

SaveKey, una aplicació per emmagatzemar contrasenyes

Josep M^a Viladegut Pelegrí

Postgrau de desenvolupament d'aplicacions per a dispositius mòbils

Carlos Caballero González

22/06/2015



Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FITXA DEL TREBALL FINAL

| | |
|----------------------------------|---|
| Títol del treball: | SaveKey, una aplicació per emmagatzemar contrasenyes |
| Nom de l'autor: | Josep M ^a Viladegut Pelegrí |
| Nom del consultor: | Carlos Caballero González |
| Data d'entrega (mm/aaaa): | 06/2016 |
| Area del Treball Final: | Aplicacions Android |
| Titulació: | <i>Postgrau de desenvolupament d'aplicacions per a dispositius mòbils</i> |

Resum del Treball (màxim 250 paraules):

El TFP tracta sobre el desenvolupament d'una aplicació per a mòbils, en concret, una aplicació que pugui emmagatzemar dades i, seguidament, consultar-les.

Aquesta informació provindrà de les dades que utilitza l'usuari per connectar-se o entrar en un lloc web o una aplicació, per exemple, per entrar a una xarxa social on s'ha d'introduir el correu electrònic i una contrasenya.

L'usuari podrà emmagatzemar les següents dades dins de l'aplicació:

- El nom del lloc web o de l'aplicació pel qual serveix la informació
- El nom d'usuari
- La contrasenya amb la que esta registrat l'usuari
- Comentaris, que poden ser opcionals

Es tracta d'una aplicació que guarda dades privades com són les contrasenyes, per tant, és important que estiguin xifrades en tot moment.

Una vegada es vagi a guardar la informació, aquesta serà xifrada prèviament al seu emmagatzematge.

Les dades s'emmagatzemaran en una base de dades local, és a dir, dins al mateix mòbil, la qual es podrà consultar sense necessitat de tenir cap connectivitat a una xarxa. Però també hi haurà l'opció de fer una còpia a un núvol (utilitzant *Parse*) per no perdre la informació en cas d'esborrar l'aplicació.

Per evitar que qualsevol persona pugui obrir l'aplicació i consultar les dades guardades, en el moment d'encendre l'aplicació, també es demanarà una contrasenya per poder entrar a la pantalla principal i veure la informació.

Abstract (in English, 250 words or less):

This FPP(Final Post-graduate Project) is about the development of a mobile application that can store data and, after that, obtain it again.

This data is actually the information used by the user to log into a website or some other application, for instance, a social network where the user needs to introduce his e-mail and password.

The user can store the following data in the application:

- The name of the website or application where the data needs to be used in.
- The user name.
- The password for the user name mentioned above.
- Some comments, which can be optional.

Since the application is used to store sensitive data, like passwords, the need of keep the data encrypted at any moment is highly relevant.

Once the information is ready to be stored, it will be encrypted just before storing it.

The data will be stored into a local database, that means the database is generated inside the user's own phone, and can be consulted without the need of being connected to any network. Nonetheless, the user has also the option of storing a backup copy of his information into a cloud system (using *Parse*), so it can be restored in case the application is uninstalled.

To avoid any other person opening the application and obtaining the user's personal information, a password will be required any time the application is opened in order to get to the main view and see all the stored information.

Paraules claus (entre 4 y 8):

Android, xifrat de dades, base de dades local, base de dades remota (Parse.com)

Índex

| | |
|---|-----------|
| 1. Introducció..... | 1 |
| 1.1. Context i justificació del Treball..... | 1 |
| 1.2. Objectius del Treball..... | 2 |
| 1.3. Enfocament i mètode seguit..... | 3 |
| 1.4. Planificació del Treball | 3 |
| 1.5. Breu resum de productes obtinguts..... | 4 |
| 1.6. Breu descripció dels altres capítols de la memòria | 5 |
| 2. Android | 6 |
| 2.1. Què és Android? | 6 |
| 2.2. Historial de versions..... | 7 |
| 2.3. Arquitectura | 8 |
| 2.3.1. Aplicacions | 9 |
| 2.3.2. Entorn de l'aplicació..... | 9 |
| 2.3.3. Biblioteques | 9 |
| 2.3.4. Runtime de Android | 9 |
| 2.3.5. Nucli Linux | 9 |
| 3. Tecnologia utilitzada..... | 10 |
| 3.1. Parse.com | 11 |
| 3.2. Ionic framework..... | 12 |
| 3.3. Appverse web | 14 |
| 4. Tipus d'aplicacions | 15 |
| 4.1. Aplicacions natives..... | 15 |
| 4.2. Aplicacions web | 15 |
| 4.3. Aplicacions híbrides | 17 |
| 5. Descripció de les funcionalitats i anàlisis | 19 |
| 5.1. Diagrama de casos d'ús..... | 19 |
| 5.2. Especificació del cas d'ús més significatiu | 21 |
| 5.3. Diagrama de flux | 22 |
| 5.4. Diagrama de classes..... | 23 |
| 5.5. Diagrama del model relacional de la base de dades..... | 25 |
| 5.5.1. Base de dades remota..... | 26 |
| 5.5.2. Base de dades local | 27 |
| 6. Disseny de la interfície | 28 |
| 6.1. Wireframe..... | 28 |
| 6.2. Wireflow | 32 |

| | |
|---|-----------|
| 7. Implementació | 33 |
| 7.1. Decisions | 33 |
| 7.1.1. Xifrat AES i Hash SHA256 | 33 |
| 7.1.2. Aplicació multiplataforma i aplicació web | 34 |
| 7.1.3. Més d'un usuari per dispositiu | 34 |
| 7.2. Desenvolupament de l'aplicació | 35 |
| 7.2.1. Fils d'execució (Multithreading) | 37 |
| 7.2.2. Ús de SharedPreferences | 38 |
| 7.2.3. Estat de la xarxa i <i>BroadcastReceiver</i> | 41 |
| 7.2.4. Aplicació multilingüe | 42 |
| 8. Aplicació final d'Android | 43 |
| 8.1. Pantalla de log in (pantalla inicial) | 43 |
| 8.2. Pantalla de registre | 48 |
| 8.3. Pantalla principal (llistat de valors) | 50 |
| 8.3.1. Consultar les dades | 52 |
| 8.3.2. Sincronitzar les dades | 54 |
| 8.4. Pantalla de creat/modificar dades | 56 |
| 9. Aplicació web | 58 |
| 9.1. Vista inicial | 58 |
| 9.2. Vista principal (llistat de claus) | 61 |
| 9.3. Afegir claus | 63 |
| 10. Aplicació final de Ionic | 64 |
| 10.1. Pantalla inicial, de login o registre | 64 |
| 10.2. Llistat de claus | 66 |
| 10.3. Afegir clau | 68 |
| 11. Conclusions | 69 |
| 12. Glossari | 70 |
| 13. Bibliografia | 71 |
| 14. Annexos | 74 |
| 14.1. Execució de l'aplicació web | 74 |
| 14.2. Execució de l'aplicació híbrida | 75 |
| 14.3. Execució de l'aplicació Android | 76 |
| 14.4. Vídeos demostratius | 76 |
| 14.4.1. Android | 76 |
| 14.4.2. Ionic | 77 |
| 14.4.3. Aplicació web | 77 |

Llista de figures

| | |
|---|----|
| Figura 1: Versions d'Android | 8 |
| Figura 2: Arquitectura Android..... | 8 |
| Figura 3: Execució d'una aplicació Ionic amb el navegador | 13 |
| Figura 4: Figura de "Understanding Your Mobile Application" | 17 |
| Figura 5: Diagrama de casos d'ús..... | 20 |
| Figura 6: Diagrama de flux | 22 |
| Figura 7: Diagrama UML complet..... | 23 |
| Figura 8: UML de la classe model | 24 |
| Figura 9: Pantalla de log in..... | 28 |
| Figura 10: Pantalla de registre | 28 |
| Figura 11: Pantalla de log in amb error | 29 |
| Figura 12: Llista de claus emmagatzemades | 29 |
| Figura 13: Pantalla principal sense resultats | 30 |
| Figura 14: Formulari per crear una clau | 30 |
| Figura 15: Detalls d'una clau | 31 |
| Figura 16: Formulari per editar una clau | 31 |
| Figura 17: Estructura clau-valor | 38 |
| Figura 18: Arxiu de Login creat per les <i>SharedPreferences</i> | 40 |
| Figura 19: Contingut del fitxer creat per les <i>SharedPreferences</i> | 40 |
| Figura 20: Contingut del fitxer creat per les <i>SharedPreferences</i> xifrat | 41 |
| Figura 21: Pantalla de registre amb botó desactivat (Android)..... | 42 |
| Figura 22: Pantalla de registre amb botó activat (Android) | 42 |
| Figura 23: Pantalla inicial de login (Android) | 44 |
| Figura 24: Pantalla de login amb missatge d'error (Android) | 44 |
| Figura 25: Pantalla de registre (Android)..... | 48 |
| Figura 26: Pantalla de registre amb error de contrasenyes diferents (Android) 49 | |
| Figura 27: Pantalla de registre amb error d'email incorrecte (Android) | 49 |
| Figura 28: Disseny d'un element de la llista | 50 |
| Figura 29: Pantalla principal sense dades (Android) | 51 |
| Figura 30: Pantalla amb dades (Android)..... | 51 |
| Figura 31: <i>AlertDialog</i> amb tota la informació de l'element (Android)..... | 53 |
| Figura 32: Edició d'un valor (Android) | 53 |
| Figura 33: <i>AlertDialog</i> per eliminar un valor (Android)..... | 53 |
| Figura 34: <i>ProgressDialog</i> de sincronització (Android) | 55 |
| Figura 35: Formulari per crear una nova clau (Android)..... | 56 |
| Figura 36: Error de camps buits (Android) | 56 |
| Figura 37: Pantalla de login (App web) | 59 |
| Figura 38: Formulari de registre (App web)..... | 60 |
| Figura 39: Error al login (App web)..... | 60 |
| Figura 40: Llistat de claus (App web) | 61 |
| Figura 41: Pantalla principal sense valors (App web)..... | 62 |
| Figura 42: Editar un valor (App web)..... | 62 |
| Figura 43: Formulari per afegir una nova clau (App web) | 63 |
| Figura 44: Pantalla de login (Ionic)..... | 65 |
| Figura 45: Pantalla de registre (Ionic) | 65 |
| Figura 46: Pantalla de login amb error (Ionic) | 65 |
| Figura 47: Llista de claus (Ionic)..... | 66 |

| | |
|---|----|
| Figura 48: Pantalla principal sense dades (Ionic) | 66 |
| Figura 49: Eliminar valors de la llista (Ionic) | 67 |
| Figura 50: Formulari per crear un nou valor (Ionic) | 68 |
| Figura 51: Editar/Consultar una clau (Ionic) | 68 |

Llista de taules

| | |
|--|----|
| Taula 1: Aplicacions semblants existents | 3 |
| Taula 2: Classificació de versions d'Android | 7 |
| Taula 3: Comparativa del tipus d'aplicacions. Font original: "Understanding Your Mobile Application" | 18 |

1. Introducció

1.1. Context i justificació del Treball

Actualment existeixen quantitat de llocs webs, aplicacions, ordenadors que es poden tenir a la feina, a casa, el portàtil, o inclús la configuració de la Wi-Fi de casa on per poder accedir-hi es necessita un usuari i una contrasenya. Hi ha tants llocs que pot ser un problema recordar-les totes, fins a l'extrem de que quan fa uns dies que no s'utilitza, es pot oblidar.

Personalment, fins ara no s'havia tingut mai aquest problema, però actualment, tant en el lloc de treball com en el mateix campus de la UOC, cada X mesos (normalment, cada 3) és obligatori canviar la contrasenya, tenint de pensar-ne una de diferent utilitzant paraules que no estiguin al diccionari, que tinguin majúscules i minúscules, símbols i altres caràcters extra, la qual cosa, són un problema de recordar.

També s'ha escollit aquest tema per l'afició a desenvolupar aplicacions per Android. Es va començar a estudiar el sistema operatiu durant els estudis en el grau d'enginyeria informàtica, on es va fer el primer contacte com a desenvolupador Android.

Seguidament, es va seguir estudiant el llenguatge i, a més a més, desenvolupar petites aplicacions per al dispositiu propi.

Unint el problema viscut amb l'afició, ha estat interessant fer aquest TFP, de forma que es segueix aprenent el llenguatge natiu de Android al mateix moment que es resolen problemes que tothom pot trobar-se.

1.2. Objectius del Treball

Els objectius són els següents:

- Implementar una aplicació Android per poder gestionar les contrasenyes de l'usuari:
 - Afegir dades
 - Modificar dades
 - Consultar dades
 - Eliminar dades
- Emmagatzemar les dades en una base de dades local
- Importar i/o exportar la informació a un Cloud
- Clau mestra per poder accedir al contingut de l'aplicació
- Tenir un disseny accessible i usable per facilitar l'ús a qualsevol usuari, evitant possibles problemes i malestar de les persones per no saber-ho fer servir.
- Xifrar la informació amb AES, de forma que, per molt que es pugui accedir a la base de dades, el seu contingut serà il·legible.
- Poder utilitzar l'aplicació sense necessitat de connexió a Internet, obtenint la informació de la base de dades del dispositiu.
- Implementar una aplicació web on poder gestionar la informació emmagatzemada al Cloud per l'usuari:
 - Afegir dades
 - Modificar dades
 - Consultar dades
 - Eliminar dades

1.3. Enfocament i mètode seguit

L'objectiu del projecte es desenvolupar una aplicació ja existent al mercat (Taula 1). A partir de l'anàlisi d'aquestes s'ha plantejat un projecte amb algunes funcionalitats similars i altres de noves, com pot ser el seu ús sense necessitat de connexió a xarxa.

| Aplicació | Sistema Operatiu | Característiques |
|-----------------------------|--|---|
| KEEPER | <ul style="list-style-type: none">• iOS• Android• Windows Phone• Blackberry | <ul style="list-style-type: none">• Suporta navegadors web• Autocompletat de formularis• Compartir dades |
| Google Authenticator | <ul style="list-style-type: none">• iOS• Android• Blackberry• Windows Phone (Authenticator) | <ul style="list-style-type: none">• Suporta navegadors web• Verificació d'usuari amb codi generat per la mateixa aplicació• Generació de codis aleatoris |
| LastPass | <ul style="list-style-type: none">• iOS• Android• Windows Phone• Blackberry | <ul style="list-style-type: none">• Suporta navegadors web• Autocompletat de formularis• Sincronització amb multidispositius• Contrasenya mestra• Contrasenyes amagades fins desbloquejar |

Taula 1: Aplicacions semblants existents

1.4. Planificació del Treball

Per crear la planificació s'ha generat un diagrama de Gantt, el qual, degut a la seva longitud, s'ha afegit en un fitxer apart per poder veure's millor [Adjunts\Gantt.png](#).

En aquest es mostra, separat per tasques i/o pràctiques, la durada de cadascuna amb la seva data d'inici i la seva data de fi.

1.5. Breu resum de productes obtinguts

El resultat del TFP ha estat una aplicació amb totes les funcionalitats/objectius aconseguits (indicats a l'Apartat 1.2).

També s'ha desenvolupat una aplicació web i una híbrida per no tenir la necessitat d'instal·lar l'aplicació i utilitzar-la des d'un navegador, i per no limitar-la solament als dispositius Android, ja que és multiplataforma.

Aquesta està feta amb Ionic[1], i no conté base de dades locals ja que tot ho fa directament amb la base de dades remota.

1.6. Breu descripció dels altres capítols de la memòria

El document que conté la memòria del treball de final de postgrau s'estructura de la següent forma:

- En el capítol 1 s'explica tota la introducció al treball explicant les motivacions per les quals s'ha desenvolupat aquesta aplicació i els objectius de l'aplicació.
- El capítol 2 es centra en Android i una mica de la seva història ja que és un dels temes principals del projecte i que avui en dia es troba per tot arreu.
- En el capítol 3 es descriuen les tecnologies utilitzades per dur a terme tot el treball i per poder crear tant l'aplicació Android, la web i la híbrida. Seguidament hi ha una petita explicació de les tecnologies més importants
- El capítol 4 compara el tipus d'aplicacions que es poden desenvolupar per a mòbils
- El capítol 5 conté tota la part de l'estudi de les funcionalitats que ofereix l'aplicació, és a dir, un estudi previ. Casos d'ús, diagrama de flux, diagrama de classe, etc.
- Al capítol 6 hi ha explicat el prototip amb el que es va basar l'aplicació, acompanyat d'imatges i amb un prototip interactiu (wireflow)
- En el capítol 7 es tracten els punts i decisions més importants durant el desenvolupament
- En el capítol 8, 9 i 10 s'explica en detall l'aplicació resultant d'Android, web i Ionic. Hi ha incloses captures de pantalles per mostrar el resultat final de l'aplicació.
- Finalment, en el capítol 11 s'expressen les conclusions que s'han tret després de dur a terme aquest treball i possibles línies de futur sobre l'aplicació, ja siguin millores o accions futures.

2. Android

2.1. Què és Android?

Android és un sistema operatiu basat amb el nucli de Linux dissenyat per dispositius mòbils amb pantalla tàctil, com telèfons intel·ligents (smartphone) o tabletas, desenvolupat per Android i seguidament comprat per Google.

Avui en dia, aquest sistema operatiu s'utilitza en gran varietat de dispositius, a part dels comentats anteriorment, també hi ha ordenadors, netbooks, rellotges de polsera i altres.

Està dissenyat per un baix consum i, principalment, per la integració de la xarxa implicant poder estar connectat a Internet sempre que es vulgui, juntament amb grans capacitats de memòria i de potència de processament.

El fet d'estar basat en Linux ajuda a obtenir un sistema operatiu lliure, gratuït i multi plataforma, i juntament amb l'existència d'eines de programació gratuïtes, fan que hi hagi una gran quantitat d'aplicacions disponibles. És a dir, qualsevol usuari pot programar en el sistema i introduir els programes al seu dispositiu sense necessitat de pagar res. Aquest fet és un gran avantatge pels fabricants i els desenvolupadors ja que d'aquesta forma, el cost per llençar una aplicació o un dispositiu es molt baix.

2.2. Historial de versions

Des de el seu llançament al 2008, Android ha anat arreglant el seu Sistema Operatiu agregant noves funcionalitats, corregint errors o actualitzant funcionalitats anteriors.

