Disseny i Desenvolupament d'una Aplicació de Crowd Sensing (Back End)

Martín de Jong

Resum– Aquest projecte consisteix en el disseny i desenvolupament d'una aplicació web genèrica que serveix per enviar alertes d'incidents a la carretera o relacionats amb la seguretat ciutadana de manera anònima. L'aplicació pot utilitzar Internet o xarxes oportunistes per a transmetre aquests incidents. A més, està dissenyada de manera que es pot adaptar fàcilment a diferents usos, a banda dels dos mencionats anteriorment.

Paraules clau– Desenvolupament Web, Back-end, Python, Flask, DTN, PostgreSQL, Crowd-Sensing.

Abstract– This project consists in the design and development of a generic web application to send reports of road or citizen security incidents. To transmit those incidents, it is able to use the Internet or opportunistic networks. Also, it is designed to be easily adaptable to multiple use cases besides the two mentioned before.

Keywords– Web Development, Back-end, Python, Flask, DTN, PostgreSQL, Crowd-Sensing.

1 Introducció

Des que es van popularitzar els telèfons intel·ligents, han anat adquirint un espai en la generació de tràfic a la xarxa que avui en dia segueix en creixement. A l'any 2017, un 63% de les visites a pàgines web es van fer a traves de dispositius mòbils [1]. A més, des del 2015 fins avui en dia el tràfic web generat pels dispositius mòbils ha augmentat del 31.16% al 51.89% [2]. Tenint en compte aquestes dades, veiem que les aplicacions pels dispositius mòbils són cada dia més rellevants i les tendències ens mostren que en el futur ho seran encara més.

L'augment de l'ús d'*Internet* a dispositius mòbils, però, té alguns inconvenients. A diferència de dispositius estàtics com els ordinadors de sobretaula, la comunicació dels telèfons mòbils depèn d'una infraestructura sense cables que no té cobertura a tots llocs. A més, una companyia que proveeix de serveis d'*Internet* pot localitzar la posició d'un dispositiu constantment, la qual cosa podria suposar una invasió de la privacitat de l'usuari.

Al projecte de recerca *Crowd eAssessment* [3] s'havia implementat un servei web que tenia accés mitjançant tant *Internet* com xarxes oportunistes, en concret, *Delay and Disruption Tolerant Networks (DTN)* [4]. Gràcies a això, aquest servei podia rebre incidències de col·legis electorals sense utilitzar *Internet* i evitant així alguns problemes de cobertura i possibles seguiments per part dels proveïdors de servei de xarxa. No obstant, aquell desenvolupament tenia algunes mancances degut a que s'havia implementat específicament per aquell projecte.

Aquest projecte pretén adaptar aquella aplicació [3] per fer-la més genèrica, de manera que amb el mateix servei es puguin fer diferents aplicacions amb usos diversos. La implementació que es vol realitzar actualment contempla dos casos: incidents a la carretera com embussos o accidents, i incidents relacionats amb la seguretat ciutadana com atracaments o vandalisme.

Així doncs, aquest treball se centrarà en el *back-end* de l'aplicació, que s'ha desenvolupat juntament amb un *front-end* que correspon a un altre treball.

Tenint això en compte, els objectius establerts per aquest projecte són els següents, per ordre de prioritat:

1. Dissenyar un servei web genèric. La idea principal d'aquest projecte és fer genèric un servei que fins ara estava fet per una aplicació concreta [3].

2. Que el servei permeti rebre incidències, processar-les i emmagatzemar-les.

3. Comprovar que la incidència prové d'un usuari legítim. L'aplicació només permet reportar incidències

• E-mail de contacte: martin.dejong@e-campus.uab.cat
• Mención realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Sergi Robles Martínez (Departament d'Enginyeria de la Informació i Comunicacions)
• Curs 2018/19

a usuaris autoritzats, tot i que es faci de manera anònima.

4. Rebre validacions de les incidències i mantenir una puntuació sobre la veracitat de les últimes.

5. Permetre obtenir informació sobre els incidents. Aquest objectiu és especialment necessari per poder fer aplicacions que utilitzin el servei.

6. Proveir un mecanisme que permeti gestionar un conjunt d'usuaris de l'aplicació i que aquests puguin reportar incidències de manera anònima. Com s'ha dit anteriorment, només poden reportar incidents persones autoritzades. Per assolir això, s'han de poder gestionar els usuaris perquè es puguin autenticar les seves incidències sense revelar la seva identitat.

7. Permetre que aquest servei web funcioni a través d'*Internet* i de *DTN* [4]. Com que *Internet* depèn d'una infraestructura externa, això comporta possibles problemes de monitorització que podrien fer que es perdés l'anonimat. A més, es depèn d'aquesta infraestructura per poder reportar incidents.

8. Validar que els servei web s'adapta correctament a diferents aplicacions de report d'incidents.

9. Comprovar que l'aplicació funciona correctament amb les xarxes oportunistes [4].

Per tal d'assolir aquests objectius, s'ha estructurat aquest projecte en dos serveis: Un que reculli i gestioni els incidents reportats, i un altre que gestioni els usuaris que reporten incidents. Amb aquesta estructura, els usuaris de l'aplicació s'identificaran amb pseudònims que només coneixerà el gestor d'usuaris.

Així doncs, el gestor d'incidents no podrà saber quin usuari està reportant la incidència, i l'únic que podrà fer és demanar al gestor d'usuaris si el pseudònim és vàlid. D'altra banda, el gestor d'usuaris no tindrà accés als incidents reportats, només a les relacions entre els usuaris i els pseudònims.

En aquest article es farà una breu explicació de l'estat de l'art. A continuació, s'exposarà breument la metodologia utilitzada al desenvolupament. Després, s'enumerarà la captura dels requisits de tots els projectes. Seguidament, es procedirà a explicar el disseny, la implementació i les proves per a cadascun dels serveis i s'analitzaran els resultats obtinguts. Finalment, s'exposaran les conclusions finals del treball.

2 ESTAT DE L'ART

En aquest apartat s'explicarà l'estat de l'art de l'àmbit del projecte. Principalment, s'analitzarà el projecte [3] que ha servit de punt de partida pel treball actual. A més, s'analitzaran les tecnologies actuals en quant al desenvolupament d'aplicacions web.

Actualment hi ha una manca d'aplicacions que aprofitin les possibilitats que ofereixen les xarxes oportunistes. El projecte [3] consistia en una aplicació *Android* i dos serveis que utilitzaven la llibreria *aDTNPlus* per tal de reportar incidents en processos electorals utilitzant aquestes xarxes. A més, aquesta llibreria està instal·lada en un conjunt de nodes que serveixen per transmetre els *Bundles* amb la informació. Aquests nodes són *Raspberry Pis* amb una imatge *Linux* preconfigurada i emeten una xarxa *Wi-Fi* amb un *SSID* concret, que és on es connecta l'aplicació *Android* per enviar missatges.

