, tindrà la funció de respondre peticions que provinents dels clients, i servir-les minimitzant i optimitzant les comunicacions entre client i servidor, per això, s'ha dissenyat un intercanvi de missatges amb JSON, que, a l'estar construït amb AngularJS el Front-end, refrescarà la pàgina amb les dades, que rebi del servidor.

En segon lloc, aquesta entitat, és capaç de mantenir una Base de dades, on es guardaran les dades de l'aplicació, i on estaran desades, tant, informacions dels diferents usuaris de l'aplicació, com els KPI's extrets dels *logs* analitzats o *logs* que s'han decidit guardar, per motius de definició per part dels usuaris.

A més el back-end, té la capacitat de llençar codi PowerShell, contra una altra màquina, és a dir, per a la descàrrega dels *logs* dels diferents servidors, ho fem mitjançant un codi parametritzat en PowerShell, desenvolupat exclusivament per aquest projecte, on s'especifica un directori on desar els *logs*, i un servidor on llençar el powershell, un cop descarregats els *logs* dels diferents servidors, el back-end, escanejarà el fitxer descarregat, i n'extraurà mètriques, per posteriorment, desar-les en la base de dades de l'aplicació, el model de la qual s'explicarà més endavant en aquest mateix informe.

El back-end, també té la capacitat de fer una anàlisi proactiu, és a dir, el servidor, es pot programar perquè un cert nombre de vegades al dia, llenci el codi PowerShell contra tots els servidors desats a la base de dades, i així assegurar que mai, es perden *logs* de cap servidor, ja que si és així, s'entén que l'aplicació no tindria sentit.

Juntament amb aquesta capacitat proactiva d'anàlisi de *logs*, té la capacitat, també proactiva, d'eliminar registres antics, és a dir, amb un paràmetre de configuració establert a la Base de dades es podrà indicar al back-end, quin és el nombre màxim de dies que els KPIs i els *logs* especials han d'estar emmagatzemats, i si la data dels *logs* desats és anterior, s'eliminen de la base de dades, per no ocupar tant espai a la base de dades. Aquest apartat aporta un punt a favor de l'aplicació de cara a auditories que es puguin arribar a donar.

Seguretat al Back-End

Pel que fa a la seguretat de l'aplicació, es van establir dos tipus de problemàtiques principals que podrien sorgir durant el desenvolupament, aquestes problemàtiques eren, un atac de *man in the middle*, on podien ser interceptades les comunicacions entre client i servidor, al viatjar en clar. També, va sorgir la problemàtica de l'atac de *spoofing of identity*, fent un usuari entrar com si fos un altre.

Pel que fa als atacs de *man in the middle,* es va decidir, incorporar al back-end, un certificat HTTPS, per fer les connexions xifrades, punt a punt i poder estar lliures d'aquest tipus d'atac.

Per generar aquest certificat, hem generat una clau RSA aleatòria de 2048 bits, mitjançant una llavor aleatòria, i amb aquesta clau, generarem un certificat auto signat. Aquest certificat ha estat generat des del terminal de Linux amb OpenSSL.

Per últim, pel que fa als atacs d'*Spoofing of identity*, per a una aplicació construïda com la d'aquest projecte (basada en perfils), solen ser comuns, degut a usuaris que volen accedir a determinades parts de l'aplicació, on no hi poden accedir per permisos.

La solució que es proposa és utilitzar, JWT[15], o JSON Web Tokens, amb el maneig de les peticions per part dels servidors, sense augmentar de forma significativa el pes dels missatges entre client i servidor, el back-end, generarà, per cada sessió del front-end activa, un token signat per ell, amb una data de caducitat de dues hores, i durant aquest temps, només podrà fer peticions, qui estigui *loggejat* al servidor amb aquest token, i un cop es fa *LogOut* es destrueix el token, ja que no té validesa sense la signatura del servidor.

A més, també hi ha control, per quan s'accedeix a la URI del servidor, sense Token, amb un token amb la signatura no vàlida, o amb el token no vàlid.

Al cap i a la fi, tots els tokens que el servidor no pugui

validar, sigui pel motiu que sigui, el servidor no donarà una resposta vàlida, i no es rebran les dades que se sol·liciten des del front end, o des d'un navegador si la crida es feta mitjançant la URI.

A més, afegir, un parell d'aspectes de seguretat de l'aplicació, el primer aspecte seria, que s'han canviat els ports per defecte dels serveis com PostgreSQL, que s'ha canviat al port 7234 (Originalment 5432), i el port de flask que originalment és el 5000 i en aquest projecte s'ha configurat al port 7000.

A més, al codi no s'ha inclòs l'usuari administrador per fer les interaccions de l'aplicació amb la base de dades, s'ha creat un usuari amb els permisos mínims per realitzar les accions necessàries pel correcte funcionament de l'aplicació.

Model de dades

S'ha aconseguit un model final, on, permet realitzar una aplicació dinàmica, on es permet la incorporació de noves dades, perquè a l'augmentar l'abast del projecte, no comporti canvis grans com la creació massiva de taules. Aquesta dinamicitat que s'ha donat a les dades, ha estat orientada de cara a augmentar l'escalabilitat de l'aplicació en futures implementacions, ja sigui afegint noves entitats a analitzar (no només limitar l'aplicació a analitzar *logs*, ja que podria tornar-se obsoleta en poc temps) o definint noves directives de KPIs sobre les dades recollides i emmagatzemades al servidor.