Una dada curiosa és que cada actualització rep, en Anglès, nom a diferents postres, i estan ordenats alfabèticament. Aquestes actualitzacions són:

| Plataforma | Nom Original | Traducció | API Level |
|---------------------|--------------------|--------------------|-----------|
| Android 1.0 | Apple Pie | Pastel de poma | 1 |
| Android 1.1 | Banana Bread | Pa de plàtan | 2 |
| Android 1.5 | Cupcake | Magdalena (muffin) | 3 |
| Android 1.6 | Donut | Rosquilla | 4 |
| Android 2.0/2.1 | Éclair | Pastel francès | 7 |
| Android 2.2 | Froyo | logurt gelat | 8 |
| Android 2.3 | Gingerbread | Pa de gingebre | 10 |
| Android 3.0/3.1/3.2 | Honeycomb | Panell de mel | 13 |
| Android 4.0 | Ice Cream Sandwich | Gelat de sandvitx | 15 |
| Android 4.1/4.2/4.3 | Jelly Bean | Gominola | 16/17/18 |
| Android 4.4 | KitKat | KitKat | 19 |
| Android 5.0 | Lollipop | Piruleta | 21 |
| Android 5.1 | Lollipop | Piruleta | 22 |
| Android 6.0 | M | - | - |

Taula 2: Classificació de versions d'Android

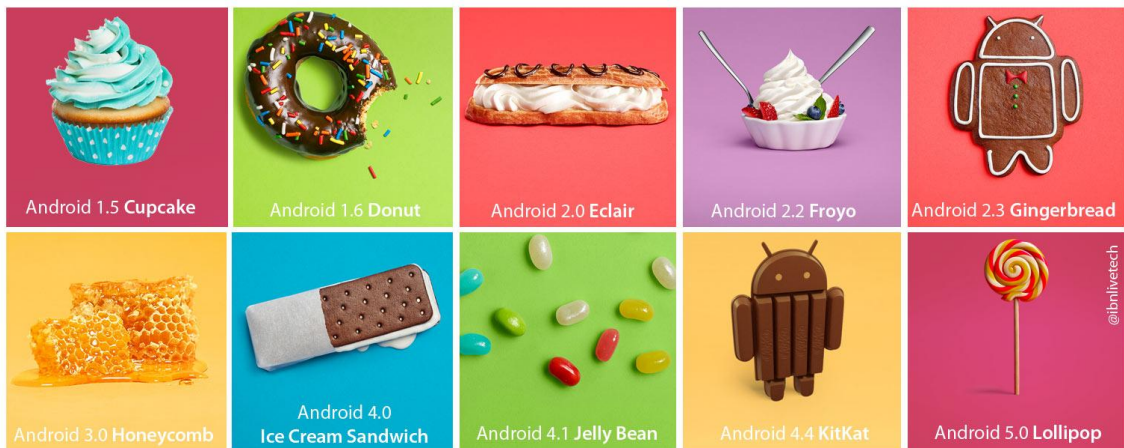


Figura 1: Versions d'Android

2.3. Arquitectura

L'arquitectura d'Android es pot dividir en cinc capes[2]:

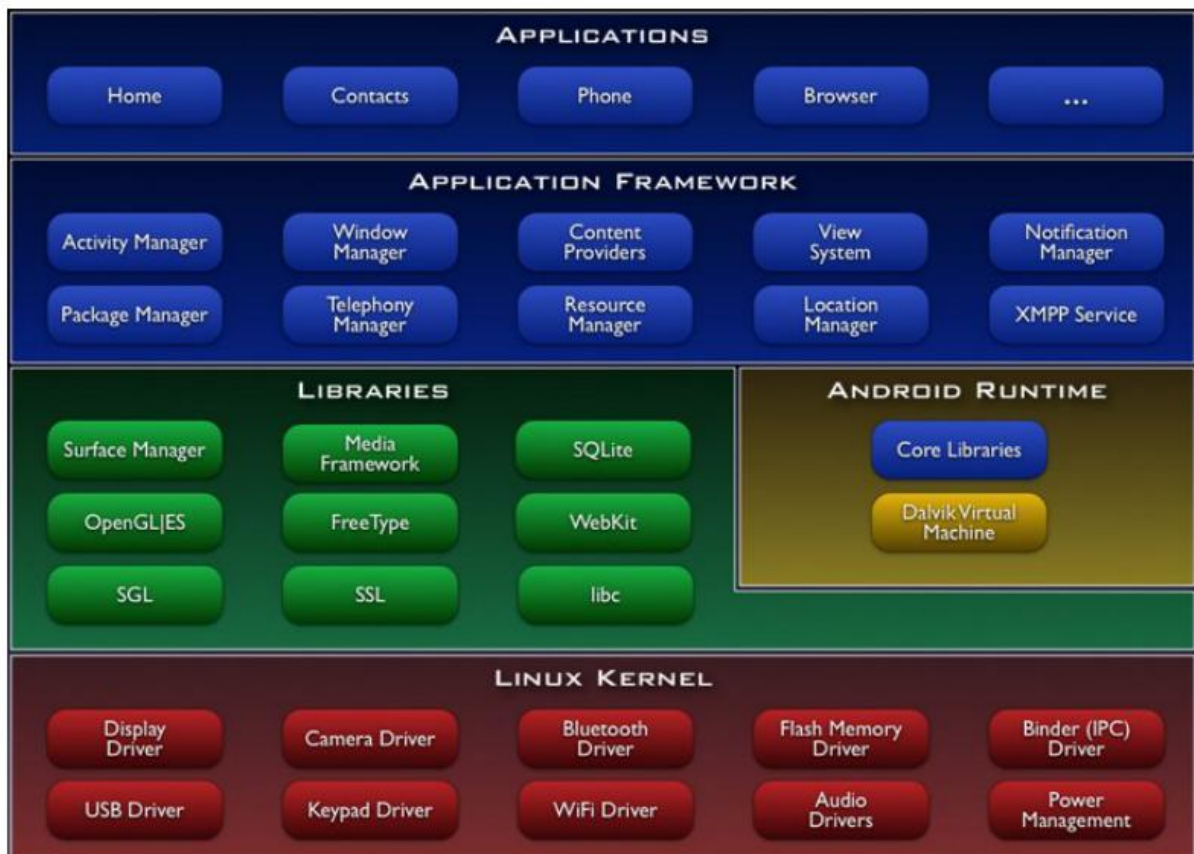


Figura 2: Arquitectura Android

2.3.1. Aplicacions

Aquesta capa inclou totes les aplicacions del dispositiu, ja siguin natives com instal·lades de fàbrica i de l'usuari, com per exemple: programes de SMS, calendari, mapes, client de correu electrònic, etc.

2.3.2. Entorn de l'aplicació

Proporciona una plataforma de desenvolupament lliure per aplicacions, i els desenvolupadors tenen total accés a ells.

Aquesta capa per simplificar la reutilització dels components; qualsevol aplicació pot publicar les seves capacitats i una altra aplicació pot utilitzar-les (subjecte a unes normes de seguretat).

2.3.3. Biblioteques

Android inclou un conjunt de biblioteques en C/C++ utilitzades en varis components del sistema. Moltes d'aquestes utilitzen projectes de codi obert. Algunes d'elles són: biblioteques de medis, biblioteques de gràfics, 3D, SQLite, i moltes altres.

2.3.4. Runtime de Android

Està basat amb el concepte de màquina virtual utilitzant Java. Donades les limitacions dels dispositius on té de córrer Android (poca memòria i processador limitat) no era possible utilitzar una màquina virtual Java estàndard. Google va crear una nova màquina anomenada Dalvik, que podria ser millor i evitar aquestes limitacions.

Android permet utilitzar múltiples màquines virtuals per cada aplicació de forma eficient.

Runtime també incorpora un conjunt de llibreries base les quals contenen la majoria de les funcions disponibles de les llibreries de Java.

2.3.5. Nucli Linux

El nucli de Android està format pel sistema operatiu Linux versió 2.6. Aquesta capa proporciona serveis com la seguretat, el control de memòria, el multiprocés, la pila de protocols i el suport de drivers pels dispositius.

Aquest nivell actua com a capa d'abstracció entre el hardware i la resta de la pila de software.

3. Tecnologia utilitzada

- Android SDK: és una eina de desenvolupament d'aplicacions per Android amb la qual, apart de poder crear aplicacions, també es pot executar emuladors del sistema Android de qualsevol versió, es pot testejar el codi i depurar-lo.
- Eclipse: entorn de desenvolupament de codi obert multi plataforma que permet la programació amb llenguatges com *Java*, *C*, *C++*, *Python*, etc. A més a més es poden instal·lar plugins, com pot ser el *Android Development Tools* que integra la programació Android dins de l'entorn.
- Android Studio: entorn de desenvolupament oficial per desenvolupar aplicacions Android.
- Brackets: editor de text *open source* dissenyat per crear webs. Juntament amb la quantitat de plugins que disposa, es pot convertir en un entorn de desenvolupament molt potent.
- Parse.com: plataforma web que pot servir com a backend de qualsevol aplicació.
- Genymotion: emulador Android molt potent. Més ràpid que els emuladors del SDK, i amb més funcionalitats que aquest sempre i quan es pagui la llicència (24,99€/mes).
- Ionic: framework gratuït i *open source* per desenvolupar aplicacions híbrides basades en HTML5, CSS3 i/o Sass i AngularJS.
- Appverse web: framework *open source* que incorpora múltiples tecnologies de codi obert per desenvolupar webs amb HTML, AngularJS i Sass.
- Bitbucket.org: gestor de versions amb el qual es poden crear projectes privats.
- CryptoJS: llibreria de xifrat per Javascript que s'utilitza en l'aplicació web i Ionic.

3.1. Parse.com

Parse[3] és un *Backend as a Service (BaaS)*, és a dir, un servei per vincular les aplicacions amb un Cloud que té unes funcionalitats específiques que són:

- Emmagatzematge i gestió de dades
- Notificacions push
- Analítiques de les dades
- ...

Ofereix un SDK per poder afegir-lo al projecte, el qual pot ser de qualsevol de les plataformes:

- Mòbils:
 - iOS
 - Windows Phone
 - Android
 - Unity
 - Xamarin
 - React
- Escriptori + web
 - OSX
 - Windows
 - Javascript
 - Unity
 - PHP
 - .NET

I una vegada integrat, es poden utilitzar totes les funcionalitats de les que disposa.

És gratuït però amb certs límits:

- 30 peticions per segon com a màxim
- 20GB d'emmagatzematge
- 2TB de transferència d'arxius
- 1.000.000 receptors de notificacions push

Resumidament, és una forma molt simple d'afegir un backend amb totes les funcionalitats dins d'una aplicació que ho necessiti.

3.2. Ionic framework

Ionic[1] és un nou framework per crear aplicacions híbrides basades en HTML5, CSS i Javascript. Exactament utilitza Sass[4] com a complement d'estils, i per la part del model, utilitza AngularJS[5]. La base de tot el framework és "Apache Cordova"[6] amb el que es pot aprofitar per compilar l'aplicació amb *Phonegap Build* i tots els plugins que ofereix per utilitzar el hardware del dispositiu.

La principal diferència és que s'ha extret el *jQuery* que utilitzava Phonegap en el seu codi.

Per instal·lar-lo es necessari tenir *NodeJs*[7], amb el qual seguidament es procedeix a fer la instal·lació:

```
$ npm install -g cordova ionic
```

Amb això fet, es pot crear un projecte

```
$ ionic start <nom_aplicació> <tipus_template>
```

Per provar-la, s'ha de ficar dins la carpeta del projecte i executar-la de la forma preferida ja que hi ha varies formes (com a exemple s'utilitza el sistema operatiu Android, tot i que és el mateix pels altres sistemes):

- En un emulador Android: primer s'ha d'afegir la plataforma Android dins al projecte (es necessari tenir el SDK d'Android instal·lat) i seguidament ja es podrà executar

```
$ cd helloWorld  
$ ionic platform add android
```

- En un dispositiu físic Android: igual que abans

```
$ cd helloWorld  
$ ionic platform add android
```

- Al navegador: s'obrirà un servidor local on s'executarà l'aplicació. Cada vegada que hi hagi un canvi, aquest ho detectarà i actualitzarà l'aplicació al navegador.

```
$ cd helloWorld
```

El resultat de l'execució amb navegador és el que es pot veure a la Figura 3.

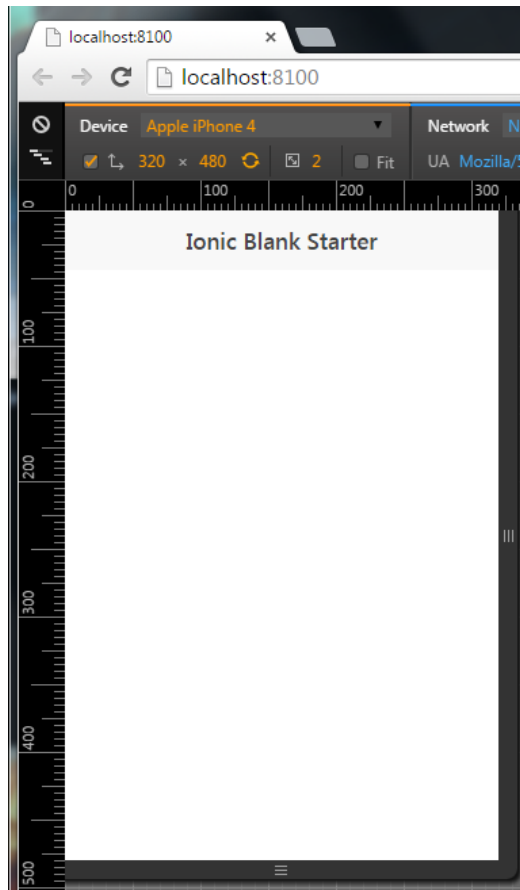


Figura 3: Execució d'una aplicació Ionic amb el navegador

3.3. Appverse web

Appverse[8] web és una plataforma de desenvolupament d'aplicacions web de codi obert desenvolupat per programar webs d'una forma simple de cara al programador. Utilitza les tecnologies de HTML5, CSS3 i Javascript, les mateixes que per fer una aplicació híbrida, per això existeix *Appverse Mobile*, la única diferència és que aquest segon incorpora les eines necessàries per utilitzar tot el hardware i les funcionalitats del dispositiu.

La principal característica és que, quan es comença un projecte, tota l'estructura d'aquest ja esta creada i no fa falta anar afegint complements (sempre i quan no siguin essencials), és a dir, no cal afegir coses com pot ser: crear un fitxer executable de Javascript, afegir un instal·lador de dependències, etc.

En resum, és un *boilerplate* de HTML5 amb certs complements afegits, el qual es pot descarregar lliurement i començar a escriure codi.

4. Tipus d'aplicacions

4.1. Aplicacions natives

Les aplicacions natives són aquelles que es desenvolupen per un sistema operatiu concret, sense possibilitat de traspasar-lo a un altre, utilitzant un llenguatge propi del SO i amb el seu SDK.

Per exemple, si es fa una aplicació Android, la programació es farà amb llenguatge Java i sol es podrà utilitzar en dispositius amb el mateix sistema.

Els avantatges que té aquest tipus d'aplicació són:

- Es pot accedir a totes les funcionalitats del dispositiu, com poden ser el GPS, els contactes, la càmera, etc.
- El rendiment és més elevat.

Els inconvenients:

- Necessitat de programadors amb coneixements en Java i en Android per poder aprofitar la màxim totes les funcionalitat.
- Una aplicació serveix per un únic sistema, pel que si es vol en un altre, serà necessari desenvolupar una altra aplicació amb un altre llenguatge i s'haurà de pagar com una aplicació nova. A més a més, es necessitarà un altre programador amb els coneixements pel nou llenguatge.

4.2. Aplicacions web

Les aplicacions web són les que estan preparades per executar-se des del navegador d'Internet del dispositiu. És una pàgina web (normalment programada en HTML5 i Javascript) a la que s'ha afegit els complements necessaris per semblar una aplicació nativa tot i que no fa falta instal·lar-la.

Els avantatges són:

- Són multiplataforma ja que estan desenvolupades com una web, el que significa que es pot utilitzar des de qualsevol dispositiu.
- Una sola aplicació per tots els sistemes. No és necessari crear aplicacions diferents per cada sistema operatiu.

- Un sol programador per fer la web, sense necessitat d'un programador diferent per cada sistema
- Es pot actualitzar la web sense que ningú es vegi afectat directament necessitant descarregar la nova aplicació
- No fa falta distribuir-la per les tendes d'aplicacions i pagar els costos de registre.

Els inconvenients:

- Les interfícies són més pobres ja que un mateix disseny afecta a tots els dispositius
- Més lent i necessitat de xarxa
- Quasi que no té accés a les funcionalitats del dispositiu
- No es poden crear notifikacions

4.3. Aplicacions híbrides

Les aplicacions híbrides són una unió de les natives i de les web, intentant aprofitar la facilitat de programació en web i l'ús en multiplataforma, i les prestacions del dispositiu amb natiu.

Tenen un codi escrit en HTML5 i JavaScript però amb mètodes extra per poder accedir a les funcionalitats del dispositiu, com si fos una aplicació nativa.

Els avantatges són:

- Una sola aplicació per tots els sistemes operatius
- Accés a les funcionalitats del dispositiu
- Fàcil desenvolupament
- Un únic programador amb coneixements de web per desenvolupar una aplicació pels diferents SO

Els inconvenients:

- No són tan ràpides com poden ser les natives

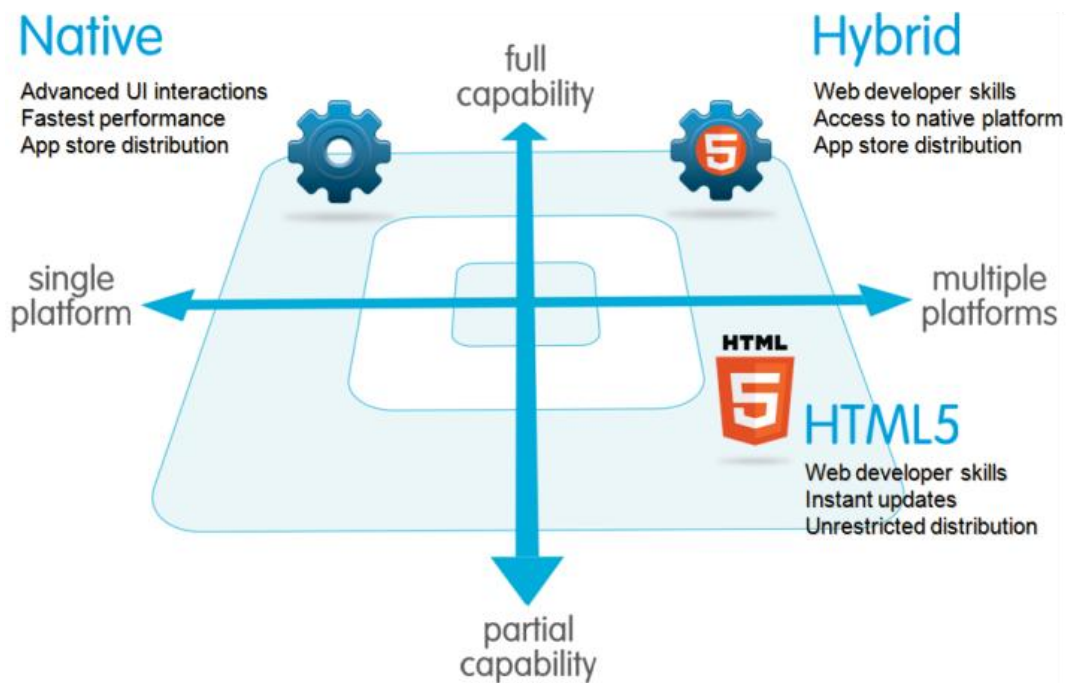


Figura 4: Figura de "Understanding Your Mobile Application"

| | Native | HTML5 (web) | Hybrid |
|---------------------------|---------------------|------------------------|--------------------------------|
| App Features | | | |
| Graphics | Native APIs | HTML, Canvas, SVG | HTML, Canvas, SVG |
| Performance | Fast | Slow | Slow |
| Native look and feel | Native | Emulated | Emulated |
| Distribution | Appstore | Web | Appstore |
| Device Access | | | |
| Camera | Yes | No | Yes |
| Notifications | Yes | No | Yes |
| Contacts, calendar | Yes | No | Yes |
| Offline storage | Secure file storage | Shared SQL | Secure file system, shared SQL |
| Geolocation | Yes | Yes | Yes |
| Gestures | | | |
| Swipe | Yes | Yes | Yes |
| Pinch, spread | Yes | No | Yes |
| Connectivity | Online and offline | Mostly online | Online and offline |
| Development skills | ObjectiveC, Java | HTML5, CSS, Javascript | HTML5, CSS, Javascript |

Taula 3: Comparativa del tipus d'aplicacions. Font original: "Understanding Your Mobile Application"

5. Descripció de les funcionalitats i anàlisis

5.1. Diagrama de casos d'ús

En la Figura 5 es pot veure el diagrama de casos d'ús de l'aplicació que s'està desenvolupant, la qual consta de sis funcionalitats principals.

La primera és la del registre i log in. Sense passar per aquesta, no es poden dur a terme les altres, per tant, es pot dir que hi ha la pre-condició de que si l'usuari no s'ha loguejat, no pot entrar a la part principal de l'aplicació. En cas de ser un usuari nou, s'haurà de registrar complimentant un formulari.

Una vegada s'ha superat la primera funcionalitat, es mostra la llista de claus emmagatzemades al dispositiu en la pantalla principal, des de la qual es pot accedir a la resta de funcionalitats.

Es poden crear noves claus a través d'un formulari, on una vegada esta omplert, es guarda la informació a la base de dades local.

Aquestes claus guardades, es poden consultar per veure la informació més detallada, esborrar i modificar. Per modificar es mostra un formulari com l'anterior amb el qual es poden editar les dades.

Finalment, totes les claus guardades es poden sincronitzar amb un Cloud de forma que des de l'aplicació web es podran consultar les dades que hi ha, i també, des d'altres dispositius es podrà recuperar tota la informació.

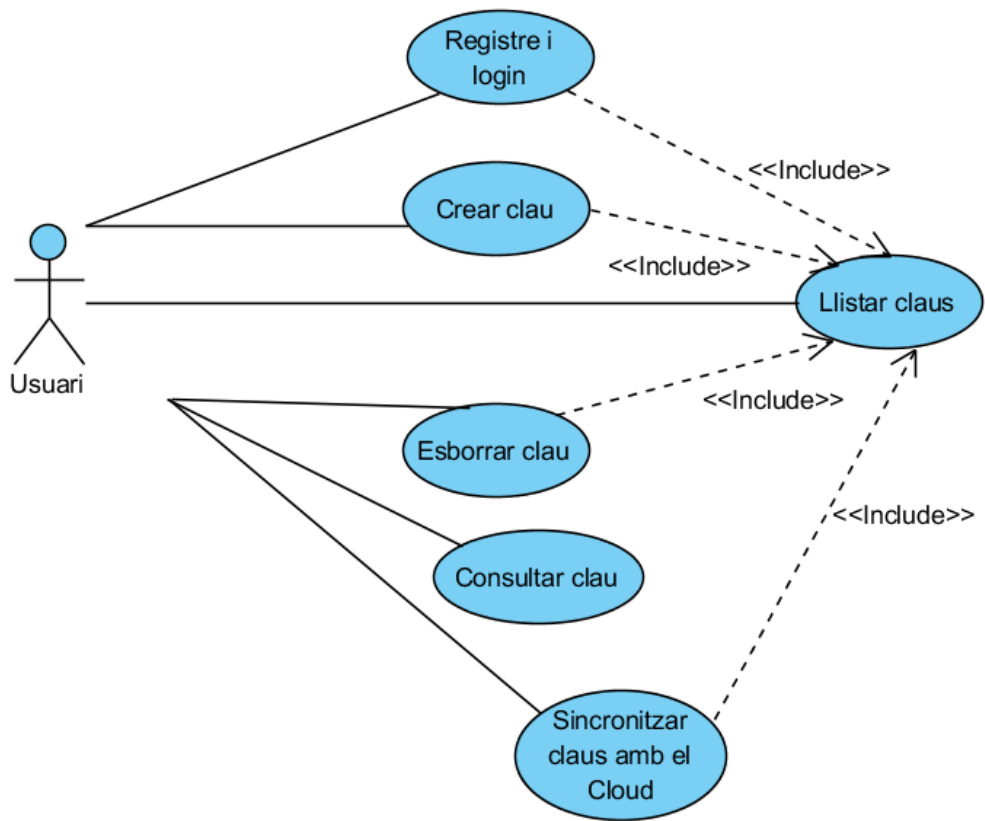


Figura 5: Diagrama de casos d'ús

5.2. Especificació del cas d'ús més significatiu

Cas d'ús: Sincronitzar claus amb el Cloud

Actors: Principal: Usuari

Propòsit: Sincronitzar la informació de la base de dades local amb la emmagatzemada al Cloud

Descripció: Un usuari executa l'aplicació per consultar les seves dades i gestionar-les. Vol emmagatzemar tota la informació al Cloud pel que pitja el botó de "Sincronització", llavors el dispositiu s'encarrega d'agafar totes les dades de la base de dades local que no han estat sincronitzades prèviament ni modificades una vegada sincronitzades, i les envia cap al Cloud, el qual, solament s'encarrega d'emmagatzemar la informació. Amb les dades que han estat modificades després de la seva sincronització, l'aplicació s'ocupa de buscar la mateixa informació al Cloud per obtenir-la (utilitzant el id de la dada), modificar-la i tornar-la a guardar amb els nous valors.

