AGENTS MÒBILS EN EMERGÈNCIES. iMABETT: TRIATGE ELECTRÒNIC EN iPHONE

Memòria del projecte de final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per Gueorgui Bojidarov Radev i dirigit per Ramon Martí Escalé.

Bellaterra, Juny de 2010

El firmant, Ramon Martí Escalé, professor del Departament d'Enginyeria de la Informació i de les Comunicacions de la Universitat Autònoma de Barcelona

CERTIFICA:

Que la present memòria ha sigut realitzada sota la seva direcció per Gueorgui Bojidarov Radev

Bellaterra, Juny de 2010

Firmat: Ramon Martí Escalé

A tots els que m'heu recolzat i m'heu donat cuport aquests anys.

Agraïments

M’agradaria donar les gràcies a totes les persones amb qui he tingut el plaer d’estar durant aquests cinc anys de carrera. A l’Isaac, a l’Oriol i especialment al Samuel amb qui tant bé ens ho hem passat, tant poc hem dormit, tant hem treballat i tants bons moments hem viscut.

També voldria agraïr als projectistes del SENDA, al departament i especialment al meu director de projecte, Ramon Martí, per la seva paciència i els consells que m’ha ofert per poder realitzar amb èxit aquest projecte.

I gràcies a la meva familia, que m’ha donat suport tots aquests anys i sobretot en els moments difícils en els que més ho he necessitat.

Índex

Índex de figures

Capítol 1

Introducció

En el món en el que vivim succeeixen, de forma inevitable, catàstrofes on la vida de molta gent corre perill. Un exemple seria el tsunami provinent de l'oceà Índic que a l'any 2004 es va emportar la vida de més de 300,000 persones. De fet s'estima que aquest tsunami s'endugué directa o indirectament la vida d'aproximadament un milió de persones. La pregunta que ens sorgeix inevitablement és: "Es podria haver fet alguna cosa per evitar-ho o per dismiuir la magnitud de la catàstrofe?".

En aquest tipus de catàstrofes és molt important millorar la nostra resposta a aquests esdeveniments i una manera de millorar-la seria optimitzar l'organització dels recursos d'emergència. La manera de gestionar aquests recursos és de vital importància, ja que d'això pot dependre la vida de milers de persones. Per tal de maximitzar el número de persones salvades es precisa que la distribució d'aquests recursos sigui eficient, ràpida i ben organitzada. Un mètode per aconseguir això és l'anomenat procés de triatge. Aquest consisteix en prioritzar i gestionar els pacients en funció de la seva gravetat. Durant aquest procés s'avalua l'estat de cada pacient i se li posa una etiqueta amb la prioritat d'assistència mèdica per tal de que posteriorment, quan hi hagi personal mèdic disponible es pugui atendre certs pacients segons la gravetat del seu estat de salut i les seves necessitats per realitzar l'assistència mèdica.

Capítol 2

Definició del projecte iMABETT

2.1 Objectiu principal del projecte

L'objectiu d'aquest projecte és desenvolupar un sistema iMABETT dissenyat per dispositius iPhone 3GS per a gestionar casos d'emergència a través d'agents mòbils. Aquest sistema ha de permetre que a cada pacient se li coloqui una etiqueta com les que s'utilitzen en el triatge manual però amb una etiqueta RFID que permeti identificar-lo posteriorment. Llavors des de l'iPhone es procediria a identificar quin tipus d'etiqueta li pertoca segons l'estat en el que es troba i aquesta informació junt amb les seves dades personals s'emmagatzemaria dintre d'un agent sobre plataforma JADE. L'objectiu és que en un futur els agents mòbils creats pel programa tinguin un behaviour (comportament) que els permeti integrar-se en infraestructures de tractament d'agents de triatge.

A més en el sistema hi haurà un lector d'etiquetes (tags del triatge) que seria un dispositiu que permeti llegir les etiquetes. Una altra cosa a tenir en compte serà la opció d'anotar en quin lloc es troba el pacient del qual s'estan prenent les dades, per això es farà ús del chip GPS de l'iPhone. Cal considerar que tot el personal que farà servir els dispositius interactuarà amb aquests de forma tàctil tant per escollir les opcions com per escriure-hi qualsevol cosa, per això cal fer el sistema intuïtiu i fàcil d'utilitzar.

Avui en dia hi ha certs terminals mòbils que estan guanyant força i pels que

s'estan produint cada cop més aplicacions. Un d'aquests dispositius és l'iPhone, però és realment un sistema obert que admet aplicacions de tot tipus i en diferents llenguatges? Un altre objectiu d'aquest projecte és el de comprovar, justament, en quina mesura Apple limita el desenvolupament de software per aquest dispositiu i quines solucions hi ha davant de les limitacions imposades.

Per tant, resumint, els objectius del projecte són:

- Desenvolupament del sistema iMABETT per a terminals iPhone 3GS per realitzar triatge electrònic.
- Integració d'un dispositiu lector RFID per la identificació dels pacients.
- Disseny i implementació d'una interfície intuïtiva i clara que permeti la seva utilització per a gent no experimentada.
- Integració en una plataforma d'agents JADE.
- Integració amb el mòdul GPS del dispositiu per enmagatzemar la localització dels pacients.
- Comprovar les facilitats i les limitacions en el desenvolupament d'aplicacions per iPhone.
- Testeig del sistema.

Per això es prendrà com a punt de partida el sistema MABETT (Mobile Agent Based Electronic Triage Tag) dissenyat pel NOKIA N810 en Java, el qual funciona sobre la plataforma JADE, al dispositiu iPhone 3GS per tal de que el personal d'assistència de catàstrofes pugui classificar als pacients. Aquest sistema ja està implementat pel dispositiu NOKIA N810, no obstant es vol ampliar el número de terminals incloent d'aquesta forma l'iPhone 3GS entre els terminals que poden utilitzar aquest sistema.

2.2 Motivacions del projecte

La motivació d'aquest projecte resideix en la voluntat d'ampliar el catàleg de dispositius amb els que es pot utilitzar el sistema MABETT. D'aquesta forma a part d'ampliar els possibles mòbils amb els que es pot fer el triatge es demostra la portabilitat del programa, la qual ve donada pel fet que està programat en Java.

A més hi ha un altre motiu per fer d'aquest projecte una recerca interessant, i aquest és comprovar la mal·leabilitat de l'iPhone i la facilitat d'adaptació de programes per aquest. A més, així es provarà en quina mesura l'iPhone es pot considerar un mòbil amb un sistema tancat i limitat per Apple.

2.3 Viabilitat del projecte

En aquesta part de la memòria prèvia s'analitzarà si el projecte és viable o no. L'anàlisi es dividirà en: viabilitat tècnica, viabilitat operativa, viabilitat econòmica, viabilitat legal, planificació temporal i finalment una breu conclusió.

2.3.1 Viabilitat tècnica

Pel desenvolupament del projecte es precisa que es coneguin els llenguatges de programació Java i C/C++. Per treballar amb el sistema MABETT s'ha d'adaptar la plataforma JADE, la qual està implementada en Java. Si les parts de la l'interfície gràfica i el GPS no es poden implementar també en Java s'hauran de programar en el llenguatge que utilitza l'iPhone, el qual és el Objective C, força semblant a C/C++. Per desenvolupar el projecte s'han de tenir coneixements sobre programació orientada a objectes i Mobile Agents a l'àmbit de la plataforma JADE.

El material necessari consta, per començar, d'un ordinador al despatx ofert pel departament i un portàtil Mac per tal de fer el projecte. A més es precisarà un iPhone per tal de fer el programa i testejar la seva correcta integració. Per acabar es faran servir etiquetes RFID incorporades a les etiquetes de triatge habituals i dispositius lectors de RFID per tal de que llegeixin les etiquetes RFID dels

pacients.

2.3.2 Viabilitat operativa

Tot i que en un principi s'havia especulat sobre la possibilitat de substituir el sistema manual actual pel sistema digital amb els dispositius mòbils i etiquetes RFID, finalment s'ha optat per incorporar les etiquetes RFID a les etiquetes de paper habituals. Això és degut a que poden sorgir situacions imprevistes que facin que el sistema no funcioni correctament i no podem oblidar que la vida de moltes persones pot estar en perill, un exemple de situacions perilloses seria que s'acabés la bateria dels dispositius o que hi hagués interferències en la senyal. Per aquesta raó durant la incorporació del sistema en el tractament de situacions d'emergència s'ha d'anar amb molt de compte i no es creu que la substitució d'un sistema per l'altre sigui una bona idea. El procediment que s'intentarà planificar serà la combinació dels dos sistemes de forma que s'obtinguin els avantatges d'ambdós i que si el nou sistema fallés es pogués seguir treballant amb el mètode de triatge tradicional.

2.3.3 Viabilitat econòmica

Del software que s'utilitzarà una part és gratuïta, una altra ve ja incorporada amb el sistema Mac i hi ha una part que ja ha estat adquirida per part del departament (SDK de l'iPhone). A l'inici del projecte ja es disposa de tot aquest software. L'iPhone 3GS ja ha estat adquirit pel departament i es podrà disposar en breu temps després de l'inici del projecte.

Des de l'inici del projecte es disposa tant de l'ordinador del despatx com del Mac per part del projectista. També es disposa d'etiquetes RFID, però en quant al dispositiu lector de les etiquetes s'està analitzant la possibilitat d'adquirir-ne un model nou i diferent del fet servir fins ara en aquest sistema (IDBlue).

2.3.4 Viabilitat legal

Tot i que en la Llei Orgànica 15/1999, del 13 de Desembre, de Protecció de Dades de Caràcter Personal (LOPD ([LOPD])) ni en el Reial Decret 1332/1994, del 20 de Juny, no existeix cap definició de dades mèdiques, se les considera dades especialment protegides, ja que formen part de la intimitat i privacitat de les persones.

El paràgraf 3 de l'article 7 de la LOPD expressa:
Los datos de carácter personal que hagan referencia al origen racial, a la salud y a la vida sexual sólo podrán ser recabados, tratados y cedidos cuando, por razones de interés general, así lo disponga una ley o el afectado consienta expresamente.

L'article 8 exposa:
Sin perjuicio de lo que se dispone en el artículo 11 respecto de la cesión, las instituciones y los centros sanitarios públicos y privados y los profesionales correspondientes podrán proceder al tratamiento de los datos de carácter personal relativos a la salud de las personas que a ellos acudan o hayan de ser tratados en los mismos, de acuerdo con lo dispuesto en la legislación estatal o autonómica sobre sanidad.

Per tant podem concloure que, com que els usuaris del sistema que prendran i tractaran aquestes dades és personal mèdic, no hi ha cap problema jurídic. No obstant, en investigacions futures seria adient afegir algun tipus de seguretat i/o encriptació al sistema per tal de que els agents no puguin ser interceptats per personal aliè al sistema.

2.3.5 Conclusió de la viabilitat

Com a conclusió de les diferents parts de l'estudi de viabilitat del projecte podem concloure que el projecte és viable. Per tant ara ja es pot començar a treballar en el projecte tot sabent que és un projecte viable i factible.

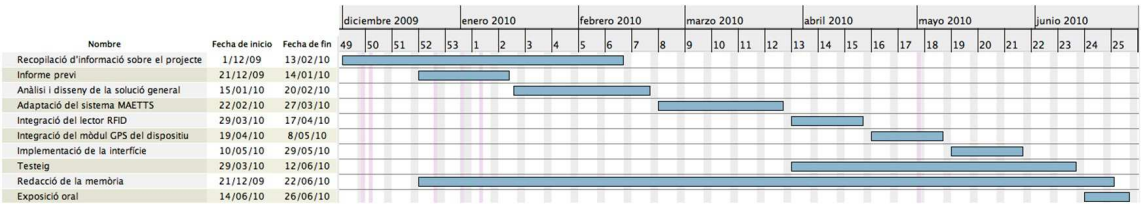


Figura 2.1: Diagrama de Gantt previst

2.3.6 Planificació del projecte

La recopilació d'informació va començar amb les reunions organitzades pel departament amb l'objectiu d'informar-nos sobre els diversos projectes oferts, això va ser el dia 1 de desembre del 2009. A partir de la primera reunió amb el tutor del projecte (el dia 21 de desembre) es va començar a desenvolupar l'informe previ i en quant s'acabi aquest informe es procedirà a analitzar la situació actual del sistema per tal de començar a dissenyar i implementar. Això inclou coses com investigar executar codi en Java a l'iPhone, com adaptar la plataforma Jade. Tan bon punt comenci el segon semestre es començarà el desenvolupament del projecte en sí i el límit per la seva finalització està concretat en el dia 22 de juny.

De totes formes es farà el possible per tenir acabat tant el projecte com la seva memòria per abans d'aquesta data per si sorgissin coses per canviar o afegir. Finalment, com que no es sap la data concreta de presentació del projecte, s'haurà de preparar el document de presentació pel primer possible dia de presentació, és a dir el 28 de juny. El procediment que es seguirà es pot veure al diagrama de Gantt de la figura 2.1 consistirà en començar analitzant i dissenyant en general com estarà format el programa i com es combinaran entre elles, a partir d'aquí procedirem a seguir els mateixos passos per cadascuna de les parts: anàlisi, disseny, implementació i prova. Les parts previstes per implementar són:

- Desenvolupament del sistema iMABBET: es definiran les especificacions d l'aplicació i s'implementarà.

- Integració del lector RFID: aquesta part permetrà a l'iPhone connectar-se amb el dispositiu lector d'etiquetes RFID.