La idea d'aquell projecte era que amb un dispositiu mòbil es reportessin les incidències i, quan aquest dispositiu entrés en el rang d'algun node, li enviés el missatge. Un cop enviat, aquest es transmet de node a node per la *DTN* [4] fins arribar al gestor d'incidents, que la tracta i la desa a la seva base de dades.

D'altra banda, cada dia s'està fent més popular l'ús de *frameworks* que permeten fer aplicacions web d'una sola pàgina i que reben les dades d'*APIs REST*. Aquest esquema és actualment el més utilitzat per desenvolupar aplicacions ja que ofereix una experiència d'usuari més fluida respecte a les webs de múltiple pàgina. A més, aquest esquema permet reduir la càrrega dels servidors traslladant part de la lògica de les webs als clients. Finalment, també permeten fer diferents pàgines web que consumeixin les mateixes dades, per la qual es poden fer serveis més genèrics.

Així doncs, s'ha analitzat l'estat de l'art actual. S'ha vist com està el projecte en el qual es basa l'aplicació actual i s'ha revisat l'estat d'algunes tecnologies actuals per fer desenvolupament web.

3 METODOLOGIA

En aquest apartat s'explicarà la metodologia utilitzada pel desenvolupament del projecte i s'explicarà com s'ha aplicat al projecte.

El model de desenvolupament utilitzat és un model iteratiu, ja que es van fent petites parts de l'aplicació i avaluant-les constantment. Això es fa per poder analitzar en cada moment del desenvolupament possibles canvis que facilitin la implementació, que puguin tenir en compte nous requisits o per detectar errors de disseny o d'implementació aviat i evitar perdre molt de temps arreglant-los.

Seguint aquesta metodologia iterativa, s'han fet reunions periòdiques per decidir com fer els models de dades, quines tecnologies utilitzar i com implementar cada funcionalitat nova. A més, s'han anat refinant detalls de l'aplicació a mesura que s'anaven implementant, en comptes de fer un disseny fix i desenvolupar-lo sense tenir en compte possibles canvis de requisits. Finalment, al provar les funcionalitats a mesura que es van implementant s'han pogut detectar errors de disseny abans de tenir el projecte molt avançat.

Per a aquest projecte, la primera acció que s'ha fet és plantejar el problema a resoldre i captar els requisits que ha d'assolir el back-end de l'aplicació. Aquests requisits no eren fixes i podien variar en el temps en funció de l'avenç del projecte.

Després, per a cada servei web, s'han seguit els procediments següents per desenvolupar-los:

Inicialment, s'han captat els requisits funcionals específics de cada servei.

Un cop captats, s'ha dissenyat el model de dades necessari per poder oferir cada servei i s'ha generat un diagrama Entitat-Relació (ER).

Amb els requisits i els models de dades establerts, s'han triat les tecnologies necessàries per implementar el servei.

Seguidament, s'han implementat els serveis web que per a satisfer els requisits establerts. Aquesta implementació s'ha de manera gradual i s'han anat provant les noves funcionalitats a mesura que s'han implementat.

Un cop desenvolupats els serveis, s'ha iniciat una fase de proves d'integració dels serveis. En aquesta fase s'ha provat que el servei s'adapti a les diferents aplicacions dissenyades per reportar incidents i que permet la seva connectivitat per *DTN* [4].

Com s'ha vist, s'ha exposat la metodologia seguida pel desenvolupament del projecte i s'ha explicat com s'ha aplicat al projecte.

4 REQUISITS

En aquest apartat s'explicarà tot el procés de captura de requisits de l'aplicació, tant els globals com els específics de cada servei.

Inicialment, s'ha plantejat el problema a resoldre, que consisteix en fer dos serveis web genèrics que permetin crear aplicacions de report i validació d'incidències anònimes. Els actors que s'han identificat són els següents: l'usuari, que reporta incidències de manera anònima, valida incidències d'altres usuaris i les ha de poder veure a les aplicacions; el gestor d'incidències, que té un registre de totes les incidències i la seva puntuació; i el gestor d'usuaris, que coneix tots els usuaris i els seus pseudònims, però no les incidències que reporten.

Després, s'han capturat els requisits globals del projecte. Per fer-ho, s'ha analitzat les accions que han de poder fer tots els actors i s'ha establert que son els següents:

- S'han de poder reportar incidents de manera anònima.
- S'han de poder validar aquests incidents.
- Ha de ser un servei genèric per a qualsevol tipus d'aplicació.
- S'ha de poder verificar que un incident pertany a un usuari vàlid.
- S'ha de mantenir una puntuació dels incidents en funció de les validacions.
- S'ha de proveir informació dels incidents i dels usuaris a les APIs.
- Ha de funcionar mitjançant *DTN* [4] i *Internet*.

Un cop captats els requisits globals, s'han captat els requisits específics del gestor d'incidents i del gestor d'usuaris.

Pel gestor d'incidents, aquests requisits són els següents:

- Ha de poder emmagatzemar els incidents que rep.
- S'han de poder organitzar els incidents en categories i subcategories.
- Ha de poder verificar que aquests incidents procedeixen d'un usuari vàlid mitjançant el gestor d'usuaris.
- Ha de rebre validacions d'incidents, autenticar-les (igual que els incidents), i mantenir una puntuació de cada incident.
- Ha d'exposar informació dels incidents i la seva puntuació.
- Ha d'exposar informació filtrada d'incidents (data de report, zona, cerques, etc.).
- Ha d'exposar informació de les agregacions estadístiques de les dades.

D'altra banda, pel gestor d'usuaris, es tenen els següents requisits:

- Ha de poder autenticar usuaris i identificar-los amb rols.
- Ha de poder generar pseudònims i emmagatzemar-los.
- Ha de poder rebre pseudònims i validar que pertanyen a un usuari real.
- Ha d'exposar informació sobre els usuaris als administradors.

Com es pot veure, s'han captat els requisits globals del projecte i aquells més concrets dels dos gestors.

5 GESTOR D'INCIDENTS

El sistema desenvolupat es composa de dos gestors (Fig. 3). En aquest apartat s'exposarà el procés de desenvolupament del gestor d'incidents i en l'apartat següent el del gestor d'usuaris.

Primerament, s'exposarà el disseny dels models de dades d'aquest servei i el de l'aplicació que l'ofereix. Un cop fet això, s'explicarà la seva implementació i les tecnologies utilitzades.

5.1 Disseny

En aquest apartat s'explicarà i justificarà el model de dades a triar.

Com es pot observar a la figura (Fig. 1), el gestor d'incidents té quatre models de dades.

El model principal del gestor és el d'incidents i s'encarrega d'emmagatzemar tota la informació donada quan es reporta un incident. En aquesta taula s'emmagatzemen dades de l'incident, com la causa, la localització, una descripció, etc. A més, es poden adjuntar imatges o vídeos en el camp "*media*". També consta de camps que serveixen per identificar la persona que l'ha reportat. Finalment, pot tenir camps "*extra*"en cas que una aplicació els necessiti.