Un cop finalitzat tot el procés, es mostrarà un text a la pantalla indicant que la sincronització s'ha completat o, en cas de error, s'indicarà que hi ha hagut un error.

Errors:

- No hi ha connexió a Internet per fer la sincronització
- No hi ha dades emmagatzemades a la base de dades local
- No hi ha dades emmagatzemades a la base de dades remota

5.3. Diagrama de flux

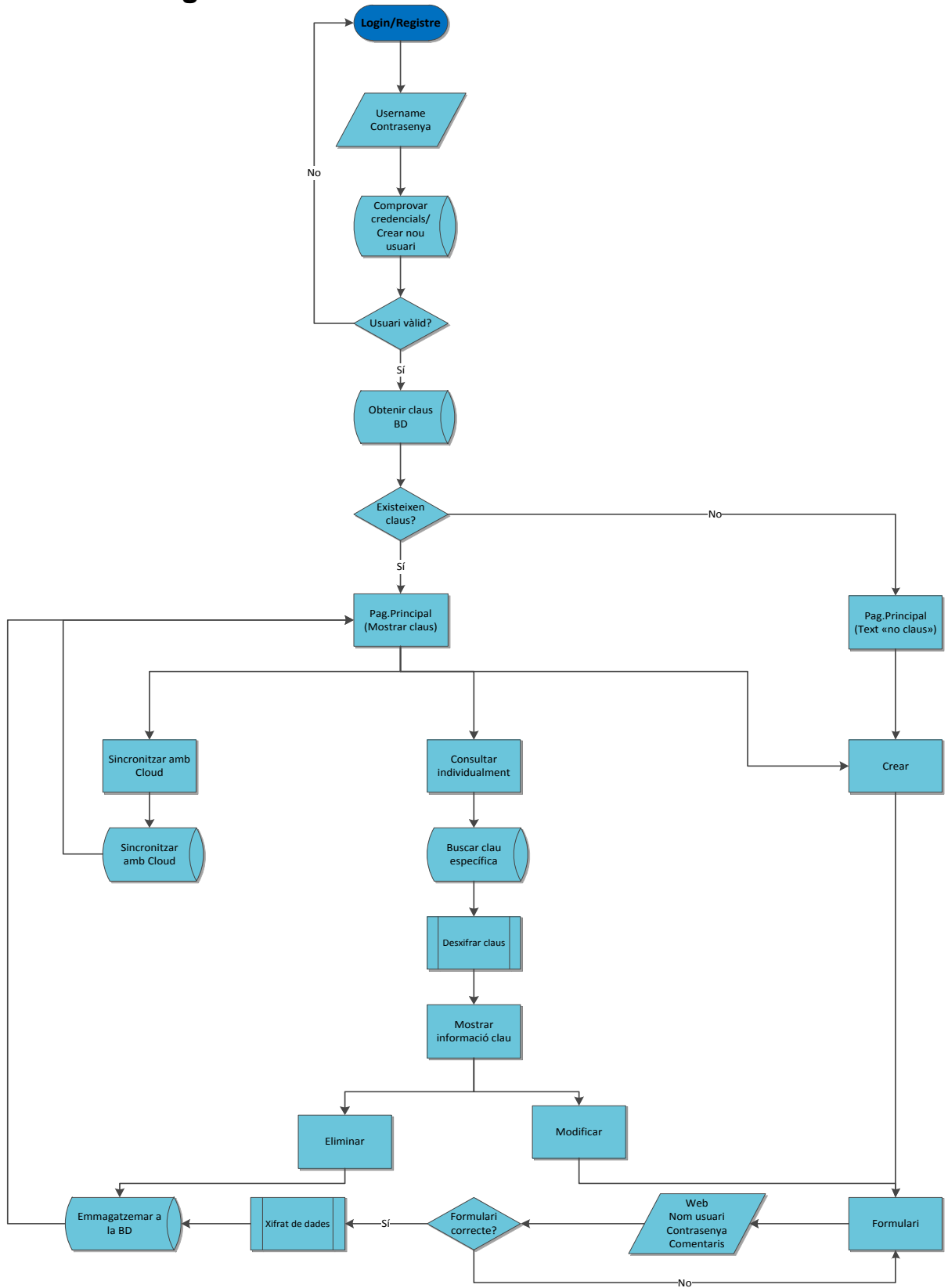


Figura 6: Diagrama de flux

5.4. Diagrama de classes

En la Figura 7 es pot veure el diagrama de classes resultant de l'aplicació que s'ha creat.

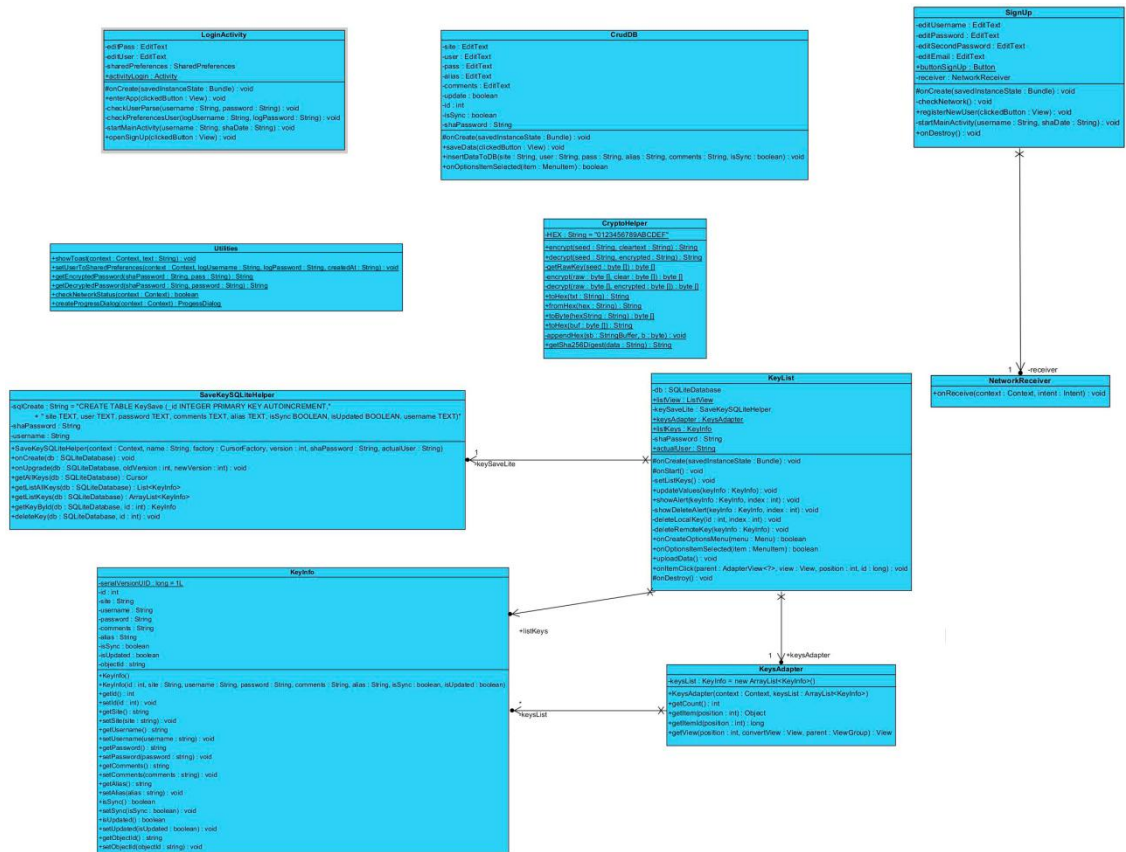


Figura 7: Diagrama UML complet¹

La classe model del projecte és la *KeyInfo*, vista en la Figura 8, i és la que representa un objecte de la base de dades amb tots els seus atributs:

- id: identificador de l'element
- lloc
- nom d'usuari
- contrasenya
- comentaris
- àlies

¹ Per consultar la imatge en forma completa, mirar en l'arxiu "[Adjunts\Diagrama de classes.png](#)"

- *isSync*: valor que indica si l'element s'ha sincronitzat amb la base de dades remota
- *isUpdated*: valor que indica si l'element s'ha modificat per saber que quan es faci la sincronització, s'ha d'actualitzar l'objecte a la base de dades remota
- *objectId*: identificador que dona la base de dades remota a l'element

A més a més conté totes les operacions getters i els setters per poder obtenir o guardar valors referents a l'objecte.

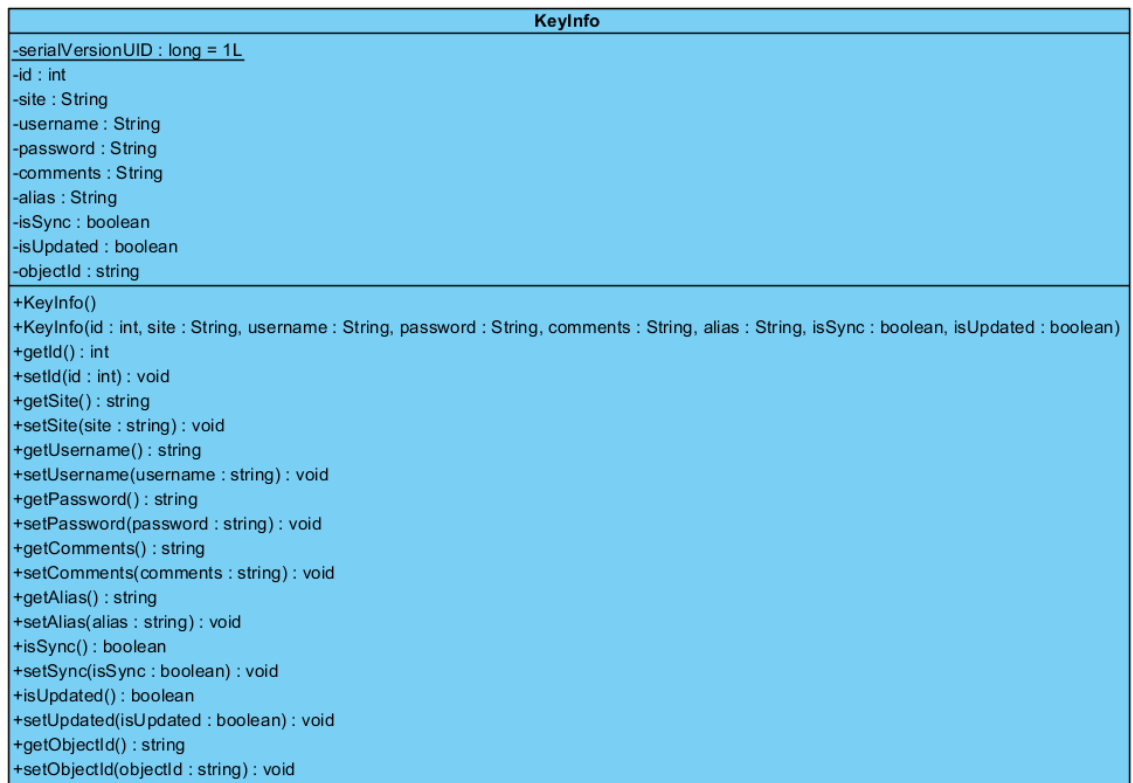


Figura 8: UML de la classe model

Les classes d'interacció amb l'usuari:

- LoginActivity
- SignUp
- KeyList
- CrudDB

són pantalles de l'aplicació, és a dir, són les classes que mostraran una interfície a l'usuari i controlaran totes les accions que es puguin dur a terme, per exemple, accions en pitjar un botó.

Les classes de suport són:

- Utilities
- CryptoHelper

En el diagrama es pot veure que només tenen operacions i que quasi no tenen cap atribut. El motiu és que han estat creades per ajudar a no tenir tants mètodes dins de les classes d'interacció amb l'usuari.

Es pot veure que la classe *SignUp* utilitza la *NetworkReceiver*, la qual és un servei que està atent per veure si canvia l'estat de la xarxa i poder dur a terme accions quan passi això.

Finalment, la classe *KeyList* es relaciona amb:

- *KeysAdapter*: s'encarrega de crear la llista de claus i mostrar-les d'una forma personalitzada. Es podria dir que és un adaptador entre una llista d'objectes i la llista que es mostrarà a la pantalla.
- *SaveKeySQLiteHelper*: classe que dur a terme tota la connexió amb al base de dades local (la crea i l'obra, a més a més d'afegir algun mètode per obtenir la llista de valors que conté).
- *KeyInfo*: classe model comentada uns punts abans.

5.5. Diagrama del model relacional de la base de dades

El model relacional està dividit en dues parts, una base de dades remota i una de local, degut a que l'aplicació permet la sincronització, deixant que totes les dades guardades al dispositiu es copiïn a un Cloud.

D'aquesta forma, per exemple, en cas de canviar de dispositiu, només farà falta pujar les dades d'un dispositiu i descarregar-se-les des de l'altre, sense necessitat d'estar introduint les dades manualment una a una.

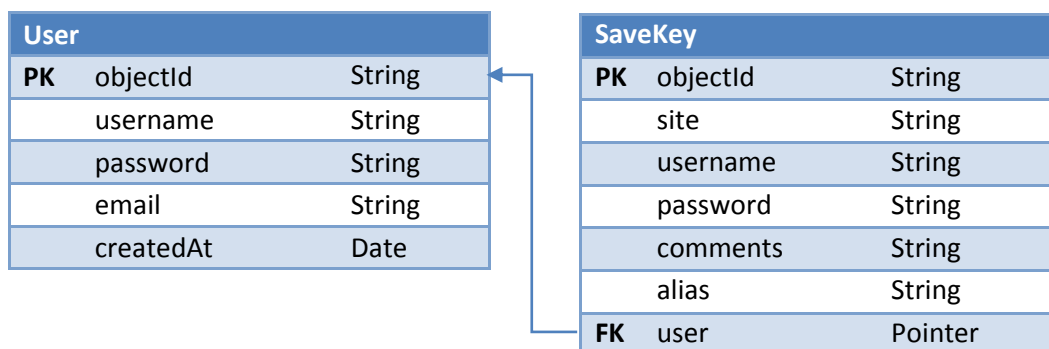
5.5.1. Base de dades remota

Aquesta base de dades el localitza a la web de *Parse.com* en la qual s'ha creat un projecte per poder-hi emmagatzemar les dades provinents de l'aplicació Android.

Per defecte, l'aplicació web ja ofereix unes taules creades que són la d'usuaris i la de rols, amb unes columnes pròpies, però solament s'ha aprofitat la primera. A més a més, també dona molts serveis implementats amb els que gestionar els valors sobre la taula, com per exemple, el registre o login, i amb els quals des d'Android es poden utilitzar i així evitar línies de codi comprovant si l'usuari s'ha registrat, si hi ha un altre usuari amb les mateixes dades, etc.

Per guardar les dades, s'ha creat una taula *SaveKey* amb tots els atributs i, a més a més, una referència cap a l'usuari del qual pertany aquell valor.

El resultat és el següent:



5.5.2. Base de dades local

A diferència de l'anterior, en la base de dades local no fa falta utilitzar una taula expressament per usuaris, ja que no i pot haver dos usuaris amb el mateix nom degut a que *Parse* evita que puguin haver-hi repeticions d'usuaris. Per tant, només es guarda el nom de l'usuari (penúltim camp).

També hi ha tres camps nous dins la taula que són els de:

- *isSync*: valor que indica si l'element s'ha sincronitzat
- *isUpdated*: valor que indica si l'element s'ha modificat per saber que quan es faci la sincronització, s'ha d'actualitzar l'objecte a la base de dades remota
- *objectId*: identificador que dona la base de dades remota a l'element. Aquest es *null* fins que no es sincronitza amb el Cloud, ja que és un valor generat en ell.

Finalment, la taula és:

| SaveKey | |
|---------------|---------|
| PK _id | Int |
| site | String |
| user | String |
| password | String |
| comments | String |
| alias | String |
| isSync | Boolean |
| isUpdated | Boolean |
| username | String |
| objectId | String |

6. Disseny de la interfície

6.1. Wireframe

Amb la Figura 6 i les figures de l'apartat de Disseny de la interfície es podrà detallar més clarament la navegació dins de l'aplicació, juntament amb els processos que es van duent a terme (es pot veure la imatge del diagrama de flux al fitxer "[Adjunts\DiagramaFlux.pdf](#)", i les del wireframe dins la carpeta "[Adjunts\Wireframe](#)").

Començant des del log in (Figura 9) o registre (Figura 10), es demana al usuari que introdueixi un nom d'usuari i una contrasenya, els quals a continuació seran validats amb les credencials emmagatzemades al dispositiu. En cas de no ser correctes, es mostrarà un text com el de la Figura 11 indicant que hi ha hagut un error de dades. En cas de ser correctes, s'entrarà a la funcionalitat de llistar les claus, les quals són obtingudes de la base de dades i es mostren uns camps significatius, tal i com es pot veure en la Figura 12.

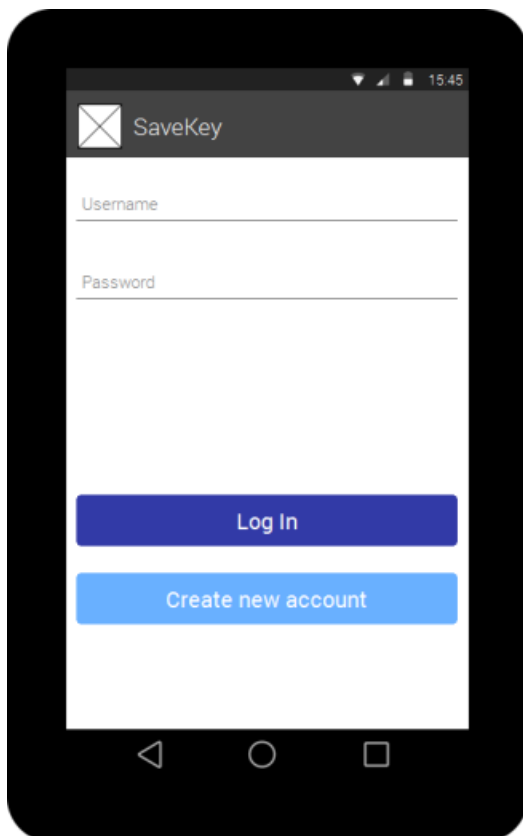


Figura 9: Pantalla de log in

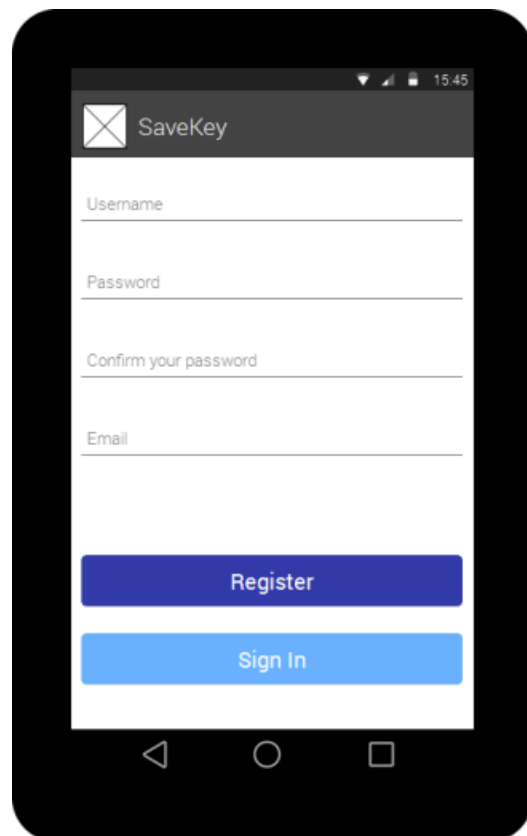


Figura 10: Pantalla de registre

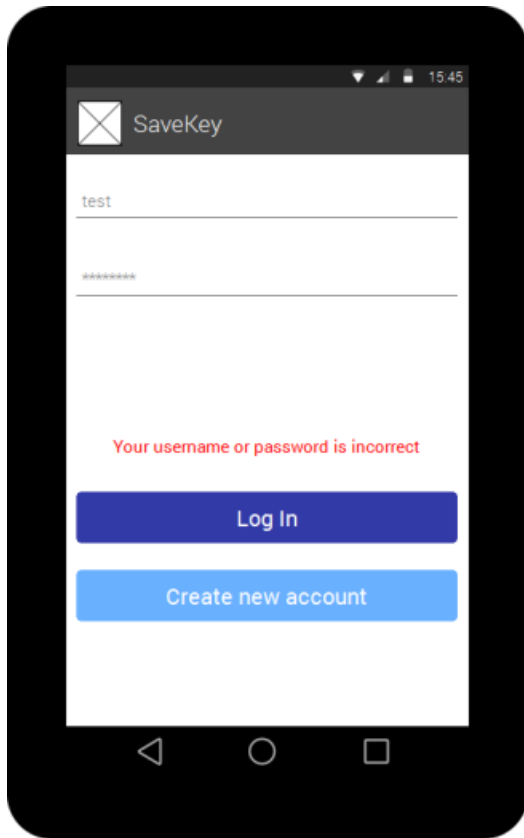


Figura 11: Pantalla de log in amb error

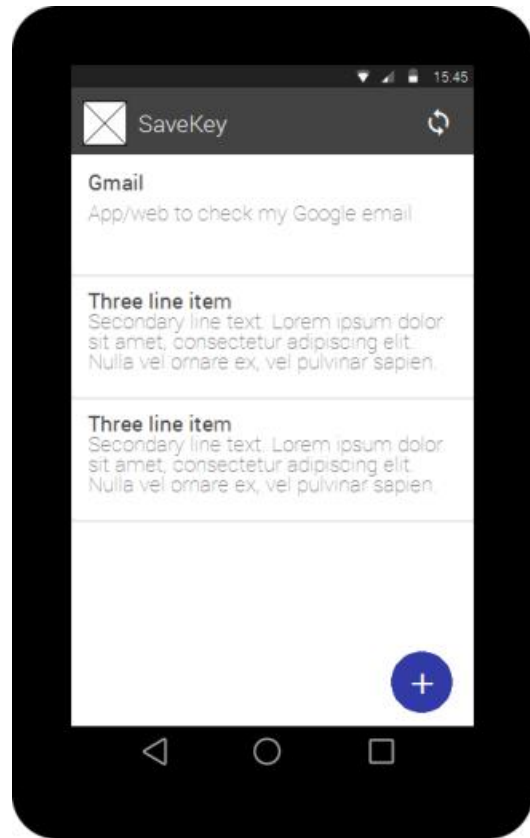


Figura 12: Llista de claus emmagatzemades

Si no existeix cap dada, es veurà un text com el de la Figura 13, però es podrà accedir al formulari de la Figura 14 amb el que, una vegada omplert i pitjat el botó de guardar, es comprovarà que els camps no estiguin buits. Si es supera la validació, les dades es xifran per evitar que qualsevol persona pugui veure la informació que hi ha guardada a la base de dades, i seguidament seran emmagatzemades.

