Push Quizzes: eina d'enviament de preguntes via push notifications

Xavier Molina Franco

Resum– Push Quizzes és una eina que permet enviar preguntes a un conjunt de persones i rebre les respostes. L'eina està formada per tres components: Push Quizzes Server, una aplicació mòbil per Android i una per iOS. El servidor conté una aplicació web capaç d'enviar les preguntes via *push notifications* i una base de dades per emmagatzemar les preguntes i respostes. Les dues aplicacions mòbils són capaces de rebre les notificacions i contestar les preguntes mitjançant un camp de text.

Paraules clau– push notification, Firebase, NodeJS, NPM, MongoDB, iOS, Android, HTTP, servidor, webapp, Jade, base dades, API

Abstract– Push Quizzes is a tool that allows you to send questions to a group of people and receive their answers. The tool consists of three components: Server Push Quizzes, a mobile application for Android and iOS. The server contains a web app able to send questions via *push notifications* and a database to store both questions and answers. Both mobile applications can receive notifications and answer the questions via text field.

Keywords– push notification, quiz, Firebase, NodeJS, NPM, MongoDB, iOS, Android, HTTP, XCode, server, webapp, Jade, database, API

1 INTRODUCCIÓ

SEGONS un estudi realitzat per la consultoria d'estratègia digital Ditrendia al 2016 [1], a Espanya un 80% de la població té un telèfon intel·ligent i de mitjana, comprova el mòbil fins a 150 vegades al dia. L'ús generalitzat d'aquests dispositius presenta grans oportunitats per arribar de forma ràpida i efectiva als usuaris de les aplicacions. La més efectiva són les *push notifications*, que consisteixen en l'enviament d'informació a tots aquells dispositius que tenen instal·lada una mateixa aplicació. Un cop reben aquesta notificació, els usuaris poden fer-ne clic i ser portats cap a l'aplicació.

Aquest projecte consisteix en el desenvolupament d'una eina que permet enviar preguntes als mòbils d'un conjunt de persones i recollir-ne les seves respostes. Un escenari típic d'ús serà un professor posant tasques a fer per part dels alumnes fora de l'aula.

Hi ha notificacions que també es poden utilitzar en ordinadors tradicionals o via web. En aquest projecte ens centrem en el seu ús en telèfons mòbils.

Les *push notifications* ens permeten obtenir dues dades interessants:

- El grau d'interès dels usuaris: si aquests fan clic a la notificació i contesten la pregunta, podem deduir que estan interessats i implicats, és a dir, podem comprovar la seva fidelització.
- La resposta a la pregunta ens aporta informació sobre allò que s'està preguntant.

Comencem el document presentant els objectius del projecte (sec.2). A continuació, es defineixen els conceptes i les idees més rellevants que formen part del context del treball (sec.3). Després, es descriu la metodologia de treball utilitzada, es detalla el desenvolupament de cada una de les parts que conformen el projecte (sec.4) i el desplegament dels components. Per últim, s'exposen els diferents experiments i proves realitzats (sec.5), així com els resultats (sec.6) que s'han obtingut i finalment, les conclusions de tot el projecte (sec.7).

• E-mail de contacte: xavier.molinaf@gmail.com
• Menció realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Àngela Fabregues Vinent (dEIC)
• Curs 2016/17

2 OBJECTIUS

La finalitat del projecte és obtenir respostes a preguntes. Aquesta finalitat es cobreix en satisfer els següents objectius:

- Objectiu 1: Obtenir respostes a preguntes formulades als usuaris. L'usuari en veure la pregunta podrà respondre. Les respostes s'han de recollir per a poder-les mostrar a qui ha formulat la pregunta. Aquesta persona no és un usuari qualsevol, serà un administrador.
- Objectiu 2: Poder arribar a l'usuari en qualsevol moment sense haver d'esperar-nos a que obri l'aplicació. Volem que les respostes arribin aviat sense haver de demanar als usuaris que estiguin pendents.
- Objectiu 3: Cobrir la major part de dispositius mòbils en ús.

3 ESTAT DE L'ART

El principal protocol de comunicació que fan servir les aplicacions és HTTP [2] per a connectar-se a servidors web amb APIs disponibles amb informació i procediments. En HTTP és el client, per exemple el mòbil, qui inicia la comunicació enviant la petició. En canvi, la principal característica de la *tecnologia push* es que la petició d'enviament té origen en el servidor. Els *serveis push* treballen per tant amb un model publicador/subscriptor, on el publicador és qui envia la informació i els subscriptor qui la rep.

Per exemple, aquesta tecnologia s'utilitza en una extensió del protocol de correu electrònic IMAP [3], en la qual el servidor avisa al client quan té correus nous disponibles. Les aplicacions de missatgeria instantània o els missatges SMS [4] utilitzen també *tecnologies push*. D'aquesta manera, el client no ha de demanar contínuament al servidor si hi ha missatges nous, sinó que és el servidor qui els enviarà quan estiguin disponibles.

Actualment en el mercat hi ha molts servidors *push* disponibles, per aquest projecte ens hem centrat en les opcions gratuïtes més importants:

- One Signal [5]. Ofereix el servei de *push notifications* i mostra estadístiques de "engagement" a partir de la interacció dels usuaris amb les notificacions.
- Firebase Notifications [6]. Forma part del servei Firebase Cloud Messaging (FCM) [7] que ofereix la plataforma Firebase de Google. Aquesta plataforma ofereix serveis com bases de dades en temps real, autenticació d'usuaris, estadístiques analítiques o monetització a partir d'anuncis [8].

Les dues opcions permeten l'enviament il·limitat de notificacions, sense límit de connexions. Disposen de "kits" de desenvolupament (SDK) pels dos sistemes operatius mòbils principals: Android i iOS. Ambdues disposen també d'APIs per implementar el seu servei en un servidor web propi. Cal destacar que les dues opcions són compatibles entre elles i es podrien utilitzar conjuntament per aprofitar els beneficis de les dues.

Les alternatives de pagament no ens aporten grans avantatges respecte el que necessitem i tenim amb les gratuïtes.

Pel desenvolupament d'aplicacions web hi ha moltes tecnologies disponibles. Algunes de les més destacables són:

- PHP [9]: llenguatge de la banda del servidor (*server-side*) que permet fer pàgines web dinàmiques introduint codi a l'HTML.
- Java Server Faces (JSF) [10]: framework que permet desenvolupar pàgines web amb el llenguatge de programació Java.
- Node.js [11]: és un entorn d'execució multiplataforma que permet executar codi Javascript al costat del servidor. La seva principal característica és que està basat en events i funciona de manera asíncrona. Aquesta característica fa aquesta opció molt interessant quan s'han de fer lectures i escriptures a fitxers o bases de dades i es vol evitar el bloqueig de l'execució de codi mentre es realitzen aquestes accions. Això s'aconsegueix mitjançant l'ús de "callbacks", que són funcions que es posen per paràmetre i es criden un cop s'ha finalitzat el procés que s'estava esperant i s'ha obtingut una resposta. Un dels avantatges de Node.js és la gran quantitat de mòduls que hi ha disponibles per ampliar les seves funcionalitats.

Hi ha altres alternatives com Ruby on Rails [12] o Django[13], que són frameworks que faciliten el desenvolupament web i utilitzen els llenguatges Ruby i Python respectivament.

