SISTEMA DE MISSATGERIA PER A MÒBILS BASAT EN XARXES Wi-Fi

Memòria del Projecte Fi de Carrera
d'Enginyeria Tècnica de Telecomunicacions
Especialitat Sistemes Electrònics
realitzat per
Roberto Rodríguez Fernández
i dirigit per
Helena Rifà Pous
Bellaterra 16 de juny de 2008.

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Helena Rifà Pous, professora de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per Roberto Rodríguez Fernández.

I, perquè així consti, signa aquest certificat.

Signat: .....

Bellaterra (Cerdanyola del Vallès), 16 de juny de 2008.

SISTEMA DE MISSATGERIA PER A MÒBILS BASAT EN XARXES Wi-Fi

DESCRIPCIÓ, *ABSTRACT* I PARAULES CLAU

DESCRIPCIÓ:

El projecte sobre el que versa la present memòria pretén dur a terme l'anàlisi, disseny i implementació d'un sistema de missatgeria per a mòbils basat en xarxes Wi-Fi.

Amb aquest propòsit, i amb posterioritat a la valoració de totes les alternatives i eines disponibles, s'ha desenvolupat una aplicació en llenguatge J2ME amb configuració CLDC 1.1 i perfil MIDP 2.0.

ABSTRACT:

The project on which it turns the present memory tries to realise the analysis, design and implementation of a system of mail for mobiles based on Wi-Fi networks.

With this intention and after the valuation of all the alternatives and tools available, an application has been developed in J2ME language, with configuration CLDC 1.1 and MIDP 2.0 profile.

PARAULES CLAU:

J2ME (*Java 2 Micro Edition*), Wi-Fi, DSR (*Dinamic Source Routing),* AODV (*Ad-hoc On-Demand Distance Vector routing),* CDC (*Connected Device Configuration*), CLDC (*Connected Limited Device Configuration*), Netbeans, Eclipse i Gel.

SISTEMA DE MISSATGERIA PER A MÒBILS BASAT EN XARXES Wi-Fi

TAULA DE CONTINGUTS

SISTEMA DE MISSATGERIA PER A MÒBILS BASAT EN XARXES Wi-Fi

1. INTRODUCCIÓ

Avui dia, existeixen comunicacions per via aèria entre diferents dispositius. El *bluetooth*, els infrarojos o el Wi-Fi en són exemples.

Els sistemes de comunicació per *Wireless* (sense cables) s'implementen cada vegada amb més assiduïtat en els dispositius electrònics com ordinadors portàtils, telèfons mòbils o PDAs, per posar alguns exemples.

La conseqüència més òbvia és la possibilitat d'enviament de paquets de dades entre diferents dispositius que es troben connectats a una mateixa xarxa de dades, enviament que, d'altra banda, es duu a terme mitjançant un determinat protocol de comunicació.

L'objectiu d'aquest projecte és el d'implementar un sistema d'enviament de dades entre dispositius mòbils que disposen de connexió Wi-Fi. L'aplicació d'aquest sistema ha de permetre les transmissions multi salt (*multihop*), és a dir, ha de permetre que un usuari pugui enviar un missatge a un altre usuari que no està en el seu radi de cobertura. El mecanisme per aconseguir-ho consisteix en el recolzament en usuaris intermediaris que s'utilitzen com a pont per arribar fins al destinatari final. D'aquesta manera, donats un usuari origen i un de destí que es troba fora de l'abast de l'origen, els paquets d'informació s'envien des de l'origen fins al destí mitjançant la transmissió a usuaris intermediaris que retransmeten les dades entre ells.

Més endavant es farà un estudi per avaluar quin és el protocol d'encaminament de paquets més recomanable per aquest projecte i en quin llenguatge de programació s'implementarà per tal de tenir un abast més gran a la galeria de telèfons mòbils i PDAs del mercat actual.

S'estudiarà i comparará els *softwares* i eines de programació que existeixen per desenvolupar les aplicacions, cóm es poden adquirir, cóm s'instal·len i cóm es fan servir, així com les seves funcions, els avantatges que presenten i també les seves carències.

2. DEFINICIÓ I OBJECTIUS

Suposem dos mòbils que s'ubiquen prou separats entre ells a l'espai com per no situar-se dins del radi de cobertura que s'estableix entre ells. El que es pretén amb aquest projecte és aconseguir que una comunicació basada en xarxes Wi-Fi sigui possible, de forma que, a tal fi, s'utilitzin d'altres mòbils que es trobin ubicats a posicions intermitges, i que tindran per funció actuar de pont d'informació entre el dispositiu d'origen i el de destí. Amb la finalitat d'elaborar un estudi de la viabilitat d'implementació d'aquest tipus d'aplicacions multi-salt amb mòbils, es faran proves de camp per tal d'analitzar els riscos del projecte. La implementació es durà a terme fent servir un protocol reactiu que s'implementarà fent servir el llenguatge J2ME o *Java 2 Micro Edition*.

En definitiva, en el present projecte es desenvolupa una aplicació en J2ME amb la que es pugui enviar i rebre missatges de text entre dispositius mòbils allunyats però que tinguin instal·lada l'aplicació que concretament ens ocupa.

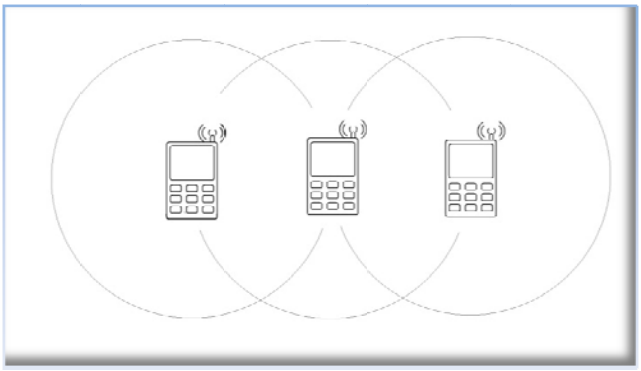


Figura 2.1: Tres mòbils

La **figura 2.1**, **Tres mòbils**, el que intenta il·lustrar és quines són les cobertures dels mòbils dibuixats. El dispositiu situat més a l'esquerra, al que anomenarem "mòbil A", no té al seu abast de cobertura el telèfon mòbil ubicat més a la dreta, és a dir, el "mòbil C". En canvi, sí té al seu abast el que hi ha enmig d'ambdós, el mòbil B, que, d'altra banda, sí té en el seu radi d'acció el mòbil C. És a dir, el mòbil A no veu el mòbil C, però sí veu el B. El mòbil B veu a tots dos mòbils, l'A i el C, i aquest darrer, evidentment, només veu el B.

Suposem que el mòbil A vol comunicar-se via missatgeria amb el mòbil C. Del que es tractaria és de fer l'enviament mitjançant el dispositiu ubicat en la situació intermitja, el B, que per la seva banda faria el re-enviament cap al C. D'aquesta manera, el missatge viatjaria del mòbil A al B, i del B al C.

Aquest sistema d'enviament és extrapolable a una situació anàloga en la que es trobarien implicats diversos mòbils intermitjos, de manera que el missatge viatjaria des del mòbil origen fins al destinatari de la informació a partir dels diversos re-enviaments que es farien entre els mòbils intermediaris ubicats entre tots dos.

En definitiva, gràcies al protocol d'encaminament que més endavant definirem, si un mòbil vol efectuar l'enviament d'un missatge, podrà establir comunicació tant amb dispositius que estiguin al seu abast, com amb mòbils que no tinguin connectivitat directe, ja sigui perquè la seva ubicació és massa llunyana, o simplement per trobar-se situats darrera d'un obstacle que impedeix la comunicació directe. La condició necessària a tal fi serà la de trobar dispositius intermitjos que puguin desenvolupar les funcions de re-enviament de la informació.

3. PLANIFICACIÓ I ESTUDI ECONÒMIC

Per tal de confeccionar la planificació del projecte, s'ha de tenir en consideració un seguit d'aspectes a valorar. Així doncs, s'ha d'avaluar el millor llenguatge de programació a escollir, s'han d'estudiar les tecnologies implicades, etc. En conseqüència, donat el ventall

de telèfons que s'oferten al mercat, el disseny s'ha confeccionat de manera que el present projecte és compatible amb qualsevol dispositiu que complís dos requisits:

- Incorporar la tecnologia Wi-Fi i
- Ser compatible amb J2ME.

4. XARXES Wi-Fi

En el present apartat d'aquesta memòria, el que farem serà realitzar una breu introducció al que són les xarxes Wi-Fi. A continuació passarem a parlar del que són els protocols d'encaminament. Realitzarem una breu descripció i comparació entre dos d'ells, el DSR i l'AODV i, finalment, exposarem les raons per les quals n'escollirem un de tots dos.

4.1 INTRODUCCIÓ A LES XARXES Wi-Fi

Wi-Fi són les sigles dels termes anglesos *Wireless Fidelity*. Es tracta d'un conjunt d'estàndards per a xarxes sense fils que es basen en les especificacions IEEE 802.11, i que van ser creats amb l'objectiu de fomentar la tecnologia sense fils i assegurar la compatibilitat d'equipaments, és a dir, la interoperabilitat. Amb unes altres paraules, tots aquells equips que tinguin el segell Wi-Fi poden treballar conjuntament sense problemes, i això independentment del fabricant de cadascun d'ells. Originàriament, va ser creat per ser utilitzat a xarxes locals sense fils, però actualment és habitual que s'utilitzi per accedir a Internet.

L'any 2003 va ser quan la tecnologia Wi-Fi va irrompre en el món dels sistemes sense fils, i no és agosarat afirmar que és possible que la seva aparició suposi una nova era en el sector de les comunicacions. Amb la seva aparició, sobtadament, dispositius lleugers i fàcils de fer servir van incorporar un espectre radioelèctric gratuït no subjecte a llicències, amb la qual cosa es va obrir la porta a l'accés sense fils de banda ampla a Internet en el mercat de masses. Per tant, ja des del seu naixement, aquesta tecnologia es perfila com a molt prometedora pel que fa referència a l'accés a distància de baix cost.

El terme Wi-Fi habitualment es sol referir a les normes tècniques que es poden fer servir per crear xarxes d'àrea local sense fils o WLAN. Com ja hem introduït inicialment en el present apartat, la norma Wi-Fi és en realitat una certificació que es pot aplicar a productes que satisfan certs criteris d'interfuncionament o interoperabilitat. D'altra banda, la tecnologia WLAN de connexió sense fils en una xarxa s'utilitza per connectar ordinadors personals o d'altres equips a una xarxa local. Les WLAN poden ser xarxes privades, com per exemple les instal·lacions que es poden dur a terme en un domicili particular, o bé poden ser xarxes públiques i de curt abast, com les que podem trobar a sales d'espera d'aeroports, cafeteries, o barriades de veïns.

Són nombroses les tècniques que permeten crear WLANs. Com ja hem comentat, les que han despertat més interès són les que es corresponen amb la família d'especificacions sense fils 802.11, i que van ser elaborades per grups de treball de l'institut d'enginyers elèctrics i electrònics (IEEE) dels Estats Units d'Amèrica. L'especificació més utilitzada actualment és la 802.11b, i es caracteritza per utilitzar bandes industrials, científiques i mèdiques (per tant, bandes ISM) a la freqüència de 2,4GHz (freqüència a la que, d'altra banda, treballen els forns microones domèstics).

És habitual que es faci servir el terme Wi-Fi fent referència únicament a equips 801.11b, tot i que, generalment, per l'usuari el terme és sinònim de xarxes i dispositius WLAN. D'altra banda, algunes organitzacions prefereixen l'expressió Radio-LAN o RLAN en substitució de WLAN o Wi-Fi.

Són molts els aspectes que han contribuït a l'èxit de la tecnologia Wi-Fi. Una de les claus que ha permès que la tecnologia Wi-Fi tingués tanta difusió ha estat que permet compartir la mateixa connexió a Internet per part de diversos usuaris, connexió que, a més a més, es pot establir sense cap tipus de cable, amb la qual cosa es facilita la mobilitat d'ordinadors i qualsevol altre dispositiu que s'estigui utilitzant amb aquesta finalitat. La popularitat dels sistemes Wi-Fi també rau en què els equips necessaris per crear els denominats *hotspots* no són gaire costosos. El fet que treballi en l'espectre ISM, que no precisa de llicència en molts països, també ha contribuït al seu èxit. Val a dir que la seva utilització no està gaire reglamentada, i aquesta situació ha estat un aspecte que ha afavorit el seu baix cost.

Tot aquesta amalgama de fets i casuístiques, en combinació amb una important demanda per part dels usuaris, ha redundat en el fet que s'ha fomentat l'expansió espectacular de la Wi-Fi en els països desenvolupats. Tal és aquesta realitat, que alguns analistes comparen aquest fenomen amb el creixement sobtat d'Internet a meitat de la dècada dels 90.

La majoria dels observadors preveuen un creixement espectacular dels sistemes Wi-Fi en els propers anys i, paral·lelament, un creixement exponencial de les ventes d'equips i de les instal·lacions de *hotspots*. Alguns símptomes ja són ara per ara palpables, serveixi d'exemple que s'han instal·lat WLANs en alguns vols aeris comercials i línies de ferrocarril, ja ho hem esmenat anteriorment, i han començat a aparèixer hotspots comunitaris que permeten compartir l'amplada de banda i posar en comú recursos entre els veïns d'una mateixa barriada. De tot això es dedueix que els operadors de *hotspots* Wi-Fi no són únicament empreses relacionades amb les telecomunicacions tradicionals.

Totes les estimacions i previsions d'ingressos estan relacionades amb la utilització dels sistemes Wi-Fi per a l'accés a Internet. Tot i això, a l'horitzó està despuntant nous telèfons amb els que es poden realitzar trucades VoIP (protocol de veu per Internet) des d'una WLAN i, en aquest mateix camí, recentment s'han desenvolupat normes de la IEEE que admeten la transmissió de vídeo en temps real, vies per les quals es podrien generar ingressos addicionals.

Malgrat tot, no tothom està convençut de l'èxit arrasador dels sistemes Wi-Fi. Així, hi ha qui es preocupa per la seguretat de la transferència de dades mitjançant aquest sistema, i tot i que els fabricants semblen decidits a garantir la seguretat de la informació, aquestes inquietuds podrien obstaculitzar l'èxit comercial del sistema Wi-Fi, un bon exemple és el sector de la telebanca, àmbit en el que la seguretat és una pedra angular molt important.

Una altra incògnita que cal explorar és si els consumidors estan disposats a pagar serveis Wi-Fi. És una realitat que moltes organitzacions ofereixen accés de franc a títol promocional, però també és innegable que ja es comença a facturar l'accés quan els clients regressen al recinte amb regularitat.

4.2 PROTOCOLS D'ENCAMINAMENT

Abans de parlar del tipus de protocol de comunicacions convé que parlem del que anomenem encaminament de les dades.

Donada una xarxa de comunicacions, a qualsevol punt de connexió que es vol incorporar a ella se li assigna una direcció que consta de quatre números de 256 posicions, i que s'anomena direcció IP. Aquesta direcció IP estableix les credencials d'identificació dins de la xarxa d'Internet. D'aquesta forma es pot identificar cada punt d'accés a la xarxa d'Internet.

L'encaminament és el mecanisme que fa possible l'enviament de paquets d'informació d'un punt a un altre de la xarxa. Un paquet d'informació que viatja d'un punt a un altre de la xarxa està configurat de manera que incorpora una capçalera d'informació addicional que inclou la direcció de destí del missatge, com si es tractés d'una carta enviada mitjançant el correu convencional.

Així doncs, l'encaminament no és altre cosa que el procés que cada xarxa implementa amb l'objectiu de fer arribar d'una manera eficient aquests paquets d'informació al seu destí final.

No totes les xarxes treballen amb el mateix mètode d'encaminament, de manera que es farà servir un mètode o un altre en funció dels requeriments que cada una d'elles implementa. En qualsevol cas, el que principalment es busca és un compromís prou adequat entre la velocitat d'enviament i la seguretat, entenent per seguretat el fet de què cap paquet d'informació es perdi i quedi sense arribar al seu destí.

Posem un exemple que il·lustri amb més claredat aquesta idea. Suposem que la transmissió de les dades s'efectua dins del context d'una entitat bancària. En aquest cas el tipus d'encaminament que es farà servir serà aquell en el que tots els paquets arribin de manera eficient al seu destí malgrat la velocitat de l'enviament no sigui la més ràpida possible, o el que és el mateix, prima la seguretat sobre la rapidesa. En canvi, suposem ara que el context

en el que ens ubiquem és el de l'establiment d'una videoconferència. En aquesta nova situació no és tan prioritari que tots els paquets arribin al seu destí, si se'n perden pel camí no és una situació greu, ara el que és important és que es pugui seguir la conversació en temps real, tot i que es trobi a faltar alguna imatge durant la comunicació. Aquest és un cas clar en el que prima la velocitat sobre les dades.

La conseqüència que se'n desprèn d'aquest raonament és que les xarxes faran servir diferents protocols d'encaminament en funció de les seves pròpies necessitats.

Existeixen diferents tipus de protocols per a xarxes, i es poden classificar principalment en dues categories: els protocols reactius i els protocols proactius. A continuació en parlem de cadascun d'ells.

Els protocols proactius es caracteritzen per estar formats per nodes que tenen una taula d'encaminament per a diferents direccions IP. Amb l'objectiu d'assolir una resposta més ràpida, de forma periòdica es realitza una actualització d'aquesta taula, de manera que constantment es realitza l'enviament de paquets de dades per tal d'actualitzar la taula d'encaminament.

Tant a las xarxes on es fan servir protocols reactius com aquelles que utilitzen els proactius, els nodes realitzen un enviament de paquets fent un *broadcast*, en d'altres paraules, des del node origen l'enviament es duu a terme a tots els nodes adjacents, i aquests, per la seva part, efectuen un reenviament també als seus nodes adjacents, de manera que el paquet que s'ha emès es va estenent per tota la xarxa fins que arriba al node destinatari. A cada node intermediari es va apuntant els diferents salts que es van efectuant entre node i node per a que hi quedi constància, creant taules d'encaminament, de forma que així s'habilita la possibilitat d'elecció del camí més curt a seguir pels paquets d'informació posterior. Tots dos protocols es diferencien en la manera de construir aquestes taules d'encaminament o taules de *routing*.

Els protocols proactius confeccionen les taules d'encaminament a priori, abans de que sigui necessari el seu us, de manera que, amb aquest propòsit, de forma periòdica es realitzen enviament de paquets de gestió que tenen per finalitat actualitzar aquestes taules.

D'altra banda, els protocols reactius només efectuen l'enviament de missatges d'actualització de les taules de *routing* quan es sol·licita l'enviament d'un missatge. D'aquí es dedueix que no introdueixen tant *overhead* a la xarxa però, en contrapartida, es produeix un retard a causa de la creació d'aquesta taula d'encaminament, per la qual cosa no són útils per enviar un missatge que hagi d'arribar al destinatari de manera instantània.

Per aquesta raó, es podria dir que els protocols reactius treballen sota demanda o, expressat d'una altra manera, quan un node emissor té la necessitat de conèixer la ruta a seguir fins a un node destí, inicia un procés de descobriment amb la finalitat d'esbrinar quin es el camí més òptim que la informació ha de seguir. Aquest procés rep el nom de *Route Discovery*, i finalitza quan s'arriba al node destí o bé quan totes les possibles rutes han estat examinades. La taula de *routing* o d'encaminament s'emmagatzema a la memòria *cache*, a on hi romandrà fins que el node destí sigui inaccessible, o bé s'esborra perquè la ruta ha expirat, és a dir, fa molta estona que no s'ha fet servir.

Acabarem aquest apartat enumerant, simplement, alguns exemples d'aquest tipus de protocol: DSR, *Dinamic Source Routing*; AODV, *Ad-Hoc On-Demand Distance Vector Routing*; TORA, *Temporary Ordered Routing Algorithm*; i ABR, *Associative-Based Routing*.

4.3 COMPARACIÓ

Ens centrarem en el DSR i el AODV. Tots dos són protocols reactius, i ambdós realitzen activitats d'encaminament sota demanda. Principalment, es diferencien en què DSR utilitza encaminament a l'origen, però AODV fa servir taules i números de seqüència en els nodes de destí. A continuació parlem amb una mica més de deteniment de cadascun d'ells. Un cop analitzats tots dos passarem a comparar-los.

4.3.1 DSR (Dinamic Source Routing)

Tal i com ja hem apuntat anteriorment, la característica principal del DSR és que fa servir encaminament a l'origen. Per tal de poder operar d'aquesta manera, l'emissor ha de conèixer la ruta complerta salt-a-salt (*hop-by-hop*) fins al node destinatari. La *Route Cache* és l'espai de memòria a on s'emmagatzemen aquestes rutes. D'altra banda, els paquets de dades que contenen la ruta complerta des de l'origen fins al destí es troben a la capçalera del propi paquet.

Quan un node en la xarxa *ad-hoc* intenta enviar un paquet de dades a un node destí del que no coneix encara la ruta, utilitza el procediment de *Route Discovery* per determinar dinàmicament quin és el camí fins al node destinatari en qüestió.

El mecanisme *Route Discovery* funciona inundant la xarxa amb paquets de petició de ruta (*Route Requets Packets-RREQ*). Aquests paquets són reenviats d'uns nodes a una altres fins que s'arriba al node destinatari de la petició, o bé fins que s'arriba a un node intermig que incorpora a la *Route Cache* el camí que s'ha de seguir dins al node destí.

El node destí, o bé el node que tingui a la *Route Cache* l'entrada del node destí, envia una resposta al node que ha realitzat la petició (*Route Reply* – RREP), resposta que inclou la ruta sol·licitada.

En cas de què qualsevol enllaç de la ruta en l'origen caigui, es notifica al node emissor mitjançant un paquet d'error (*Route Error* RERR). Aleshores, el node origen s'elimina l'enllaç en qüestió de la seva *Route Cache*. Però si aquesta ruta és encara necessària, s'ha de començar novament el procediment de descobriment de ruta.

El protocol DSR és força agressiu pel que fa referència a l'encaminament en l'origen i l'ús de *Route Caches*. Qualsevol node intermig en la ruta que enllaça el node origen i el destinatari, actualitza la seva informació a la seva corresponent *Route Cache* per a possibles usos futurs. D'altra banda, no és necessari l'ús de mecanismes especials per a la detecció de bucles.

4.3.2 AODV (Ad-hoc On-Demand Distance Vector routing)

Sense que hi hagi encaminament en l'origen, AODV fa servir les entrades contingudes a la taula d'encaminament tant per enviar els paquets de dades al destinatari de la informació com per propagar els *Route Replay* fins al mateix origen.

El protocol AODV utilitza números de seqüència mantinguts en cada node destí per determinar el grau de validesa de la informació d'encaminament i també per prevenir l'existència de bucles en efectuar l'enrutament. Amb aquesta finalitat, aquets números de seqüència viatgen en tots els paquets d'encaminament.

Una característica important del protocol AODV pel que fa referència a la taula d'encaminament, és el fet que l'enrutament d'estats es basa en *timers* (*timer-based states*), de manera que la taula d'encaminament expira en cas de què no s'hagi fet servir recentment, o el que és el mateix, per desús.

D'altra banda, també es manté un conjunt de nodes predecessors per a cada entrada en la taula d'encaminament, i denoten el conjunt de nodes veïns que fan servir l'entrada de la taula per encaminar els paquets de dades. En cas de què un enllaç pròxim al salt hagi caigut, se'ls i comunica als nodes mitjançant paquets del tipus *Route Error*, de forma que cada node predecessor reenvia aquest *Route Error* al seu propi conjunt de nodes predecessors, amb la qual cosa s'eliminen totes les rutes existents que fan servir l'enllaç que ha caigut.

4.3.3 DSR *vs* AODV

El protocol AODV adopta un mecanisme molt diferent per mantenir la informació d'encaminament. A diferència del protocol DSR, que pot tenir entrades múltiples a la *Route Cache* per a cada node destí, AODV utilitza taules d'encaminament tradicionals, de manera que només tenim una entrada per a cada node destinatari possible.

En DSR, el node origen conté i apunta el *path* complert per on han de passar els paquets, de manera que als nodes intermitjos no els hi cal disposar de cap informació addicional, és el node origen qui els hi proporciona les dades que necessiten. D'altra banda, en AODV el node origen no coneix el camí sencer, de forma que simplement sap que per tal de que el missatge arribi al node destinatari, ha d'encaminar el paquet cap a un node en concret, però desconeix quins són els següents nodes per on es passarà la informació a partir d'ell, ja que l'encaminament el van decidint els nodes intermitjos.

Tant el protocol DSR com l'AODV descobreixen rutes només mitjançant la presència de paquets de dades que necessiten ser encaminats. En tots dos, el descobriment de rutes es basa en un procés de cicles de consultes i respostes, i en la informació d'enrutament emmagatzemada en forma de taules d'encaminament pel cas del protocol AODV o de *Route Caches* pel DSR.

El protocol DSR té accés a una major quantitat d'informació d'encaminament que AODV. Així, fent servir un simple cicle consulta – resposta, en DSR l'origen pot aprendre les rutes fins als nodes intermitjos que conformen el camí fins al node destinatari del missatge. De fet, cada node intermig pot aprendre les rutes cap a cada node de la pròpia ruta que s'ha de seguir.

Per un altre costat, el fet de què DSR contesti a totes les peticions de la recerca de nodes destí, permet que l'origen aprengui rutes alternatives fins al node destinatari de la informació, i que seran de gran utilitat en cas de què per alguna raó la ruta primària (la més curta) no funcioni. Un cop una ruta és viable per a la transmissió de la informació, la resta de camins són ignorats.

En canvi, en el protocol AODV el mecanisme d'eliminació de rutes basat en l'ús de paquets Route Error és més conservartiu. Mitjançant la llista de nodes predecessors, els paquets d'error arriben a tots els nodes fent servir l'enllaç caigut o com a part de la ruta cap a un node destinatari. D'altra banda, en DSR els nodes ubicats entre l'enllaç caigut i el node destí no són informats ràpidament de l'error, però els nodes entre l'origen i l'enllaç caigut sí que se n'adonen immediatament. A continuació es valorarà quin és el millor protocol d'encaminament.

4.4 JUSTIFICACIÓ DEL PROTOCOL ESCOLLIT

El protocol que millor s'adapta a aquest projecte és el de tipus reactiu AODV.