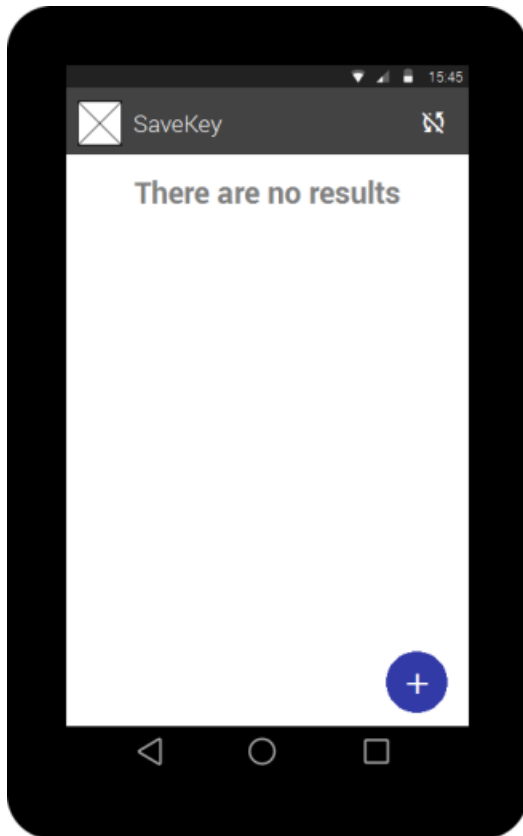


Figura 13: Pantalla principal sense resultats

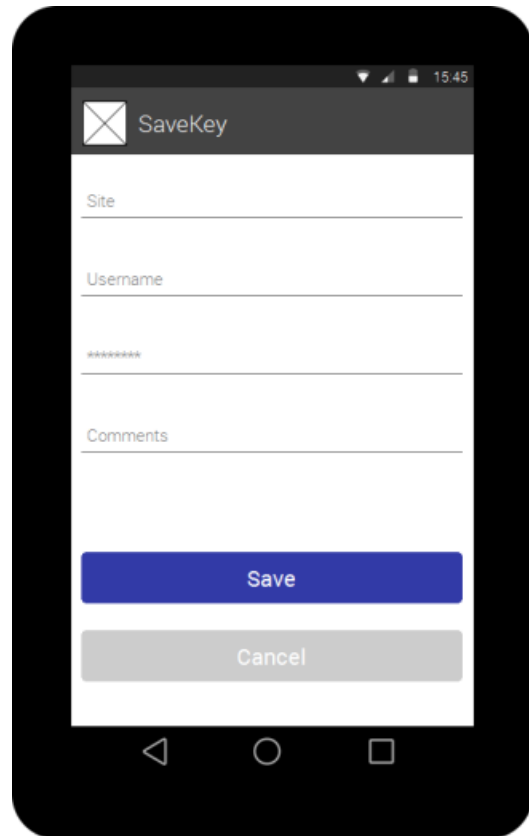


Figura 14: Formulari per crear una clau

Una vegada guardat, es tornarà a la pantalla principal, la qual ja tindrà claus emmagatzemades i ja es podrà consultar la informació detallada de cada clau. Per fer-ho, l'aplicació s'encarregarà d'obtenir tota la informació emmagatzemada de la clau, desxifrar-la i mostrar-la, tal i com es veu a la Figura 15.

Des de la informació detallada es podrà editar i/o esborrar els valors de la dada. En cas d'esborrar-la, l'aplicació la eliminarà de la base de dades i tornarà a la pàgina principal. En cas d'editar-la, es mostrarà un formulari amb el que es podrà modificar la informació i guardar-la (Figura 16). Igual que quan es crea una clau, la informació serà xifrada i seguidament emmagatzemada.

Per últim, tal i com esta comentat a l'apartat de "Diagrama de casos d'ús", es pot sincronitzar la base de dades local amb una base de dades remota utilitzant el botó de dalt a la dreta de la Figura 12 (les dues fletxes en forma de cercle), de forma que l'aplicació envia totes les dades que no hagin estat sincronitzades ni modificades després de ser sincronitzades, de forma que sol fa falta emmagatzemar-les. En cas de que hagi estat modificada després de la seva sincronització, serà necessari buscar aquella dada dins el Cloud, i actualitzar-la a la última versió.

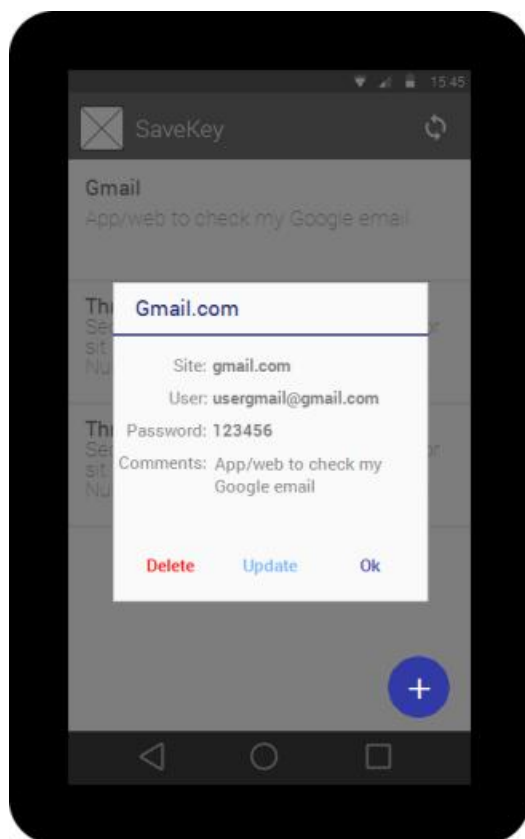


Figura 15: Detalls d'una clau

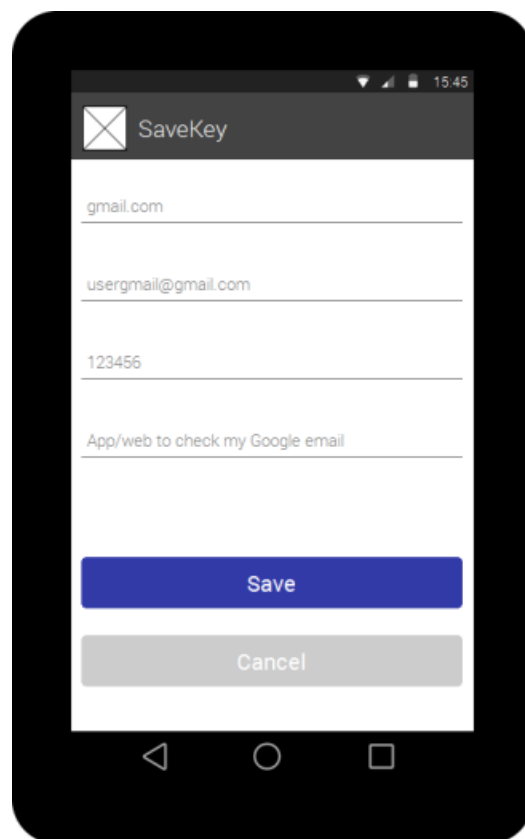


Figura 16: Formulari per editar una clau

6.2. Wireflow

El prototip interactiu es pot consultar utilitzant l'enllaç públic² o directament l'arxiu "[Adjunts\SaveKey - PrototipInteractiu.vp](#)", el qual serà necessari obrir amb el programa "Justinmind"[10].

² <https://www.justinmind.com/usernote/tests/15939387/15939424/15939434/index.html>

7. Implementació

7.1. Decisions

7.1.1. Xifrat AES i Hash SHA256

Pel xifrat de les dades es va utilitzar l'esquema de xifrat per blocs anomenat AES[11]. Aquest necessita una contrasenya la qual al principi s'utilitzava el hash de la contrasenya del usuari que entrava a l'aplicació. És a dir, primer s'obtenia el resultat d'aplicar l'algoritme SHA256[12] a la clau mestra de l'usuari i seguidament, amb el resultat es feia el xifrat AES:

$$\begin{aligned} shaPass &= Sha256(contrasenya_mestra) \\ xifratAES &= AES(clau, text) = AES(shaPass, text) \end{aligned}$$

On text era la cadena de caràcters a emmagatzemar, és a dir, la contrasenya de la clau que s'estava emmagatzemant ja fos perquè s'acabava de crear, o perquè s'havia modificat.

Seguidament es va pensar que, en cas que en un futur es fes una pantalla d'usuari on pogués editar les seves dades, si canviava la seva pròpia contrasenya general, el xifrat ja no serviria ja que el resultat del *Sha256* seria diferent al anterior.

Finalment es va decidir utilitzar una data com a xifratge, exactament la data de registre del usuari juntament amb el hash *Sha256* ja que aquesta mai serà modificada ni es podrà modificar.

$$\begin{aligned} shaDate &= Sha256(data_registre) \\ xifratAES &= AES(clau, text) = AES(shaDate, text) \end{aligned}$$

7.1.2. Aplicació multiplataforma i aplicació web

En un principi es volia crear una aplicació web per poder gestionar les dades sense necessitat del dispositiu o de l'aplicació. Aquest portal estaria fet amb *AngularJS*, *CSS3* i *HTML5*. El resultat es pot veure a l'apartat de "Aplicació web"

Però, per que no aprofitar-ho i utilitzar-ho per a fer una aplicació multiplataforma ja que el codi és quasi igual i a més a més, es podrà utilitzar a diferents sistemes operatius? El framework *Ionic* utilitza els mateixos complements que la web, per tant, solament feia falta adaptar les pantalles i afegir les possibles funcionalitats de més que pogués tenir el dispositiu.

El resultat es pot veure a l'apartat de "Aplicació final de Ionic"

7.1.3. Més d'un usuari per dispositiu

Al principi es va pensar de limitar l'aplicació a un dispositiu de forma que solament es pogués entrar amb unes credencials i solament hi hagués les dades d'aquell usuari.

Només feia falta limitar l'entrada d'usuaris a l'aplicació, mirant que si ja hi havia *SharedPreferences*, no es podria entrar. Tampoc no hagués fet falta el camp "usuari" de la taula *SaveKey* de la base de dades local, ja que s'utilitza per mostrar solament les dades de l'usuari que ha entrat.

Però es va pensar que, les aplicacions les té d'utilitzar qui vulgui i des d'on vulgui, ja sigui des d'una tablet o des d'un mòbil. Per exemple, en cas que una parella comparteixin una tablet, no podran separar les seves comptes ja que el primer que utilitzi l'aplicació, serà l'únic que la podrà fer servir.

D'aquesta manera es va crear el nou camp dins la base de dades i es va permetre que quan l'usuari introduís unes credencials diferents a les emmagatzemades a les *SharedPreferences*, es validarien al backend per saber si aquest usuari existia en algun altre dispositiu, i en cas afirmatiu, podria entrar a l'aplicació i consultar les seves dades.

7.2. Desenvolupament de l'aplicació

Un dels principals objectius durant el desenvolupament de l'aplicació era el d'intentar mantenir el codi d'una forma estructurada i neta, amb l'ús de mètodes i d'altres classes que donessin suport. D'aquesta forma es podia dividir el codi i no ficar-lo tot a la activitat principal la qual s'encarrega de dur a terme la majoria de tasques com són, sincronitzar els valors (pujar-los i baixar-los del backend, xifrar i desxifrar les dades, mostrar-les, etc.).

El projecte disposa d'una classe anomenada *Utilities* que conté varies funcionalitats que s'utilitzaven en diferents classes, i per evitar la repetició del codi, s'afegia a aquesta classe a la qual es podia accedir des de qualsevol lloc. Aquests mètodes són, per exemple, un text per pantalla del tipus *Toast* que ofereix Android. Per crear-lo es fa mitjançant la següent línia de codi:

```
Toast.makeText(context, text, durada).show();
```

Per no tenir d'introduir aquest codi per totes les classes, es va preferir afegir-lo dins una classe i només fa falta cridar-li utilitzant:

```
Utilities.showToast(context, text);
```

Aquest no és un exemple molt clar ja que solament s'estalvia escriure dues paraules més, però hi ha moments com pot ser el fet de xifrar o desxifrar una clau que pot ser més extens i que és necessari tant quan s'afegeix una clau, quan es consulta, o quan es sincronitzen els valors, és a dir, que s'utilitzen des de punts diferents i d'activitats diferents.

En resum, la principal funcionalitat d'aquesta classe és reduir al màxim les línies de codi i el codi redundat a les activitats de l'aplicació.

També hi ha una classe més que s'encarrega de fer el xifrat i desxifrat amb AES degut a que realitzar aquesta tasca requereix gran quantitat de línies de codi. Realment les dues classes comentades podrien estar unides en una sola, però dividint-les d'aquesta forma queda més clar el motiu de la seva creació i tampoc es fa una classe molt extensa amb codi, ja que esta ben vist que les classes no siguin molt llargues (tot i que amb Android pot ser una mica complicat)

Un altra bona pràctica que s'ha intentat aplicar és el fet de mantenir el més separat possible l'ús de la xarxa, tot i que, avui en dia tothom disposa de tarifes de dades al seu dispositiu o es pot connectar a una xarxa Wifi.

El que s'ha fet ha estat deixar que l'usuari sincronitzi les dades quan ell o desitgi, afegint un botó al a barra de l'aplicació (*ActionBar*) el qual una vegada pitjat, es sincronitza amb les dades emmagatzemades remotament. En cas de no disposar de connexió a la xarxa, es mostra un text indicant el problema.

Pel que fa al log in, la primera vegada que s'obra l'aplicació i s'intenti entrar, es buscarà l'usuari al backend, i en cas d'existir deixarà entrar-lo a més a més de guardar-lo a les *SharedPreferences*[14] del dispositiu per tal de que en futures entrades, només faci falta comprovar aquest fitxer (s'explica més endavant).

Amb el registre, aquest aprofita la base de dades remota per donar-se d'alta, comprovant que el paràmetres introduïts per l'usuari no existeixen i ja estiguin emmagatzemats, com per exemple, que el nom d'usuari no el tingui una altra persona.

Una vegada registrat, també es guarden els paràmetres dins a les *SharedPreferences* per evitar l'ús de les dades en pròximes entrades.

7.2.1. Fils d'execució (Multithreading)

Quan s'executa una tasca lenta, és millor separar-la del fil principal d'execució ja que és el que s'encarrega de la interfície de l'usuari i pot quedar-se bloquejada. Per separar-ho es poden utilitzar els fils de Java (*Thread*), tot i que Android disposa d'una classe anomenada *AsyncTask*[13] la qual facilita la tasca. Aquesta disposa dels mètodes que s'executen al fil per, quan acabi l'execució, mostrar-ho a la pantalla d'alguna forma.

Aquestes funcions són:

- *onPreExecute()*: s'executa abans de començar el procés. Pot servir, per exemple, per inicialitzar un diàleg de progrés.
- *doInBackground()*: part principal de l'execució, on s'indiquen les tasques que s'han de dur a terme al fil secundari.
- *onProgressUpdate()*: rep la informació sobre el progrés de la tasca. S'acostuma a utilitzar per mostrar l'estat de la tasca en el diàleg de progrés.
- *onPostExecute()*: s'executa quan s'acaba el mètode *doInBackground()*. Rep el resultat de l'execució per tractar-lo i mostrar-lo per pantalla, per exemple, afegint els elements obtinguts a una llista.

En l'aplicació s'utilitza el fil en el moment de sincronitzar les dades amb el Cloud, ja que és una tasca lenta i, a més a més, una vegada acabada l'execució, actualitza la llista principal de claus afegint els nous valors descarregats.

```
protected List<ParseObject> doInBackground(Void... params)
{
    // Obtiene todos los datos del Cloud del usuario actual
    ParseQuery<ParseObject> query = ParseQuery
        .getQuery("generalSaveKey");
    query.whereEqualTo("user", ParseUser.getCurrentUser());
    List<ParseObject> queryResult = null;

    try {
        queryResult = query.find();
    } catch (ParseException e) {
        Log.e("Error", e.getMessage());
        e.printStackTrace();
    }
    return queryResult;
}
```

7.2.2. Ús de SharedPreferences

S'utilitza aquesta memòria del dispositiu per separar la informació de l'aplicació (les dades emmagatzemades) de les credencials de l'usuari que la utilitza, és a dir, també s'hagués pogut emmagatzemar les dades de login dins d'una base de dades al dispositiu o a la mateixa en la que s'emmagatzemen totes les claus, però, en cas que algú pogués extreure la base de dades del mòbil, tindria tota la informació (tot i que estaria xifrada).

Per fer-ho solament fa falta instanciar la interfície *SharedPreferences*[14] amb el nom donat a la informació guardada (com si fos un nom d'una taula d'una base de dades) i ja es pot accedir al contingut:

```
SharedPreferences sharedPreferences =  
getSharedPreferences("Login", Context.MODE_PRIVATE);
```

Amb aquesta nova instància es pot comprovar si té valors emmagatzemats de forma que es podrà saber si l'usuari ja ha entrat alguna altra vegada o no.

La informació que s'emmagatzema dins de les *SharedPreferences* té una estructura clau-valor, tal i com es veu a la Figura 17, i per obtenir les dades guardades sol cal buscar la clau.

| key | value |
|-----------|-------|
| firstName | Bugs |
| lastName | Bunny |
| location | Earth |

Figura 17: Estructura clau-valor

Per fer-ho, es pot utilitzar algun dels mètodes *get* dels que disposa la classe, passant-li el text o valor de la clau, i també marcant-li un valor per defecte en cas de que no es trobi cap clau igual a la indicada:

```
String prefUsername =  
sharedPreferences.getString("username", null);
```

Tal i com es pot veure a la línia de codi, es vol obtenir el valor que té la clau "username" (primer paràmetre) i en cas de no trobar-lo, el valor per defecte serà null (segon paràmetre).

Seguidament sol fa falta comprovar si el valor és null per saber si s'han utilitzat les *SharedPreferences* o no, i d'aquesta forma, saber si s'ha de validar el usuari sobre aquests o sobre el backend.

En cas de no estar creades, ja sigui perquè és la primera vegada que s'utilitza l'aplicació o perquè l'usuari es registra, és necessari escriure tota la informació introduïda als camps (editText) dels formularis de les pantalles de login o de registre per a les pròximes execucions ja tenir-lo guardat.

Això es fa utilitzant una interfície preparada per editar els valors de les *SharedPreferences* anomenada

SharedPreferences.Editor[15].

Aquesta interfície consta de mètodes *put* per emmagatzemar les dades de forma clau-valor. Per exemple s'utilitza per guardar el nom d'usuari:

```
//Instància de la interfície
SharedPreferences.Editor sharedEditor =
sharedPreferences.edit();

//Afegeix amb clau "username" el valor de "username"
sharedEditor.putString("username", username);

//Neteja les sharedPreferences d'altres possibles valors
guardats
sharedEditor.clear();

//Guarda els canvis dins de les SharedPreferences
sharedEditor.commit();
```

El problema que hi ha és que el fitxer on s'emmagatzema aquesta informació es accessible, sempre i quan el dispositiu sigui un emulador o un amb permisos d'administrador.