Pel que fa a les aplicacions mòbils, hi ha diferents opcions a l'hora de desenvolupar per Android i iOS, destaquem dues:

- Programació nativa. Consisteix a programar amb el SDK propi del sistema operatiu. En el cas d'Android i iOS, ambdós sistemes tenen IDEs pròpies per desenvolupar aplicacions: Android Studio i XCode respectivament. Generalment, és la opció més òptima en quant al rendiment i l'execució, i és la que té més suport i esperança de vida útil. En el cas d'Android, el llenguatge de programació és Java. En iOS hi ha dos llenguatges disponibles Objective C i Swift, l'últim d'ells és el recomanat per Apple [14].
- Programació web. Consisteix en programar una aplicació mitjançant HTML i Javascript (html/js), de manera que amb un sol desenvolupament obtenim una aplicació empaquetada per ser executada en qualsevol plataforma. El rendiment d'aquest tipus d'aplicacions acostuma a ser menor ja que depèn d'una capa intermitja que fa de navegador. Tampoc és recomanable utilitzar aquest mètode quan s'han de fer servir opcions pròpies del sistema operatiu, com per exemple les notificacions.

4 DESENVOLUPAMENT

Donat l'objectiu 1 del projecte que requereix el poder obtenir respostes a preguntes formulades, s'ha vist la necessitat de desenvolupar una aplicació web capaç de rebre peticions HTTP amb les respostes i guardar-les en una base de dades. Aquesta mateixa aplicació serveix també per a poder formular i enviar les preguntes.

L'enviament de les preguntes als usuaris s'ha fet amb la *tecnologia push* per tal de poder cobrir l'objectiu 2, que ens requeria d'arribar a l'usuari sense haver-nos d'esperar que

obrís l'aplicació. Per tant, les preguntes formulades s'envien a un *server push* que s'encarrega de fer arribar les *push notifications* amb la pregunta als usuaris.

Anomenem a l'aplicació web Push Quizzes Server i l'hem desenvolupada per tal que faci servir com a component extern Firebase Cloud Messaging, que l'anomenem Firebase Server Push. Ens hem decantat per l'opció de Google per la possibilitat futura d'utilitzar alguns dels altres serveis que ofereix la plataforma Firebase.

Per tal de satisfer l'objectiu 3 i cobrir la majoria de dispositius, hem implementat dues aplicacions mòbils natives, una per Android i l'altre per iOS.

A la figura 1 es mostren les diferents interaccions i comunicacions que es duen a terme entre els diversos components:

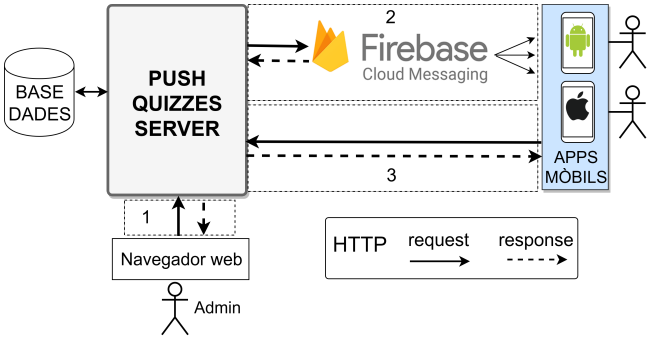

Fig. 1: Comunicacions entre els components

L'usuari "admin" fa servir el navegador per a accedir a Push Quizzes Server i introduir una pregunta. Aquest servidor fa una petició a Firebase per demanar que envii la *push notification*. Firebase ho fa i els mòbils la reben. Quan l'usuari del mòbil clica en la notificació, se li mostra un camp de text per poder respondre. En fer-ho, envia la resposta al Push Quizzes Server utilitzant peticions HTTP.

Per desenvolupar el projecte s'ha fet servir una metodologia seqüencial en cascada. La idea inicial era seguir una metodologia àgil, però finalment es va canviar per una més tradicional degut a que el desenvolupament només incloïa una sola persona. En primer lloc es van definir una sèrie de funcionalitats a implementar per cadascuna de les aplicacions i es van anar desenvolupant. Pel desenvolupament del codi font de les diferents aplicacions s'ha utilitzat un repositori de tipus Git anomenat GitLab [15]. Això ha permès tenir un bon control de les versions del codi, així com de les modificacions que s'han anat realitzant.

A les següents subseccions es detalla el desenvolupament de cadascun dels componenents que conformen el projecte. Finalment es mostra com es va dur a terme el desplegament del Push Quizzes Server en un servidor del IIIA-CSIC.

4.1 Firebase Server Push

Firebase disposa de tres tipus diferents de compte: Spark, Flame i Blaze. Pel desenvolupament del projecte s'ha utilitzat un compte del tipus Spark (gratuït). Els altres dos tipus permenten ampliar els límits que hi ha en altres funcionalitats de Firebase, com per exemple la base de dades en temps real. En aquest cas, com només s'utilitza el servei de *push notifications*, hem triat la més senzilla.

Per poder registrar-se a Firebase és necessari tenir un compte de Google. Un cop creat l'usuari només s'ha de crear un projecte i afegir les aplicacions que es volen enllaçar a aquell compte, les aplicacions que rebran les notificacions. Les apps mòbils s'identifiquen amb un codi del tipus "com.nom_app" que s'ha de triar al començar a desenvolupar l'aplicació. Al afegir cadascuna de les aplicacions es generen uns arxius amb les dades del compte: "google-services.json" en el cas d'Android, i "GoogleService-Info.plist" en el cas d'iOS. Aquests fitxers s'han d'instal·lar a l'interior del codi de cadascuna de les aplicacions perquè puguin rebre les notificacions d'aquell compte.

Amb la creació del projecte, Firebase genera un codi anomenat "Clau del servidor" que serà la que es configurarà al nostre Push Quizzes Server perquè les preguntes s'enviïn a través del nostre compte de Firebase i finalment arribin a les aplicacions mòbils.

Firebase utilitza la subscripció a temes per decidir a qui s'envien les notificacions. En el cas del nostre projecte es vol que arribin a tots aquells usuaris que tinguin l'aplicació d'Android o iOS instal·lada. Per això s'ha definit un sol tema anomenat "global" al que es subscriuen automàticament les aplicacions mòbils.

Firebase proporciona una consola per l'enviament de les notificacions que va bé per a fer proves però resulta farragosa i incòmode de fer servir. És per això que es va decidir implementar una consola pròpia dins el component Push Quizzes Server molt més intuïtiva i fàcil d'utilitzar.

4.2 Push Quizzes Server

L'aplicació web s'ha implementat amb Node.js [16]. S'ha triat aquesta tecnologia degut al seu bon rendiment gràcies a què funciona de forma asíncrona i per la gran quantitat de mòduls que hi ha disponibles. Els mòduls han permès facilitar l'implementació d'algunes funcionalitats com per exemple la gestió de peticions HTTP.

El desenvolupament del Push Quizzes Server segueix el patró Model View Controller (MVC) [17]. MVC separa les dades i la lògica de negoci d'una aplicació de la interfície d'usuari i el mòdul encarregat de gestionar els esdeveniments i les comunicacions.

El codi es divideix en 3 parts principals:

- Vista: tots els documents que es serveixen al navegador web. S'ha utilitzat HTML, CSS i Jade [18].
- Model-Controlador: és tota la part que s'executa en el propi servidor.
- Persistència: la base de dades on s'emmagatzemen totes les dades. S'ha utilitzat MongoDB [19].