Relacionat amb aquest primer model, hi ha el de validacions, que és la que guarda totes les validacions relacionades amb un incident i és la que manté una puntuació d'aquest.

Finalment, hi ha els models de categories i subcategories, que serveixen per classificar els diferents incidents i variaran en funció de l'aplicació que els utilitzi.

Gràcies a que el model d'incidents és flexible i que es classifiquen per categories dinàmiques es poden crear aplicacions de report d'incidents diferents que facin servir instàncies personalitzades del mateix back-end.

Seguidament, s'explicarà com s'ha realitzat el disseny de l'aplicació i quins factors s'han tingut en compte.

Per tal de fer el servei fàcil de mantenir i extensible, s'ha optat per un disseny modular dels *endpoints*. Així doncs,

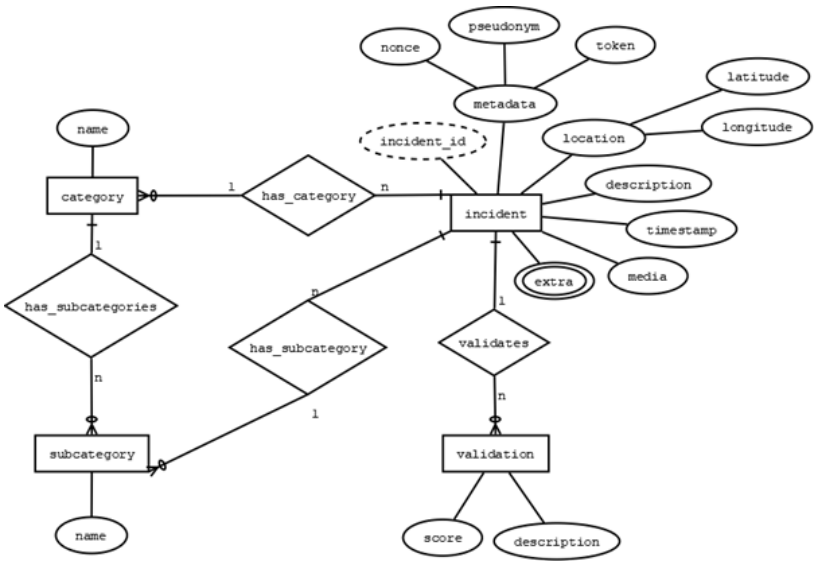


Fig. 1: Model de dades del gestor d'incidents

s'han agrupat les funcionalitats que ha d'oferir en mòduls independents i s'ha seguit una estructura comú. D'aquesta manera, es pot navegar fàcilment pel projecte i trobar les parts que es volen modificar o estendre.

A més, s'ha seguit una estructura semblant a la *Model View Controller (MVC)*, separant els controladors dels models, per tal de tenir separada la part de gestió de models de dades de l'aplicació de la de funcionalitats.

També hi ha un mòdul extra que conté les utilitats i constants de l'aplicació, que s'utilitzen en diferents mòduls.

Finalment, l'últim mòdul és un *worker* que s'inicia al mateix temps que l'aplicació i que capta els reports d'incidents i validacions que venen per la *DTN* [4].

5.2 Implementació

A la secció següent s'exposaran els detalls de la implementació del gestor d'incidents. Primer de tot, s'enumeraran i justificaran les tecnologies utilitzades. Un cop explicat això, es procedirà a detallar la implementació del servei.

Les tecnologies utilitzades per implementar aquest gestor han sigut les següents:

S'ha triat *Flask* [5] com a *framework* de l'aplicació per aprofitar part del codi del projecte *Crowd eAssessment* [3], sobretot la implementació del *socket* de *DTN* [4].

Pel mateix motiu s'ha mantingut el sistema gestor de bases de dades *PostgreSQL* [7]. No obstant, un altre motiu per mantenir-lo és que és un sistema molt complet i ens permetia guardar les dades en els formats que necessitàvem (com, per exemple, guardar *UUIDs* [8] pels models).

A més, s'ha utilitzat *SQLAlchemy* [6] per gestionar els models de dades i fer de connector entre l'aplicació i la base de dades. Entre altres funcionalitats, aquest *framework* simplifica les consultes i gestiona les connexions simultànies a base de dades.

Finalment, s'ha utilitzat la llibreria *aDTNPlus* [3] proporcionada pel departament d'Enginyeria de la Informació i Telecomunicacions per tal de gestionar aquelles peticions que provenen de nodes oportunistes.

A continuació, s'explicarà la implementació del servei que gestiona els incidents:

Els models de l'aplicació consisteixen en classes de *SQLAlchemy* [6] que representen les diferents taules del model de dades. La idea és que no implementin cap altre tipus de lògica que no sigui afegir, llegir, modificar i eliminar elements de cada model.

D'altra banda, els controladors implementen la lògica del servei web i preparen les dades per guardar-les a la base de dades, o per donar-les al públic. Els controladors que hi ha actualment es corresponen tots a un *endpoint* diferent del servei web. Per tal de fer l'aplicació modular, s'han agrupat els *endpoints* en diferents *blueprints* de *Flask*.

El primer mòdul i el més important és el d'incidents. Aquest permet afegir els incidents que arriben i emmagatzemar-los. A més, ofereix la possibilitat d'obtenir una llista paginada d'incidents (amb informació reduïda) o bé el detall complet d'un incident en concret mitjançant la seva *ID*. També permet filtrar aquests incidents en funció de l'antiguitat o de si han estat validats. Finalment, té un *endpoint* per fer una cerca senzilla d'incidents.

A més, es té el mòdul de categories que simplement permet llistar les categories i subcategories de l'aplicació.

El mòdul de validacions és el que ofereix el servei de rebre validacions i actualitzar la puntuació d'un incident concret.

També hi ha el mòdul d'estadístiques que exposa el conjunt d'agregacions següent: Nombre de incidents confirmats, incidents agrupats per temps, incidents agrupats per zona. A més, totes aquestes agregacions es filtren per categoria.

Finalment, com s'ha dit a l'apartat de disseny (Sec. 5.1), hi ha un altre mòdul que gestiona les peticions que provenen de les xarxes oportunistes. Aquest mòdul només ofereix dos de les funcionalitats de l'aplicació: reportar incidents i validacions. El que s'ha implementat és un *worker* que està escoltant a la xarxa oportunista amb un *socket* de la llibreria *aDTNPlus* [3] i que, quan rep peticions, les reenvia a l'aplicació i es tracten de la mateixa manera que els incidents normals.