En primer lloc, a diferència dels nodes tradicionals, ens trobem amb què els nodes Wi-Fi estan en constant moviment. En cas de fer servir un protocol DSR, la ruta estaria a l'origen i s'hauria d'actualitzar per *broadcast* constantment. Aquesta forma seria molt agressiva, sobretot si tenim en consideració l'esmentat moviment físic dels dispositius, d'on es conclou que estaríem sempre en *Route Discovery*. Fent servir AODV cada node té una taula, i l'origen fa enviaments al seu node consecutiu. En cas de què es produeixin errors a un altre node, aleshores sí que es fa una *Route Discovery*, però només s'actualitzen les taules dels nodes que es veuen afectats pel node que no funciona.

D'altra banda, s'ha de tenir en consideració que els missatges enviats seran missatges curts i la forma més estesa és l'enviament de Datagrames. Fent servir AODV podem fer paquets més petits, ja que no han de portar la ruta des de l'origen.

5. LLENGUATGE I EINES DE DESENVOLUPAMENT

A continuació parlarem de quin és el llenguatge de programació que farem servir en el present projecte, així com quines són les eines de desenvolupament disponibles.

5.1 J2ME

Les sigles J2ME es corresponen amb *Java 2 Micro Edition*, i no és altra cosa que un conjunt d'eines i APIs que permeten el desenvolupament d'*applets* i aplicacions.

Originàriament, el Java cobria les necessitats de mercat, ja que permetia el desenvolupament de qualsevol demanda que es pogués originar. Tot i això, amb el pas del temps i la velocitat d'innovació de les noves tecnologies, han anat apareixent noves exigències que s'han hagut de cobrir.

Actualment, els productes de software tendeixen a ser distribuïts i a compartir informació des de diferents ubicacions físiques, és el que es denomina *Enterprice Aplications*. Sun, l'empresa creadora, va presentar J2EE, Java *2 Enterprice Edition*, amb l'objectiu de cobrir aquestes necessitats presents al mercat.

Les aplicacions distribuïdes suposen el tractament de tasques específiques de la xarxa, operacions d'entrada / sortida (E/S) així com el tractament de bases de dades sota un enfocament totalment diferent al que en els seus orígens permetia una aplicació convencional. A més a més, aquestes aplicacions requereixen la gestió de dades que ocupen una considerable quantitat d'emmagatzematge en memòria i per tant presenten una complexitat addicional pel que fa al disseny de les APIs associades. Es per totes aquestes raons que es va integrar el paquet J2EE en el ja conegut J2SE.

Si J2EE va néixer com a resposta als requisits que presentaven les noves aplicacions, no cal dir que també va sorgir la necessitat d'un paquet que pogués donar suport a les aplicacions dissenyades per a telefonia mòbil, PDAs, agendes electròniques, etc. Tots aquests dispositius es caracteritzen per gaudir d'una memòria relativament petita, per tenir limitacions computacionals, per disposar de displays de petita grandària, etc., i per aquestes raons J2ME resum les funcionalitats de J2SE adaptant-les als requisits mínims necessaris que són aplicables als dispositius mòbils sense fils, tal i com s'intenta representar a la **figura 5.1**, **J2ME, J2SE i J2EE**.

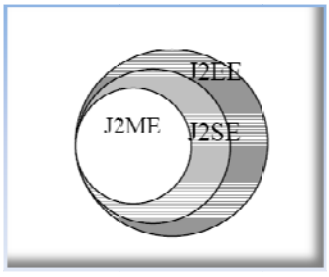


Figura 5.1: J2ME, J2SE i J2EE

Per tant, J2ME està orientat a dispositius que tenen recursos limitats, és a dir, dispositius amb un hardware amb característiques com memòria i velocitat molt inferiors a les d'un ordinador convencional, com per exemple un telèfon mòbil.

Tal i com s'esdevé amb J2ME, la principal virtut que caracteritza a J2ME és la de poder executar-se sobre diferents plataformes, de fet, gairebé tots els mòbils fabricats avui en dia incorporen suport Java, i les excepcions més significatives són el *iPhone* i els mòbils amb Windows Mobile. Malgrat això, no tots els mòbils tenen el mateix suport de J2ME.

J2ME ha estat creat per adaptar-se a les característiques des nous dispositius sense fils com ara telèfons mòbils i PDAs. En conseqüència, existeixen diferències amb la versió estàndard destinada a PCs, J2SE. Aborden a continuació, i de forma esquemàtica, les principals:

1. TIPUS DE DADES: J2ME suporta només un subconjunt de tipus de dades de J2SE. En particular, els tipus *float* i *double* no són suportats, i això és així per dos motius, en primer lloc, la majoria de dispositius CLDC no tenen unitat de coma flotant, d'altra banda, es tracta d'una operació molt costosa.

2. PREVERIFICACIÓ: En J2SE es realitza la verificació de codi en temps d'execució, en canvi, en J2ME una part d'aquesta verificació es realitza *off-line*, és a dir, fora del dispositiu. La finalitat que es persegueix és reduir la càrrega de la màquina.

3. DESCRIPTOR I MANIFEST: Quan es duu a terme l'empaquetament d'arxius en J2ME, es fa necessària la inclusió d'uns arxius especials que contenen informació de les aplicacions que inclouen. Aquests arxius reben el nom de manifest i descriptor.

4. NOVA LLIBRERIA GRÀFICA: Mitjançant el paquet *Icdui*, J2ME defineix un nou conjunt de classes per a la creació d'interfases gràfiques. Aquestes classes estan adaptades a dispositius amb memòries molt limitades i pantalles de mida reduïda.

5. DESAPARICIÓ DEL *MAIN*: Al contrari que a les aplicacions de l'edició estàndard de Java, en J2ME no existeix una funció *main* com a punt d'entrada per a l'execució del programa que es confecciona. En J2ME, l'equivalent a aquest *main* és el mètode estrat *app*.

6. ABSÈNCIA DE RECOL·LECTOR DE BROSSA: Desapareix el recol·lector de brossa amb l'objectiu de reduir la utilització de memòria, de forma que en J2ME és necessari eliminar de forma explícita tots aquells elements que no es volen fer servir en un futur.

5.1.1 DISPOSITIUS COMPATIBLES

Centrant-nos en la telefonia mòbil, presentem a continuació la llista de dispositius amb els que es podria gaudir de l'aplicació que es pretén dissenyar, dispositius, per tant, que incorporen tecnologia Wi-Fi i són compatibles amb J2ME.

BenQ-Siemens P51	HTC S620	Nokia E61	Qtek 9090
Blackberry 6280	HTC S710	Nokia E61i	Qtek 9100
Blackberry 7105T	HTC S730	Nokia E65	Qtek 9600
Blackberry 7130C	HTC Shift	Nokia E70	Qtek S200
Blackberry 8700G	HTC Touch	Nokia E90	Samsung SGH I600
Blackberry 8820	HTC Touch Dual	Nokia N80	Samsung SGH P200
Blackberry Pearl 8120	HTC TyTN	Nokia N800	SonyEricsson P1i
Grundig GR980	HTC TyTN II	Nokia N81	SonyEricsson P990i
Grundig X5000	Nokia 6086	Nokia N91	SonyEricsson W960i
HTC Advantage X7500	Nokia 6136	Nokia N92	Toshiba TSG500
HTC P3300	Nokia 6301	Nokia N93	Toshiba TSG900
HTC P3350	Nokia 770	Nokia N93i	Vodafone 1240
HTC P3600	Nokia 9300i	Nokia N95	
HTC P4350	Nokia 9500 Comunicator	Qtek 2020i	
HTC P6300	Nokia E51	Qtek 8310	
HTC P6500	Nokia E60	Qtek 9000	

D'altra banda, existeixen dispositius que suporten J2ME i, malgrat no portar integrada la tecnologia Wi-Fi, se'ls hi pot instal·lar per mitjà de la ranura d'expansió miniSD. Aquest seria el cas, per exemple, del telèfon mòbil Motorola Q9.

Finalment, la compatibilitat amb PDAs i portàtils no és directa, ja que utilitzen la JVM standard i no J2ME. En qualsevol cas, fer el pas a J2SE no es gaire costós, i seria una possibilitat a contemplar.

5.1.2 CONFIGURACIONS I PERFILS

J2ME es divideix en configuracions (*configurations*), perfils (*profiles*) i APIs opcionals.

Una configuració defineix un tipus de dispositiu en funció de les seves característiques hardware, és a dir, les seves limitacions i les seves capacitats associades, i li assigna una màquina virtual i un conjunt d'APIs adequats a aquest hardware.

Actualment existeixen dues configuracions:

- CDC, *Connected Device Configuration*: És utilitzat sobretot en sistemes de telemetria, automoció i domòtica.
- CLDC, *Connected Limited Device Configuration*: Es tracta d'una versió més limitada, i és la que ens interessa perquè és present en la majoria dels dispositius mòbils.

Donada una configuració, un perfil ens defineix certes característiques concretes, com seria la interfase de l'usuari. Per a la configuració que ens interessa, la CLDC, existeixen tres perfils possibles:

- MIPD, *Mobile Information Device Profile*: És la que es fa servir en els telèfons mòbils, i per tant és la que haurem de considerar per dur a terme el present projecte.
- IMP, *Information Module Profile*: És una variant de l'anterior perfil, però aquí no hi ha interfase d'usuari.
- DoJa: Aquest perfil està destinat a un tipus de mòbils japonesos.

Finalment, en funció de com hagi estat dissenyat i implementat, cada dispositiu mòbil pot incloure suport per a diferents APIs opcionals. Donat que aquestes capacitats varien en funció del dispositiu que tractem, cal remetre's a les especificacions tècniques de cadascun d'ells per descobrir quins APIs suporten. Posarem un exemple, un dispositiu amb *bluetooth* gama mitja/alta pot venir preparat per llegir i processar informació geolocalitzada amb la *Location* API (JSR 179), però per cerciorar-nos, hem de consultar les especificacions que el caracteritzen.

D'altra banda, cada fabricant sol realitzar implementacions de les APIs J2ME pels seus mòbils. Tot i que per fer-ho es basen en una implementació de referència de Sun, a la pràctica ens podem trobar amb diferències significatives en el comportament de diferents dispositius per a una mateixa funció, fent que no funcioni en alguns mòbils el que sí funciona a d'altres. En algunes situacions, això implica la necessitat de desenvolupar més d'una versió per a una mateixa aplicació.

A continuació, anem a aprofundir amb una mica més de detall en cada una de les configuracions que hem esmentat, de manera que analitzarem per a quins dispositius estan orientats i quins són els requeriments bàsics de hardware que necessita cada una d'elles.

5.1.2.1 CONFIGURACIÓ CDC

La configuració CDC està orientada a dispositius que es connecten de manera intermitent a una xarxa. Cobreix tot el contingut de la màquina virtual de Java, sent molt similar a J2SE. De fet, la diferència rau justament en les característiques tècniques dels dispositius que fan servir la configuració, com la capacitat de memòria o les prestacions dels displays.

A continuació resumim els requeriments necessaris pels dispositius amb configuració CDC tal i com queden explicitats a les especificacions oficials de J2ME:

- Processador de 32 bits.
- Un mínim de 2 MB de memòria total disponible per a Java 2. Aquí queda inclosa la memòria RAM, flash o ROM.

- Es requereix que el dispositiu gaudeixi d'absoluta funcionalitat amb Java 2 *"Blue Book" virtual machine*.
- El dispositiu ha de permetre la connectivitat a algun tipus de xarxa, malgrat es tracti d'una connexió intermitent de tipus Wi-Fi de banda limitada (tot i ser de 9600 bps o inclús de menor velocitat).
- El dispositiu ha de tenir algun tipus d'interfície amb cert grau de sofisticació.

5.1.2.2 CONFIGURACIÓ CLDC

La configuració CLDC és més prevalent en el context J2ME. Es va distribuir per primera vegada a l'any 1999 amb l'objectiu de crear un denominador comú mínim a la plataforma Java, més concretament per a temes de *networking*, E/S, seguretat i biblioteques nucli.

Sota aquesta configuració, exposem a continuació els requeriments que han de complir els dispositius:

- El dispositiu ha de tenir entre 160 i 512 KB de capacitat de memòria total disponible per a la plataforma Java. Queden incloses les memòries RAM, flash i ROM.
- El dispositiu pot tenir potència limitada, com ara la potència d'operació.
- El dispositiu ha de permetre la connectivitat a algun tipus de xarxa, malgrat es tracti d'una connexió intermitent de tipus Wi-Fi de banda limitada (tot i ser de 9600 bps o inclús de menor velocitat).
- El dispositiu ha de tenir algun tipus d'interfície amb cert grau de sofisticació.

5.1.3 JUSTIFICACIÓ DEL PERFIL ESCOLLIT

A l'apartat anterior hem abordat el tema de les configuracions dels equips, i hem vist que poden ser de dos tipus, CDC o CLDC. En funció de la configuració que fem servir haurem d'utilitzar una màquina virtual diferent. Així, si escollim la configuració CLDC emprarem

la KVM, *Kilo Virtual Machine*, però si ens decantem per la configuració CDC utilitzarem la CVM, *Compact Virtual Machine*.

Un cop analitzades les dues configuracions, prendrem com a referència el mercat pel que fa referència a la telefonia mòbil per tal de decantar-nos per un tipus de configuració o altre, basant-nos, simplement, en quina de les dues opcions està més implementada en els mòbils i PDAs que es comercialitzen actualment.

Tenim només sis dispositius al mercat que fan servir la configuració CDC 1.0 (J2ME Polish, 2007), són els següents: Nokia 9300, Nokia 9300i, Nokia 9500i, les sèries 90 de Nokia, Sony-Ericsson M600, i Sony-Ericsson p990i. Així doncs, es tracta d'aparells de gamma alta, de fet, els quatre Nokia en qüestió són dispositius batejats com a *comunicators*, ja que donades les seves funcionalitats tenen característiques pròpies d'una oficina mòbil o d'un ordinador portàtil de mides reduïdes, quedant allunyats de les funcions simples d'un telèfon mòbil.

Pel que fa referència als dispositius de mercat que treballen amb configuració CLDC, ens trobem amb un ventall molt més ampli de possibilitats, ja que 273 dispositius fan servir CLDC 1.0, 7 dispositius utilitzen CLDC 1.0.4 i, finalment, 296 treballen amb CLDC 1.1.

Donades aquestes dades, la decisió es fa força evident, sobretot si considerem que qualsevol dispositiu amb configuració CDC suporta una aplicació desenvolupada per CLDC.

En definitiva, en base a tota la informació recopilada, per desenvolupar el present projecte, utilitzarem la configuració CLDC amb perfil MIDP, ja que aquesta és la configuració present a la majoria de mòbils, i també és aquest el perfil que utilitzen, ja que així hi haurà present una interfase d'usuari.

L'última versió de CLDC és la 1.1, i l'última MIDP és la 2.0. Són les que incorporen els telèfons mòbils més recents, mentre que per a models més antics ens podem trobar amb la versió CLDC 1.0 que, per exemple, no suporta operacions de punt flotant. De igual manera, podria ser que incorporessin la versió MIDP 1.0, que no dóna accés a funcions de

so o pantalla complerta. Tot i això, desenvoluparem el projecte a partir de les versions més modernes.

Però amb això no tenim prou, convé abordar també un altre aspecte important, i és l'estudi de l'entorn més adequat per desenvolupar aplicacions J2ME. Partirem de la mateixa suposició raonable que hem seguit amb l'elecció de la configuració: volem que s'executi en la major varietat possible de telèfons mòbils disponibles al mercat.

Ja hem comentat anteriorment que J2ME no pot garantir que un mateix codi s'executi de manera correcta en tots els dispositius que implementin les mateixes especificacions, i d'aquí deduïm que podem arribar a un punt en què es faci necessari desenvolupar vàries versions de la nostra aplicació en funció del dispositiu a on s'hagi d'executar.

Sigui com sigui, la programació es durà a terme a partir de llibreries genèriques, excloent les que són específiques de cap marca comercial en particular. Amb això aconseguirem donar el codi la característica de la portabilitat, que és el que ens interessa. L'únic problema que se'n pot derivar és que haguem de compilar-lo específicament per a cada dispositiu.

Les marques que lideren el mercat pel que fa referència a la comercialització de dispositius mòbils amb suport J2ME varien en funció del sector de públic específic al que dirigeixen els seus productes. Tot i això, podrem treballar amb fabricants tals com Nokia, Samsung, Motorola, Sony Ericcson i RIM (els fabricants de *BlackBerry*). Per tant, en aquest projecte es farà servir un entorn de desenvolupament que ens permeti utilitzar les SDKs i els emuladors compatibles amb aquestes marques.

En definitiva, la plataforma Java 2 *Micro Edition* implementa un llenguatge Java per a varis dispositius: telèfons mòbils, PDAs, Internet *screenphones*, *set top boxes* digitals per a TV, sistemes de navegació automotiva, commutadors i *routers* de xarxa, components per a l'automatització residencial, etc.

Perseguint l'objectiu d'ajustar-nos més fidelment a la configuració de la major part del mòbils que actualment es comercialitzen, haurem d'implementar l'aplicació a desenvolupar en J2ME amb configuració CLDC 1.1 i perfil MIDP 2.0.

5.1.4 XARXES *PEER TO PEER* SOBRE Wi-Fi I ALTRES PROTOCOLS DE CONNEXIONS PER A XARXES SENSE FILS

En aquest apartat comentarem algunes de les implementacions i serveis sense fils dels què es pot gaudir actualment. Més específicament, intentarem comparar les xarxes *peer to peer* sobre Wi-Fi amb altres protocols amb els que ens podem trobar a les connexions vinculades a xarxes sense fils, més concretament WAP, SMS, i I-MODE. En definitiva, es pretén aclarir què és J2ME i què no és, evitant que el lector es pugui veure immers en algun tipus de confusió. Altre punt a tenir en compte és el fet de què els següents sistemes fan servir la xarxa de telefonia de l'operadora i per tant no són serveis gratuïts.

WAP:

La tecnologia WAP, acrònim de *Wireless Application Protocol*, és una tecnologia introduïda en el mercat l'any 1995. Es caracteritza per permetre que dispositius sense fils rebin dades procedents d'Internet, dades que es mostraran a les seves corresponents pantalles. Es tendeix a pensar que es tracta d'una tecnologia que dóna suport a buscadors Web per a mòbils, però en realitat no es tracta d'una aplicació, sinó d'un protocol de connexió per xarxes sense fils.

SMS:

Acrònim de *Short Messaging Service*, es tracta d'una tecnologia que dóna suport a l'enviament i la recepció de missatges de text de longitud curta en dispositius mòbils. Una característica interessant de SMS és l'ús d'una missatgeria unificada, per la qual cosa permet l'accés a missatges de veu, correu electrònic i l'enviament i recepció de faxs des d'un dispositiu mòbil.

I-MODE:

Introduït al mercat per la companyia japonesa NTT DoCoMo, aquesta és una tecnologia que competeix amb WAP, ja que també ofereix un mecanisme d'accés a Internet mitjançant l'ús de dispositius mòbils. I-MODE disposa d'un llenguatge d'etiquetes similar a HTML que rep el nom de *compact*, cHTML. La major part dels seus usuaris es troben al Japó i a d'altres països asiàtics.

5.2 IDE's DE DESENVOLUPAMENT

Passem ara a analitzar quins són els principals IDE's, *Integrated Development Enviroment*, en els quals podem emmarcar aquest projecte per a, finalment, escollir l'opció que s'adeqüi millor a les nostres necessitats i expectatives.

5.2.1 NETBEANS

La primera opció que analitzarem la trobem a la pàgina Web de Sun Microsystem, és el *Wireless Toolkit*. Es tracta d'un software gratuït desenvolupat per aquesta empresa per simplificar el cicle de desenvolupament de J2ME. Es pot descarregar de la plana http://wireless.java.sun.com/allsoftware/, i per possibilitar el seu ús és necessari instal·lar prèviament el programa J2SDK, *Java 2 Software Development Kit*, també descarregable des de l'adreça http://java.sun.com/j2se. El *Wireless Toolkit* és una eina que permet crear i obrir projectes, però també possibilita compilar i emular les aplicacions per mitjà del programa *Ktools*. Malgrat tot aquest potencial, *Ktools* no deixa editar el codi font, amb la qual cosa és necessari descarregar d'aquesta mateixa pàgina alguns editors, com ara el *Forte*. Aquí fins i tot hi podem trobar alguns editors dedicats a una marca de telèfons en concret, com és el cas del Netbeans.

A l'apèndix I s'hi pot trobar amb detall quin és el procés a seguir per tal de dur a terme la instal·lació del Netbeans.

5.2.2 ECLIPSE

L'Eclipse és un altre programa que pot suplir les nostres necessitats un cop fetes les modificacions adients a tal fi. Es pot descarregar lliurement des de la pàgina Web http://www.eclipse.com. En aquest cas, podem utilitzar un *plugin* anomenat EclipseMe que és de força utilitat com a ajuda en el desenvolupament d'aquest programa. L'EclipseMe

també el podem descarregar lliurement des d'Internet, més concretament des de l'adreça http://eclipseme.sourceforge.net.

A l'apèndix II hi trobarem els successius passos que s'han de prendre en el procés d'instal·lació de l'Eclipse.

5.2.3 GEL

Finalment, l'última opció de software de desenvolupament que considerarem és el programa GEL. També és gratuït, i es tracta d'un editor que facilita molt l'edició amb eines tals com l'*auto-complete*. És possible descarregar-lo des de la pàgina web http://www.gexperts.com.

Els detalls referents a la seva instal·lació es poden trobar a l'apèndix III.

5.2.4 JUSTIFICACIÓ DE L'IDE ESCOLLIT

En Primer lloc, dir que totes les IDE's comentades en aquest projecte són *softwares* gratuïts que es poden descarregar des de les respectives pàgines web oficials.

Donat el ventall d'IDE's considerat, l'elecció s'ha decantat pel NetBeans IDE RC1. Es tracta d'una Release Candidate, i és una versió encara no definitiva publicada de forma prèvia a la versió que sí ho és. Actualment ja està disponible la versió 6.0, i val a dir que, aparentment, no presenta cap diferència.

Pel que fa referència als altres IDE's considerades, es poden fer algunes observacions que, s'ha de dir, són de caràcter subjectiu:

- GEL: Tot i que no s'ha pogut testar sota els sistemes operatius Linux ni Windows XP, s'ha comprovat que presenta problemes de compatibilitat amb Windows Vista.

Això ha estat un aspecte determinant pel projecte, ja què majoritàriament s'ha desenvolupat en un PC que treballa amb aquest sistema operatiu instal·lat.

- Eclipse: Està enfocat per al llenguatge Java. Tot i que es pot baixar un *Pluggin* anomenat EclipseME, dóna múltiples errors a l'hora d'emular i compilar. A més a més, en el procés de depuració del codi font permet l'ús d'algunes sentències que els dispositius CLDC no sempre suporten.

- Wireless Toolkit: Aquesta opció en concret no ha estat descartada, però val a dir que no es tracta exactament d'un IDE, la corresponent aplicació per a PC és un emulador anomenat KTools que deixa obrir projectes però, en canvi, no deixa editar-los. Es fa servir ja que el propi programa NetBeans emula amb el KVM de Wireless Toolkit, és a dir, a efectes, cada cop que s'emula amb el IDE NetBeans el procés és exactament igual al resultant d'una emulació que es dues a terme amb Ktools.

És necessari puntualitzar que els programes anteriors s'emmarquen en el context del desenvolupament i l'emulació de programes.

Però, d'altra banda, val a dir que també s'ha realitzat un ampli treball de camp, és a dir, s'han fet múltiples proves amb dos dispositius mòbils reals, en concret, un telèfon Nokia model N95 i un altre Nokia, el N95 8Gb. L'instal·lador d'aplicacions que s'ha utilitzat per tal de disposar de l'aplicació a ambdós dispositius ha estat el Nokia PC Suite.

6. ANÀLISI I DISSENY

Tal i com s'ha esmentat anteriorment, es farà servir un protocol reactiu AODV en Wi-Fi.
El tipus de connexió que s'ha fet servir és Ad-hoc per mitjà de l'enviament de Datagrames.
Tot i que teòricament s'ha fet una implementació sencera del programa, a la practica no s'ha pogut completar tota l'aplicació.

Dels objectius inicials dels que partíem, s'han pogut completar de manera eficient dos MIDlets:

- Un MIDlet d'enviament per Wi-Fi amb la possibilitat de l'enviament d'un missatge curt tant a una direcció IP determinada com per *broadcast* i fent servir un port determinat dins de la direcció IP.

- Un MIDlet de recepció que pot rebre el Datagrama i convertir la informació enviada en un *string* (cadena de caràcters) que pugui ser visualitzat per pantalla.

Alguns punts relatius a aquests MIDlets han estat modificats per tal de poder afegir algunes funcions, com ara un canvi de direccions IP a l'enviament o possibilitar l'edició del text del missatge, en el moment de la impressió de la memòria.

6.1 LLIBRERIES UTILITZADES

Les llibreries que s'han fet servir en el disseny de l'aplicació en J2ME són les següents:

- javax.microedition.midlet.*
- javax.microedition.lcdui.*
- javax.microedition.io.*
- java.lang.*
- org.netbeans.microedition.lcdui.WaitScreen
- org.netbeans.microedition.util.SimpleCancellableTask

Cal puntualitzar que el signe d'asterisc comporta que les llibreries siguin importades íntegrament. Considerem un exemple que il·lustri aquesta idea, en cas de què volguéssim importar la classe Datagrama faríem servir la sentència "import javax.microedition.io.Datagram", ara bé, mitjançant aquesta altra "import javax.microedition.io.*" podem utilitzar totes les classes que comencin per "javax.microedition.io", com per exemple "javax.microedition.io.DatagramConnection", sense la necessitat d'haver d'importar-les totes una a una.

Algunes llibreries estan ja incloses en el perfil CLDC però, malgrat tot, poden ser requerides per l' emulador o l'editor, i si no s'importen toparem amb errors. Aquest seria el cas de "java.lang.*"

Quan es fa servir una sentència suggerida pel programa, en el nostre cas el NetBeans, la resta de llibreries són descarregades automàticament per ell mateix. Així, en el cas de fer servir en l'aplicació una pantalla d'espera, en lloc de definir una classe, s'importa del paquet del propi IDE.