Tot el codi font es troba disponible al següent repositori del GitLab de l'IIIA-CSIC [1].

En les següents subseccions es detalla cada una d'aquestes parts.

[1] Repositori Push Quizzes Server: `https://gitLab.iiia.csic.es/xaviermolinaf/pushQuizzes_Server_BD.git`

4.2.1 Vista

Com s'ha comentat en la secció anterior, Firebase disposava d'una consola pròpia per enviar les notificacions, però era poc intuïtiva. Una de les motivacions del projecte és que qualsevol tipus d'usuari sigui capaç d'utilitzar l'eina i enviar una pregunta. Per això es va implementar una consola molt simple per enviar les preguntes. A més a més, des de l'aplicació web es poden consultar dades i hi ha una pàgina d'ajuda a mode de manual.

Hi ha dos tipus de pàgines: les bàsiques o estàtiques que es van implementar amb HTML i les dinàmiques, que contenen consultes a la base de dades i es van fer amb Jade. Com veurem al següent subapartat, Jade és un motor de plantilles que permet modificar dinàmicament un document semblant a HTML amb dades.

Per la part visual es va dissenyar un document CSS comú minimalista i senzill. Es va optar per aquest disseny per transmetre sensació de simplicitat i fer una interfície amigable i fàcil d'utilitzar.

A la figura 2 es mostra un diagrama de navegació amb les diferents pantalles disponibles a l'aplicació web.

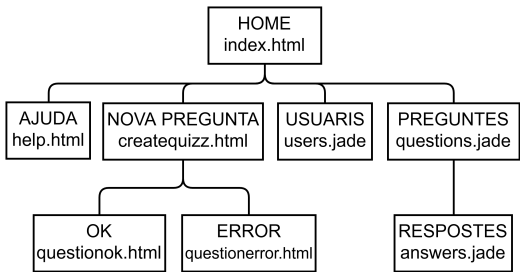

Fig. 2: Diagrama de navegació

4.2.2 Model-Controlador

Aquesta és la capa principal del servidor web de Push Quizzes, és la que s'encarrega de gestionar totes les peticions HTTP (controlador) i fer les lectures i escriptures a la base de dades (model).

Pel desenvolupament del servidor s'han utilitzat els següents moduls de NodeJS:

- Express [20]. Conté moltes funcionalitats que faciliten la gestió d'HTTP. S'ha utilitzat per gestionar les diferents peticions GET i POST que arriben al servidor web. Amb el seu ús s'implementa de forma transparent l'API REST, que permet personalitzar l'adreça URL per tal de no mostrar la ubicació dels diferents arxius al servidor web entre d'altres característiques.

- Body Parser [21]. Permet extreure de forma senzilla la informació que contenen les dades de les peticions (en format JSON).

- MongoDB. És el sistema de base de dades escollit, a diferència de SQL, és un sistema no relacional orientat a documents senzill d'utilitzar.

- FCM-Push [22]. Firebase disposa d'una llibreria oficial per poder implementar al teu propi servidor web, però aquesta és poc intuïtiva, pel que s'ha utilitzat una altre llibreria que l'utilitza, però l'ús de la qual és molt més senzill. Actua a mode de interfície. Per autenticarte amb Firebase s'utilitza la clau de servidor que s'obté del seu lloc web, tal com s'ha explicat a la secció 4.1.

TAULA 1: PETICIONS AL SERVIDOR

Tipus	URL	Descripció
GET	/	Mostra la pantalla de benvinguda amb les opcions disponibles.
GET	/users	Mostra els usuaris registrats.
GET	/questions	Mostra les preguntes.
GET	/answers	Mostra les respostes de cada usuari.
GET	/check	Serveix per comprovar si el servidor està funcionant.
GET	/createquizz	Mostra la consola d'enviament de preguntes.
POST	/newquizz	Gestiona l'enviament de la nova pregunta a Firebase i la guarda a la BD.
POST	/newasw	Petició amb la resposta a una pregunta. La guarda a la BD.
POST	/newuser	Petició de nou usuari. El crea a la BD i envia ID.

Aquestes dependències es poden instal·lar automàticament utilitzant el fitxer "packages.json" inclòs al repositori i la comanda "npm install".

Els dos documents principals són "server.js", que conté tot el codi del servidor, i "default.json", que conté la configuració necessària per fer-lo funcionar. En aquest últim fitxer és on s'haurà d'introduir la clau de servidor obtinguda al crear el projecte de Firebase, i opcionalment les dades de la base de dades si es volen modificar.

El funcionament de l'aplicació NodeJS és la següent: escolta el port 3000, gestiona les peticions HTTP de tipus GET i POST que arriben i fa lectures i escriptures a la base de dades quan convé. Cal destacar que per totes les peticions s'utilitza HTTP en comptes de la seva versió més segura HTTPS [23]. Es va decidir no utilitzar el protocol més segur per la naturalesa de les dades que s'envien, ja que no són dades sensibles.

A la taula 1 es mostren les diferents peticions que es gestionen i quines són les respostes.

Quan es crea una nova pregunta, s'envia un missatge HTTP mitjançant el mòdul FCM-Push cap al servidor de Firebase. Aquest missatge conté la pregunta i altres dades necessàries pel tractament de la notificació a l'aplicació mòbil. A la figura 3 es pot veure el flux de missatges que s'envien des que s'escriu una nova pregunta a la consola fins que aquesta arriba a l'aplicació mòbil via *push notification*. Les fletxes representen un missatge HTTP i el text el seu contingut. El missatge amb el nom "notificació" és el que conté les dades de la notificació. A la taula 2 es pot veure el contingut del cos del missatge. La primera part determina a qui s'enviarà la notificació, en aquest cas a tots aquells que estiguin registrats al tema "global". Com veurem en seccions posteriors, al instal·lar-se l'aplicació i registrar el seu nom i cognom, l'usuari queda automàticament subscrit a aquest tema. La part de "data" conté la informació necessària per l'aplicació mòbil (ID de la pregunta i la pregunta). La ID de la pregunta és necessària per després enviar-la

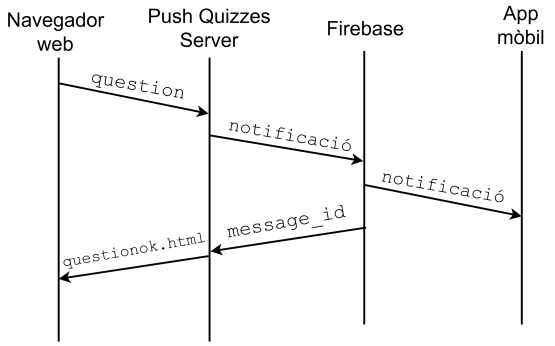

Fig. 3: Missatgeria generada al crear una nova pregunta

TAULA 2: COS DEL MISSATGE A FIREBASE

to: /topics/global
data: question_id question
notification: title body click_action: *OPEN_QACTIVITY* sound: *default* priority: *high*

juntament a la resposta cap al servidor i que es pugui emmagatzemar a la base de dades. El contingut de "notification" és el que es mostra en la notificació que rep l'usuari. L'atribut "click_action" s'envia sempre però només es fa servir a l'aplicació d'Android per determinar quina és l'acció a realitzar quan es fa clic a la notificació.