Podem veure que al model de l'Apèndix A12 tenim 3 parts diferenciades, la part dels usuaris, la part de l'extracció, anàlisi i desat d'informació, i la part de cerca de *logs* especials.

La part d'usuaris està composta per dues taules, *Users* i *Profile*, en aquestes dues taules, tenim tota la informació dels usuaris, i el seu perfil, creat en taula separada per facilitar la feina de creació i eliminació de perfils, si mirem la taula de perfils, tenim una columna *Available*, en aquesta columna podem indicar si el perfil està disponible per ser triat a l'alta d'un nou usuari, principalment ha estat dissenyada, perquè el perfil d'administrador estigui, per seguretat, limitat, i no es pugui assignar a un usuari en una nova alta.

La part dels *logs* especials té tres taules, *SearchFor* i *ColumnSearch*, aquestes taules, s'han definit per, poder tenir un escaneig més en profunditat dels *logs* que es volen desar, per tant podem definir les columnes de la taula que recuperem amb els esdeveniments del servidor, i a la taula *SearchFor*, la paraula dins de la columna que es vol buscar, així no només desem *logs* amb errors, sinó que també podem buscar reinicis del servidor, errors inesperats als servidors, problemes diversos als servidors, etc.

La resta de taules, són les encarregades de portar l'anàlisi del fitxer que es descàrrega, les encarregades de desar els vectors de resultats dels *logs* que s'han analitzat, i es van analitzant dia a dia. Fent taules per fer el desament de les dades dinàmic i que no depengui de descriptius desats en una única taula.

Les taules de *Server* i *Entity*, aquí es desaran les entitats i el tipus d'entitat, respectivament, que podrà tenir l'aplicació en un futur, en la fi del projecte només hi ha incorporats servidors, però definint un nou tipus d'entitat, es podria arribar a incloure aplicacions de servidors o altres entitats productores de *logs* que serien interessants analitzar.

Per últim, per fer l'aplicació el més configurable possible, s'ha creat la taula *AppConfig*, aquesta taula guardarà registres dels paràmetres bàsics que pugui tenir l'aplicació i hagi de tenir desats, com per exemple la periodicitat amb la qual eliminarà els *logs* antics i els dies que s'han de poder tenir guardats aquests *logs* com a màxim. Aquesta taula, amb futures actualitzacions de l'aplicació, si són necessaris més paràmetres, s'haurien d'afegir noves columnes.

El disseny de la base de dades ha estat testejat durant tot el desenvolupament, i s'ha vist que és un disseny robust, i coherent per la magnitud del projecte, afegint, com s'ha dit a l'inici d'aquest secció, la característica d'escalable degut als pocs canvis que s'han de fer al model per realitzar modificacions al funcionament del projecte.

4.2.2 Front-End

Aquesta secció, seguint amb la línia de l'anterior, pretén detallar el desenvolupament realitzat sobre el front-end de l'aplicació, la part del projecte sobre la que l'usuari ha d'interactuar, i per tant la que ha d'estar més cuidada visualment.

Introducció

Durant el projecte, s'ha considerat que aquesta part de l'arquitectura del projecte, era la més important, i era sobre la que més s'havia de treballar, per evitar per una aplicació, poc funcional, o molt complexa pel seu ús i implementació a l'empresa, tot això sense menystenir el back-end, en cap dels casos, ja que pràcticament cada funcionalitat que s'ha implementat al front-end té la seva part al back-end.

Per entrar a l'aplicació tindrem un control d'accés basat en perfils, o rols, aquests perfils són, en ordre de més permisos a menys, Administrador, amb permisos de superusuari a tota l'aplicació, BasicUser, que pot gestionar la descàrrega de *logs*, però no pot gestionar els servidors, ni usuaris, i per últim tenim l'Analyzer, que només té accés als dashboards.

Les funcionalitats de l'aplicació les podem llistar de la següent manera:

- Administrar el perfil.
- Descàrrega de *Logs.*
- Gestió de servidors, que inclou l'alta i la baixa dels mateixos.
- Dashboard diari.
- Dashboard general.
- Gestió de *logs* desats especials.
- Gestió d'usuaris, que inclou l'alta, la baixa i el canvi de contrasenya dels usuaris.
- *LogOut.*

A continuació es detallarà les funcionalitats de l'aplicació seguint l'estructura del menú lateral del front-end.

El meu perfil

Aquesta opció del menú, permet a l'usuari veure dades del seu perfil desades a la base de dades, es mostraran dades com el nom complet de l'usuari, el seu nom d'accés a l'aplicació i el perfil. Aquesta opció, no plantejada als objectius inicials del projecte s'ha cregut que aporta un valor afegit a l'aplicació, al poder realitzar accions que no siguin enterament relacionades amb l'anàlisi de *logs*.

Dins d'aquesta secció, s'han implementat dues funcionalitats, de cara a ajudar a l'usuari a administrar el seu compte, aquestes dues funcionalitats són, la possibilitat de modificació de la contrasenya, i la possibilitat de modificació del nom complet d'usuari.

L'opció de canviar la contrasenya, permet a l'usuari mitjançant un formulari senzill, canviar la contrasenya l'accés a l'aplicació, aquest formulari consta de tres camps, contrasenya antiga, nova contrasenya i confirmació de la nova contrasenya.