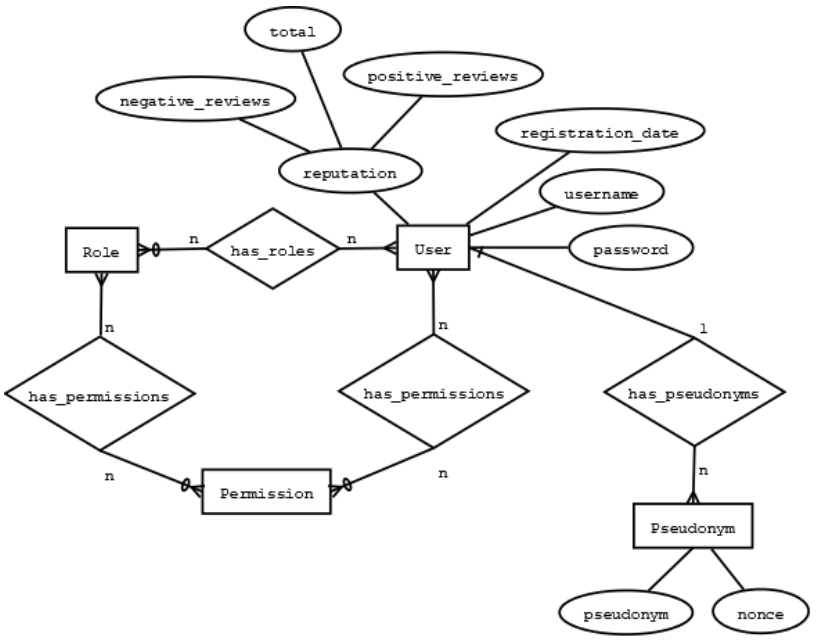


Fig. 2: Model de dades del gestor d'usuaris

6 GESTOR D'USUARIS

En aquest apartat es repetirà l'estructura que s'ha fet a la secció anterior però amb el gestor d'usuaris.

6.1 Disseny

Inicialment, tal i com es va fer amb l'altre gestor, es procedirà a explicar el model de dades utilitzat.

Per implementar el model del gestor d'usuaris s'ha seguit el model que es veu a la figura anterior (Fig. 2). Com podem veure, hi ha quatre models de dades:

El primer model és el d'usuaris (*User*). Aquest conté els camps de *"username"* i *"password"* per permetre als usuaris registrar-se i autenticar-se. També conté el camp de *"registration date"* que permet saber quan s'ha registrat l'usuari. A més, pot tenir rols (que agrupen permisos) i permisos que determinen que pot fer. Cada usuari té una llista de cinc pseudònims que poden utilitzar per reportar incidències de manera anònima. Finalment, conté camps de puntuació que permeten mesurar la fiabilitat de cada usuari.

El model de permisos (*Permission*) representa una acció (o conjunt d'accions) que pot fer un usuari que el posseeixi. Per exemple, el permís d'editar usuaris permet modificar els camps d'altres usuaris a la base de dades, així com eliminarlos.

A més, hi ha el model de rols (*Role*), que representa els rols que poden tenir els usuaris. La funcionalitat d'aquests és agrupar permisos de manera lògica. Per exemple, un rol d'administrador podria tenir permisos per editar usuaris o per eliminar incidències. Amb aquesta estratègia, només s'ha d'afegir el rol d'administrador a un usuari en comptes de donar-li els dos permisos descrits anteriorment, tenint en compte que un administrador sempre ha de poder fer aquelles dues accions.

Finalment, es té el model de pseudònims (*Pseudonym*), que conté els pseudònims utilitzats al gestor d'incidents per tal de reportar incidents de manera anònima. Cada pseudònim està format per un *"pseudonym"* i una *"nonce"* que s'utilitzen validar que una incidència s'ha reportat per un usuari vàlid.

A més, hi ha altres taules auxiliars que s'utilitzen per fer totes les relacions *"N to N"*. Aquestes relacions són *"User-Permission"*, ja que un usuari pot tenir molts permisos i un permís pot pertànyer a varis usuaris. També s'han afegit taules auxiliars a les relacions *"User-Role"* i *"Role-Permission"* pels mateixos motius que *"User-Permission"*.

Per acabar, respecte al disseny de l'aplicació, s'ha seguit la mateixa estructura que al gestor d'incidents. No obstant, aquest servei no té cap *endpoint* que utilitzi les *DTN* [4] i, per tant, no té els mòduls *worker* que tenia el gestor d'incidents.

6.2 Implementació

En aquesta secció, a l'igual que en el gestor d'incidents, s'exposaran les tecnologies utilitzades i s'analitzarà la implementació de l'aplicació.

Pel que fa a les tecnologies, s'han utilitzat les mateixes que a l'altre gestor, a excepció de l'*aDTNPlus* [3], ja que aquest servei no té cap funcionalitat amb xarxes oportunistes.

En canvi, s'ha afegit una nova tecnologia, que serveix per donar una autenticació basada en *JSON Web Token (JWT)* [9].

Tenint el model definit, s'ha procedit a implementar els diferents controladors per tractar tots els *endpoints* que es requereixen pel funcionament de l'aplicació. Aquests, de la mateixa manera que el gestor d'incidents, s'han organitzat en *blueprints*. En aquest cas, els mòduls són els següents:

En primer lloc, es té el mòdul de *User*, que és el mòdul principal del gestor d'usuaris. Aquest conté mètodes per registrar usuaris i eliminar-los. També permet obtenir un usuari per la seva *ID*, obtenir els seus pseudònims i obtenir una llista d'usuaris on es mostra una part de les dades.

Després, es té el mòdul de *Confirmation*, que està pensat per rebre validacions del gestor d'incidents i confirmar si el pseudònim que les ha reportat és vàlid. Només té dos

endpoints, el de confirmar un incident i el d'una validació. En el cas de les validacions, també actualitza la puntuació de l'usuari que ha reportat l'incident.

També hi ha el mòdul de *Role*, que és un mòdul senzill que permet llistar els rols que hi ha a l'aplicació.

Finalment, es té el mòdul *Verify*, que s'encarrega de l'autenticació dels usuaris. Funciona de manera diferent a una autenticació normal, ja que retorna la llista dels pseudònims i altres dades de l'usuari. Això està pensat pel funcionament amb *DTN* [4], ja que no es poden enviar molts missatges i s'havia d'establir un mètode on es retornessin totes les dades necessàries amb una sola petició. No obstant, també s'utilitza un mètode convencional d'autenticació d'*APIs* basat en *JWT* [9] per gestionar els permisos d'accés a la resta d'*endpoints* quan s'accedeix per *Internet*.

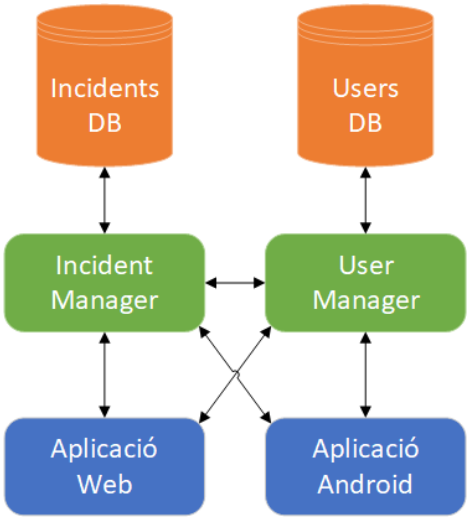


Fig. 3: Esquema del conjunt d'aplicacions i serveis.