- Integració del mòdul GPS: s'haurà de fer servir el chip GPS integrat amb l'iPhone per interactuar amb el sistema START.
- Interfície d'usuari: disseny d'una interfície gràfica simple i clara conservant totes les possibilitats del sistema original.

Capítol 3

Estat de l'art

L'aplicació que es desenvolupa en aquest projecte abarca diferents tecnologies i utilitza diferents llenguatges i elements d'aquests, per això és precís descriure i analitzar tots aquests elements per tal de que s'entengui la raó del seu ús a la vegada que es valora el seu potencial. El iMABETT està pensat per ser executat en un dispositiu iPhone 3GS, que utilitza el llenguatge Objective-C de forma nativa, i està previst que fasi un triatge seguint el protocol START. A més, permet utilitzar etiquetes RFID per a identificar als pacients i els situa amb l'ajuda del GPS per poder-los ubicar en el lloc de la catàstrofe. A continuació es descriuen tots aquests elements.

3.1 Triatge

El triatge ([TRIAGE]) és un mètode de la medicina d'emergències i desastres per la selecció i classificació ràpida de pacients en funció dels recursos mèdics dels que es disposa. Aquest mètode fou utilitzat per primera vegada per un metge cirurgià de Napoleó anomenat Dominique-Jean Larrey (1766-1842) per tractar als ferits en els camps de batalla on lluitaven els exèrcits de Napoleó.

En aplicar aquest mètode s'assigna a cada pacient un nivell de prioritat posant una etiqueta com la que es pot veure a la figura 3.1 a cada un amb el color de la prioritat que li correspon segons el seu estat. D'aquesta forma es pot seguir algun

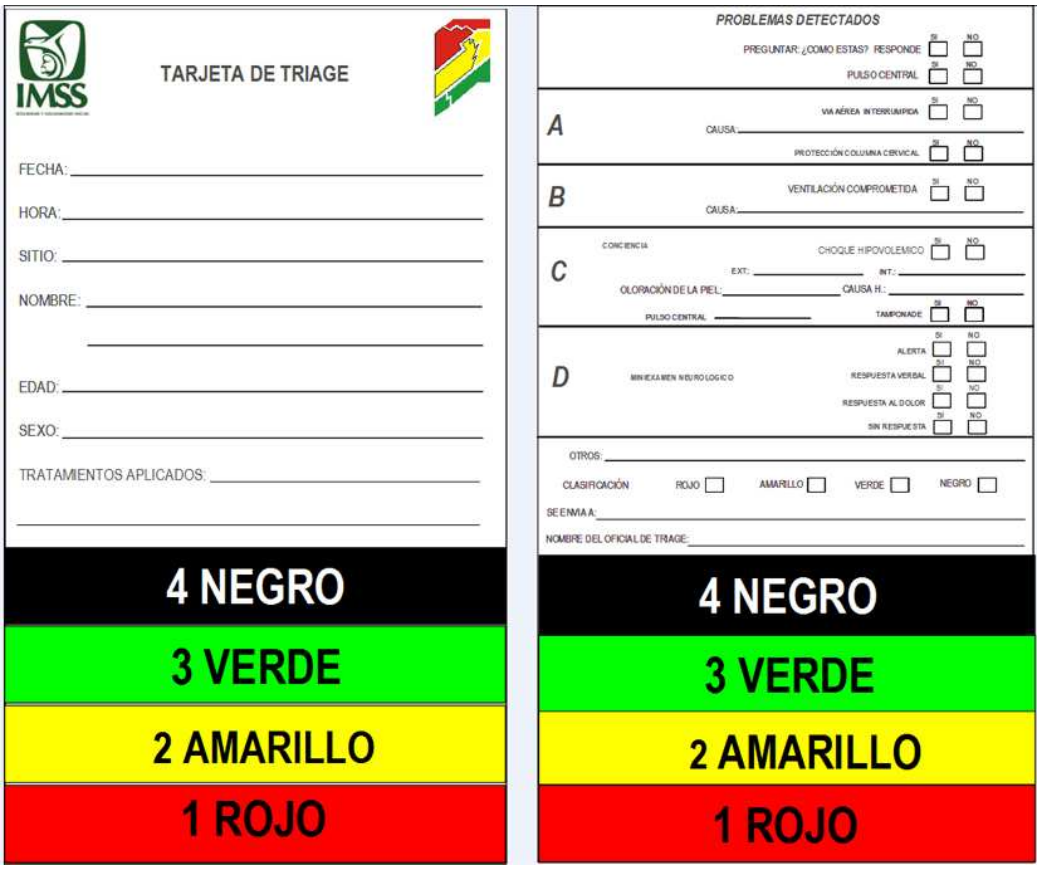


Figura 3.1: Exemple d'etiqueta de triatge manual

ordre de prioritats alhora de tractar els pacients en els casos en que és impossible tractar-los a tots de cop.

A més, en fer el triatge es poden aplicar diferents protocols i diferents ordres de prioritat. Per exemple, en situacions normals potser es dóna més prioritat als pacients més greus, mentre que en situacions de demanda massiva es donaria prioritat a les víctimes amb major possibilitat de supervivència.

3.1.1 START

Un protocol de triatge concret és el START (Simple Triage and Rapid Treatment). Aquest protocol fou dissenyat l'any 1983 per personal del Hoag Memorial Hospital i els bombers de Newport Beach (Califòrnia).

És anomenat simple perquè està orientat a personal amb poca experiència mèdica. Això el fa molt útil en ocasions d'emergència grans, ja que sigui perquè el personal mèdic no ha arribat o perquè no donen abast, el triatge pot ser efectuat per gent amb poca preparació mèdica, és a dir bombers, policies, bombers,...

En aquest protocol hi ha quatre tipus d'etiquetes pels pacients:

Verda (MINOR): el pacient presenta petites ferides però la seva atenció no és urgent.

Groga (DELAYED): el pacient necessita atenció però sense gaire urgència.

Vermella (IMMEDIATE): el pacient precisa atenció immediata ja que es troba en estat greu.

Negra (DECEASED): el pacient és mort i ja no s'hi pot fer res per salvar-lo.

Segons aquest protocol primer es miraria quins pacients poden caminar, és a dir que estan sans, per assignar-los etiquetes verdes. Llavors es procediria pacient per pacient amb els següents passos; si el pacient respira, si no respirés se li netegen les vies respiratòries i es fan les maniobres destinades a fer respirar el pacient, si llavors respira se li posa l'etiqueta vermella i si no, malauradament, és mort i no es pot fer res, per tant se li posaria una etiqueta negra. Si des d'un començament el pacient respirés des del començament es mesuraria amb quina freqüència respira, si fossin més de 30 respiracions per minut se li posaria l'etiqueta vermella. Si les respiracions fossin menys de 30 per minut es miraria si se li pot trobar el pols, si no se li trobés el pols i en pressionar un moment els dits tardessin més de 2 segons en recuperar el seu color natural se li posaria una etiqueta vermella. Si pel contrari els dits tardessin menys de 2 segons en recuperar el color natural o si se li hagués trobat el pols des d'un principi se li comprova l'estat mental mirant si el pacient pot seguir instruccions senzilles, si aquest fos el cas se li posaria una etiqueta groga i si no una de vermella. Tot aquest procés es troba resumit a la figura 3.2.

Hi ha diversos sistemes que fan el triatge de forma electrònica, aquests sistemes són anomenats ETT (Electronic Triage Tag), alguns exemples són WIIS-

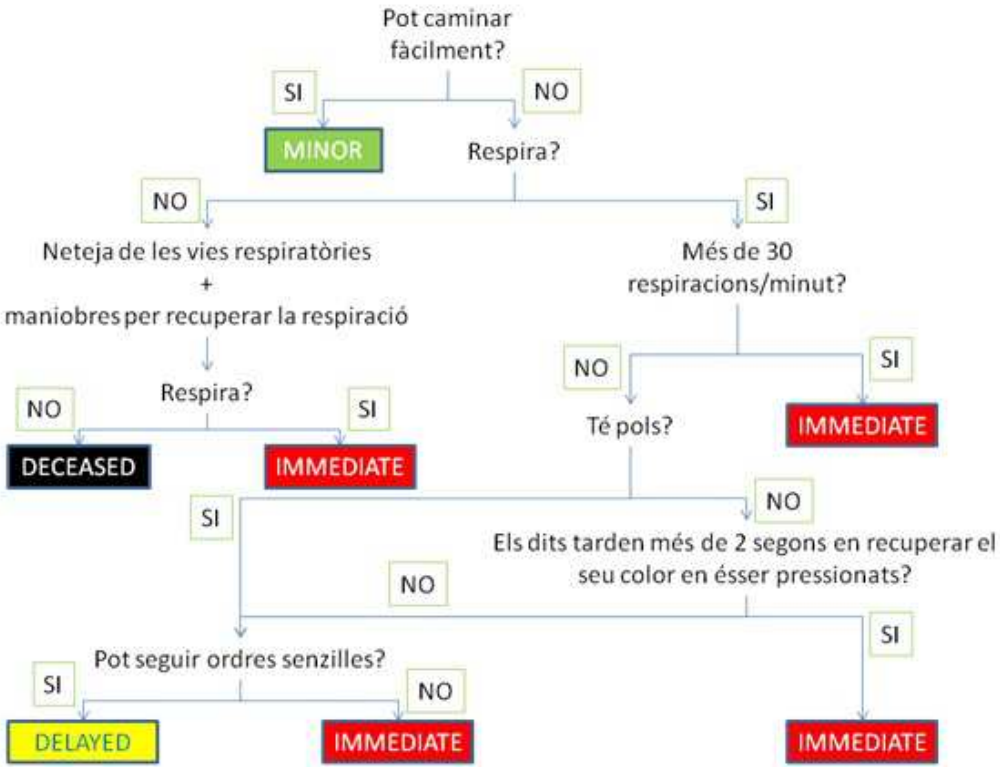


Figura 3.2: Arbre de decisions del protocol START

ARD, ARTEMIS i MABETT. El sistema Wiisard([Wiisard]) fa servir triatge electrònic en situacions d'emergència a través d'una PDA amb memòria no volàtil la qual es conecta via wireless IEEE 802.11. A la PDA no només es guarda la informació del triatge sinó també altra informació rellevant del pacient. ARTEMIS([ARTEMIS]) és un projecte de recerca que pretén desenvolupar un sistema integrat que permeti el triatge remot i automàtic i tractament d'informació d'emergències. Fa servir una PDA amb sensors connectats als pacients.

Per aquest projecte es farà servir el sistema anomenat MABETT (Mobile Agent Electronic Triage Tag System) ([MABETT]), el qual fa servir el protocol START i agents mòbils per tal de que en situacions de catàstrofe s'apliqui el protocol START i mitjançant agents mòbils es transmet informació dels pacients que encara es troben en el lloc de la catàstrofe a l'hospital on seran atesos. El procediment consisteix en que en un dispositiu mòbil es crea un agent mòbil destinat a recollir dades del lloc de la catàstrofe quan es troba amb algun pacient amb etiqueta; aquest agent s'envia a l'hospital, on transmet les dades a un servidor que gestionarà la informació ja sigui per reservar sang dels grups sanguinis dels ferits o bé per avisar de la quantitat de pacients que encara es troben al lloc de la catàstrofe per exemple.

3.2 RFID

El RFID (Radio Frequency IDentification) ([RFID]) és un sistema d'emmagatzematge i recuperació de dades en uns dispositius anomenats etiquetes o tags RFID mitjançant, tal i com el seu nom indica, radio freqüència. Aquestes etiquetes són llegides amb lectors RFID. Tot i que el sistema té algunes similituds amb els codis de barres típics de les botigues entre els seus avantatges es troba el fet que el lector no cal que enfoqui amb cap làser a l'etiqueta.

Un altre dels avantatges dels tags RFID és que poden emmagatzemar molta informació, mentre que el codi de barres ofereix només un identificador, és a dir que mentre els codis de barres permeten identificar un tipus d'element les etiquetes RFID permeten identificar cada element individual dels diferents tipus

permetent així una identificació més concreta. Per altra banda la tecnologia RFID permet llegir diverses etiquetes a la vegada mentre el codi de barres permet llegir només una a cada instant. Per últim les etiquetes RFID permeten que la informació emmagatzemada sigui sobreescrita i són molt més resistents que els codis de barres, els quals es poden trencar o embrutar fàcilment esdevenint il·legibles i no es poden sobreescriure. Les etiquetes estan formades per dues parts: un circuit integrat que emmagatzema informació i una antena que rep i transmet els senyals.

3.3 elements de hardware

A continuació es descriuran en termes generals els elements de hardware que estan involucrats en el projecte.

3.3.1 iPhone 3GS

L'iPhone 3GS ([IPHONE3GS]) és un telèfon inteligent i multimedia de la marca Apple que funciona amb el sistema operatiu iOS ([IOS]) (fet servir pels dispositius iPod, iPhone i iPad) i està basat en UNIX. Té connexió a internet, pantalla tàctil amb tecnologia multitàctil, càmara, Bluetooth i GPS. No disposa de teclat físic però per escriure es fa servir un teclat virtual que apareix quan es precisa introduir text. El primer iPhone va sortir al mercat l'any 2007 i el model 3GS és la tercera generació d'aquest mòbil, el qual va sortir al mercat al 2009. Actualment ja està a punt de sortir al mercat la quarta generació (iPhone 4) amb una nova versió del sistema operatiu.

Característiques hardware

- Mida: 115,5 mm X 62,1 mm X 12,3 mm.
- Pes: 135 gr.
- Xarxes disponibles.
  - UMTS/HSDPA (850, 1.900, 2.100 MHz).