A la part de front-end, es comprova que les contrasenyes noves siguin iguals i que tinguin el format correcte, i un cop es confirmi que són iguals i el format és acceptat, s'envien les tres contrasenyes al back-end, mitjançant una petició, i al back end es compara la contrasenya antiga amb la qual l'usuari ha proporcionat, si es valida, es desa la nova contrasenya a la base de dades amb un hash sha-256, i es retorna un codi de confirmació al front-end, i si no coincideixen les antigues contrasenyes, s'envia codi d'error.

L'opció de modificació de nom, molt més senzilla que l'anterior, només precisa d'un formulari d'un sol camp, on l'usuari, introdueix el nou nom que vol adquirir, i en fet *submit*, el nom canvia.

Amb aquesta funcionalitat s'ha definit que, l'usuari no tingui restriccions en canviar-se el nom complet que reflecteix l'aplicació, ja que, no és un camp crític per a l'accés i ús de l'aplicació.

El procés de canvi és senzill, l'usuari entra un nom, el front-end valida el format, i l'envia al servidor, on després d'una validació, fa una actualització del registre de l'usuari a la base de dades.

Descàrrega de *logs*

Aquesta funcionalitat de l'aplicació permet escollir, actualment, un servidor, un tipus de log que es desitja descarregar, s'envia una petició al servidor, aquest realitza els processos pertinents per descarregar, parsejar i extreure KPIs, i envia una resposta de confirmació o d'error al front-end.

Aquesta vista de l'aplicació consta d'un formulari de dos camps, on escollir, servidor i tipus de log, en transmetre el formulari, s'envia petició de descàrrega al servidor, i aquest llença un PowerShell contra el servidor escollit, per l'usuari, i analitzarà els *logs*, desant els KPIs a la base de dades, si no hi ha cap problema, retornarà un codi de validació, en cas contrari, es retornarà un codi d'error, i es mostrarà a l'aplicació que tot ha anat bé o en cas contrari que alguna cosa ha sortit malament.

S'ha afegit un tercer tipus de missatge, on, si no hi ha *logs* nous per descàrrega, l'aplicació avisa a l'usuari que no hi ha *logs* nous a descarregar.

Es creu que l'explicació anterior queda més aclarida si va acompanyada d'un diagrama de seqüència, com el que apareix a la secció d'apèndix A1.

Gestió de servidors

L'alta de servidors, implementa un formulari on, en la versió final del projecte, s'ha d'indicar la IP del servidor i el sistema operatiu.

Un cop s'introdueixen les dades del servidor, aquestes s'envien al servidor que crearà una carpeta que tindrà com a nom la IP del servidor, i en la que s'hi guardarà els fitxers de log, la ruta d'aquesta carpeta queda desada en la base de dades per fer de forma dinàmica la descàrrega sense importar que sigui servidor o aplicació.

El servidor queda donat d'alta, a partir d'aquest moment, aquest servidor apareixerà en els desplegables de descàrrega de *logs* i de generació de Dashboards, i estarà disponible perquè el thread de descàrrega en background del servidor, l'agafi i descarregui tots els seus *logs*.

El flux d'alta podria quedar perfectament representat en el diagrama de seqüència que apareix a la secció d'apèndix A2.

La baixa de servidors és la secció contrària a l'anterior, aquí s'eliminaran els servidors de la base de dades, en la vista que se'ns presenta, apareix una taula amb tots els servidors i una petita mostra d'informació d'ells, i un botó, on podem eliminar el servidor.

Un cop eliminat el servidor, no es podrà tornar a recuperar, i en cas de necessitar-lo de nou s'hauria de tornar a donar l'alta. D'altra banda la carpeta amb els *logs* que es descarreguen, no queda eliminada, i segueix disponible per a la consulta.

Un diagrama aclaridor d'aquesta secció, podria ser el que es presenta a la secció d'apèndix A3.

Dashboard diari

Aquesta opció implementada es considera, juntament amb la següent, les funcionalitats més importants del front-end, ja que bàsicament, són funcions, bàsiques que havia d'haver a l'aplicació.

En primer lloc descriurem la funcionalitat del dashboard diari, aquest apartat, ens proporciona, un formulari amb tres camps on especificarem, quin és el servidor sobre el que volem generar el report, quin tipus de *logs* volem recuperar, i de quin dia ho volem fer, tot això s'envia al servidor, i retorna un JSON amb les dades necessàries per construir el Dashboard.

El dashboard, es construeix, consumint l'API de google charts, i es generen diferents tipus de gràfics. Una vegada generats els gràfics, tenim l'opció de generar un report, i el navegador ens generarà un informe en PDF, que l'usuari podrà consultar malgrat ja no es trobin els *logs* a la base de dades.

Per aclarir la funcionalitat, es presenta un diagrama de seqüència a la secció A4 de l'apèndix, on es detalla la interacció entre client i servidor.

Dashboard general

Aquesta funcionalitat, semblant a la següent, conté el

mateix procés que l'anterior, recapte de dades, generació de dashboard i possibilitat de generar l'informe.

En primera instància, es recullen les dades de servidor, tipus de log a analitzar, i la possibilitat de 5, 10 o 15 últims dies de *logs*, un cop tramesos aquests paràmetres, com en el cas anterior, el servidor retorna un JSON amb les dades que es mostraran al dashboard.

Per entendre la generació del dashboard i com arriben les dades al client, tenim en compte el diagrama de la secció A5 de l'apèndix.