7 PROVES

A continuació, s'explicaran les proves que s'han dut a terme durant la implementació dels dos serveis i les que s'han fet un cop estaven tots dos completats.

7.1 Proves individuals

En paral·lel a la implementació s'ha anat validant cada funcionalitat nova que es feia. Per fer-ho, es provaven entrades determinades i s'esperava una resposta correcta, siguin les dades demanades o un missatge d'error amb el codi *HTTP* correcte. Un cop acabat un gestor, s'han refet les proves per validar que totes les funcionalitats d'aquest funcionaven correctament.

7.2 Proves d'integració entre serveis

La primera fase de proves tenia per objectiu validar que ambdós gestors funcionessin correctament per separat. Un cop finalitzada, s'ha procedit a comprovar que la comunicació entre ells fos correcta. L'únic punt de conflicte en aquesta part es donava quan es volia reportar un incident o una validació. Això era degut a que era l'únic moment on el gestor d'incidents necessitava del gestor d'usuaris per confirmar que un incident o una validació havia estat reportada per un usuari correcte. A més, el gestor d'usuaris necessitava saber de qui era l'incident validat per tal d'actualitzar la seva puntuació.

7.3 Proves d'integració amb l'aplicació

Un cop validat tot el back-end, s'ha procedit a provar que els dos gestors poguessin ser utilitzats tant per les aplicacions *Android* com per les aplicacions *Web* que s'han dissenyat a aquest projecte. Per això, s'han desplegat els dos serveis i s'ha procedit a provar el funcionament de les aplicacions quan els utilitzen. Com es pot observar (Fig. 3), les aplicacions utilitzen els dos serveis alhora i, per tant, és necessari que tots dos proporcionin els serveis correctament per assegurar que aquestes es comportin adequadament.

A més, s'han de definir correctament unes interfícies entre l'aplicació i els serveis per tal que es puguin fer servir de manera homogènia a tots llocs. Per això, es van anar provant totes les funcionalitats de les aplicacions per tal de comprovar que donessin els resultats esperats. Per exemple, per provar la funcionalitat d'afegir un usuari s'emplenava el formulari de registre i es mirava a la base de dades que s'hagués afegit correctament. Un cop registrat, també es comprovava que sortís a la llista d'usuaris registrats a l'aplicació *Web*, que es pogués identificar amb la contrasenya que havia donat, i que pogués visualitzar totes les seves dades personals.

La majoria de problemes relacionats amb aquesta part consistien en errors de interfícies, que es van solucionar modificant tant els gestors com les aplicacions. Un exemple d'això és que els gestors, quan no troben un recurs, retornen un codi d'error 404 (Resource Not Found) i les aplicacions esperaven un *JSON* amb un camp d'error.

Un altre problema que va sorgir és que els navegadors web tenen una gestió de Cross-Origin Resource Sharing (*CORS*) [10] que obliga a fer una negociació prèvia a la petició d'un recurs. Per sort, *Flask* ofereix dos mòduls que gestionen això de manera automàtica, però que s'havien d'instal·lar. Aquests mòduls són *Flask-CORS* i *Flask-Jsonpify* [5]. El primer permet fer tota la negociació prèvia entre el navegador i el gestor (enviant una petició de tipus *OPTIONS*) i el segon gestiona que la resposta tingui el format *JSONP* per tal que el navegador accepti la resposta.

7.4 Proves amb xarxes oportunistes

Finalment, s'ha procedit a provar l'última funcionalitat requerida pels objectius del projecte, que l'aplicació funcionés amb xarxes oportunistes.

Per aconseguir transmetre els missatges per aquestes xarxes, s'ha hagut de desplegar tota la plataforma implementada al projecte *Crowd eAssessment* amb l'*aDTNPlus* [3].

Aquesta plataforma està formada per un conjunt de *Raspberry Pi* amb una imatge *Linux* que té instal·lat l'*aDTNPlus* i està configurat per exposar una xarxa *Wi-Fi* pels clients. Per poder aconseguir això, la *Raspberry* ha de tenir connectats dos dispositius *USB*, el *Wi-Pi*, que s'utilitza per transmetre els missatges entre els nodes de la *DTN* [4] (inclòs el servidor), i l'*Edimax*, que és el que permet exposar la xarxa *Wi-Fi*.

D'altra banda, el servidor també necessita la seva pròpia

Wi-Pi per rebre els missatges dels altres nodes *DTN* [4], i ha de tenir instal·lat i configurat el servei de l'*aDTNPlus*. Amb tot això configurat, es necessita utilitzar la llibreria *Python* proporcionada amb l'*aDTNPlus* i utilitzar el socket que proporciona per escoltar i rebre els incidents.

Amb la plataforma desplegada, l'aplicació *Android* dissenyada per aquest projecte envia dades a una *Raspberry*, i aquesta els transmet cap al gestor d'incidents mitjançant aquesta xarxa oportunista.

8 Verificació i Resultats

En aquesta secció es repassaran tots els objectius, es mirarà que s'hagin assolit amb la implementació dels dos serveis i presentaran els resultats obtinguts.

8.1 Anàlisi dels objectius

En aquest apartat s'analitzaran tots els objectius del projecte i s'explicarà com s'han assolit.

S'han implementat totes les funcionalitats requerides pel projecte. El resultat d'això són dos serveis completament independents que ofereixen una sèrie de funcionalitats necessàries per tenir una aplicació de gestió d'incidents.

A continuació, es procedirà a enumerar els objectius i explicar com s'han assolit amb la implementació dels dos serveis:

1. Dissenyar un servei web genèric: aquest objectiu s'ha assolit generant un format estàndard de missatges que dona certa flexibilitat a l'hora de fer noves aplicacions. Aquesta flexibilitat ve donada per una varietat de camps opcionals i uns camps extra per algunes funcionalitats específiques que pogués tenir una aplicació en el futur. A més, té un sistema de categories i subcategories que es poden canviar en funció de l'aplicació que es vulgui utilitzar.

2. Que el servei permeti rebre incidències, processar-les i emmagatzemar-les: això s'ha resolt amb la implementació del gestor d'incidents, ja que aquest s'encarrega precisament d'emmagatzemar les incidències reportades pels usuaris.

3. Comprovar que una incidència prové d'un usuari legítim: per assolir aquest objectiu, s'ha fet un mecanisme pel qual el gestor d'incidents pot demanar al gestor d'usuaris si un pseudònim pertany a un usuari real, de manera que el gestor d'incidents no sap qui és, però sap que té permisos per reportar incidents.

4. Rebre validacions d'incidències i mantenir una puntuació sobre la veracitat: Aquest objectiu ha estat assolit gràcies al sistema de validacions entre el gestor d'usuaris i el d'incidents, que permet mantenir una reputació pels usuaris. A més, es poden veure aquells incidents que han estat confirmats i es manté la llista de validacions, positives i negatives, d'un incident en concret.

5. Permetre obtenir informació sobre els incidents: Actualment el gestor d'incidents exposa varis *endpoints* que permeten accedir a la informació total dels incidents. Així doncs, aquest objectiu s'ha assolit correctament.