Figura 3.3: iPhone 3GS

  - GSM/EDGE (850, 900, 1.800, 1.900 MHz).
  - Wi-Fi (802.11b/g).
  - Bluetooth 2.1 + EDR.
- A-GPS (Sistema de Posicionament Global Assistit) + brúixola digital.
- Pantalla tàctil amb tecnologia multi tàctil de 3,5 polzades.
  - Resolució de 480 X 320 píxels.
- Autonomia:
  - Fins a 9 hores utilitzant wi-fi.
  - Fins a 300 hores en repòs.
- Micròfon i altaveus.
- Connector per connectar a PC o MAC via USB.
- No disposa de targeta de memòria.
- Memòria de 16 GB o 32 GB (l'iPhone utilitzat disposa de 16 GB).

- Càmera de 3,2 mpx.
- CPU: Samsung ARM Cortex A8 (600 MHz).
- Botons i pestanyes físics:
  - Botó d'encendre/apagar/bloquejar.
  - Botons de pujar/baixar volum.
  - Botó inici.
  - Pestanya so/silenci.

Característiques software

- Sistema operatiu iOS versió 3.1.2
- Mono tasca (no es permet que funcionin dos o més programes a la vegada)
- Els programes s'instalen mitjançant el programa Itunes amb connexió a l'Apple Store
- No es permet instal·lar oficialment programes que no compten amb la supervisió d'Apple
- No suporta l'execució de programes fets en Java ni hi ha cap paquet oficial per executar codi en Java
- Els programes fets per l'iPhone han d'estar programats en Objective-C, llenguatge que conforma un superconjunt de C
- El sistema operatiu està dividit en quatre capes:
  - Cocoa touch: capa que conté els frameworks utilitzats més sovint pels desenvolupadors
  - Media Services: capa que proveu a l'iPhone d'àudio, vídeo, animació i gràfics.

– Core Services: aquesta capa compren la base sobre la que les dues capes anteriors estan fetes.

– Core OS: capa que es relaciona amb el hardware i que conté els serveis de més baix nivell, com ara maneig de memòria, tractament dels arxius del sistema i threads.

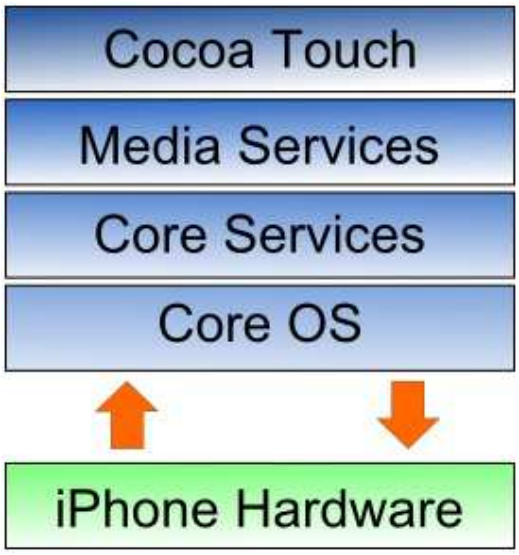


Figura 3.4: Estructura de la plataforma de desenvolupament per iPhone

Instal·lació de paquets no oficials

Per tal d'instal·lar programes i paquets no oficials (no aprovats per Apple) en general a l'iPhone cal fer un procés via software anomenat Jailbreak ([JAILBREAK]). Posteriorment es poden instal·lar en els dispositius instal·ladors de software no oficial com ara Icy, Cydia i Installer entre altres. S'estima que un 8-9% dels productes d'Apple estan jailbreakejats i aquesta pràctica no es considera il·legal.

El primer Jailbreak data del 10 de juliol del 2007, quan es va aconseguir posar com a to de trucada qualsevol música guardada a un iPhone. Més tard es va començar a desenvolupar jocs no aprovats per Apple, programes en general, intèrprets i compiladors de llenguatges no soportats pels dispositius,...

Actualment és possible jailbreakejar qualsevol dispositiu excepte l'iPhone 4 i els dispositius que tenen el sistema operatiu iOS 4. I, tot i que quan es modifica el software d'un dels dispositius d'aquesta forma es perd la garantia, el programa Itunes (oficial d'Apple) permet restaurar el sistema operatiu original i tenir el dispositiu com recent estrenat.

Alguns exemples de paquets que ens seran útils són:

Jikes : aplicació que permet compilar i executar codi en java a l'iPhone

OpenSSH : paquet que permet connectar-se via SSH a l'iPhone

MobileTerminal : aplicació que permet obrir una o diverses consoles per executar comandes UNIX

3.3.2 Etiquetes RFID

Segons l'aplicació on es vulgui utilitzar la tecnologia RFID es pot escollir entre diversos tipus d'etiquetes segons quina sigui més adequada pel cas particular.

- Etiquetes actives Aquestes etiquetes tenen la seva pròpia font d'alimentació que proveu d'energia al circuit integrat i l'antena incorporada. Per aquesta raó l'abast de la senyal d'aquestes etiquetes augmenta considerablement fins a uns centenars de metres, a la vegada que també augmenta la seva mida i preu. Això si, s'ha d'anar en compte per si s'acaba l'energia de la font d'alimentació, tot i que degut a la baixa necessitat energètica això tarda força a passar.

- Etiquetes semi-passives Les etiquetes semi-passives són aquelles que tenen font d'alimentació, tot i que només la utilitzen pel circuit integrat, per tant les antenes d'aquestes etiquetes tenen menys abast que les de les etiquetes actives. Gràcies a l'alimentació del circuit integrat aquestes etiquetes permeten tenir una màquina d'estats interna per oferir respostes basades en l'estat anterior de l'etiqueta. A més aquestes etiquetes tenen una fiabilitat similar a les etiquetes actives mentre tenen un temps de vida superior ja que es redueix el consum energètic.

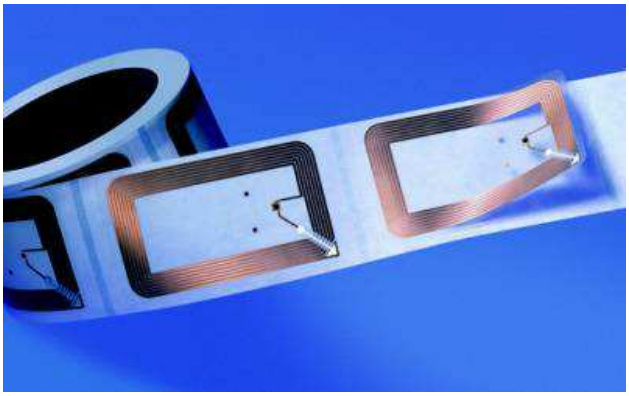


Figura 3.5: Etiqueta RFID

- Etiquetes passives Es tracta d'etiquetes que no necessiten alimentació interna i que proveeixen al circuit de l'alimentació precisada induint corrent elèctric amb l'antena a partir de les senyals rebudes del lector RFID.

És per aquests avantatges respecte a altres sistemes com el codi de barres que s'ha decidit utilitzar aquesta tecnologia per la identificació dels pacients.

3.3.3 GPS

El GPS (Global Positioning System) ([GPS]) és un sistema global de navegació per satèl·lit que permet situar-nos a escala mundial. El seu origen és, com tants altres avenços tecnològics, una investigació del departament de Defensa dels Estats Units d'Amèrica i data de l'any 1973.

Aquest sistema consta de dues parts: els satèl·lits i els dispositius receptors GPS. A l'espai, al voltant de la Terra, hi ha entre 24 i 32 satèl·lits destinats a oferir aquest servei a 20.200 km de la Terra amb òrbites sincronitzades de 11h58m de període per tal de cobrir tota la superfície de la Terra. A la Terra hi ha un dispositiu receptor GPS per localitzar cada objecte, persona o vehicle que es vulgui saber on és. Amb els receptors es pot saber la latitud, la longitud, l'altura i l'hora on es troba quan rep informació de com a mínim 4 satèl·lits mitjançant triangulació.

Actualment els receptors GPS militars tenen una precisió prou gran com perquè els errors es puguin comptar en centímetres, mentre que els receptors d'ús quotidià

dels que disposem als cotxes i als mòbils tenen un error d'uns pocs metres.

Els receptors reben concretament un conjunt de valors anomenat almanac que inclou la informació útil per realitzar la localització, la qual està inclosa en un paquet anomenat efemèrides. Les efemèrides inclouen informació com l'hora, dia, estat de funcionament del satèl·lit, la seva posició a l'espai,... A partir d'aquesta informació el receptor i utilitzant el rellotge intern, el receptor calcula la distància a la que es troba respecte a cada satèl·lit i d'aquesta forma pot saber on es troba relativament a ells. Si a més li sumem la informació de la posició concreta d'aquests satèl·lits el receptor pot saber la posició absoluta on es troba.

3.4 Elements de software

Tot seguit s'exposen els elements de software que tenen relació amb el projecte.

3.4.1 Agents

Hi ha diverses definicions del que és un agent i la seva definició. En el Maes, Pattie (1995) se'l defineix com: "Els agents autònoms són sistemes computacionals que s'executen en plataformes complexes i dinàmiques, perceben i actuen autònomament en els entorns i a través d'això acompleixen una sèrie d'objectius o tasques per les que han estat dissenyats". A partir d'aquesta definició, una de les més exteses, en podem extreure les característiques d'aquests softwares i a més se'n poden afegir algunes que no sempre hi són presents:

- Goal driven: els agents tenen un objectiu que tenen que cumplir durant la seva execució i pel que han estat dissenyats (recopilació d'informació, distribució de notícies,...).

- Comportament: cada agent té el seu comportament (behaviour) que marca les decisions a prendre durant la seva execució.

- Reactivitat: els agents reaccionen als canvis produïts en la plataforma on s'estan executant.

- Pro-activitat: els agents no només actuen en resposta a esdeveniments, són capaços d'exhibir decisions enfocades a acomplir els seus objectius.

- Autonomia: cada agent es pot executar de forma completament autònoma, sense necessitar d'interacció amb l'usuari.

- Habilitat social: els agents són capaços d'intercanviar informació entre ells si s'escau per complir els seus objectius.

- Mobilitat: alguns agents són capaços de moure's entre plataformes i fins i tot entre diferents dispositius.

3.4.2 MABETT

El MABETT (Mobile Agent Based Electronic Triage Tag) es una implementació de MAETTS (Mobile Agent Electronic Triage Tag System) ([MAETTS]) que es va dissenyar l'any 2008 en el projecte de fi de carrera per Xavier Jurado. Aquest programa fou dissenyat en Java i serveix per fer triatge electrònic mitjançant un dispositiu mòbil. El dispositiu pel que va ser dissenyat concretament és el NOKIA N810 i la interfície, que va ser dissenyada fent servir la llibreria SWT per Java, es pot veure a la figura 3.6.

Aquest programa permet fer triatge seguint el protocol START i emmagatzemar les etiquetes de triatge creades en agents sobre una plataforma JADE. L'objectiu del programa era oferir quatre funcionalitats principals:

1. Fer etiquetes seguint un arbre de decisió binari i emmagatzemant-les en agents mòbils

2. Consultar les etiquetes creades

3. Configurar un compte enrere per implementar un TTR (Time To Return) pels agents

4. Guardar informació de la persona que realitza el triatge

Figura 3.6: Vista de la interficie del MABETT

El TTR consisteix en que es guardi el temps que queda per tal de que la persona que porta el dispositiu vagi a un hospital i permeti que els agents mòbils migrin cap als servidors de l'hospital, d'aquesta forma es volia fer que els agents amb les etiquetes anessin cap al dispositiu de la persona que més aviat podria permetre'ls arribar a un hospital. Al final del projecte es va aconseguir crear agents que no eren mòbils i tot i que el compte enrere es va implementar la funcionalitat del TTR no va ser possible. Encara que fer que els agents siguin mòbils no té una complexitat molt gran, per tal de que tingui sentit s'hauria d'acabar de dissenyar una infraestructura on hi hagi inclosos els servidor que en un principi s'haurien de trobar als hospitals per tal de rebre als agents i mostrar les dades al personal d'aquests.

Els principals components del MABETT són:

- Mòdul lector RFID: dispositiu que s'utilitza per llegir les etiquetes RFID
- Mòdul GPS: component que ens permet realitzar la localització
- Agent interfície gràfica: agent que s'encarrega de permetre la interacció amb l'usuari a través de la pantalla sense precisar coneixement previ del

mètode de triatge START

- Agent d'emmagatzematge: agent que guarda les dades d'una etiqueta creada mitjançant el programa i el mètode START

D'aquests components els dos agents es mantindran mentre que el GPS es substituirà pel que té l'iPhone i el lector RFID s'haurà d'analitzar més endavant si es pot aprofitar el mateix.

A continuació es mostren els diferents paquets en els que està subdividida l'aplicació:

- mabett: l'arrel del projecte, contindrà les classes destinades a iniciar l'aplicació.
  - mabett.core: conté les classes de control.
  - mabett.devices: conté les classes corresponents als mòduls hardware de l'aplicació.
    - mabett.devices.bt: pel mòdul de lectura de l'RFID.
    - mabett.devices.gps: en el cas del mòdul de localització GPS.
  - mabett.gui: conté totes les classes de la interfície gràfica.
    - mabett.gui.resources: conté els recursos de la interfície (com per exemple les imatges) i les classes que els gestionen.
    - mabett.gui.triage: per totes les classes de la interfície destinades a realitzar el procés de triatge.
    - mabett.gui.widgets: conté les classes que implementin controls gràfics personalitzats.
  - mabett.triage: en formen part totes les classes de l'agent d'emmagatzematge.
  - org.tzi.rfid.jidblue: el controlador del lector RFID del projecte jIDBlue (és el nom original del paquet).