6.2 CASSOS D'ÚS

El primer pas que s'ha de dur a terme és la configuració del terminal per tal d'executar l'aplicació. Els terminals estaran connectats sense fils en una xarxa ad-hoc mitjançant un direccionament IP privat, i el direccionament s'ha de realitzar de forma estàtica. Val a dir que cada telèfon s'ha de configurar de manera diferent d'acord amb el model de dispositiu que utilitzem, però aquí redactarem una explicació en termes genèrics.

La xarxa que s'ha de configurar es la mateixa que fa servir una aplicació ja existent i que s'ha utilitzat com a recolzament per tal de fer el projecte. El lector pot dirigir-se a l'apartat número set denominat "Aplicació" per tal d'ampliar aquesta informació.

El següent que és necessari fer és definir un punt d'accés que modela la xarxa ad-hoc. Per tal d'aconseguir-ho, polsem la tecla "Menú" i anem a a "Herramientas", a continuació seleccionem "Ajustes" i finalment "Conexión". Ara s'ha de seleccionar "Puntos de acceso", i en fer-ho apareixerà una llista amb els punts d'accés existents, però nosaltres haurem de crear un de nou, i amb aquest objectiu seleccionarem "Opciones", seguidament "Punto de acceso nuevo", i finalment "Usar ajustes predeterminados".

Ara ja estem en condicions de definir els paràmetres del punt d'accés, que seran aquests:

- "Nombre de la conexión: Salta 2.0"

- “Portador de datos”: LAN inalámbrica”
- “Nombre de red WLAN: Red Salta 2.0”
- “Estado de la red: Pública”
- “Modo de red WLAN: Ad-hoc”
- “Modo seguridad WLAN: Red abierta”
- “Pagina de inicio: Ninguna”

A continuació es requereix l’assignació d’una direcció IP. Per tal de fer això i sense sortir de la finestra de configuració del punt d’accés, s’ha de seleccionar “Opciones”, a continuació “Ajustes avanzados”, seguidament “Ajustes IPv4i” i ja estarem en condicions d’establir els paràmetres de direccionament. A mode d’exemple, a continuació s’indica el valor de direccionament privat de classe C en un terminal:

- “Dirección IP del telèfon: 192.168.1.1”
- “Máscara de subred: 255.255.255.0”
- “Pasarela predeterminada: 192.168.1.254”
- “Dirección de DNS: Automática”

Ara només cal polsar “Atrás” fins arribar als punts d’accés i observar que hi ha un de nou.

És necessari seguir aquests passos per a cada terminal, i s’ha de tenir en conte que es requereix canviar la direcció IP per a cada telèfon.

La instal·lació a cada dispositiu mòbil es fa de forma diferent, i en el nostre cas hem utilitzat el programa Nokia PC suite. L’arxiu que es fa servir es el pfc2130406.jar, en alguns mòbils es pot copiar a la seva memòria i un cop fet això es pot instal·lar sense cap mena d’ajuda externa.

Una vegada instal·lat hem de considerar que per enviar missatges es fa servir el MIDlet “enviament6.0”, seleccionem la icona i ens demanarà permís per connectar-se a una xarxa, seleccionem “Yes” i seguidament ens farà escollir la xarxa que s’utilitzarà, la que establirem serà la xarxa “Salta2.0”, i com a resultat el dispositiu vibrarà lleugerament.

La següent pantalla és la que ens permet escriure el missatge que, d'altra banda, ha de ser de 16 caràcters, de forma que quan el *string* arribi a una longitud de 16 s'enviarà directament. En cas de que la llargària sigui inferior a 16, és necessari omplir els caràcters que falten encara que sigui mitjançant espais en blanc. En arribar a 16 i enviar-se s'informa a l'usuari de què el missatge s'ha enviat i també apareix per pantalla el missatge en qüestió.

Per poder rebre el missatge s'ha d'obrir el MIDlet "recepcio6.0", es demana permís per fer servir la connexió Wi-Fi, i els passos que s'han de seguir són els mateixos que per l'enviament. Un cop connectats al "Salta2.0" apareixerà per pantalla el missatge "Esperant rebre Datagrama", i en cas de què arribi un datagrama, simultàniament el dispositiu vibrarà, apareixerà el missatge i s'il·luminarà la pantalla durant un segon.

Anem ara a veure amb una més de detall l'aplicació. A tal fi incorporarem imatges, i sempre situarem a l'esquerra la fotografia del mòbil transmissor i a la dreta la del dispositiu receptor.

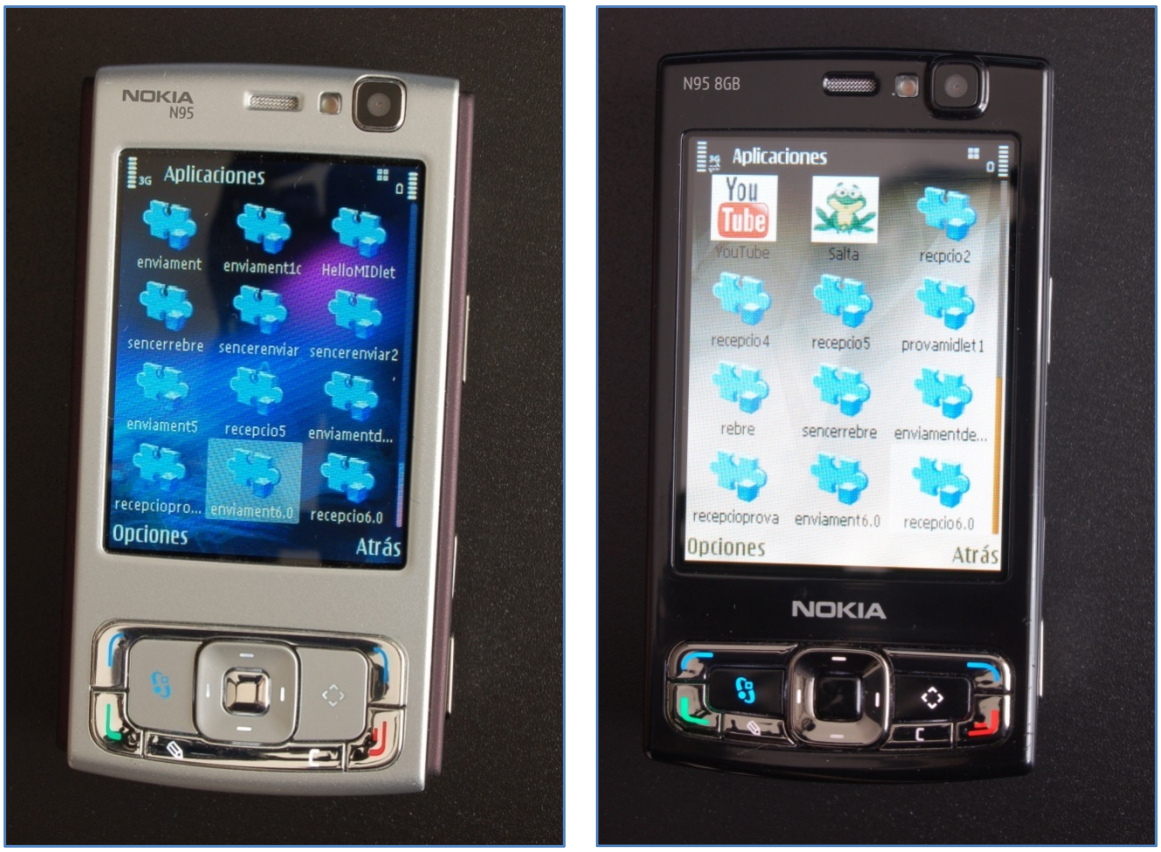


Figura 6.2.1: Menús

En aquestes primeres captures incorporades a la **figura 6.2.1**, **Menús**, podem visualitzar els menús d'aplicacions d'ambdós telèfons. Tot i que no es pot apreciar amb total nitidesa, al mòbil de l'esquerra hi està seleccionat el MIDlet "enviament6.0" i a dispositiu de la dreta hi està el MIDlet "recepcio6.0".

El següent pas serà entrar dins de cada MIDlet per poder fer servir les seves funcions. Una vegada s'han obert els respectius MIDlets, en tots dos dispositius s'ha d'autoritzar a l'aplicació "pfc2130406" a fer servir la xarxa. No s'especifica la xarxa en qüestió ja que ho pregunta a continuació, tal i com es mostra a les captures corresponents a la **figura 6.2.2**, **Confirmació**. Com el programa fa servir una connexió per enviaments de Datagrames sobre direccionament IP, només ens apareixeran el ventall de xarxes que funcionen amb aquest protocol. És per aquesta raó que no es preguntarà si es desitja fer servir, per exemple, el port d'infrarojos o una connexió *bluetooth*.

Figura 6.2.2: Confirmació

En les següents captures de pantalla que queden reflectides a la **figura 6.2.3**, **Salta2.0**, hi tenim els punts d'accés que hi ha enregistrats a cada mòbil. Farem servir la connexió "Salta2.0" que ja havia estat configurada prèviament, i que recordem que és de tipus ad-hoc. Com ja sabem, a cada mòbil se li ha assignat una direcció IP diferent, 192.168.1.1 per al dispositiu de l'esquerra o emissor, i 192.168.1.2 per al de la dreta, que és el receptor.

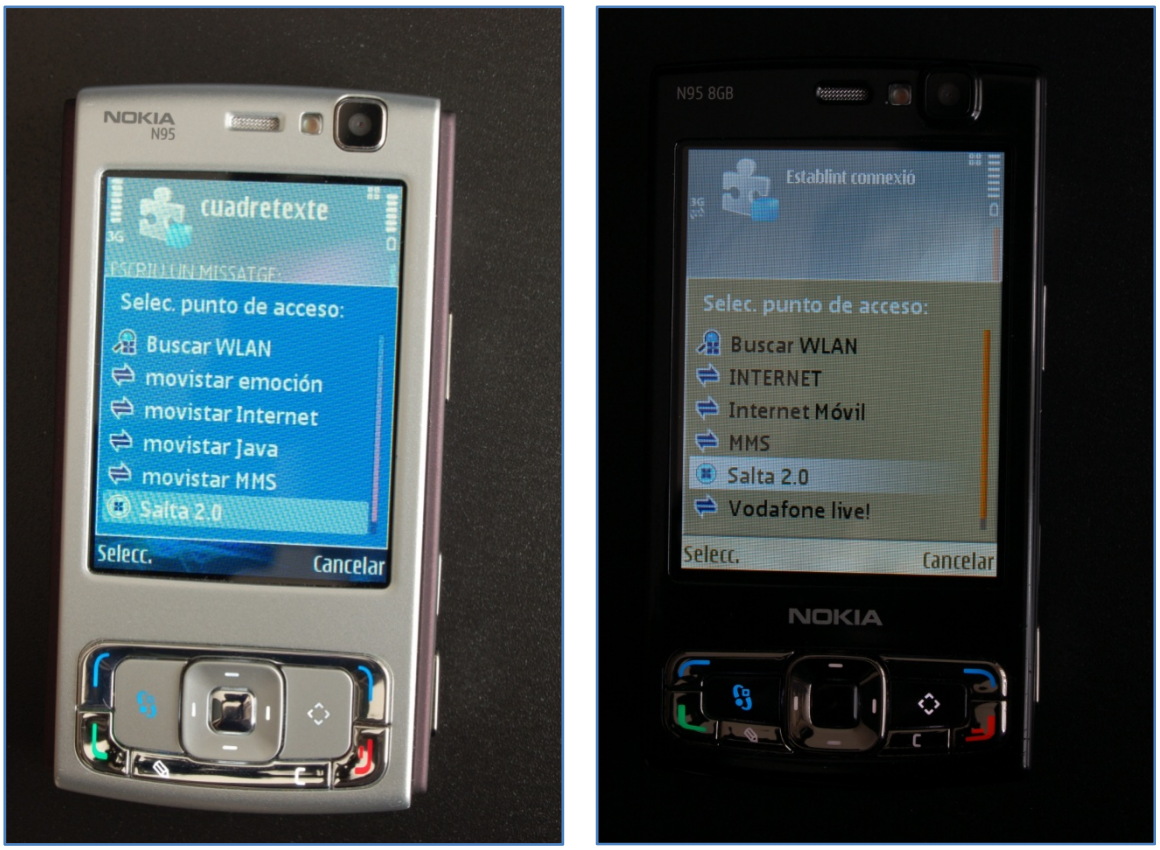


Figura 6.2.3: Salta 2.0

El següent pas que hem de seguir és diferent per a cada dispositiu.

Mentre que al mòbil emissor s'ens demana que escriguem un missatge, al receptor de la dreta s'està establint la connexió, tal i com es visualitza a la **figura 6.2.4**, **Connexió**, i una vegada establerta s'informa del fet, tal i com es veu en la foto inferior incorporada a la **figura 6.2.5**, **Espera** , on es pot llegir que està esperant rebre el Datagrama.

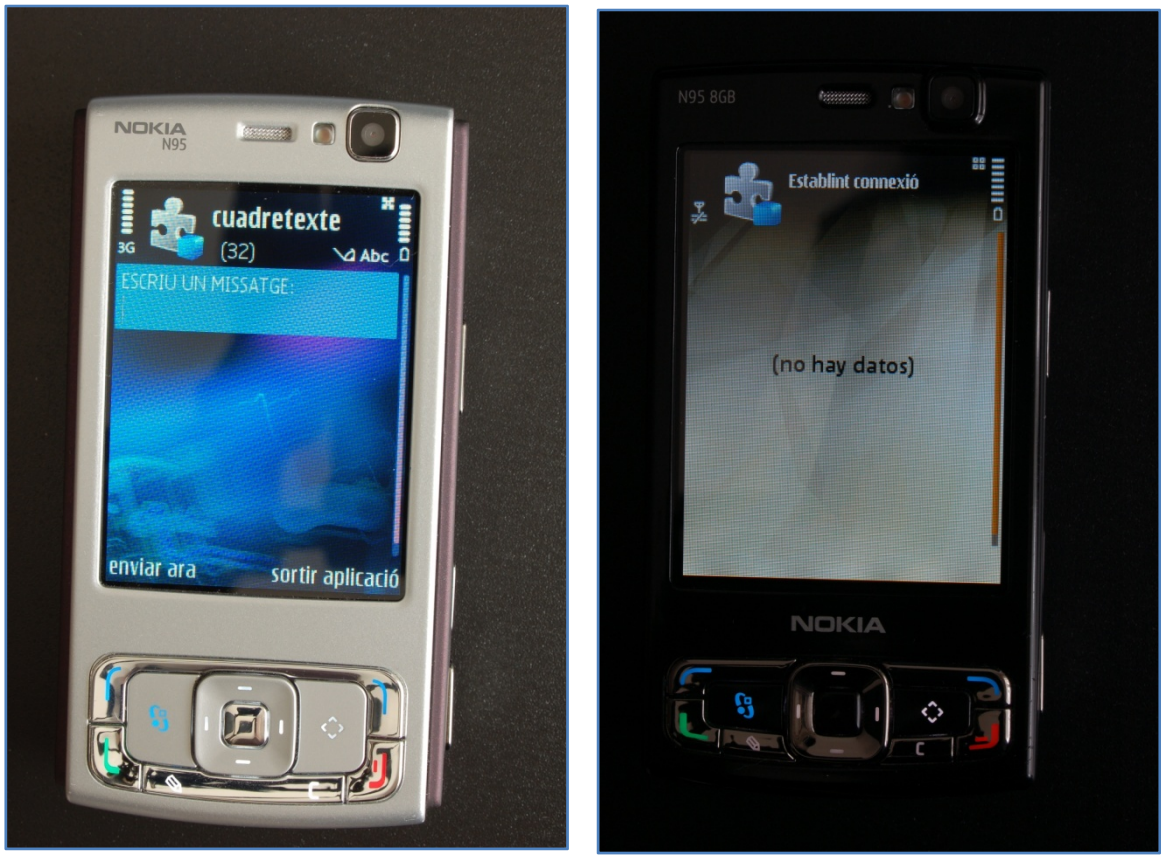


Figura 6.2.4:Connexió

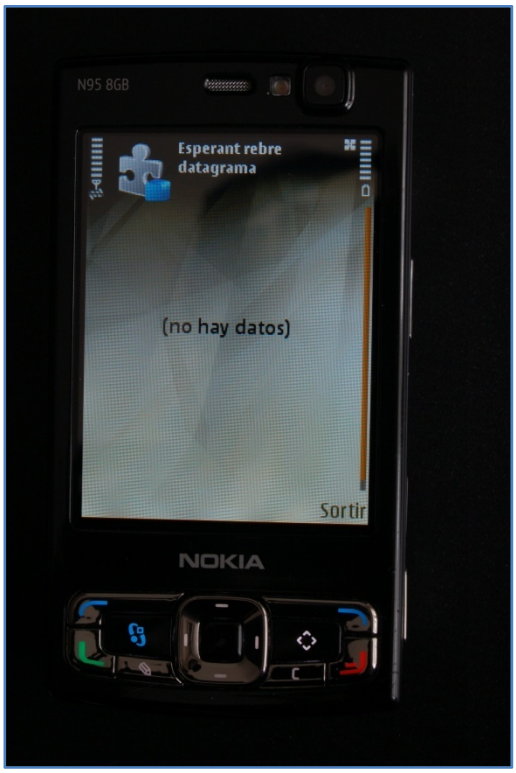


Figura 6.2.5: Espera

Doncs bé, només queda escriure el missatge. Com s'ha comentat anteriorment, hem d'omplir fins a 16 caràcters i després, de forma automàtica, s'enviarà el missatge.

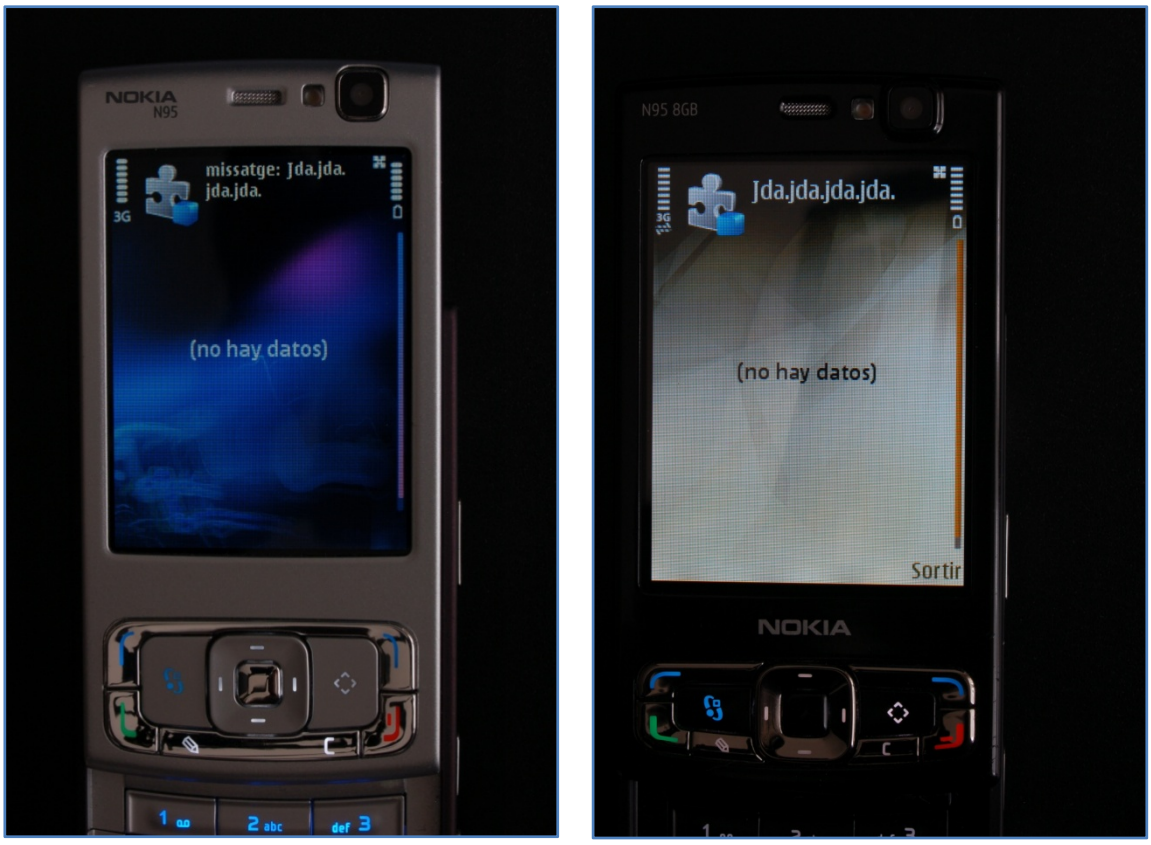


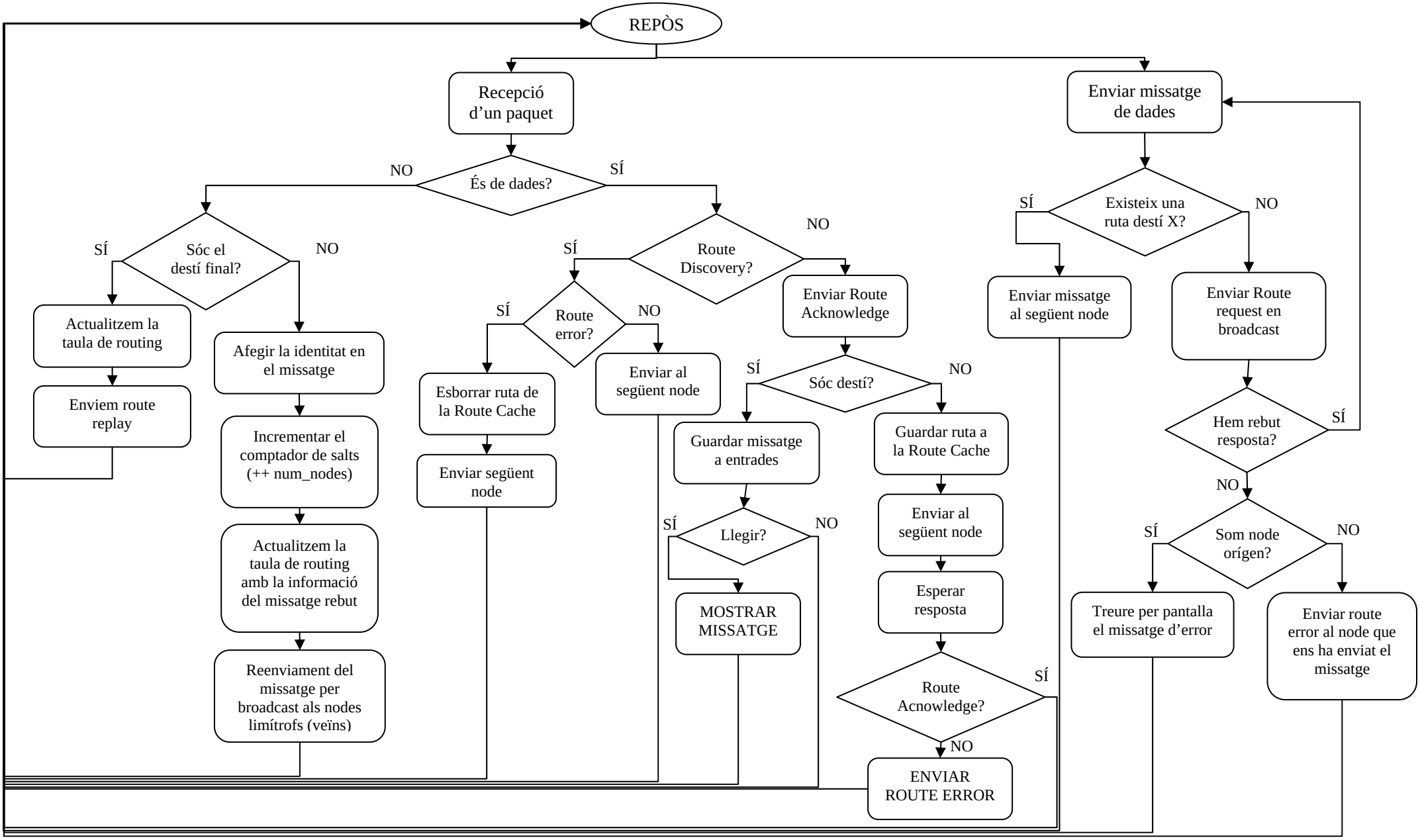
Figura 6.2.6: Missatge

Una vegada completat els 16 caràcters, en el emissor es passa a la següent pantalla on es mostra el missatge i, de forma immediata, el dispositiu receptor vibra, s'encén la retroil·luminació de la pantalla (si estava apagada) i es mostra el missatge per pantalla. La **figura 6.2.6**, **Missatge**, mostra aquests fets i el missatge enviat, <<Jda.Jda.Jda.Jda.>>

6.3 DIAGRAMA DE FLUX

A continuació presentem el diagrama de flux que inclou les diferents casuístiques amb les que ens podem trobar en el sistema de missatgeria que ens ocupa. Tot i això, aquest diagrama representa el projecte inicial complert, però la part realitzada només n'és una part d'ell.

Tal i com es pot observar, es parteix d'una situació de repòs. A partir d'aquí ens podem trobar en dues situacions: que l'usuari es disposi a enviar un missatge de dades, o bé que en el dispositiu mòbil es produeixi la recepció d'un paquet. Els diferents passos a seguir segons sigui un cas o l'altre queden reflectits en ell.


REPÒS
Recepció d'un paquet
És de dades?
NO
SÍ
Sóc el destí final?
SÍ
NO
Actualitzem la taula de routing
Enviem route replay
Afegir la identitat en el missatge
Incrementar el comptador de salts (++ num_nodes)
Actualitzem la taula de routing amb la informació del missatge rebut
Reenviament del missatge per broadcast als nodes limítrofs (veïns)
Route Discovery?
SÍ
NO
Route error?
SÍ
NO
Esborrar ruta de la Route Cache
Enviar següent node
Enviar al següent node
Enviar Route Acknowledge
Sóc destí?
SÍ
NO
Guardar missatge a entrades
Llegir?
SÍ
NO
MOSTRAR MISSATGE
Guardar ruta a la Route Cache
Enviar al següent node
Esperar resposta
Route Acnowledge?
SÍ
NO
ENVIAR ROUTE ERROR
Enviar missatge de dades
Existeix una ruta destí X?
SÍ
NO
Enviar missatge al següent node
Enviar Route request en broadcast
Hem rebut resposta?
SÍ
NO
Som node orígen?
SÍ
NO
Treure per pantalla el missatge d'error
Enviar route error al node que ens ha enviat el missatge

7. APLICACIÓ

Com ja hem comentat anteriorment, l'aplicació definitiva s'ha batejat amb el nom de "pfc2130406", i consta de dos MIDlets, un per l'enviament i altre per a la recepció, "enviament6,0" i "recepció6.0" respectivament.