6. Proveir un mecanisme que permeti gestionar un conjunt d'usuaris de l'aplicació i que aquests puguin reportar incidències de manera anònima: aquest objectiu s'ha assolit amb la separació de l'aplicació en dos serveis diferents i la gestió dels pseudònims explicada en l'apartat anterior. Així com el gestor d'incidents no sap qui ha estat l'entitat que ha reportat un incident, el gestor d'usuaris no sap quins incidents ha reportat cadascú.

7. Permetre que aquest servei web funcioni a través d'*Internet* i de *DTN* [4]: gràcies als workers implementats a l'aplicació, aquesta pot rebre incidents mitjançant les xarxes oportunistes de la mateixa manera que els que venen per *Internet*. Així doncs, aquest objectiu s'ha assolit.

8. Validar que els servei web s'adapta correctament a diferents aplicacions de report d'incidents: això s'ha comprovat al fer les proves d'integració entre els serveis i les diferents aplicacions (tant *Web* com *Android*) i s'han ajustat tots dos serveis per poder donar una interfície utilitzable per aquestes.

9. Comprovar que l'aplicació funciona correctament amb les xarxes oportunistes: s'han realitzat proves de funcionament i de rendiment amb les *DTN* [4] utilitzant la infraestructura proporcionada [3]. Així doncs, s'ha comprovat que aquesta aplicació funciona correctament amb aquest tipus de xarxes.

Com s'ha vist, s'han assolit tots els objectius del projecte i s'ha explicat quina part de l'aplicació resultant els compleix.

8.2 Anàlisi dels gestors

En aquest apartat es mostraran alguns exemples de funcionalitats dels gestors. El que s'ha fet és tenir els serveis sense cap tipus de dades i executar accions a l'aplicació *Web*. Després de cada acció, s'ha visualitzat a l'aplicació *Web* que l'acció ha repercutit realment.

Inicialment, es provarà de fer una autenticació amb un usuari incorrecte.

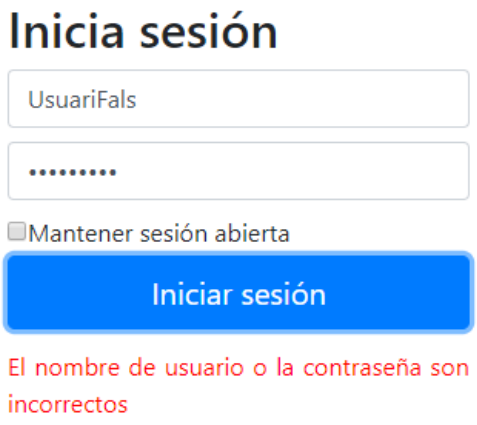
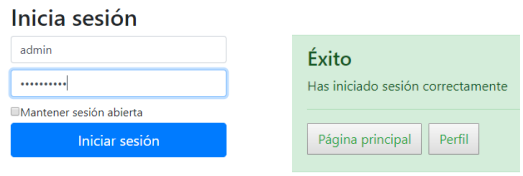


Fig. 4: Exemple d'una autenticació incorrecta.

Com es pot observar (Fig. 4), al intentar autenticar-nos amb un usuari que no existeix o amb una contrasenya

errònia, l'*API* retorna que l'usuari no existeix. Cal notar que no es donarà cap pseudònim ni es permetrà l'accés als altres *endpoints* de l'aplicació sense una autenticació vàlida.

Seguidament, es prova de fer una autenticació amb un nom d'usuari i una contrasenya vàlides.



(a) Formulari d'autenticació.

(b) Missatge d'autenticació correcta.

Fig. 5: Exemple d'autenticació correcta.

A la figura anterior (Fig. 5) es pot apreciar com amb un nom d'usuari i contrasenya correctes es pot accedir als serveis que ofereix el gestor d'usuaris.

A continuació, es procedeix a reportar un incident, validar-lo amb un altre usuari i a mirar els efectes que provoquen aquestes accions a les dades dels gestors.

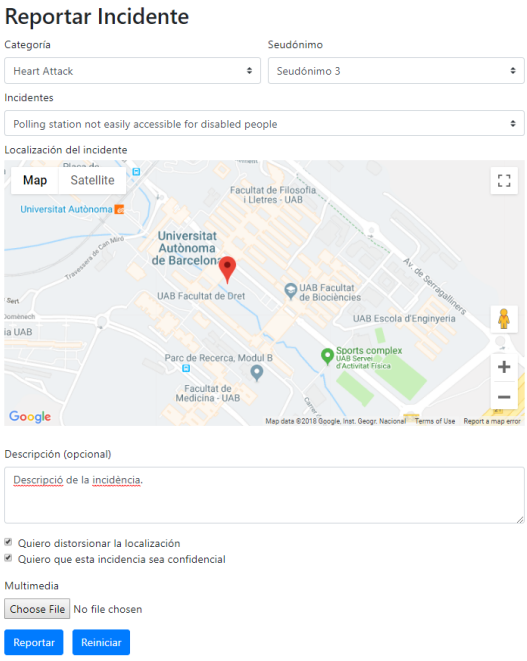


Fig. 6: Formulari de report d'incidents.

Amb el formulari de report d'incidents (Fig. 6) es proporcionen totes les dades per reportar un incident. En aquest cas, la categoria, la subcategoria, el pseudònim a utilitzar, la ubicació, una descripció i no s'adjunta cap imatge. També s'especifica que s'ha de distorsionar la ubicació i que la incidència és confidencial.

Cal notar que la categoria i la subcategoria provenen també d'un *endpoint* de la *API*, de manera que si aquestes canvien a la base de dades, les opcions disponibles també ho fan.

Posteriorment, es canvia d'usuari i es valida l'incident. Això no només farà que l'incident estigui confirmat, sinó que a més actualitzarà la puntuació de l'usuari que l'ha reportat.

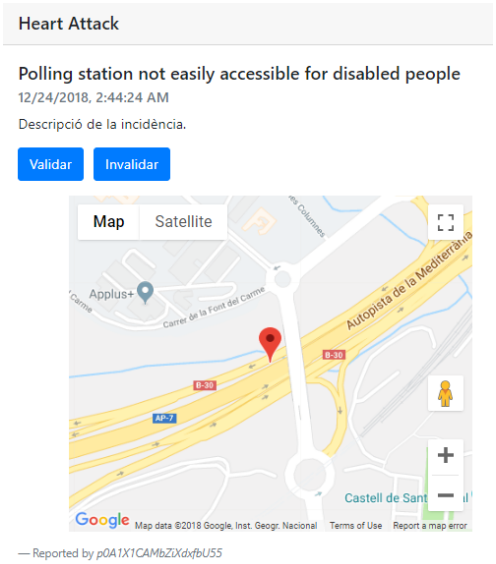


Fig. 7: Formulari de validació d'incidents.

