Monitoratge constant del ritme cardíac amb dispositius portables

Treball Fi de Grau

Grau en Enginyeria Informàtica. Pla 2012

Document: Memòria
Alumne: Òscar Raya I Casanova
Director/Tutor: Carles Pous Sabadí, Maria Beatriz Lopez Ibañez
Departament: Enginyeria Elèctrica, Electrònica i Automàtica
Àrea: Enginyeria de Sistemes i Automàtica
Convocatòria: 09/2018

Índex

1 Introducció **4**
1.1 Motivacions 4
1.2 Propòsit 5
1.3 Objectius 5
1.4 Abast 5

2 Estudi de viabilitat **6**
2.1 Recursos tecnològics 6
2.2 Costos 7
2.3 Viabilitat legal 8

3 Metodologia **9**
3.1 Que és el desenvolupament en espiral? 9
3.1.1 Avantatges 9
3.1.2 Inconvenients 10
3.2 Perquè s'ha escollit el desenvolupament en espiral? 10
3.3 Fases del desenvolupament en espiral 10
3.3.1 Model original de Boehm 11
3.3.2 Model típic de sis regions 11
3.3.3 Model win-win 12

4 Planificació **13**

5 Marc del treball **18**
5.1 Els dispositius vestibles 18
5.1.1 Polseres intel·ligents 18
5.1.2 Altres dispositius 19
5.2 Aplicacions de monitoratge amb dispositius vestibles 20
5.3 Equip de treball 20

6 Requisits del sistema **21**
6.1 Requisits no funcionals 21
6.2 Requisits funcionals 22
6.2.1 Requisits del servidor 22
6.2.2 Requisits de l'aplicació 23

7 Estudis i decisions **24**
7.1 Programari . . . 24
7.1.1 Android . . . 24
7.1.2 IntelliJ Idea . . . 25
7.1.3 MySQL . . . 25
7.1.4 SQLite . . . 25
7.1.5 Retrofit 2.0 . . . 25
7.1.6 Spring vs Struts . . . 26
7.1.7 JSON Web Token . . . 26
7.2 Maquinari . . . 27
7.2.1 Polsera . . . 27
7.2.2 Mòbil . . . 28
7.2.3 Servidor . . . 28

8 Anàlisis i disseny del sistema **29**
8.1 Anàlisis del sistema . . . 30
8.1.1 Diagrames de casos d'ús . . . 30
8.1.2 Fitxes de cas d'ús . . . 31
8.1.3 Diagrames de classes . . . 40
8.2 Disseny del sistema . . . 49
8.2.1 Interfície d'usuari . . . 49
8.2.2 Diagrama d'activitats de l'aplicació . . . 53

9 Implementació i proves **54**
9.1 Implementació del servidor . . . 54
9.2 Implementació de l'aplicació . . . 57
9.2.1 El servei de Bluetooth . . . 57
9.2.2 Gestió de la connexió Bluetooth . . . 58
9.2.3 El servei de Crides Rest . . . 59
9.2.4 Protocol Mi Band 1S . . . 59
9.2.5 Emmagatzematge i enviament de les dades . . . 60
9.2.6 Mantenir els serveis funcionant . . . 60
9.3 Proves . . . 61
9.3.1 Proves del servidor . . . 61
9.3.2 Proves de l'aplicació . . . 61

10 Implantació i resultats **62**
10.1 Implantació del servidor . . . 62
10.2 Implantació de l'aplicació . . . 62
10.3 Resultats . . . 63

11 Conclusions **64**

12 Treball Futur **65**

Bibliografia **67**

13 Annexos **69**
13.1 Barry Boehm 69
13.2 Llei Orgànica de Protecció de Dades de Caràcter Personal 70
13.3 Secure Hash Algorithm 70
13.4 JSON Web Token 71

14 Manuals **72**
14.1 Manuals d'instal·lació 72
14.1.1 Instal·lació del servidor 72
14.1.2 Instal·lació de l'applicació 76
14.2 Manual d'usuari 78
14.2.1 Iniciar sessió 78
14.2.2 Registrar-se 79
14.2.3 La pantalla principal 79
14.2.4 Tancar la sessió 81
14.2.5 Tancar l'aplicació 81
14.2.6 Emparellar una polsera 81
14.2.7 Desemparellar una polsera 83
14.2.8 La pantalla del dispositiu 84
14.2.9 Captura del ritme cardíac 85

Capítol 1

Introducció

Els *wereables* o dispositius portables s'han anat fent més populars dia a dia gràcies al seu abaratiment i a les diverses funcionalitats que ofereixen. Molts d'ells permeten monitorar accions o constants vitals de l'usuari tals com els passos que dóna o bé el seu ritme cardíac. El laboratori d'Enginyeria de Control i Sistemes Intel·ligents de la UdG (eXiT) treballa amb diversos d'aquests dispositius. Entre ells podem trobar la Mi Band 1s de Xiaomi o la Fitbit de Amazon, els quals, ofereixen la possibilitat de fer un recompte de passos que fa l'usuari aixi com monitorar el ritme cardíac.

En l'actualitat han sorgit molts projectes que utilitzen aquests dispositius en tota classe de situacions. En alguns llocs, fins i tot, es comença a fer servir en l'àmbit mèdic per portar una mesura constant dels pacients que necessiten més vigilància.

En aquest projecte ens centrarem en el monitoratge del ritme cardíac fent ús de la banda Mi Band 1s. Actualment però aquesta, i d'altres del mercat, no ofereixen de forma oficial la possibilitat de recollir de forma constant i ininterrompuda les dades monitorades en un servidor centralitzat, així que és proposa la implementació d'una aplicació mòbil per tal de fer-ho possible.

1.1 Motivacions

La principal motivació, a l'hora de dur a terme aquest projecte, ha estat l'interès per veure com funcionava la connectivitat amb aquests dispositius portables. A més, m'ha semblat d'especial interès el funcionament de les polseres a l'hora de fer el càlcul del ritme cardíac, ja que és una tecnologia que no coneixia.

D'altra banda, també interessa disposar d'aquest mecanisme de recol·lecció de dades per tal de poder fer recerca i investigació amb les dades obtingudes, així com oferir noves eines de docència per a la Universitat.

Per últim, sempre motiva intentar crear des de zero una aplicació amb tots els passos que això comporta.

1.2 Propòsit

El propòsit d'aquest projecte és el de desenvolupar una aplicació per a dispositius mòbils la qual ens permeti monitorar el ritme cardíac de l'usuari a través de la Mi Band i enviar totes les dades recol·lectades a un servidor, que també desenvoluparem, per tal de poder processar-les en el futur o bé simplement visualitzar-les.

Per tal d'aconseguir-ho s'haurà de fer l'anàlisi de la connectivitat Bluetooth de les Mi Band i utilitzar el seu protocol per tal de recol·lectar les dades preses pel dispositiu.

Amb la finalització d'aquest projecte, estarem oferint la possibilitat de poder recol·lectar, en un servidor, totes aquestes dades de diferents usuaris d'una forma senzilla i barata. Ja que només es necessita un dispositiu mòbil amb Android i una Mi Band 1S, que són molt barates en comparació als dispositius de mesura professionals, es podrà fer servir en practiques de la Universitat relacionades amb la recol·lecció i tractament de dades. Cal dir, però, que aquests dispositius no compten amb el mateix grau de precisió que els equips de mesura professionals. Així doncs no es podrien utilitzar en situacions on la precisió fos d'especial interès.

1.3 Objectius

Els objectius del projecte són:

- Dissenyar i implementar una aplicació d'Android que demani el ritme cardíac que és mesurat per la Mi Band. L'aplicació haurà d'enviar, de forma periòdica, les dades recol·lectades a un servidor.
- Dissenyar i implementar una base de dades que contingui la informació de cada usuari, així com les dades obtingudes d'aquests al llarg del temps.
- Dissenyar i implementar un servidor, amb accés a la base de dades, amb el seu protocol per tal de poder comunicar-se amb l'aplicació.
- Explorar les diferents funcionalitats que es poden extreure de processar les dades obtingudes per l'usuari.
- Analitzar la viabilitat del monitoratge constant amb aquests dispositius en l'actualitat i en el futur.

1.4 Abast

En aquest projecte no es busca fer una aplicació comercialitzable sinó tan sols un prototip que ens doni una base considerablement gran en el que respecta al monitoratge de dades. Així doncs tot i que la seguretat és molt important quan es tracten amb dades de caràcter personal, en aquest projecte no se li ha donat prioritat i ens hem centrat més en un aspecte més tècnic.

Capítol 2

Estudi de viabilitat

En aquest capítol es mostren tots els recursos necessaris per a tal de dur a terme el desenvolupament del projecte, així com els possibles costos i la problemàtica d'aquest.

2.1 Recursos tecnològics

Representa les eines (hardware i software) necessàries per a tal de desenvolupar i mantenir l'aplicació.

- Recursos desenvolupament aplicació mòbil:
  - Una banda Mi Band 1S de Xiaomi
  - Un mòbil amb Android 5.0 o superior que suporti Bluetooth LE i tingui accés a Internet
  - Un cable USB a micro USB per tal de connectar el mòbil amb l'ordinador i poder testejar l'aplicació
  - Un ordinador amb IntelliJ IDEA per tal de programar l'aplicació
  - SDKs d'Android API 21 i superiors i SDK de Java per tal de poder compilar l'aplicació
  - Diverses llibreries gratuïtes de Java i d'Android com ara Retrofit 2.0 per a tal de gestionar la connexió amb el servidor
- Recursos desenvolupament servidor:
  - Un ordinador que actuí com a servidor
  - Un *router* per donar accés a Internet al servidor
  - SDK de Java per tal de compilar l'aplicació del servidor
  - MySql 8.0 per mantenir la Base de Dades

- IntelliJ IDEA amb Spring Framework per tal de programar l'API REST del servidor
- Apache Tomcat per tal s'executar el servidor
- Postman per testejar les crides rest

2.2 Costos

Els costos del projecte estaran conformats pel preu i manteniment dels recursos necessaris així com del personal necessari per al desenvolupament i manteniment de l'aplicació.

A la taula [2.1] podem veure els costos dels recursos necessaris.

Recurs	Cost (€)
Ordinador/Servidor	650.00
Mi Band 1S de Xiaomi	20.00
Mòbil amb Android 5.0 o superior	130.00
Cable USB (inclòs amb el mòbil)	0.00
Total	800.00

Taula 2.1: Costos dels recursos.

Per altra banda, els costos del personal es veuen reflectits a la taula [2.2]. Les tasques estan pensades per dur-se a terme per diferents perfils de personal, però han estat efectuades per un becari pel preu de **6,75 €/hora**.

Tasca	Perfil	Hores	Cost (€)
Anàlisi i Disseny de l'aplicació	Dissenyador	100	675.00
Anàlisi i Disseny de la Base de Dades	Programador	40	270.00
Anàlisi protocol Mi Band 1S	Programador	20	135.00
Implementació de l'aplicació	Programador	170	1147.50
Implementació Servidor i BD	Programador	30	202.50
Documentació	Programador	48	324.00
Total	–	408	2754.00

Taula 2.2: Costos del personal.

Cal destacar que el programari utilitzat és lliure o bé s'han utilitzat versions per a ús educatiu/no comercial. Així doncs, si el projecte es vol utilitzar fora d'un àmbit educatiu/no comercial, s'hauran de pagar les llicències comercials dels programes indicats a la taula [2.3].

Programari	Cost (€/mes)
IntelliJ IDEA Ultimate	14.90
Total	14.90

Taula 2.3: Costos del programari.

També s'han de tenir en compte les despeses reflectides a la taula [2.4], de les quals només se'n tindrà en compte un petit percentatge, ja que aquestes despeses són atribuïdes a molts projectes diferents.

Despesa	Cost (€/mes)
Internet fibra simètrica 300 Mbps	40.00
Electricitat gastada pel servidor	20.00
Local físic del servidor	240.00
Total	300.00

Taula 2.4: Costos de les despeses.

S'ha establert que la despesa d'aquest projecte representa un 1% del cost total de les despeses de manteniment, així doncs el manteniment de l'aplicació tindria una despesa de 3.00 € al mes.

En total l'aplicació hauria tingut un cost de **3554 €** més una despesa de **3.00 €** al mes. Cal recordar, però, que l'aplicació s'ha desenvolupat en un entorn Universitari, que ja compta amb la majoria de recursos necessaris, de manera que el seu cost real tan sols ha estat el de la compra de la Mi Band 1S i el sou del programador. Així que el cost real del projecte ha estat de **2774 €**.

2.3 Viabilitat legal

Aquesta aplicació envia dades de caràcter personal a la base de dades d'un servidor. Aquestes dades es poden considerar dades mèdiques. Per tal d'evitar qualsevol problema legal relacionat amb la protecció de dades de caràcter personal, s'ha optat per dur a terme una desidentificació de les dades, de manera que ningú pot saber de qui són les dades guardades.

Capítol 3

Metodologia

Per tal de desenvolupar aquest projecte s'ha escollit una metodologia en la qual el desenvolupament es du a terme en espiral.

3.1 Que és el desenvolupament en espiral?

El desenvolupament en espiral és una metodologia de cicle de vida del software definida per primer cop el 1986 per Barry Boehm[13.1].

Aquest model es desmarca del model tradicional en cascada i aporta un nou enfocament que se centra en els possibles riscos de desenvolupament.

Primerament, s'avaluen les diferents alternatives de desenvolupament i s'opta per la de menor risc. Un cop obtingut un primer prototip, es tornen a avaluar les noves alternatives de desenvolupament que poden sorgir del prototip i es torna a escollir la de menor risc.

S'anirà fent aquest procés iteratiu, generant una espiral de desenvolupament, fins que el producte sigui acceptable. Un cop acceptat el producte, s'utilitza aquest mateix model per actualitzar-lo i mantenir-lo.

3.1.1 Avantatges

- S'adapta al llarg del desenvolupament i s'incorpora en el manteniment.
- Reacciona millor que altres metodologies davant de riscos en cadascun dels seus nivells evolutius.
- Si s'aplica correctament, la consideració de riscos tècnics a cada etapa del projecte, redueix els riscos abans que es transformin en problemes.
- En grans sistemes la productivitat és major.

3.1.2 Inconvenients

- No s'aconsella fer-lo servir en sistemes petits a causa de la seva elevada complexitat.
- És costos i necessita molt de temps de desenvolupament.
- Requereix experiència en la identificació de riscos.
- Pot semblar un model difícil de controlar per als clients.

3.2 Perquè s'ha escollit el desenvolupament en espiral?

Quan estava iniciant el projecte em vaig trobar amb molts dubtés a l'hora de dur a terme el desenvolupament i, tot i que no és recomanable fer servir el model en espiral per a projectes de poca envergadura, vaig trobar adequat fer servir aquest model per diverses raons:

La primera de totes era la necessitat de fer servir un model flexible. Els models clàssics, com ara el desenvolupament en cascada, requereixen una anàlisi i disseny previs de l'aplicació, els quals han de ser els definitius. En canvi, el desenvolupament en espiral incorpora l'ús de prototips i permet canvis en el disseny i en qualsevol de les etapes de producció.

Vaig trobar molt atractiva la idea de poder implementar un primer prototip al qual pogués anar millorant afegint funcionalitats tenint en compte els riscos de desenvolupament que em poguessin portar cadascuna d'elles.

També volia fer un projecte el qual sempre es pogués millorar i trobo que, per a un TFG, és adequat presentar un prototip, el qual pugui ampliar-se en moltes direccions diferents, i no pas presentar una aplicació acabada i segellada.

Finalment, em veia incapaç de dissenyar tot un projecte d'aquesta grandària sense caure en errors i problemes. I, tot i que es necessita saber avaluar els riscos per fer servir aquesta metodologia, trobo que és més fàcil avaluar els riscos i evitar-los que no pas solucionar els possibles problemes que podria tenir més endavant originats per un mal disseny.

Cal mencionar que al llarg d'aquest projecte s'han tingut en compte les valoracions i avaluacions tant del tutor com d'una de les companyes del grup de recerca, de manera que aquests actuarien amb el rol de clients, directius i/o administradors del projecte.

3.3 Fases del desenvolupament en espiral

El model en espiral divideix les tasques en diverses **regions de tasques**. Cada una conte un conjunt de tasques que s'adapten al projecte. S'itera contínuament sobre aquestes regions per tal de desenvolupar el projecte. Existeixen diverses modificacions del model espiral amb diferent nombre de regions.

3.3.1 Model original de Boehm

En el model original de Boehm no hi ha un nombre determinat d'iteracions. Consta de 4 regions de tasques:

- Planejament: es determinen els objectius, les alternatives i les restriccions
- Anàlisi de riscos: s'identifiquen i/o es resolen els riscos
- Enginyeria: es desenvolupa el producte
- Avaluació: s'avaluen els resultats obtinguts (el client pot intervenir)

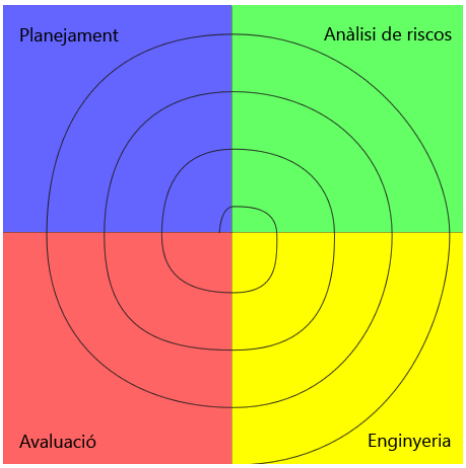

Figura 3.1: Diagrama model en espiral

3.3.2 Model típic de sis regions

Aquest model és una extensió del model original en què s'afegeixen dues noves regions per tal de millorar el tracte amb el client:

- Comunicació amb el client: tasques per tal d'establir comunicació entre desenvolupador i client
- Planejament: es determinen els objectius, les alternatives i les restriccions
- Anàlisi de riscos: s'identifiquen i/o es resolen els riscos
- Enginyeria: es desenvolupa el producte
- Construcció i adaptació: construir, provar, instal·lar i proporcionar suport d'usuari
- Avaluació: s'avaluen els resultats obtinguts (el client pot intervenir)

3.3.3 Model win-win

Aquest ultim model defineix un conjunt d'activitats de negociació al principi de cada pas al voltant de l'espiral. Les noves activitats definides són:

- Identificació del sistema o subsistemes claus dels directius
- Determinació de les condicions de victòria dels directius
- Negociació de les condicions de victòria amb tots els afectats

Capítol 4

Planificació

En aquest capítol parlarem sobre com s'ha planificat el treball i els objectius que s'han anat marcant al llarg del projecte.

En utilitzar una metodologia de desenvolupament en espiral, no es pot fer una planificació precisa de les tasques que s'hauran de dur a terme al llarg del projecte. Així i tot, podem exposar els diferents objectius del projecte i les tasques que s'han necessitat per tal d'assolir-los.

Aquest projecte va començar a principis de Febrer de 2018 i s'ha invertit unes 20 hores setmanals fins a finals de Juliol. Durant el mes d'Agost s'ha documentat el projecte i s'han fet proves i testos.

Els objectius que han anat sorgint d'acord amb els requeriments del projecte, dels estudis realitzats i de les valoracions del tutor i companys han estat els següents:

- Establir comunicació entre l'aplicació mòbil i la polsera. Caldrà preparar les interfícies i estructures de dades necessàries per a la comunicació. A la taula [4.1] podem veure les tasques que han estat necessàries.

Tasca	Hores
Crear pantalla per escanejar dispositius	8
Crear un Servei d'Android que gestioni la connexió Bluetooth	40
Escanejar i filtrar polseres Mi Band 1S	8
Crear pantalla per fer pair amb un dispositiu	4
Descobrir el protocol de la polsera Mi Band 1S	20
Dissenyar un sistema per enviar comandes en forma d'accions	12
Implementar accions de les comandes que pot rebre la polsera	8
Inicialitzar la polsera amb les comandes adequades	8
Testos de vibració, leds, notificacions, etc	4
Total	112

Taula 4.1: Tasques comunicació de l'aplicació amb la polsera.