Per veure'l s'ha de mirar dins la carpeta de l'aplicació generada al dispositiu, normalment seguint l'estructura de carpetes:

```
data --> data --> <paquet_de_aplicació> --> shared_pref
```

Dins s'hi troba un fitxer amb extensió *xml* amb el nom que s'ha marcat quan s'han creat la configuració de les *preferences*.

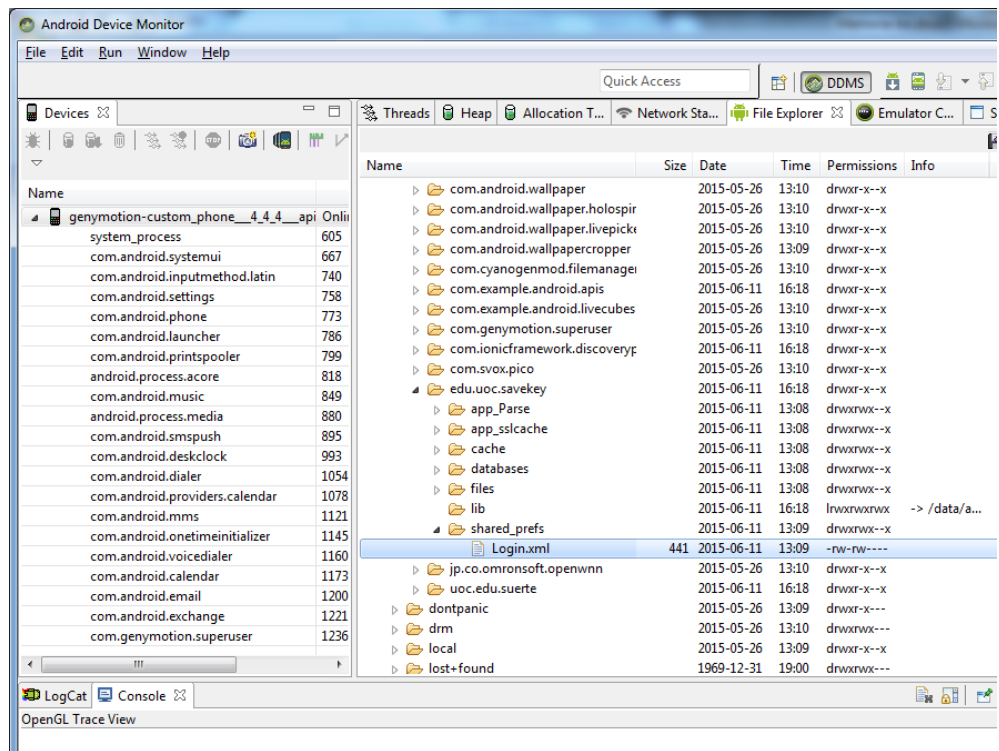


Figura 18: Arxiu de Login creat per les *SharedPreferences*

Si s'observa el contingut del fitxer (Figura 19) es podrà veure que s'ha guardat el usuari i contrasenya que ha iniciat sessió.

```

1  <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2  <map>
3    <string name="password">123456</string>
4    <string name="username">Usuari01</string>
5  </map>

```

Figura 19: Contingut del fitxer creat per les *SharedPreferences*

Tal i com es pot veure, el text es completament llegible i simplement veient el que hi ha escrit, ja es pot suposar quin és la contrasenya i quin és el nom d'usuari. Per evitar tot això, es fa ús de la criptografia.

Al final, aplicant tot el comentat, el fitxer acaba guardant tres paràmetres que són el nom d'usuari i la contrasenya, per les pròximes vegades que s'entri a l'aplicació, i també emmagatzema la data en que ha estat creat aquest usuari pel xifrat de les dades de l'aplicació, obtenint un resultat com el de la Figura 20.

```
1 <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2 <map>
3   <string
4     name="password">9d643b6bedf402d96a3571248b7eedbae1586b38c51fe116e4977f4cacf9a697846d9c6216a30
5     eb016fdccbb52a994818c6e7c713a14a795f06ac3fdac298d23f49e2dfab2c94ec65c71c4fb3a8a9aa</string>
6   <string name="username">cb0465ab4470db535867c672df405ac7</string>
7   <string
8     name="createdAt">97028c683d1c5d58d10b53b3293a6bc5ce847d2a6048570d7192838e21af9efd</string>
9 </map>
```

Figura 20: Contingut del fitxer creat per les *SharedPreferences* xifrat

7.2.3. Estat de la xarxa i *BroadcastReceiver*

Abans de dur a terme qualsevol tasca que necessiti d'Internet, sempre es fa una comprovació de l'estat de la xarxa per saber si fa falta executar la tasca o no[16]. En cas de no tenir Internet, es mostra un text a l'usuari indicant el problema.

Un exemple és el de sincronitzar les dades, el qual té de pujar i descarregar les dades de la base de dades remota, i per tant, fa falta l'ús d'Internet per poder accedir. De forma que, una vegada pitjat el botó que farà aquesta funció, si la xarxa no esta activa, es mostra un text de "Es necessita connexió per sincronitzar les dades".

En la pantalla d'inscriure's també hi ha aplicada aquesta comprovació, però a més a més de mostrar el text, també activa o desactiva el botó que permet crear la nova compta (Figura 21). Mentre no hi hagi connexió, el botó estarà desactivat, però quan es torni a establir la xarxa, s'activarà (Figura 22) i viceversa.

Això es fa utilitzant un component d'Android anomenat *BroadcastReceiver*[17] el qual serveix per registrar els canvis en l'aplicació o el sistema, de forma que, en aquest cas, s'encarrega de comprovar si hi ha canvis en l'estat de la xarxa, i en cas afirmatiu activa o desactiva el botó comentat.

Quan s'entra a la pantalla de crear una nova compta, aquesta activa el receptor de xarxa, per a que quan hi hagi un canvi pugui portar-se a terme l'acció necessària. Quan es destrueix l'activitat, el receptor és desactiva evitant que es quedi consumint bateria en segon pla.

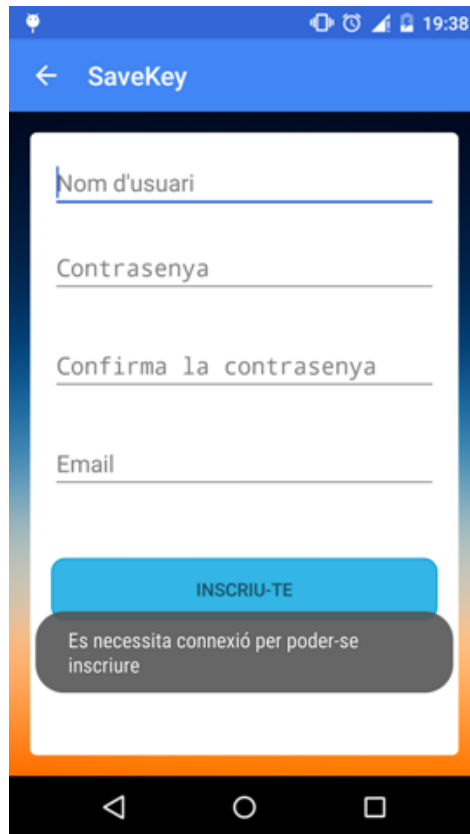


Figura 21: Pantalla de registre amb botó desactivat (Android)

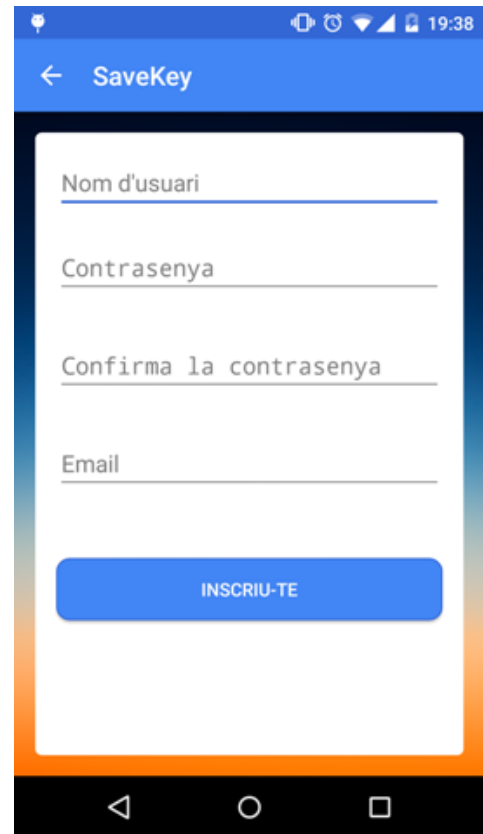


Figura 22: Pantalla de registre amb botó activat (Android)

7.2.4. Aplicació multilingüe

Tots els textos de l'aplicació s'adapten al idioma que té el dispositiu instal·lat, entre català (per defecte), castellà o anglès[18]. Per fer-ho sol és necessari crear una nova carpeta "values" seguit del codi del llenguatge[19] dins del projecte amb el fitxer "strings.xml" dins seu, de forma que quedaria:

```
MyProject/
  res/
    values/
      strings.xml
    values-es/
      strings.xml
    values-fr/
      strings.xml
```

Per exemple, la carpeta "values-es" contindria el fitxer amb els textos en castellà.

L'únic que s'ha de fer es mantenir el nom del *string* igual als diferents idiomes perquè l'aplicació sàpiga quin té d'agafar.

8. Aplicació final d'Android

8.1. Pantalla de log in (pantalla inicial)

Quan s'obra l'aplicació, la primera pantalla que es troba és al de log in on l'usuari ha d'introduir les seves credencials dins de dos camps de text per poder accedir al contingut de l'aplicació (Figura 23). En cas que sigui la primera vegada i no estigui registrat, serà necessari passar per la Pantalla de registre i crear un nou usuari el qual després podrà utilitzar per entrar.

Si es disposa d'usuari i contrasenya es poden introduir els valors dins els camps de text de la pantalla inicial per donar-li al botó d'entrar. Una vegada s'ha pitjat aquest botó, l'aplicació s'encarrega de comprovar aquestes credencials introduïdes.

Tal i com s'ha explicat en l'apartat 7.2.2 Ús de `SharedPreferences`, es mira si existeixen les `SharedPreferences` al dispositiu per saber si la validació es podrà fer local o s'haurà de fer a la base de dades remota.

En cas que no existeixi l'usuari o els camps de text estiguin buits, es mostra un `Toast` indicant l'error (credencials incorrectes, Figura 24, o falta omplir els camps).

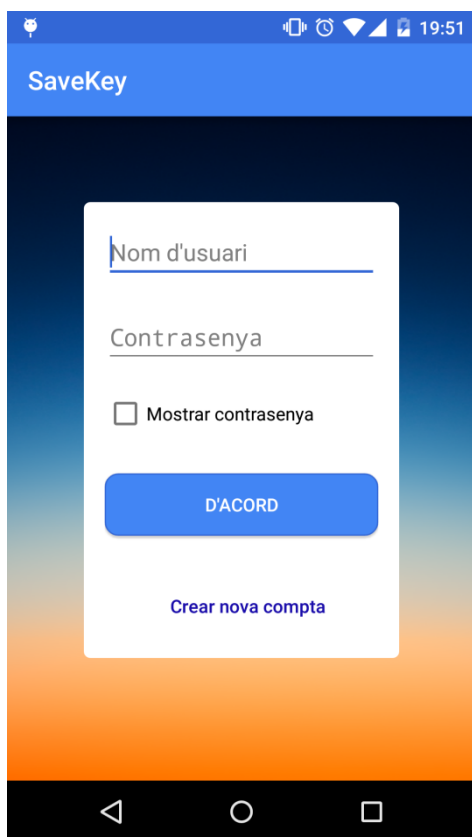


Figura 23: Pantalla inicial de login (Android)

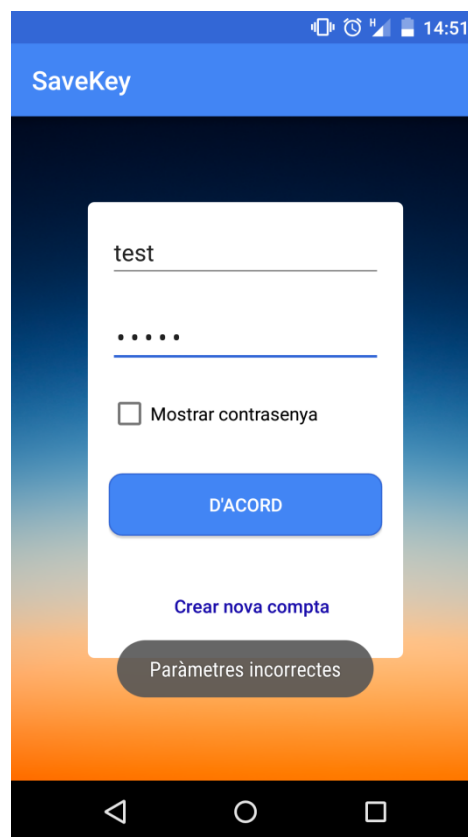


Figura 24: Pantalla de login amb missatge d'error (Android)

8.1.1. Validació remota

Aquesta validació sol es farà la primera vegada que s'accedeixi a l'aplicació sense passar pel registre, ja que voldrà dir que no s'han guardat les credencials al dispositiu i per tant que s'ha de comprovar si l'usuari existeix a la base de dades remota per, seguidament guardar-lo.

Primer de tot, abans d'enviar els paràmetres cap a la base de dades remota, es fa un hash de la contrasenya per evitar que si s'agafa un paquet durant la transmissió es pugui veure la contrasenya i, a més a més, totes les contrasenyes emmagatzemades estan amb format hash per tal de no tenir cap text llegible.

Seguidament, la base de dades dona una resposta conforme l'usuari es correcte o no. En cas negatiu, es mostra un text per pantalla indicant que els paràmetres són incorrectes. En cas afirmatiu, s'obté la resposta de la qual s'extreu la data de creació de l'usuari amb la que es farà el xifrat de les dades.

Seguidament, aquest valors han de ser emmagatzemats a les *SharedPreferences* però de forma xifrada. Per fer-ho s'utilitza l'identificador del dispositiu en el que s'està executant l'aplicació el qual serà utilitzat com a part de la clau del xifrat AES.

Aquest identificador s'obté a través del component *TelephonyManager*[20] d'Android, amb el qual es pot accedir als serveis del dispositiu i a la seva informació.

```
TelephonyManager telephonyManager = (TelephonyManager)
context.getSystemService(Context.TELEPHONY_SERVICE);
```

Primer s'obté la referència al *TelephonyManager* i seguidament, aprofitant els seus mètodes es pot fer el *getDeviceId* el qual retorna el IMEI[21] del dispositiu.

A continuació, per afegir un *Salt*[22], s'utilitza la contrasenya que s'ha introduït al *EditText* de la pantalla inicial, i es concatena amb el identificador obtingut anteriorment. El resultat d'aquesta unió s'utilitzarà com a contrasenya pel xifrat AES:

$$passAES = IMEI + loginPassword$$

Per exemple:

$$IMEI = 123456789123456$$
$$loginPassword = contraseña123456$$
$$passAES = 123456789123456contraseña123456$$

Aquesta contrasenya serà utilitzada per xifrar els paràmetres de login de l'usuari (nom d'usuari i contrasenya) amb l'algoritme de xifrat AES.

A la contrasenya de l'usuari, a més a més, se li aplica un hash abans de fer-li el xifrat, igual que la data de creació, tot i que aquesta no serà xifrada. Finalment l'algoritme acabarà sent:

$$encryptUsername = AES(passAES, username)$$
$$shaPassword = Sha256(contrasenya_mestra)$$
$$encryptPassword = AES(passAES, shaPassword)$$
$$shaCreatedAt = Sha256(userCreatedAt)$$

L'execució de tots aquests passos (xifrar i emmagatzemar les credencials) els du a terme el mètode *setUserToSharedPreferences*:

```
public static void setUserToSharedPreferences(Context context,
String logUsername, String logPassword, String createdAt)
{
    // Obtiene el identificador del dispositivo (el IMEI en
    caso de tener)
    TelephonyManager telephonyManager = (TelephonyManager)
    context.getSystemService(Context.TELEPHONY_SERVICE);
    String deviceId = telephonyManager.getDeviceId();
    String oPass = deviceId + logPassword;

    // Cifra el nombre de usuario y la contraseña
    String encryptUsername = getEncryptedPassword(oPass,
    logUsername);
    String encryptPass = getEncryptedPassword(oPass,
    CryptoHelper.getSha256Digest(logPassword));

    // Añade las credenciales a las SharedPreferences
    SharedPreferences sharedPreferences =
    context.getSharedPreferences(
    "Login", Context.MODE_PRIVATE);
    SharedPreferences.Editor sharedEditor =
    sharedPreferences.edit();
    sharedEditor.putString("username", encryptUsername);
    sharedEditor.putString("password", encryptPass);
    sharedEditor.putString("createdAt", createdAt);
    sharedEditor.clear();
    sharedEditor.commit();
}
```

On els arguments *logUsername* y *logPassword* són els paràmetres introduïts al formulari de la pantalla, i l'últim argument *createdAt* és la data de creació de l'usuari obtinguda de la base de dades remota, i que arriba a aquest mètode amb el hash aplicat. Al final, l'arxiu de configuració és el que es pot veure a la Figura 20.

En cas de no tenir connexió de dades, es mostrarà un text indicant la necessitat d'Internet per poder realitzar la comprovació.

8.1.2. Validació local

Aquesta validació es farà la resta de vegades que s'accedeixi a l'aplicació i consisteix en comprovar si els paràmetres introduïts al formulari inicial són els mateixos que els que hi ha guardats a l'arxiu de configuració, tot i que, s'haurà de fer amb els valors xifrats ja que estan emmagatzemats d'aquesta forma.

Resumidament, una vegada introduïdes les credencials i pitjat el botó, es xifrarà les dades igual que l'algoritme anterior (amb el IMEI i la contrasenya introduïda), i es mirarà si el resultat obtingut és el mateix que el que hi ha a les *SharedPreferences*.

En cas afirmatiu, s'entrarà a la pantalla principal de l'aplicació. En cas negatiu, es comprovarà si l'usuari existeix a la base de dades remota. Aquesta segona comprovació es fa per a no limitar l'aplicació a un sol usuari per dispositiu, és a dir, si només s'accepta el primer usuari amb el que s'ha entrat, llavors cap més usuari podrà accedir sense abans haver desinstal·lat l'aplicació i esborrat totes les dades de la base de dades local.

D'aquesta forma, el que fa es comprovar si existeix en remot per validar si és un usuari completament diferent, per exemple, una parella que comparteixen una tablet on hi ha instal·lada l'aplicació i es va intercanviant el compte.

L'única limitació que té és que cada vegada que es faci un canvi de compte serà necessari la connexió de dades per poder dur a terme la validació remota comentada a l'apartat 8.1.1.

8.2. Pantalla de registre

Des de la pantalla inicial es pot pitjar sobre el text/botó de crear una nova compta per accedir al formulari de registre, el de la Figura 25. Aquest disposa de 4 camps:

- Nom d'usuari
- Contrasenya
- Confirmar contrasenya
- Correu electrònic

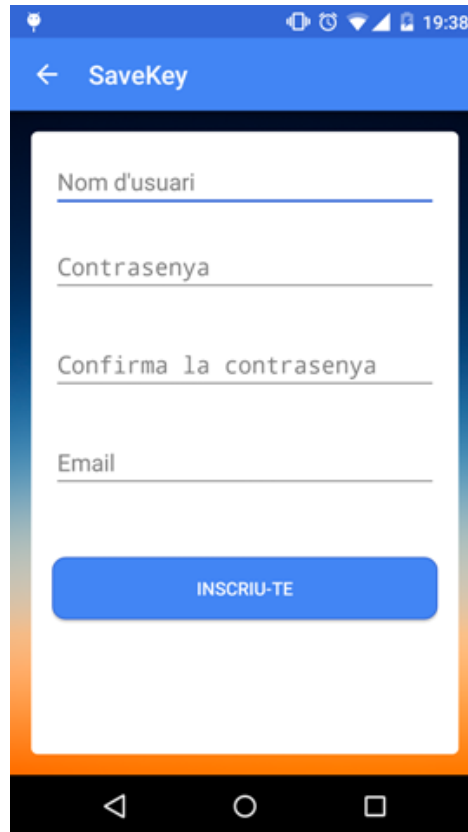
The image shows a mobile application interface for registration. At the top, there is a blue header bar with a back arrow and the text "SaveKey". Below the header, there is a white form area with four input fields, each with a label above it: "Nom d'usuari", "Contrasenya", "Confirma la contrasenya", and "Email". At the bottom of the form area, there is a blue button with the text "INSCRIU-TE". The entire form is set against a dark background. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps buttons.

Figura 25: Pantalla de registre (Android)

Una vegada s'entra a la pantalla, s'activa el *BroadcastReceiver* comentat a 7.2.3 *Estat de la xarxa i BroadcastReceiver* per saber si es disposa o no de connexió de xarxa ja que el fet de crear un nou usuari es fa directament sobre la base de dades remota, i per tant, es necessita connexió per poder enviar els paràmetres.

Quan es pitja sobre el botó d'inscriure's, es realitzen una sèrie de comprovacions abans de crear el nou usuari.

1. Comprovar que no hi hagi cap camp buit
2. Comprovar que les dues contrasenyes són iguals com a mètode de seguretat

3. Finalment comprova que l'email té el format correcte utilitzant els patrons que ofereix Android[23]

Si es superen totes aquestes validacions, es fa el sha256 de la contrasenya i s'envia juntament amb la resta de camps cap a la base de dades remota, on aquesta s'encarregarà de comprovar que aquells paràmetres no estiguin repetits amb algun altre usuari. En cas d'haver-hi algun problema, es mostrarà un text indicant-lo (Figura 26, Figura 27).