4.2.3 Model

El model representa la capa de persistència, l'emmagatzemament de les dades. Per a la seva implementació es van considerar dues opcions: utilitzar el servei de base de dades que oferia Firebase o utilitzar alguna tecnologia al propi servidor de Push Quizzes. Finalment es va optar per la segona opció, ja que el servei de BD de Firebase estava limitat pels comptes gratuïts. Una altre motivació va ser l'oportunitat d'aprendre una nova tecnologia d'ús més general i amb bona compatibilitat amb NodeJS.

Per utilitzar MongoDB és necessari tenir el mòdul corresponent a la nostra aplicació NodeJS, i a més a més s'ha d'instal·lar MongoDB al dispositiu on estigui corrent el servidor.

Amb tot instal·lat, l'aplicació NodeJS crearà la base de dades (si no existeix) amb les dades introduïdes al fitxer de configuració "default.json" comentat anteriorment. Per defecte, MongoDB funciona en el port 27017.

La informació de la base de dades es pot consultar independentment de l'aplicació amb la Mongo Command Shell fent "mongo". Una consulta de la col·lecció d'usuaris amb format json ben formatat es faria així: "db.users.find().pretty()".

A diferència de les tecnologies utilitzades durant el grau, MongoDB és una base de dades no relacional orientada a documents. Totes les dades es guarden en documents de tipus BSON [24], que és una especificació de JSON preparada per augmentar el rendiment de les lectures i escriptures. A les bases de dades relacionals les dades s'emmagatzemen en taules, a MongoDB l'equivalent són les col·leccions de documents. Tal com es mostra en la taula 3, s'han creat dos col·leccions, una amb els usuaris ("users") i una amb les preguntes i les respostes corresponents ("questions").

TAULA 3: COS DEL MISSATGE A FIREBASE

Col·lecció	**Atributs**
users	_id name lastname creation_date
questions	_id question question_date answers: *user_id* *answer* *answer_date*

Una particularitat de la col·lecció de preguntes i respostes és que dins l'atribut "answers" es va guardant un conjunt d'atributs per cada resposta. A MongoDB hi ha una mida màxima de 16MB pels fitxers BSON [25], això fa que aquesta implementació pugui desbordar el fitxer en cas de tenir molts usuaris. No ha estat el cas, però una possible solució seria crear una col·lecció nova només de respostes i anar afegint les seves IDs a la col·lecció de preguntes.

4.3 Aplicacions mòbils

S'han desenvolupat dues aplicacions mòbils, una pel sistema operatiu iOS i una per Android. Es van escollir aquests dos perquè cobreixen pràcticament tot el mercat mòbil.

La funcionalitat bàsica de l'aplicació és poder rebre preguntes via *push notification*, fer clic i poder respondre la pregunta mitjançant un camp de text. S'han intentat fer les dues aplicacions pràcticament iguals tant funcionalment com visualment. Per la interfície s'ha optat per un disseny simple i intuïtiu. El diagrama de navegació de les dues aplicacions es mostra a la figura 4. Com podem veure, les aplicacions estan formades per sis pantalles diferents.

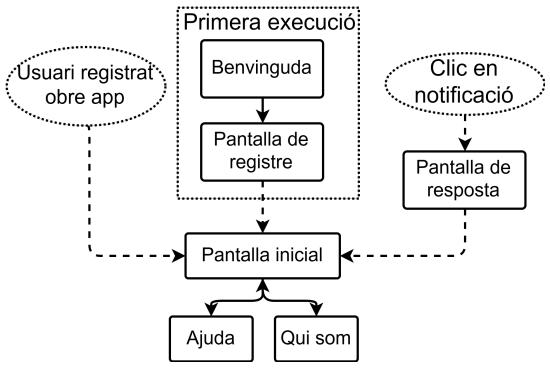

Fig. 4: Diagrama de navegació apps mòbils

Per poder emmagatzemar correctament les respostes de cada usuari a la base de dades, es va decidir incloure un

petit formulari de registre que demana el nom i cognoms de l'usuari el primer cop que s'obre l'aplicació. Aquesta informació s'envia en una petició HTTP cap al servidor, que retorna un ID d'usuari (el generat per la BD). Aquest codi identificatiu s'emmagatzema al dispositiu per tal d'enviar-lo a cada resposta. El mètode d'emmagatzemament de cada aplicació es detalla a les següents seccions. Aquest procés es mostra a la figura 5.

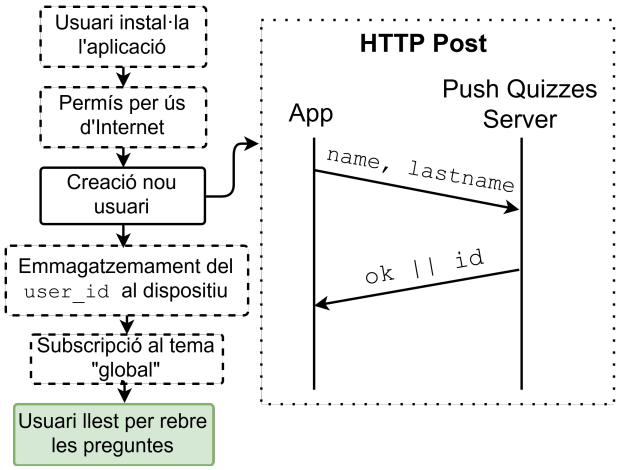

Fig. 5: Procés de registre d'usuari

Un cop l'usuari es registra correctament, ja està preparat per rebre les *push notifications*. Quan rep una nova pregunta i fa clic en la notificació, és portat a una pantalla amb un formulari de resposta, tal com mostra el diagrama de navegació de la figura 4. Un cop completada la resposta, s'envia mitjançant un missatge HTTP que conté la ID de la pregunta (rebuda amb la pregunta), la resposta i la ID d'usuari que es troba emmagatzemada al dispositiu. D'aquesta manera es guardarà a la BD la resposta juntament amb l'usuari que l'ha enviat. A la figura 6 es mostra l'intercanvi de missatges al enviar la resposta.

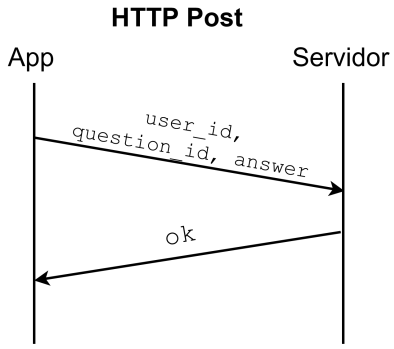

Fig. 6: Intercanvi de missatges al enviar una resposta

Finalment, hi ha una pantalla inicial que dóna accés a dues pantalles més la d'ajuda i la de contacte (qui som). La primera mostra algunes preguntes comuns (FAQ) i permet comprovar l'estat del servei i els permisos i la segona mostra les dades dels participants al desenvolupament. A les següents subseccions es detalla el desenvolupament de cadascuna de les aplicacions.

4.3.1 Aplicació per iOS

El desenvolupament de l'aplicació per iOS s'ha realitzat en la seva totalitat amb l'eina XCode (versió 8) només disponible per ordinadors MAC. La versió del sistema operatiu utilitzada ha estat MacOS Sierra. Durant el desenvolupament estava previst que es fes servir un MAC i un iPad del IIIA-CSIC amb tot l'entorn de desenvolupament instal·lat i un usuari habilitat. Finalment es va decidir, però, realitzar-ho tot de zero per aprendre a instal·lar i configurar tot. Això es va traduir en més temps de dedicació i en aprenentatge. El llenguatge de programació triat ha estat Swift. Com no havíem treballat anteriorment ni amb Swift ni amb Objective C, es va decidir triar la opció recomanada per Apple, Swift.