Una vegada instal·lada l'aplicació apareixen dues noves icones al menú del mòbil, cadascuna d'elles es corresponen amb un dels MIDlets. Ambdós, encara que estan inclosos en una mateixa aplicació, podrien ser-ne dos d'independents.

Els MIDlets de les aplicacions s'han elaborat a partir de l'opció visualMIDlet i no com un MIDlet de codi en blanc. La diferència és que l'IDE NetBeans dóna l'oportunitat d'afegir algunes opcions a partir de la paleta, arrossegant-les fins la finestra "Device Screen" que representa la pantalla del dispositiu, i a partir d'aquest punt es pot anar amb l'opció "Go to source" a la part del codi font on està i editar-lo per fer les modificacions oportunes.

Les emulacions s'han realitzat primer des de l'IDE, però estem tractant una aplicació d'emissió i recepció, i donat que l'emulador no permetia comprovar l'enviament i la recepció correctes del missatge, s'ha hagut de fer una comprovació directament en els dispositius físics, per a continuació fer un "*build*" i després instal·lar l'arxiu ".jar" al mòbil i així validar el funcionament per a cada modificació que s'ha anat fent.

D'altra banda, ens hem trobat amb el problema de què algunes modificacions no eren possibles, ja que existien predeterminacions establertes pel visualMIDlet amb la impossibilitat d'edició, inclús no hi havia l'opció de comentar-les. Cal afegir que és probable que al codi font dels MIDlets hi hagi algunes sentències que no s'utilitzen, això és degut a que el propi IDE les afegeix, però no tenen cap utilitat i, malgrat tot, no donen problemes en compilar, fent innecessària la depuració.

Tal i com ja hem anat analitzant, estan implementades en J2ME, configuració CLDC-1.1 i MIDP-2.0, per tant, l'aplicació pot instal·lar-se en gran varietat de mòbils, fins i tot als dispositius més recents amb configuracions MIDP-2.1.

Passem ara a parlar de l' "enviament6.0":

Aquest MIDlet té l'opció d'editar el missatge des del mòbil, la mida predeterminada és d'un *string* de 16 caràcters, però és fàcilment configurable a més o menys caràcters des del el codi font. Per tal d'aconseguir-ho només cal canviar una variable de tipus sencer ("int"), i que es troba ubicada a la sentencia de la línia 401, "while (textedeldgram1.size() < 16)", simplement s'ha de modificar el valor de 16 per la mida desitjada.

D'altre banda, obre una connexió abans d'enviar el Datagrama, amb la qual cosa es demana permís per poder fer servir la xarxa Wi-Fi. Aquesta connexió es la següent: 192.168.1.255:32767. La primera part és la direcció IP i el número que hi ha després dels dos punt és el port. Pel que fa a la direcció IP, observis que acaba en 255, això significa que farà un *broadcast*, és adir, totes les direccions IP que comencin per 192.168.1. rebran el mateix missatge. Si es volgués fer una connexió amb un dispositiu en concret es pot configurar fàcilment en el codi font canviant la direcció IP en la sentencia de la línia 391: "...(DatagramConnection) Connector.open("datagram://192.168.1.255:32767");".

La xarxa que fa servir el MIDlet és una xarxa ad-hoc que es va configurar en el mòbil per poder fer servir una altre aplicació, però no està inclòs dins del codi de la aplicació pfc2130406, s'ha de configurar en el propi menú de cada mòbil que tingui connexió Wi-Fi.

Abordem ara la "Recepcio6.0":

El MIDlet de recepció també fa servir una connexió basada en Datagrames, tal i com fa l'enviament, però en el codi font no s'estableix la direcció IP, ja que pot rebre des de qualsevol direcció, en canvi sí que està indicat el port. Aquest MIDlet està fet per rebre Datagrames i mostrar-los per pantalla, de forma que quan ha rebut un missatge fa una senyal de vibració, s'il·lumina la pantalla i simultàniament es visualitza el missatge.

Per rebre un missatge ha d'estar el MIDlet activat, en cas contrari no el rep. Quan s'activa el MIDlet ens demana autorització per fer servir la connexió Wi-Fi, una vegada autoritzat s'ha d'escollir una de les xarxes que ja estan registrades, i el procés a seguir és idèntic al de l'enviament.

Ara parlarem de quins són els avantatges i desavantatges de l'aplicació:

Alguns desavantatges tenen solució, però també s'ha de tenir en compte que els objectius inicials del projecte eren molt ambiciosos i difícilment realitzables dins del termini de temps estipulat. Aquests desavantatges són les següents:

- A la part de l'enviament no hi ha la possibilitat d'escollir la direcció IP des del mòbil, s'ha de fer des del codi font.

- Des del mòbil no es pot editar la mida dels missatges.

- El consum de la bateria augmenta considerablement, sobretot en la recepció. S'ha provat de deixar actiu el MIDlet de recepció en el dos dispositius que s'han fet servir per realitzar el projecte. El primer és un Nokia N95 amb una bateria BL-5F de 950mAh i 3'7V, ha pogut romandre encès 53 minuts. L'altre dispositiu és un Nokia N95 8GB amb una bateria BL-6F de 1200mAh i 3'7V, aquest ha durat 47 minuts. S'han fet les proves amb les bateries acabades de carregar suposadament al màxim i, a més a més, els dos dispositius tenen aproximadament la mateixa antiguitat i són pràcticament iguals.

- Encara no s'ha implementat la possibilitat de memoritzar els missatges ni una agenda on guardar els contactes amb la seva IP corresponent.

- Tot i que fa servir una xarxa ad-hoc i està basat en un sistema salt a salt, només s'ha arribat a realitzar un MIDlet de transmissió i un de recepció de mòbil a mòbil.

Els avantatges són, en part, resultat dels desavantatges anteriors, veiem-ho:

- Es tracta d'una aplicació amb moltes possibilitats d'ampliació.

- Presenta la característica de l'escalabilitat.

- La part d'enviament, recepció i el *broadcast* funcionen correctament, per tant, amb certes modificacions o ampliacions i a partir del que ja està confeccionat es pot arribar a fer una aplicació molt més complerta.

- Utilitza una connexió Wi-Fi, això fa que tant l'enviament com la recepció siguin totalment gratuïts.

- És fàcilment aplicable en molts contextos, i per tant presenta moltes utilitats, tant empresarials com quotidianes, ja que permet l'enviament de dades a distàncies més llargues que les que possibilita un sistema basat en *bluetooth*.

Finalment, farem referència a altres aplicacions existents:

Ja hem explicitat que aquest projecte està basat en una versió comercial anomenada "Salta2.0", la qual es podia adquirir des de la pàgina web de la empresa que el comercialitzava, www.salta.es, per un import de descarrega de 40€. Actualment no està disponible, ja que l'aplicació va ser venuda a una altra empresa que de moment no la comercialitza. De fet, aquesta aplicació també va ser el resultat d'un projecte realitzat per varis alumnes de la Universitat de Cartagena.

La nostra aplicació està realitzada per poder funcionar amb la mateixa xarxa Wi-Fi que el Salta2.0, però són totalment incompatibles. D'altre banda, l'aplicació que s'ha confeccionat en aquest projecte està realitzada fent servir llibreries i classes públiques a l'abast de qualsevol programador, en canvi, l'aplicació "Salta2.0" fa servir classes privades fetes per els mateixos projectistes, i per tant no dóna l'oportunitat de modificar res, amb la complicació afegida de què el codi font corresponent no incorpora comentaris que permetin conèixer el funcionament de l'aplicació.

8. CONCLUSIONS

Originàriament els objectius del present projecte eren d'una envergadura gran i ambiciosa, sobretot si tenim en consideració la duració del treball que s'emmarca en un curs lectiu.

Tot i que aparentment semblava que estàvem fent un projecte que abordava un tema molt actual, trobar informació referent a aquesta temàtica va resultar una tasca àrdua, comportant molta feina, tant pel que fa a la documentació referent a JAVA per a mòbils, o J2ME, com als IDE's per implementar els MIDlets.

Pràcticament no s'han trobat llibres físics que tractin la matèria del projecte de manera profunda. La documentació més complerta es troba a pàgines en anglès d'Internet i a les web *sites* oficials de descarrega dels IDE's. Sovint, alguns tutorials publicats i pàgines personals no fan servir dades contrastades, induint al lector a confusions i a invertir temps en depurar codis font d'exemple erronis.

La major part del temps i dels esforços s'han invertit en la recerca d'informació, en l'elaboració d'estudis de mercat i a fer l'anàlisi d'estudis de viabilitat per justificar els protocols i llenguatges d'implementació. Tasques com les descarregues, la comprovació i la comparació dels diferents IDE's han comportat molta feina, i el disseny teòric dels algoritmes de funcionament també ha requerit molta inversió de temps.

Una vegada fet tot això ha quedat poc temps per a la implementació de la aplicació, malgrat tot, s'ha pogut completar la part fonamental del projecte, podent realitzar l'enviament tant en mode *broadcast* com a una direcció IP concreta, a més de rebre correctament el missatge i mostrar-lo per pantalla.

De tot això es desprèn que aquest projecte pot ser modificat i ampliat en cursos posteriors per d'altres alumnes que poden desenvolupar els canvis pertinents.

9. LÍNIES DE FUTUR

La tasca que s'ha dut a terme amb aquest projecte pot ser estesa i ampliada, per exemple, en projectes posteriors. Alguns dels aspectes principals sobre els que es pot treballar serien aquests:

- Desenvolupament de noves versions de l'aplicació que la fessin extensible a d'altres dispositius mòbils amb els que, ara per ara, no és compatible.

- Ampliar també l'abast de compatibilitat a ordinadors portàtils i PDA's.

- Tractar i treballar el tema de la seguretat de les dades, ja que es un aspecte que en el present treball no s'ha tingut en consideració.

10. APÈNDIX I: INSTAL·LACIÓ DEL NETBEANS

Anem a concretar amb una mica més de detall quins són els passos a realitzar per tal de dur a terme les instal·lacions corresponents. La **figura I.1**, **Instal·ladors**, mostra els diferents programes que es poden descarregar des de la pàgina web oficial de Sun:

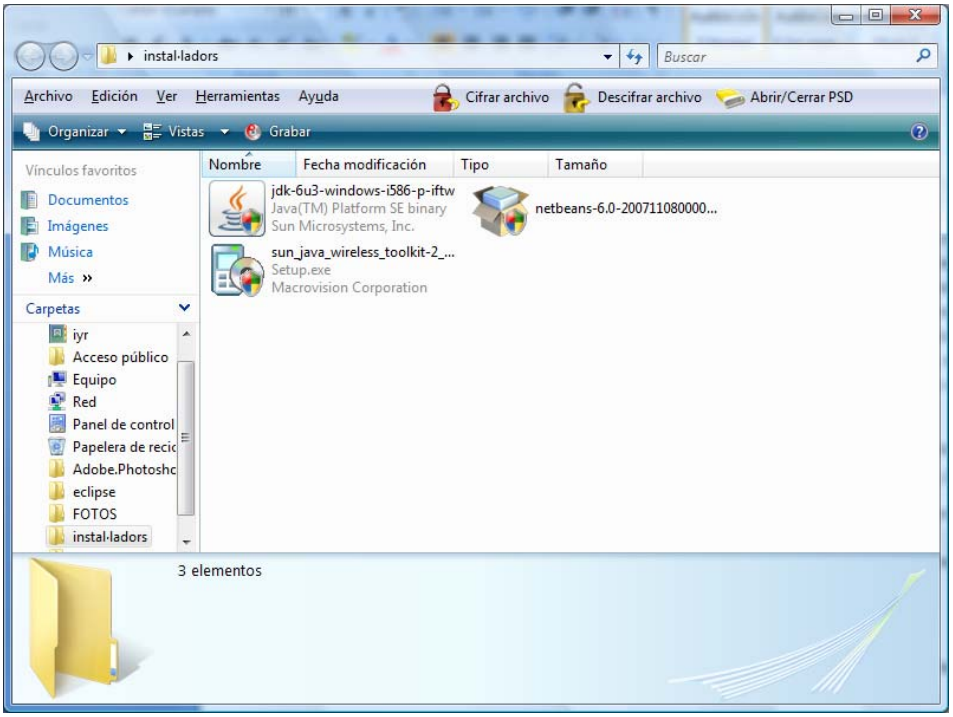


Figura I.1: Instal·ladors

Tal i com s'ha esmentat abans, en primer lloc hem d'instal·lar el J2SDK, que es correspon amb la icona ubicada a la cantonada superior esquerra de l'anterior captura de pantalla. Un cop fet això tindrem un directori nou, més concretament el de la ruta que s'hagi especificat en el diàleg de la instal·lació. Per defecte s'escull el directori arrel de la memòria principal, en el nostre cas c:/.

Un cop finalitzat el diàleg de la instal·lació podem passar a executar el *Wireless Toolkit*, que a l'anterior figura el tenim situat a la part inferior esquerra, és el fitxer de *setup* que rep el nom de sun_java_wireless_toolkit-2. Si fem servir l'explorador veurem un nou directori a la ruta especificada.

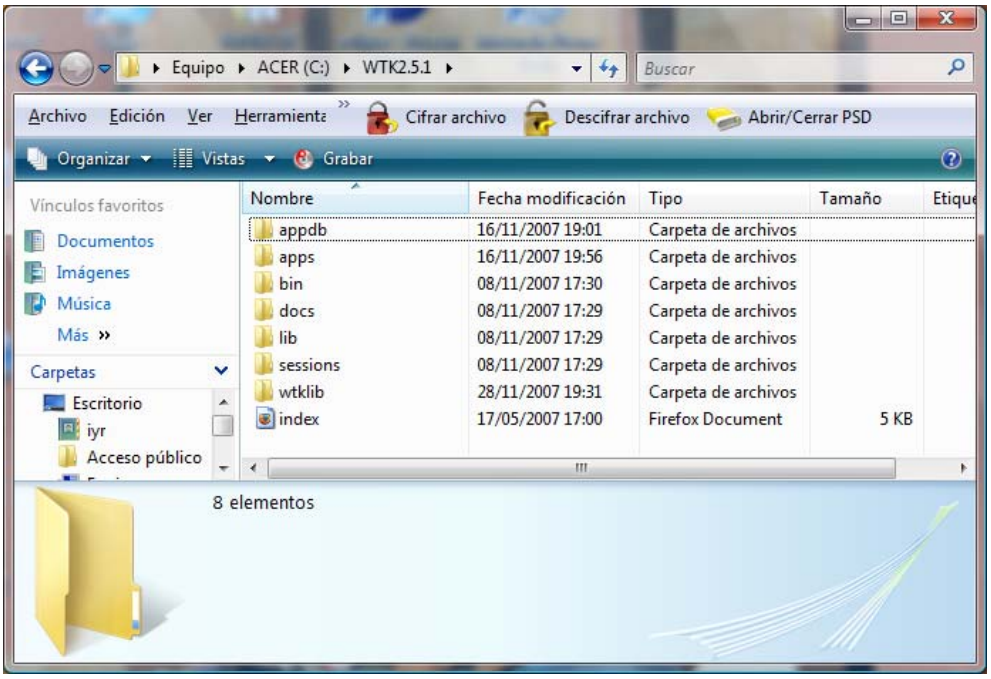


Figura I.2: WTK2.5.1

Tal i com podem veure a la **figura I.2**, **WTK2.5.1**, el directori es diu WTK2.5.1 i, tanmateix, consta de diferents subdirectoris: *appdb*, *apps*, *bin*, *docs*, *lib*, *sessions*, i *wtklib*.

Per poder executar el programa hem d'obrir la subcarpeta *bin* i executar l'aplicació que s'anomena *Ktoolbar*, tal i com podem apreciar a la **figura I.3**, **Bin**:

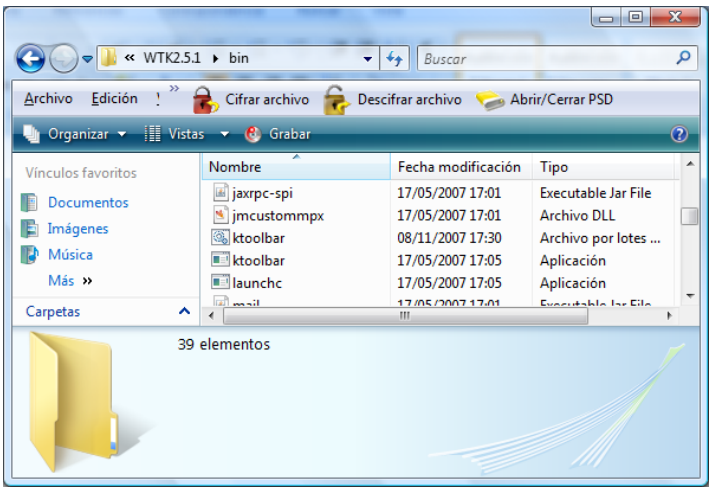


Figura I.3: Bin

Ara estem en disposició de fer servir l'eina *Wireless Toolkit for CLDC*, se'ns obrirà la següent imatge que mostra la **figura I.4**, **Wireless Toolkit for CLDC**:

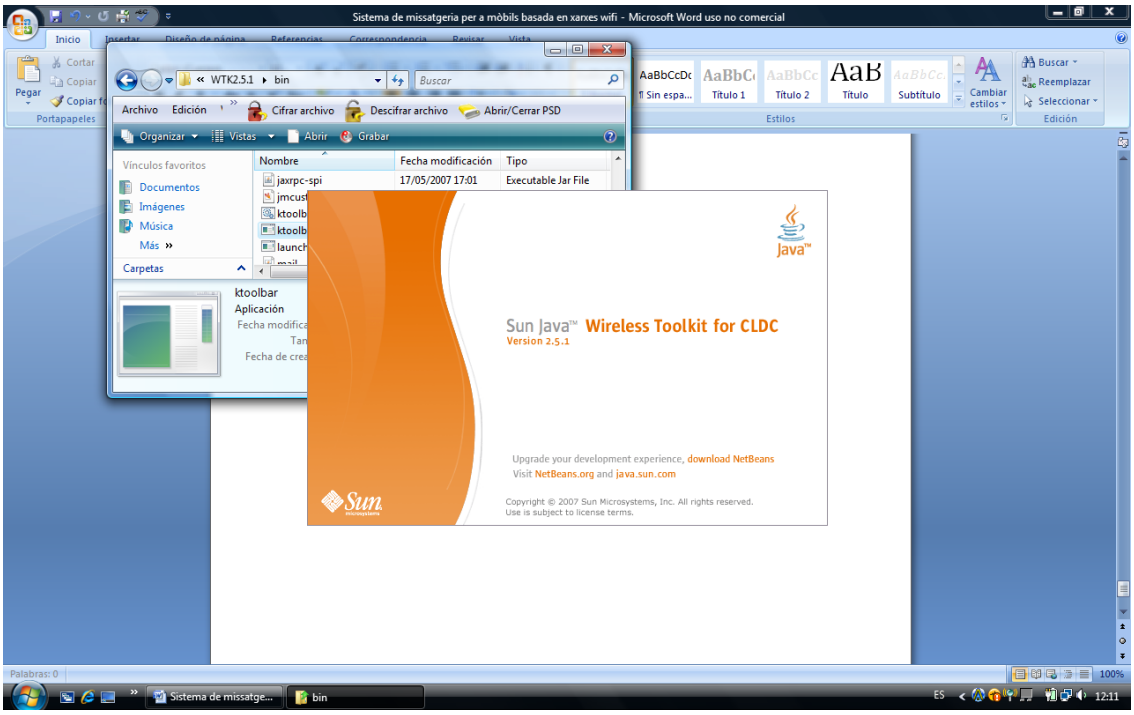


Figura I.4: Wireless Toolkit for CLDC

Que té el l'aspecte que mostra la **figura I.5**, **Aspecte del Wireless Tookit for CLDC**:

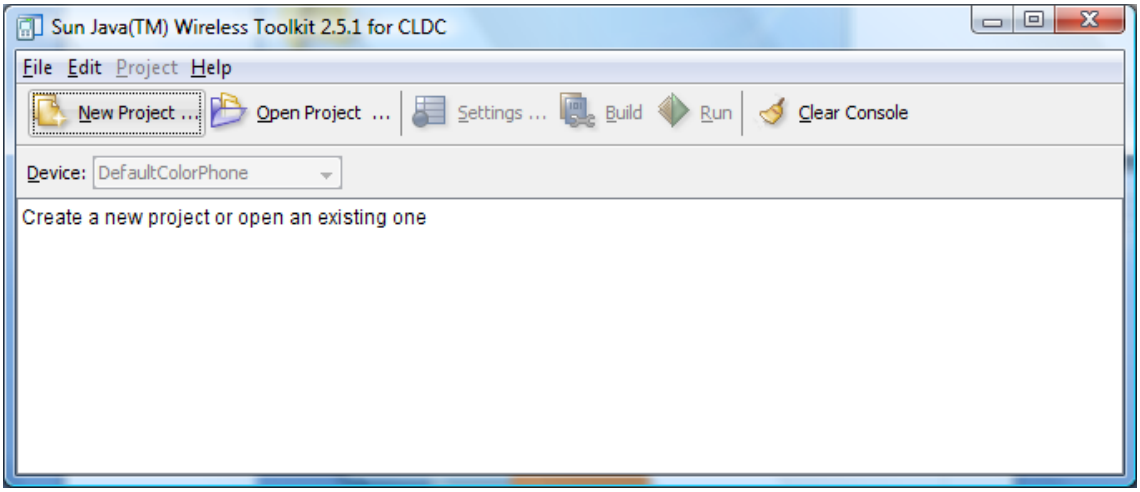


Figura I.5: Aspecte del Wireless Tookit for CLDC

Com es pot observar i hem comentat anteriorment, es tracta d'una eina força simple, tant és així que fins i tot no té editor de projectes. Tal i com es va esmenar, l'editor que es pot utilitzar pot ser una altra eina de la pròpia organització Sun, *Forte*, o bé una altra més concreta, com és el cas de *NetBeans*. Amb el *Wireless Toolkit* podem començar un projecte, però no el podem editar, és a dir, és possible obrir directoris de treball i crear els diferents subdirectoris necessaris per fer funcionar el projecte, però per poder realitzar edicions hem d'instal·lar un editor, aquí es considerarà el *Netbeans*. A continuació es descriuen els passos a seguir per a la seva instal·lació.