Però si tot funciona correctament s'emmagatzemarà aquell nou usuari amb la contrasenya amb el hash aplicat. Per això la resta de peticions s'han de fer de la mateixa forma, sinó el valor mai serà igual.

Com a resultat, la base de dades retorna l'usuari creat, amb el que es pot obtenir la data de creació i guardar les dades necessàries a les *SharedPreferences*, tal i com esta explicat a la part final de l'apartat 8.1.1 Validació remota.

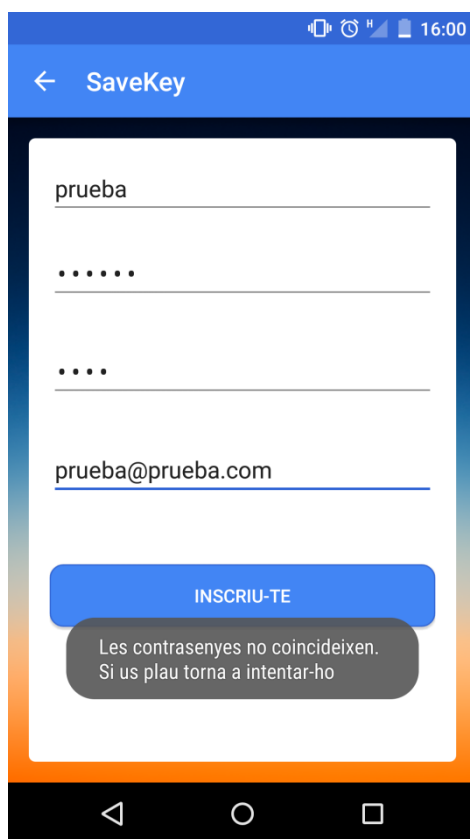


Figura 26: Pantalla de registre amb error de contrasenyes diferents (Android)

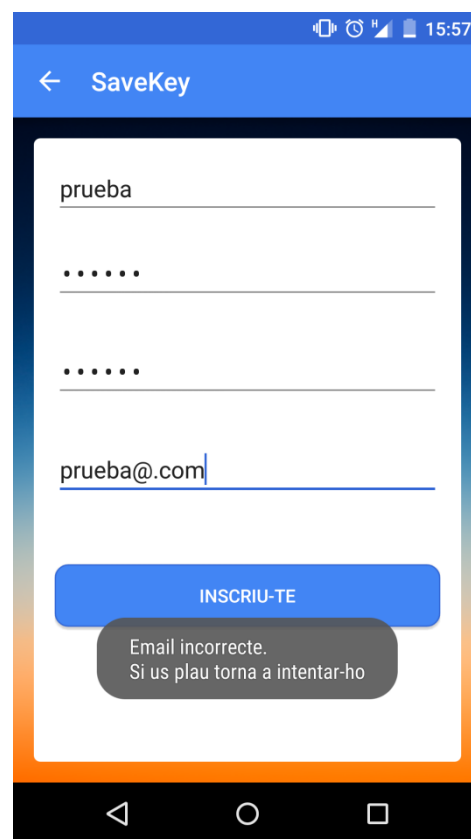


Figura 27: Pantalla de registre amb error d'email incorrecte (Android)

8.3. Pantalla principal (llistat de valors)

Quan s'accedeix a aquesta pantalla, primer de tot s'instancia la classe *SaveKeySQLiteHelper* que és la classe que s'encarrega del control de la base de dades del dispositiu, fent així que es creï o s'obri la taula que conté tota la informació.

Seguidament, s'obtindrà una llista amb el identificador, el nom del lloc i l'àlies de totes les dades corresponents a l'usuari, i passarà per un adaptador per mostrar-les amb un format personalitzat, on es mostraran els dos camps. Aquest adaptador hereta de la classe *BaseAdapter*[24], la qual serveix bàsicament per adaptar llistes de dades a llistes de vista amb una estructura personalitzada. Simplement s'ha de generar una vista nova amb els components visuals preferits pensant que aquella vista serà una fila de llista. En l'aplicació s'ha creat el disseny de la Figura 28, el qual conté un text amb el nom del lloc, un text amb l'àlies, i dins d'una redona la qual canvia de fons segons l'element, conté la primera lletra del lloc.



Figura 28: Disseny d'un element de la llista

En cas de no haver-hi dades, es mostra un *TextView* amb un escrit conforme no hi ha dades emmagatzemades, igual que la Figura 29. Per afegir-ne es pot fer de dues formes, o bé, creant noves dades, o si ja s'ha utilitzat l'aplicació anteriorment, recuperant les dades de la base de dades remota.

Per crear-les es fa pitjant el *Android Floating Button*[25] (el botó rodó de la part inferior a la dreta de la pantalla) el qual obrirà la Pantalla de creat/modificar dades.

Una vegada omplerta la base de dades, es podrà veure el resultat de l'estil comentat abans aplicat a la llista (Figura 30).

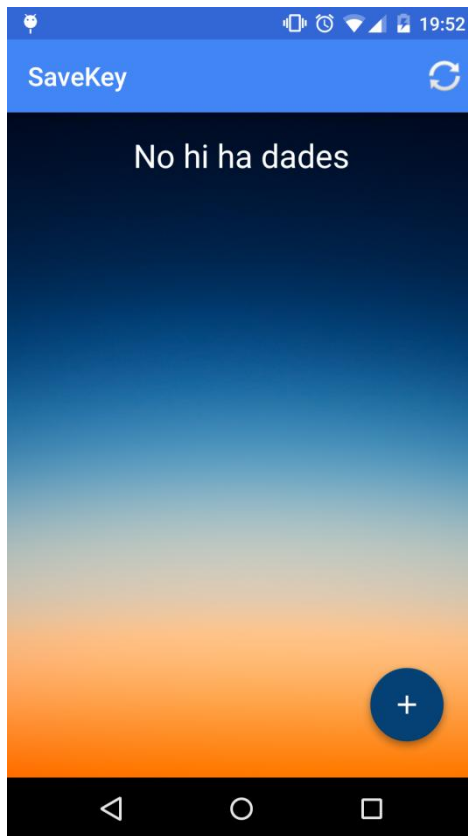


Figura 29: Pantalla principal sense dades (Android)

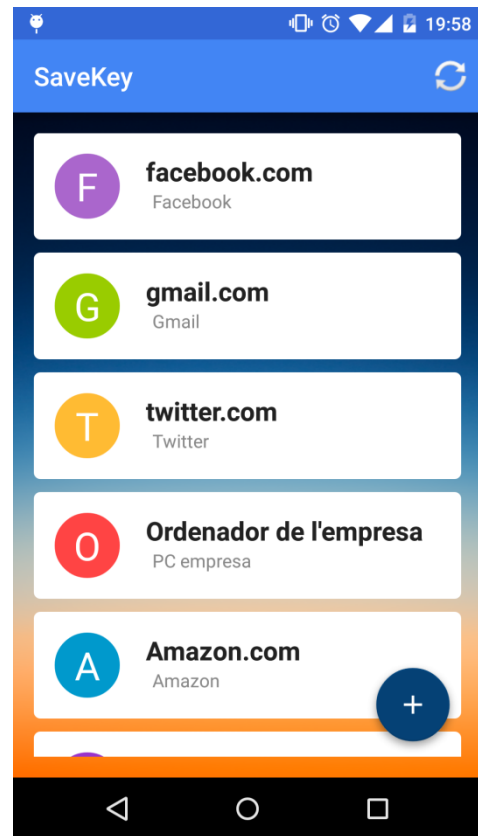


Figura 30: Pantalla amb dades (Android)

Amb aquestes dades afegides es poden fer dues coses:

8.3.1. Consultar les dades

La llista consta d'un *OnItemClickListener*[26], una interfície que s'encarrega de dur a terme unes tasques programades quan es pitja sobre un element del *AdapterView*[27], en aquest cas, la llista principal la qual és un *ListView*.

La tasca que executa en l'aplicació és la de mostrar totes les dades del valor que s'ha seleccionat. Primer es fa una petició a la base de dades per obtenir tota la informació referent a l'element, però sense oblidar que hi ha les dades encriptades i per tant, que no són llegibles (apartat 7.1.1), de forma que, utilitzant la mateixa contrasenya amb la que s'ha xifrat, es desxifra el text, i seguidament es mostra tota la informació dins un quadre de diàleg (Figura 31), juntament amb tres botons que permetran:

- Tancar la finestra
- Modificar el valor: si s'apreta sobre aquest botó, s'obrirà la mateixa pantalla amb la que es creen els valors però, dins dels camps de text s'afegirà la informació del valor, amb el resultat de la Figura 32, on es podran modificar les dades i tornar-les a guardar amb la informació actualitzada.
- Eliminar el valor: en el moment d'eliminar un objecte de la llista, es pot triar si solament es vol esborrar del dispositiu, o si també es vol eliminar de la base de dades remota degut a que és un valor que mai més es tornarà a fer servir. Aquesta elecció es fa sobre un segon *AlertDialog* el qual conté els dos botons comentats (Figura 33).

En les dues opcions el que es fa és buscar l'objecte amb el mateix identificador que el que s'ha seleccionat per, a continuació, esborrar-lo.

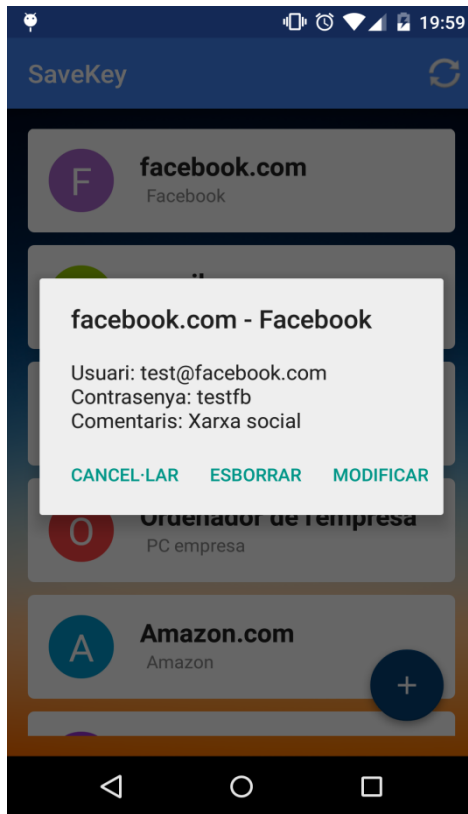


Figura 31: *AlertDialog* amb tota la informació de l'element (Android)

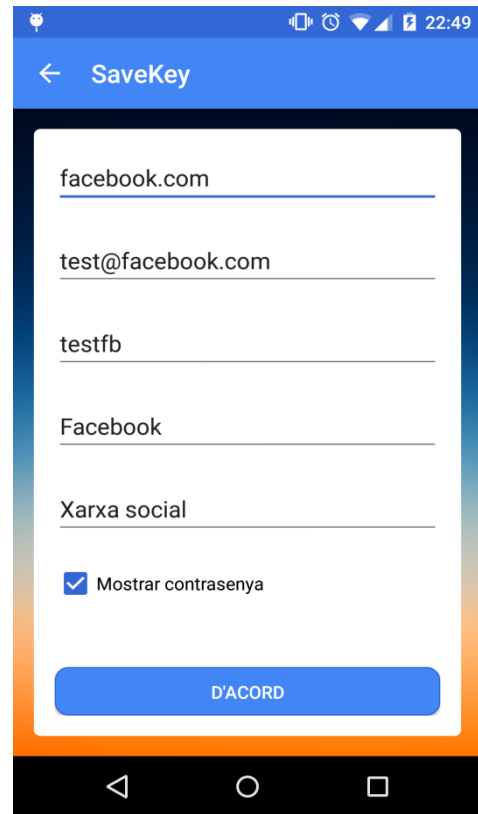


Figura 32: Edició d'un valor (Android)

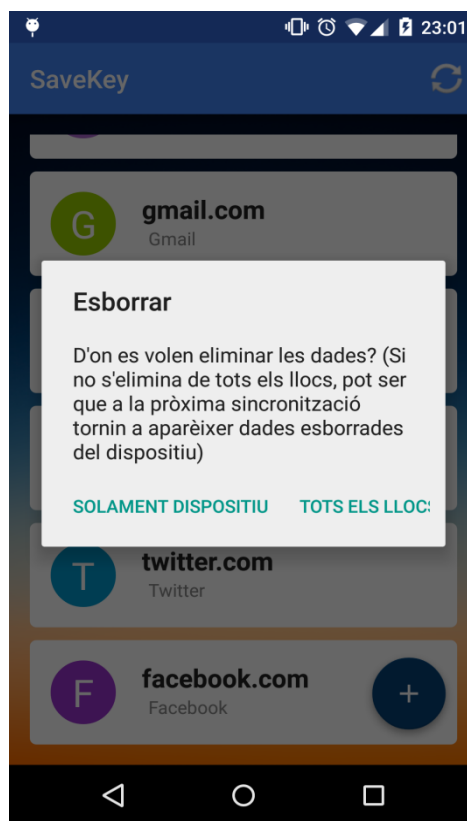


Figura 33: *AlertDialog* per eliminar un valor (Android)

8.3.2. Sincronitzar les dades

Aquesta part s'encarrega d'actualitzar totes les dades, tant local com remot i com a resultat d'aquesta acció hi ha d'haver els mateixos objectes a les dues bases de dades.

Aquest mètode esta dividit en dos parts, la de baixar dades i la de pujar-les.

La primera consisteix en obtenir totes les dades del Cloud i guardar-les al dispositiu. Per fer-ho s'obtenen tots els elements de la base de dades remota i, per cada element, es busca si existeix a la base de dades local (utilitzant el camp "objectId" de les dues bases de dades). En cas que aquell id ja existeixi, s'ignora el valor i es passa al següent, però en cas contrari, és a dir, que no existeix, s'agafa l'objecte i se li afegeixen els paràmetres addicionals abans de ser emmagatzemat localment. Aquest paràmetres són:

- Nom d'usuari que esta utilitzant l'aplicació
- Valor "isSync" a *true*, indicant que l'objecte ja esta sincronitzat
- Valor "isUpdated" a *false*, marcant que no s'ha modificat

El segon pas és el d'obtenir totes les dades del dispositiu per enviar-les cap al remot. Simplement es fa una consulta a la base de dades obtenint tots els valors que no estiguin sincronitzats (ja que el valor "isSync" de la base de dades indica si ho han estat, o també es pot mirar si el valor té "objectId" ja que sol s'afegeix quan s'ha sincronitzat), o que s'hagin modificat des de la data de la seva creació i seguidament s'envien per a ser emmagatzemats al Cloud.

```
Cursor cursor = db.rawQuery  
("SELECT * FROM KeySave WHERE username=? AND isSync=0 OR  
isUpdated=1", new String[]{username});  
//0 = false, 1 = true
```

En cas d'haver estat modificats després de la seva sincronització, serà necessari buscar aquell valor a la base de dades remota, mitjançant la columna "objectId", per actualitzar-lo amb la nova informació.

A mesura que es van pujant i baixant les dades, també es van canviant els valors de si l'objecte ha estat sincronitzat i/o actualitzat per evitar que en la següent sincronització es tornin a pujar i puguin haver repeticions de dades. A més a més, com que les dades ja surten xifrades de les dues base de dades, no cal tractar-les, i ja es poden emmagatzemar directament.

Ja que la tasca pot ser lenta, mentre s'està executant es mostra un *ProgressDialog* com el de la Figura 34, indicant que l'aplicació no està aturada sinó que s'estan duent a terme unes tasques.

A més a més, quan s'està executant una tasca en un *AsyncTask* és necessari controlar els canvis d'orientació de la pantalla ja que, si es gira mentre està en execució, segurament l'aplicació s'aturi. Per tant, una forma simple d'evitar això és que justament quan es comença a dur a terme la tasca, es desactiva la rotació, evitant que al rotar el dispositiu es mogui la pantalla, i quan acaba l'acció, tornar a activar la rotació.

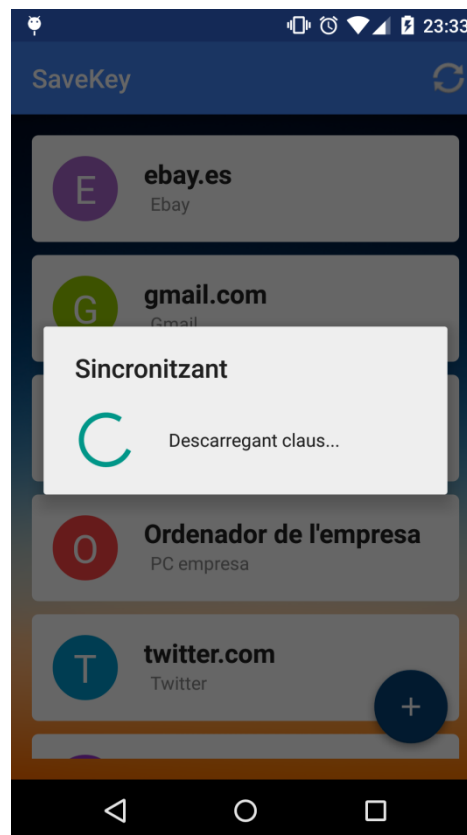


Figura 34: *ProgressDialog* de sincronització (Android)

Quan s'acaba tot el procés es mostra un text indicant que la sincronització a estat completa i s'afegeixen els nous valors a la llista principal (sempre i quan hi hagi nous valors).

8.4. Pantalla de creat/modificar dades

Aquesta pantalla consta d'un formulari amb tots els camps necessaris per crear una nova clau, tal i com es pot veure a la Figura 35.

Els camps de lloc, usuari i contrasenya són obligatoris, per tant, si no s'omplen i s'apreta el botó, es mostrarà un missatge d'error igual que el de la Figura 36.

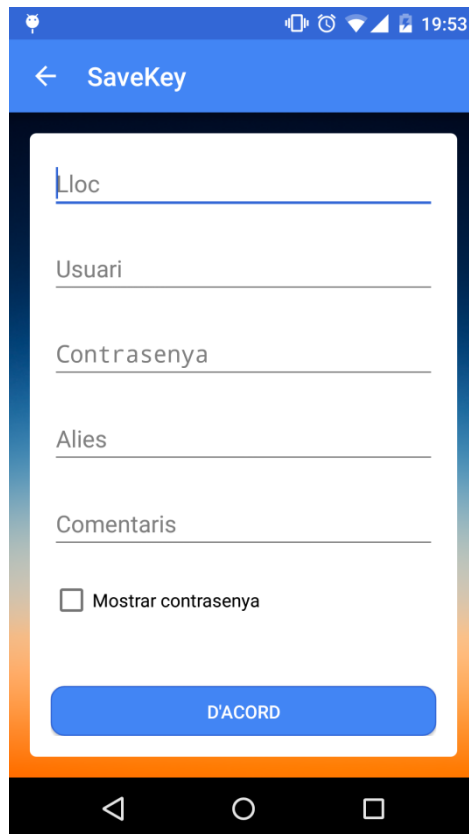


Figura 35: Formulari per crear una nova clau (Android)

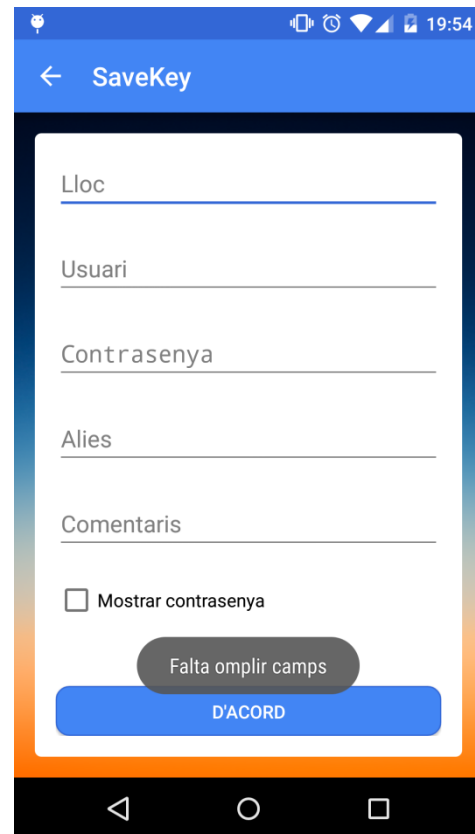


Figura 36: Error de camps buits (Android)

Una vegada es tinguin els valors introduïts, l'aplicació s'encarregarà d'emmagatzemar la nova clau, amb la qual, l'únic important és el fet de xifrar la clau.

Per fer-ho, s'utilitza l'algoritme explicat a l'apartat Xifrat AES i Hash SHA256, el qual consisteix en fer un xifrat AES aprofitant la data de creació de l'usuari.