Al formulari de validació d'incidents (Fig. 7) es poden veure les dades de l'incident que s'acaba de reportar, i es donen les opcions per a validar-lo o invalidar-lo. En aquest cas, es procedeix a validar l'incident i s'observen els canvis que això comporta. També es pot veure com s'ha distorsionat la ubicació, ja que, en comptes de ser la mateixa que s'ha enviat, és un punt proper a aquesta.

Finalment, es mira quins efectes han tingut aquestes accions en els gestors, un altre cop utilitzant la web per visualitzar-los.

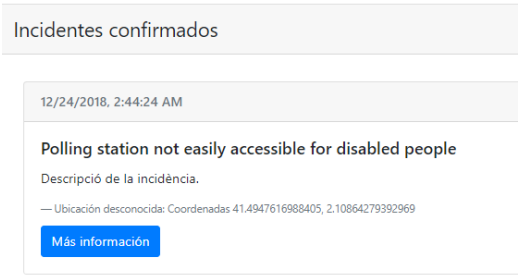


Fig. 8: Llista d'incidents confirmats.

Com es pot apreciar (Fig. 8), al validar l'incident, aquest ha passat a estar a la llista d'incidents validats (que s'obté d'un *endpoint* del gestor d'incidents).

Nombre de usuario	Seudónimos	Alfa	Beta	Reputación
admin	6vC4hu1iloZ6H3cujp9c A7blodsahoSdUlyM3UXj p0A1X1CAMbZiXdxfbU55 EKODcOG6gBFRLKUqsGdf TmACGWemltDSiR7uvgo6	1	0	1

Fig. 9: Puntuació de l'usuari actualitzada.

A més, es pot veure (Fig. 9) com la puntuació de l'usuari s'ha actualitzat d'acord amb la confirmació de l'incident.

Així doncs, s'han provat diverses funcionalitats amb l'ajuda de la pàgina web i s'ha pogut observar el funcionament de la part principal dels dos gestors.

8.3 Anàlisi del rendiment

En aquest últim apartat es procedirà a analitzar els resultats d'eficiència del servei. Per fer-ho, s'enviaran múltiples peticions concurrentment al servidor i s'analitzarà en quin moment és incapaç de respondre a totes alhora i comença a tenir retards en la resposta.

Per fer aquesta part, s'ha utilitzat la web que s'ha desenvolupat amb aquest projecte. A més, s'ha fet que la web s'executi en una altra màquina que no és la que executa cap dels gestors. D'aquesta manera, es té una aproximació del que és capaç de fer el servei en una situació més propera a la realitat. No s'ha fet amb l'aplicació *Android* ja que aquesta tenia problemes de rendiment a l'hora d'enviar paquets i aquests no ens permetien obtenir dades fiables.

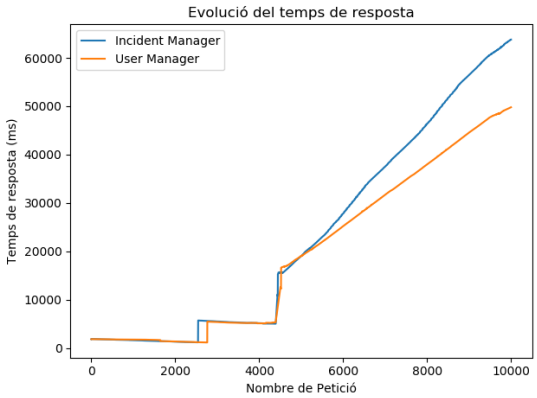


Fig. 10: Comparativa de rendiment entre els gestors.

A la figura anterior (Fig. 10), podem observar com els dos gestors tenen un temps de resposta molt semblant. Això és un resultat esperat ja que tots dos servidors utilitzen les mateixes tecnologies per gestionar les dades. La diferència que es veu entre els dos gestors està produïda pel fet que les peticions al gestor d'incidents contenen més dades que les que es fan al gestor d'usuaris.

Es pot observar com, aproximadament, a partir de la petició *4500* comença a col·lapsar-se el servei i, per tant, es comencen a incrementar els temps de resposta.

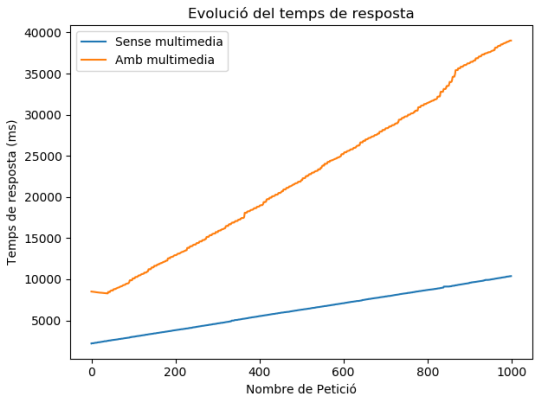


Fig. 11: Comparativa de rendiment amb o sense imatges.

Com podem veure (Fig. 11) el fet de afegir multimèdia als incidents afecta molt al rendiment, ja que el temps de resposta augmenta considerablement. Això és degut, un altre cop, a que s'han de processar més dades. Això provoca que es trigui més a transmetre les dades per la xarxa i que el gestor d'incidents necessiti més temps per tractar-les.

Finalment, s'analitza el rendiment de les peticions provinents de la *DTN* [4] comparant-lo amb les provinents d'*Internet*. Per fer-ho, es compararà el temps de resposta entre les peticions enviades amb l'aplicació *Android* mitjançant *Internet* i les enviades amb la mateixa aplicació mitjançant les xarxes oportunistes.

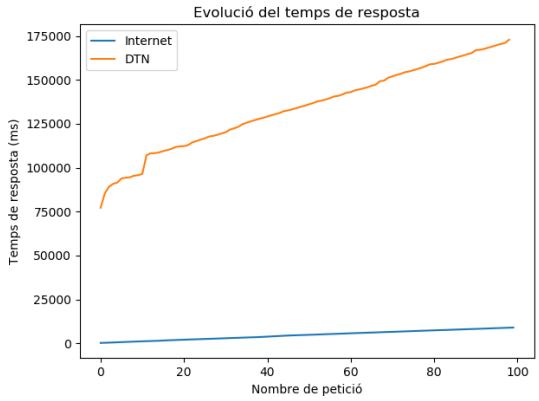


Fig. 12: Comparativa de rendiment entre Internet i DTN.

Com es pot observar (Fig. 12), el rendiment d'*Internet* supera àmpliament al de la *DTN* [4]. Això és d'esperar per dos raons.

La primera és que la infraestructura d'*Internet* ofereix un millor rendiment que la que es té amb les xarxes oportunistes. Les *Raspberry Pi* se saturaven ràpidament amb la quantitat de missatges enviats. A això també s'ha d'afegir que les llibreries estàndard de *Linux* estan molt optimitzades per enviar dades de manera molt eficient.