Per acabar m'agradaria mostrar el diagrama de seqüència que recupera l'execució de la generació de l'informe un cop s'ha generat el dashboard, aquest diagrama està situat a la secció A6 de l'apèndix.

***Logs* desats**

Aquesta secció de l'aplicació, té la funcionalitat de mostrar, els *logs* que s'han detectat com a sospitosos a l'escanejar el fitxer descarregat amb els *logs* dels servidors, en aquesta vista, es mostrarà una taula, en la qual trobarem una petita mostra d'informació, i un botó a prémer.

Aquest botó està implementat, perquè canviï la vista, i ensenyi el detall del log que s'ha pitjat, i poder veure així tota la informació recopilada del log, incloent-hi el missatge.

Es dóna aquest matís del missatge degut a que la descàrrega de *logs* triga 5 vegades més si es porta el missatge, i per tant, es va prendre, com a decisió de disseny, només llençar el recuperador del missatge quan trobem un log especial.

Gestió d'usuaris

L'alta d'usuaris és una eina, on gràcies a un formulari, podem donar d'alta usuaris a la base de dades, per donar-los accés a l'aplicació, podent-ne escollir el perfil per cadascun d'ells.

En aquesta, s'han implementat diverses comprovacions per no provocar redundàncies a la base de dades, per exemple, cada cop s'insereix un usuari a la base de dades, ja que en el moment d'enviar la informació del nou usuari, el servidor, pot respondre que el nom d'accés de l'usuari ja existeix.

A més per decisions de disseny que es van prendre, el perfil administrador, està deshabilitat a la base de dades, per evitar, precisament una assignació indiscriminada de privilegis, quan es creïn usuaris.

Per acabar amb l'eina d'administració d'usuaris, descriurem l'eina de baixa d'usuaris, que implementa, altres funcions que no són només la baixa d'usuaris.

Les funcions que pot fer aquesta opció, són, com el seu no indica donar de baixa usuaris, però a més s'ha implementat, una funcionalitat addicional per canviar contrasenyes als usuaris que ho demanin a l'administrador.

Com a decisió de desenvolupament, es va definir que els usuaris, a diferència dels servidors, no s'eliminessin de la base de dades, sinó que denegarem l'accés, i no podrà fer *LogIn*, i en la casuística que aquest usuari volgués tornar, només hauríem de donar-lo d'alta.

En cas del canvi de contrasenya de l'usuari, farem exactament el mateix pas que a la secció del meu perfil, però, no haurem de posar la contrasenya antiga, així, un usuari podrà demanar el canvi de contrasenya en cas d'oblit.

La vista mostrarà una taula on tenim les dades dels usuaris, i dues columnes amb botó de donar de baixa/alta i un altre per canviar la contrasenya.

Per deixar clar aquest procés d'alta i baixa d'usuari, ho aclariré amb el diagrama de seqüència descrit a la secció de l'annex A9.

LogOut

Aquesta opció, per acabar, permet a l'usuari sortir de l'aplicació i esborrar totes les dades de sessió que estan desades al *SessionStorage* del navegador, un cop finalitzada la sessió, es tornarà a la pàgina on s'hi pot fer *LogIn*, per aclarir millor el procés de *logOut*, s'adjunta un diagrama de seqüència a la secció A10.

4.2.3 Altres consideracions que es volen tractar

Back-end

A continuació es volen explicar algunes funcionalitats addicionals sobre el back-end, i que es creu que són rellevants per deixar de banda en el dossier final del projecte.

El back-end s'ha programat per mòduls, és a dir, aquesta part del projecte, no s'han utilitzat classes, sinó que, hem partit d'un mòdul principal que és el que conté les URI's de flask per accedir al servidor, i des d'aquí cridarem a diferents mòduls, com poden ser comprovar si l'intent d'accés d'un usuari és legítim, o donar l'ordre a un mòdul per descarregar i parsejar els *logs*.

El back-end ha estat desenvolupat amb una visió oberta al futur de l'aplicació, és a dir, l'aplicació s'ha desenvolupat amb el principal objectiu (a banda de ser totalment funcional a la finalització del projecte), de ser el més escalable i robust a canvis possible, això s'explica com, que les funcions i procediments implementats, no només estan pensats per les característiques que té en el moment de la finalització, sinó, que si s'estén la funcionalitat en futures versions, sigui el més fàcil possible.

Un exemple del que s'ha tractat en el paràgraf anterior és que, per llegir els fitxers de *logs* i analitzar-los, als mètodes encarregats passarem, el directori on estan continguts els fitxers, això farà que, a l'hora d'introduir noves entitats com aplicacions, no s'haurà de fer un canvi en aquest aspecte, ja que recuperarem el fitxer i l'analitzarem sense diferenciar entre entitats.

D'igual manera, s'ha emprat la reutilització de funcions programades pel projecte, com poden ser les que accedeixen a la base de dades i recuperen dades, ja que es considerava excessiu fer un mètode d'accés per a cada tipus de dades a recuperar o inserir, es va decidir fer un "model" sobre el que accedirem a la base de dades.

A més, s'ha implementat al back-end, un logging dels esdeveniments que succeeixen al servidor, així podem tenir un fitxer *.log*, per saber que passa en tot moment en el servidor, i poder detectar ràpidament fallades, ja que

deixarien informació a aquest fitxer.

Front-end

Pel que fa a les funcionalitats addicionals del front-end, les comentem a continuació, ja que es creu que són importants.