- Capturar el ritme cardíac des de la polsera i mostrar-lo a l'aplicació. A la taula [4.2] podem veure les tasques que han estat necessàries.

Tasca	Hores
Crear pantalla per mostrar el ritme cardíac	1
Enviar comandes per iniciar la captura del ritme cardíac	8
Rebre notificacions del ritme cardíac i mostrar-lo	12
Total	21

Taula 4.2: Tasques captura ritme cardíac.

- Crear la base de dades i el servidor de crides rest. A la taula [4.3] podem veure les tasques que han estat necessàries.

Tasca	Hores
Instal·lació de la base de dades amb MySql	4
Disseny de la base de dades	4
Disseny del servidor amb Spring	8
Crear crides rest per a l'autenticació	4
Crear crides rest per a la transferència de fitxers	4
Crear crides rest per a la gestió d'usuari	4
Total	28

Taula 4.3: Tasques del servidor i la base de dades.

- Establir comunicació de l'aplicació amb el servidor. S'ha establit una lògica d'accés i registre per a l'usuari. A la taula [4.4] podem veure les tasques que han estat necessàries.

Tasca	Hores
Crear un Servei d'Android que gestioni la connexió	20
Dissenyar la lògica de sessió	2
Crear pantalla de càrrega on s'obté la informació del servidor	8
Crear pantalla de registre	8
Crear pantalla d'accés d'usuari	8
Crear pantalla d'inici amb accessos a altres pantalles	8
Total	52

Taula 4.4: Tasques del servidor i la base de dades.

- Actualitzar la dinàmica d'autenticació de manera que s'utilitzin sessions amb **Tokens**. A la taula [4.4] podem veure les tasques que han estat necessàries.

Tasca	Hores
Investigar sobre les sessions amb tokens	20
Modificar les crides rest per tal que requereixin Tokens	6
Modificar la crida d'accés i de registre per tal que retornin Tokens	6
Afegir una nova crida per refrescar els Tokens de sessió	4
A l'aplicació utilitzar els Tokens com a autenticació	4
Total	40

Taula 4.5: Tasques autenticació per sessions.

- Fer el **login** i iniciar la connexió amb la banda de manera automàtica en iniciar l'aplicació. A la taula [4.6] podem veure les tasques que han estat necessàries.

Tasca	Hores
Investigar com guardar un fitxer de preferències	4
Guardar i utilitzar l'adreça MAC de la banda	2
Guardar i utilitzar els Tokens de sessió	4
Total	10

Taula 4.6: Tasques generació i enviament de fitxers.

- Generar fitxers amb les dades de ritme cardíac capturades per la banda i enviar-los al servidor. A la taula [4.7] podem veure les tasques que han estat necessàries.

Tasca	Hores
Obrir un fitxer en iniciar la captura del ritme cardíac	8
Escriure les dades que es van capturant al fitxer	2
Tancar el fitxer i enviar-lo al servidor	8
Total	28

Taula 4.7: Tasques generació i enviament de fitxers.

- Permetre a l'usuari accedir com a un invitat. Molt útil per tal de fer testos ràpids. A la taula [4.8] podem veure les tasques que han estat necessàries.

Tasca	Hores
Afegir una crida rest per a generar un usuari invitat	2
Afegir l'accés com a usuari invitat a la pantalla d'accés	1
Total	3

Taula 4.8: Tasques per permetre un usuari invitat.

- Mantenir l'aplicació funcionant a tota hora. Les connexions amb la polsera i el servidor s'havien implementat a través de Serveis d'Android per tal de poder dur a terme aquest objectiu. Cal destacar que aquest ha estat un dels objectius més difícils d'assolir i s'han anat fent les diferents tasques d'aquest mentre s'assolien altres objectius. A la taula [4.9] podem veure les tasques que han estat necessàries.

Tasca	Hores
Mantenir els serveis oberts quan l'aplicació s'apaga	10
Comprovar periòdicament que els serveis estiguin funcionant	5
Engegar els serveis quan el telèfon s'obre	2
Evitar el **doze mode** d'Android	15
Total	32

Taula 4.9: Tasques per mantenir l'aplicació funcionant a tota hora.

- Millorar la seguretat de la base de dades. A la taula [4.10] podem veure les tasques que han estat necessàries.

Tasca	Hores
Millora algorisme de Hash per guardar la contrasenya	1
Deidentificació de les dades	2
Total	3

Taula 4.10: Tasques per millorar la seguretat.

- Convertir la pàgina principal en un perfil d'usuari amb la capacitat de modificar les dades. Fins ara les dades d'usuari eren o bé predefinides o bé les que ja hi havia a la base de dades. A la taula [4.11] podem veure les tasques que han estat necessàries.

Tasca	Hores
Implementar un objecte expansible	4
Dissenyar el perfil d'usuari fent ús de l'objecte expansible	2
Afegir boto per connectar-se/accedir a la polsera	1
Total	7

Taula 4.11: Tasques perfil d'usuari.

- Guardar les dades capturades en una base de dades local del mòbil en comptes d'anar-les guardant en un fitxer. A la taula [4.12] podem veure les tasques que han estat necessàries.

Tasca	Hores
Dissenyar i implementar la base de dades amb **Sqlite**	2
Guardar les dades capturades a la base de dades local	1
Obtenir les dades de la base de dades com a un fitxer csv	2
Total	5

Taula 4.12: Tasques implementació **Sqlite**.

- Dissenyar i implementar el perfil de la polsera. La pantalla anterior tan sols servia per capturar el ritme cardíac, en la nova pantalla es pot veure la informació de la polsera i activar/desactivar la captura de dades així com enviar les dades al servidor. Per tal de fer el disseny d'aquesta pantalla s'ha reaprofitat l'objecte expansible creat quan dissenyàvem la pantalla de perfil d'usuari.

- Enviar els fitxers de dades periòdicament. Fent ús d'alarmes i *broadcast receivers* s'ha assolit ràpidament, ja que aquest han estat utilitzats per fer la comprovació que els serveis funcionin de forma periòdica.

Cada cop que s'assolia un objectiu s'han fet testos, avaluat riscos i corregit errors. En total s'han destinat al voltant de 50 hores per fer-ho. És aquí on es pot veure l'augment de temps necessari en el desenvolupament en espiral.

Capítol 5

Marc del treball

En aquest capítol parlarem de diversos conceptes generals relacionats amb el projecte.

Primer parlarem sobre els diversos dispositius vestibles que existeixen i més concretament sobre les polseres de monitoratge. Seguidament sobre aplicacions que permeten monitorar l'usuari.

5.1 Els dispositius vestibles

Els dispositius vestibles o **wereables** són tots aquells dispositius electrònics els quals s'incorporen en alguna part del cós i interactuen amb l'usuari duent a terme una o algunes tasques concretes.

Els primers dispositius vestibles podrien datar de dècada dels anys 60, però no ha estat fins al 2010 que la tecnologia ha avançat prou per fer que aquests es vegin més atractius per als consumidors.

Amb la pujada d'interès per part dels consumidors, ha anat augmentat el nombre d'empreses que s'han interessat en el desenvolupament de dispositius vestibles. Així doncs, any rere any, han anat augmentant el ventall de dispositius que podem trobar al mercat i les funcions que aquests poden dur a terme.

Avui en dia la indústria dels dispositius vestibles representa una de les indústries tecnològiques amb més futur i s'aplica en tot tipus de camps des de l'oci o l'esport (com ara les Google Glass o la FitBit), fins a la medicina o la seguretat (possibilitats que estan essent explorades majoritariament per Universitats).

5.1.1 Polseres intel·ligents

Les polseres intel·ligents són un dels dispositius vestibles més populars que hi ha actualment. Aquestes, ofereixen un ampli ventall de funcionalitats que han fet que a poc a poc, i gràcies al fet que el preu ha anat disminuint, comencin a formar part de la vida diària de multitud de persones.

Algunes persones utilitzen aquesta classe de polseres com a despertador o alarma. D'altres configuren els seus dispositius per tal que els avisin mitjançant leds i vibracions dels missatges que reben de forma personalitzada. També es poden utilitzar per desbloquejar el mòbil si la polsera es troba prou a prop.

A nosaltres ens interessen les funcionalitats que permeten monitorar l'usuari. Algunes de les dades que es poden arribar a monitorar en conjunt amb el nostre dispositiu mòbil són:

- Ritme cardíac
- Monitorització dels Passos
- Monitorització de la son
- Monitorització de l'estres
- Glucosa en sang
- Kilòmetres recorreguts

No és d'estranyar doncs que el context de les polseres intel·ligents hagi evolucionat des d'utilitzar-se per saber el teu ritme cardíac quan fas esport fins a utilitzar-se per monitorar la nostra salut. I, en molts llocs, ja s'han començat a fer servir per controlar pacients mèdics.

5.1.2 Altres dispositius

El món dels dispositius vestibles no es limita només a les polseres. Existeixen tota classe de dispositius vestibles com ara ulleres amb visors i sensors, sabates intel·ligents, etc. En general qualsevol peça de roba o accessori que ens puguem imaginar podria esdevenir en un dispositiu intel·ligent el qual monitores l'usuari o simplement oferís qualsevol altre tipus de funcionalitat.

Alguns dels dispositius existents són:

- Accessoris intel·ligents: polseres, rellotges, arracades, etc
- Ulleres intel·ligents amb pantalles i càmeres
- Sabates intel·ligents que poden mesurar les passes i els quilòmetres fets
- Cascos per a bombers equipats amb GPS i sensors, millorant la seguretat del bomber
- Vestits que canvien de mida, forma i color
- Roba capaç de monitorar les constants vitals d'un nadó

5.2 Aplicacions de monitoratge amb dispositius vestibles

El creixement dels dispositius vestibles no només afecta aquelles companyies que els desenvolupen. Han sorgit moltes petites empreses i desenvolupadors que, aprofitant els dispositius oferits per grans empreses com Google, Amazon o Xiaomi, han començat a desenvolupar aplicacions de tot tipus que utilitzen aquests dispositius.

Algunes d'aquestes aplicacions sincronitzen diferents dispositius de diferents marques. D'altres implementen noves funcionalitats que no tenen les aplicacions oficials. Finalment, també tenim aquelles que tan sols se centren més en una funcionalitat ja existent i la milloren.

En el grup de recerca eXiT ens interessava tenir una aplicació capaç de fer un monitoratge constant al llarg del dia i que centralitzés les dades capturades en un servidor. Malauradament, si bé hem pogut trobar algunes aplicacions que permeten el monitoratge constant del ritme cardíac, aquestes no permeten centralitzar les dades en un servidor o bé són aplicacions massa grans o bé no tenen exactament les funcionalitats que ens interessen.

Així doncs amb aquest projecte busquem solucionar aquest buit oferint una aplicació capaç de monitorar les 24 hores del dia el ritme cardíac amb la Xiaomi Mi Band 1S i centralitzar les dades en un servidor. També, es vol que l'aplicació sigui capaç d'evolucionar en un futur i acceptar diferents dispositius així com capturar tota classe de dades.

5.3 Equip de treball

Les persones que col·laboren en el desenvolupament d'aquest projecte són:

- Els tutors i directors del treball Carles Pous Sabadí i Maria Beatriz Lopez Ibañez
- La membre del grup de recerca eXiT Natalia Mordvaniuk com a guia i consultora
- L'alumne Òscar Raya I Casanova

Capítol 6

Requisits del sistema

En aquest capítol s'exposaran tots els requisits que ha de complir el sistema per tal d'assolir els objectius marcats.

6.1 Requisits no funcionals

Els requisits no funcionals són aquells que indiquen com ha de ser el sistema i no pas que s'ha d'implementar. Aquests són:

- **Seguretat:** El sistema ha de complir amb la normativa de seguretat vigent referent a la protecció i transferència de dades.
- **Fiabilitat:** El sistema s'ha de mantenir funcionant les 24 hores del dia i ésser flexible a possibles fallades reiniciant-se a si mateix en cas d'esser necessari.
- **Connectivitat:** El sistema ha de poder transmetre dades a través de Bluetooth i de Wifi.
- **Escalabilitat:** El sistema s'ha de poder adaptar per tal que el facin servir una gran quantitat de persones.
- **Variabilitat:** El sistema s'ha de poder adaptar a les diferents necessitats dels usuaris. Això comporta que l'usuari sigui capaç d'utilitzar l'aplicació amb fins diferents dels que se li donen en el grup.
- **Adaptabilitat:** El sistema ha d'estar preparat per tal d'adaptar-se a nous dispositius i protocols. També ha de permetre incorporar noves funcionalitats.

6.2 Requisits funcionals

Els requisits funcionals són aquells que indiquen que ha de fer el sistema. El nostre sistema consta de dos subsistemes: aplicació i servidor, amb els seus respectius requeriments. S'han determinat els següents requisits del sistema:

- **Precondicions:** L'usuari disposa d'un mòbil Android 5.0 o superior amb Bluetooth i d'una polsera Xiaomi Mi Band 1S. L'administrador del sistema disposa d'un servidor amb base de dades.
- **Flux Principal:** El flux principal del sistema segueix els següents punts:
  1. L'usuari accedeix a l'aplicació.
  2. L'usuari es posa la polsera i la connecta amb l'aplicació.
  3. L'usuari activa la captura de dades.
  4. L'aplicació envia les dades capturades al servidor de forma periòdica.
- **Postcondicions:** Les dades emmagatzemades estan disponibles per a la seva revisió i tractament.

6.2.1 Requisits del servidor

El servidor tan sols s'utilitza per guardar les dades preses per l'aplicació així que té molt pocs requisits:

- **Autenticació d'usuaris:** El servidor haurà d'implementar les següents funcionalitats que identifiquen i autentiquen a un usuari:
  - Registre d'un usuari
  - Accés a una sessió d'usuari
  - Refrescar la sessió d'un usuari
  - Accés a una sessió d'usuari invitat
- **Administració d'usuaris:** El servidor ha d'implementar les següents funcionalitats relacionades amb l'administració d'usuaris:
  - Modificar el sexe d'un usuari
  - Modificar l'any de naixement d'un usuari
  - Modificar l'altura d'un usuari
  - Modificar el pes d'un usuari
  - Mostrar la informació d'un usuari (id, sexe, naixement, altura i pes)
- **Administració de fitxers:** El servidor ha d'implementar les següents funcionalitats relacionades amb l'administració de fitxers:
  - Rebre un fitxer de dades per part d'un usuari i guardar-lo
  - Mostrar una llista dels fitxers d'un usuari
  - Servir un fitxer d'un usuari

6.2.2 Requisits de l'aplicació

L'aplicació té els següents requeriments funcionals:

- **Registre d'usuaris:** L'aplicació compta amb una pantalla per registrar usuaris en el sistema.
- **Inici de sessió:** L'aplicació compta amb una pantalla per permetre que l'usuari inici de sessió.
- **Usuari invitat:** Permetre que un usuari inici de sessió com a usuari invitat.
- **Tancar sessió:** Un cop iniciada la sessió l'usuari ha de poder tancar-la.
- **Modificar dades d'usuari:** L'usuari pot modificar les seves dades a través de la pantalla corresponent.
- **Escanejar polseres:** L'usuari pot escanejar polseres des de la pantalla corresponent.
- Connectar-se a una polsera un cop escanejada.
- Veure informació de la polsera que tenim connectada.
- Desconnectar-se de la polsera que tenim connectada.
- Capturar el ritme cardíac a través de la polsera connectada.
- **Enviar fitxer:** Enviar fitxers amb les dades de ritme cardíac al servidor manualment i de forma periòdica un cop al dia.

Capítol 7

Estudis i decisions

En aquest capítol parlarem sobre els diferents estudis que s'han dut a terme per tal de realitzar aquest projecte així com les decisions que s'han pres d'acord amb aquests.

7.1 Programari

En aquesta secció s'explica el programari que s'ha utilitzat en el projecte i, en els casos adients, se'n fa una descripció breu del seu funcionament i una comparació amb altres alternatives.

7.1.1 Android

Avui dia, Android té la major quota de mercat respecte el que fa a sistemes operatius. Com podem veure a la taula [7.1], el 2016, Android mostrava un gran avantatge en la quota de mercat respecte del seu competidor principal **iOS**.

Sistema Operatiu	Quota de mercat (%)
Android	81.7
iOS	17.9
Windows	0.3
Altres	0.1

Taula 7.1: Quota de mercat dels SO en els telèfons mòbils el 2016 segons Gartner

Si només tenim en compte els dos sistemes operatius principals, podrem veure a més que el desenvolupament amb Android és molt més barat que amb iOS. A més, hi ha una àmplia diversitat de dispositius Android assequibles mentre que els dispositius que utilitzen iOS són pocs i cars.

IntelliJ Idea ens indica que un **71.3%** dels dispositius Android utilitzen la versió 5.0 o superior, de manera que les aplicacions que es programen per a aquesta versió tenen una quota de mercat considerable. A més, la nostra aplicació requereix **Bluetooth LE** que està implementat a partir de la versió 4.3.

Així doncs, s'ha decidit programar la nostra aplicació amb Android 5.0.

7.1.2 IntelliJ Idea

Actualment els programadors d'Android utilitzen dos IDEs diferents:

- IntelliJ Idea
- Android Studio

Si bé Android Studio és l'IDE oficial per al desenvolupament d'aplicacions d'Android aquest està basat en IntelliJ Idea. Així doncs, l'experiència a l'hora de programar amb aquests dos IDEs és bastant similar.

Com que no tenia experiència prèvia utilitzant Android Studio i a més també havíem de programar un servidor rest, s'ha optat per fer servir IntelliJ Idea que no està tan centrat en el desenvolupament d'aplicacions Android.

7.1.3 MySQL

MySQL és un sistema de gestió de bases de dades relacional. Les diferències internes entre els diferents sistemes de gestió de bases de dades són molt poques. Així doncs, ja que MySQL es considera el més popular de tots ells se l'ha escollit per implementar la base de dades del servidor.

7.1.4 SQLite

SQLite és, igual que MySQL, un sistema de gestió de bases de dades relacional continguda en una llibreria relativament petita i escrita en C. A diferència dels sistemes de gestió de bases de dades client-servidor, SQLite no és un procés independent, sinó que es fa servir des del programa principal formant part d'aquest. Així doncs gràcies a SQLite podem tenir bases de dades en gairebé qualsevol dispositiu i entorn.

En el nostre cas l'hem fet servir per poder tenir una base de dades local que guardi les dades monitorades en el nostre dispositiu Android. S'ha vist necessari guardar aquestes dades en una base de dades per tal de millorar la fiabilitat i persistència de l'aplicació respecte d'anar guardant les dades en un fitxer.

7.1.5 Retrofit 2.0

Retrofit 2.0 és un client REST per a Android i Java. Aquest permet fer tot tipus de peticions web i és una de les llibreries més fàcils i rapides d'aprendre. En la nostra aplicació Android es fa servir per establir una comunicació amb el servidor.

7.1.6 Spring vs Struts

Per tal de programar les crides del servidor podríem fer-ho sense fer servir llibreries, però avui en dia existeixen *frameworks* que fan la major part de la feina i que són altament configurables. Els dos *frameworks* més popular per desenvolupar servidors web en Java són Spring i Struts.

Essencialment, Spring i Struts segueixen la mateixa metodologia però els dos es configuren diferent. Struts involucra en la seva configuració fitxer XML mentre que Spring només utilitza codi Java. Entre els dos ens hem decantat per Spring, ja que és el més popular de tots i m'ha semblat que la seva configuració és més clara i senzilla. D'altra banda, comptava amb una companya dins del grup de recerca amb experiència prèvia amb aquest *framework*.