A la mateixa finestra on estan els dos anteriors instal·ladors hi trobem una altra icona que porta per nom netbeans-6.0-2007110800000... Si hi fem doble *click* se'ns obrirà un altre diàleg d'instal·lació automàtica, tal i com es veu a la **figura I.6**, **Instal·lador NetBeans**:

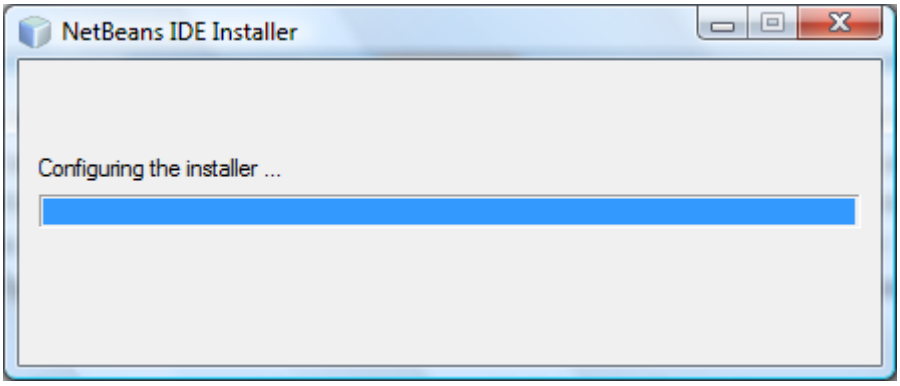


Figura I.6: Instal·lador NetBeans

La **figura I.7**, **Benvinguda a la instal·lació**, ens dona l'oportunitat de seleccionar què volem instal·lar. A tal fi, simplement hem de fer *click* sobre el botó *Customize*...:

Figura I.7: Benvinguda a la instal·lació

Hem de tenir totes les caselles seleccionades, tal i com es mostra a la **figura I.8**, **Customize Installation**:

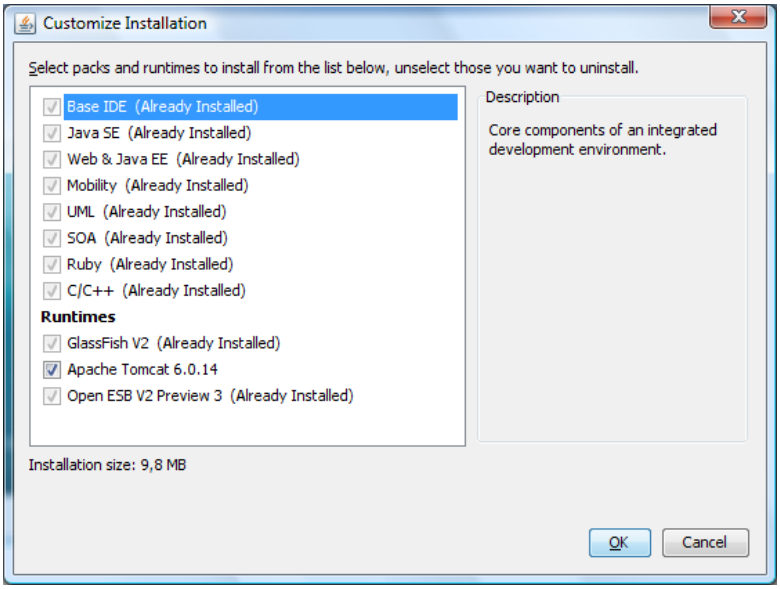


Figura I.8: Customize Installation

Aleshores, se'ns torna a donar la benvinguda a la instal·lació, tal i com es mostra a la **figura I.9**, **NetBeans IDE Installer**:

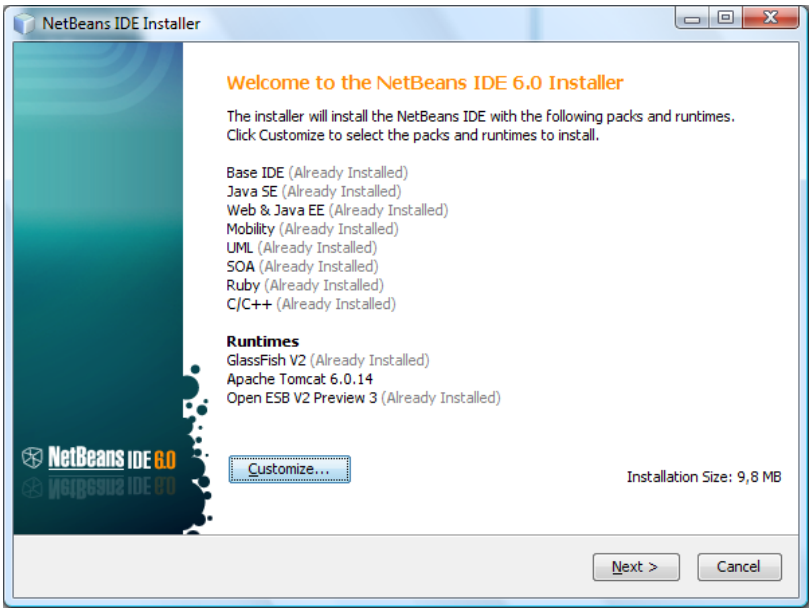


Figura I.9: NetBeans IDE Installer

Si polsem el botó *Next*, se'ns obrirà una finestra amb les condicions de la llicència, condicions que hem d'acceptar per poder realitzar la instal·lació, la **figura I.10**, **Condicions de llicència**, ho mostra:

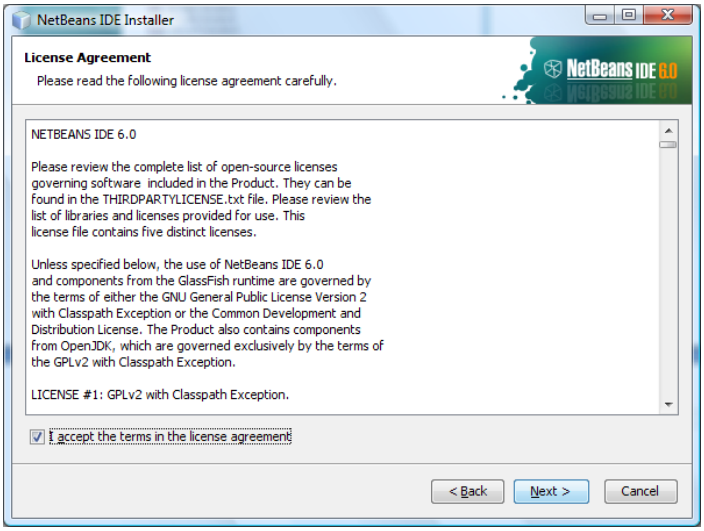


Figura I.10: Condicions de llicència

La **figura I.11**, **Directori**, mostra la finestra que ens demana el directori on volem realitzar la instal·lació:

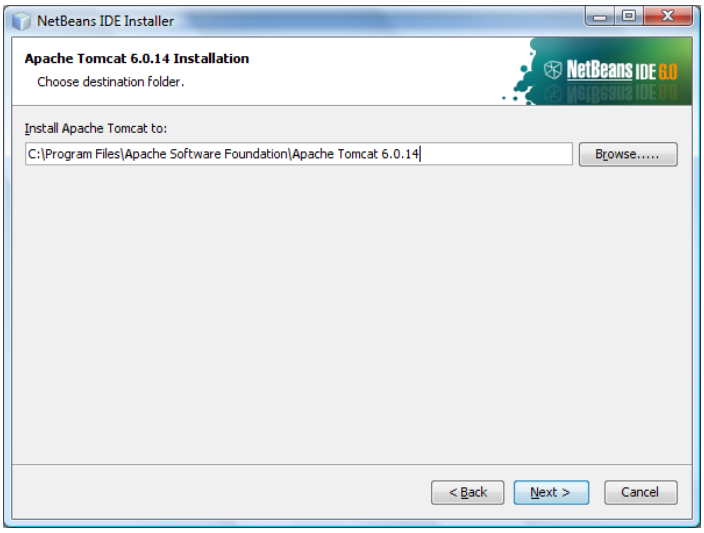


Figura I.11: Directori

I en aquest moment, ja simplement arribem a la finestra que ens dóna l'opció de procedir amb la instal·lació, cancel·lar, o tornar enrere per fer alguna modificació. La **figura I.12**, **Instal·lació**, n'és la imatge:

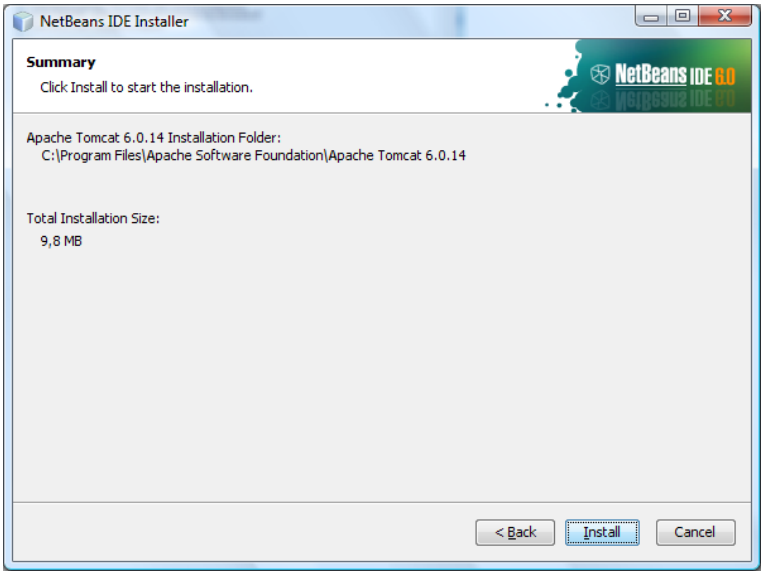


Figura I.12: Instal·lació

Si fem *click* sobre el botó *Install*, ens apareixerà una finestra amb una barra d'estat que avança a mida que transcorre el temps, tal i com es veu a la **figura I.13**, **Barra de temps**:

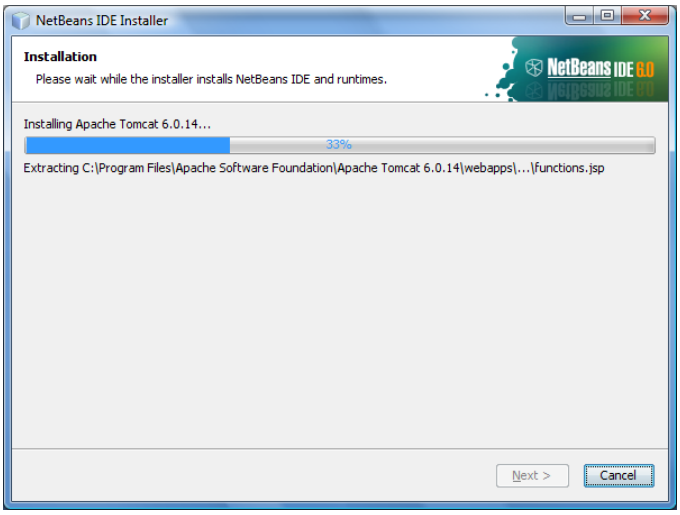


Figura I.13: Barra de temps

La finestra que mostra la **figura I.14**, **Instal·lació completada**, ens informa de què hem arribat al final de la instal·lació, ja només cal polsar el botó *Finish*:

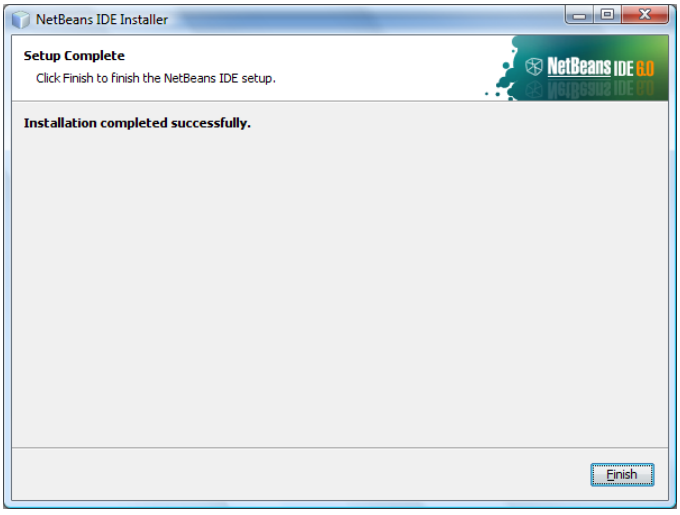


Figura I.14: Instal·lació completada

Tant el *Wireless Toolkit* com el *Netbeans* creen una icona a l'escriptori. Si fem doble *click* a la del *Netbeans* veurem que la seva aparença és la que veiem a la **figura I.15**, **NetBeans**:

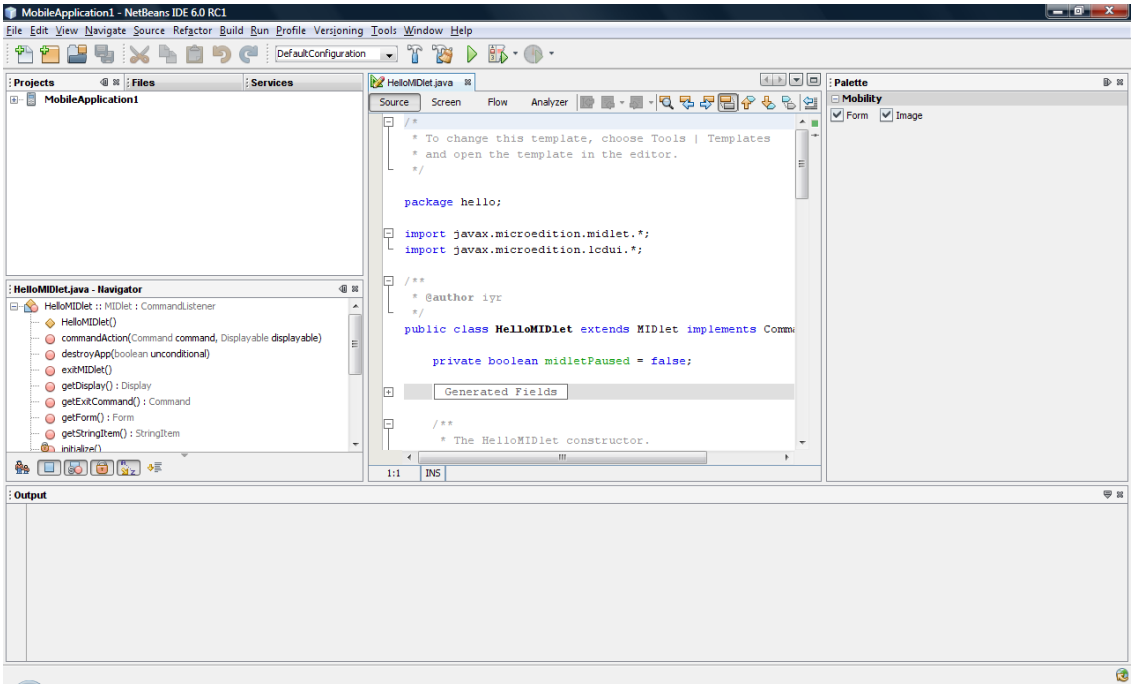


Figura I.15: NetBeans

Tal i com es pot observar a l'anterior captura de pantalla, disposem d'una aplicació que, finalment, ens permet editar el codi del projecte.

Ara que ja tenim instal·lades les eines que ofereix *Sun Microsystems* des de la seva pàgina web, anem a mirar d'altres opcions també disponibles. Passem doncs a aprofundir en la instal·lació del programa *Eclipse*.

11. APÈNDIX II: INSTAL·LACIÓ DE L'ECLIPSE

Per instal·lar l'Eclipse, des de la pàgina http://eclipse.org hem de buscar l'opció *downloads* en el quadre de l'esquerra, tal i com mostra la **figura II.1**, **Downloads**:

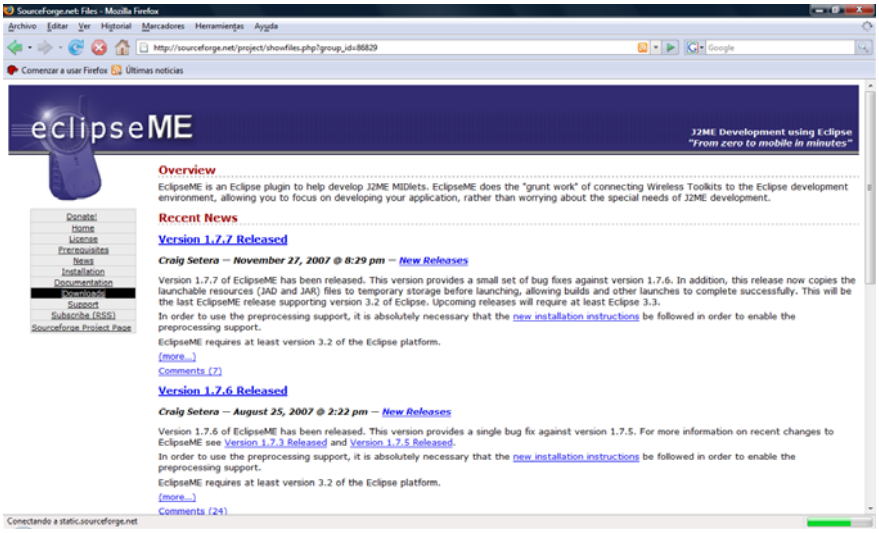


Figura II.1: Downloads

I per descarregar el programa simplement s'ha de polsar la opció *download* que es visualitza en color verd, tal i com el lector pot apreciar a la **figura II.2**, **Download EclipseMe**:

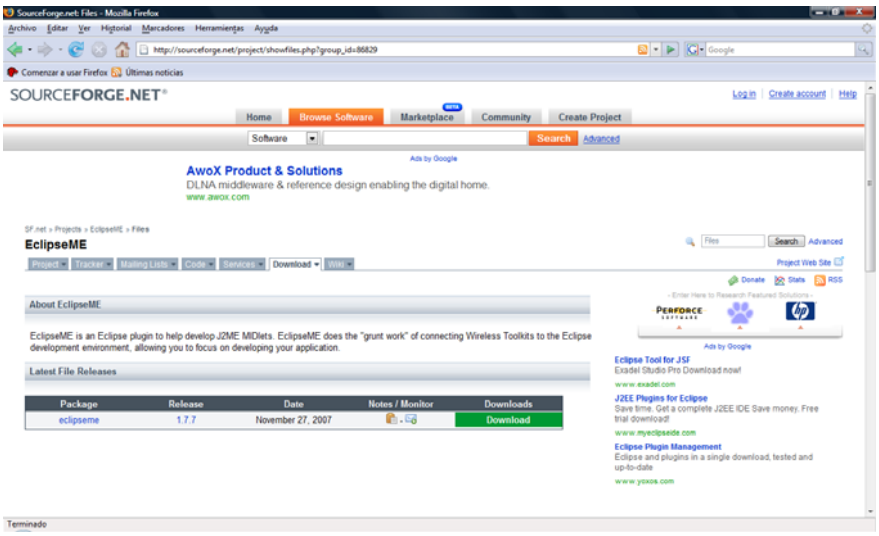


Figura II.2: Download EclipseMe

Amb això ens baixarem un arxiu comprimit que té el contingut, una vegada descomprimit, que es manifesta a la següent imatge, la **figura II.3**, **Contingut download EclipseMe**:

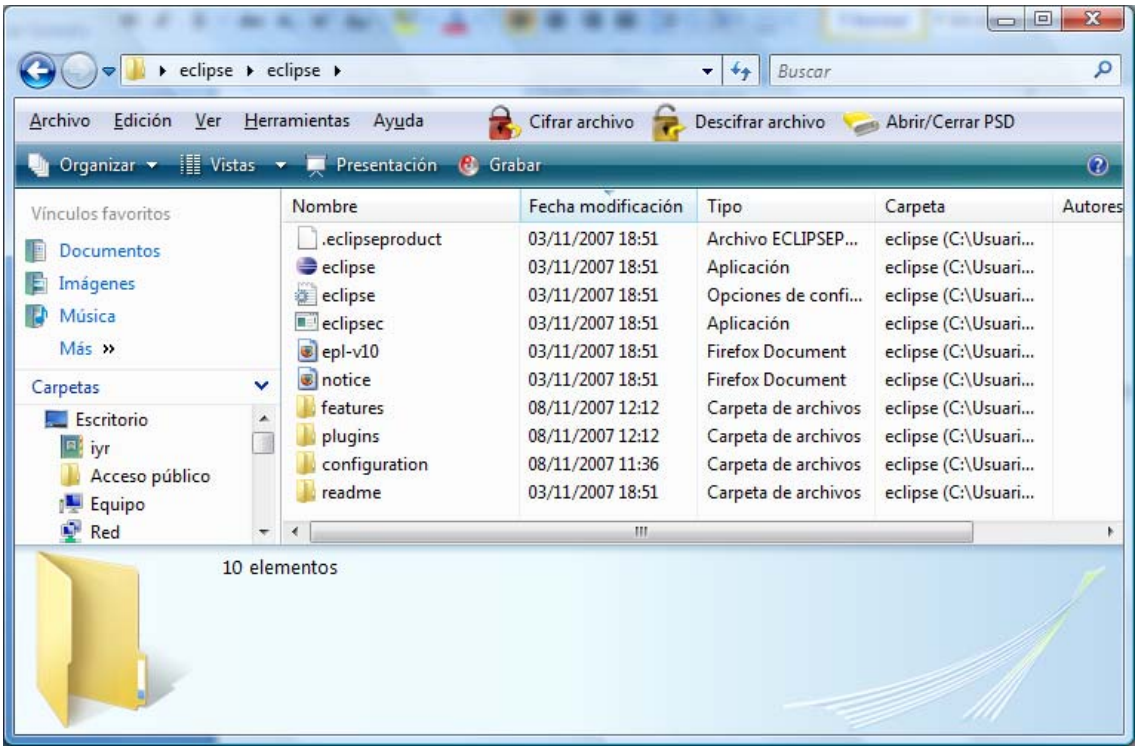


Figura II.3: Contingut download EclipseMe

Hem d'executar l'aplicació *Eclipse*, i amb això ja obtenim el programa obert amb un directori de treball en la ruta que se li hagi especificat.

Ara hem de passar a d'instal·lar el *plugin* de la versió MicroEdition, a tal fi seguirem un seguit de passos. La pantalla de benvinguda és la mostrada per la **figura II.4**, **Welcome to EclipseMe**:

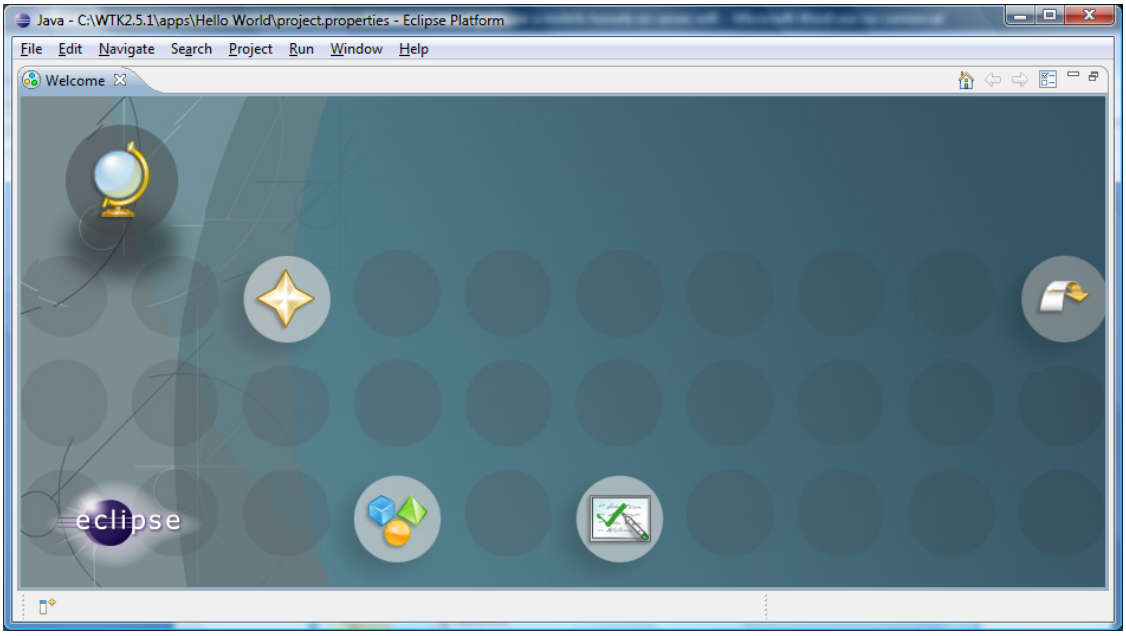


Figura II.4: Welcome to EclipseMe

Fem un *click* a la pestanya *Help*, seleccionem *software upgrate* i després *find and install*. Passarem a una pantalla que és la que queda reflexada a la **figura II.5**, **Install/Update EclipseMe**. En aquest punt, hem d'escollir la segona opció, *Search for new features to install*, i a continuació hem de fer *click* sobre el botó *Next*:

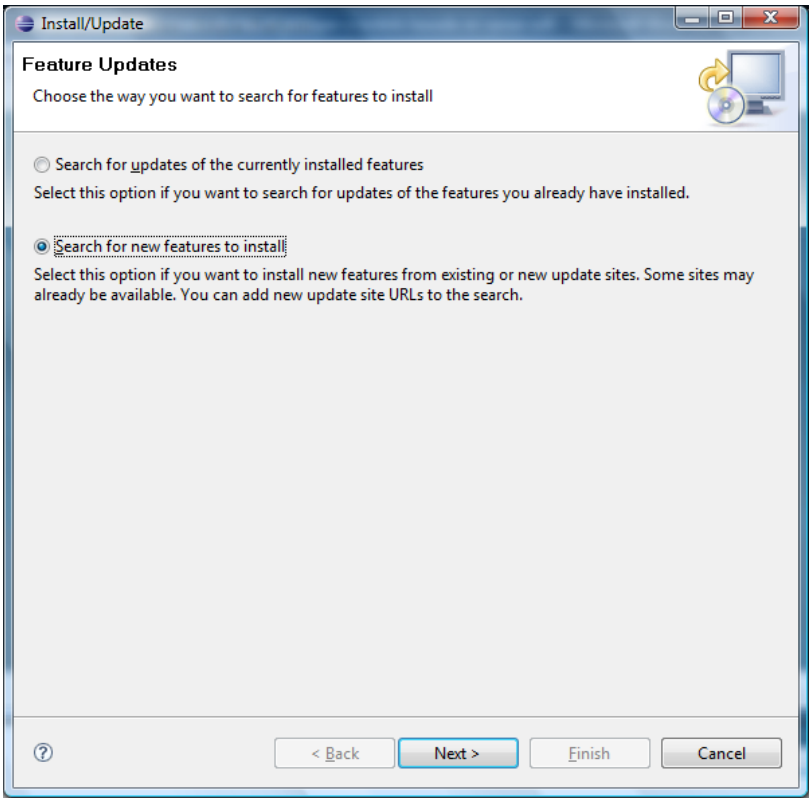


Figura II.5: Install/Update EclipseMe

Se'ns obrirà una finestra que ens demana els *sites* a incloure a la recerca d'informació per a la instal·lació. Hem d'escollir *EclipseMe Update Site*, i marcar la casella *Ignore features not applicable to this enviroment*, la **figura II.6**, **Update sites to visit**, ho mostra:

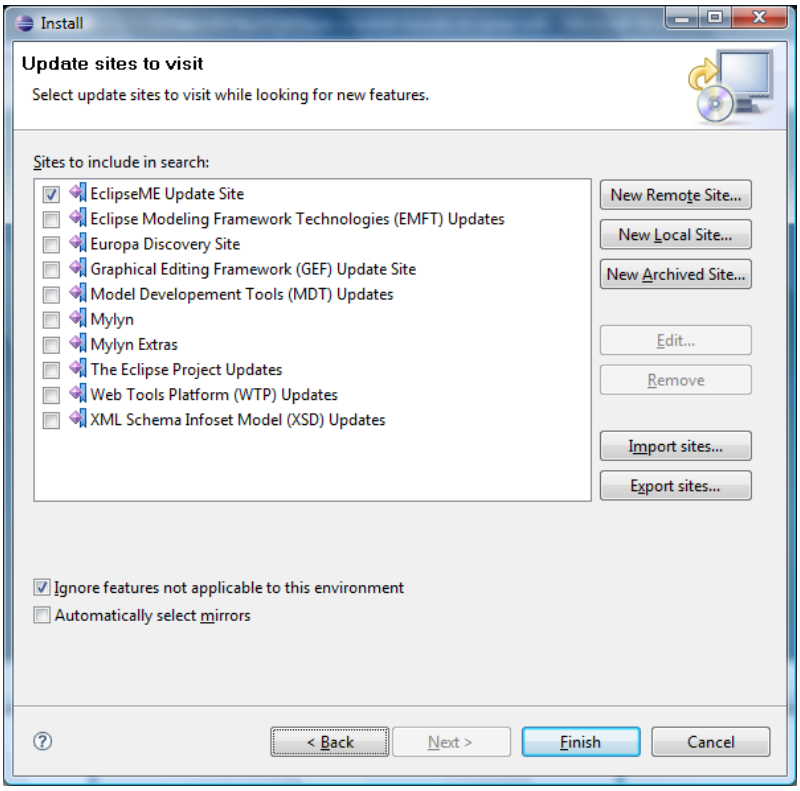


Figura II.6: Update sites to visit

Clickem el botó *New Remot Site...* a l'anterior quadre de diàleg, i ens apareixerà un nou quadre de diàleg com el que es mostra a la **figura II.7**, **New Update Site**. En ell hi hem d'omplir el camp *Name* amb *EclipseME Update Site* i el camp *URL* amb http://eclipseme.org/updates.