El primer a realitzar va ser crear un nou projecte a Xcode i instal·lar els mòduls, llibreries i dependències necessàries [26]. Per fer-ho es va utilitzar el gestor de dependències CocoaPods [27]. Els "pods" són les llibreries externes a utilitzar. Pel projecte s'han utilitzat les següents:

- Firebase i Firebase Messaging: llibreria oficial de Firebase necessària per utilitzar els seus serveis. A més a més, es va haver d'incloure el fitxer "GoogleService-Info.plist" amb les dades del compte de Firebase.
- Alamofire [28]: llibreria que permet l'enviament i gestió de peticions HTTP de forma senzilla i intuïtiva.

Per desenvolupar una aplicació iOS que faci ús de *push notifications* és necessària una compte de desenvolupador. L'IIIA-CSIC ens va donar accés a un compte amb privilegis d'administrador. El primer pas va ser generar un certificat personal per permetre l'autenticació al servei de Firebase, aquest certificat s'havia de pujar al projecte de Firebase. Des de la configuració del compte de desenvolupador d'Apple es va activar el servei de *push notifications* per l'aplicació.

L'eina Xcode facilita un simulador per provar l'aplicació, però les funcionalitats de xarxa no funcionen pel que era necessari utilitzar un dispositiu físic. Per això es va crear un "perfil de aprovisionamiento" a la configuració del compte de desenvolupador i es van registrar els usuaris corresponents. Un cop fet això, va ser necessari mirar molta documentació per entendre com desenvolupar amb l'eina. En iOS el codi s'estructura mitjançant el patró "Model View Controller" (MVC).

Les vistes es dissenyen a l'anomenat Storyboard, que és una pissarra on es pot dissenyar la part gràfica de cada una de les pantalles que conformen l'aplicació. A més a més, permet enllaçar les diferents vistes entre elles mitjançant accions (p.e botons). En el cas d'aquest projecte, per cada una d'aquestes vistes hi ha un "controller", és a dir, una classe que gestiona la funcionalitat a nivell de codi de la vista. També hi ha una classe especial anomenada "App Delegate" que controla el funcionament general de l'aplicació. Gestiona els serveis d'aplicació (com la connexió a la xarxa) i gestiona les notificacions.

Com s'ha vist a la secció 4.3, és necessari detectar si l'usuari s'ha registrat o no per mostrar la pantalla de registre al iniciar o no, i també poder emmagatzemar el codi d'usuari. A l'aplicació iOS això s'ha implementat mitjançant una base de dades del tipus "clau-valor" anomenada "UserDefaults". Aquesta BD està inclosa en l'aplicació, per tant,

si es desinstal·la aquestes dades es perden. Per una banda es guarda un booleà que es posa a "True" un cop es registra l'usuari i que fa que no s'executi el "tour de benvinguda". Per l'altre, es guarda la ID d'usuari que retorna el servidor.

Cal destacar que en el cas de iOS al accedir per primer cop a l'aplicació es demanen permisos per rebre notificacions, si aquests no s'accepten no es rebran preguntes. Des de la pantalla d'ajuda o des de la configuració del dispositiu però, es poden activar.

Per poder instal·lar l'aplicació a dispositius iOS s'ha generat un arxiu .ipa que es troba disponible a la pàgina on està desplegat el servidor[2].

El codi font de l'aplicació iOS es troba a un repositori Git [3]. La versió mínima d'iOS per poder utilitzar l'aplicació és iOS 8.

4.3.2 Aplicació per Android

Pel desenvolupament s'ha utilitzat Android Studio en un ordinador amb el sistema operatiu Windows 10, ja que aquesta eina proporcionada per Google és l'estàndard per desenvolupar per Android i facilita molt la integració dels serveis de Firebase dins del projecte. El llenguatge de programació utilitzat és Java.

El primer pas va ser crear un projecte buit d'Android Studio. El següent pas va ser declarar totes les dependències necessàries al "build.gradle", fitxer que conté tots els mòduls que s'han de carregar per poder compilar l'aplicació. Les dependències utilitzades en aquest projecte han estat:

- Firebase: Firebase Messaging 10.0.1. Llibreria oficial de Firebase necessària per utilitzar els seus serveis.
- Fast Android Networking [29]. Llibreria molt senzilla utilitzada per l'enviament de peticions HTTP. Fa ús d'un altre llibreria anomenada OkHttp [30] menys intuïtiva.

Hi ha dos tipus de classes diferents: les activitats i els serveis. Les activitats són components de l'aplicació que conten una pantalla amb la que els usuaris poden interactuar. El serveis permeten realitzar operacions de llarga durada, en segon pla, com per exemple esperar notificacions.

Un dels fitxers més importants d'un projecte d'Android és el "Android Manifest". En aquest document s'han de declarar totes les activitats i serveis que conformen el projecte, així com la informació principal del projecte (p.e nom de l'app). També es va haver de triar un SDK mínim, és a dir, a partir de quina versió d'Android serà compatible l'aplicació. Per qüestions visuals es va triar que el SDK mínim fos la 21 (Android Lollipop).

Per cada pantalla de les mostrades al diagrama de navegació (figura 4) s'ha implementat una activitat per gestionar el funcionament. A Android la part visual de les pantalles es dissenya en els fitxers "layouts", que utilitzen el llenguatge XML. Hi ha un per cadascuna de les activitats. El flux entre les diferents pantalles es gestiona mitjançant "intents", és la manera d'invocar una activitat i fer que es mostrin.

El tractament de les notificacions a Android depèn de l'estat en què es troba l'aplicació quan arriba. Si es troba en segon pla, és el propi sistema Android qui crea la notificació i gestiona el comportament al fer el clic. En canvi, si es troba en primer pla, és la pròpia aplicació qui s'encarrega de crear-la i fer-la visible. És per això que es necessari un servei per poder gestionar el comportament en primer pla. Es diu "Messaging Service".

Amb això el problema del primer pla va quedar solucionat, però va aparèixer un inconvenient amb les notificacions en segon pla. Com és el propi sistema qui s'encarrega de gestionar la notificació, en fer clic a la notificació, per defecte s'obre la pantalla inicial. En el nostre cas, ens interessava que s'obrís la pantalla del formulari per respondre la pregunta. Per solucionar-ho es va utilitzar el "click_action" comentat en anteriors seccions. Afegint "click_action=OPEN_QACTIVITY" al contingut de les notificacions i enllaçant aquesta marca a la activitat de la resposta de preguntes al Android Manifest, es va solucionar.

Per tant, quan es fa clic en la notificació es crea un "intent" i es llança la pantalla amb el formulari de resposta. Per poder mostrar la pregunta rebuda en aquesta vista, el contingut rebut a la notificació s'afegeix al "intent" de forma que al carregar la vista es puguin posar les dades en pantalla. D'aquesta manera obtenim també la ID de pregunta que haurem d'enviar amb la resposta.