Spring s'organitza implementant controladors per a diferents grups de crides. A més aquests controladors poden tenir accés a diferents taules de la base de dades o bé a fitxers del sistema.

7.1.7 JSON Web Token

Al llarg del desenvolupament del servidor va sorgir la necessitat d'integrar un sistema de sessions d'usuari. Amb aquesta idea en ment es va començar a explorar com implementar un sistema que guardes les sessions obertes en el servidor. Aquesta idea però no m'agradava, ja que els serveis *restfull* se suposa que no guarden cap mena d'estat. Així doncs vaig explorar i explorar pel web fins que vaig trobar una alternativa: les sessions que es guarden en la part del client en comptes del servidor.

Aquesta classe de sessions en comptes d'emmagatzemar-se en el servidor, com és comú en moltes aplicacions, s'emmagatzemen en la part del client i es fan servir per autenticar la seva identitat en el servidor.

Però com funcionen aquestes sessions? Com poden assegurar que aquella persona és la que diu que és? La resposta cau en la criptografia. El servidor posa a disposició del client una crida per obtenir un Token. Aquest Token només pot ser validat pel servidor el qual l'ha creat.

En la nostra aplicació hem utilitzat JSON web Tokens amb la llibreria jjwt. Els JWT [13.4] funcionen de la següent manera:

1. El client demana al servidor un Token de sessió proporcionant les seves dades d'autenticació.

2. El servidor comprova la validesa de les dades i genera un Token. Aquest és la concatenació de dos objectes Json expressats en base 64 i un Hash del segon objecte. El primer objecte especifica el tipus de Token, el segon especifica un identificador per l'usuari i una data de caducitat i el Hash és generat amb una clau que només coneix el servidor.

3. L'usuari utilitza el Token rebut per part del servidor com a autenticació.

4. El servidor comprova que aquell Token que ha rebut per l'usuari hagi estat creat per ell. Ell és l'únic capaç de fer aquesta comprovació, ja que és necessària la paraula clau que només coneix el servidor.

7.2 Maquinari

En aquesta secció s'expliquen els criteris que s'ha tingut en compte a l'hora de triar el maquinari utilitzat en el desenvolupament del projecte.

7.2.1 Polsera

Els factors determinants alhora d'escollir una polsera han estat els següents:

- Que pugui capturar el ritme cardíac
- Precisió de les dades recol·lectades
- Cost del dispositiu
- Durada de la bateria
- Que el protocol permeti extreure les dades de la polsera

Entre aquelles polseres que són capaces de calcular el ritme cardíac i que permeten extreu-ren les dades s'han seleccionat aquelles més populars i s'ha estudiat la seva viabilitat en el nostre projecte per tal d'escollir-ne una.

7.2.1.1 Sensor de ritme cardíac

La tecnologia utilitzada per mesurar el ritme cardíac és la mateixa per a totes aquestes polseres. Aquesta tecnologia és anomenada **Fotopletismografia**.

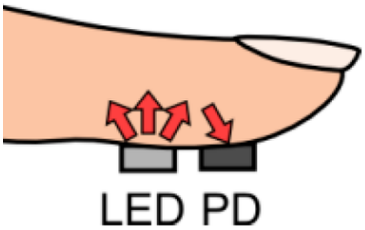

Figura 7.1: Esquema sensor cardíac

La Fotopletismografia utilitza leds i fotosensors per tal d'identificar les pulsacions. Com que la sang és vermella aquesta absorbeix tots els colors excepte el vermell. Així doncs en posar un led sobre la pell es pot identificar les pulsacions notant els canvis de la quantitat de llum que ha estat reflectida, tal i com es mostra a la figura [7.1], o que ha aconseguit arribar a l'altre costat.

La majoria de sensors de ritme cardíac utilitzen leds de color verd, ja que sembla que són els més precisos, però es podrien utilitzar leds de qualsevol altre color.

7.2.1.2 Comparativa i elecció

Les polseres que s'han tingut en consideració han estat les que es mostren a la taula [7.2].

Polsera	Preu	Autonomia
Fitbit Charge HR	70€ - 150€	5 dies
Xiaomi Mi Band 1S	15€ - 20€	20 dies
Xiaomi Mi Band 2	20€ - 25€	10 dies

Taula 7.2: Taula comparativa de diferents polseres d'activitat.

Vista la comparativa de la taula [7.2] hem decidit desenvolupar la nostra aplicació per la polsera Xiaomi Mi Band 1S. Si bé la Fitbit Charge HR és aclamada com la millor entre les 3 polseres i la Mi Band 2 es considera també millor que la seva predecessora, s'ha determinat que allò que ofereixen de més no era el que buscàvem i en canvi, ja que incorporen més funcionalitats i una pantalla la seva bateria dura menys.

Val a dir que els valors d'autonomia de la taula són els indicats per a un ús normal de les polseres. Des del grup de recerca eXiT volem que aquestes estiguin 24/7 prenen dades així que l'autonomia de la polsera es redueix dràsticament de 20 dies a uns 2 dies, tal i com és demostra a la secció 10.3.

7.2.2 Mòbil

Per desenvolupar aquest projecte qualsevol mòbil amb Android 5.0 o superior i que tingui Bluetooth és acceptable. Així doncs, he fet servir el meu mòbil personal un Xiaomi Redmi Note 4X amb Android 7.0.

7.2.3 Servidor

Per tal de desenvolupar l'aplicació del servidor era necessari poder obrir ports del router. La xarxa de la Universitat de Girona és molt restrictiva així que durant el desenvolupament s'ha optat per obrir el servidor en un ordinador personal de casa meva. En la implantació final del sistema el servidor s'haurà d'obrir dins la xarxa de la Universitat.

Capítol 8

Anàlisis i disseny del sistema

El nostre sistema, tal com es mostra a la figura [8.1], està conformat per 3 nodes: una polsera, un mòbil i un servidor. La polsera i el mòbil és comuniquen mitjançant una connexió Bluetooth mentre que el servidor i el mòbil ho fan a través d'Internet amb connexions Wifi i/o Ethernet.

Figura 8.1: Diagrama de comunicacions en el sistema

8.1 Anàlisis del sistema

En aquesta secció procedirem a fer una anàlisi del sistema, tant de l'aplicació mòbil com de l'aplicació del servidor.

8.1.1 Diagrames de casos d'ús

Els diagrames de casos d'ús especifiquen la interacció i el comportament de l'aplicació. S'han fet dos diagrames de casos d'ús, un per al servidor [8.2] i un per a l'aplicació mòbil [8.3].

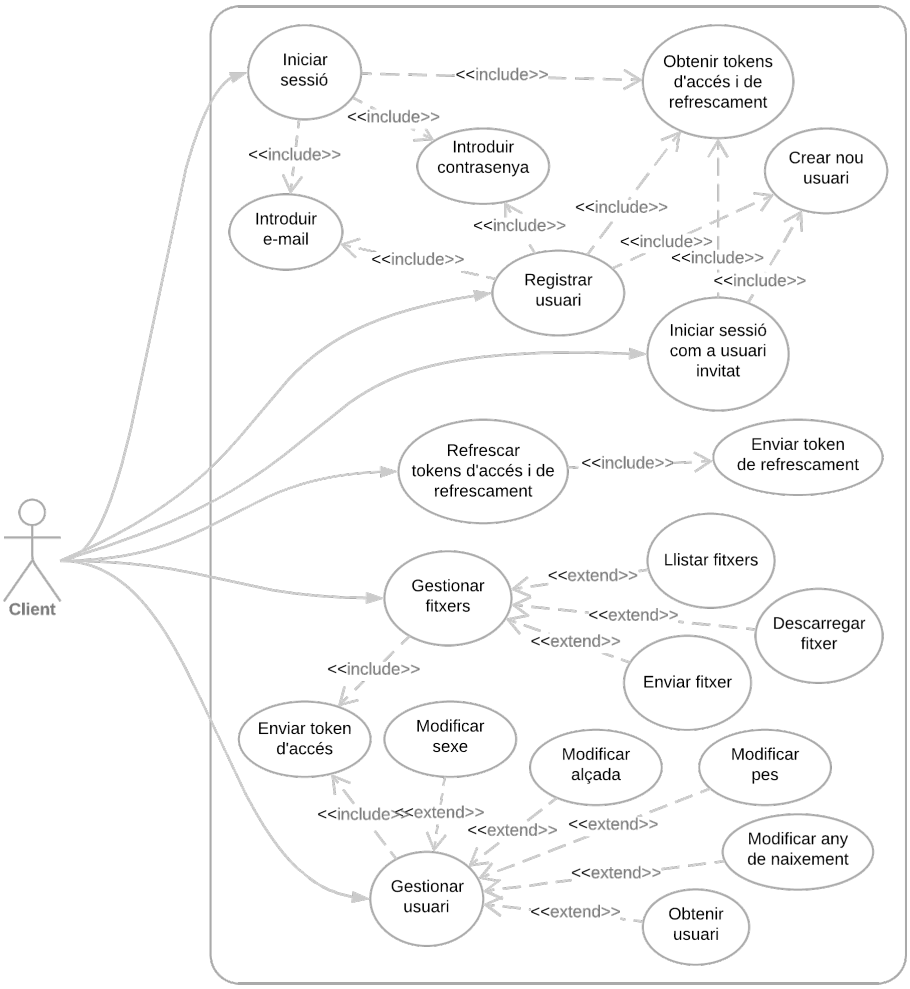

Figura 8.2: Diagrama de casos d'ús de l'aplicació del servidor

Qualsevol cas d'ús en el servidor pot ser utilitzat per qualsevol agent extern,

com ara l'aplicació mòbil o un navegador, sempre que aquest conegui les dades d'autenticació en els casos necessaris.

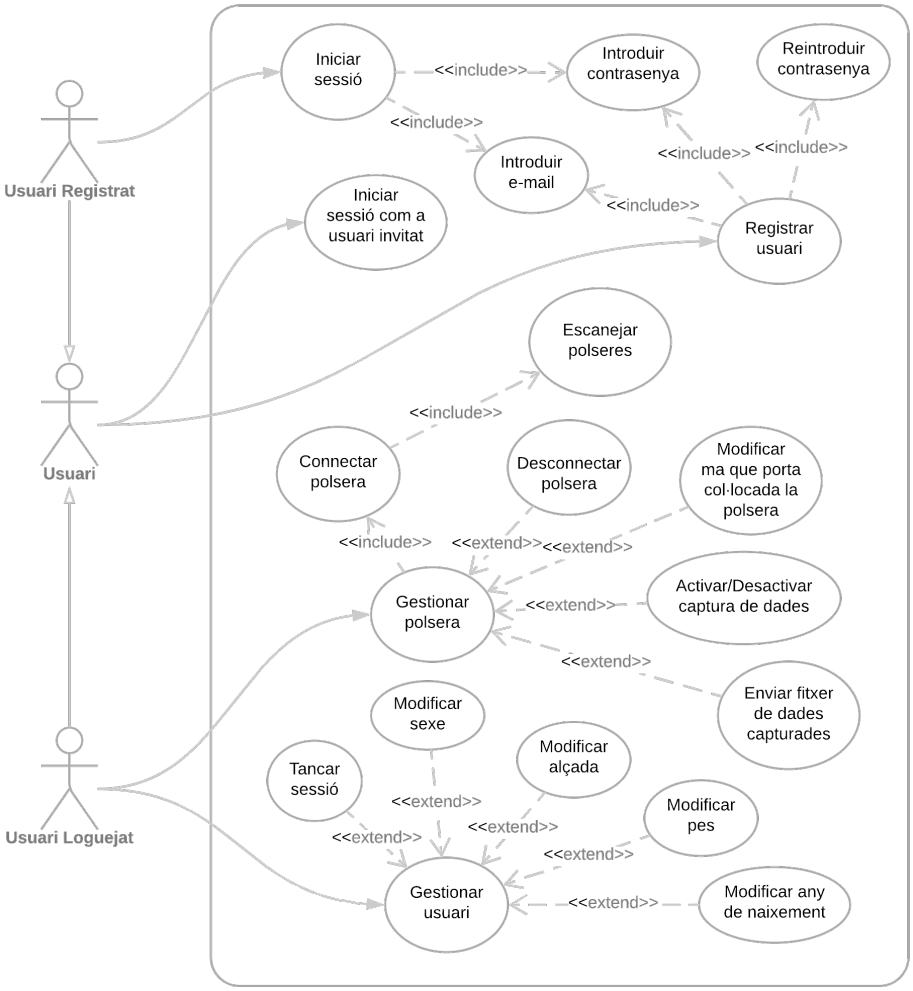

Figura 8.3: Diagrama de casos d'ús de l'aplicació mòbil

Com es pot observar en els dos diagrames, podem trobar casos d'ús similars entre l'aplicació i el servidor, ja que aquests mostren les interaccions entre aplicació i servidor.

8.1.2 Fitxes de cas d'ús

Les fitxes de cas d'ús especifiquen en més detall els diferents casos d'ús dels diagrames de l'apartat anterior.

8.1.2.1 Fitxes de cas d'ús del servidor

Cas d'ús	**Iniciar sessió**
Descripció	Inicia la sessió d'un usuari existent
Actor	Client
Precondició	Existeix un usuari amb l'email i la contrasenya indicats
Flux principal	1. El client fa una petició post amb email i contrasenya 2. Generar hash amb email i contrasenya 3. Obtenir usuari amb usuari.hash == hash 4. Obtenir id d'usuari 5. Generar token d'accés amb id 6. Generar token de refrescament amb id 7. Enviar tokens al client
Flux alternatiu	3.1 Si usuari no existeix enviar missatge d'error **Wrong password or username!** al client
Postcondició	S'han retornat tokens de sessió

Cas d'ús	**Registrar usuari**
Descripció	Registra un nou usuari
Actor	Client
Precondició	Format d'email i contrasenya vàlids i no repetits
Flux principal	1. El client fa una petició post amb email i contrasenya 2. Generar hash amb email i contrasenya 3. Comprovar si existeix usuari amb usuari.hash == hash 4. Crear un nou usuari amb email i contrasenya 5. Obtenir id d'usuari 6. Generar token d'accés amb id 7. Generar token de refrescament amb id 8. Enviar tokens al client
Flux alternatiu	3.1 Si usuari existeix enviar missatge d'error **Account already exist!** al client
Postcondició	S'han retornat tokens de sessió i s'ha creat un nou usuari

Cas d'ús	**Iniciar sessió com a usuari invitat**
Descripció	Inicia la sessió com a un usuari invitat
Actor	Client
Precondició	-
Flux principal	1. El client fa una petició get 2. Crear un nou usuari 3. Obtenir id d'usuari 4. Generar token d'accés amb id 5. Generar token de refrescament amb id 6. Enviar tokens al client
Flux alternatiu	-
Postcondició	S'han retornat tokens de sessió i s'ha creat un nou usuari

Cas d'ús	**Refrescar sessió**
Descripció	Refresca els tokens d'accés i de refrescament
Actor	Client
Precondició	Token de refrescament valid
Flux principal	1. El client fa una petició post amb token de tipus refrescament 2. Validar token 3. Obtenir id del token 4. Generar token d'accés amb id 5. Generar token de refrescament amb id 6. Enviar nous tokens al client
Flux alternatiu	2.1 Si token és invalid enviar missatge d'error al client
Postcondició	S'han retornat nous tokens de sessió

Cas d'ús	**Obtenir usuari**
Descripció	Obtenir les dades d'un usuari
Actor	Client
Precondició	Token d'accés valid
Flux principal	1. El client fa una petició get amb token d'accés 2. Validar token 3. Obtenir id del token 4. Obtenir usuari amb usuari.id == id 5. Enviar usuari al client
Flux alternatiu	2.1 Si token és invalid enviar missatge d'error al client 4.1 Si usuari no existeix enviar missatge d'error **User doesn't exist!** al client
Postcondició	S'ha retornat la informació de l'usuari

Cas d'ús	**Modificar usuari**
Descripció	Modificar les dades d'un usuari
Actor	Client
Precondició	Token d'accés valid
Flux principal	1. El client fa una petició put amb token d'accés i nouUsuari 2. Validar token 3. Obtenir id del token 4. Obtenir usuari amb usuari.id == id 5. Modificar usuari amb les dades de nouUsuari 6. Enviar missatge de confirmació al client
Flux alternatiu	2.1 Si token és invalid enviar missatge d'error al client 4.1 Si usuari no existeix enviar missatge d'error **User doesn't exist!** al client
Postcondició	S'han modificat les dades d'un usuari

Cas d'ús	**Llistar fitxers**
Descripció	Llistar els fitxers d'un usuari determinat
Actor	Client
Precondició	Token d'accés valid
Flux principal	1. El client fa una petició get amb token d'accés 2. Validar token 3. Obtenir id del token 4. Seleccionar llista de fitxers del directori amb nom == id 5. Enviar llista de fitxers al client
Flux alternatiu	2.1 Si token és invalid enviar missatge d'error al client
Postcondició	S'han llistat els fitxers d'un usuari

Cas d'ús	**Servir fitxer**
Descripció	Serveix un fitxer al client
Actor	Client
Precondició	Token d'accés valid i fitxer existeix
Flux principal	1. El client fa una petició get amb token d'accés a fitxer i paràmetre nom de fitxer 2. Validar token 3. Obtenir id del token 4. Obtenir nom del token 5. Obtenir fitxer amb nom de fitxer en el directori amb nom == id 6. Enviar fitxer al client
Flux alternatiu	2.1 Si token és invalid enviar missatge d'error al client 4.1 Si nom del token és diferent a nom del fitxer enviar missatge d'error al client
Postcondició	S'ha servit un fitxer

Cas d'ús	**Rebre fitxer**
Descripció	Rep un fitxer per part del client
Actor	Client
Precondició	Token d'accés valid i fitxer no null
Flux principal	1. El client fa una petició post amb token d'accés i un fitxer 2. Validar token 3. Obtenir id del token 4. Guardar fitxar en el directori amb nom == id 5. Enviar missatge de confirmació al client
Flux alternatiu	2.1 Si token és invalid enviar missatge d'error al client
Postcondició	S'ha pujat un fitxer

8.1.2.2 Fitxes de cas d'ús de l'aplicació

Cas d'ús	**Iniciar sessió**
Descripció	Iniciar la sessió amb email i contrasenya
Actor	Usuari Registrat
Precondició	Usuari existeix, connexió a Internet habilitada
Flux principal	1. L'usuari introdueix l'email i la contrasenya 2. Validar format de l'email 3. Validar format de la contrasenya 4. Enviar petició d'inici de sessió al servidor amb email i contrasenya 5. Esperar resposta per part del servidor 6. Guardar els tokens de sessió retornats amb la resposta
Flux alternatiu	2. Si el format no és valid indicar-ho a l'usuari 3. Si el format no és valid indicar-ho a l'usuari 5. Si la resposta és negativa indicar-ho a l'usuari
Postcondició	L'usuari ha iniciat sessió

Cas d'ús	**Registrar usuari**
Descripció	Registrar un usuari amb email i contrasenya
Actor	Usuari
Precondició	Usuari no existeix, connexió a Internet habilitada
Flux principal	1. L'usuari introdueix l'email, la contrasenya i un altre cop la contrasenya 2. Validar format de l'email 3. Validar format de la contrasenya 4. Comprovar coincidència de les contrasenyes 5. Enviar petició de registre d'usuari al servidor amb email i contrasenya 6. Esperar resposta per part del servidor 7. Guardar els tokens de sessió retornats amb la resposta
Flux alternatiu	2. Si el format no és valid indicar-ho a l'usuari 3. Si el format no és valid indicar-ho a l'usuari 4. Si les contrasenyes no coincideixen indicar-ho a l'usuari 6. Si la resposta és negativa indicar-ho a l'usuari
Postcondició	L'usuari ha iniciat sessió desprès de registrar-se