En primer lloc, s'ha seguit una programació modular, ja que cada controlador d'angularJS s'ha desat en un fitxer diferent, dividint les funcionalitats de l'aplicació en diferents controladors. També cal destacar que s'ha seguit una programació orientada a events, utilitzant els events d'angular i una estructura de controladors jeràrquica.

Es vol destacar, que l'aplicació està enterament realitzada en castellà, a més, en relació a l'aspecte de l'aplicació s'han utilitzat les icones d'HTML, on podem incloure al front-end icones per fer la interfície més interactiva.

S'ha de dir que en cada crida al back-end, excepte l'inici de sessió, s'envia sempre el token de sessió per validar la identitat a l'accedir a les dades.

A més, s'ha treballat amb bootstrap perquè l'aspecte de l'aplicació, canviï segons la mida del dispositiu en el que s'estigui visualitzant, per a l'aplicació s'ha volgut pendre partit d'aquesta tecnologia, amb la seva tècnica del grid de Bootstrap.

5 DISCUSSIÓ I QUALITAT DELS RESULTATS

5.1 Resultat final, presentació de resultats

Al final del desenvolupament del projecte, s'ha aconseguit una aplicació completa d'anàlisi de *logs* en Windows.

S'aconsegueix una estructura d'API REST on el client consumeix aquesta API, i extreu la informació que necessita del back-end.

S'ha desenvolupat un front-end amb AngularJS, HTML i Bootstrap, amb una interfície user friendly, aquesta aplicació és capaç d'interactuar amb el servidor implementat en el framework de python Flask, per satisfer la llista de requeriments proposats en l'inici d'aquest mateix projecte i estesos durant les diferents setmanes.

El client es comunica amb el servidor mitjançant peticions POST, amb les que provoca accions al servidor, per o bé, aconseguir informació de la base de dades per omplir formularis, o també, per analitzar, o fer una gestió de les dades que hi han desades a la base de dades.

El servidor és capaç per si sol, de fer una anàlisi proactiva dels *logs* de tots els servidors que tingui introduïts en el sistema, i així mantenir una anàlisi constant dels *logs* i no perdre dades en l'anàlisi dels *logs*, a més mitjançant un sistema d'alertes, es podrà fer un seguiment més exhaustiu dels *logs* que hi ha als sistemes de l'empresa

S'implementa una gestió de dades des del frontend per facilitar la feina dins del sistema a l'administrador, aquestes implementacions són alta i baixa d'usuaris, i alta i baixa de servidors.

A més es poden consultar en detall la informació dels *logs* especials que estan desats a la base de dades, en cas d'arribar una alerta.

S'implementen tant al servidor com al client, mecanismes com JWT, certificat HTTPS o variables de sessió, per garantir una comunicació segura entre client i servidor, i on aquestes comunicacions són autèntiques i fiables.

S'implementa un sistema dinàmic, on les dades no depenen de l'estructura de la base de dades, gràcies a això s'ha aconseguit un sistema de BD, robust i sense redundàncies.

S'aconsegueix fer una anàlisi basada en el back-end dels *logs*, on després és explotat per gràfiques interactives, al fornt-end, sense carregar de tràfic la xarxa al només passar JSON en comptes de fitxers sencers.

S'estableix al client una estructura òptima de controladors d'AngularJS, jeràrquica, on es centra en les variables d'$*scope*, i en la programació orientada a events.

Capacitat de generar Reports de dashboard que es visualitza per pantalla, per així guardar el resultat, o per a futures comparacions o per reportar algun problema.

S'implementa un logging de la mateixa aplicació, així es pot tenir un registre d'events de la mateixa i fer-ne un seguiment en cas de falla.

S'implementa un sistema amb el qual, es poden definir directives per desar *logs* especials a la base de dades diferenciant-los de la resta al parsejar els *logs* del servidor, amb aquesta característica, s'implementa un sistema de visualització d'aquests.

5.2 Discussió de resultats

A causa de l'estudi previ que es va fer a l'inici del projecte, es podria dir que hem realitzat una versió, d'un estàndard d'anàlisi de *logs* per a Windows, com quasi no hi ha al mercat, ja que les poques que hi ha, són migrades des de Linux com Win2Ban[16] és una migració de Fail2Ban.

Aquest estàndard d'anàlisi de *logs* és totalment funcional respecte a altes estàndards d'anàlisi que podem trobar a internet. Podria ser, i aquest projecte s'ha fet amb la intenció de ser-ho, un sistema de monitoració, que pugui ser implantat en l'entorn d'una companyia, i sigui totalment fiable.

Respecte a coneixements previs, podríem dir que al fer un sistema client/servidor, hem tractat una en més o menys grau algunes àrees de la informàtica.

En primer lloc, en aquest projecte s'ha tractat la seguretat informàtica, implementant mecanismes criptogràfics per gestionar les peticions al servidor i el control d'accés. A més, s'ha implementat un certificat autosignat per xifrar les connexions entre client i servidor, per fer-les segures i no revelar informació en les comunicacions. Tota aquesta part estan reforçats pels coneixements de tecnologies de la informació i les comunicacions.

A més s'ha utilitzat la criptografia per desar les contrasenyes dels usuaris a la base de dades, amb l'algoritme SHA-256, per no tenir les contrasenyes en clar a la base de dades. També s'ha implementat un sistema

de perfils on es té més o menys accessos a l'aplicació segons el perfil que tingui assignat l'usuari.