Com a iOS, necessitem detectar si l'usuari s'ha registrat o no per mostrar la pantalla de registre al iniciar o no, i també poder emmagatzemar el codi d'usuari. A Android això s'ha implementat mitjançant una base de dades del tipus "clau-valor" anomenada "SharedPreferences". Aquesta BD està inclosa en l'aplicació, per tant, si es desinstal·la aquestes dades es perden. Per una banda es guarda un booleà que es posa a "True" un cop es registra l'usuari i que fa que no s'executi el "tour de benvinguda". Per l'altre, es guarda la ID d'usuari que retorna el servidor.

Per poder instal·lar l'aplicació a dispositius Android s'ha generat un arxiu .apk que es troba disponible a la pàgina on està desplegat el servidor [2].

El codi font de l'aplicació Android es troba a un repositori Git[4].

4.4 Desplegament

El servidor i l'aplicació web desenvolupada en NodeJS es poden utilitzar de forma local per enviar preguntes, però per poder rebre les repostes és necessari que aquest estigui desplegat en un servidor web públic, accessible de forma externa.

A l'IIIA-CSIC ens varen donar accés a un servidor Linux Debian per tal de desplegar la nostra eina. També van habilitar un domini. En el servidor ja es trobava instal·lat tant NodeJS com MongoDB, així que només vam haver-ho de configurar per la nostra aplicació. Això si, totes les proves s'han hagut de fer desplegant en aquest servidor.

Pel desplegament es va utilitzar Nginx [31] i Supervisor [32]. El primer actua com a servidor web escoltant el port 80. Tot el tràfic que arriba és redirigeix cap a la nostra aplicació NodeJS, que corre en el port 3000. El segon és un gestor de processos que permet mantenir l'execució de l'aplicació NodeJS de manera permanent. Actualment es troba tot desplegat en aquest enllaç [2].

[2]Push Quizzes Server: `http://push.edu.iiia.csic.es/`

[3]Repositori app iOS: `https://gitLab.iiia.csic.es/xaviermolinaf/pushQuizzes_iOS.git`

[4]Repositori app Android: `https://gitLab.iiia.csic.es/education-tfgs/pushQuizzes.git`

5 EXPERIMENTS

En aquest apartat es presenten les proves més interessants que es van fer per les diferents aplicacions.

5.1 Aplicacions mòbils

Tant per iOS com per Android es van realitzar les mateixes proves.

Els dispositius que es van utilitzar per Android:

- Google Nexus 6P amb la versió 7.0.1 d'Android.
- Google Nexus 4 amb la versió 5.1 d'Android.

Els dispositius que es van utilitzar per iOS:

- iPhone 5 amb la versió d'iOS 10.3.2.
- iPad Pro amb la versió d'iOS 10.3.2.

Es van escollir aquests dispositius perquè un d'ells és més gran que l'altre i en el cas d'Android per provar diverses versions. Les proves es van repetir en cadascun dels dispositius.

5.1.1 Prova 1: aplicació simple

Un cop creat el projecte, afegides les dependències i creada una pantalla principal, es va provar de compilar i obrir l'aplicació en els dispositius. Tant en el cas d'Android com iOS l'aplicació es va llançar i va mostrar la pantalla inicial correctament.

5.1.2 Prova 2: notificacions

Un cop configurat Firebase a les dues aplicacions i abans de tenir implementat el servidor propi, es va utilitzar la consola de Firebase per provar l'enviament i recepció de les *push notifications*.

Es van provar tres casuístiques diferents:

- Aplicació oberta (en primer pla).
- Aplicació minimitzada (segon pla sense tancar-la del gestor).
- Aplicació tancada.

A la primera prova, a iOS no va funcionar en cap de les casuístiques. El problema era que la subscripció al tema "global" no es feia a l'instant correcte. Un cop solucionat l'error, va funcionar a tots els casos i dispositius provats.

Amb Android va haver més problemes, en el primer cas no apareixia la notificació, en els altres dos casos sortia però no sortia la pantalla adequada al fer clic. El primer error es va solucionar implementant una classe nova. El segon cas no es va poder solucionar fins a la implementació de la consola pròpia del servidor per poder enviar el "click_action".

5.1.3 Prova 3: usabilitat i interfície

Per poder comprovar si l'aplicació era suficientment simple i intuïtiva, un cop estava tot funcionant correctament, es va deixar provar l'aplicació a diferents persones. Els perfils de les persones que la van provar són els següents:

- Perfil 1: Dona de 25 anys, estudiant.
- Perfil 2: Dona de 41 anys, llicenciada en Economia.
- Perfil 3: Home de 62 anys, estudis bàsics.

A cadascun d'ells se'ls hi va deixar un dispositiu amb l'aplicació instal·lada i se'ls hi va explicar en què consistia. Els dos primers van utilitzar dispositius iOS i l'altre Android. El primer pas va ser obrir-la i registrar-se. Els tres van poder completar el registre, però en el cas de la tercera persona va tenir problemes per veure alguns texts.

Després es va enviar una pregunta. En aquell moment tots tenien l'aplicació oberta menys la segona persona, que va trigar uns segons en veure que li havia arribat una pregunta. Tots van poder contestar les preguntes correctament.

Per tant, en tots els casos provats es va aconseguir enviar la resposta correctament. Les conclusions que es van extreure van ser que l'aplicació complia el seu objectiu de ser simple i intuïtiva, funcionava bé en els casos provats i que potser s'havia de considerar fer tot el contingut una mica més gran.

5.2 Push Quizzes Server

En el cas del servidor, les proves més importants que es van realitzar van ser per l'enviament de les preguntes i les escriptures a la base de dades.

5.2.1 Proves d'enviament de preguntes

Per comprovar que funcionava correctament l'enviament de preguntes des de la consola pròpia. Per fer-ho simplement es van anar enviant preguntes i es va anar comprovant que tant els dispositius Android com els dispositius iOS les rebessin. A les primeres proves no es va aconseguir que arribessin, i el motiu va ser per una mala configuració del mòdul FCM-Push. Tal i com s'ha comentat en seccions anteriors, per poder fer-lo funcionar era necessari una clau de servidor. No funcionava perquè la clau que introduïa no era la correcte, utilitzava una altra clau que proporciona Firebase per altres funcions.

Un cop ja arribaven, ens vam adonar que amb el mòbil amb el so activat no sonaven. El motiu va ser que calia enviar dins el cos de la notificació l'atribut "sound" i calia determinar una prioritat alta.

5.2.2 Proves de la base de dades

Amb tota la missatgeria funcionant correctament, només quedava comprovar que tot s'emmagatzemava correctament a la base de dades. Per provar-ho, es van desinstal·lar les aplicacions als dispositius mòbils (per borrar les dades d'usuari) i es van registrar de nou.

Es va anar a la BD i es va comprovar que s'afegissin els usuaris. Ho va fer correctament.

Després es va comprovar que al crear una pregunta aquesta s'emmagatzemés correctament i retornés la ID de la pregunta. Per això es va crear una pregunta des de la consola i mitjançant comandes del tipus "console.log" es va comprovar que la ID era correcte. També va funcionar en el cas provat.

Finalment, es va provar l'emmagatzemament de les respostes. Es va enviar una pregunta i es va contestar des de dos dispositius. Al mirar la BD vam veure que només s'havia guardat la última resposta. Vam mirar els logs i realment

havien arribat les dues, però una havia sobre-escrit a l'altre. Això va fer adonar-nos que l'escriptura de les respostes no s'estava fent correctament, s'anaven sobre-escrivint.