3.4.3 Java

Java ([JAVA]) és un llenguatge de programació orientat a objectes que va sorgir a principis dels 90 essent desenvolupat per Sun Microsystems. Es tracta d'un llenguatge molt extès i molt utilitzat, en part gràcies al fet que és un llenguatge independent de la plataforma, és a dir que si fem un programa en Java aquest programa podrà ser executat en qualsevol plataforma que corri Java. Això si, el Java que corri ha de tenir les llibreries que s'utilitzen en el programa. Això s'aconsegueix gràcies al fet que s'han fet màquines virtuals per molts sistemes que fa de pont entre el sistema operatiu i el programa en Java.

Inicialment la idea era fer córrer Java en qualsevol electrodomèstic però es va desestimar, no obstant es va decidir utilitzar la idea però a l'àmbit d'Internet. En conseqüència es va aconseguir que s'executessin programes dintre d'una pàgina web. La implementació original del compilador, la màquina virtual i les biblioteques de classes de Java van ser desenvolupats el 1995 i des de llavors Sun ha anat controlant el seu desenvolupament i evolució a través del Java Community Process.

En el llançament de Java (1996) se li va anomenar JDK però al cap de dos anys es va dividir en tres versions amb la finalitat de cobrir els diferents entorns d'aplicació:

J2SE (Java 2 Platform Standard Edition) per entorns de mitjana gama i estacions de treball. A més, aquí es trobaria l'usuari normal d'un PC.

J2EE (Java 2 Platform Enterprise Edition) per entorns distribuïts empresarials o d'Internet.

J2ME (Java 2 Platform Micro Edition) per entorns amb recursos limitats com ara telèfons mòbils, PDAs,...

D'altra banda hi ha moltes altres màquines virtuals que permeten l'execució de codi Java tot i no ser programades per Sun Microsystems. Hi ha algunes que són lliures i altres que no, per exemple entre les nolliures trobem Azul VM, JBed i OJVM. I entre les implementacions lliures podem trobar Jamvm, JESSICA, Kaffe i TinyVM.

El JamVM és una màquina virtual de Java de codi lliure desenvolupada amb la finalitat de crear una màquina virtual de Java extremadament petita comparada amb altres. Suporta Java Native Interface (JNI) que serveix per fer crides de codi en altres llenguatges i viceversa. A més suporta els següents processadors: AMD64, ARM (el de l'iPhone), i80486, MIPS i PowerPC.

JADE

JADE ([JADE]) és una plataforma completament implementada en Java pel desenvolupament d'agents. Jade compleix l'estàndard FIPA i exigeix tenir almenys la versió 1.4 de Java. Es tracta d'una plataforma de codi lliure i distribuïda per Telecom Italia. Concretament permet al programador:

- Crear agents.
- Programar behaviours (comportaments) dels agents.
- Comunicació ACL-FIPA.
- Pot executar-se en diferents màquines virtuals de Java i els agents es poden executar concurrentment i intercanviar-se missatges.
- Els agents s'organitzen en contenidors: 1 de general i n no principals i connectats al principal.

SWT

SWT (Standard Widget Toolkit) ([SWT]) és una API gràfica de codi lliure per la plataforma Java. En els seus orígens va ser creat per IBM, però ara està sent mantinguda per Eclipse Foundation i Eclipse IDE. És una alternativa a AWT i Swing, les quals són desenvolupades per Sun Microsystems com a part de J2SE. Per mostrar la interfície el SWT accedeix a les llibreries GNU natives del sistema operatiu utilitzant JNI, tot i que les aplicacions amb interfície SWT són portables les implementacions del SWT són diferents per cada plataforma (perquè els accessos a les llibreries del sistema operatiu seran diferents).

Com que el SWT permet la portabilitat dels programes els programes tindran les mateixes funcionalitats en les diferents plataformes, tot i que tindran diferents aparences ja que utilitza el model de interfície típica del sistema operatiu on s'estigui executant. Com a avantatges respecte a les alternatives és més ràpid i consumeix menys.

3.4.4 Objective-C

L'Objective-C ([OBJC]) és un llenguatge de programació orientat a objectes creat el 1980 per la corporació StepStone. El 1992 va ser alliberat i actualment és utilitzat per la programació per GNUstep, Mac OS i iOS. L'objective-c és considerat una extensió potent del ANSI C i, per tant, constitueix un superconjunt de C.

Per compilar programes fets en aquest llenguatge podem utilitzar el compilador gcc de GNU en qualsevol sistema operatiu i en Mac OS X es pot utilitzar a part del gcc que ve de sèrie amb el sistema operatiu, el Xcode. Xcode és l'eina bàsica per programar per iPhone, iPod i iPad emprant Objective-C i la llibreria COCOA.

Pel fet de ser una extensió de C compatible enrere, l'Objective-C ha heretat moltes característiques de la sintaxis:

- Sentencies de control de flux.
- Tipus de dades fonamentals, estructures i punters.
- Conversions explícites i implícites entre tipus.
- Àmbits de les variables: globals, estàtiques i locals.
- Funcions i sintaxis.

Els arxius en aquest llenguatge porten l'extensió .m i en aquests es pot fer servir tant codi en C com en Objective-c.

COCOA

COCOA ([COCOA]) és un entorn de programació orientat a objectes que permet el desenvolupament d'aplicacions natives per Mac OS X i iOS. És una de les 5 Apis principals de Mac OS X, entre les quals es troben Carbon, POSIX, X11 i Java.

Normalment les aplicacions que utilitzen COCOA són desenvolupades amb el Xcode i el Interface Builder (per la GUI). De totes formes COCOA pot ser emprat usant altres llenguatges com ara Python, Perl i Ruby, sempre i quan s'utilitzin les eines de bridging adequades. COCOA està format per dos frameworks principals que utilitzen el prefix NS (NeXTSTEP) en totes les seves classes:

Foundation Kit : permet manipular strings, utilitzar contenidors i iteracions, computació distribuïda, fer loops i altres funcionalitats que no estan lligades a la interfície

Application Kit : inclou el codi que permet crear i interactuar amb la interfície

Capítol 4

Anàlisi

Per tal d'orientar el nostre projecte i que al final es pugui concloure que acaba amb èxit és precís marcar una sèrie de requeriments funcionals i no funcionals que s'han d'acabar complint. En aquest capítol s'exposen tots aquests requisits.

4.1 Requeriments funcionals

Els requeriments funcionals són els requisits que defineixen el comportament intern del programa/software. En el cas d'aquest projecte són els següents:

- El sistema ha de tenir una interfície gràfica que permeti a l'usuari realitzar el triatge electrònic.
- L'usuari no té la necessitat de conèixer l'arbre de decisions que compren el protocol START.
- En crear una etiqueta de triatge l'usuari ha de poder afegir informació que consideri rellevant sobre el pacient.
- S'ha de poder consultar la ubicació del pacient i guardar-la en l'agent per saber posteriorment on es troba.
- S'han de poder consultar les etiquetes colůlocades als pacients per tal d'identificar quines dades pertanyen a cada un.

- Per cada pacient s'ha de crear una etiqueta i per cada etiqueta creada s'han d'emmagatzemar les dades en un agent.
- S'ha de poder desfer qualsevol pas durant el procés de triatge per si hi ha hagut algun error en introduir les dades o en emprar la interfície gràfica.
- S'han de poder emmagatzemar les dades de l'usuari per incloure la seva identificació en els agents creats.
- S'han de poder emmagatzemar els tags RFID de les etiquetes creades així com el tipus d'etiqueta segons la prioritat d'atenció que precisi.

4.2 Requeriments no funcionals

Els requeriments no funcionals són tots aquells requisits que no descriuen les funcions a realitzar, però si coses que s'han d'intentar acomplir en l'operació del sistema. Els requeriments no funcionals d'aquest projecte són els següents:

- La interfície ha de ser tant intuïtiva com es pugui.
- La interfície haurà de ser accessible de forma tàctil.
- Com que els usuaris pels que està destinada l'aplicació té coneixements mínims mèdics el temps d'aprenentatge hauria de ser molt curt.
- El triatge d'una persona s'ha de poder completar ràpidament (1 minut) per tal de poder maximitzar el número de pacients als que se'ls hi fa triatge.
- S'ha d'intentar minimitzar els recursos de còmput del sistema ja que es tracta d'un dispositiu mòbil.
- S'ha de poder crear etiquetes fins i tot quan no es disposa del tag RFID o de dades GPS.
- S'ha d'intentar que el sistema romangui sent tant portable com es pugui per utilitzar-lo en altres dispositius.

Capítol 5

Disseny i implementació

En aquest apartat s'expliquen els diferents dissenys i implementacions que s'han contemplat pel desenvolupament d'aquest projecte, així com les decisions preses i les raons d'aquestes. Tot i que el disseny ha anat canviant durant el transcurs d'aquest projecte s'ha procurat mantenir els objectius i en la mesura del possible les funcionalitats i el rendiment d'aquest.

5.1 Disseny

En un principi, en contemplar la situació inicial, es veia evident que ja que l'aplicació havia estat dissenyada en Java per permetre la portabilitat i com que la plataforma Jade està implementada per Java s'hauria d'intentar executar-la des de l'iPhone amb els mínims canvis possibles. D'aquesta forma, a part d'aconseguir fer una portabilitat simple, es demostraria el nivell de portabilitat de l'aplicació original. A més, en seguir utilitzant l'API SWT s'aconseguiria una interfície bastant similar, tot i que variaria lleugerament en ser diferent el sistema operatiu.

D'entrada el principal problema és que en un iPhone no es pot executar programes Java per defecte, i evidentment tampoc es poden compilar. Per això s'ha hagut de buscar opcions no oficials d'Apple ([JAVAIPHONE]). A més, s'ha hagut de trobar alguna implementació del SWT per iOS, ja que es necessita per visualitzar la interfície en el sistema operatiu de l'iPhone (iOS).

Respecte a la part de connexió per bluetooth amb el lector RFID i la localització GPS ja s'observarà com implementar-les en quant s'aconsegueixi executar l'aplicació.

5.2 Implementació

El primer objectiu era aconseguir que es pogués executar codi Java en l'iPhone i com que no hi havia (i encara ara no hi ha) cap paquet oficial ofert per Apple, es va haver de procedir amb el procés d'alliberar-lo (Jailbreak) per tal de poder executar codi propi. Tot seguit es van buscar per Internet les diferents solucions per executar codi Java a l'iPhone.

La primera que va sorgir va ser la d'instal·lar una màquina virtual anomenada JamVM que va ser adaptada per iPhone per Jay Freeman, l'autor de Cydia (un dels instal·ladors d'aplicacions i paquets no oficials més populars). L'adaptació es va fer l'any 2007 i des de llavors es pot executar codi Java en un iPhone via terminal i sense GUI. Fins ara no hi ha hagut cap actualització i això és un problema, ja que si bé el Java ha anat evolucionant i ara es troba en la versió 6 update 20 (s'espera el llançament de Java SE 7 i la versió actual del llenguatge és la per finals d'aquest any) i el JamVM conseqüentment ha anat evolucionant (la versió actual és la 1.5.4), la versió del JamVM que està adaptada per l'iPhone és la 1.5.1b2-3 (la qual inclou el Java 1.5.0). Això va donar que pensar que potser hi hauria problemes en quant a la versió del Java i per aquest motiu es va decidir buscar altres mètodes d'executar el programa.

Una altra solució que va sorgir va ser la de utilitzar un traductor anomenat alcheMO desenvolupat per l'empresa Innaworks, el qual transforma programes en Java a C++ (suportat per l'iPhone). Aquest software permet convertir aplicacions de J2ME en aplicacions natives de l'iPhone. El problema amb aquest mètode és que aquest projecte funciona sobre JADE i un traductor de codi no podrà traduir totes les llibreries d'aquesta plataforma i a més fer-ho compatible posteriorment per enviar agents a les altres plataformes amb JADE.

Finalment també es va trobar que Sun Microsystems s'havia proposat a l'any

2008 de fer una màquina virtual de Java per l'iPhone, però degut a que Apple no desitja que s'executi codi en llenguatges fora de l'Objective-C no permet aquest tipus d'aplicacions i des de llavors no se'n sap res sobre aquest projecte.

En conclusió, la opció més viable era la d'utilitzar el JamVM junt amb alguns altres paquets i intentar arrancar el programa original sense les parts de Bluetooth i GPS.

En concret es van instal·lar els següents paquets:

- Classpath: GNU Classpath necessari pel Java.
- JamVM: màquina virtual de Java.
- Jikes: compilador de Java.
- OpenSSH: permet connectar-se a l'iPhone via SSH per copiar arxius.
- MobileTerminal: aplicació que ens permet obrir una terminal a l'iPhone per a executar comandes UNIX i treballar amb els programes en Java.

En resum la idea és compilar l'aplicació original MABETT via Eclipse en un ordinador i crear un JAR executable configurant com a main class: BootAgent.class. Aleshores s'hauria de fer una connexió via SSH com a root a l'iPhone per transferir aquest JAR. I finalment s'hauria d'obrir una terminal a l'iPhone, anar a la carpeta on s'ha copiat l'aplicació i executar-la amb la comanda:

```
$ java -jar MABETT.jar
```

Per la compilació es va actualitzar el SWT a la versió més actual (3.5.2) i es va crear el JAR. Per comprovar que tot funcionava correctament es va arrancar el JAR en un Mac OS X 10.6.4 i va donar alguns problemes. Fins que es va trobar per Internet que per executar programes que utilitzen SWT en Mac OS X cal fer la crida de la següent forma:

```
$ java -jar MABETT.jar -XstartOnFirstThread
```

Realitzant la crida d'aquesta forma tot el programa funcionava bé en el Mac. Però en fer tot el procés i intentar arrancar el programa des de l'iPhone donava un error que mencionava el SWTResourceManager i java.lang.reflect.Method.invokeNative. Per la qual cosa es va deduir que era problema del SWT, la versió amb la que estava compilat el JAR no era compatible amb el iOS. Acte seguit es va buscar per Internet i no es va trobar cap implementació ni del SWT ni de les altres llibreries similars (AWT, Swing,...).