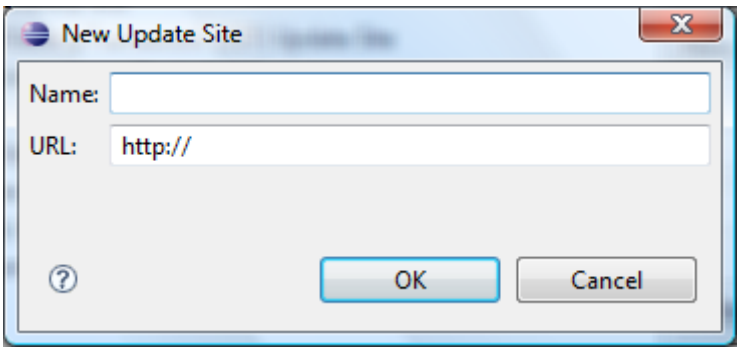


Figura 5.2.2.7: New Update Site

Aquest pas és necessari per tal que ens aparegui l'opció *EclipseME Update Site* al quadre exposat a la **figura II.6**. A la captura que hi hem incorporat a la redacció d'aquest text ja hi consta degut a què en l'*Eclipse* que es fa servir d'exemple ja hi és instal·lat a l'ordinador

des del qual s'està redactant la present memòria, però no seria el cas a una instal·lació nova, és a dir, una instal·lació a la que es parteix de zero.

Una vegada fet aquest pas, ens apareixerà la finestra que es mostra a la **figura II.8**, **Update sites to visit**, on s'ha de marcar l'opció *EclipseME Update Site* i a continuació polsar el botó *Next*:

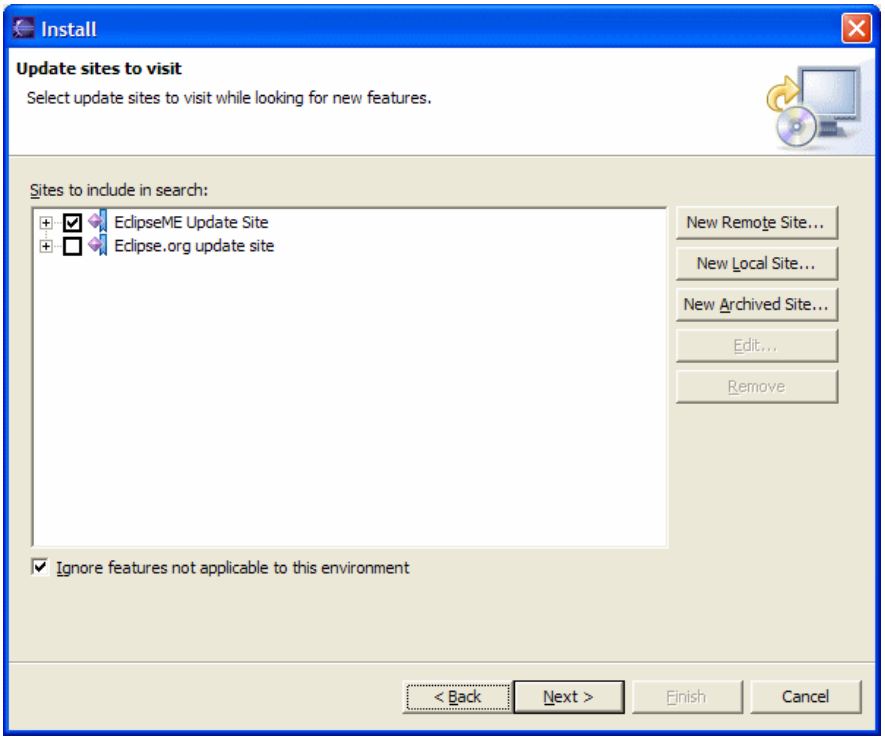


Figura 5.2.2.8: Update sites to visit

A la nova finestra que ens apareix, **figura II.9**, **Search Results**, seleccionem l'opció *EclipseME*, i a partir d'aquí fem *click* al botó *Next*.

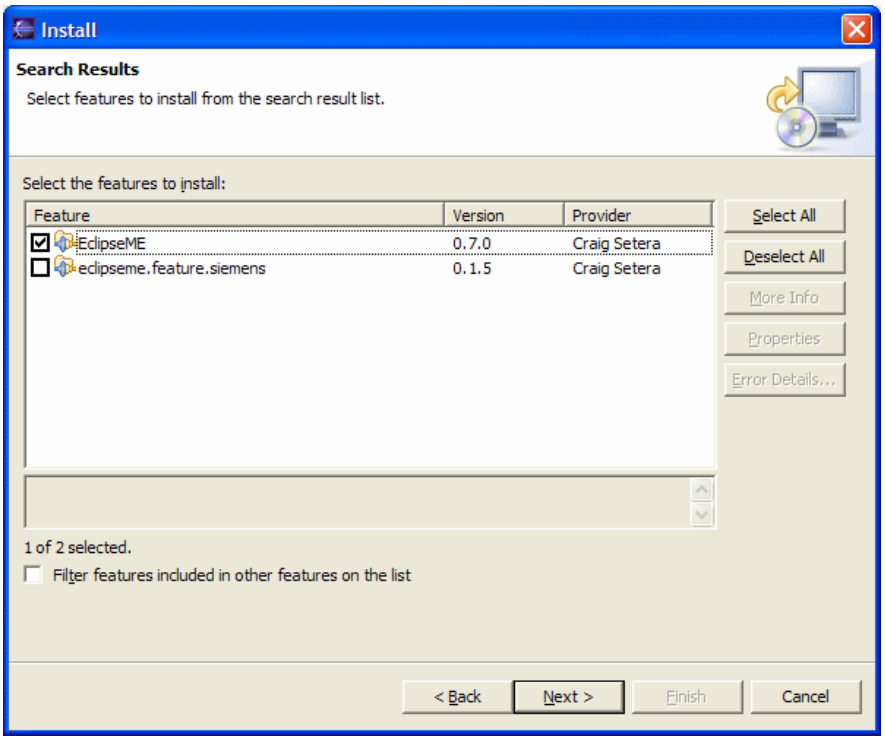


Figura II.9: Search Results

El següent pas que s'ha de realitzar és acceptar els termes de la llicència, i a partir d'aquí ja és possible continuar amb el botó *Next*, tal i com es mostra a la **figura II.10**, **Feature Licence**, on s'evidencia que s'ha de marcar la casella I *accept the terms in the license agreements*:

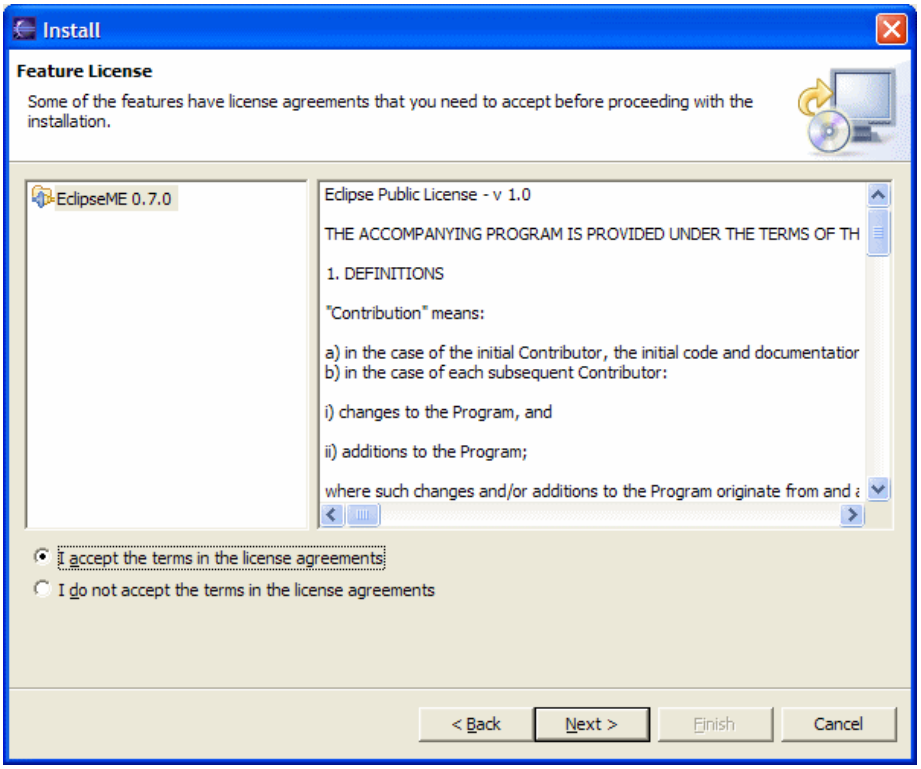


Figura II.10: Feature Licence

Finalment, la següent pantalla ens indica el directori on per defecte es durà a terme la instal·lació, si no el volem modificar simplement cal fer *click* sobre el botó *Finish*. La **figura II.11**, **Install Location**, ho exposa:

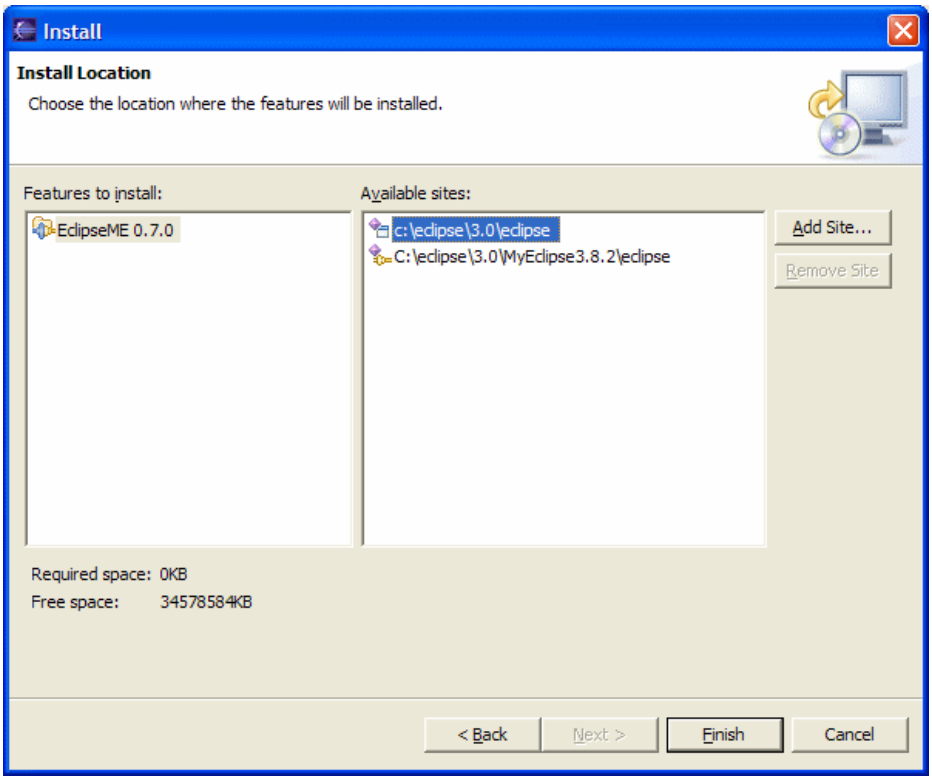


Figura II.11: Install Location

En aquest punt es començaran a instal·lar les opcions demanades. En acabar apareixerà una finestra com la que exposem a continuació a la **figura II.12**, **Feature Verification**:

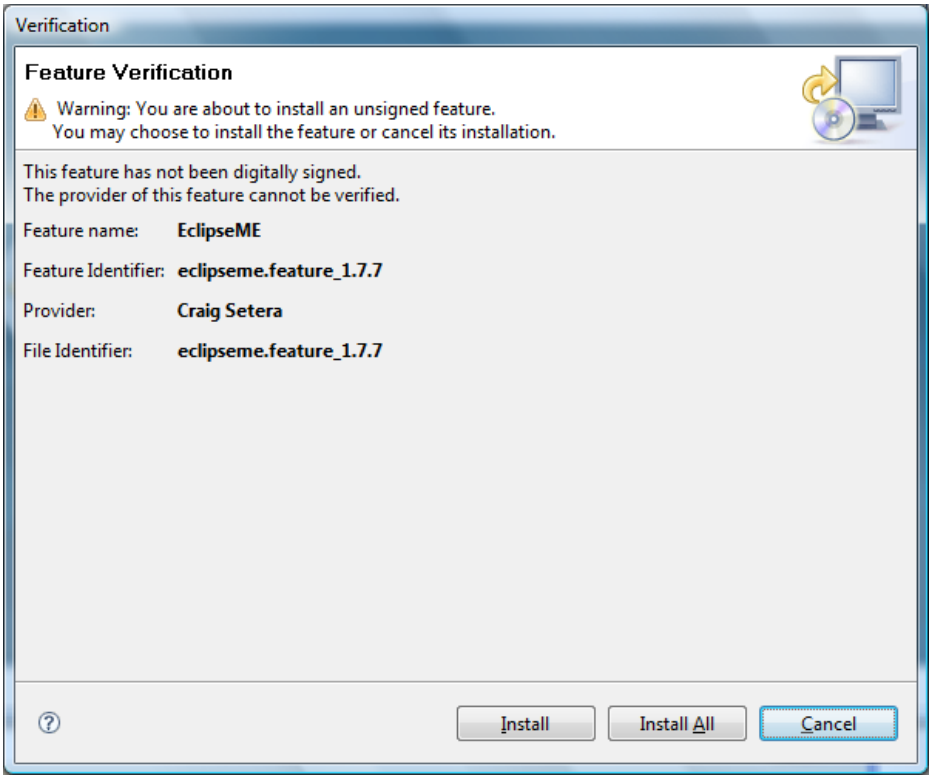


Figura II.12: Feature Verification

Optem pel botó *Install* i la següent finestra emergent ens preguntarà si volem reinicialitzar l'ordinador per tal que la instal·lació es dugui a terme de manera apropiada, així doncs fem *click* sobre el botó *Yes*.

Tornem a obrir el programa i veurem que tot ja és preparat per poder començar a treballar. És possible comprovar que efectivament aquest és el cas si anem a la pestanya *Window* i escollim *Preferences*, ja que hi trobarem les dades que mostren la **figura II.13**, **Preferences**:

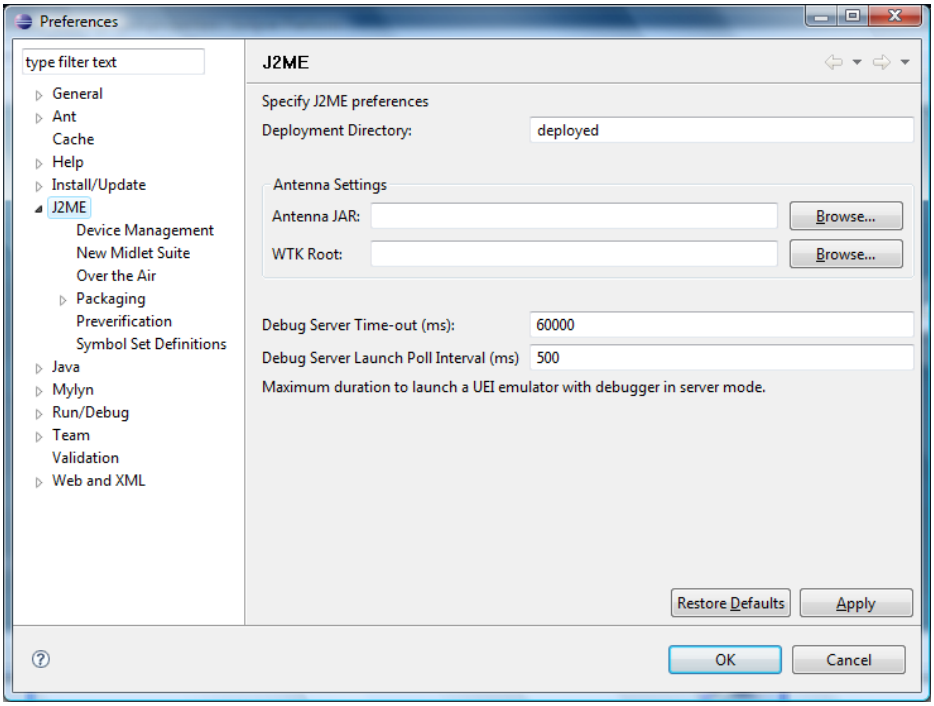


Figura II.13: Preferences

Com es pot observar a la part esquerra de la finestra, l'aplicació ja està preparada per J2ME, tal i com necessitem per desenvolupar el projecte que ens ocupa. Així doncs, hem assolit el nostre objectiu.

12. APÈNDIX III: INSTAL·LACIÓ DEL GEL

Si descarreguem l'arxiu d'instal·lació del Gel, veurem que té l'aspecte que es mostra a la **figura III.1**, **GExpertes Inc**:

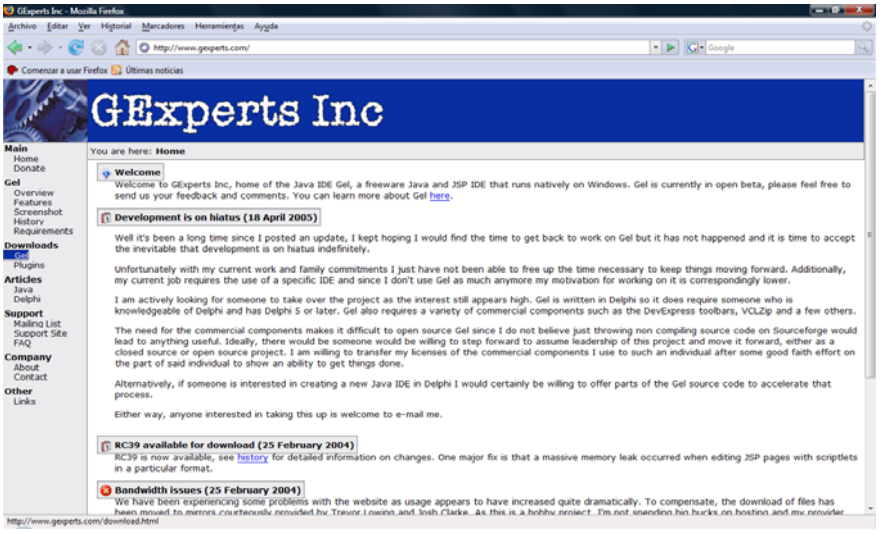


Figura III.1: GExperts Inc

Ens descarregarem la versió més recent des de la pàgina *download* de GEL. La **figura III.2**, **Latest Version**, ho reflexa:

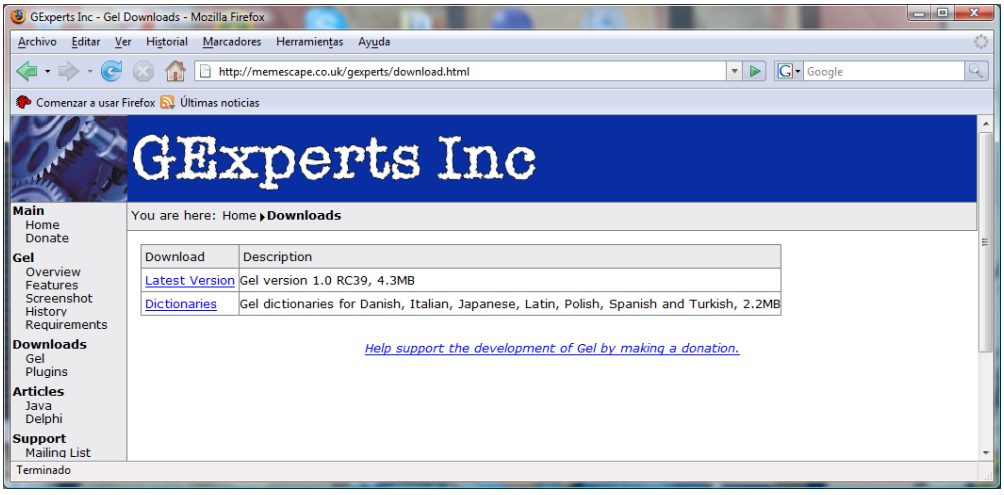


Figura III.2: Latest Version

L'arxiu en qüestió estarà comprimit en format *zip*, un cop descomprimit disposarem d'un arxiu executable *setup.exe*.

Després de polsar *Next* i acceptar les condicions de la llicència, se'ns preguntarà el directori d'instal·lació. Si no ho modifiquem, per defecte s'indica el directori *programs folder* o el que és el mateix, arxius de programes.

D'altra banda, també hi ha un diàleg que permet escollir la incorporació d'icones a l'escriptori, al menú d'inici o a la barra de tasques. En aquesta ocasió, procedirem més breument i simplement incorporarem una de les imatges que fan referència a la instal·lació, la que es correspon amb la **figura III.3**, **Setup-Gel**, ja que el procés no és complicat i és bastant senzill, sobretot si no hi ha hagut problemes en el seguiment de les instal·lacions exposades amb anterioritat i ja s'està familiaritzat amb el procés:

Figura III.3: Setup-Gel

Un cop finalitzat, ja es pot executar el programa directament. Té l'aspecte que queda reflectit a la **figura III.4**, **Gel**:

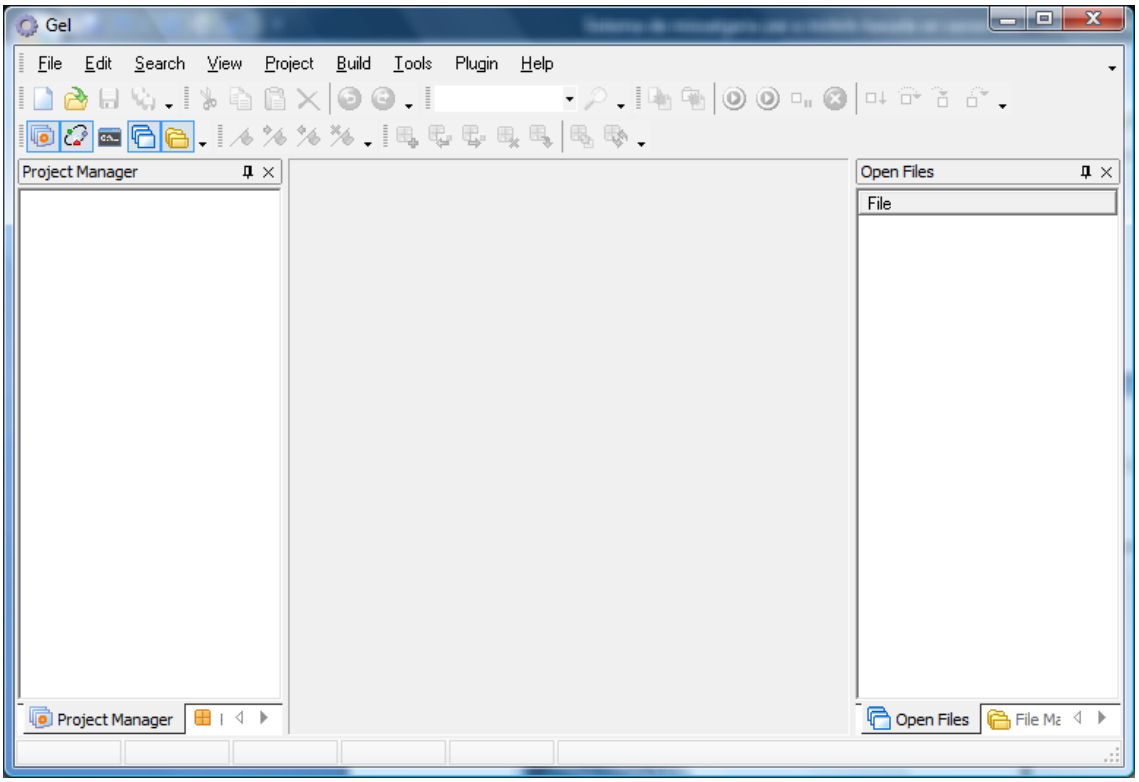


Figura III.4: Gel

13. APÈNDIX IV: FUNCIONAMENT DELS IDE's. PROGRAMACIÓ DE MIDlet's AMB KTOLLS

Per tal de conèixer quin és el funcionament dels programes, el que primer hem de fer és estudiar cóm funcionen els projectes J2ME.

Quan s'obre un nou projecte en J2ME, es creen diferents directoris i arxius, el conjunt rep el nom de *MIDlet suite*.

Els directoris que es creen són, més concretament, els següents: *bin*, *classes*, *docs*, *lib*, *res*, *src*, *tmpclasses* i *tmplib*. Convé descriure'ls breument:

- BIN: Aquí s'inclouen dos arxius quan s'empaqueta el projecte, un arxiu *.jad* que és el descriptor de la *MIDlet suite* i/o un manifest amb informació del projecte, i un altre arxiu que té una extensió *.jar* i és on està pròpiament l'aplicació comprimida i preparada per ser instal·lada al dispositiu real. D'altra banda, també pot incloure un

fitxer HTML que fa servir internament quan s'executa l'ordre RUN via OTA. Cal afegir que l'arxiu manifest no s'ha de confondre amb el *descriptor.jad*, ja que el manifest ens dóna informació com el nom, la versió, el venedor, etc., en canvi, el descriptor informa dels requeriments del sistema.

- CLASSES: Aquest directori és utilitzat pel *Toolkit* per guardar les fonts *class* compilades.
- LIB: És el lloc on queden incloses les llibreries de tercers.
- RES: Aquí queden guardades les imatges, els sons i altres recursos. Es situa a l'arrel del *MIDlet suite JAR*.
- SRC: A aquesta carpeta s'hi col·loquen els fitxers font.
- TMPCLASSES: Es tracta d'un directori que fa servir el *Toolkit*.
- TMPSRC: És un altre directori que també fa servir el *Toolkit*.