En segon lloc, s'han aplicat tècniques i coneixements pròpies de la gestió i administració de bases de dades, ja sigui creant les taules, els usuaris de la base de dades, o fins i tot, optimitzant les taules perquè la base de dades pugui créixer dinàmicament, i sense fer créixer la base de dades en una gran quantitat. A més les taules estan optimitzades perquè les consultes no siguin massa pesades.

En tercer lloc, s'han implantat tècniques de desenvolupament de software per desenvolupar, una aplicació com un desenvolupament FullStack, i desenvolupant el servidor per mòduls, on siguin fàcilment reemplaçables si es necessita un canvi a alguna part de l'aplicació.

En quart lloc, s'han implementat coneixements de desenvolupament web, amb tecnologies actuals i molt populars com poden ser AngularJS, i Bootstrap, a més de fer la integració de Google Charts, per aconseguir, una aplicació web, d'una sola pàgina, totalment funcional i sense cap falla.

Per aconseguir una comprovació del bon funcionament de l'aplicació, podem esmentar que el test funcional exhaustiu que s'ha passat amb èxit, provant els *edge* i els *corner cases,* provant els valors màxims i mínims a l'aplicació com a *edge cases*, i provant valors fora de rang per als *corner cases,* totes les proves han donat resultats positius per a la conclusió del test funcional i el tancament del desenvolupament.

Per últim, s'han fet servir coneixements de desenvolupament en python, per implementar tots els algoritmes més importants al servidor, per poder extreure tota la informació dels servidors i servir-la al client.

Des del meu punt de vista, aquest projecte, és perfectament assolible per un enginyer informàtic, ja que s'ha desenvolupat una aplicació *FullStack*, on, tant el backend com el front-end, han estat implementats des de zero.

6 Conclusió

En resum, el TFG dona una eina de valor a l'empresa perquè pugui utilitzar per tenir un control més estricte dels *logs* dels diferents servidors d'aquesta.

S'ha acabat culminant el projecte del TFG, amb un sistema, que compta amb un client, realitzat amb AngularJS, que consumeix una API REST, construïda al servidor, aquesta, respon en forma de JSON a les peticions.

El servidor és totalment autònom, ja que és capaç de realitzar una anàlisi proactiva, dels *logs* dels servidors, i desar-los a la base de dades, esperant que siguin consultades.

L'aplicació, per la seva part, serà la part que interactuarà, amb l'usuari, i que permetrà al mateix fer una gestió de l'aplicació, en funció del seu perfil, i accedir a certs continguts, restringits a altres usuaris.

En l'aplicació es poden generar diferents dashboards per poder fer una anàlisi completa i aconseguir un tractament de la informació, fàcil d'interpretar i de produir.

Val a dir també que l'empresa, en farà ús d'aquesta aplicació, ja que és una proposta de l'empresa, i comporta una millora substancial del control d'esdeveniments dels servidors de l'empresa.

Per desgràcia, hi ha objectius que no s'han pogut assolir, en concret un objectiu, el que proposava fer l'aplicació robusta i compatible, amb qualsevol sistema operatiu de l'empresa, i que fos transparent per l'usuari, de quin servidor estan consultant els *logs*.

Un altre Objectiu que no s'ha assolit ha estat l'anàlisi dels *logs* dels Firewall, ja que com es va comentar al principi del projecte, aquest anava a ser canviat a inicis del 2019, i per limitacions de temps, no es podia implementar aquesta millora al projecte.

Elements no tractats que haguessin estat bé tractar haguessin estat, per exemple, fer una espècie de Wiki, per poder, mantenir una informació, on, per cada tipus de *log* tenim una resolució típica, per poder aplicar-ne una solució ràpida en cas d'incidència crítica.

L'empresa, com a impulsora del projecte juntament amb mi té interès a utilitzar i integrar l'eina al marc operacional de l'empresa, això es farà amb facilitat, ja que l'aplicació s'ha desenvolupat sota el marc de l'empresa, especialment dissenyada per a l'empresa

Els principals plans de futur, i línies de millora, pel que respecta al projecte, seria, estendre l'aplicació a tots els sistemes i fer l'aplicació, o el sistema, un sistema multiplataforma d'estàndard de *logs*, on estigués recolzada en Windows i en Linux majoritàriament.

Altres línies de millora, serien implementar al projecte en un futur, aplicacions o el mateix Firewall, on s'analitzin els *logs* de tots ells, i es pugui tenir un control total dels esdeveniments que succeeixen a l'empresa i ser un sistema prou competitiu, com perquè estigui inclòs en el programari bàsic dels servidors, a més de poder fer modificacions a l'arquitectura perquè els servidors de l'empresa avisin quan s'ha produït un *log* especial. Per últim val a dir que l'aplicació està adaptada per fer el canvi de llenguatge, fent ús d'un *lenguage service* d'AngularJS, per fer l'aplicació multillenguatge.

Agraïments

M'agradaria agrair el paper que ha pres l'empresa en el desenvolupament del projecte, sobretot agrair a la Bea, a en Xavi, a l'Oscar, i a en Moi per l'ajuda que m'han donat i el seu suport. També m'agradaria agrair als meus pares i a l'Ainoa per la paciència que han tingut amb mi i el suport que m'han donat.