Cas d'ús	**Iniciar sessió com a invitat**
Descripció	Iniciar la sessió com a un usuari invitat
Actor	Usuari
Precondició	Connexió a Internet habilitada
Flux principal	1. L'usuari prem el boto d'inici de sessió com a invitat 2. Enviar petició d'inici de sessió com a invitat al servidor 3. Esperar resposta per part del servidor 4. Guardar els tokens de sessió retornats amb la resposta
Flux alternatiu	6. Si la resposta és negativa indicar-ho a l'usuari
Postcondició	L'usuari ha iniciat sessió com a invitat

Cas d'ús	**Tancar sessió**
Descripció	Tancar la sessió d'usuari
Actor	Usuari Loguejat
Precondició	L'usuari ha iniciat sessió
Flux principal	1. L'usuari obre l'aplicació 2. L'usuari prem el boto de tancar sessió 3. S'eliminen els tokens de sessió 4. Es redirigeix a l'usuari a l'activitat d'inici de sessió
Flux alternatiu	-
Postcondició	S'ha tancat la sessió

Cas d'ús	**Modificar usuari**
Descripció	Modificar les dades d'usuari
Actor	Usuari Loguejat
Precondició	L'usuari ha iniciat sessió, connexió a Internet habilitada
Flux principal	1. L'usuari obre l'aplicació 2. L'usuari prem la dada que vol modificar en el perfil 3. L'usuari selecciona un nou valor per a la dada premuda 4. Es fa una petició al servidor per modificar la dada 5. Es rep confirmació per part del servidor
Flux alternatiu	-
Postcondició	S'han modificat dades d'usuari

Cas d'ús	**Connectar polsera**
Descripció	Connectar una polsera a traves de Bluetooth
Actor	Usuari Loguejat
Precondició	L'usuari ha iniciat sessió
Flux principal	1. L'usuari obre l'aplicació 2. L'usuari prem el boto per escanejar polseres 3. Es mostra una llista amb llista amb les polseres escanejades 4. L'usuari escull la polsera amb la que es vol connectar 5. La polsera vibra 6. L'usuari prem la polsera per autoritzar la connexió 7. L'usuari és redirigit al perfil de la polsera
Flux alternatiu	-
Postcondició	S'ha connectat una polsera

Cas d'ús	**Capturar dades**
Descripció	S'inicia la captura del ritme cardíac
Actor	Usuari Loguejat
Precondició	L'usuari ha iniciat sessió i ha connectat una polsera
Flux principal	1. L'usuari obre l'aplicació 2. L'usuari prem el boto per accedir al perfil de la polsera 3. L'usuari prem el boto per activar la captura de dades 4. S'envia una comanda a la polsera per activar la captura del ritme cardíac 5. Cada 10 segons l'aplicació envia una comanda a la polsera per reactivar la captura de dades 6. La polsera envia el ritme cardíac a l'aplicació constantment 7. L'aplicació mostra el ritme cardíac actual en el perfil de la polsera
Flux alternatiu	-
Postcondició	S'esta capturant el ritme cardíac

Cas d'ús	**Enviar dades**
Descripció	S'envien les dades capturades al servidor
Actor	Usuari Loguejat
Precondició	L'usuari ha iniciat sessió, ha connectat una polsera i disposa de connexió a Internet
Flux principal	1. L'usuari obre l'aplicació 2. L'usuari prem el boto per accedir al perfil de la polsera 3. L'usuari prem el boto per enviar les dades capturades 4. És guarden les dades capturades a la base de dades local en un fitxer 5. S'envia un crida post al servidor amb el fitxer i un token d'accés 6. És rep confirmació per part del servidor
Flux alternatiu	6.1 Si la resposta és negativa, se li indica a l'usuari
Postcondició	S'esta capturant el ritme cardíac

Cas d'ús	**Desconnectar polsera**
Descripció	Es desconnecta una polsera
Actor	Usuari Loguejat
Precondició	L'usuari ha iniciat sessió i ha connectat una polsera
Flux principal	1. L'usuari obre l'aplicació 2. L'usuari prem el boto per accedir al perfil de la polsera 3. L'usuari prem el boto desconnectar la polsera 4. Es tanca la connexió amb la polsera 5. S'eliminen les dades relacionades amb la polsera 6. Es redirigeix a l'usuari al seu perfil
Flux alternatiu	-
Postcondició	S'esta capturant el ritme cardíac

8.1.3 Diagrames de classes

En aquest apartat mostrarem els diagrames de classes tant de l'aplicació com del servidor. Tant l'aplicació com el servidor estan programades sobre Frameworks complexos (Android i Spring) per tant s'ha decidit simplificar els diagrames.

8.1.3.1 Diagrama de classes del servidor

L'aplicació del servidor està conformada per controladors que són carregats dinàmicament. Primer de tot mostrarem però els diagrames de classes de dos grups d'utilitats que s'han creat per tal d'ajudar al servidor. Tenim les utilitats d'autenticació [8.4] i les utilitats d'emmagatzematge [8.5].

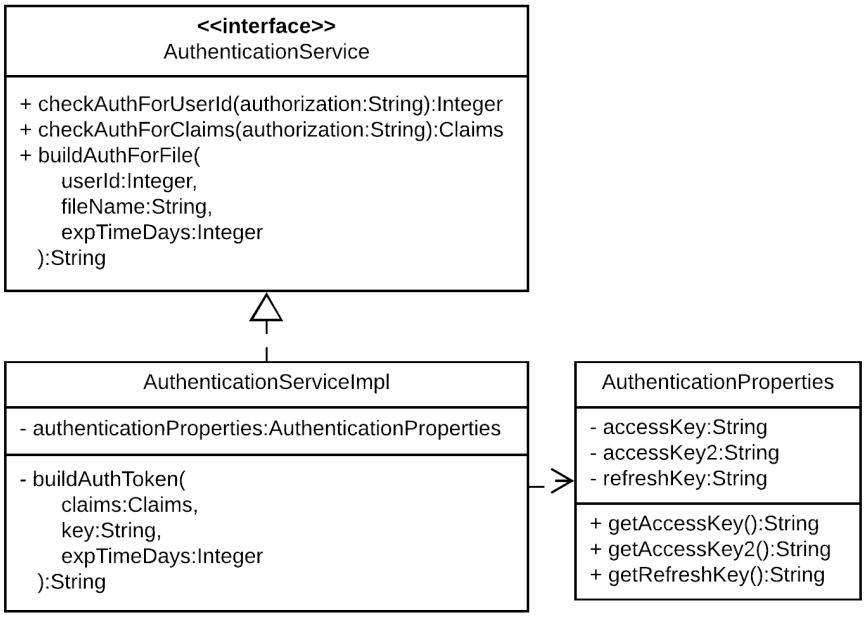

Figura 8.4: Diagrama de classes de les utilitats d'autenticació

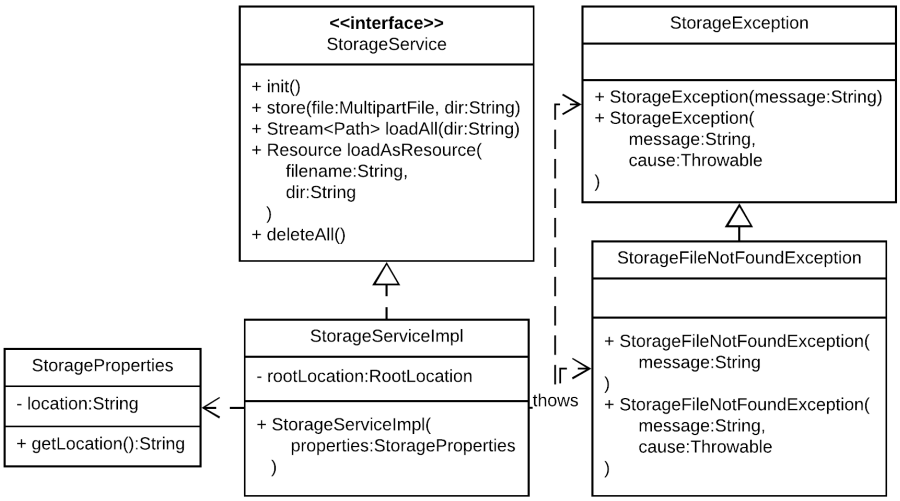

Figura 8.5: Diagrama de classes de les utilitats d'emmagatzematge

Seguidament a la figura 8.6 podem observar les classes dels controladors que han conformat el servidor. S'han programat un total de 4 controladors, 3 d'ells hereten del controlador principal el qual gestiona excepcions.

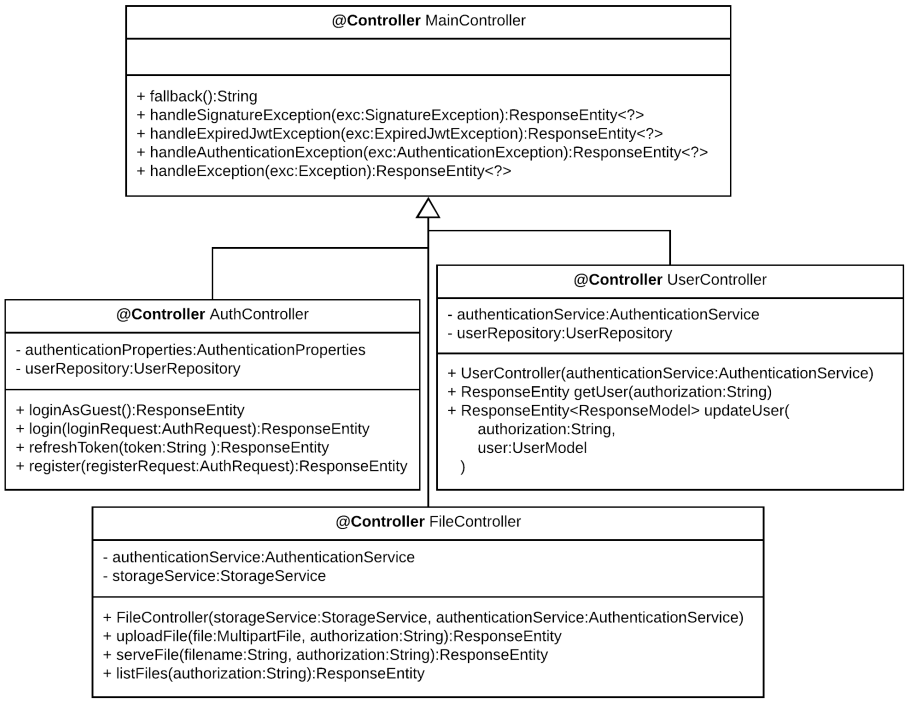

Figura 8.6: Diagrama de classes dels controladors del servidor

Com podem observar els controladors que tenim són:

- MainController - Gestiona excepcions i fallades
- AuthController - Gestiona l'autenticació d'usuaris
- FileController - Gestiona les crides relacionades amb fitxers
- UserController - Gestiona les crides relacionades amb els usuaris

Els controladors utilitzen les utilitats de les figures 8.4 i 8.5. També utilitzen el repositori d'usuaris i els models que veiem a la figura 8.7 per tal d'accedir a la base de dades i enviar i rebre informació. El repositori d'usuaris també representa, juntament amb la classe entitat Usuari, l'única taula existent en la base de dades. Els fitxers no queden indexats en la base de dades i s'accedeix a ells gràcies a les utilitats d'emmagatzematge.

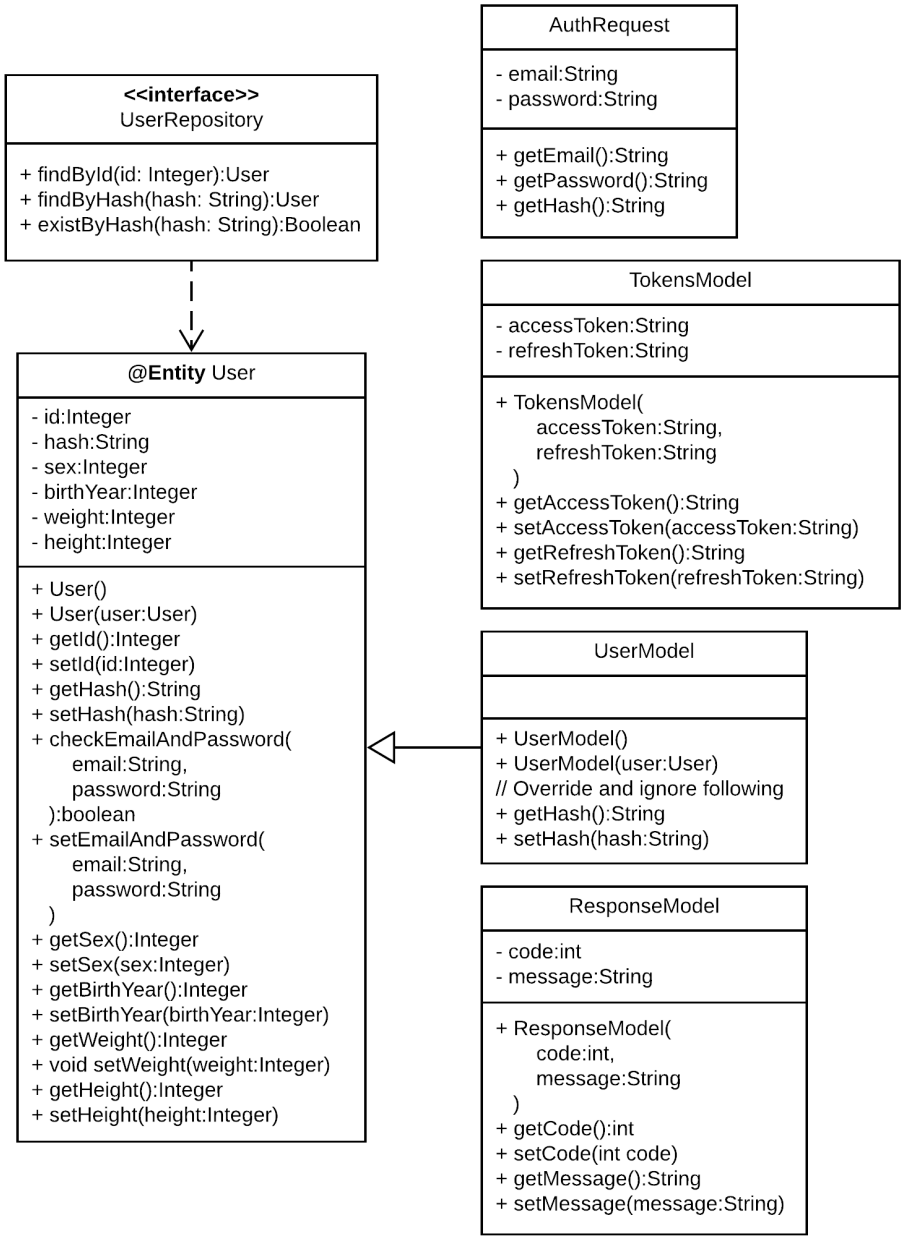

Figura 8.7: Diagrama de classes dels models i els repositoris

8.1.3.2 Diagrama de classes de l'aplicació

Per tal de simplificar el diagrama de classes s'han separat les classes en diferents grups.

Per tal d'enviar i rebre informació amb el servidor s'han implementat els Models de la figura 8.8.

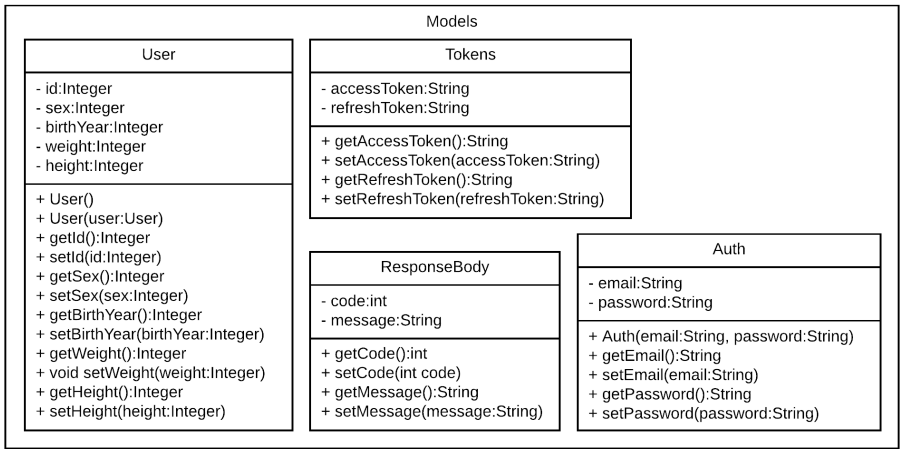

Figura 8.8: Diagrama de classes dels models

També s'han implementat algunes utilitats bàsiques que són utilitzades per diverses activitats i serveis. Les diferents utilitats és poden observar a la figura 8.9.

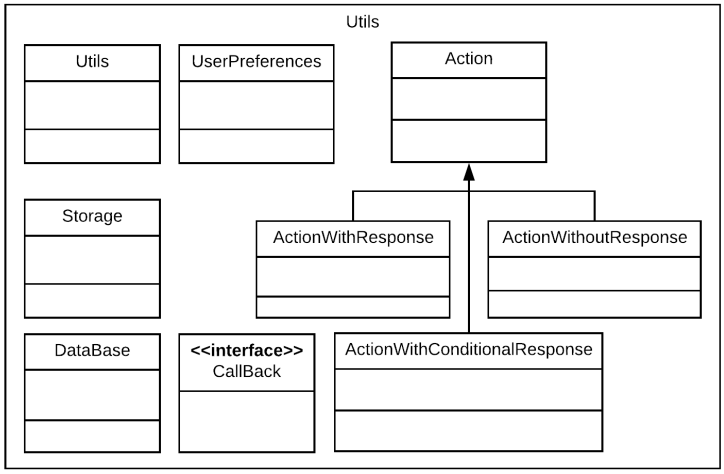

Figura 8.9: Diagrama de classes de les utilitats

La polsera que hem fet servir té 3 serveis Bluetooth, així doncs s'han definit aquests 3 serveis en 3 classes diferents. També, tal com es mostra a la figura 8.10, s'han implementat diverses utilitats per tractar les dades que s'envien i es reben de la polsera.

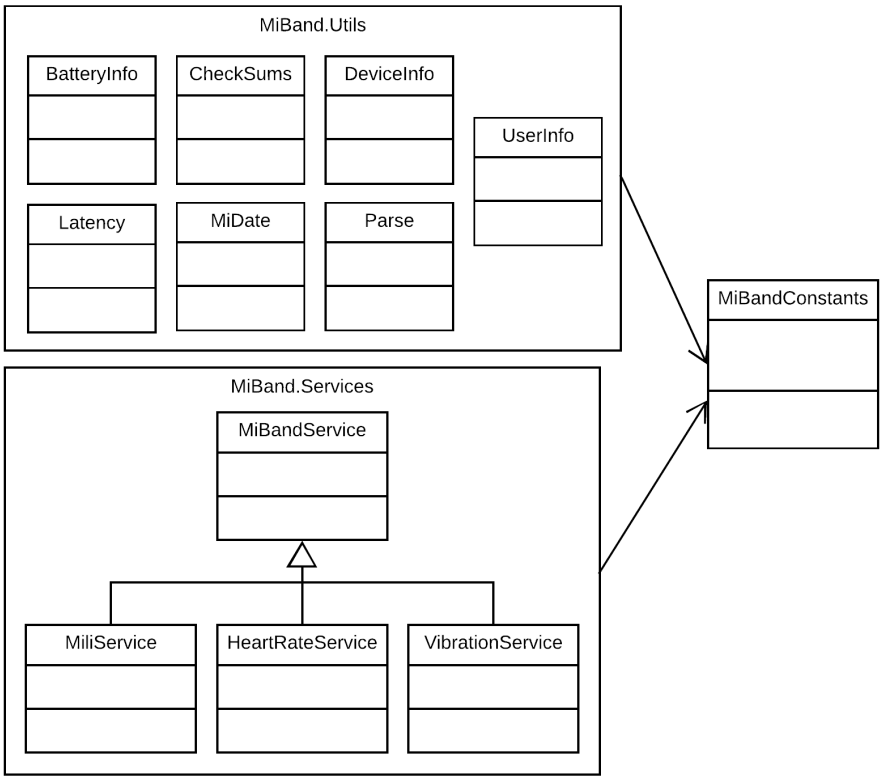

Figura 8.10: Diagrama de classes relacionades amb la polsera Mi Band

Ens ajudem a més d'una classe que hem anomenat **MiBandConstants** per guardar-hi el protocol de la polsera.