La segona té a veure amb els protocols utilitzats. Amb *Internet*, l'entrega és molt més senzilla ja que, si el servidor no està connectat al moment d'aquesta, el missatge es perd. D'altra banda, amb *DTN* [4] això no es permet. A més, els missatges que utilitzen aquestes xarxes es transmeten entre els diferents nodes per augmentar la probabilitat d'entrega, ja que és possible que un altre node sí es pugui connectar amb el destinatari. Degut a això, l'entrega del missatge amb la *DTN* és més complexa i, per tant, té un rendiment inferior.

Cal notar que, en alguns casos, els missatges enviats per la *DTN* [4] es perdien degut a la saturació dels nodes. A *Internet*, en canvi, mentre el servidor estigués connectat, tots els missatges arribaven correctament.

Com s'ha vist, s'ha analitzat el rendiment de les *APIs* en varis escenaris i s'han obtingut mesures de l'eficiència d'aquests en les màquines on s'ha provat. A més, s'ha avaluat la eficiència de transmetre incidents per la xarxa *DTN* [4] i s'ha comparat amb els resultats de transmetre'ls per *Internet*.

9 Conclusions

En aquest apartat s'exposaran les conclusions extretes en la realització del projecte.

Durant l'elaboració d'aquest, s'han utilitzat coneixements d'assignatures fetes a la carrera. Exemples d'aquestes assignatures han sigut *Tecnologies per al Desenvolupament d'Internet i Web*, *Sistemes i Tecnologies Web*, *Xarxes*

i *Garantia de la Informació i Seguretat*.

A més, s'han adquirit molts coneixements nous durant la planificació i desenvolupament del treball, sobretot relacionats amb el món del desenvolupament d'aplicacions web. Entre aquests, els més importants són els següents: s'ha après a seleccionar noves tecnologies a utilitzar, tenint en compte varis criteris, com la facilitat d'ús, l'eficiència o la compatibilitat amb el projecte; s'han obtingut coneixements sobre com desenvolupar *APIs REST* seguint els estàndards i guies d'estil; s'ha après que els navegadors tenen polítiques de *CORS* [10] que obliguen a que una *API* hagi de tenir una configuració afegida a més de retornar les dades que se li demanen; finalment, s'ha entès el concepte d'interfície aplicant-lo als dos gestors de manera que es puguin comunicar correctament amb vàries aplicacions alhora.

Respecte als objectius, s'ha assolit la totalitat d'aquests, ja que totes les funcionalitats propostes han estat provades i funcionen en escenaris generals. A més, s'ha aconseguit que els gestors serveixin per a varis tipus d'aplicacions de report d'incidents canviant les categories i subcategories, i oferint un format flexible de dades. També s'ha aconseguit que la *API* funcionés amb la plataforma *aDTNPlus* [3], de manera que es pot utilitzar amb xarxes oportunistes.

Després d'analitzar l'estat actual del projecte i de les *APIs*, es proposen les següents millores, tant a nivell de funcionalitat com a nivell de millora de la qualitat del codi:

- *API REST* completa: degut als requisits del projecte, només s'ha implementat aquella part de la *API* que es requeria per cobrir les funcionalitats de les aplicacions. No obstant, es podria fer per cada recurs la *API* sencera, de manera que quedaria un back-end que ofereix moltes més possibilitats a les aplicacions.

- Explorador de l'*API*: es podria implementar un explorador que permetés conèixer i accedir a tots els *endpoints* de l'aplicació de manera que un nou programador pogués observar com pot utilitzar l'*API*. Aquest explorador també serviria com a una documentació senzilla de veure mostrant exemples de peticions i respostes per a cada *endpoint*.

- Test automàtic: amb aquests tests, el manteniment del software en un futur seria més senzill i segur.

- Generació d'un *package* de l'aplicació: Actualment el que es té és un projecte *Python* amb un fitxer *requirements.txt* que conté les dependències. Es podria generar un *package* que contingui tota l'aplicació i que sigui molt més fàcil d'instal·lar i executar.

- Ampliar la *API* de *Python* de l'*aDTNPlus* [3]: tot i que l'aplicació ja funciona correctament amb la *API* actual, aquesta només exposa una petita part de les funcionalitats de l'*aDTNPlus*. Es podria ampliar de manera que exposés tots els seus mètodes. A més, es podria afegir compatibilitat amb *Python 3*.

En conclusió, aquest treball ha servit per adquirir coneixements nous, s'ha complert la totalitat dels objectius proposats i es té una llista d'elements a millorar o afegir en un futur.

10 AGRAÏMENTS

En primer lloc, agraeixo a la meva família per donar-me tot el suport necessari durant la carrera. Pel mateix motiu agraeixo a la meva parella, Reina Suzuki.

A continuació, al meu company de feina i de projecte José M. Malaguilla, per tots els dies que ens hem hagut de quedar elaborant el projecte i aconseguint que l'aplicació fos funcional i de qualitat.

També agraeixo als professors Sergi Robles i Guillermo Navarro per donar-nos la oportunitat de realitzar aquest projecte i per tot el suport durant totes les fases d'aquest.

Finalment, a la resta de professors que ens han ajudat mentre el desenvolupàvem, especialment a M. Carmen de Toro, Adrià Sánchez i Marc Dalmau.

REFERÈNCIES

[1] E. Enge, "Mobile vs Desktop Usage in 2018: Mobile takes the lead", Stone Temple, 27 d'abril de 2018 [En línia]. Disponible a: https://www.stonetemple.com/mobile-vsdesktop-usage-study/ Consulta: 9 d'octubre de 2018.

[2] Statista, "Percentage of mobile device website traffic worldwide from 1st quarter 2015 to 2nd quarter 2018", Statista, Juliol de 2018 [En línia]. Disponible a: https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/ Consulta: 9 d'octubre de 2018

[3] Navarro-Arribas, G., Borrell, J., Martí R., Robles, S., Herrera, J., "Crowd eAssessment", Projecte Retos Colaboración, RTC-2014-2546-7, Ministerio de Economía y Competitividad, 2014.

[4] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and Weiss, H., "DelayTolerant Networking Architecture", RFC 4838, Abril de 2007

[5] Pallet. "Flask v1.0.2" (Maig de 2018) [en línia]. Disponible a: `http://flask.pocoo.org/`. Consulta: 23 de novembre de 2018.

[6] Copeland, R., Myers, J., "Essential SQLAlchemy, 2nd Edition", California: O'Reilly Media, 2016.

[7] PostgreSQL Global Development Group, "PostgreSQL v11.1", (Novembre 2018) [en línia], Disponible a: `https://www.postgresql.org`. Consulta: 23 de novembre de 2018.

[8] P. Leach, Microsoft, M. Mealling, Refactored Networks LLC, R. Salz, Datapower Technology Inc. "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, Juliol de 2005

[9] M. Jones, Microsoft, J.Bradley, Ping Identity, N. Sakimura, NRI, "JSON Web Token (JWT)", RFC 7519, Maig de 2015

[10] van Kesteren, A, "Cross-Origin Resource Sharing", (Desembre 2018) [en línia], Disponible a: `https://www.w3.org/TR/cors/`. Consulta: 21 de novembre de 2018.