Amb això es va concloure que aquest disseny i implementació no eren viables i que s'hauria de buscar alguna solució diferent. El que era segur era el fet que la part de la interfície no es podia fer en Java, en canvi la part de l'execució de la plataforma JADE no es podia estar segur si funcionaria.

Potser es podria partir el programa i que la part de Java s'encarregués de JADE i els agents i un altre programa s'encarregués de interactuar amb l'usuari...

5.3 Redisseny

En base a que el primer disseny no va funcionar es va haver de buscar altres solucions i com que les altres opcions inicials (alcheMO i JVM de Sun Microsystems) eren inviables el millor era trobar alguna solució similar al disseny inicial. En ser la part d'interfície gràfica la que no funcionava es va decidir que s'hauria d'implementar utilitzant COCOA i en Objective-C. D'aquesta forma la part de la interfície gràfica seria nativa per iPhone i en ser inviable portar la part de JADE a Objective-C es va decidir de mantenir-la en Java. D'aquesta forma tenim dos programes que units fan la funcionalitat del MABETT. Per aquesta raó es va decidir utilitzar el paradigma de sistema Client-Servidor estant ambdós a la mateixa màquina.

El paradigma Client-Servidor és un esquema molt estudiat a les assignatures relacionades amb xarxes i sistemes distribuïts en el que hi ha un programa Client que fa peticions a un programa Servidor i aquest li dóna resposta.

El Client és la part activa, ja que és el que fa les peticions i d'aquesta manera inicia la comunicació, llavors espera la resposta del servidor. Pot connectar-se a diversos servidors a la vegada i habitualment és el que interactua amb l'usuari

final mitjançant una interfície gràfica.

El Servidor es pot considerar que és la part contrària, és a dir la part passiva que en iniciar-se espera a que arribin les sol·licituds per atendre-les. En rebre una sol·licitud fa el procés pel qual ha estat dissenyat i envia una resposta al client que pot ser un resultat de les operacions fetes o simplement un missatge confirmant que l'acció s'ha realitzat amb èxit. Habitualment poden atendre sol·licituds de molts clients a la vegada i no és gaire freqüent que interactuïn directament amb l'usuari, si necessiten alguna informació envien una sol·licitud al client perquè demani la informació pertinent a l'usuari per ells.

S'ha escollit aquest disseny ja que és relativament simple, de fàcil manteniment i les funcionalitats estan dividides de forma clara. A més és l'esquema que més s'adequa a les necessitats del projecte, ja que hi ha una part que pot processar i l'altra part pot interactuar amb l'usuari en una situació en la que cap part pot assolir les funcionalitats de l'altra. Per tant la part de Java que executa JADE i que crea els agents seria la part del servidor i la part d'interfície d'usuari en Objective-C i utilitzant COCOA seria el client. Per la comunicació s'ha escollit utilitzar un socket local, donat que és una forma molt simple de comunicació entre programes i tant Objective-C com Java en permeten l'ús.

Les parts de comunicació amb el lector RFID via Bluetooth i amb el chip GPS del dispositiu es va decidir que formarien part del client ja que COCOA i els altres frameworks pel desenvolupament d'aplicacions per l'iPhone haurien d'atorgar al programador les eines per interactuar amb aquestes funcionalitats.

5.4 Implementació final

5.4.1 Servidor

Aquesta part és un subconjunt modificat del MABETT i, per tant, està feta en Java. Hi ha alguns paquets del MABETT que s'han tret ja que ja no calen, aquests són aquells paquets que s'encarregaven de la interfície gràfica, la comunicació Bluetooth i la comunicació amb el chip GPS. A més, les llibreries del SWT que

estaven incloses també s'han tret ja que ja no es fan servir. També s'ha decidit treure el paquet core que s'encarregava del control i això es fa directament des de l'arrel del projecte.

D'aquesta forma el programa està format pels següents paquets:

- mabett: l'arrel del projecte, conté les classes destinades a iniciar el servidor i esperar i administrar les sol·licituds del client.
  - mabett.triage: en formen part totes les classes de l'agent d'emmagatzematge.

Tal i com s'ha vist l'esquema s'ha simplificat enormement ja que moltes parts han passat a formar part del client. D'aquesta forma han quedat la base del projecte on es troben les classes que inicien el servidor i que esperen i atenen les sol·licituds dels clients i el paquet de triatge que s'encarrega d'emmagatzemar les dades rebudes des de la base en agents.

Arrel del *MABETT*

Aquí es trobat la classe inicial que s'encarrega d'iniciar el servidor anomenada Boot la qual és un agent que crida a la classe gestor tot passant-li el controlador del contenidor d'agents. A més, se li ha donat un behaviour anomenat GuiBehaviourque instancia a un comportament predefinit a Jade anomenat OneShotBehaviour, que ens serveix perquè és simple i només s'ha d'executar una vegada.

La classe gestor és la que s'encarrega de les funcionalitats del servidor pròpiament dites. En quant es crea un objecte gestor es crea un socket i en cridar el seu mètode handleSocket es comença a escoltar pel port 7781 (agafat aleatòriament entre els ports lliures) esperant un missatge del client. En quant rep un missatge del client s'encarrega de parcejar-lo i extreure'n les dades que conté per tal d'introduir-les en un TriageDataAgent. En quant acaba amb la creació de l'agent envia un missatge de confirmació al client dient: "Etiqueta creada!". Acte seguit torna a esperar un nou missatge del client pel mateix port per tal de crear la següent etiqueta.

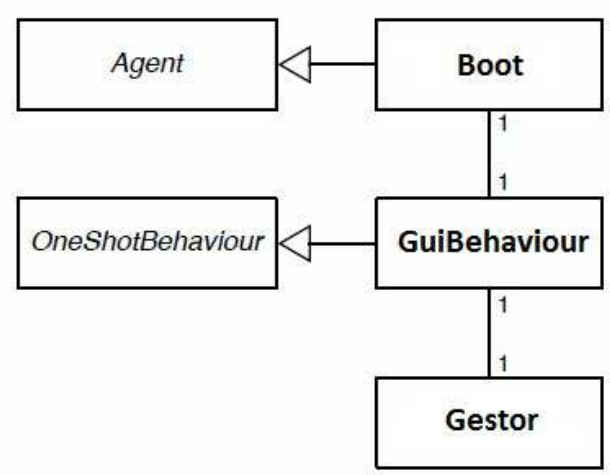


Figura 5.1: Diagrama de classes de Boot i Gestor

Packet *mabett.triage*

◇ `TriageTagAgent i TriageTagMobileAgent`

Cada cop que el servidor rep un missatge del client a través del socket es crea un agent etiqueta amb totes aquestes dades guardades. Com que actualment els agents no són mòbils i no han de fer res més que romandre a la plataforma on han estat creats, aquests no precisen de cap behaviour.

Actualment la classe agent que s'utilitza per emmagatzemar les dades d'una etiqueta és la TriageTagMobileAgent, la qual implementa un agent mòbil. Si es volguès fer proves amb agents estàtics n'hi hauria prou amb crear un TriageTagAgent.

◇ `TriageTag`

Internament els agents etiquetes contenen un objecte cada un de la classe TriageTag amb les dades preses als pacients. Aquesta classe es pot considerar com la implementació de les etiquetes de triatge tradicionals, però a part de les dades bàsiques com ara el tipus d'etiqueta s'han afegit un seguit d'anotacions que poden ser útils al personal mèdic pel posterior tractament dels pacients.

Figura 5.2: Missatge de confirmació del servidor rebut pel client

A més, una part d'aquests camps són signats, això vol dir que juntament amb aquestes dades es guarda l'identificador de l'usuari que les ha enregistrat. D'aquesta forma les dades es divideixen en:

- *Dades anònimes:* dades de les quals no interessa qui les ha anotat.
- *Dades signades:* dades que si interessa qui i quan els ha emplenat.

Les dades estan dividides en diferents classes, d'aquesta forma han estat agrupades segons la similitud entre elles. Les dades signades són subclasses de la classe AsRequired, la qual enregistra l'identificador d'usuari i l'hora automàticament quan es crea una instància de la classe. La classe TriageTag també és una subclasse de AsRequired, d'aquesta forma es pot saber qui ha fet l'etiqueta i a quina hora ha estat feta. Gràcies a això es pot saber si l'etiqueta ja és massa antiga com per tenir-la en compte o si bé és prou recent com per tractar-la.

L'identificador RFID de l'etiqueta es guarda directament a la classe TriageTag, mentre que la resta de dades es reparteixen entre les següents classes:

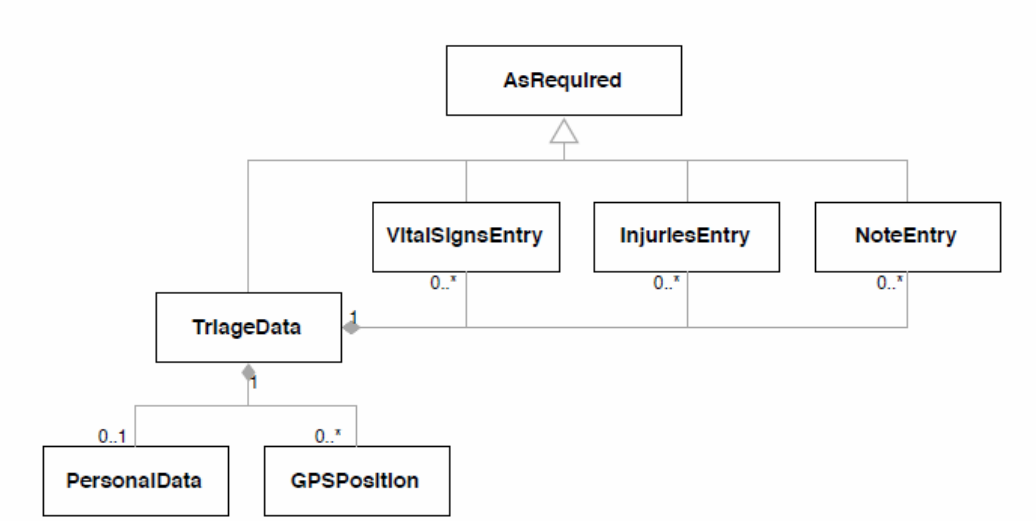


Figura 5.3: Diagrama de classes del triatge

◇ *PersonalData*

Aquesta classe s'encarrega d'emmagatzemar les dades generals del pacient de forma anònima:

- Nom del pacient: String.
- Sexe: enumeració Gender, de valors MALE i FEMALE.
- Edat: Integer.
- Adreça: String.
- Identificació: representa qualsevol codi d?identificació que el pacient, com per exemple el seu DNI. String.

◇ *VitalSignsEntry*

Aquesta classe guarda les dades que constitueixen les senyals vitals del pacient signades per l'usuari que les ha pres i anotat:

- Pulsacions per minut: Integer.
- Respiracions per minut: Integer.
- Test capil·lar en segons: Integer.

  - Estat del pacient: enumeració LabelStatus de valors DECEASED, IMMEDIATE, DELAYED o MINOR. Segons el tipus d'etiqueta que se li ha de posar al pacient.
  - Estat mental: enumeració MentalStatus de valors SANE o INSANE.

◇ *`NoteEntry`*

Permet guardar anotacions preses pel personal mèdic i signades que no es troven entre les dades predefinides:

  - Nota: String.

◇ *`GPSPosition`* Aquesta classe guarda la posició des d'on s'han pres les dades i, per tant, la posició on es troba el pacient de forma anònima amb els següents atributs:

  - Latitud: Double.
  - Longitud: Double.

5.4.2 Client

Tal i com s'ha exposat amb anterioritat, el client es la part del sistema amb la que l'usuari interactua. En la realització d'aquesta interfície s'ha procurat seguir un estil que fos similar al de les aplicacions de l'iPhone no només perquè els elements estan ja implementats, sinó també per seguir tot l'estil d'interfície que segueixen les aplicacions per aquest dispositiu.

A més a més s'ha procurat que la interfície fos relativament similar al MABETT per tal de facilitar l'adaptació del personal mèdic a les diferents interfícies que conformarà el conjunt d'aplicacions MABETT en el futur.

Finalment, caldria remarcar que s'ha intentat minimitzar la interfície per tal de que fos de fàcil ús en un dispositiu on la pantalla és de 3,5 polzades i la interacció es fa mitjançant la mateixa pantalla de forma tàctil. Per les imatges i icones de les diferents parts de la interfície s'han aprofitat les imatges del MABETT.

Vista general

El disseny general de l'aplicació consta d'una vista anomenada MainWindow, la qual ofereix quatre vistes (pestanyes) diferents a través d'una Tab bar, que és un element de COCOA situat a la part inferior de la pantalla que és el relatiu de les pestanyes que hi ha a altres programes per canviar entre diferents vistes.

Gràcies a aquest element es pot escollir entre les següents vistes: Triatge (la vista que es mostra en arrencar l'aplicació), llistat d'etiquetes, compte enrere i configuració.