Seguidament, a la figura 8.11, podem observar les diferents activitats que conformen la nostra aplicació. Comptem amb un total de 7 activitats més una, que serveix com a base per a aquelles activitats que requereixen una connexió Bluetooth de forma immediata.

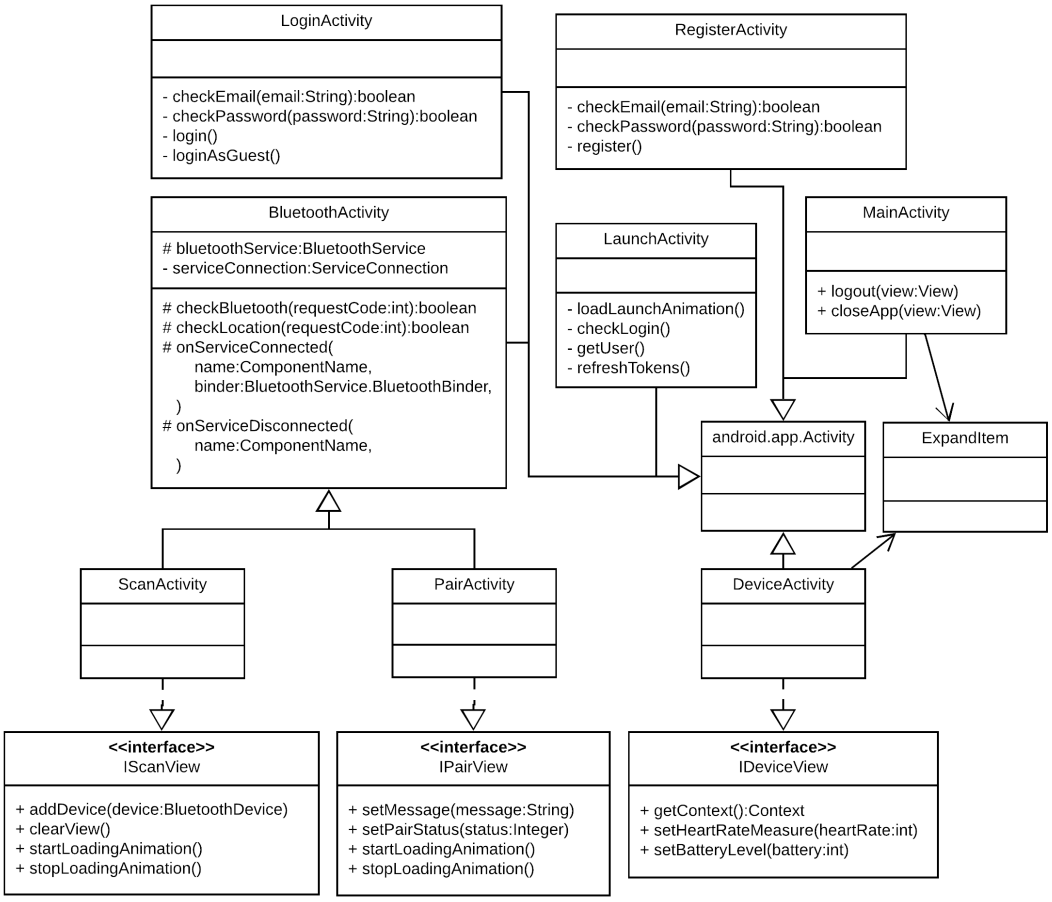

Figura 8.11: Diagrama de classes de les activitats d'Android

També podem veure que algunes activitats implementen interfícies. Aquestes interfícies estan relacionades amb les que implementa el **BluetoothService** i serveixen per establir una comunicació entre el servei i les activitats.

Finalment dues activitats, **DeviceActivity** i **MainActivity**, utilitzen un element que s'expandeix per tal de mostrar informació més detallada o permetre modificar alguna dada.

La majoria de la feina en l'aplicació recau sobre els dos serveis que s'ha implementat.

D'una banda, tenim el servei de crides rest de la figura 8.12 que compta amb un **BroadcastReceiver** que l'inicialitza quan s'ha d'enviar un fitxer i amb 3 APIs de crides rest.

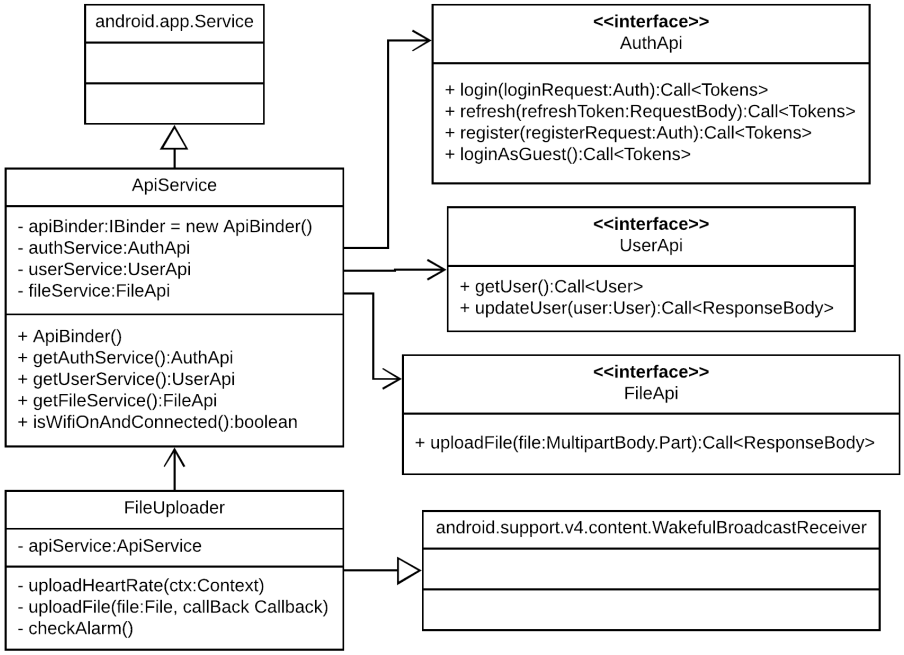

Figura 8.12: Diagrama de classes del servei de crides api rest

D'altra banda, a la figura 8.13 podem observar el servei de Bluetooth, el qual també compta amb un **BroadcastReceiver** que l'inicialitza. En aquest cas es busca mantenir el servei de Bluetooth sempre actiu mentre s'estiguin capturant dades. Així doncs el **BroadcastReceiver** s'assegura de reiniciar el servei sempre que calgui.

El servei de Bluetooth fa servir la classe **ConnectionManager** per tal de gestionar la connexió amb el dispositiu. Aquesta classe conte a més una cua d'accions que serveix per fer comandes de forma ordenada i sense superposicions.

Per a cada dispositiu nou que incloguem en l'aplicació s'haurà d'implementar una classe especifica que hereti de la classe **ConnectionManager**.

Cal destacar també la presència de les interfícies que ajuden al servei a comunicar-se amb les activitats i a organitzar els diferents estats d'una connexió.

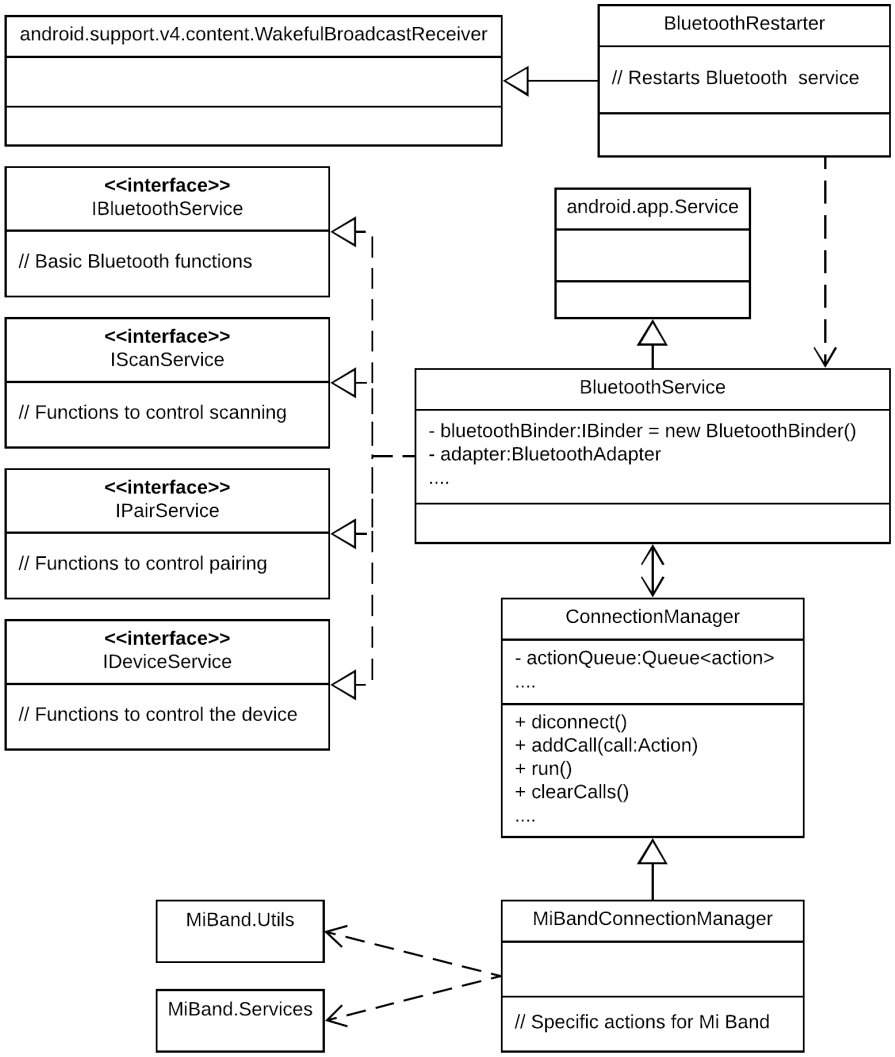

Figura 8.13: Diagrama de classes del servei Bluetooth

8.2 Disseny del sistema

El nostre sistema s'ha dividit en dues aplicacions:

- **Aplicació del servidor**
  Segueix el disseny establert pel framework Spring. És a dir, utilitza controladors per tal d'agrupar crides rest.

- **Aplicació per a mòbil**
  És una aplicació Android en la qual s'han fet servir serveis. En el següent apartat explicarem la seva interfície d'usuari.

8.2.1 Interfície d'usuari

La interficié d'usuari esta conformada per un total de 7 pantalles:

LAUNCH

Aquesta és la pantalla que s'inicia quan s'obre l'aplicació. La seva funció és la de discernir si l'usuari ha iniciat sessió i redirigir a l'usuari a la pantalla d'inici de sessió o a la pantalla principal.

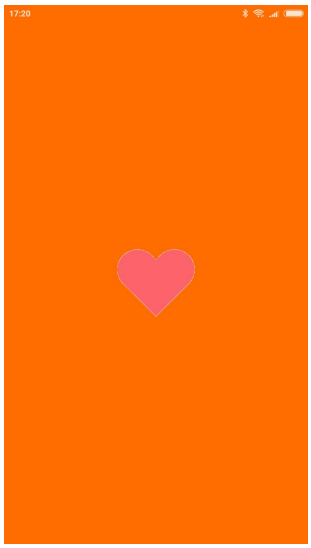

Figura 8.14: Pantalla d'inicialització

LOGIN

La pantalla d'inici de sessió permet a un usuari, enregistrat prèviament, iniciar sessió fent servir el seu e-mail i la seva contrasenya.

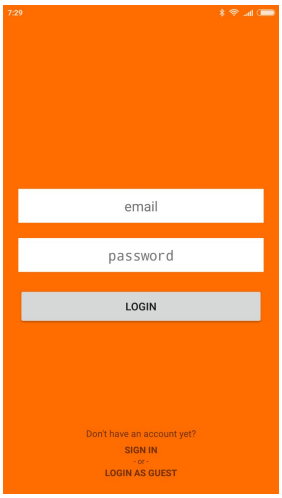

Figura 8.15: Pantalla d'inici de sessió

REGISTER

Aquesta pantalla permet registrar un usuari amb e-mail i contrasenya.

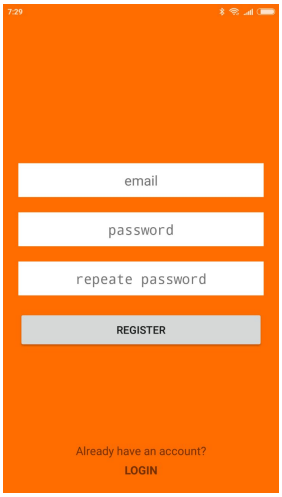

Figura 8.16: Pantalla de registre

MAIN

A la pantalla principal podem observar i modificar les dades de l'usuari així com tancar la sessió. També podem accedir al perfil de la polsera o connectar-ne una.

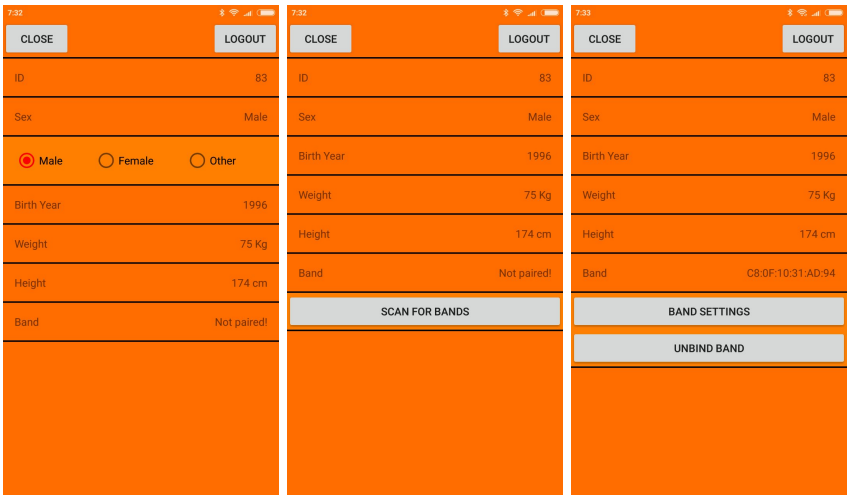

Figura 8.17: Pantalla principal

SCAN

En accedir aquesta pantalla es comencen a mostrar les polseres que es van escanejant i se'n pot seleccionar una per tal d'establir-hi una connexió.

Figura 8.18: Pantalla d'escaneig

PAIR

La pantalla d'emparellament redirigeix a l'usuari a la pantalla del dispositiu un cop s'ha establert la connexió.

Figura 8.19: Pantalla d'emparellament

DEVICE

La pantalla del dispositiu mostra informació bàsica sobre aquest i permet activar i desactivar la captura de dades així com enviar les mesures al servidor.

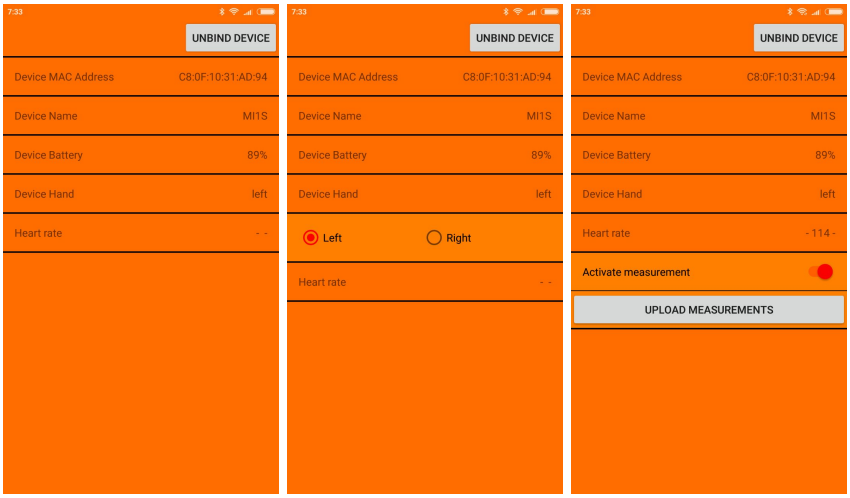

Figura 8.20: Pantalla del dispositiu

8.2.2 Diagrama d'activitats de l'aplicació

A la figura 8.21 és mostra un diagrama que descriu el flux que segueixen les activitats de l'aplicació.

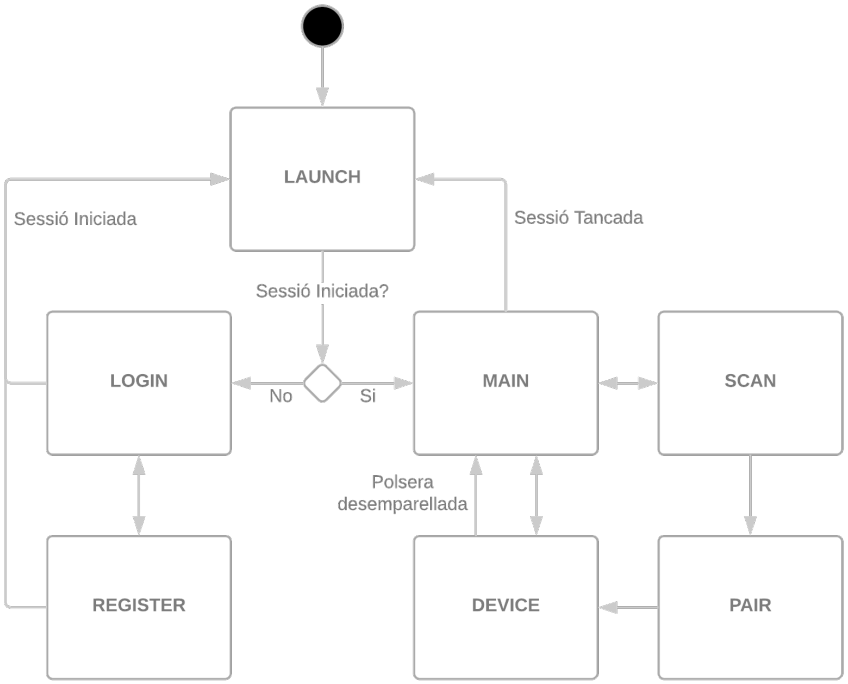

Figura 8.21: Pantalla del dispositiu

Capítol 9

Implementació i proves

9.1 Implementació del servidor

Pots descarregar el codi font del servidor en el següent repositori: https://github.com/arracso/FitServer.

L'aplicació del servidor està programada mitjançant el framework Spring. Com a tal, té una estructura molt senzilla en la qual només cal definir els següents elements:

- Entitats
- Repositoris
- Controladors

Amb les **Entitats** definim i creem les taules de la base de dades. En la nostre aplicació tant sols contem amb una sola taula: **User**, amb la informació bàsica de l'usuari. L'estructura d'una entitat és la següent:

```
import javax.persistence.*;

@Entity
public class BasicEntity {
    // Attributes...
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    private Integer id;

    @Basic(optional = true)
    private String name;

    // Constructors...
    // Public methods...
}
```

Com podem observar, tan sols cal definir els atributs de l'Entitat amb variables de tipus bàsic tals com Integer, String, etc. Les entitats han de tenir l'anotació **@Entity**.