Per exemple, es crea una clau nova amb els valors següents i després es pitja sobre guardar:

- Lloc: www.amazon.es
- Usuari: test@amazon.com

- Contrasenya: 123456
- Àlies: Amazon
- Comentaris: Tenda online

Seguidament l'aplicació agafarà la contrasenya introduïda (valor que es dirà "pass") per encriptar-la amb l'algoritme AES prèviament al seu emmagatzematge. Tal i com s'ha comentat, per fer el xifrat és necessari una clau, la qual ha estat transferida des de la pantalla inicial al obrir la nova pantalla (dins del *Intent*[28] que ha obert l'activitat):

```
//La pantalla principal afegix la clau dins del Intent
Intent intent = new Intent(getApplicationContext(), CrudDB.class);
intent.putExtra("shaPass", shaPassword);

//La pantalla d'afegir un nou element obté la clau de dins del
Intent
shaPassword = getIntent().getStringExtra("shaPass");
```

Amb això ja es pot dur a terme l'algoritme AES:

$$xifratAES = AES(clau, text) = AES(shaPassword, pass)$$

```
//La pantalla principal afegix la clau dins del Intent
String encPass = Utilities.getEncryptedPassword(shaPassword, pass)
```

Finalment, s'insereix el valor a la base de dades i a la llista principal.

9. Aplicació web

L'aplicació web es va desenvolupar com a unes primeres proves d'aprenentatge de l'aplicació de *Appverse Web*[8].

A diferència d'Android, aquesta aplicació treballa directament amb el backend.

Al principi no xifrava les dades, ja que el projecte es va derivar cap a aplicació híbrida, i en aquella es va fer que xifres, però finalment també es va afegir el xifrat a aquest.

Per fer-ho s'utilitza la llibreria *CryptoJS*[30] la qual dona certs algorismes de xifrat i també per fer el hash:

```
//Hash SHA256
var hash = CryptoJS.SHA256(value);

//Encriptació AES
var encryptAES = CryptoJS.AES.encrypt(value, aesPassword);

//Desencriptació AES
var decryptAES = CryptoJS.AES.decrypt(encryptAES, aesPassword);
```

Es van seguir els mateixos processos de xifrat que Android (7.1.1), amb la mateixa clau igual, però el problema va aparèixer en veure que el xifrat AES de Javascript i el xifrat AES d'Android (Java) eren diferent.

9.1. Vista inicial

La primera pàgina que es pot veure es la de login (Figura 37) i registre (Figura 38), la qual conté un formulari on introduir les credencials d'usuari per entrar, amb dos botons, un per entrar, i un per mostrar el formulari de registre.

Mentre el formulari no sigui correcte, no s'activarà el botó per entrar i/o registrar. És a dir, el mateix llenguatge de *AngularJs* fa comprovacions sobre el formulari del tipus:

- Tots els camps que s'hagin indicat amb un "required" no poden estar buits
- L'email té de tenir un format correcte amb un símbol d'arrova

En el login, una vegada es pitjo el botó d'entrar, aquest buscarà a la base de dades remota si existeix aquest usuari per deixar-lo entrar.

Si és un registre, es buscarà que no hi hagi coincidències de nom i després s'entrarà al contingut de l'aplicació.

En cas d'error, es mostrarà un text indicant el problema, igual que el de la Figura 39.

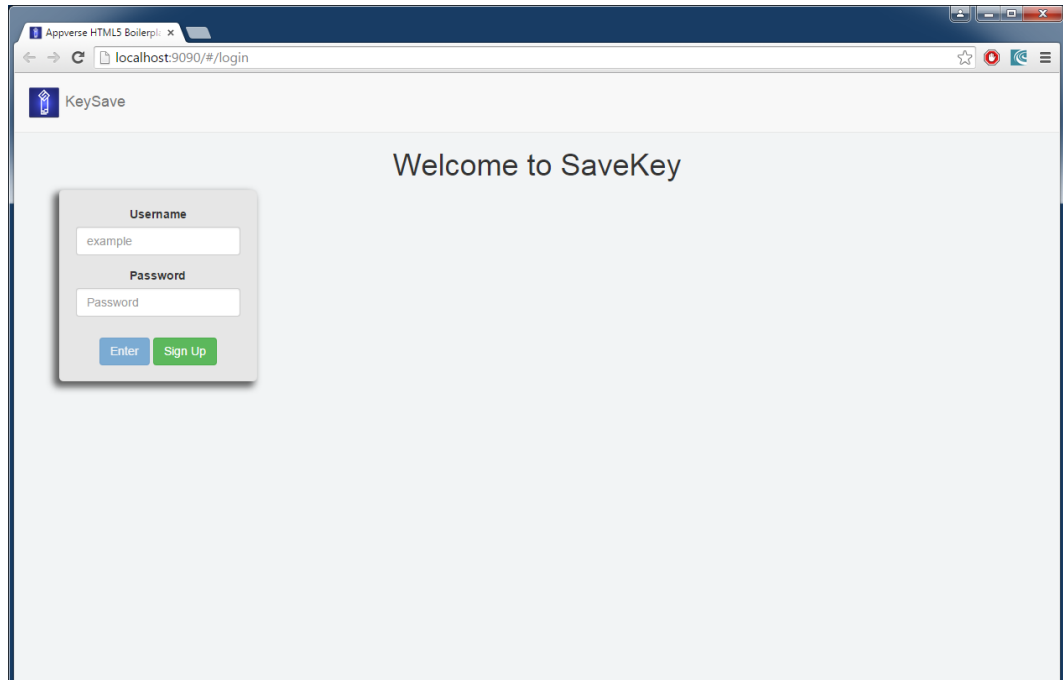


Figura 37: Pantalla de login (App web)

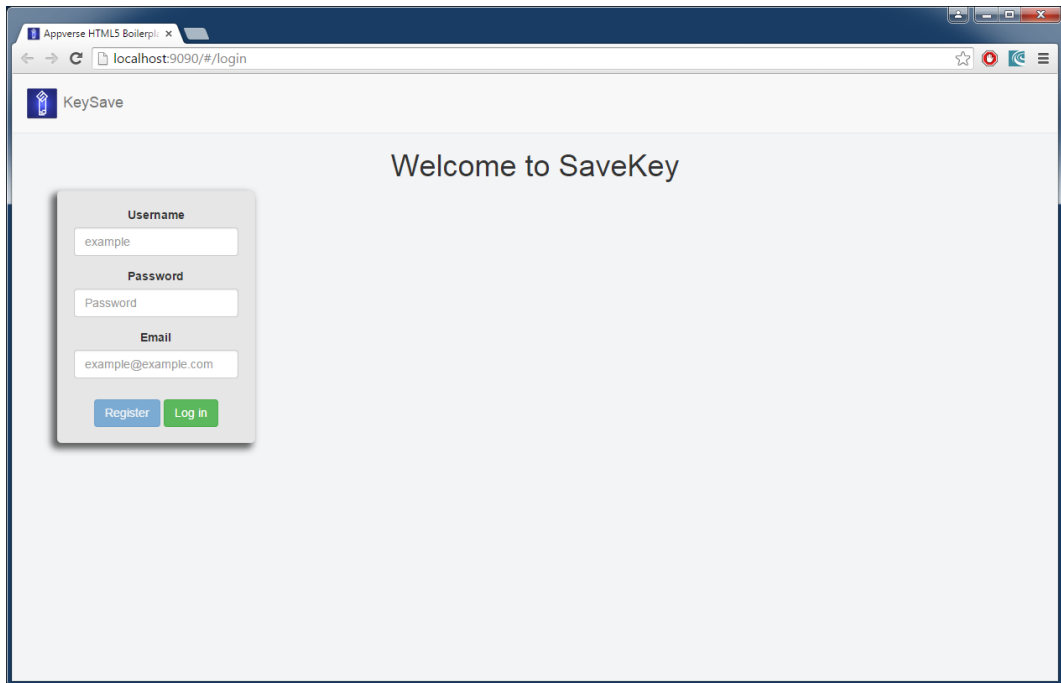


Figura 38: Formulari de registre (App web)

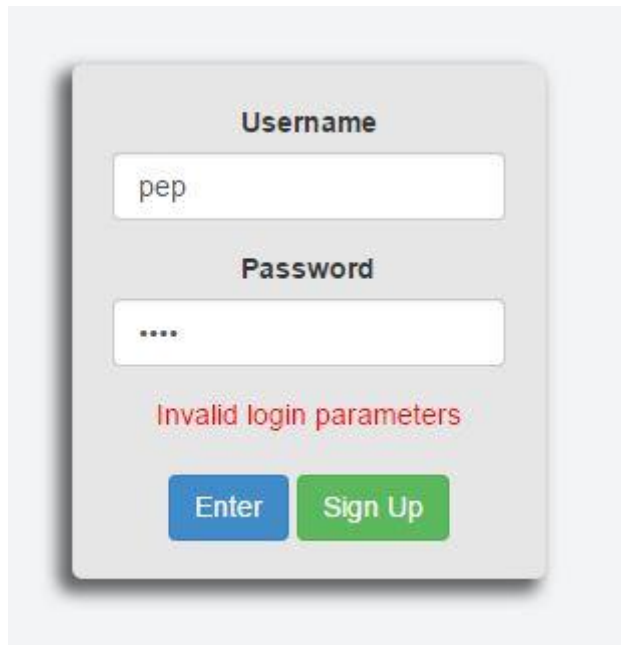


Figura 39: Error al login (App web)

9.2. Vista principal (llistat de claus)

Una vegada s'entri, es farà una petició al backend per obtenir totes les claus que l'usuari té emmagatzemades, es desxifraran i seguidament es mostraran per pantalla, tal i com es veu a la Figura 40.

També pot passar que no hi hagi claus, pel que es mostrarà un text com el de la Figura 41 que ja ho indica.

Cada valor conté dos botons per gestionar la clau:

- **Eliminar:** quan es pitja, l'aplicació agafa el identificador del valor, el busca al backend i l'elimina, al mateix moment també l'elimina de la llista de valors que es mostren a la pantalla principal
- **Editar:** aprofitant el complement "Angular-xeditable"[31] es va fer que si es pitjava aquest botó, els paràmetres de l'element seleccionat es convertissin en element editables, com si fos un formulari, i allí es poguessin editar i guardar, fent que s'actualitzin a la base de dades (Figura 42).

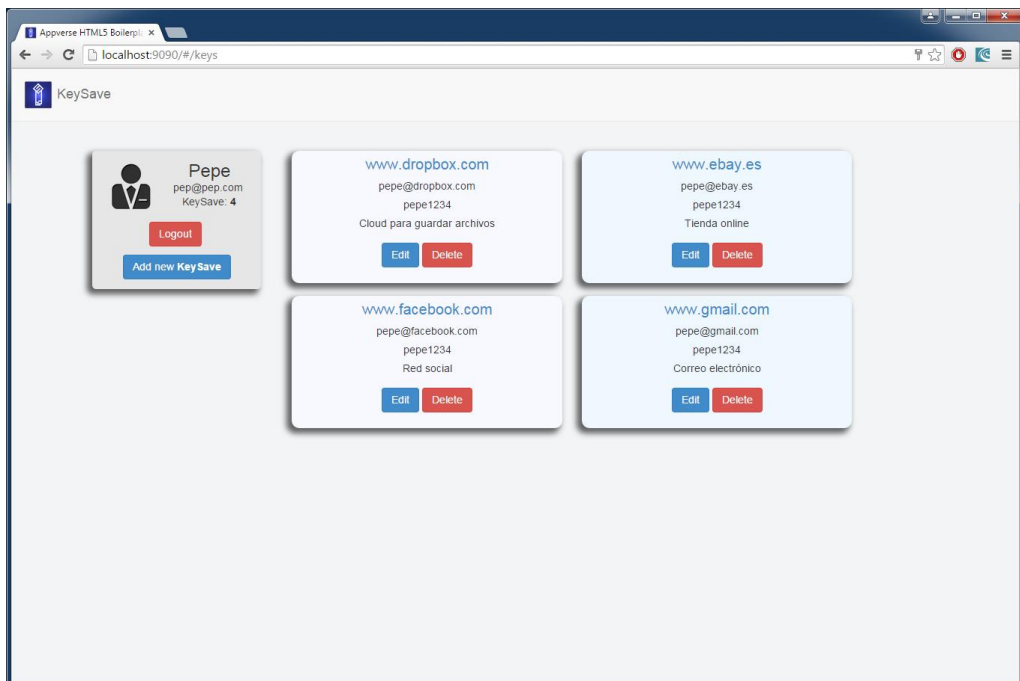


Figura 40: Llistat de claus (App web)

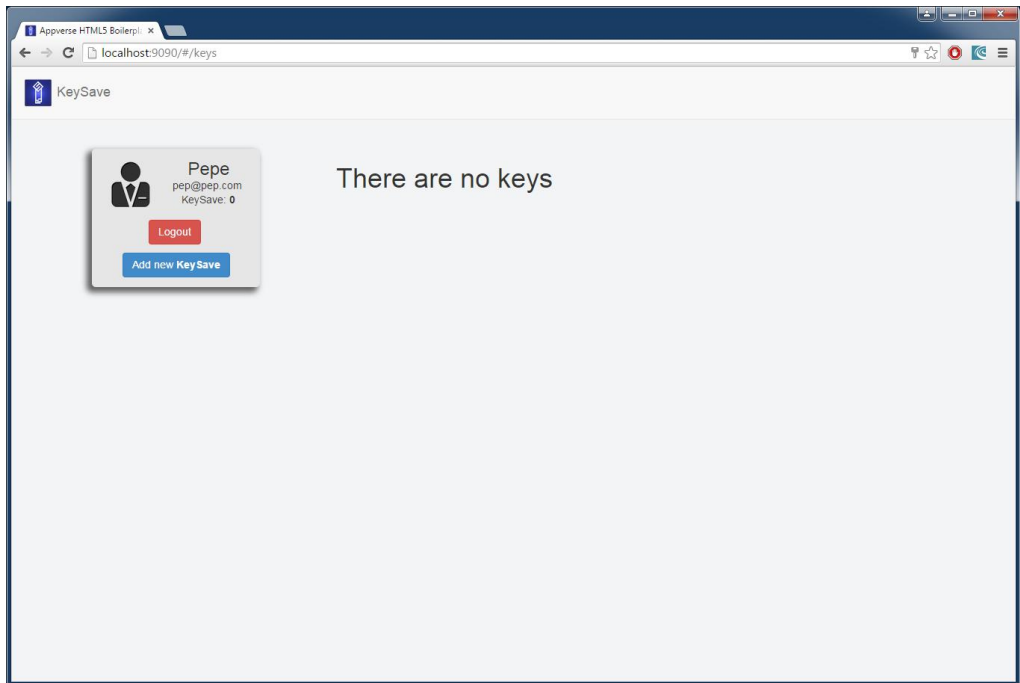


Figura 41: Pantalla principal sense valors (App web)

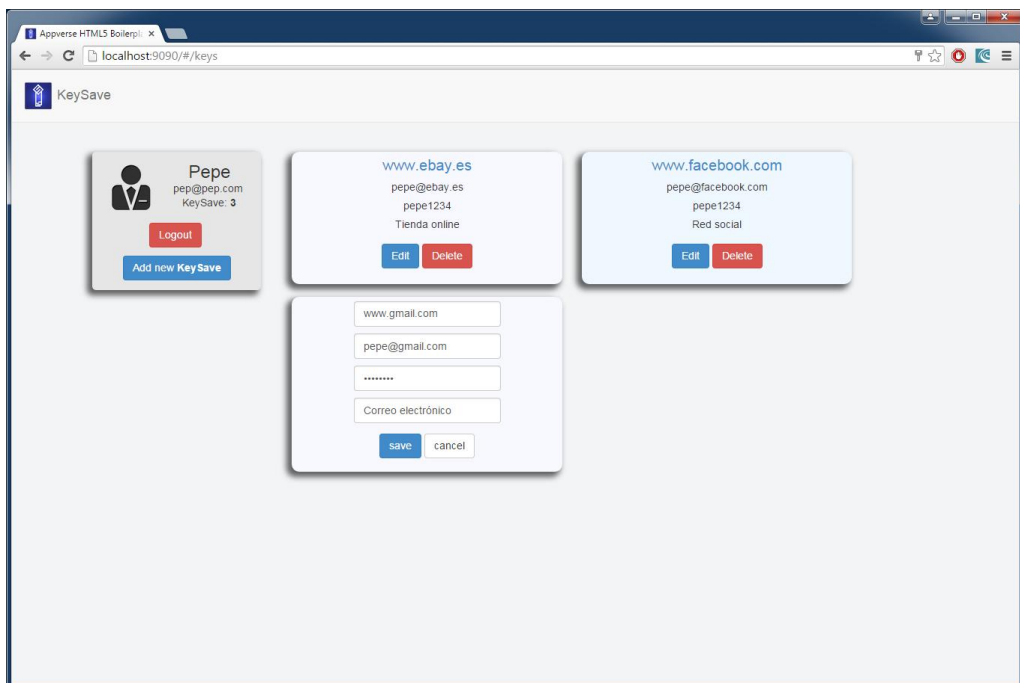


Figura 42: Editar un valor (App web)

9.3. Afegir claus

Dins del menú de l'usuari on hi ha la seva informació, també hi ha dos botons, un de logout per sortir de l'aplicació i tornar a la pantalla d'inici, i un per afegir una nova clau.

Aquest segon desplega un *modal*[32] que conté un formulari per introduir les dades (Figura 43).

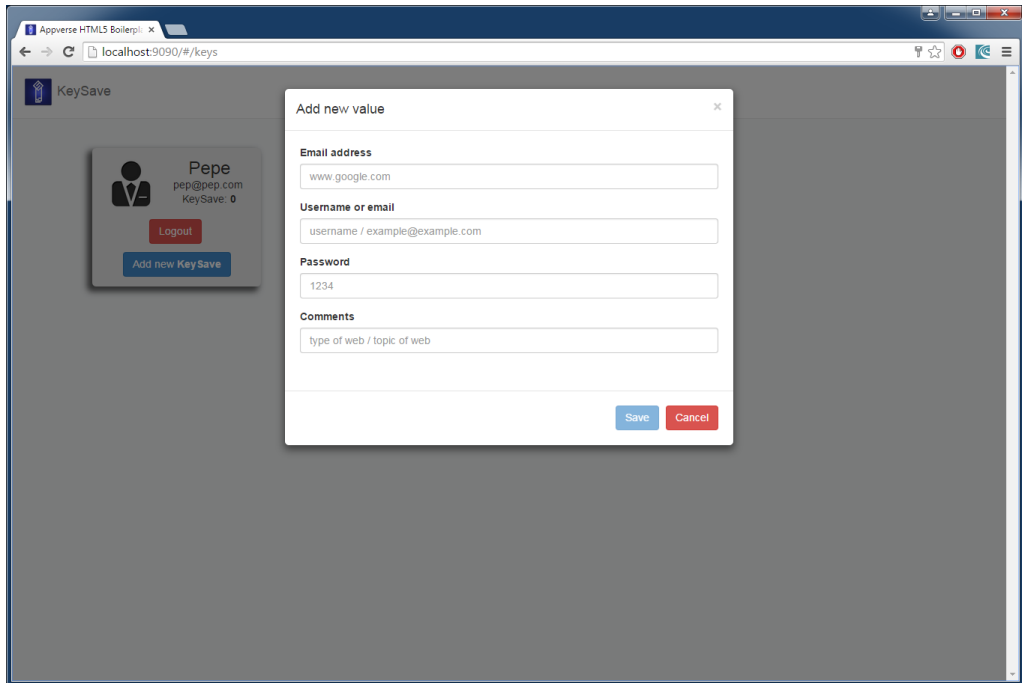


Figura 43: Formulari per afegir una nova clau (App web)

Una vegada omplert i pitjat el botó de guardar, l'aplicació xifra la contrasenya introduïda i després l'envia cap a la base de dades, on s'emmagatzemarà.

Seguidament s'afegirà a la llista de valors per mostrar-lo com una dada més a la pantalla principal.

10. Aplicació final de Ionic

Tal i com s'ha comentat en l'apartat anterior, a partir del codi de l'aplicació web, es va generar una aplicació híbrida.

Degut a que *Appverse web* utilitza *AngularJS*, es va buscar un framework que també l'utilitzes, pel que es va triar Ionic.

Tots els canvis fets de l'aplicació web a la híbrida van ser per aprofitar al màxim les funcionalitats creades per Ionic, per adaptar-se millor al dispositiu, i per semblar més atractiu.

Per exemple, la forma de mostrar els valors es farà utilitzant una de les directives creades pel framework anomenada “ion-list”[33] (es veurà més endavant).

10.1. Pantalla inicial, de login o registre

En obrir l'aplicació, igual que en totes les altres, la primera pantalla que es troba és la de login(Figura 44) amb la qual es pot accedir a la de registre(Figura 45).

Les dues consten d'un formulari, on la primera comprovarà que les credencials existeixin i en la segona es crearan.

Com l'anterior, el botó principal d'entrar o de registrar no estarà actiu fins que no estiguin els camps omplerts i amb el patró correcte, per exemple el correu té de tenir el símbol “@”.

La funcionalitat interna és la mateixa, enviar les dades al backend perquè aquest doni la resposta adequada:

- Si es el login, comprovi que l'usuari existeix
- Si es el registre, comprovi que no existeixi un usuari igual i l'emmagatzema

En cas d'error, es mostrarà un text justament damunt del botó del botó principal (botó d'entrar o botó de registre), com el de la Figura 46.