Triatge

Aquesta vista presenta, a la vegada, un seguit de vistes que permeten prendre les decisions necessàries per seguir el protocol START. La navegació entre aquestes vistes ha estat implementada utilitzant una Navigation bar, la qual és també un element de interfície de COCOA situat a la part superior de la pantalla que permet la navegació entre diferents vistes i permet tornar a la vista anterior en qualsevol moment per si l'usuari s'equivoca.

Les vistes amb les preguntes que representen l'arbre de decisions són les següents:

- FirstQuestController: és la primera qüestió del protocol i pregunta si el pacient pot caminar, es decideix a quina vista es va en funció de si es diu que si o no.
  - SI: mostra la vista Minor (finalitzar etiqueta).
  - NO: mostra la vista SecondQuestController.
- SecondQuestController: pregunta si el pacient respira.
  - SI: mostra la vista ThirdQuestController.
  - NO: mostra la vista ThirdBQuestController.
- ThirdQuestController: pregunta quina freqüència respiratòria té el pacient.

  - Freqüència>=30: mostra la vista Immediate (finalitzar etiqueta).
  - Freqüència<30: mostra la vista ForthQuestController.

- ThirdBQuestController: pregunta si han funcionat les maniobres respiratòries i en cas afirmatiu quina és la freqüència respiratòria actual.
  - SI: mostra la vista Immediate (finalitzar etiqueta).
  - NO: mostra la vista Deceased (finalitzar etiqueta).

- ForthQuestController: pregunta si el pacient té pols radial i en cas afirmatiu quin és aquest (batecs per minut).
  - SI: mostra la vista FifthQuestController.
  - NO: mostra la vista FifthBQuestController.

- FifthQuestController: pregunta si el pacient pot seguir comandes simples.
  - SI: mostra la vista Delayed (finalitzar etiqueta).
  - NO: mostra la vista Immediate (finalitzar etiqueta).

- FifthBQuestController: pregunta quina repleció capilar té el pacient (en segons).
  - Repleció>2: mostra la vista FifthViewController.
  - Repleció<=2: mostra la vista Immediate (finalitzar etiqueta).

A més hi ha 4 vistes que són les encarregades de finalitzar les etiquetes. A totes hi ha els mateixos camps però el fons va variant segons el tipus d'etiqueta:

- Minor: verd, simbolitza la salud.
- Delayed: groc, simbolitza atenció.
- Immediate: vermell, simbolitza emergència.
- Deceased: negre, simbolitza la mort.

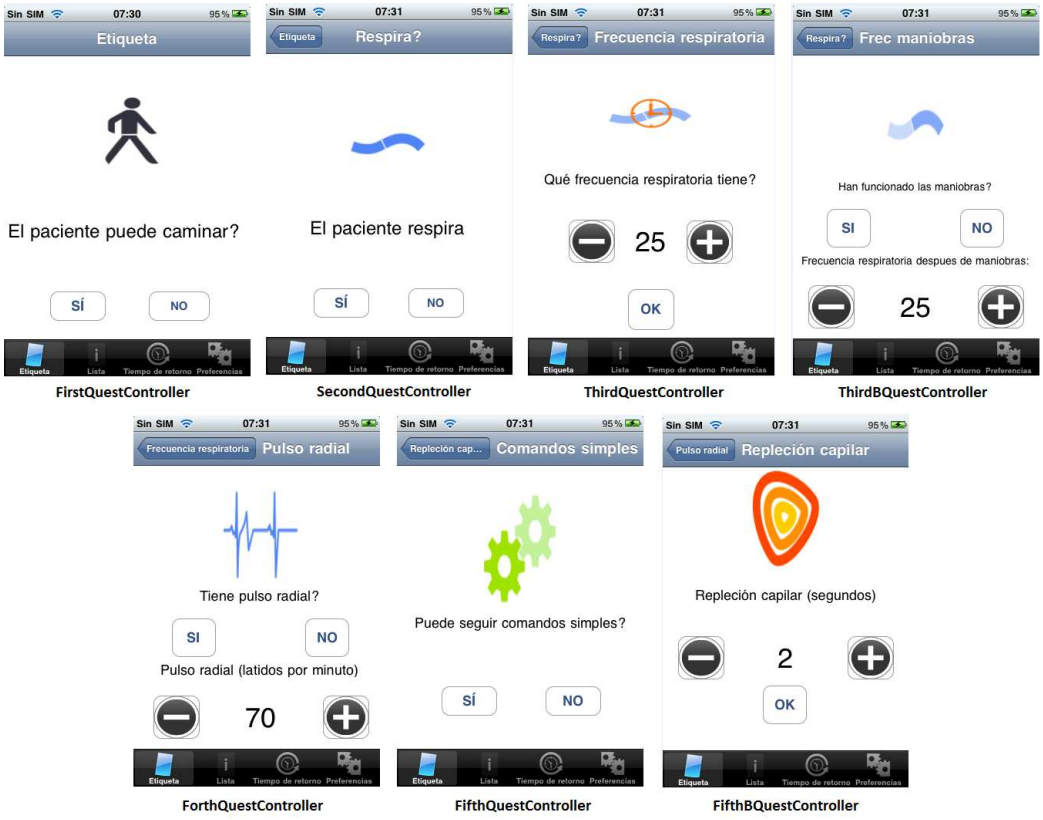


Figura 5.4: Vistes del qüestionari del protocol START

Totes aquestes vistes tenen 9 camps:

- Tag: per introduir el tag RFID de l'etiqueta manualment (UITextField: element d'interfície que permet introduir text).
- Nom: per introduir el nom del pacient (UITextField).
- Identificació: per introduir alguna dada d'identificació personal del pacient (UITextField).
- Gènere: per introduir el sexe del pacient (UISegmentedControl: element de interfície que permet escollir una opció entre n opcions).
- Notes adicionals: per informació adicional que l'usuari vulgui anotar (UITextField).

Figura 5.5: Vistes de les etiquetes finals

- Latitud: camp que s'actualitza automàticament en rebre informació del mòdul GPS amb la latitud on es troba l'usuari (UILabel).

- Longitud: camp que s'actualitza automàticament en rebre informació del mòdul GPS amb la longitud on es troba l'usuari (UILabel).

En el camp de gènere n'hi ha prou amb clicar sobre el sexe de la persona (per defecte està escollit masculí). En la resta de camps (menys el de localització) n'hi ha prou amb clicar sobre ells per tal de que es mostri un teclat tàctil a la pantalla, en acabar d'introduir el text es clica a la tecla "Aceptar"del teclat i aquest s'amaga.

Finalment hi ha un botó que s'encarrega de recopilar les dades que s'han anat introduint juntar-les en un missatge, enviar-les pel socket cap al servidor i guardar el tag de l'etiqueta en el llistat d'etiquetes que s'han creat en aquella sessió. Un cop enviades espera un missatge de confirmació per part del servidor, el mostra per pantalla i torna a mostrar la primera pregunta (FirstQuestController) per tal de poder crear una nova etiqueta.

Llistat d'etiquetes

Aquesta vista ofereix una llista (UITableView) de les etiquetes creades en la mateixa sessió i separades segons la categoria de l'etiqueta (Minor, Delayed, Immediate, Deceased). Aquesta vista no ofereix cap funcionalitat fòra de saber quines etiquetes s'han creat en la sessió de quin tipus són i identificar-les pel tag RFID.

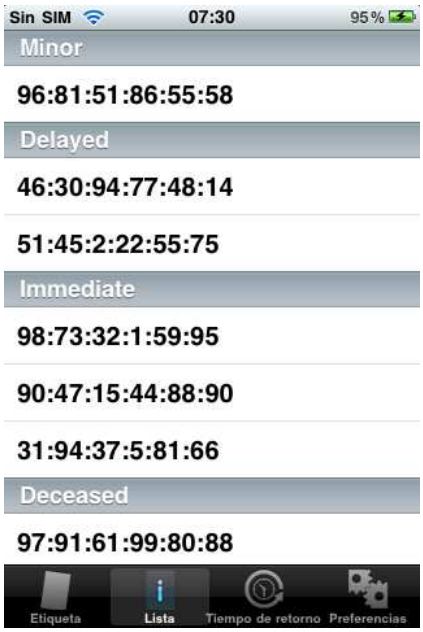


Figura 5.6: Vista del llistat d'etiquetes creades

Compte enrere

Aquesta vista permet programar un compte enrere modificant les hores, els minuts i els segons amb un botó per augmentar i un per disminuir per cada un d'aquests camps. Evidentment hi ha un botó per posar-lo en marxar i parar-lo i aquest compte enrere segueix funcionant encara que l'usuari es trobi en una altra vista.

La idea de cara a desenvolupaments futurs és que aquest compte enrere serveixi de Time To Return (TTR), és a dir que representarà el temps que queda per tal de que l'usuari torni a l'hospital i que els agents mòbils puguin migrar cap al servidor de l'hospital. A més, si es trobés un dispositiu amb un TTR menor els agents mòbils hauran de migrar cap aquell dispositiu per tal d'arribar el màxim d'aviat a l'hospital.

Figura 5.7: Vista del compte enrere

Configuració

Aquesta vista serveix per introduir i emmagatzemar les dades de l'usuari que està utilitzant el dispositiu. En obrir l'aplicació les dades de l'últim usuari que havia fet servir l'aplicació es carreguen i es poden modificar quan es vulgui. En cap moment hi ha fitxes o sessions de diferents usuaris al programa, aquests camps són merament per poder marcar les etiquetes amb l'identificador de la persona que les ha creat. Els camps són concretament el nom d'usuari i l'identificador d'aquest, ambdós es poden modificar mitjançant un UITextField i es guarden automàticament

Mòdul GPS

En haver-hi una part que estava implementada en el llenguatge per defecte de l'iPhone es va decidir que la millor opció seria que aquesta mateixa part interactuï amb el chip GPS integrat del dispositiu. Per aquesta finalitat Apple ha posat a disposició dels desenvolupadors les classes CLLocation i CLLocationCoordinate2D.

La classe CLLocation serveix per gestionar el mòdul i així indicar-li paràme-

Figura 5.8: Vista de la configuració de l'usuari

tres com la precisió. En aquesta aplicació s'ha posat com a precisió kCLLocationAccuracyBest, la qual és la màxima que permet l'iPhone (menys de 10 metres d'error).

Aquestes classes i en general el mòdul GPS s'utilitzen a les vistes que serveixen per finalitzar les etiquetes (Minor, Delayed, Immediate i Deceased) i fan que els UILabels on es mostra la longitud i la latitud s'actualitzin al començament i cada cop que varien aquests.

Mòdul Bluetooth

La idea inicial era que en intentar finalitzar qualsevol etiqueta es realitzes una comunicació via Bluetooth amb el lector RFID per obtenir el tag de l'etiqueta física en qüestió. Igual que en el cas del mòdul GPS es va suposar que el més fàcil era realitzar la comunicació des del client ja que estava fet en el llenguatge de programació per defecte de l'iPhone.

Després d'investigar es va descobrir que Apple només permet l'ús del mòdul Bluetooth per mans lliures. En total hi ha 25 perfils Bluetooth de comunicació

entre dispositius i l'iPhone només permet lús de Hands-Free Profile (HFP) i de Headset Profile (HSP). Tampoc es pot enviar missatges creats manualment per cap canal del Bluetooth des d'una aplicació iPhone. Per aquesta mateixa raó la comunicació amb el lector RFID via Bluetooth esdevé impossible.

En conseqüència el pas dels tags del lector a l'iPhone ha de ser de forma manual, és a dir que quan el lector llegeixi una etiqueta i mostri per pantalla el codi del tag de l'etiqueta l'usuari haurà de copiar aquest mateix codi manualment cap a l'iPhone a la vista de finalitzar etiqueta on es trobi.

5.4.3 Socket

Com ja s'ha mencionat la comunicació entre el client i el servidor es produeix mitjançant un socket local. El socket utilitzat és de tipus PF_INET i SOCK_STREAM, que és relativament el més habitual i com que les comunicacions no són excessivament freqüents tampoc hi ha gaire diferència amb els altres tipus. Com a port s'ha escollit el 7781 ja que en consultar un llistat de ports no utilitzats a l'iPhone apareixia aquest entre altres.

Es va decidir enviar un únic missatge amb totes les dades de l'etiqueta en format UTF8 de cop per simplificar les comunicacions, s'ha fet servir el símbol @ com a separador entre els diferents camps dintre del missatge. El format del missatge és el següent:

<identificador de l'usuari>@<tipus d'etiqueta>@<tag RFID>@<nom del pacient>@<identificador del pacient>@<edat del pacient>@<direcció del pacient>@<gènere del pacient>@<notes adicionals>@
<Latitud>@<Longitud>@<respiracions>@<estat mental>@<pols radial>@<repleció capilar>

Els identificadors, el tag, el nom, l'edat, la direcció i les notes addicionals són directament strings i si no estan omplerts la cadena és buida. Com a gènere es passa 0 si és masculí i 1 si és femení. Les latituds es passen com si fossin strings llegits del formulari.

La respiració, pols radial, repleció capil·lar i l'estat mental estan a -1 si es desconeixen i per qualsevol altre valor és que es saben. La respiració està en

respiracions/minut, el pols està en batecs/minut, la repleció capil·lar està en segons i l'estat mental està a 0 si el pacient està en un estat mental dolent i a 1 si es troba amb un bon estat mental.

L'únic que ha de fer el servidor és anar llegint caràcter per caràcter de forma que guardi els diferents camps en els atributs del TriageTagAgent tot tenint en compte que el separador utilitzat és el @.