A més, per tal de configurar el comportament dels atributs quan aquests esdevinguin camps de la taula podem afegir diverses anotacions. Per exemple l'anotació **@GeneratedValue(strategy = GenerationType.IDENTITY)** defineix que la id es genera de manera automàtica i indica quina estratègia de generació s'utilitza.

Els **Repositoris** ens ofereixen un metode d'access a les taules creades per les entitats. Per a cada entitat crearem el seu respectiu repositori. A continuació mostrem com definir un repositori:

```
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.transaction.annotation.Transactional;

public interface BasicRepo extends CrudRepository<BasicEntity, Long> {
    // consultors
    BasicEntity findById(Integer id);
}
```

Així doncs definim un repositori utilitzant una interfície i definint els consultors que volem utilitzar. Spring és capaç de generar aquests consultors a partir del nom que tingui la funció.

Així doncs per exemple la funció de nom **finById(Integer id)** buscarà una *BasicEntity* amb la *id* que li hem passat.

Finalment, amb els **Controladors** agrupem i definim les crides rest. En la nostra aplicació hem definit 3 controladors:

- **UserController** - Agrupa les crides relacionades amb la gestió d'usuaris.
- **FileController** - Agrupa les crides relacionades amb la gestió de fitxers.
- **AuthController** - Agrupa les crides relacionades amb l'autenticació d'usuaris.

Els controlador han de portar l'anotació **@Controller** i si s'escau poden portar també l'anotació **@RequestMapping(path="/path")** indicant un path global per a totes les crides del controlador.

Dins els controladors definirem els repositoris que volem utilitzar, ajudant-nos de l'anotació **@Autowired**, així com les diferents crides rest. Aquestes poden portar diferents tipus d'anotacions, nosaltres hem utilitzat l'anotació **@RequestMapping** que és la més general i ens permet definir qualsevol tipus de crida.

A continuació veurem un exemple d'un controlador petit amb una crida rest que demana una *BasicEntity*:

```
import static org.springframework.web.bind.annotation.RequestMethod.*;

@Controller
@RequestMapping(path="/path")
public class BasicController {
    // Attributes
    @Autowired
    private BasicRepo basicRepo;

    // Rest calls
    @RequestMapping(path = "/get/{id}", method = GET)
    public @ResponseBody ResponseEntity getBasicEntity(
        @PathVariable Integer id
    ){
        BasicEntity basicEntity = userRepository.findById(userId);
        return ResponseEntity.status(HttpStatus.OK).body(basicEntity);
    }
```

A part d'aquest tres components també podem definir:

- **Models** - Són classes de Java preparades per a ser convertides a un fitxer JSON (JavaScript Object Notation) i viceversa. El seu únic propòsit és el de permetre enviar i rebre estructures de dades entre el client i el servidor.
- **Serveis** - Són classes de Java anotades amb l'anotació **@Service**, aquestes es poden enllaçar amb l'anotació **@AutoWired** a qualsevol controlador per tal d'utilitzar-los.

En el nostre servidor hem implementat 2 serveis:

- **StorageService**
- **AuthenticationService**

El primer servei s'encarrega de buscar fitxers i d'emmagatzemar-los així com de llançar excepcions quan sigui necessari. El segon servei s'encarrega de comprovar que les autoritzacions, rebudes per part, del client són valides.

Per guardar les contrasenyes s'ha fet servir l'algorisme de hash **SHA3-512** [13.3] mentre que la llibreria que genera els **JWT** [13.4] utilitza un algorisme més antic el **SHA2-256** [13.3].

9.2 Implementació de l'aplicació

Pots descarregar el codi font de l'aplicació en el següent repositori: https://github.com/arracso/HeartRate.

S'ha implementat una aplicació Android que té una quantitat relativament petita d'activitats però, al darrere, porta implementat tot un sistema de comunicació Bluetooth. Així doncs més que explicar la implementació de la interfície gràfica ens centrarem a explicar la implementació d'aquells aspectes més importants del sistema o que han portat més complicacions.

9.2.1 El servei de Bluetooth

S'ha implementat un servei per tal de gestionar la connexió amb la polsera fins i tot quan l'aplicació estigui tancada. La connexió amb la polsera te diversos estats els quals han estat organitzats en diverses interfícies que són implementades pel **BluetoothService** aquests són:

- **Escaneig** - Implementada amb la interfície IScanService.
- **Empare-llament** - Implementada amb la interfície IPairService.
- **Connectat** - Implementada amb la interfície IDeviceService.

Tot i així la lògica principal del servei esta implementada en una classe anomenada **ConnectionManager** i que es inicialitzada pel Servei de Bluetooth.

Els serveis d'Android actuen com a processos independents de l'aplicació. Per tal de poder fer-los servir des de les activitats i viceversa s'han fet servir IBinders i s'ha creat una variable en el servei per a cada activitat que el vol fer servir de forma activa.

```
1  public class BluetoothService extends Service implements
2    IBluetoothService, IScanService, IPairService, IDeviceService {
3      private final IBinder bluetoothBinder = new BluetoothBinder();
4
5      private IScanView scanView;
6      private IPairView pairView;
7      private IDeviceView deviceView;
8
9      private ConnectionManager connectionManager;
10
11     @Override
12     public IBinder onBind(Intent intent) {
13         return bluetoothBinder;
14     }
15
16     @Override
17     public int onStartCommand(Intent intent, int flags, int startId) {
18         super.onStartCommand(intent,flags,startId);
19         return START_STICKY;
```

```
    }

    public class BluetoothBinder extends Binder {
        public BluetoothService getService() {
            return BluetoothService.this;
        }
    }
}
```

Al codi podem observar una simplificació del servei de Bluetooth. A la funció **onStartCommand** definim que el servei comenci i es mantingui actiu i la classe **BluetoothBinder** ens serveix per donar un accés als mètodes públics del servei.

9.2.2 Gestió de la connexió Bluetooth

La **ConnectionManager** és una classe abstracta que haurem d'estendre per a cada dispositiu diferent que vulguem connectar amb l'aplicació. En el nostre cas només volem connectar la Mi Band 1s, així que només l'hem estesa un cop i el servei de Bluetooth realment no mira quin tipus de dispositiu estem connectant sinó que tan sols restringeix la connexió a aquestes polseres.

La classe **ConnectionManager** hereta de **BluetoothGattCallback** la qual ens permet gestionar la connexió amb el dispositiu.

Abans que la connexió pugui ser gestionada per la classe **ConnectionManager** hem d'escanejar els dispositius amb l'adaptador de Bluetooth i connectar un dels dispositius escanejats amb el **conectionManager**. A continuació pots veure les línies de codi més elementals per tal de fer-ho.

```
// Use bluetooth adapter to scan devices
adapter.getBluetoothLeScanner().startScan(scanCallback);
// Stop scan
adapter.getBluetoothLeScanner().stopScan(scanCallback);
// Connect device using connectionManager
device.connectGatt(context: this, autoConnect: false, connectionManager);
// Disconnect the device connected using the connectionManager
connectionManager.disconnect();
```

La classe **ScanCallback** rep els dispositius escanejats i decideix que fer amb ells. En el nostre cas els filtrem i els mostrem per pantalla per tal que l'usuari esculli amb quin dispositiu es vol connectar.

Un cop connectat el dispositiu hem gestionat qualsevol comanda enviada al servei a través del **connectionManager** gràcies a una cua d'accions que delimita un ordre a l'hora d'enviar les comandes. Java no ofereix una classe que implementi una cua senzilla (totes porten extres implementats que molestaven), així doncs s'ha creat una classe **Queue** que té la capacitat d'encuar, desencuar i encuar en primera posició. També hem programat diverses classes abstractes que es comporten diferent: acció amb resultat, sense resultat i amb resultat condicional.

9.2.3 El servei de Crides Rest

El servei de crides rest funciona similar al servei de Bluetooth amb la diferencia que aquest cop en comptes de enllaçar-lo amb tota la logica que porta el Bluetooth darrera definim 3 serveis diferents amb Retrofit 2.0 per tal de fer crides al servidor.

9.2.4 Protocol Mi Band 1S

Un dels principals problemes que han sorgit durant la implementació de l'aplicació ha estat el de descobrir el protocol per a la Mi Band 1S. A Internet hi ha bastants aplicacions de codi obert que ens han ajudat a esbrinar algunes de les comandes de la Mi Band 1S. Així i tot, no han estat suficients així que el que hem fet és provar una a una totes les característiques de la polsera a veure què feien. D'aquesta manera hem pogut obtenir més informació que s'ha vist reflectida en la classe **MiBandConstants**. Així i tot, alguns dels protocols, com ara el de comptar els passos, no han estat desxifrats degut a la seva complexitat.

A la figura 9.1 podem veure com s'estructuren els serveis que ofereix un dispositiu Bluetooth Low Energy o el que anomenem **Generic Attribute Profile** o **GATT**. En aquest cas ens trobem un exemple de servei de ritme cardíac.

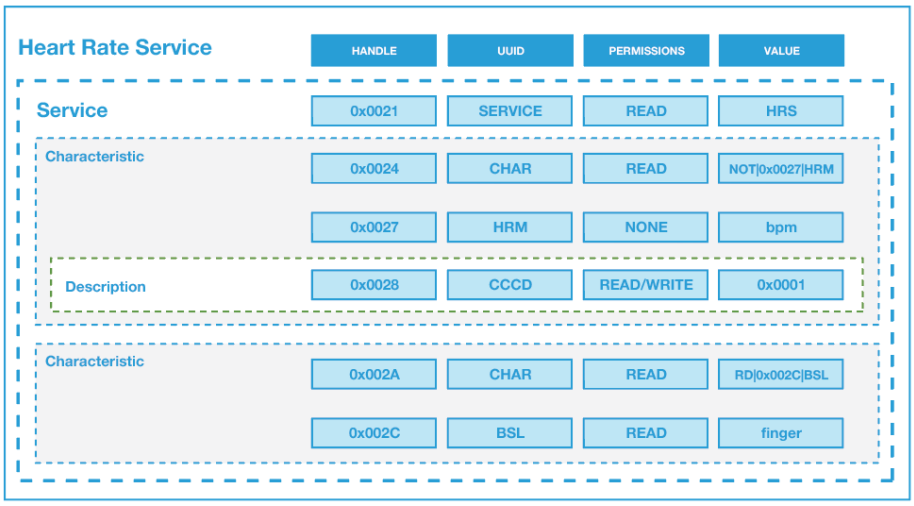

Figura 9.1: Estructura dels serveis de Bluetooth Low Energy

Al següent enllaç pot trobar totes els serveis i característiques trobats amb el seu identificador: https://github.com/arracso/HeartRate/wiki/Mi-Band-1S

9.2.5 Emmagatzematge i enviament de les dades

En un inici vaig implementar una classe, que ens ajudava a obrir un fitxer, a escriure en ell i desprès a tancar-lo, per tal de guardar el ritme cardíac. Aquest mètode però no era gaire fiable, eficient ni resistent a fallades.

Així doncs, vaig demanar consell a la Natàlia, companya del grup eXiT, i em va recomanar que utilitzes SQLite. Com ja hem explicat anteriorment SQLite és una llibreria per tal d'utilitzar Bases de Dades en gairebé qualsevol entorn.

Així doncs, un cop implementada la base de dades que guardaria el ritme cardíac, aquesta vegada es van guardant les dades capturades a la base de dades i un cop s'han d'enviar al servidor es bolquen a un fitxer CSV, que enviarem al servidor, i s'eliminen de la base de dades.

9.2.6 Mantenir els serveis funcionant

Mantenir els serveis en funcionament un cop es tanca l'aplicació ha estat el problema més considerable al llarg del desenvolupament d'aquest projecte. Per tal de mantenir els serveis funcionant s'han pres les següents mesures:

- Obtenir permisos d'inici automàtic i d'administració d'energia per part de l'usuari. D'aquesta manera podem evitar el **Doze Mode d'Android**.
- Crear alarmes que enviïn un missatge per despertar els serveis de forma periòdica.
- Crear receptors que rebin els missatge de les alarmes, així com de que s'ha reiniciat el sistema i comprovin si el servei esta obert i treballant. En cas que no ho estiguin fent els envia l'ordre per a que ho facin.
- Obligar al servei a mantenir-se despert un mínim de 10 minuts quan aquest rep l'ordre de capturar dades.

De totes aquestes mesures la d'evitar el **Doze Mode** ha estat la més complicada. Ja que les noves versions d'Android apaguen els seus serveis al cap d'uns minuts en que la pantalla esta apagada, ha estat difícil detectar aquest problema. També ha portat bastant de temps esbrinar de que es tractava exactament i idear algun mètode per solucionar-lo.

9.3 Proves

9.3.1 Proves del servidor

Les crides rest implementades al servidor han estat provades amb l'aplicació **Postman**.

En el següent enllaç podeu descarregar el fitxer de crides efectuades: https://github.com/arracso/FitServer/blob/master/FitAPI.postman_collection.json.

9.3.2 Proves de l'aplicació

Per tal de testejar l'aplicació s'han fet les següents proves amb resultats satisfactoris:

- Inici de sessió, registre i inici de sessió com a invitat amb i sense Internet.
- Modificació de les dades d'usuari amb i sense Internet.
- Obrir l'aplicació un cop iniciada la sessió amb i sense Internet.
- Enviar fitxers al servidor des de l'aplicació.
- Fer captures de dades de varis períodes de longitud (des de 10 segons fins a 2 dies).
- Enviar comandes a la polsera des de l'aplicació. Incloses comandes que no estan incloses en l'aplicació actualment com ara fer que la polsera vibri.
- Comprovar la reconnexió de la polsera en allunyar-la prou per a que aquesta es desconnecti i desprès apropar-la de nou.
- Tancar i obrir l'aplicació reiterades vegades des de el boto proporcionat o des de l'administrador de tasques i comprovar que el servei es capaç de mantenir-se viu i/o de reiniciar-se.

Capítol 10

Implantació i resultats

El sistema desenvolupat en aquest projecte està pendent de ser implantat.

Seguidament explicaré tot el necessari per tal d'implantar el sistema. A més, al capítol 14 podem trobar manuals de instal·lació per al servidor i per a l'aplicació.

10.1 Implantació del servidor

L'aplicació del servidor està previst d'instal·lar-se en un dels servidors del grup eXiT de la Universitat. Per tal de fer-ho haurem d'obrir un port de la xarxa de la Universitat i instal·lar l'aplicació WAR (Web Application Files) en un servidor Apache.

Per tal d'obtenir un WAR de l'aplicació del servidor és necessari comprar la llicència d'IntelliJ Idea.

També s'hauran de fer les configuracions pertinents de la base de dades indicades en els manuals d'instal·lació.

10.2 Implantació de l'aplicació

Per tal d'implantar completament l'aplicació faria falta registrar-la a Google Play. Google permet enregistrar un nombre il·limitat d'aplicacions a aquells que hagin pagat la quota de registre de Google Play.

Un cop registrada l'aplicació a Google Play aquesta apareixerà a la Play Store i Google s'emportarà un petit percentatge dels beneficis de publicitat, venda i/o microtransaccions que pugui haver-hi en l'aplicació.

10.3 Resultats

Tot i que el sistema no esta implantat hem pogut monitorar durant 2 dies seguits el ritme cardíac. Quan vam parar la captura de dades la polsera encara mantenia un 25% de la bateria, així que només caldrà carregar-la un cop cada dos dies. La carrega de la polsera necessita poc més de 2 hores per estar al 100%.

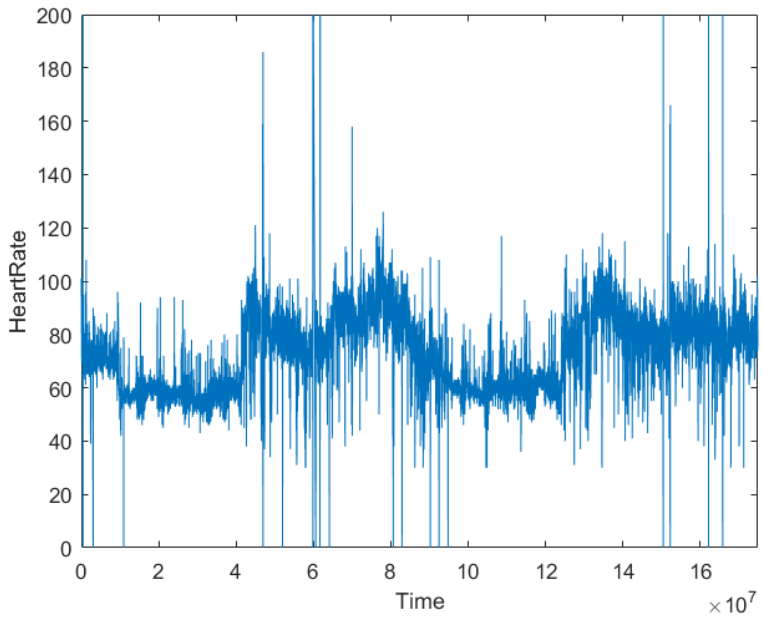

Figura 10.1: Gràfic del ritme cardíac d'una persona durant 48 hores

A la figura 10.1 es pot observar un gràfic amb el ritme cardíac capturat per la polsera en 2 dies. La polsera envia el ritme cardíac a l'aplicació cada cop que percep algun canvi en el ritme cardíac. Així doncs, el temps es mostra en mili-segons. Podem obtenir un increment de la durada de la bateria si delimitem un període de temps entre captura i captura. Així i tot, s'ha donat prioritat a implementar la captura el més continua possible per tal de poder fer servir les dades en qualsevol tractament.

En alguns punts es poden observar mesures que superen les 200 pulsacions per minut o que arriben a 0 pulsacions per minut. Això tan sols és a causa de la precisió de la polsera, si aquesta se separa una mica de la pell, immediatament envia 0 pulsacions i depenent dels moviments de l'usuari també pot arribar a donar pulsacions exageradament grans.

Deixant de banda aquestes dades atípiques, podem observar clarament els períodes en què la persona monitorada estava dormint si mirem aquells períodes en què el ritme cardíac és clarament més baix.

Capítol 11

Conclusions

Personalment, crec que amb aquest projecte he après molt. Per una banda, he pogut ampliar dràsticament el meu coneixement a l'hora de desenvolupar aplicacions per a mòbil amb Android i servidors web. D'altra banda, he après com funcionen els dispositius Bluetooth i les tecnologies que fan servir aquests dispositius per prendre mesures. Finalment, l'experiència de treballar en un equip fent projectes relacionats amb la salut ha estat molt grata i enriquidora per a mi.

Crec que els objectius plantejats s'han assolit satisfactòriament i considero que el monitoratge de constants vitals fent ús de dispositius vestibles comportarà una part important de les nostres vides en un futur. Mentre que en aquest projecte hem aconseguit mesurar constantment el ritme cardíac carregant la polsera només un cop cada dos dies, considero que en un futur no gaire llunyà, es crearan polseres que durin més temps i que a més puguin emmagatzemar les dades fàcilment evitant la necessitat d'estar connectat al telèfon mòbil les 24 hores del dia.

En el cas de la Mi Band 1S no s'han pogut guardar les dades capturades en la mateixa polsera, ja que, tot i que disposa d'una memòria interna, no té cap protocol que permeti dir-li a la polsera que guardi les dades capturades en la seva memòria. La polsera només fa servir la seva memòria interna de manera automàtica quan activa també de manera automàtica el monitoratge del ritme cardíac quan detecta que estem dormint. En el nostre projecte ens interessava que monitores sempre així que hem deshabilitat aquesta funció per evitar possibles errors.