Anem ara a analitzar cóm es crea un *MIDlet suite*. Un dels programes més senzills que podem fer servir amb aquest objectiu és el *Ktool* de *Wireless Tools kit*. Des de la finestra principal del *ktool* farem *click* sobre el botó *New Project...* La **figura IV.1**, **Create a new project**, és força aclaridora:

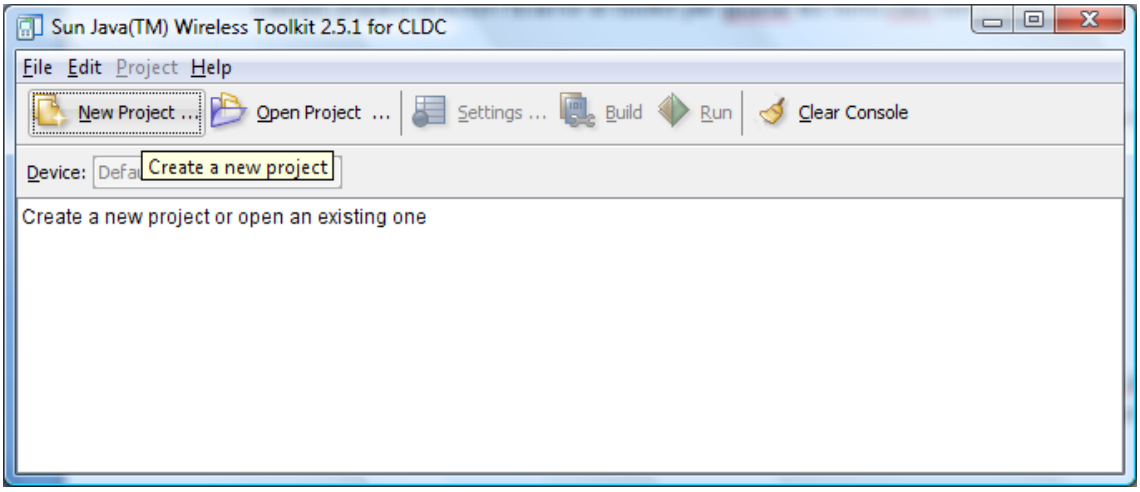


Figura IV.1: Create a new project

S'obre la següent finestra, tal i com es veu a la **figura IV.2**, **New Project**, on hem d'omplir el nom del projecte, per exemple HelloWorld, i el nom del MIDlet Class, aquí hi podem posar el mateix nom HelloWorld. Fem *click* al botó *Create Project* i ja tenim el projecte creat.

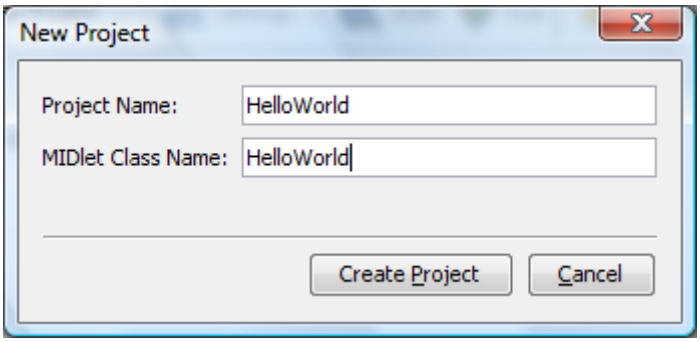


Figura IV.2: New Project

S'obrirà la finestra que es veu a la **figura IV.3**, **Settings for project Hello World**. Polsem OK i ja tenim el projecte creat en el directori predeterminat.

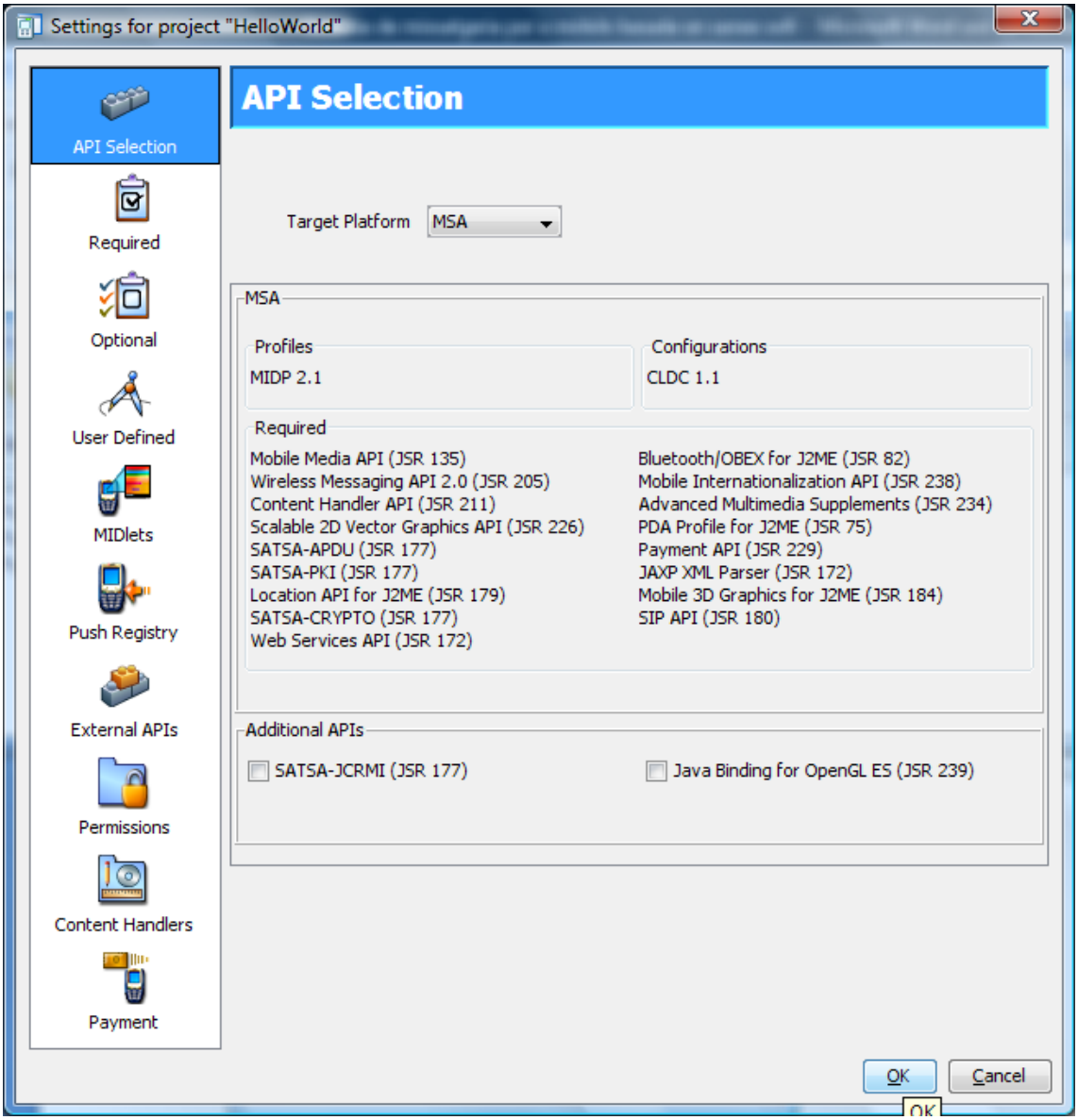


Figura IV.3: Settings for project Hello World

En aquest exemple el tenim en la ruta C:\WTK2.5.1\apps, que és la especificada quan es va configurar el directori de treball en el moment d'obrir el programa. Així és com es mostra a la següent imatge, la **figura IV.4**, **Project Settings**:

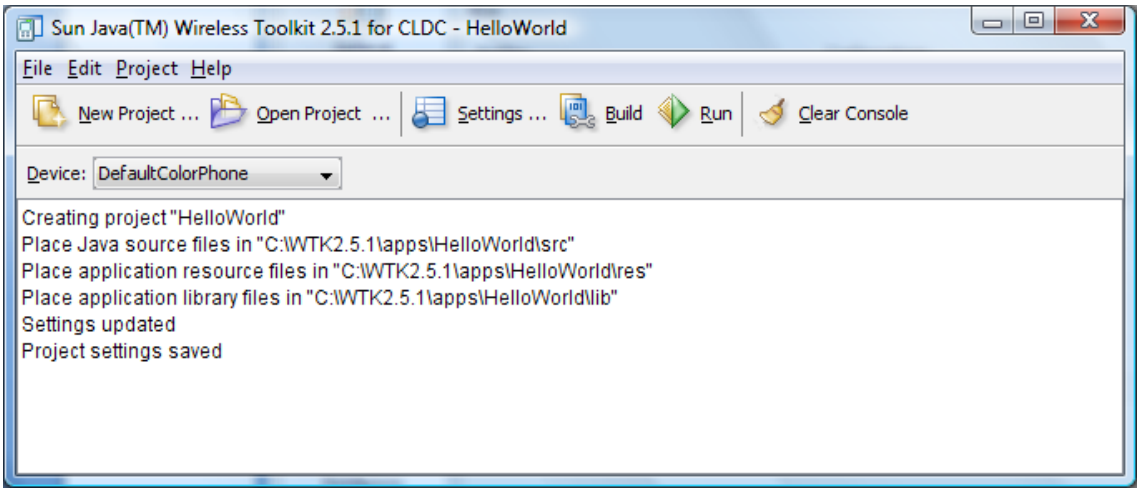


Figura IV.4: Project Settings

Si observem la finestra principal, veurem que el projecte s'ha creat. També ho podem comprovar en explorant el directori c:\WTK2.5\apps. Així queda reflectit a la **figura IV.5**, **Exploració**:

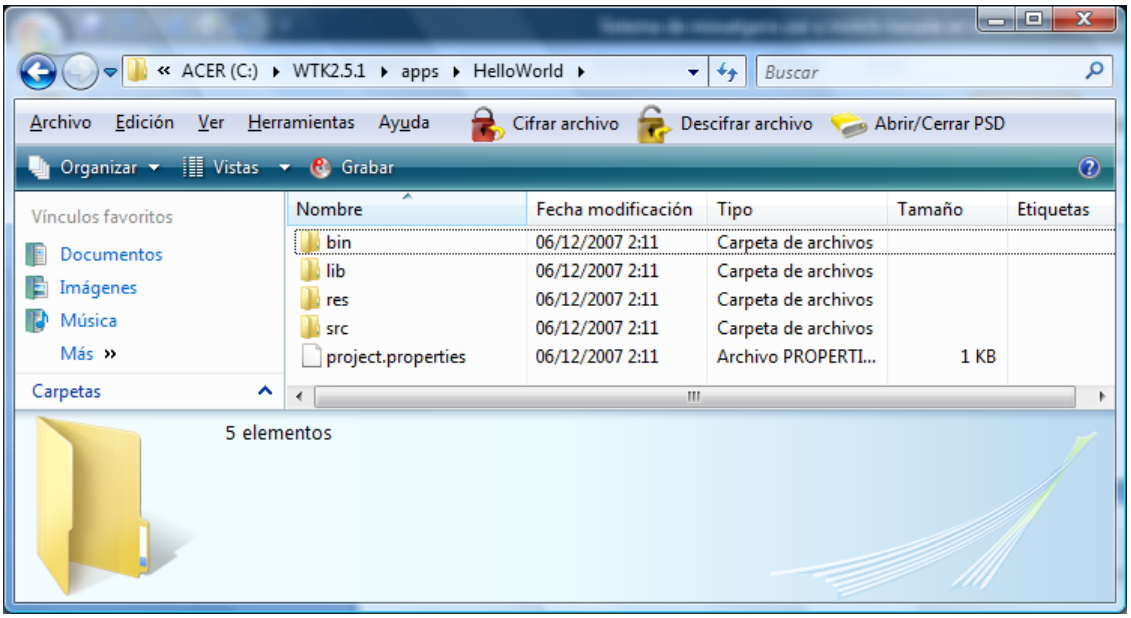


Figura IV.5: Exploració

Si ens remetem a la breu descripció que hem fet sobre les MIDlets suite, observarà el lector que es troba a faltar els directoris *tmpclasses* i *tmplib*. Això es degut a que encara no s'ha compilat cap codi. Tenim un projecte però no hem determinat encara cap programa per realitzar la compilació.

Podem provar el següent programa:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class HelloWorld extends MIDlet implements CommandListener {
 private Command exitCommand;
 private Display display;
 private Form screen;

 public HelloWorld() {
  // Obtenemos el objeto Display del midlet.
  display = Display.getDisplay(this);

  //  Creamos el comando Salir.
  exitCommand = new Command("Salir", Command.EXIT,2);

  // Creamos la pantalla principal (un formulario)
  screen = new Form("HelloWorld");

  // Creamos y añadimos la cadena de texto a la pantalla
  StringItem saludo = new StringItem("","Hola Mundo...");
  screen.append(saludo);

  // Añadimos el comando Salir e indicamos que clase lo manejará
  screen.addCommand(exitCommand);
  screen.setCommandListener(this);
 }
 public void startApp() throws MIDletStateChangeException {
  // Seleccionamos la pantalla a mostrar
  display.setCurrent(screen);
 }