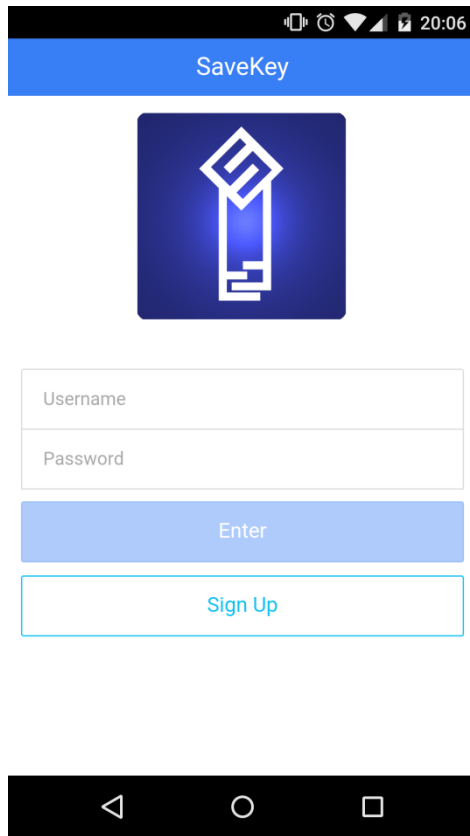


Figura 44: Pantalla de login (Ionic)

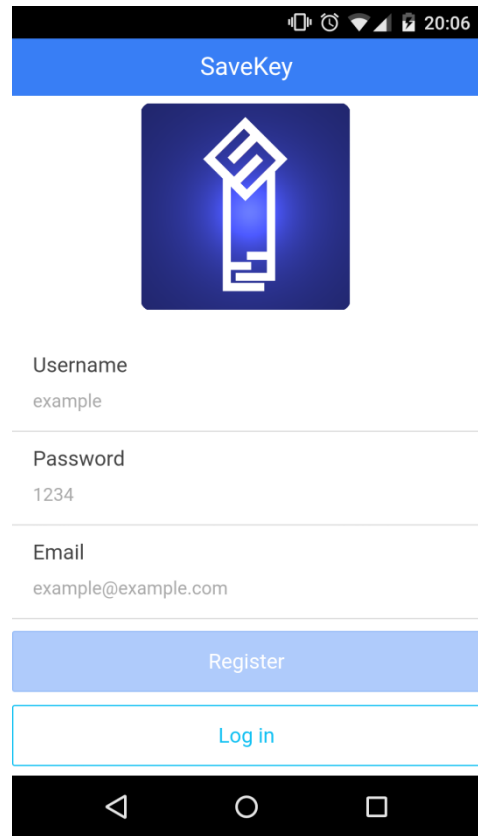


Figura 45: Pantalla de registre (Ionic)

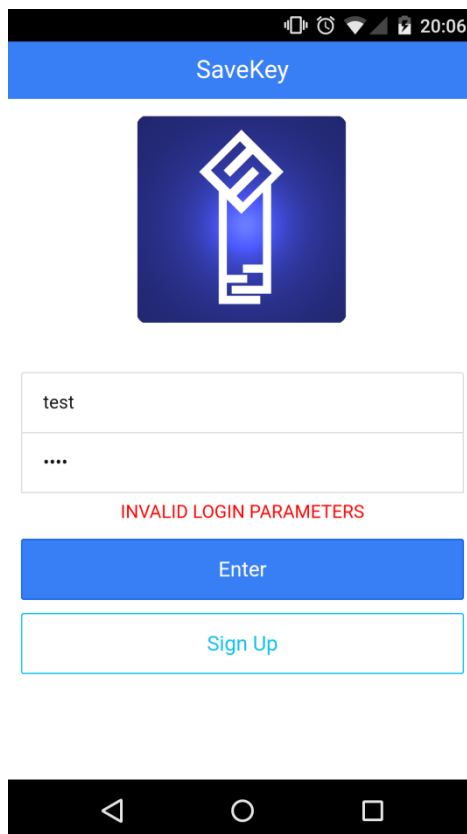


Figura 46: Pantalla de login amb error (Ionic)

10.2. Llistat de claus

Una vegada passada la primera pantalla, s'obtenen totes les claus emmagatzemades de l'usuari gràcies a la columna "user" de la base de dades de Parse.

A diferència de l'anterior aplicació, en aquesta petició solament s'obté els valors de la columna "id" i "site" que serà l'únic text que s'utilitzarà a la llista principal (Figura 47).

Si no hi ha valors dins del resultat de la petició, es mostrarà un text per indicar-ho (Figura 48).

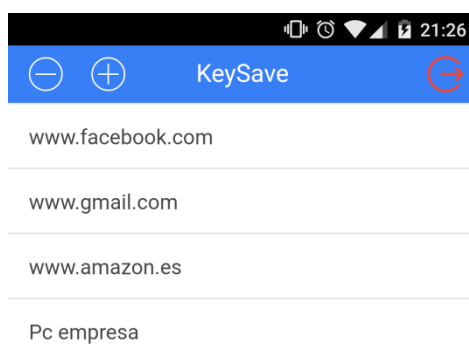


Figura 47: Llista de claus (Ionic)

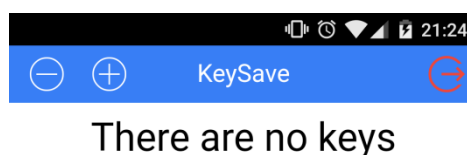


Figura 48: Pantalla principal sense dades (Ionic)

El tipus de llista utilitzat per generar la pantalla principal (*ion-list*) conté uns components que permeten interactuar amb ella per afegir un botó d'eliminar valors (botó de la part superior a l'esquerra amb un símbol "-"). Aquest fa desplegar un botó al costat de cada element de la llista, així com es veu a la Figura 49, i si es pitja un d'aquest, eliminarà la clau de la base de dades.

El botó de la part superior a la dreta (botó de color roig) serveix per sortir de l'aplicació, és a dir, fer un logout i tornar a la pàgina d'inici.

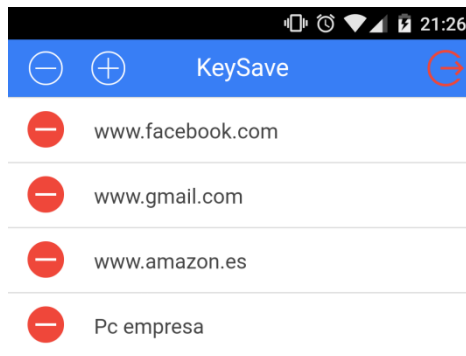


Figura 49: Eliminar valors de la llista (Ionic)

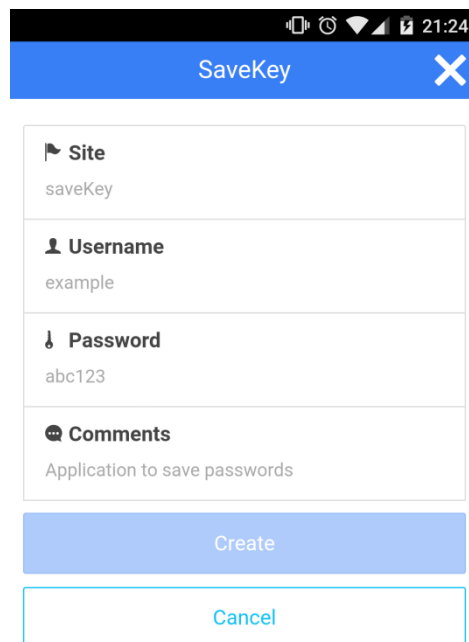
10.3. Afegir clau

El botó de la barra superior amb el símbol “+” servirà per obrir el formulari que servirà per afegir més claus (Figura 50).

Una vegada es pitjo el botó de guardar, farà els mateixos passos que les anteriors aplicacions:

- Xifrar la contrasenya amb l'algoritme AES
- Enviar-la al backend per guardar-la
- Afegir-la a la llista principal.

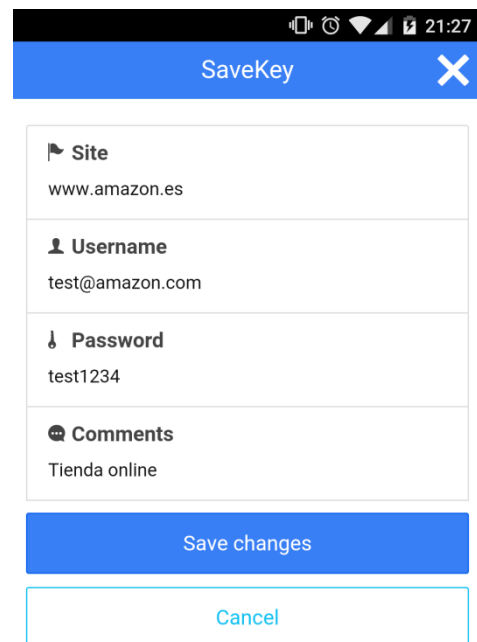
Aquest formulari també s'utilitzarà per quan es consulti un valor, de forma que al obtenir els seus paràmetres, els ficarà dins dels inputs (camps de text) per si es volen editar i guardar els canvis (Figura 51).



The screenshot shows a mobile application interface for creating a new password entry. At the top, there is a blue header bar with the text 'SaveKey' and a white 'X' icon. Below the header, there is a form with four sections: 'Site' (saveKey), 'Username' (example), 'Password' (abc123), and 'Comments' (Application to save passwords). At the bottom of the form, there are two buttons: a blue 'Create' button and a white 'Cancel' button with a blue border. The status bar at the top shows the time as 21:24.



Figura 50: Formulari per crear un nou valor (Ionic)



The screenshot shows the same 'SaveKey' form as in Figure 50, but with pre-filled data for editing. The 'Site' field contains 'www.amazon.es', 'Username' contains 'test@amazon.com', 'Password' contains 'test1234', and 'Comments' contains 'Tienda online'. The 'Create' button is replaced by a blue 'Save changes' button. The 'Cancel' button remains. The status bar at the top shows the time as 21:27.



Figura 51: Editar/Consultar una clau (Ionic)

11. Conclusions

El desenvolupament d'aquest projecte ha aportat molts coneixements i motivacions de cara al futur.

Ha ajudat a conèixer i aplicar les noves funcionalitats que incorporen les noves versions d'Android, a utilitzar mètodes i classes per crear components que estiguin atents en tot moment per qualsevol canvi en l'aplicació, ha desenvolupat una aplicació web i/o Ionic utilitzant frameworks i tecnologies d'última generació, etc.

Ajuntant tots aquest coneixements obtinguts s'ha arribat a desenvolupar l'aplicació plantejada al principi del projecte, complint tots els objectius.

L'únic que no s'ha pogut fer és el poder compartir les dades entre les aplicacions Javascript i l'aplicació Android degut a que el resultat del xifratge AES no és el mateix, i per tant, les dades xifrades en un no es poden desxifrar en l'altre.

Degut a la falta de temps no s'ha pogut fer una bona investigació per resoldre aquest problema, per això aquesta part es deixa dins dels plans de futur, per trobar una solució òptima.

Una vegada resolt el problema, s'ha de fer que la resta de dades també s'encryptin abans de ser emmagatzemades, igual que la contrasenya, de forma que la base de dades quedi completament il·legible. Actualment no s'ha fet per poder veure els resultats de les execucions i proves, però per fer-ho solament fa falta passar tots els valors al mètode que s'encarrega del xifrat i no solament la contrasenya.

Una nova funcionalitat que es pot incorporar més endavant és la d'un login diferent, és a dir, utilitzant patrons o algun tipus d'autenticació més moderna com pot ser l'autenticació facial[34], o inclús inventar algun tipus de contrasenya amb icones o dibuixos.

Aquestes idees innovadores són les que aporten interès al projecte per part dels clients, ja que sempre es reclama el més bo i nou.

Pel que fa a la planificació del projecte, s'ha seguit quasi igual, inclús en la part de web es va avançar més del esperat, però després, amb el canvi a *Ionic*, el seu aprenentatge va aturar aquest avanç tant ràpid.

També va afectar la combinació amb la vida laboral, altres assignatures i sobretot altres treballs que no estava previst que arribessin a ser tant complicats i llargs fent estar-se hores per aconseguir solucionar-lo.

12. Glossari

- Boilerplate: plantilla preparada sense modificacions des de la qual es pot començar a programar. Un exemple pot ser el d'una casa, on et donen l'estructura feta i només fa falta adaptar-la al gust i/o necessitats, és a dir, ficar els mobles, pintar-la, etc.
- SDK: Software Development Kit
- SO: Sistema operatiu

13. Bibliografia

- [1] **Ionic.** [En línia]
<http://ionicframework.com/>
- [2] **Universitat politècnica de València.** *Arquitectura de Android.* [En línia] [Consulta: 30/03/2015]
<http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android>
- [3] **Parse.** [En línia] [Consulta: 30/03/2015]
<https://parse.com/>
- [4] **Sass.** [En línia] [Consulta: 30/03/2015]
<http://sass-lang.com/>
- [5] **AngularJS.** [En línia] [Consulta: 30/03/2015]
<https://angularjs.org/>
- [6] **Apache Cordova.** [En línia] [Consulta: 30/03/2015]
<https://cordova.apache.org/>
- [7] **NodeJS.** [En línia] [Consulta: 30/03/2015]
<https://nodejs.org/>
- [8] **AppVerse.** [En línia] [Consulta: 30/03/2015]
<http://appverse.org/#/home>
- [9] **Salesforce developers.** *Native, HTML5, or Hybrid: Understanding your mobile application development options* [En línia] [Consulta: 10/06/2015]
https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options
- [10] **Justinmind.** [En línia]
<http://www.justinmind.com/>
- [11] **AES Encryption.** [En línia] [Consulta: 25/04/2015]
<http://aesencryption.net/>
- [12] **Wikipedia.** *SHA-2.* [En línia] [Consulta: 25/04/2015]
<http://es.wikipedia.org/wiki/SHA-2>
- [13] **Developers.** *AsyncTask* [En línia] [Consulta: 15/04/2015]
<http://developer.android.com/reference/android/os/AsyncTask.html>
- [14] **Developers.** *SharedPreferences.* [En línia] [Consulta: 10/04/2015]
<http://developer.android.com/reference/android/content/SharedPreferences.html>
- [15] **Developers.** *SharedPreferences.Editor.* [En línia] [Consulta: 10/04/2015]

<http://developer.android.com/reference/android/content/SharedPreferences.Editor.html>

- [16] **Developers.** *Managing Network Usage*. [En línia] [Consulta: 02/06/2015]
<http://developer.android.com/training/basics/network-ops/managing.html>
- [17] **Developers.** *BroadcastReceiver*. [En línia] [Consulta: 02/06/2015]
<http://developer.android.com/training/basics/network-ops/managing.html>
- [18] **Developers.** *Supporting Different Languages*. [En línia] [Consulta: 13/04/2015]
<http://developer.android.com/training/basics/supporting-devices/languages.html>
- [19] **Library of congress.** *ISO 639.2*. [En línia] [Consulta: 13/06/2015]
http://www.loc.gov/standards/iso639-2/php/code_list.php
- [20] **Developers.** *TelephonyManager*. [En línia] [Consulta: 02/06/2015]
<http://developer.android.com/reference/android/telephony/TelephonyManager.html>
- [21] **Wikipedia.** *IMEI*. [En línia] [Consulta: 05/06/2015]
<https://es.wikipedia.org/wiki/IMEI>
- [22] **Wikipedia.** *Salt*. [En línia] [Consulta: 02/06/2015]
[https://en.wikipedia.org/wiki/Salt_\(cryptography\)](https://en.wikipedia.org/wiki/Salt_(cryptography))
- [23] **Developers.** *Patterns – Email_Address*. [En línia] [Consulta: 26/05/2015]
http://developer.android.com/reference/android/util/Patterns.html#EMAIL_ADDRESS
- [24] **Developers.** *BaseAdapter*. [En línia] [Consulta: 26/05/2015]
<http://developer.android.com/reference/android/widget/BaseAdapter.html>
- [25] **GitHub.** *Android-floating-button*. [En línia] [Consulta: 30/05/2015]
<https://github.com/futuresimple/android-floating-action-button>
- [26] **Developers.** *AdapterView.OnItemClickListener*. [En línia] [Consulta: 10/04/2015]
<http://developer.android.com/reference/android/widget/AdapterView.OnItemClickListener.html>
- [27] **Developers.** *AdapterView*. [En línia] [Consulta: 10/04/2015]
<http://developer.android.com/reference/android/widget/AdapterView.html>
- [28] **Developers.** *Intent*. [En línia] [Consulta: 30/03/2015]
<http://developer.android.com/reference/android/content/Intent.html>

- [29] **GitHub**. *Appverse-web-html5-boilerplate*. [En línia] [Consulta: 30/03/2015]
<http://developer.android.com/reference/android/content/Intent.html>
- [30] **Code google**. *CryptoJS*. [En línia] [Consulta: 08/04/2015]
<https://code.google.com/p/crypto-js/>
- [31] **Angular-xeditable**. [En línia] [Consulta: 08/04/2015]
<http://vitalets.github.io/angular-xeditable/>
- [32] **Getbootstrap**. *Modal*. [En línia] [Consulta: 08/04/2015]
<http://getbootstrap.com/javascript/#modals>
- [33] **Ionic**. *ion-list*. [En línia] [Consulta: 20/04/2015]
<http://ionicframework.com/docs/api/directive/ionList/>
- [34] **KeyLemon**. [En línia] [Consulta: 15/06/2015]
<https://www.keylemon.com/>
- [35] **Git**. [En línia] [Consulta: 30/03/2015]
<https://git-scm.com/>
- [36] **Ruby**. [En línia] [Consulta: 30/03/2015]
<https://www.ruby-lang.org/es/>
- [37] **Compass**. [En línia] [Consulta: 30/03/2015]
<http://compass-style.org/>
- [38] Jordi Flamarich Zampalo. *Disseny d'interfícies per a dispositius mòbils*. UOC. [Consulta: 10/06/2015]
<http://cvapp.uoc.edu/autors/MostraPDFMaterialAction.do?id=217710>

14. Annexos

14.1. Execució de l'aplicació web

L'aplicació web es troba dins l'arxiu "[Apps\Codi font\SaveKey-Web.zip](#)" el qual s'ha de descomprimir.

Els requeriments previs a la seva execució són:

- Tenir *NodeJS*[7]: descarregar instal·lable de la web
- Tenir *Git*[35]: descarregar instal·lable de la web
- Tenir *Ruby*[36]: descarregar instal·lable de la web
- Tenir *Compass*[37]: instal·lar a partir de *Ruby*

Una vegada es tinguin tots, ja es poden seguir els passos marcats dins del git de *Appverse*[28] que són:

1. Afegir les eines:

```
$ npm install -g bower
```

2. Configuració del projecte:

```
$ npm install
```

3. Execució del projecte

```
$ grunt server
```

Si tot a sortit bé, automàticament es tindria d'obrir el navegador per defecte mostrant la pantalla inicial (pantalla de login) de l'aplicació web.

14.2. Execució de l'aplicació híbrida

Dins la carpeta del treball hi ha l'arxiu "[Apps\Executables\SaveKey Ionic.apk](#)" que serveix com a instal·lable de l'aplicació híbrida pels dispositius amb sistema operatiu Android.

El projecte es troba dins el fitxer "[Apps\Codi font\SaveKey-Ionic.zip](#)" el qual s'ha de descomprimir.

Els requisits previs són:

- Tenir instal·lat NodeJS: descarregar instal·lable de la web
- Tenir Git: descarregar instal·lable de la web

Amb continuació ja es pot fer la instal·lació de Ionic:

```
$ npm install -g cordova ionic
```

Finalment, dins al projecte solament fa falta instal·lar les llibreries necessàries pel projecte:

```
$ bower install
```

I la llibreria pel Sass:

```
$ npm install -g gulp  
$ npm install -g gulp-sass
```

Ja sol falta executar el projecte de la forma preferida tot i que s'aconsella fer-ho en un dispositiu físic :

1. Afegir la plataforma dins al projecte

```
$ ionic platform add android
```

2. Executar l'aplicació

```
$ ionic run android
```

14.3. Execució de l'aplicació Android

Android disposa directament de l'arxiu en format "apk" el qual ja serveix per instal·lar-lo en qualsevol dispositiu. Aquest es troba a "[Apps\Executables\SaveKey Android.apk](#)".

Pel que fa al codi font esta situat al mateix lloc que els anteriors ("[Apps\Codi font\SaveKey-Android.zip](#)").

14.4. Vídeos demostratius

En totes les aplicacions s'ha realitzat un vídeo demostratiu de la seva funcionalitat, els quals es poden trobar dins la carpeta de "[Vídeos demostratius](#)".

S'aconsella reproduir els vídeos amb un reproductor de l'estil "BSPlayer"³.

14.4.1. Android

El vídeo de l'aplicació Android es troba dins de l'arxiu comprimit "[Vídeos demostratius\SaveKey-Android.zip](#)". Esta dividit en dues parts degut a la seva mida.

A la primera part es pot veure com es crea un nou usuari i com s'afegeixen dades i es gestionen utilitzant la nova compta. També es sincronitzen els valors amb el backend.

En la segona part, s'executa l'aplicació sense Internet per mostrar que si aquell usuari ja havia entrat, no fa falta connexió per validar les credencials. A més a més, s'han afegit dades a la base de dades remota de forma que quan es sincronitzin les claus, es pugui veure com es descarreguen i s'afegeixen a la llista principal.

³ <http://www.bsplayer.com/>

14.4.2. Ionic

Aquest vídeo esta situa a "[Vídeos demostratius\SaveKey-Ionic.zip](#)".

Es pot veure com es crea un nou usuari, juntament amb els errors que es mostren en cas que els paràmetres introduïts no siguin vàlids.

Seguidament es veu com es creen les noves claus, com es consulten i com s'editen, i finalment s'esborren.

14.4.3. Aplicació web

El vídeo referent a l'aplicació web és el "[Vídeos demostratius\SaveKey-Web.zip](#)"

A la demostració es veu el mateix que l'apartat anterior:

- Es crea un nou usuari
- Es creen noves claus
- S'editen i s'esborren