Capítol 12

Treball Futur

El projecte tal com es troba actualment no és més que un prototip funcional, com a tal està subjecte a moltes millores. A continuació es detallen les millores que es podrien fer en un futur:

- Implementar el protocol https en el servidor.
- Encriptar automàticament els fitxers de dades que es pugen al servidor.
- Fer una millora estètica de l'aplicació.
- Permetre que un usuari invitat sigui promogut a un usuari registrat.
- Permetre una major gestió del compte d'usuari (eliminació d'aquest, canvi de correu, canvi de contrasenya, recuperació de contrasenya, etc). Per poder fer aquesta millora seria necessari guardar el correu electrònic a la base de dades. Així doncs també s'haurà de millorar de la seguretat i complir la **LOPD** [13.2].
- Acabar de desxifrar el protocol de la Mi Band 1S.
- Ampliar el tipus de dades que es poden monitorar amb l'aplicació: GPS, passos, etc; així com permetre un monitoratge del ritme cardíac a intervals diferents per tal d'estalviar bateria.
- Implementar una aplicació web, que permeti accedir a les dades capturades de l'usuari i visualitzar-les en gràfiques. També ha de permetre que l'usuari gestioni el seu compte.
- Donar compatibilitat a altres polseres com ara la Xiaomi Mi Band 2 o la Fitbit Charge HR.
- Implementar funcionalitats extres no relacionades amb el monitoratge de dades, com ara vibracions i alarmes, per tal de fer que l'aplicació sigui més comercial.

- Traduir l'aplicació al Català i al Castella, actualment esta en angles, i permetre canviar l'idioma des de l'aplicació.
- Implementar un pretractament de les dades en l'aplicació el qual detecti i corregeixi errors de precisió dels sensors.
- Estudiar les diferents formes en que es poden tractar les dades obtingudes per tal de trobar possibles malalties o fer un seguiment de pacients.

Com podem veure, es poden fer una gran quantitat de millores en el projecte que segurament un cop fetes mostrin més vies de desenvolupament i investigació.

Bibliografia

[1] Alfonso De Frutos. Estas son las 5 mejores pulseras deportivas de 2017. https://topesdegama.com/noticias/gadgets/mejores-pulseras-deportivas-2017, 2017. Accedit: 19/08/2018.

[2] Android Developers. Schedule repeating alarms. https://developer.android.com/training/scheduling/alarms, 2018. Accedit: 14/06/2018.

[3] Android Developers. Bluetooth. https://developer.android.com/guide/topics/connectivity/bluetooth?hl=es-419, 2018. Accedit: 02/02/2018.

[4] Android Developers. BroadcastReceiver. https://developer.android.com/reference/android/content/BroadcastReceiver, 2018. Accedit: 05/06/2018.

[5] Android Developers. Servicios. https://developer.android.com/guide/components/services?hl=es-419, 2018. Accedit: 15/02/2018.

[6] Android Developers. Save data using SQLite. https://developer.android.com/training/data-storage/sqlite, 2018. Accedit: 25/05/2018.

[7] Angela Moscaritolo. El 99.6% del mercado móvil le pertenece a Android y iOS. https://latam.pcmag.com/sistemas-operativos-moviles/18490/news/el-996-del-mercado-movil-le-pertenece-a-android-y-ios, 2017. Accedit: 20/08/2018.

[8] Arantxa Asián. 5 de las mejores pulseras inteligentes que puedes comprar hoy. https://www.tuexperto.com/2017/11/15/5-de-las-mejores-pulseras-inteligentes-que-puedes-comprar-hoy/, 2017. Accedit: 19/08/2018.

[9] Dispositivos Wearables. Dispositivos Wearables. http://www.dispositivoswearables.net/, 2014. Accedit: 19/08/2018.

[10] Enrique Pérez. Cómo funciona Doze, el sistema de ahorro de batería de Android. https://elandroidelibre.elespanol.com/2015/

12/doze-sistema-ahorro-bateria-android.html, 2015. Accedit: 18/06/2018.

[11] Grupo Espiral Php. Modelo Espiral. http://modeloespiral.blogspot.com/, 2009. Accedit: 18/08/2018.

[12] Ivan. Cómo miden las pulseras deportivas nuestro ritmo cardíaco. https://elandroidelibre.elespanol.com/2016/06/como-miden-pulseras-deportivas-ritmo-cardiaco.html, 2016. Accedit: 20/08/2018.

[13] Javier Penalva. Fitbit Charge HR, análisis: el mejor cuantificador del mercado. https://www.xataka.com/analisis/fitbit-charge-hr-analisis-el-mejor-cuantificador-del-mercado, 2015. Accedit: 20/08/2018.

[14] Javier Penalva. Xiaomi Miband 2, análisis: es barata y funciona pero no esperes mucho más. https://www.xataka.com/analisis/xiaomi-miband-2-analisis-es-barata-y-funciona-pero-no-esperes-mucho-mas, 2016. Accedit: 20/08/2018.

[15] Juan Carlos González. Xiaomi MiBand 1S, análisis: el wearable para la gente que no quiere (o entiende) un wearable. https://www.xataka.com/wearables/xiaomi-miband-1s-analisis, 2015. Accedit: 20/08/2018.

[16] Les Hazlewood & Okta. JJWT. https://github.com/jwtk/jjwt, 2018. Accedit: 21/05/2018.

[17] Ok Hosting. Principales metodologías de desarrollo de software. https://okhosting.com/blog/principales-metodologias-de-desarrollo-de-software/, 2016. Accedit: 18/08/2018.

[18] sergiodxa. Introducción a JSON Web Tokens (JWT). https://platzi.com/blog/introduccion-json-web-tokens/, 2016. Accedit: 20/05/2018.

[19] T Systems. La revolución inteligente en la atención médica. https://www.t-systems.com/es/es/newsroom/perspectives/internet-of-things/iot/wearable-devices-554110, 2017. Accedit: 19/08/2018.

[20] Wikipedia. Llei Orgànica de protecció de dades de caràcter personal Jump to navigationJump to search . https://ca.wikipedia.org/wiki/Llei_Orgànica_de_protecció_de_dades_de_caràcter_personal, 2009. Accedit: 29/08/2018.

[21] Wikipedia. Barry Boehm. https://es.wikipedia.org/wiki/Barry_Boehm, 2016. Accedit: 18/08/2018.

[22] Wikipedia. Secure Hash Algorithm. https://es.wikipedia.org/wiki/Secure_Hash_Algorithm, 2018. Accedit: 13/05/2018.

Capítol 13

Annexos

13.1 Barry Boehm

Barry W. Boehm (n. 1935) és un enginyer informàtic estatunidenc i també és professor emèrit d'aquesta matèria en el departament de ciències tecnològiques de la Universitat del Sud de Califòrnia. És conegut per les seves múltiples aportacions en aquest camp.

Figura 13.1: Foto de perfil de Barry Boehm.

El 1957, va rebre el seu grau a Harvard. Va obtenir els seus graus de Màster i Doctorat en Matemàtiques a la Universitat de Califòrnia en els anys 1961 i 1964, respectivament.

El 1955, va començar a treballar com a analista de programes a General Dynamics. El 1959 va passar a la RAND Corporation, on va ser cap del departament de ciències informàtiques fins a 1973. Des d'aquell any fins a 1989 va ser científic en cap de l'equip de defensa de sistemes a TRW. Entre 1989 i 1992, va servir en el departament de defensa dels Estats Units com a director a l'oficina de les ciències i tecnologia de la informació de Defense Advanced Research Projects Agency (DARPA), i com a director del software de Director, Defense Research & Engineering a l'oficina d'informàtica. Des de 1992 és professor d'enginyeria informàtica en el departament de ciències de la informàtica a la Universitat del Sud de Califòrnia.

Les seves contribucions al camp inclouen el model espiral del procés del programari, el model constructiu del cost (COCOMO), l'acostament de la teoria W (win-win) a la determinació de la gerència i dels requisits del programari i a dos ambients avançats de la tecnologia de dotació lògica: el sistema i el Quantum de la productivitat del programari de TRW.

Actualment és membre de l'acadèmia nacional d'enginyeria i treballa en la AIAA, en ACM i en IEEE.

13.2 Llei Orgànica de Protecció de Dades de Caràcter Personal

La llei orgànica 15/1999 de protecció de dades de caràcter personal (LOPD) és una llei orgànica que té l'objecte de protegir la intimitat i la privadesa personal i familiar. Fou aprovada a les Corts espanyoles el 13 de desembre del 1999. Aquesta llei es desenvolupa fonamentada en l'article 18 de la Constitució espanyola de 1978, sobre el dret a la intimitat familiar i personal i el secret de les comunicacions.

Aquesta llei afecta a totes les dades que fan referència a persones físiques registrades sobre qualsevol suport, informàtic, o no.

Queden exempts del compliment d'aquesta normativa aquelles dades recollides per a ús domèstic, les matèries classificades de l'estat i aquells fitxers que recullen dades sobre terrorisme i altres formes de delinqüència organitzada (no s'hi inclou la delinqüència simple).

A partir d'aquesta llei es varen crear l'Agència Espanyola de Protecció de Dades, d'àmbit estatal, i l'Agència Catalana de Protecció de Dades, en l'àmbit de Catalunya, que vetllen pel compliment d'aquesta normativa.

13.3 Secure Hash Algorithm

El SHA (Secure Hash Algorithm, Algoritme de Hash Segur) és una família de funcions de xifratge hash publicada per l'Institut Nacional de Normes i Tecnologia (NIST) dels Estats Units. La primera versió de l'algoritme es va crear el 1993 amb el nom de SHA, tot i que en l'actualitat se la coneix com a SHA-0 per tal d'evitar confusions amb les versions posteriors. La segona versió del sistema, publicada amb el nom de SHA-1, va ser publicada dos anys més tard. Posteriorment s'han publicat SHA-2 el 2001 (formada per diverses funciones: SHA-224, SHA-256, SHA-384, i SHA-512) i la més recent, SHA-3, que fou seleccionada en una competició de funcions hash celebrada pel NIST el 2012. Aquesta última es caracteritza per ser la que més difereix de les seves predecessores.

Actualment s'han pogut perpetrar atacs contra SHA-0 i SHA-1, de manera que només es consideren segurs SHA-2 i SHA-3.

13.4 JSON Web Token

JSON Web Token (JWT) és un estàndard obert basat en JSON proposat per la IETF (Internet Engineering Task Force) per a la creació de tokens d'accés que permetin la propagació d'identitat i privilegis o **claims** en Angles. Por exemple, un servidor podria generar un token indicant que l'usuari té privilegis d'administrador i proporcionar-lo a un client. El client aleshores podria utilitzar el token per demostrar que està actuant com a un administrador en l'aplicació client o en algun altre sistema. El token està firmat per la clau del servidor, així que el client i el servidor són ambos capaços de verificar que el token és legítim. Els JSON Web Tokens estan dissenyats per a ser compactes, poder ser enviats en les URLs i ser utilitzats en escenaris de Single Sign-On (SSO). Els privilegis dels JSON Web Tokens poden ser utilitzats per a propagar la identitat d'usuaris com a part del procés d'autenticació entre un proveïdor d'identitat i un proveïdor de servei, o qualsevol altre tipus de privilegis requerits.

Els JWT conté els següents elements:

- **header** - JSON que conté informació sobre el tipus de token de que es tracta.

```
header = '{"alg":"HS256","typ":"JWT"}';
```

- **payload** - JSON que conté la informació del token (permisos, identitat, etc).

```
payload = '{"id": 1,"isAdmin": true}';
```

- **signature** - Hash del **header** i la **payload** firmat amb la clau del servidor.

```
key           = 'secretkey';
unsignedToken = encodeBase64Url(header)
              + '.' + encodeBase64Url(payload);
signature     = HMAC-SHA256(key, unsignedToken);
```

El token resultant serà la concatenació amb punts d'aquests 3 elements:

```
token  = encodeBase64Url(header)
       + '.' + encodeBase64Url(payload);
       + '.' + encodeBase64Url(signature);
```

Capítol 14

Manuals

14.1 Manuals d'instal·lació

Els manuals d'instal·lació mostren aquells passos necessaris per a implantar el sistema.

14.1.1 Instal·lació del servidor

El codi font del servidor és pot trobar al següent repositori de GitHub:

- https://github.com/arracso/FitServer

El servidor necessita dues instal·lacions diferents:

14.1.1.1 Instal·lació de la base de dades

Per la base de dades tenim que instal·lar MySql 8.0. En el següent enllaç es pot baixar el seu instal·lador:

- https://dev.mysql.com/downloads/windows/installer/8.0.html

Instal·larem MySql amb les configuracions per defecte exceptuant el mètode d'autenticació [figura 14.1] en el qual farem servir una versió antiga per falta d'una versió més nova per al connector de Sql.

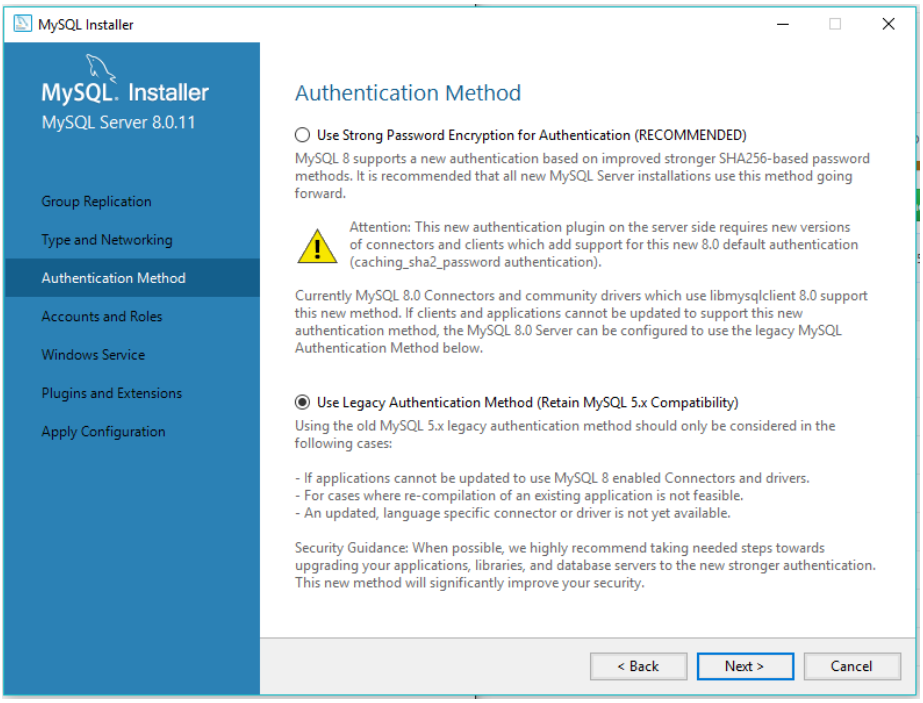

Figura 14.1: Metode d'autenticació de MySql

Finalment obrirem la línia de comandes de MySql [figura 14.2] per tal de crear la base de dades.

```
Enter password: **************
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.12 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Figura 14.2: Linia de comandes de MySql

Utilitzant la línia de comandes crearem la base de dades i un nou usuari. Aquest usuari serà el que farem servir per connectar-nos a la base de dades des de l'aplicació del servidor. Les comandes utilitzades han estat les següents:

```
1 mysql> create database db_fit;
2 mysql> create user 'username'@'localhost' identified by 'password';
3 mysql> grant all on db_fit.* TO 'username'@'localhost';
```

Un cop executat per primer cop l'aplicació del servidor és recomanable canviar els privilegis de l'usuari amb les comandes que es mostren a continuació:

```
1 mysql> revoke all on db_fit.* from 'username'@'localhost';
2 mysql> grant SELECT,INSERT,UPDATE,DELETE ON db_fit.* to 'username'@'localhost';
```

Amb la base de dades creada podem procedir a l'instal·lació de l'aplicació del servidor.

14.1.1.2 Instal·lació de l'aplicació del servidor

El servidor compta amb una arxiu anomenat **application.properties**. En ell podem configurar:

- El port en que s'obrira l'aplicació (per defecte és el 12380)
- Les claus d'encriptació per als tokens
- El directori de pujada de fitxers
- La connexió a la base de dades
- La mida màxima dels fitxers que es pugen

La propietat **spring.jpa.hibernate.ddl-auto** ha d'estar en mode **create** durant la primera execució del servidor, desprès s'haurà de canviar per **none**.

```
# application.properties

# ====== #
# Server #
# ====== #

# Open server this port
server.port = 12380

# ============== #
# Authentication #
# ============== #

# Secret keys for JS web tokens
jwt.key.access = secret_access
jwt.key.access2 = secret_access2
jwt.key.refresh = secret_refresh

# ======= #
# Storage #
# ======= #

# Directory to store uploaded files
storage.dir.upload = upload-dir

# ==================== #
# Database connection #
# ==================== #

# Database url connection
spring.datasource.url = jdbc:mysql://localhost:3306/db_fit?useSSL=false

# Username & password
spring.datasource.username = username
spring.datasource.password = password

# Keep connection alive
spring.datasource.testWhileIdle = true
spring.datasource.validationQuery = SELECT 1

# ============= #
# JPA/HIBERNATE #
# ============= #

# Log sql queries
spring.jpa.show-sql = false

# Database schema update behaviour (create, create-drop, update, none)
spring.jpa.hibernate.ddl-auto = none
```

```

# ============ #
# SERVLET/HTTP #
# ============ #

# Maximum size
spring.servlet.multipart.max-file-size=10MB
spring.servlet.multipart.max-request-size=10MB

# Enable multipart
spring.http.multipart.enabled=false
```

Obrirem **Git Bash** i clonarem el projecte de GitHub:

```
git clone git@github.com:arracso/FitServer.git
```

Ara obrirem IntelliJ Idea i seleccionarem **File > New > Project From Existing Sources..** i crearem un nou projecte en base al projecte clonat.

Un cop creat el projecte podem executar el servidor amb el boto **Run** o be si disposes d'una versió de pagament de IntelliJ Idea pots optar per compilar un arxiu WAR **(Web Application Files)** i executar-lo en un servei d'Apache.

Finalment nomes caldrà obrir el port que fa servir l'aplicació.

14.1.2 Instal·lació de l'applicació

El codi font de l'aplicació i l'apk és poden trobar en els següents enllaços:

- https://github.com/arracso/HeartRate
- https://github.com/arracso/HeartRate/blob/master/HeartRate.apk

L'aplicació utilitza un nom DNS per connectar-se a Internet. Si no disposa de permís per a canviar l'adreça IP a la qual apunta el nom DNS o ha canviat el port per defecte en el servidor, haurà de crear un projecte a IntelliJ Idea basat en el repositori de l'aplicació i canviar el valor de la variable **BASE_ADDR** de la classe **ApiService** [figura 14.3]. Finalment haurà de generar un nou arxiu APK.