Bibliografia

[1] Flask (Version 1.0.2) [Computer Software]. (2010). Recuperat de http://flask.pocoo.org

[2] PostgreSQL (10.5) [Datebase management system] (2018) Recuperat de https://www.postgresql.org/

[3] AngularJS (Version 1.6.8) [Computer Software]. (2017). Recuperat de https://angularjs.org

[4] BootsTrap (Version 4.0) [Computer Software]. (2018). Recuperat de https://getbootstrap.com
[5] Google Charts (Version 46) [Computer Software]. (2018). Recuperat de https://developers.google.com/chart/
[6] Fail2Ban (Version 0.9.3) [Computer Software]. (2015). Recuperat de https://www.fail2ban.org/wiki/index.php/Main_Page
[7] DenyHosts (Version 2.6) [Computer Software]. (2008). Recuperat de http://denyhosts.sourceforge.net/
[8] Win2Ban (Version 1.1.0) [Computer Software]. (2015). Recuperat de: https://itefix.net/win2ban
[9] Riffat Naz, M. N. A. Khan. "Rapid Applications Development Techniques: A Critical Review", International Journal of Software Engineering and Its Applications, Vol 9, pp. 12-13, 2015.
[10] Mahil Carr, Dr. June Verner. "Prototyping and Software Development Approaches", University of Hong Kong, pp. 4-6, 1997.
[11] Maximiliano Cristiá. "Introducción al Testing de Software", Universidad Nacional de Rosario, pp. 7, 2009.
[12] Vinod Kumar Chauhan. "Smoke Testing", International Journal of Scientific and Research Publications, ISSN: 2250-3153, pp. 1-2, 2014.
[13] Wingu, Manual de metodos Ágiles, www.wingu.org, pp. 17, 2016
[14] Trello (Version 2018) [Computer Software]. (2010). Visitar http://trello.com
[15] JSON Web Tokens (Version 1.0.1) [Open Standard RFC 7519]. (2018). Recuperat de https://jwt.io/

APÈNDIX

En aquest apèndix, s'inclourèn els diagràmes de seqüència, i figures que es consideren importants.