Un cop solucionat es va tornar a repetir la prova i va funcionar tot correctament.

5.3 Proves finals

Un cop fetes les proves individuals de cada component, es va fer una prova amb tots tres components a l'hora. Amb l'aplicació mòbil instal·lada a 17 dispositius (16 Android i un iOS) es va enviar una pregunta. Als usuaris se'ls hi va explicar prèviament en què consistia l'aplicació, però no se'ls hi va dir quan arribaria la pregunta.

Al cap d'unes hores es va mirar la BD per comprovar les respostes, i tots havien contestat. De mitjana van contestar en 10 minuts. La pregunta va ser si havien tingut algun problema amb les aplicacions per tal d'obtenir "feedback", tots van contestar que no havien tingut cap. Per tant, la prova va resultar molt satisfactòria.

6 Resultats

Després de les etapes de desenvolupament i proves podem dir que hem obtingut tres resultats principals: l'aplicació web desenvolupada en NodeJS juntament amb la base de dades, l'aplicació Android i l'aplicació iOS.

Tots tres components han funcionat correctament en les proves que s'han realitzat, per tant hem superat els objectius proposats.

A continuació es mostra el que s'ha aconseguit amb el desenvolupament de cadascun dels components.

6.1 Servidor web i base de dades

Com s'ha comentat anteriorment, el servidor web s'ha desenvolupat amb NodeJS. Això ha fet que sigui més senzilla la seva implementació, i ha permès posar en pràctica alguns dels coneixements que es van aprendre sobre aquesta tecnologia durant l'assignatura de Sistemes i Tecnologies Web. Pel que fa a la base de dades, MongoDB va ser la tecnologia escollida. Durant els quatre anys de grau sempre s'havia treballat amb bases de dades relacionals (SQL), així que ha sigut molt interessant provar alguna diferent. Ens ha sorprès la seva facilitat d'implementació en conjunt amb NodeJS.

Amb la implementació d'una consola pròpia per enviar les preguntes s'ha pogut dissenyar una interfície molt simple que pot ser utilitzada per tothom. A la figura 10 es pot veure una captura de l'aplicació.

6.2 Aplicacions mòbils

Amb el desenvolupament de les dues aplicacions mòbils s'han hagut d'incorporar coneixements apresos durant el grau sobre el protocol HTTP, però adaptat a tecnologies mòbils. La figura 8 mostra dues captures de les aplicacions mòbils.

En el cas d'Android, ja havíem treballat amb Android Studio i Java en assignatures com Disseny del Software, i per tant els coneixements previs han ajudat molt a desenvolupar la feina de forma més àgil.

Fig. 7: Captura de la pantalla de creació d'una nova pregunta

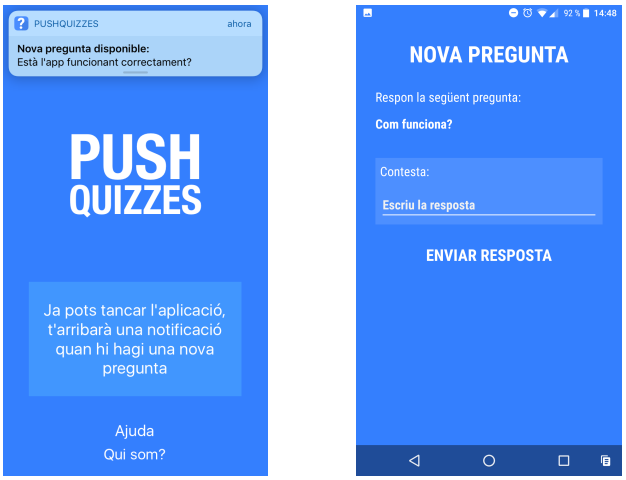

Fig. 8: Captures de pantalla apps mòbils (iOS/Android).

El desenvolupament de l'aplicació pel sistema iOS va ser més difícil, ja que tant el llenguatge de programació (Swift) com les eines a utilitzar (Xcode) eren noves per nosaltres. Això ha permès incorporar nous coneixements no vistos durant el grau.

Un dels aspectes que finalment s'ha quedat fora del projecte ha estat la publicació de les aplicacions a les "app stores" d'iOS i Android. El motiu principal ha sigut que el temps necessari per aconseguir-ho era superior al planificat inicialment. Amb la publicació al servidor dels APK i IPA creiem que ha estat suficient per compartir-la.

Un altre aspecte que es va tenir en compte i no es va aconseguir va ser la independència de les dades de Firebase. Recordem que la informació del compte de Firebase està inclosa dins el projecte d'Android Studio/Xcode, per tant, si es vol modificar s'ha de tornar compilar. S'ha intentat trobar alguna manera perquè aquesta informació pogués ser enviada des del servidor propi però no ha estat possible.

Tot i això, els objectius principals del projecte s'han completat satisfactòriament.

7 Conclusions

L'objectiu principal del projecte era el següent: desenvolupar una aplicació pels sistemes operatius mòbils Android i iOS que permetés rebre preguntes via notificacions (*push notifications*) i que aquestes puguessin ser contestades mitjançant un camp de text.

Amb el desenvolupament del projecte finalitzat, es pot

dir que els objectius marcats s'han completat de forma satisfactòria, obtenint una eina molt completa.

Durant tot el desenvolupament, s'han posat en pràctica coneixements adquirits durant el grau. Per exemple, amb el desenvolupament del servidor web es van posar en pràctica coneixements ja vistos, com la configuració d'un servidor i l'ús de NodeJS, HTML i CSS. També s'han incorporat de nous, per exemple, amb la implementació de les aplicacions mòbils s'ha après a incorporar missatgeria HTTP en aquestes, i en el cas d'iOS s'ha après a treballar amb una eina totalment nova (Xcode). També s'ha après a treballar amb llenguatges de programació nous com Jade o Swift, sent aquesta última part la més difícil del projecte.

A més a més de tot això, una de les parts més dificultoses i a l'hora més interessants, ha estat la d'iniciar i planificar un projecte de forma individual i des de zero.

Tot i haver completat els objectius proposats, el projecte podria estendre's. El treball ha estat centrat en implementar el sistema de preguntes/respostes via *push notifications*, però no s'ha fet èmfasi en el tractament d'aquestes respostes.

Per tant, una extensió molt interessant del projecte seria tractar les respostes per tal de poder extreure informació automàticament a partir de tècniques d'intel·ligència artificial.

En conclusió, amb el desenvolupament del projecte s'han posat en pràctica coneixements vistos durant el grau, i també s'han adquirits de nous, sobretot amb l'aplicació de iOS. S'han aconseguit complir els objectius marcats del projecte, i el potencial del projecte deixa les portes obertes a possibles extensions, com el tractament de les respostes.

AGRAÏMENTS

M'agradaria agrair a la meva tutora per orientar-me durant el desenvolupament del projecte i per donar-me l'oportunitat de fer-ho al IIIA-CSIC. També agrair als mestres d'assignatures com Sistemes i Tecnologies Web, Xarxes o Tecnologies de Desenvolupament per a Internet i Web, ja que els seus coneixements m'han ajudat molt a l'hora de desenvolupar el projecte.

Per últim agrair a la meva família i amics per la paciència i també per la col·laboració en les proves de les aplicacions.