```
TodoApp.java   ApiService.java
1   package edu.udg.exit.heartrate.Services;
2
3   import ...
21
22  /**
23   * Service that handles api rest services initializations and access.
24   */
25  public class ApiService extends Service {
26
27      ///////////////
28      // Constants //
29      ///////////////
30
31      private static final String BASE_URL = "http://rateserver.tk:12380";
32
```

Figura 14.3: Adreça base per a fer les crides rest

L'aplicació **HeartRate** no esta enregistrada a **GooglePlay** així que per tal d'instal·lar-la en el seu dispositiu Android haurà d'habilitar les aplicacions d'orígens desconeguts. Per fer-ho vagi a **Ajustes** > **Ajustes adicionales** > **Privacidad** i activi **Orígenes desconocidos**[figura 14.4].

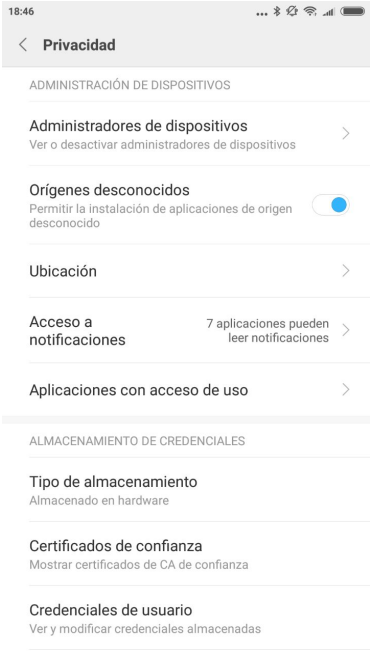

Figura 14.4: Pantalla de privacitat

14.2 Manual d'usuari

En aquest manual aclarirem qualsevol dubte que pugui tenir l'usuari a l'hora d'utilitzar l'aplicació.

14.2.1 Iniciar sessió

Per tal d'iniciar la sessió tan sols cal que obri l'aplicació i empleni el formulari d'inici de sessió [figura 14.5] amb el seu correu electrònic i la seva contrasenya. Un cop iniciada la sessió serà redirigit a la pàgina principal [14.2.3] cada cop que obri l'aplicació. En cas de no tenir un compte d'usuari, si us plau, accedeixi a la pantalla de registre [14.2.2]. També, si així ho desitja, pot iniciar sessió com a un usuari invitat polsant el boto **LOGIN AS GUEST**.

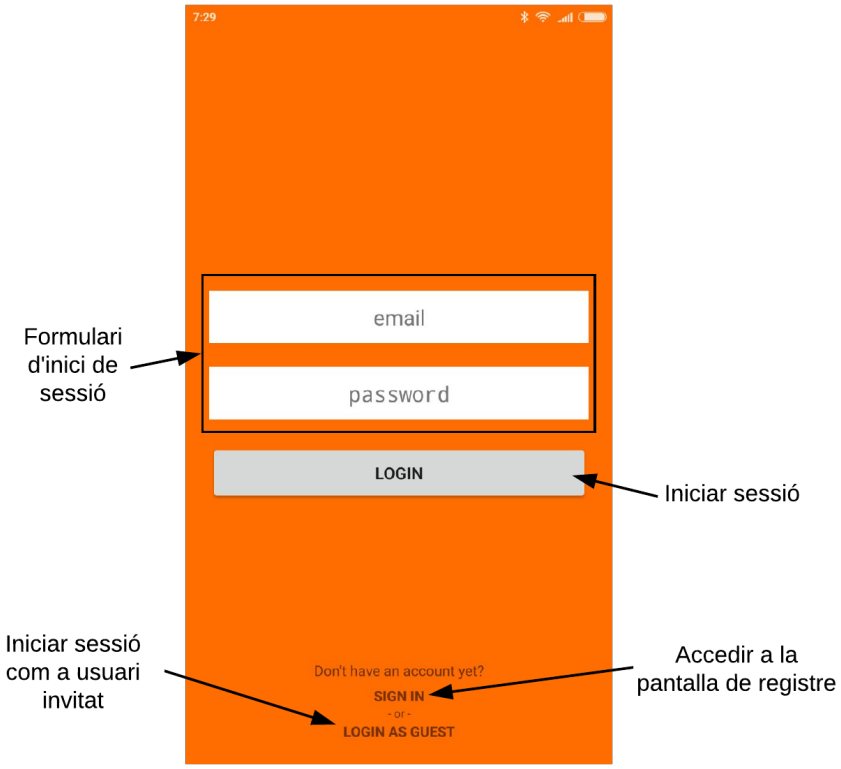

Figura 14.5: Pantalla d'inici de sessió

14.2.2 Registrar-se

Empleni el formulari de la pantalla de registre [figura 14.6] amb el seu correu electrònic i la seva contrasenya. En cas que no existeixi cap compte amb aquestes credencials se'n creara un de nou i serà redirigit a la pantalla principal [14.2.3].

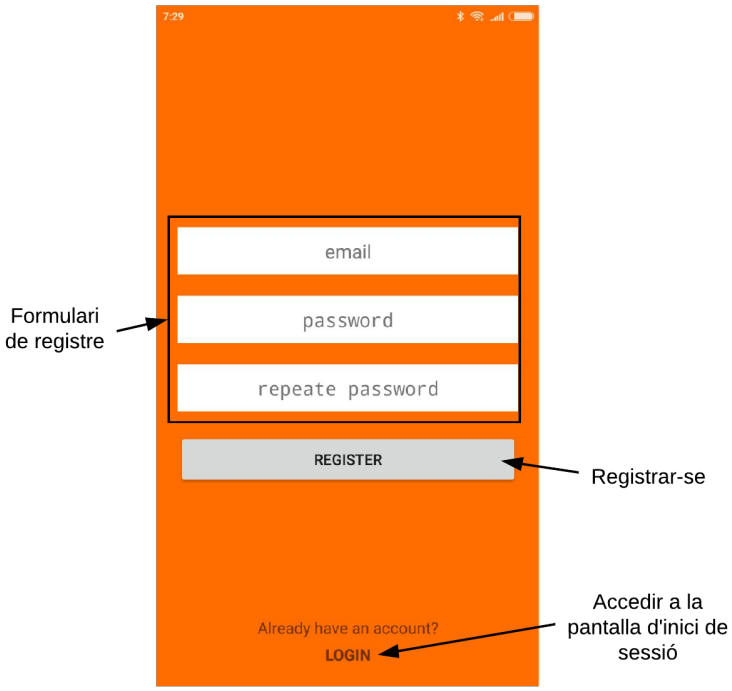

Figura 14.6: Pantalla d'inici de sessió

14.2.3 La pantalla principal

La pantalla principal és aquella que se li obrirà cada cop que obri l'aplicació si ha iniciat sessió prèviament. En ella té la capacitat d'editar les seves dades personals així com de tancar sessió o l'aplicació i accedir a la pantalla del dispositiu emparellat o emparellar-ne un. També pot desemparellar el dispositiu des de aquesta pantalla. A la figura 14.7 pot observar les diferents opcions de la pantalla principal que disposa.

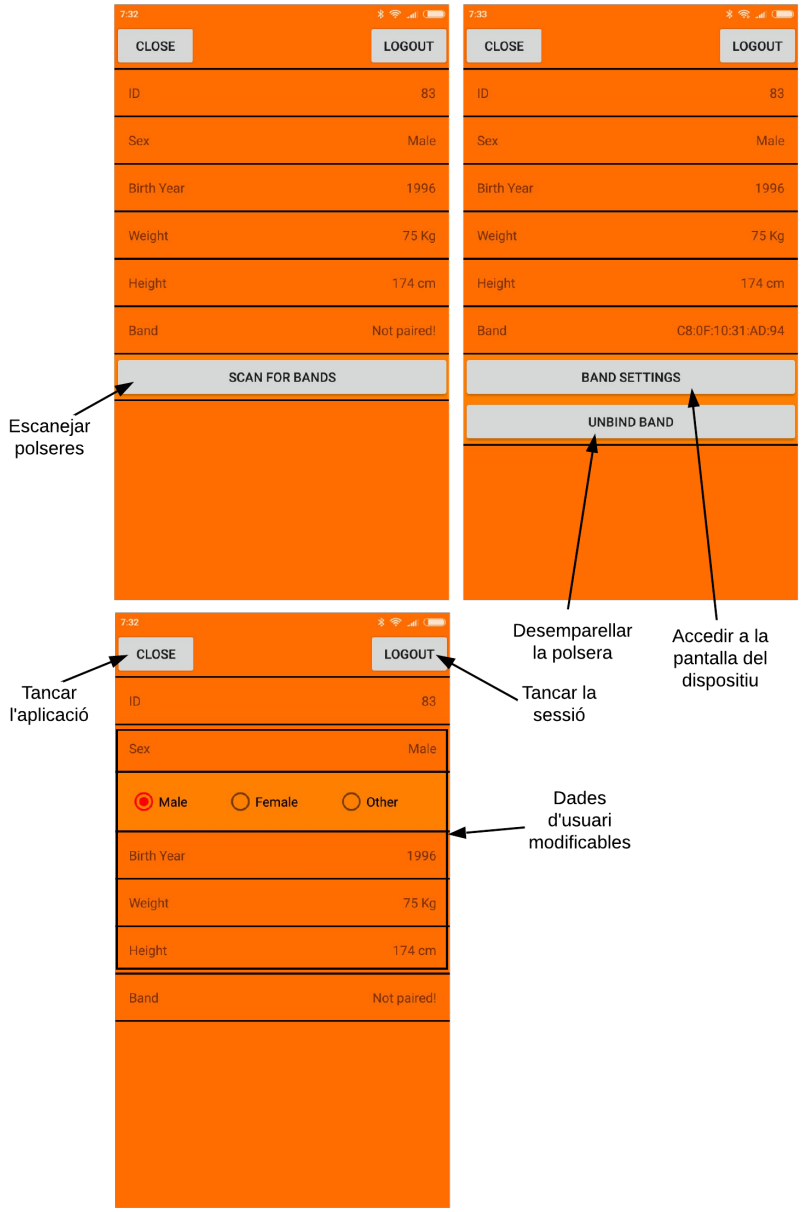

Figura 14.7: Pantalla principal

14.2.4 Tancar la sessió

Per tal de tancar la sessió accedeixi a la pantalla principal [14.2.3] i premi el boto per tancar la sessió.

14.2.5 Tancar l'aplicació

A la pantalla principal [14.2.3] disposa d'un botó per tancar l'aplicació. Tot i que pot tancar l'aplicació mitjançant el **Gestor de Tasques** del dispositiu mòbil és recomanable que no ho faci per tal d'evitar possibles problemes de funcionament a l'hora de monitorar les dades.

14.2.6 Emparellar una polsera

Per tal d'emparellar una polsera accedeixi a la pantalla d'escaneig a traves de la pantalla principal [14.2.3]. Si és el primer cop que entreu en aquesta pantalla l'aplicació us requerira permisos de localització [figura 14.8] i de Bluetooth, si aquest no esta activat.

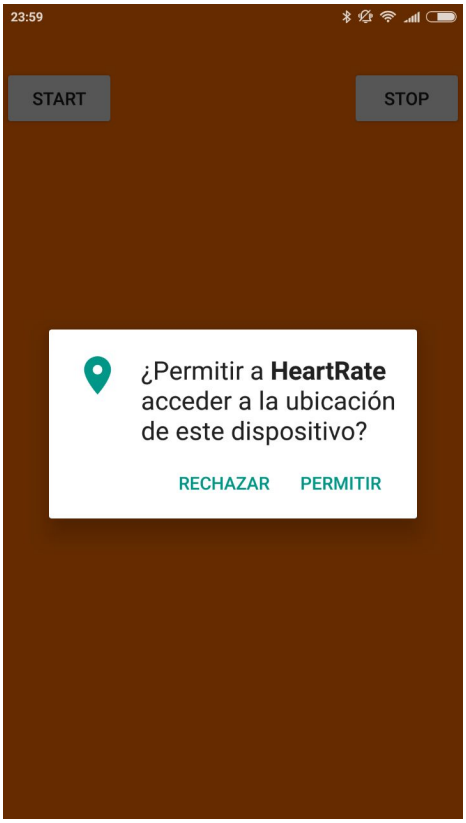

Figura 14.8: Finestra demanant permís de localització

Un cop en la pantalla d'escaneig [figura 14.9] seleccioni la polsera amb la qual vulgui emparellar-se. Automàticament l'aplicació el redirigira a la pantalla d'emparellament [figura 14.10].

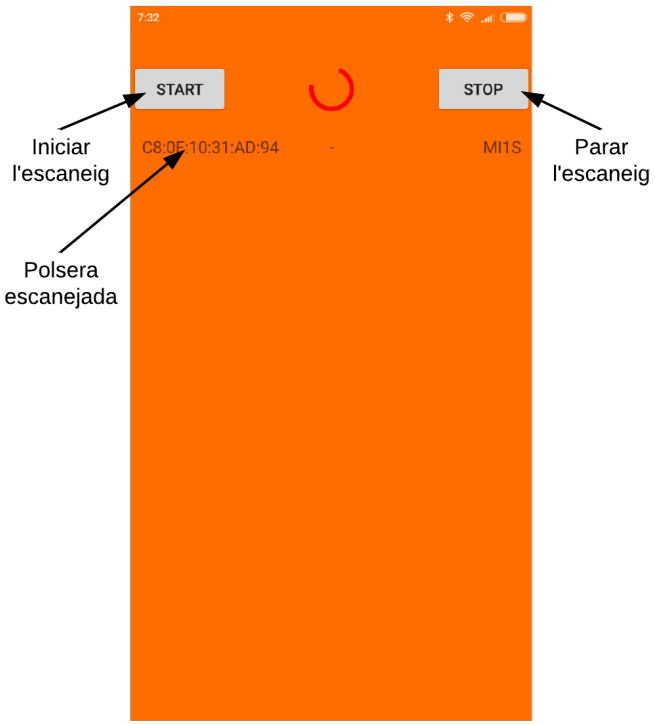

Figura 14.9: Pantalla d'escaneig

Esperi que la polsera Xiaomi Mi Band 1S vibri i premi-la diverses vegades quan aquesta ho faci.

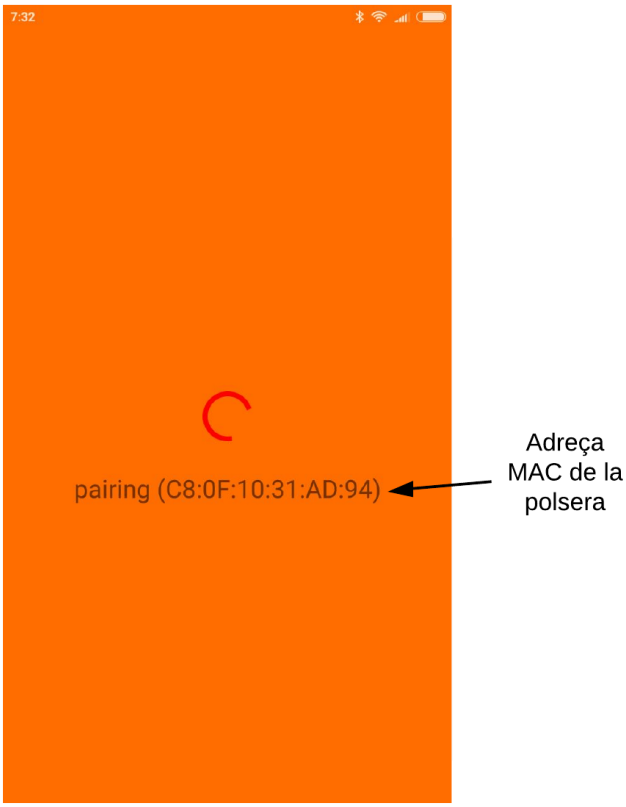

Figura 14.10: Pantalla d'emparellament

Un cop finalitzat l'emparellament serà redirigit a la pantalla del dispositiu. En cas que la polsera no vibri retrocedeixi a la pantalla principal, tanqui la sessió i reiniciï l'aplicació.

14.2.7 Desemparellar una polsera

Es pot desemparellar una polsera des de la pantalla principal [14.2.3] o be des de la pantalla del dispositiu [14.2.8].

14.2.8 La pantalla del dispositiu

Es pot accedir a la pantalla del dispositiu des de la pantalla principal [14.2.3] o bé després d'emparellar una polsera [14.2.6]. En ella podrà veure informació bàsica de la seva polsera tal com el seu nom, la seva adreça MAC o la quantitat de bateria que li resta.

A la figura 14.11 observarà les diferents funcions de la pantalla del dispositiu.

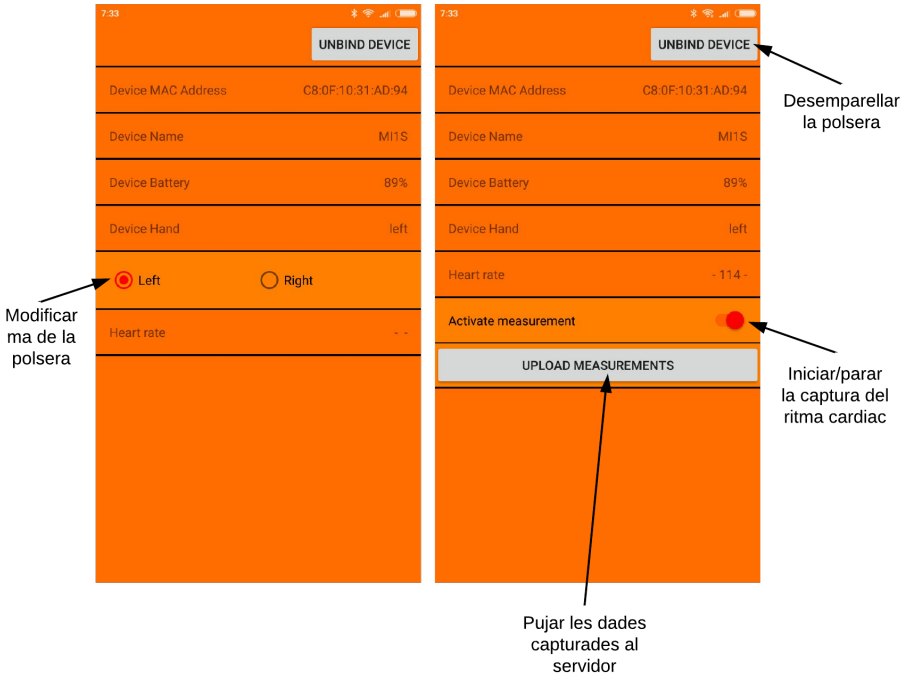

Figura 14.11: Pantalla del dispositiu

En aquesta pantalla és on ha de configurar quina ma porta la polsera, així com activar i desactivar la captura del ritme cardíac. També pot enviar les dades capturades de forma manual i desemparellar la polsera.

14.2.9 Captura del ritme cardíac

Pot activar la captura del ritme cardíac a través de la pantalla del dispositiu [14.2.8]. Al activar la captura per primer cop l'aplicació li demanara que accepti atorgar alguns permisos a l'aplicació. Premi **OK** i serà redirigit a la pantalla de permisos de l'aplicació [figura 14.12]. En la pantalla de permisos activi els permisos d'inici automàtic i desactivi l'optimització de bateria per a aquesta aplicació.

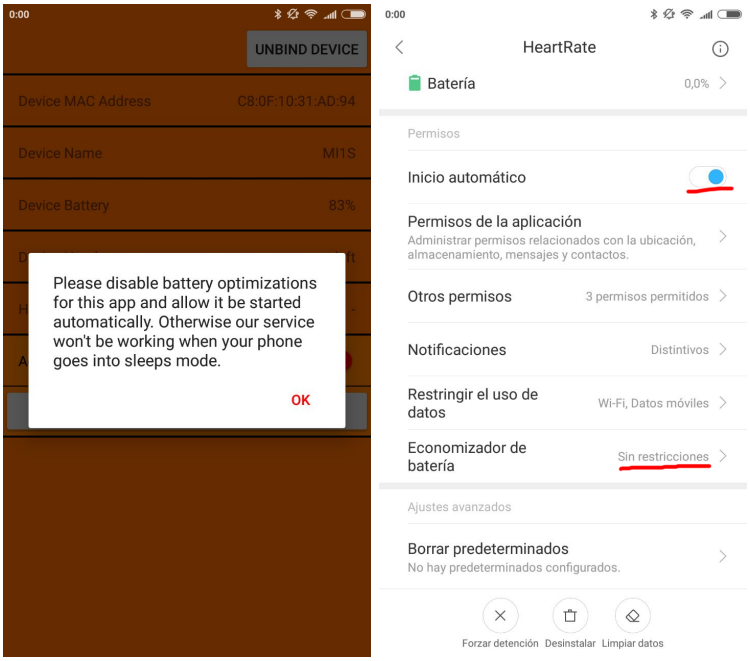

Figura 14.12: Permisos d'inici automàtic i optimitzador de bateria

Per tal de mantenir una bona precisió del càlcul és necessari que s'ajusti la polsera un dit per sota del canell sense estrènyer-la molt. També es recomana tancar l'aplicació a través de la pantalla principal [14.2.3].