Capítol 6

Resultats i proves

6.1 Resultats

Amb aquesta implementació final s'han aconseguit els objectius inicials del projecte i el sistema ja permet etiquetar pacients mitjançant l'iPhone seguint el protocol START. En crear cada etiqueta es crea un agent sobre la plataforma JADE amb tota la informació anotada i s'afegeix un element a la llista d'etiquetes creades.

D'entre les dades emmagatzemades de cada pacient es troba finalment la localització obtinguda del mòdul GPS. I, tot i que la comunicació per Bluetooth amb el lector RFID no ha estat possible, s'ha habilitat una forma manual d'introduir el tag de cada etiqueta RFID. La configuració de l'usuari com el nom i l'identificador es guarden per diferents execucions del programa i el compte enrere funciona i en quant es dissenyi un protocol de comparació dels TTR dels diferents dispositius només caldrà enviar un missatge a l'aplicació servidor per tal d'enviar els agents i per comunicar el TTR en cas de que el servidor el necessiti.

L'únic problema és que per tal d'executar la plataforma JADE hi ha d'haver la part del servidor que està implementada en Java i que per això cal instal·lar alguns paquets no oficials d'Apple. A més, per que no s'hagi d'obrir el terminal per arrencar el servidor cada cop que es vulgui utilitzar el sistema, s'ha deixat executant l'aplicació de forma ininterrompuda, la qual cosa suposa un lleuger increment en el gast energètic. Per tant, la implementació del sistema iMABETT

es pot considerar un èxit.

6.2 Proves

A continuació s'expliquen les proves que s'han dut a terme per comprovar que el sistema funciona correctament i que no hi ha errors per tal de poder constatar que el projecte ha sigut un succés.

6.2.1 Socket

Per comprovar el correcte funcionament del socket s'ha executat la part del servidor des de una connexió ssh tot mirant els paquets que rebia de forma parcejada. Per la part del client s'ha procedit a enviar diferents tipus d'etiquetes i omplint els camps amb diferents valors i amb valors nuls en els que es podia. En comprobar que en tots els casos provats el missatge rebut donava les dades que s'havien introduit al client es pot confirmar que tant la creació del missatge, com l'enviament, com el procés de parcejar es produeixen correctament. A més, cada cop que es crea una etiqueta el servidor envia un missatge de confirmació que en totes les proves realitzades el client ha rebut i ha mostrat per pantalla.

6.2.2 Interfície gràfica

S'ha fet tots els camins possibles entre l'arbre de decisió que regeix el protocol START de l'aplicació iMABETT i, a més, s'ha anat endavant i enrere en algunes decisions per comprovar que no es corrompien dades en aquests canvis de decisions. És a dir, que si marquem que te un pols de 70 bategs per minut i tirem enrere i anem per un altre camí o bé escollim un altre valor per aquesta dada, el pols que rep al final el servidor és l'últim que s'ha introduït. També s'ha comprovat el correcte funcionament del cronòmetre tot canviant els temps encara que estigues engegat el compte enrere. En arribar al final es para i llença un missatge avisant que s'ha acabat el compte enrere sense importar si estem en la vista de compte enrere o una altra. Per la part de configuració s'ha provat que un cop introduïdes

les dades dóna igual si es tanca l'aplicació o si s'apaga fins i tot l'iPhone, les dades de l'usuari es guarden cada cop que són modificades i es recuperen cada cop que s'inicia l'aplicació.

6.2.3 Agents

Finalment, per comprovar que el servidor crea l'agent, s'ha anat imprimint per consola quines accions feia durant la creació de l'agent i acaba enviant un missatge de confirmació al client per confirmar que el procés ha anat correctament. A més, si llavors es mira la llista d'etiquetes creades surt la nova etiqueta en la categoria que li pertoca segons el tipus d'etiqueta del que es tracta i si es consulten els agents que es troben en la plataforma s'imprimeixen els que s'han creat en aquella sessió.

6.2.4 Mòdul GPS

Per a comprovar que el mòdul funciona correctament s'ha de comprovar l'exactitud i l'actualització a mesura que canviem de posició. Per comprovar l'exactitud s'ha comparat les coordenades que mostra el Google Earth en un punt concret amb les que ens mostra el iMABETT en el mateix punt.

- Coordenades mostrades per Google Earth:
  - Latitud: 41 °27'56,91"N
  - Longitud: 2 °02'56,69"E
- Coordenades mostrades per iMABETT:
  - Latitud: 41,465852 => 41 °27'57,07"N
  - Longitud: 2,048997 => 2 °02'56,39"E

Tal i com es pot observar les diferències són relativament baixes. Això es pot comprovar a partir d'un simple càlcul: si cada grau de latitud equival a 111,319 km i la nostra diferència és de 0,16 segons fa que tinguem una diferència de 4,95

m; en canvi si cada grau de longitud equival a 111,131 km ([**?**]) i la nostra diferència és de 0,30 segons fa que tinguem una diferència de uns 9,26 m. Si apliquem pitàgores obtenim que la diferència entre el punt localitzat al Google Earth i a l'iPhone és d'aproximadament 10,5 m. En termes de diferenciar punts que es troben molt propers potser és un inconvenient però cal remarcar que les mesures tant del Google Earth com de l'iPhone no tenen precisió militar. A més, l'iPhone sempre marca un mateix punt amb les mateixes coordenades, per la qual cosa encara que les coordenades estiguin desplaçades es pot localitzar el mateix punt amb el mateix tipus de mòdul GPS. En quant a l'actualització de la localització es pot observar que la localització que marca el iMABETT va variant aproximadament cada 4-5 metres que ens desplacem.

6.2.5 Conclusió de les proves

Com que totes les proves han finalitzat amb éxit i s'han testejat totes les parts de l'iMABETT, no hi ha dubte que els objectius inicials han estat acomplerts. La única part que no s'ha provat és la del lector RFID ja que no té relació computacional amb el sistema degut a que les dades mostrades pel lector s'introdueixen manualment a l'iMABETT.

Capítol 7

Conclusions

En aquest projecte s'ha aconseguit crear amb èxit una aplicació que permeti realitzar un triatge electrònic emprant un iPhone. Amb el disseny i la implementació final s'ha aconseguit assolir els objectius proposats a l'inici del projecte i l'aplicació iMABETT aconsegueix acomplir la finalitat per la que ha estat creada. Permet tant la creació d'etiquetes que es guarden en agents mòbils, com el seu llistat per poder consultar quines etiquetes s'han creat. A més, és una aplicació amb un compte enrere preparat per treballar amb el TTR i permet guardar les dades de l'usuari. S'ha assolit també anotar la localització de cada pacient per poder-lo trobar posteriorment. Malauradament, no s'ha pogut establir la comunicació via Bluetooth amb el lector RFID per no permetre-ho les eines de desenvolupament de software per iPhone i les classes del framework que s'utilitzen amb aquesta finalitat. Aquest problema s'ha solventat permetent la introducció dels tags de les etiquetes RFID manualment a l'aplicació. D'aquesta forma s'ha aconseguit la integració del lector RFID tot i no ser de la forma desitjada en un principi.

S'ha aconseguit fer un disseny que s'assembla a l'aplicació MABETT a la vegada que segueix els patrons de interfície gràfica que presenten les aplicacions per iPhone, on tota la interacció és mitjançant la pantalla tàctil. Un inconvenient de les aplicacions en Java que s'executen en iPhone és que no es poden arrancar des de una aplicació nativa per iPhone. Per culpa d'aquest fet, la part de l'aplicació que s'encarrega de la plataforma JADE i la creació dels agents s'ha de deixar

executant constantment, fins i tot quan no es fa servir el client. Això és degut a que des del client no es pot arrancar i seria massa pesat connectar-se cada cop al terminal i arrencar aquesta part quan es vol fer servir el iMABETT. Cal dir que després de deixar en marxa la part de creació d'agents durant un temps no s'ha pogut observar un desgast significatiu de la bateria del dispositiu.

Hi ha un desconvenient de l'ús de iMABETT és que cal instal·lar paquets de software no oficials d'Apple, cosa que mostra l'hermeticitat amb la que ha estat dissenyat el sistema operatiu i la plataforma de desenvolupament. En base a això es pot constatar que Apple té força limitat el desenvolupament de software per iPhone. Tot i que permeti utilitzar molts recursos del dispositiu, no permet utilitzar el Bluetooth per comunicar-se amb dispositius que no siguin de tipus mans lliures i tampoc permet el desenvolupament de software en llenguatges que no siguin Objective-C. Per culpa d'això, els desenvolupadors de software que estan obligats a utilitzar altres llenguatges (en aquest cas per fer servir la plataforma JADE) han de recórrer a paquets no oficials, cosa que impossibilita la posterior publicació de les seves aplicacions a la Apple Store.

El desenvolupament que s'ha seguit finalment durant el projecte mostra algunes modificacions respecte a les previsions inicials degut als problemes que han sorgit per l'hermeticitat de la plataforma de desenvolupament de l'iPhone. En fer la primera implementació va resultar que no es podia fer el projecte tal i com s'havia dissenyat inicialment i, en conseqüència, es va haver de fer un redisseny i una nova implementació. Això ha fet que les dates de testeig i redacció d'una part de la memòria es retrassessin lleugerament.

Finalment, el projecte s'ha pogut acabar exitosament, però la part de disseny i implementació s'ha hagut de fer dues vegades a causa de problemes imprevistos.

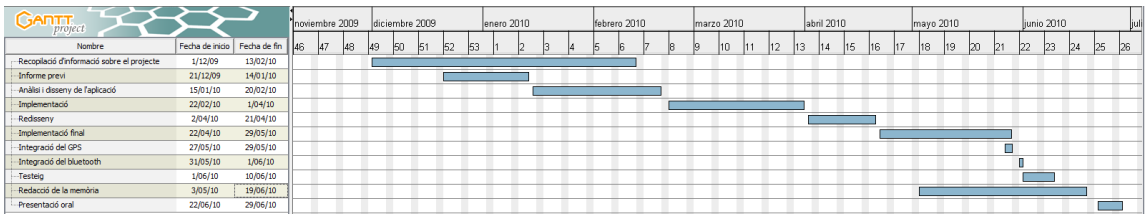


Figura 7.1: Diagrama de Gantt final

Ampliacions futures De cara a futures investigacions seria interessant desenvolupar una part essencial del sistema que seria un servidor on arribessin els agents de triatge i on es tractarien per tal de notificar al personal mèdic dels hospitals els ferits de les catàstrofes i les seves necessitats. D'aquesta forma tot el sistema cobraria sentit i es podria utilitzar. També seria important crear un protocol de comunicació entre els diferents dispositius que utilitzen el sistema MABETT i iMABETT per tal de que utilitzin el TTR (Time To Return). D'aquesta forma els agents de triatge migrarien al dispositiu que més aviat té previst arribar al servidor on es tractarien aquests agents. Finalment, no cal oblidar que les dades que es tracten en els agents són de caràcter personal i estan protegides per la Llei Orgànica de Protecció de Dades i per assegurar la seguretat dels enviaments s'hauria d'implementar algun protocol de seguretat i/o alguna encriptació per les migracions dels agents entre els diferents dispositius.

Capítol 8

Annexos

8.1 Annex 1: Instal·lació de paquets necessàris

Per tal d'utilitzar i/o desenvolupar l'aplicació iMABETT en qualsevol iPhone prèviament s'han d'instal·lar una sèrie de paquets mitjançant l'instal·lador de paquets Icy. Els paquets que s'han utilitzat són els següents:

- Classpath
- Jikes
- JamVM
- MobileTerminal
- OpenSSH

Els primers tres paquets es troben en la categoria Java de l'Icy, mentre que el MobileTerminal està a la categoria Soporte para terminal i el OpenSSH està ubicat a la categoria Redes.

Un cop s'arriba a la vista del paquet que es pretén instal·lar n'hi ha prou amb clicar al botó on posa Instalar i el programa l'instalarà automàticament. A continuació es mostra un breu esquema amb el procés a seguir per instal·lar els 5 paquets:

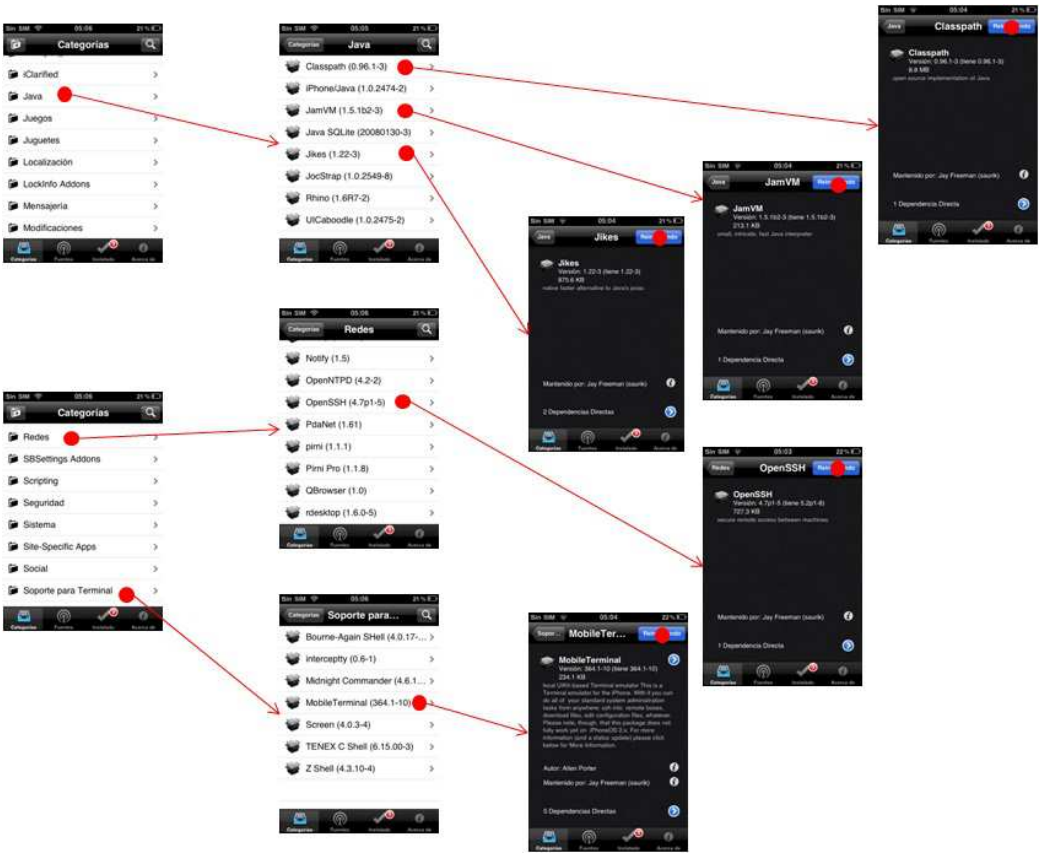


Figura 8.1: Disseny lògic de la base de dades

8.2 Annex 2: Posada en marxa del servidor

L'aplicació iMABETT necessita que el servidor, que és el que utilitza la plataforma JADE i crea els agents, s'estigui executant mentre es fa servir la part del client que constitueix una interfície en l'iPhone. Per això, el més còmode és deixar executant el servidor sempre i quan s'executi el client amb interfície i aquest envii una solicitud, que el servidor crei l'agent i respongui amb un missatge de confirmació. El servidor es pot copiar tant per ssh com connectant un cable a un ordinador i utilitzant un software de gestió de fitxers com per exemple el iPhone Explorer.