A1. Diagrama descàrrega de logs

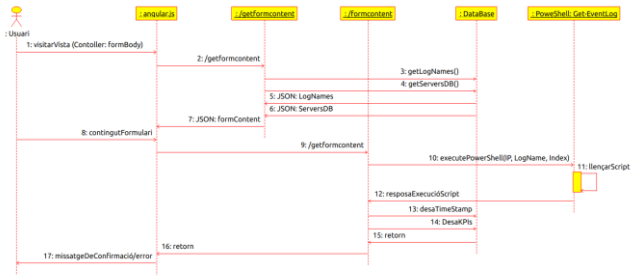

Fig. 2. Diagrama de seqüència de la descàrrega de logs

A2. Diagrama d'alta de servidors

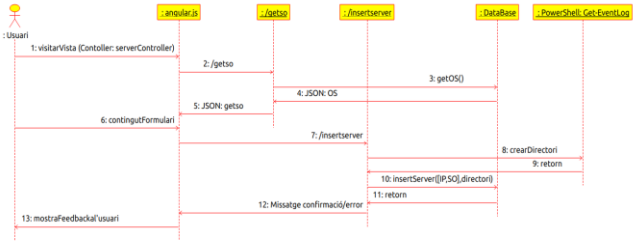

Fig. 3. Diagrama de seqüència d'alta de servidors

A3. Diagrama baixa de servidors

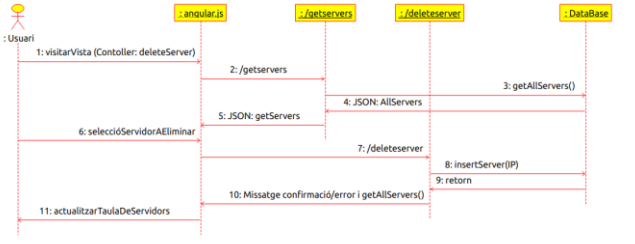

Fig. 4. Diagrama de seqüència de baixa de servidors

A4. Diagrama dashboard diari

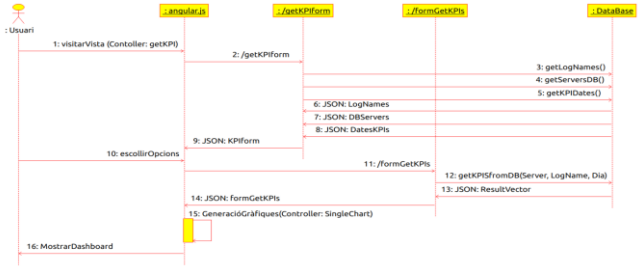

Fig. 5. Diagrama de seqüència generació dashboard diari

A5. Diagrama dashboard general

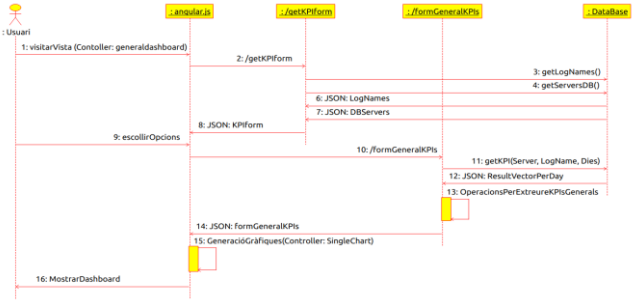

Fig. 6. Diagrama de seqüència generació dashboard general

A6. Diagrama generació informe

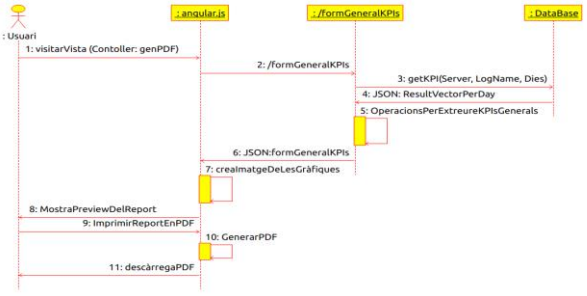

Fig. 7. Diagrama de seqüència generació informe del dashboard

A7. Diagrama logs especials

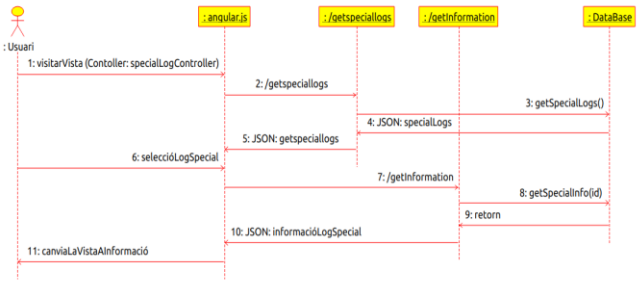

Fig. 8. Diagrama de seqüència recuperació logs desats

A8. Diagrama d'alta d'usuaris

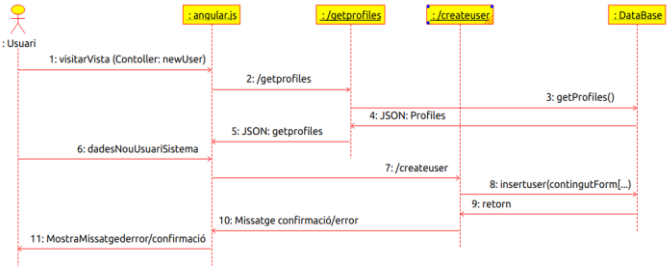

Fig. 9. Diagrama de seqüència d'alta d'usuaris

A9. DIAGRAMA BAIXA D'USUARIS

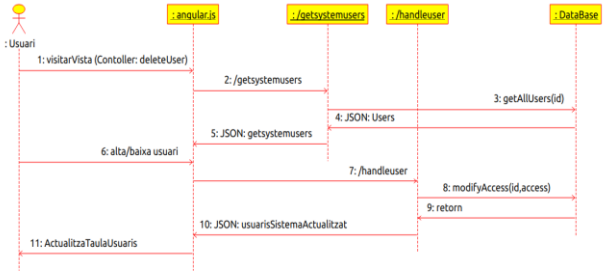

Fig. 10. Diagrama de seqüència de baixa d'usuaris

A10. DIAGRAMA DE LOGOUT

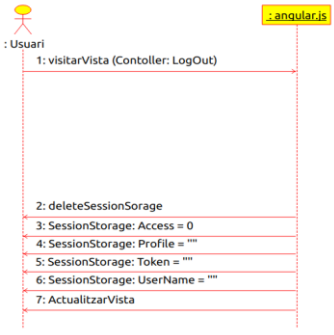

Fig. 11. Diagrama de seqüència del tancament de sessió

A11. CERTIFICAT GENERAT

Fig. 12. Certificat autosignat per iniciar el servidor amb HTTPS

A12. DIAGRAMA DE LA BASE DE DADES

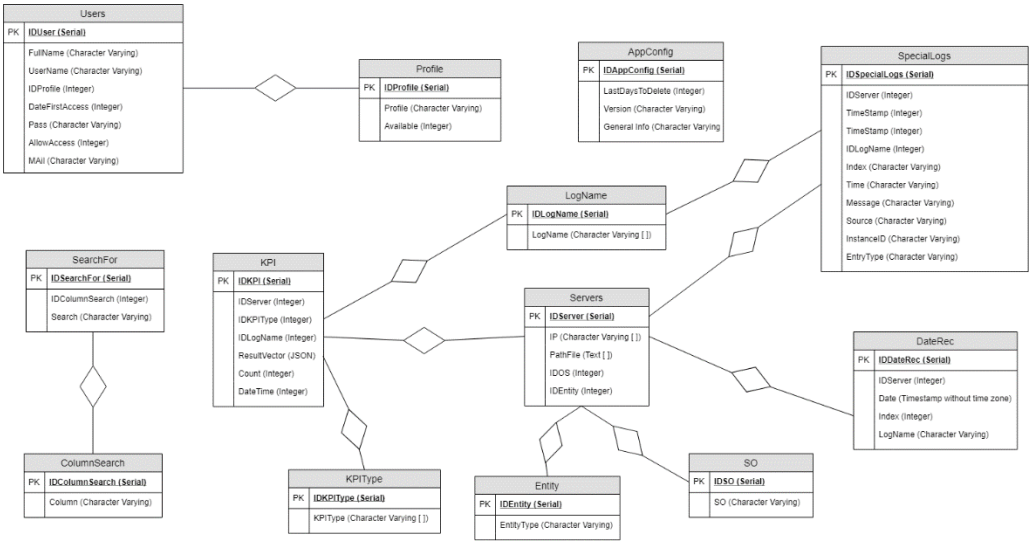

Fig. 13. Diagrama de la base de dades del projecte