REFERÈNCIES

[1] DITENDRIA. Informe Mobile en España y en el Mundo 2016. Disponible a: http://www.amic.media/media/files/file_352_1050.pdf

[2] R. Fielding, J. Reschke. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. RFC-7230. Juny 2014.

[3] B. Leiba. IMAP4 IDLE command. RFC-2177. Juny 1997.

[4] E. Wilde, A. Vaha-Sipila. Short Message Service (SMS). RFC-5724. Gener 2010.

[5] ONE SIGNAL, One Signal. Disponible a: https://onesignal.com/

[6] FIREBASE, Firebase Notifications. Disponible a: https://firebase.google.com/docs/notifications/

[7] FIREBASE, Firebase Cloud Messaging. Disponible a: https://firebase.google.com/docs/cloud-messaging/?hl=es-419

[8] FIREBASE, Products. Disponible a: https://firebase.google.com/products/

[9] PHP, ¿Qué es PHP?. Disponible a: http://php.net/manual/es/intro-whatis.php

[10] ORACLE, JavaServer Faces Technology. Disponible a: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

[11] NODEJS, NodeJS. Disponible a: https://nodejs.org

[12] RUBY, Ruby on Rails: El desarrollo web que no molesta. Disponible a: rubyonrails.org

[13] DJANGO, Meet Django. Disponible a: https://www.djangoproject.com

[14] APPLE DEVELOPER, Swift 4. Disponible a: https://developer.apple.com/swift/

[15] GITLAB, About GitLab. Disponible a: https://about.gitlab.com/

[16] Shelley Powers,*Learning Node*, 1a ed. EUA: O'Reilly Media, 2012.

[17] BORINI, Stefano. An in-depth analysis of Traditional MVC roles and components (Cap.1.4). Disponible a: https://stefanoborini.gitbooks.io/modelviewcontroller/

[18] MIT, NPM: Jade. Disponible a: https://www.npmjs.com/package/jade

[19] MONGODB, Reinventando la gestión de datos. Disponible a: https://www.mongodb.com/

[20] EXPRESSJS, Fast, unopinionated, minimalist web framework for Node.js. Disponible a: http://expressjs.com

[21] WILSON, Doug. Body-Parser (Github). Disponible a: https://github.com/expressjs/body-parser

[22] KURNIAWAN, Oscar. FCM-Push (Github). Disponible a: https://github.com/nandarustam/fcm-push

[23] E. Rescorla. HTTP over TLS. RFC-2818. Maig 2000.

[24] BSON, Bson Spec. Disponible a: http://bsonspec.org/spec.html

[25] MONGODB, MongoDB Limits and Thresholds. Disponible a: https://docs.mongodb.com/manual/reference/limits/

[26] FIREBASE, Agregar Firebase a tu proyecto iOS. Disponible a: https://firebase.google.com/docs/ios/setup

[27] COCOAPODS, What is Cocoapods. Disponible a: https://cocoapods.org/

[28] ALAMOFIRE, Alamofire. Disponible a: https://github.com/Alamofire/Alamofire

[29] SHEKHAR, Amit. A Complete Fast Android Networking Library that also supports HTTP/2 (Github). Disponible a: https://github.com/amitshekhariitbhu/Fast-Android-Networking

[30] OKHTTP, Overview. Disponible a: http://square.github.io/okhttp/

[31] NGINX, Nginx. Disponible a: https://nginx.org

[32] SUPERVISORD, Supervisor: A Process Control System. Disponible a: http://supervisord.org/

APÈNDIX

A.1 Captures aplicacions Android/iOS

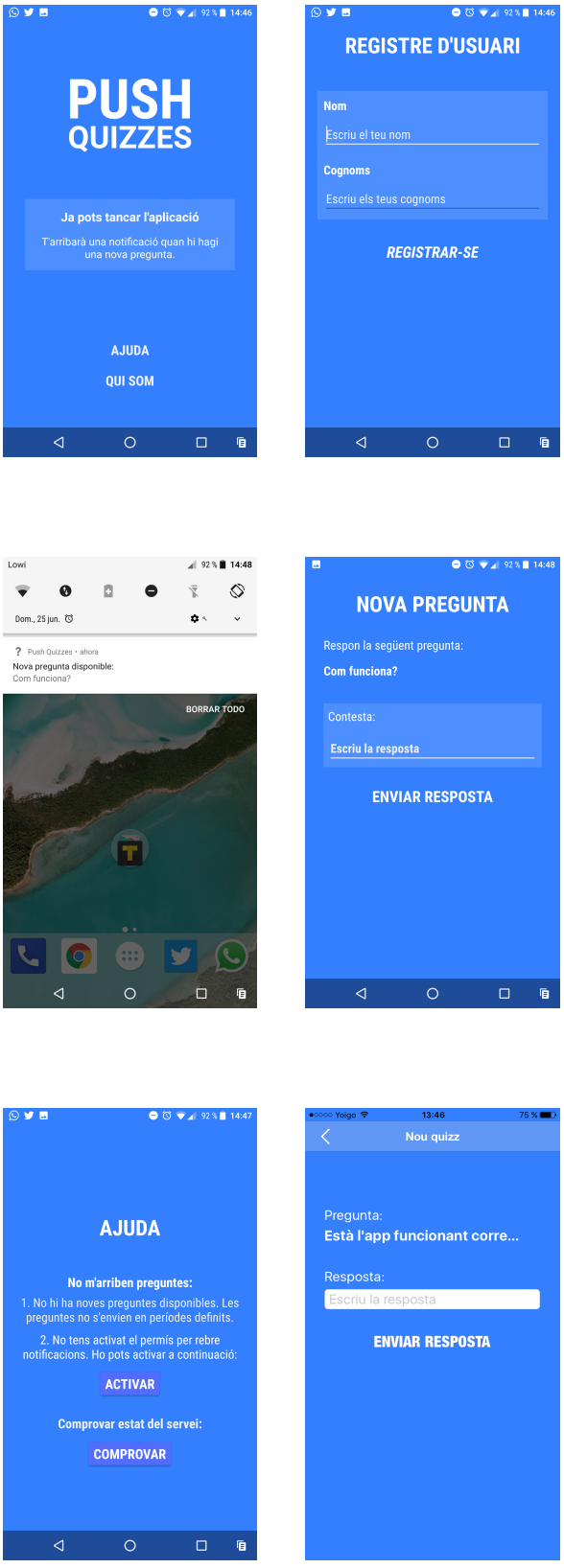

Fig. 9: Captures de pantalla apps mòbils

A.2 Captura Push Quizzes Server

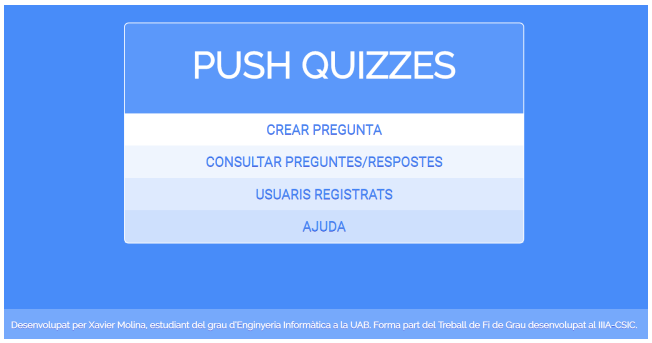

Fig. 10: Captura de la pantalla principal