Per fer això primer s'ha de connectar l'iPhone a una xarxa inal·làmbrica i descobrir la seva IP (ajustes -> WI-FI -> [SSID del WI-FI] -> Dirección IP). Llavors

des d'algun ordinador que utilitzi un sistema operatiu UNIX s'ha d'obrir una terminal i introduir les següents comandes:

```
$ ssh root"@[IP de l'iPhone]
$ password:alpine
```

Llavors s'ha de navegar fins arribar a la carpeta on es troba el servidor i introduir aquesta comanda:

```
$ nohup java -jar server.jar &
```

Llavors ja es pot sortir de la connexió SSH i el servidor es seguirà executant fins que s'apagui l'iPhone. A partir d'aquest moment es pot obrir la part de client del iMABETT des de la seva icona a l'iPhone tantes vegades com es vulgui.

8.3 Annex 3: Elements de la interfície

En la interfície del iMABETT s'han utilitzat algunes icones del MABETT i altres s'han introduit de fora.En aquest annex es mostren els diferents elements gràfics emprats en el projecte.

8.3.1 Icones de les pestanyes

En ser utilitzades com a imatge en una Tab bar del framework COCOA aquestes imatges són convertides a blanc i negre automàticament pel compilador. Aquí es mostren les imatges originals.

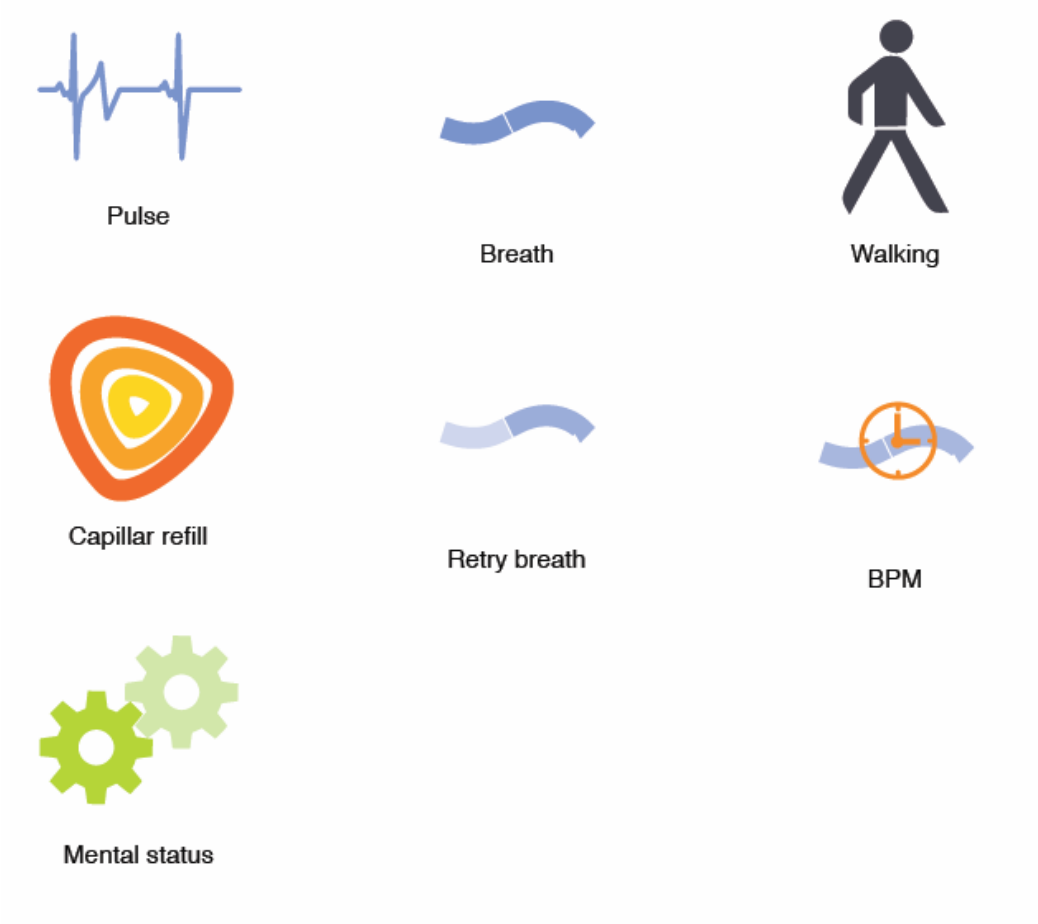
Triatge

Llistat d'etiquetes

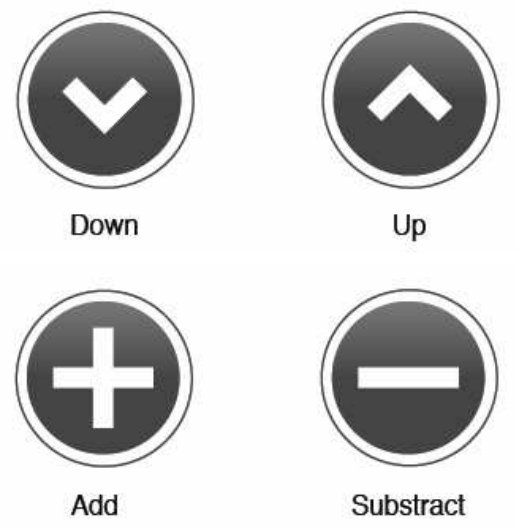
Compte enrere

Configuració

8.3.2 Icones de les vistes de triatge



8.3.3 Imatges de fons dels botons



Bibliografia

[Wiisard] Lenert L,ChanTC,GriswoldW,KilleenJ,PalmerD,KirshD,etal.Wirelessinternet information systemformedicalresponseindisasters(WIISARD).In:AMIA annual symposium proceedings; 2006.p.429Ű33.

[ARTEMIS] McGrathS,GriggE,WendelkenS,BlikeG,RosaMD,FiskeA,etal.ARTEMIS:a vision for remote triage and emergency management information integration. Dartmouth University;2003.Availableat: hhttp://www.ists.dartmouth.edu/library/15.pdfi

[MAETTS] R. Martí, S.Robles,A.Martín-Campillo, J.Cucurull: Providing early resource allocation during emergencies: The mobile triage tag. Journal of Network and Computer Applications November2009 vol. 32. no. 6, pp: 1167-1182. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2009.05.006.

[MABETT] Xavi Jurado: Dispositius handheld per a aplicacions mèdiques. Disseny i implementació d'interfícies del protocol START. Projecte de fi de carrera 06/2008.

[RFID] RFID vs Código de Barras: Obligados a respetarse <http://www.rfid-magazine.com/opinion/index.php?id=1420>

[RFID2] Ventajas de la RFID sobre el código de barras <http://www.ihg.net/java/X?cgi=lateral.rfid.VentajaVsBarras.pattern>

[RFID3] RFID, el código de barras del futuro (I Parte)
<http://www.laflecha.net/articulos/ciencia/rfid?page=5>

[TRIAGE] Triaje
<http://es.wikipedia.org/wiki/Triaje>

[GPS] Que es GPS?
<http://www.tecnoprojectltda.com/QUEESGPS.htm>

[GPS2] Qué es el GPS.
<http://www.euroresidentes.com/gps/que-es-el-gps.htm>

[GPS3] Global Positioning System
<http://en.wikipedia.org/wiki/Global_Positioning_System>

[IPHONE3GS] Especificaciones técnicas del iPhone 3GS
<http://www.apple.com/es/iphone/iphone3gs-specs.html>

[IPHONE3GS2] iPhone
<http://en.wikipedia.org/wiki/IPhone>

[IOS] The iPhone OS Architecture and Frameworks
<http://www.techotopia.com/index.php/The_iPhone_OS_Architecture_and_Framewo

[JAILBREAK] £Qué es y cómo hacer JailBreak?
<http://miipodtouch.com/que-es-y-como-hacer-un-jailbreak/>

[JAILBREAK2] Jailbreaking for iOS
<http://en.wikipedia.org/wiki/Jailbreaking_for_iOS>

[LOPD] Servicio de protección de datos.
<http://www.portaley.com/empresa/revista24042002.shtml>

[LOPD2] PRINCIPIOS DE LA PROTECCIÓN DE DATOS.
<http://noticias.juridicas.com/base_datos/Admin/lo15-1999.t2.html#a7>

[JAVA] Qué es Java
<http://www.desarrolloweb.com/articulos/497.php>

[JADE] La Plataforma de Agentes JADE: Java Agents Development Environment
<http://ants.dif.um.es/ juanbot/page_files/escuelaAgentes2005jade.pdf>

[JADE2] JADE, Java Agent Development Framework
<http://www.javahispano.org/contenidos/es/jade__java_agent_development_framework/>

[JADE3] Agentes
<http://programacionjade.wikispaces.com/Agentes#DefinicionAgente>

[JADE4] Java Agent DEvelopment Framework
<http://jade.tilab.com/>

[SWT] Algo de GUI con Java/SWT
<http://www.comunidadjava.org/?q=node/254>

[SWT2] Standard Widget Toolkit
<http://en.wikipedia.org/wiki/Standard_Widget_Toolkit>

[SWT3] SWT: The Standard Widget Toolkit
<http://www.eclipse.org/swt/>

[OBJC] Introduction to The Objective-C Programming Language
<http://developer.apple.com/mac/library/documentation/cocoa/conceptual/ObjectiveC/Introduc

[COCOA] COCOA
<http://developer.apple.com/technologies/mac/cocoa.html>

[OBJC2] UML Unified Modeling Language, juny 2005.
<http://personales.ya.com/macprog/LenguajeObjective-C.pdf>

[JAVAIPHONE] port of Java to the iPhone (with Objective-C connectors)
<http://www.saurik.com/>

[JAVAIPHONE2] Instalar, Compilar y Ejecutar Java en el iPhone
<http://www.notasdesamuel.com/instalar-compilar-y-ejecutar-java-en-el-iphone/>

[COORDENADAS] Coordenadas geográficas
<http://es.wikipedia.org/wiki/Coordenadas_geogr%C3%A1ficas>

Firmat: Gueorgui Bojidarov Radev

Bellaterra, Juny de 2010

Resum

En situacions d'emergència, on hi ha en perill un gran nombre de vides humanes, és especialment important l'ús de triatge per a organitzar els recursos disponibles per tal de poder classificar i atendre el màxim nombre de pacients en un temps limitat. Aquest projecte presenta l'anàlisi, disseny i implementació d'un sistema de triatge electrònic mitjançant un iPhone que guarda en un agent mòbil les dades del pacient, els seus signes vitals, un tag RFID que es col·loca al pacient per a identificar-lo i seva ubicació. Després d'haver realitzat una sèrie de proves sobre l'iMABETT s'ha demostrat que funciona correctament i compleix els objectius proposats.

Resumen

En situaciones de emergencia, donde está en peligro un gran número de vidas humanas, es muy importante el uso de triaje para organizar los recursos disponibles para poder clasificar i atender el máximo número de pacientes en un tiempo limitado. Este proyecto presenta el análisi, diseño e implementación de un sistema de triage electrónico mediante un iPhone que guarda en un agente móbil los datos del paciente, sus signos vitales, un tag RFID que se coloca al paciente para identificarlo i su ubicación. Después de haver realizado una serie de pruebas sobre el iMABETT se ha demostrado que funciona correctamente y cumple los objetivos propuestos.

Abstract

In cases of emergency situations, where many human lifes are in danger, it is really important to use triage in order to organize the available resources in order to classify and attend the maximum number of patients in the shortest time. This project shows the analysis, design and implementation of an electronic triage system by means of an iPhone that stores in a mobile agent patient data, vital signs, a RFID tag that is assigned to the patient in order to be able to identify him and his location. After several tests on iMABETT it has been proved that that the system works correctly and fullfills the specifications.