 public void pauseApp() {
```

```
 }

 public void destroyApp(boolean incondicional) {
 }

 public void commandAction(Command c, Displayable s) {
  // Salir
  if (c == exitCommand) {
   destroyApp(false);
   notifyDestroyed();
  }
 }
}
```

Hem d'escollir un editor per tal d'escriure el programa i guardar-lo amb l'extensió *.java, és a dir, HelloWorld.java. Cal puntualitzar que ha de tenir exactament el mateix nom que la suite per a què ho reconegui, respectant majúscules i minúscules.

Amb anterioritat hem fet referència a l'editor *Forte* de Sun, de fet, també podríem fer servir l'Eclipse, el Gel o l'aplicació NetBeans, és més, podríem utilitzar el bloc de notes de Windows. La diferència està en què els editors de java canvien les lletres de color en base a la seva funció, i dins el codi font hi trobem certes ajudes que poden ser d'utilitat. Com que ja disposem del codi escrit, simplement el copiarem i el pegarem al bloc de notes i guardarem com a HelloWorld.java en el directori *src*. Així queda reflectit a la **figura IV.6**, **Codi**:

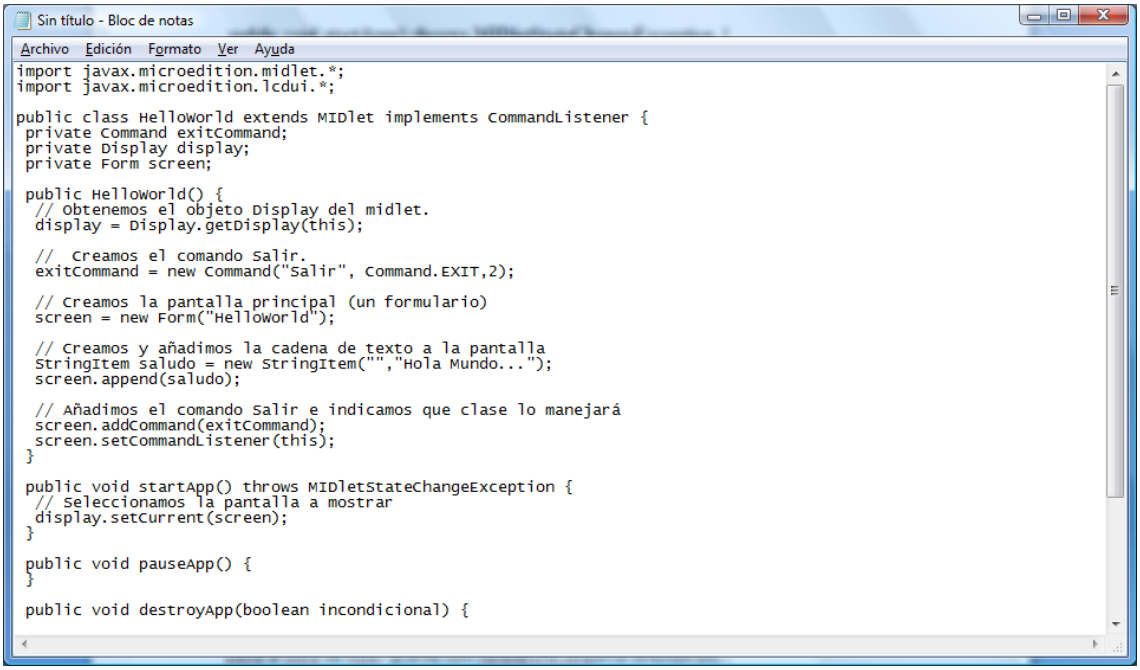


Figura IV.6: Codi

Guardarem aquest arxiu com a HelloWorld.txt, com es mostra a la **figura IV.7**, **Guardar com**:

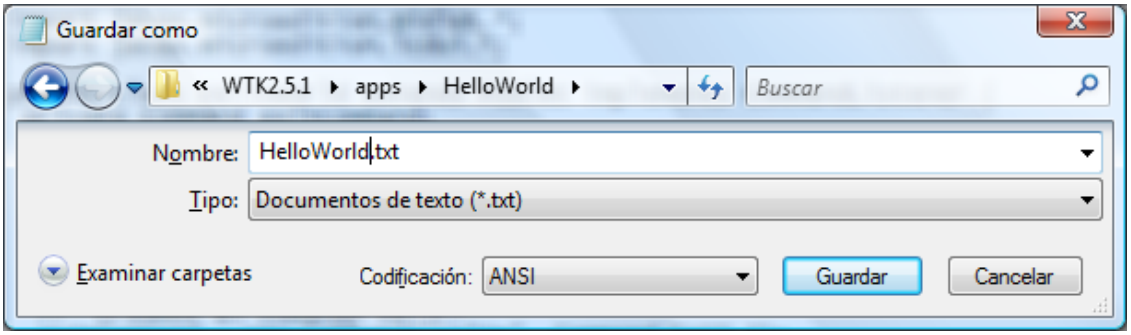


Figura IV.7: Guardar com

A continuació hem de fer *click* a la tecla *build*, i una vegada fet això l'executarem amb el botó *run*. Si tot funciona bé tindrem el resultat de l'emulador que es veu a la **figura IV.8**, **Resultat de l'emulació**:

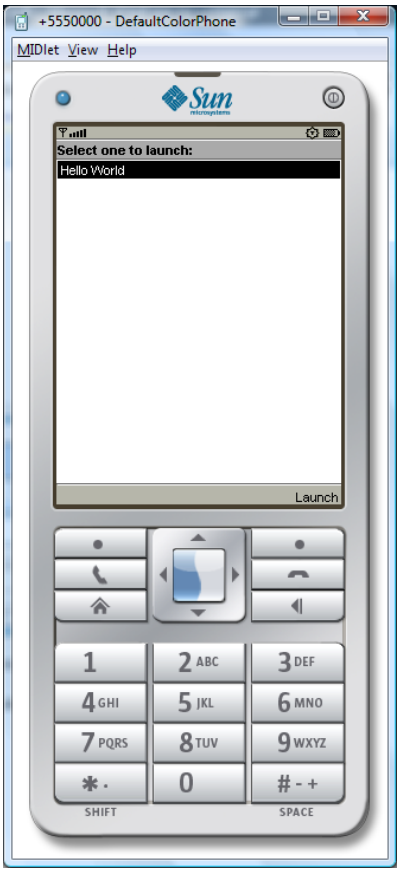


Figura IV.8: Resultat de l'emulació

De totes les aplicacions descarregades, utilitzarem per emular els programes l'aplicació *Ktoolsbar* de *Wireless tookit*, i el *NetBeans* per editar i compilar els projectes. Gel presenta alguns problemes de compatibilitat amb Windows Vista, d'aquí que, tot i poder ser utilitzat (de fet s'ha usat pel desenvolupament del present projecte) el deixem en un segon terme.

Així doncs, convé que estudiem amb una mica de deteniment cóm utilitzar l'aplicació *NetBeans*, quins són els passos a seguir per començar des del seu inici una aplicació nova, i quines configuracions s'han de fer per tal que funcioni tot de manera eficient.

Per començar a fer servir el *NetBeans* és necessari realitzar en l'aplicació un seguit d'actualitzacions associades a una sèrie de *plugins* que fan referència a la part de *mobility* i que estan vinculades als mòbils.

Així doncs, per instal·lar els *plugins* necessaris en qüestió s'ha de seleccionar l'opció *Tool* de la barra d'eines, i escollir l'opció *plugins* dins del ventall de possibilitats que ofereix el submenú. En fer-ho, ens apareixerà la finestra de la **figura IV.9**, **Plugins**:

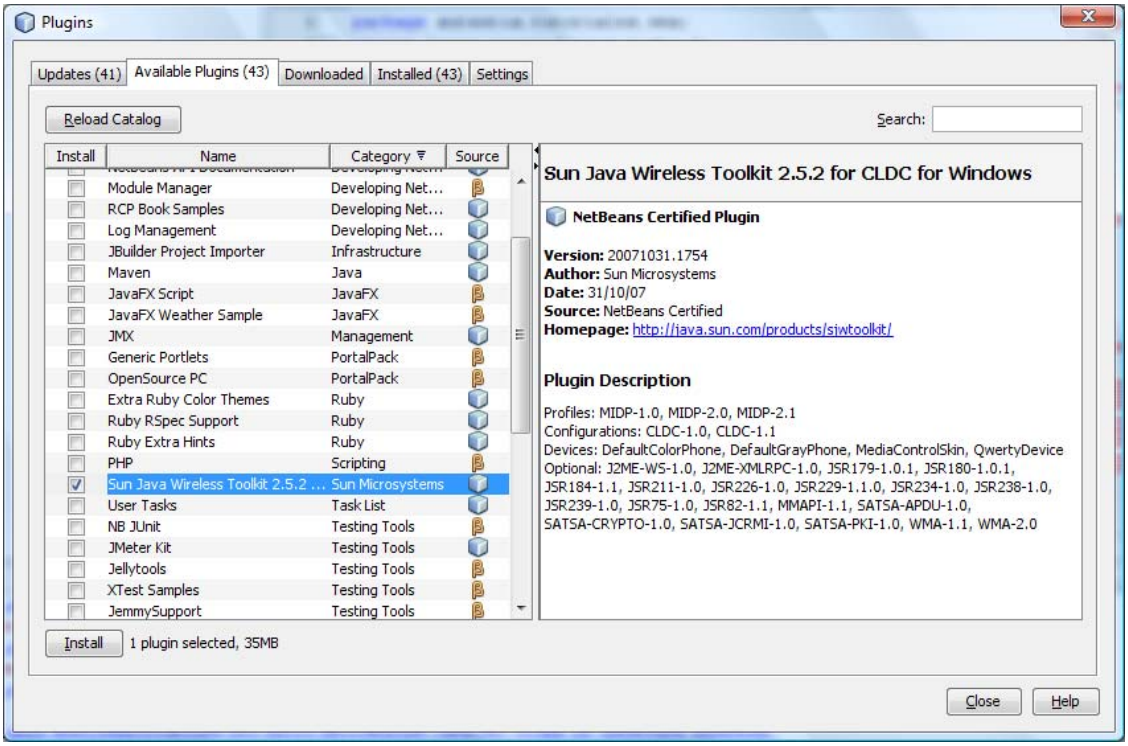


Figura IV.9: Plugins

I aquí podrem seleccionar d'entre totes les actualitzacions disponibles aquella que ens interessa. Així doncs, farem *click* a la pestanya *Available plugins* i aquí marcarem amb una V el quadret associat a *Sun Java Wireless Toolkit 2.5.2*, tal i com es veu a l'anterior captura de pantalla, **figura IV.9**, i a continuació marcarem el botó *Install*.

Ens apareixerà la finestra emergent de la **figura IV.10**, **Plugin Installer**:

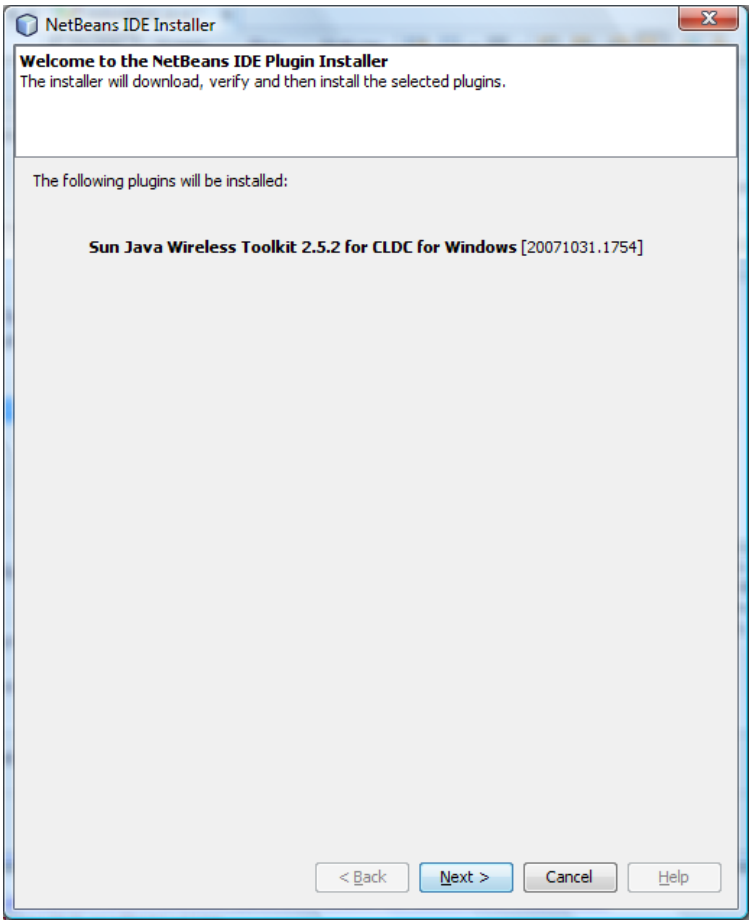


Figura IV.10: Plugin Installer

Arribats a aquest punt, hem de fer *click* sobre el botó *Next*. A continuació ens apareixerà una pantalla on hem d'indicar que acceptem les condicions d'us de la llicència, és a dir, hem de marcar *I accept the terms in all of the license agreements* i després simplement polsarem el botó *Install*.

Val a dir que és possible que ens trobem amb problemes amb la *proxy* quan intentem duu a terme l'actualització. Per solucionar-los, haurem de posar de manera manual l'URL de l'actualització, i que no és altra que http://jcp.org/en/jsr/all. El camp *port* s'ha de deixar en blanc.

Per fi, anem doncs a introduir-nos definitivament a la casuística que estem analitzant provant de fer un primer projecte.

El primer pas és seleccionar *New Project* a la pestanya *File* a la finestra que es veu a la **figura IV.11**, **Choose project**:

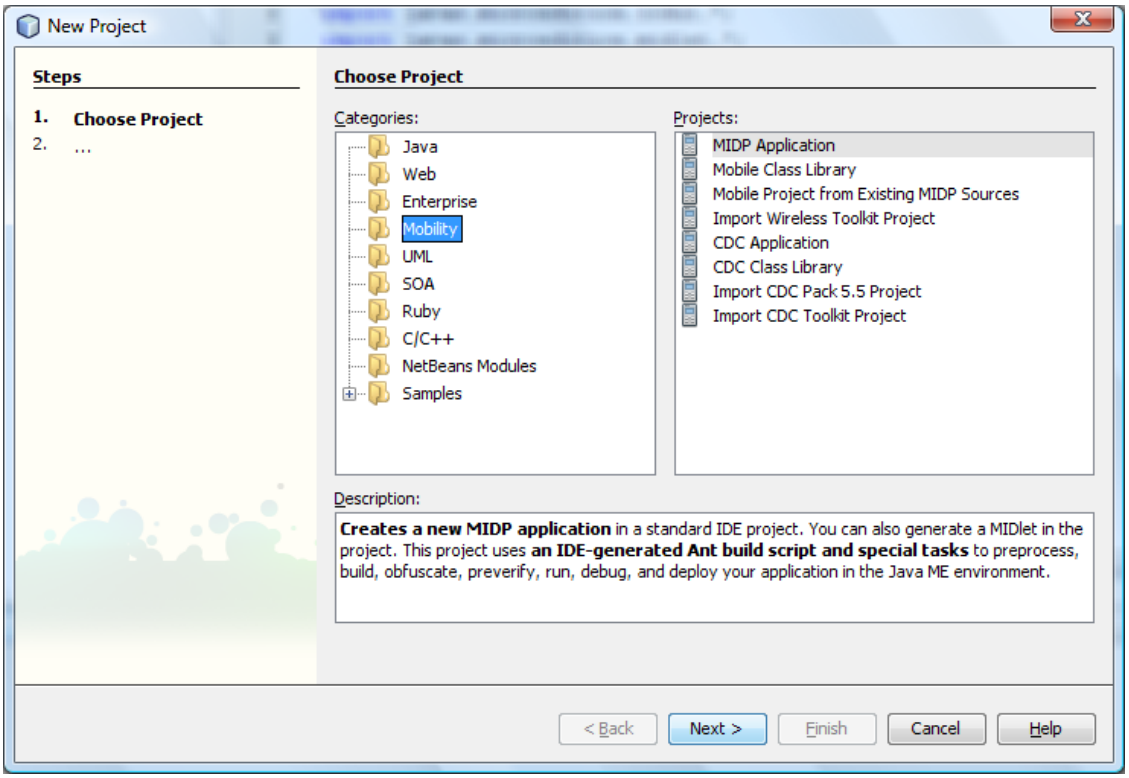


Figura IV.11: Choose project

Seleccionem la categoria *Mobility* i un projecte *MIDP Application*, tal i com s'aprecia a la finestra capturada i que precedeix aquestes línies, la **figura IV.11**. A continuació fem *click* sobre el botó *Next*. Ens apareixerà una nova finestra, més concretament, la que es mostra a la **figura IV.12**, **Name and Location**:

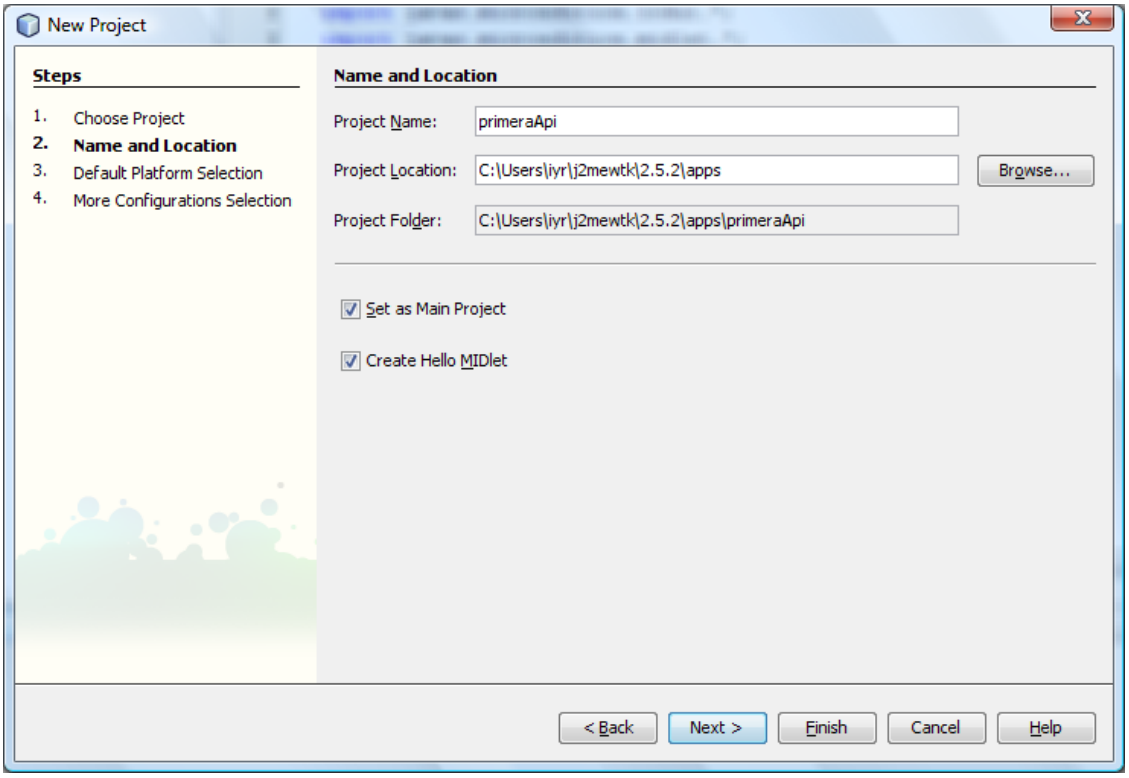


Figura IV.12: Name and Location

Ara hem d'omplir els camps d'aquesta nova finestra que ens ha aparegut. A *Project Name*, òbviament, hi escriurem el nom del projecte, en aquest cas el batejarem com "primeraApi". Marquem *Set as Main Project* i *Create Hello MIDlet* i fem click al botó *Next*.

La següent finestra que ens apareix, **figura IV.13**, **Default plataform selection**, ens demana informació sobre la plataforma que utilitzarem:

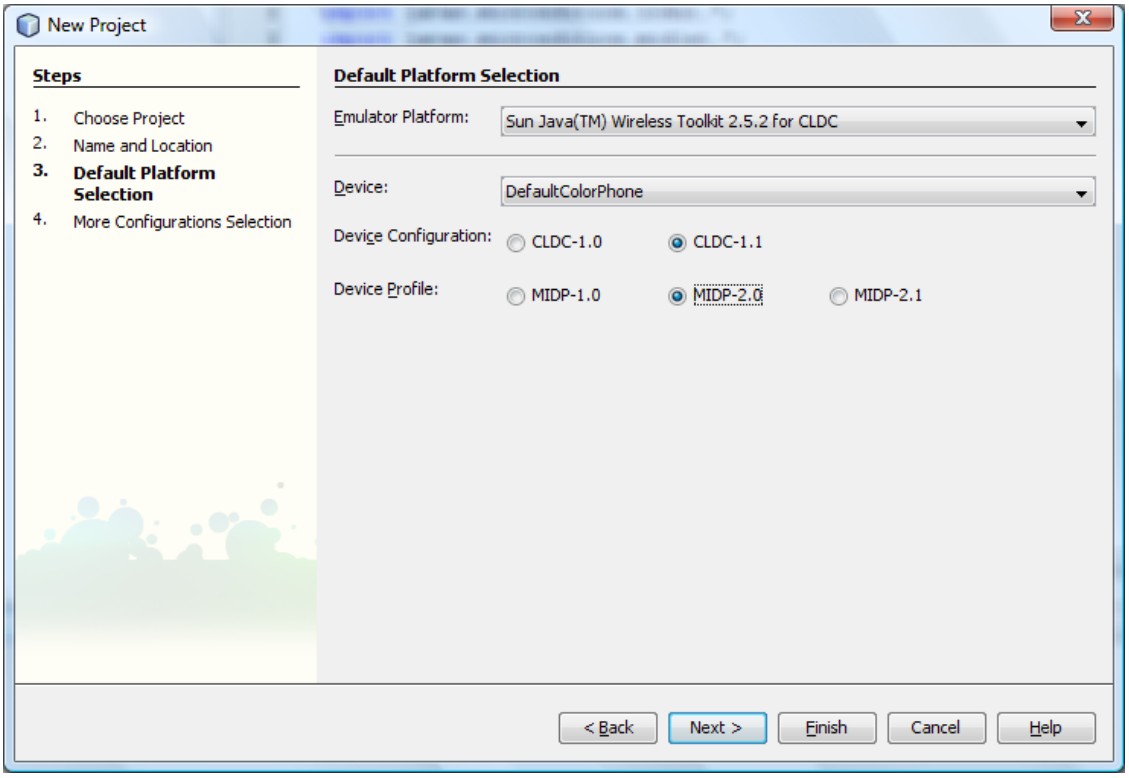


Figura IV.13: Default platform selection

En aquesta finestra hem de seleccionar al camp *Emulator Platform* l'opció *Sun Java (TM) Wireless Toolkit 2.5.2 for CLDC*. S'ha de marcar la resta de camps segons les especificacions del mòbil al que hi volem instal·lar les aplicacions. Fem *click* al botó *Next*.

La finestra que ens apareix és la que es mostra a la **figura IV.14**, **More configurations selection**:

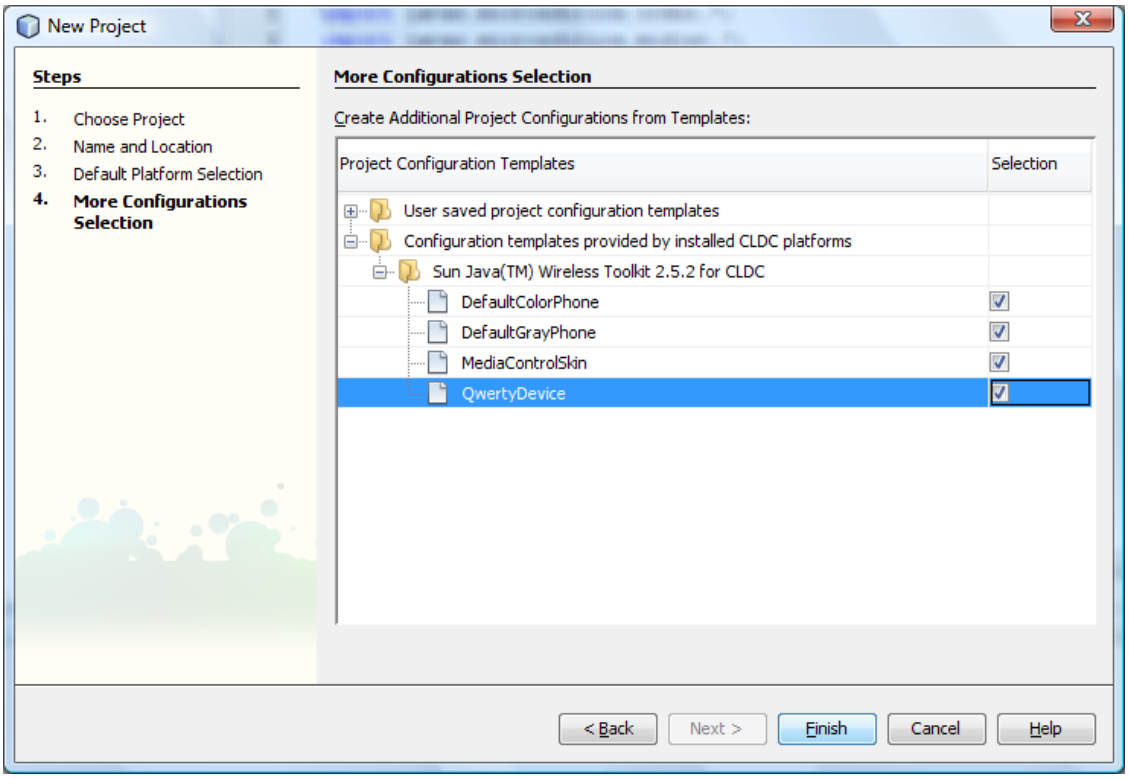


Figura IV.14: More configurations selection

Arribats a aquest punt, despleguem la carpeta *Configuration templates provided by installed CLDC platforms*, i procedim d'igual manera amb la carpeta *Sun Java (TM) Wireless Toolkit 2.5.2 for CLDC*. A continuació marquem les quatre opcions tal i com queda patent a la **figura** precedent, la **IV.14**, , i ja estem en condicions de fer *click* al botó *Finish*.

Passem a estar ubicats al punt mostrat per la **figura IV.15**, **Hello World**:

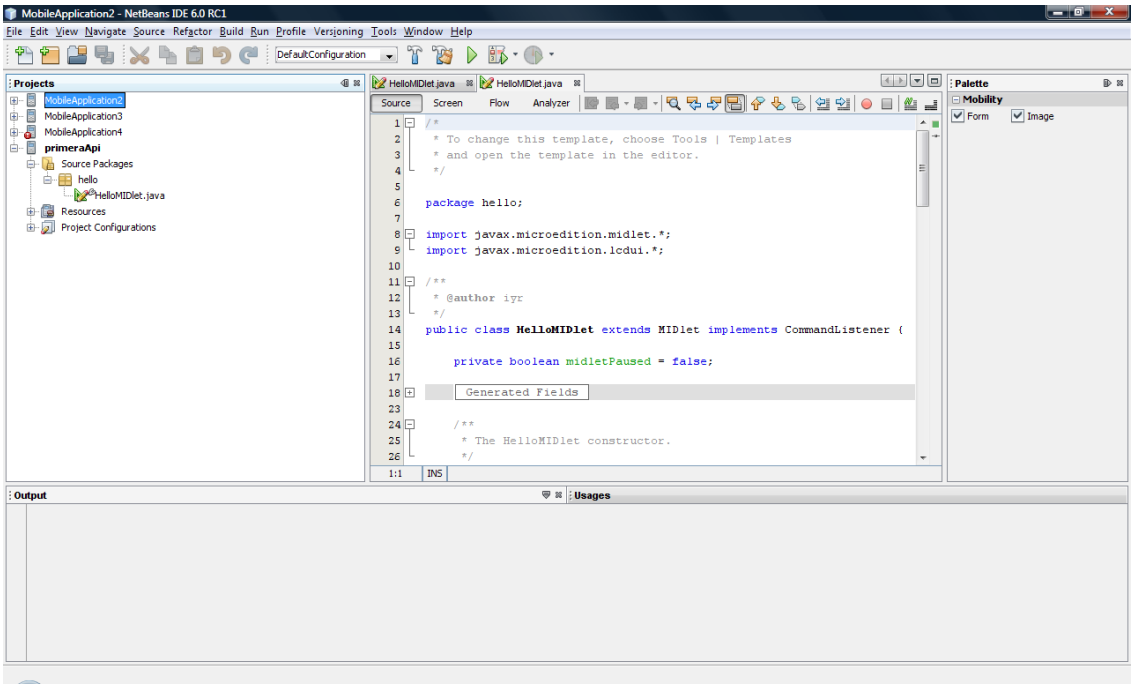


Figura IV.15: Hello World

No se li escaparà al lector que ara ja tenim el programa obert. Com en un dels passos inicials hem marcat *create HelloMidlet*, ja tenim el programa fet. Si seleccionem la pestanya *Screen* a la pantalla del codi font veurem un *preview* del que farà l'aplicació, tal i com es veu a la **figura IV.16**, **Preview**.

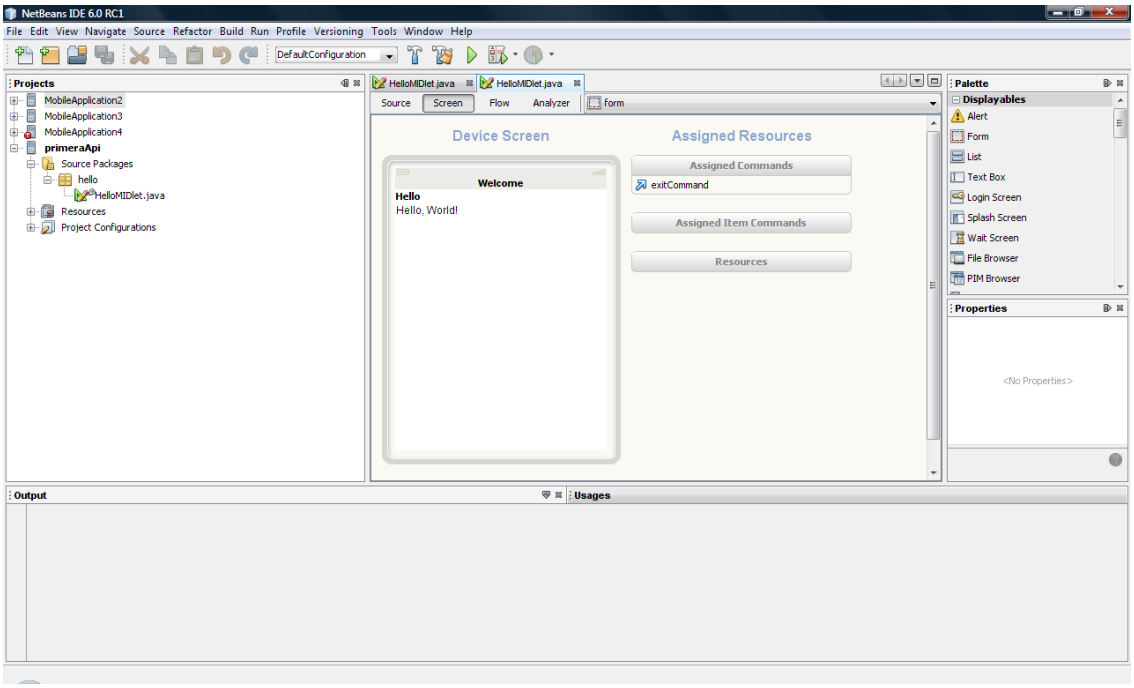


Figura IV.16: Preview

Per començar a familiaritzar-nos amb aquest entorn, podem modificar el text seleccionant directament la frase *Hello,World* i a la dreta, concretament a *properties* seleccionar *Text*. Podem modificar les paraules *Hello,World* i canviar-les per una altra frase, com per exemple "Bon dia a tothom". Tal i com es veu a la **figura IV.17**, **Bon dia a tothom**, es pot veure que la pantalla ha canviat:

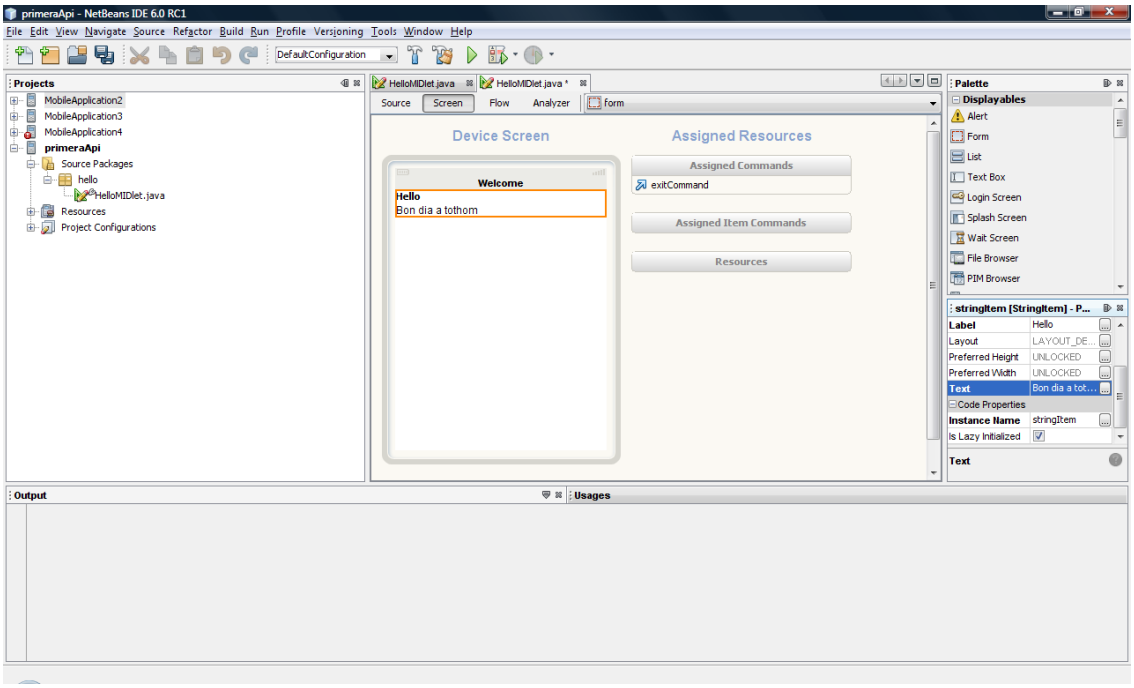


Figura IV.17: Bon dia a tothom

Ara, s'ha d'escollir l'opció *Run* que està disponible a la barra d'eines, de manera que seleccionarem *Run main project*. Una alternativa amb la que aconseguim el mateix resultat és polsant la tecla F6, ja que amb això es fa una emulació de l'aplicació en pantalla.

El resultat serà es el que queda reflectit a la **figura IV.18**, **Emulació**:

Figura IV.18: Emulació

Podem fer servir la pantalla com si d'un mòbil es tractés, de manera que si polsem la tecla que hi ha sota de la paraula *Launch*, s'activarà la aplicació dins de l'emulador. El resultat serà el que veiem en la següent **figura IV.19**, **Emulació de l'aplicació**:

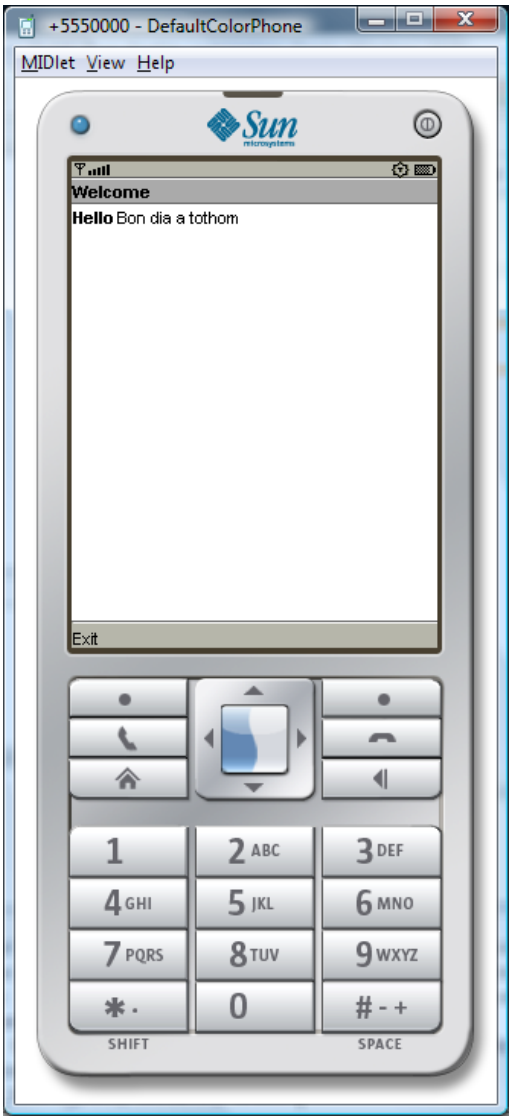


Figura IV.19: Emulació de l'aplicació

També podem fer servir el *Ktoolbar* de *Wireless Toolkit* per aconseguir el mateix resultat. Amb aquest propòsit, simplement hem de fer *click* sobre el botó *Open Project*, tal i com es mostra a la **figura IV.20**, **Open project**:

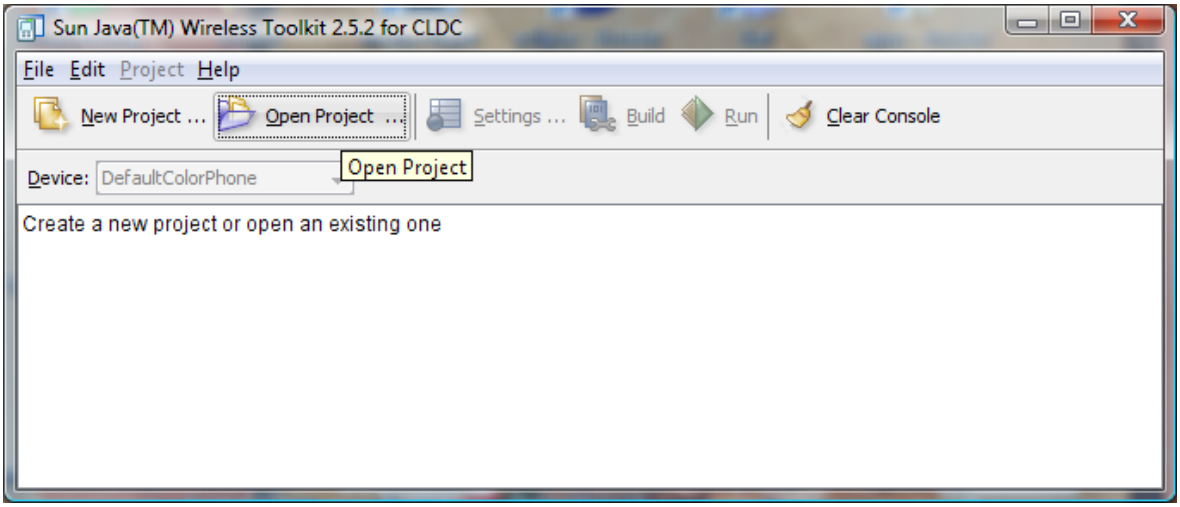


Figura IV.20: Open project

Se'ns obrirà la pantalla que es pot apreciar a la **figura IV.21**, **Projectes**, on trobarem un llistat de totes aplicacions disponibles. És possible que no localitzem l'aplicació que concretament busquem, ja que *Ktoolbar* té configurat un directori de treball, de manera que si el projecte en qüestió no es troba ubicat dins d'aquest directori, no hi serà present a la finestra que s'acaba d'obrir. Si aquest fos el cas, hauríem de fer un *copy/paste* de la ruta fins al directori de l'aplicació des de la ubicació actual fins el directori de treball del *Wireless Toolkit*.

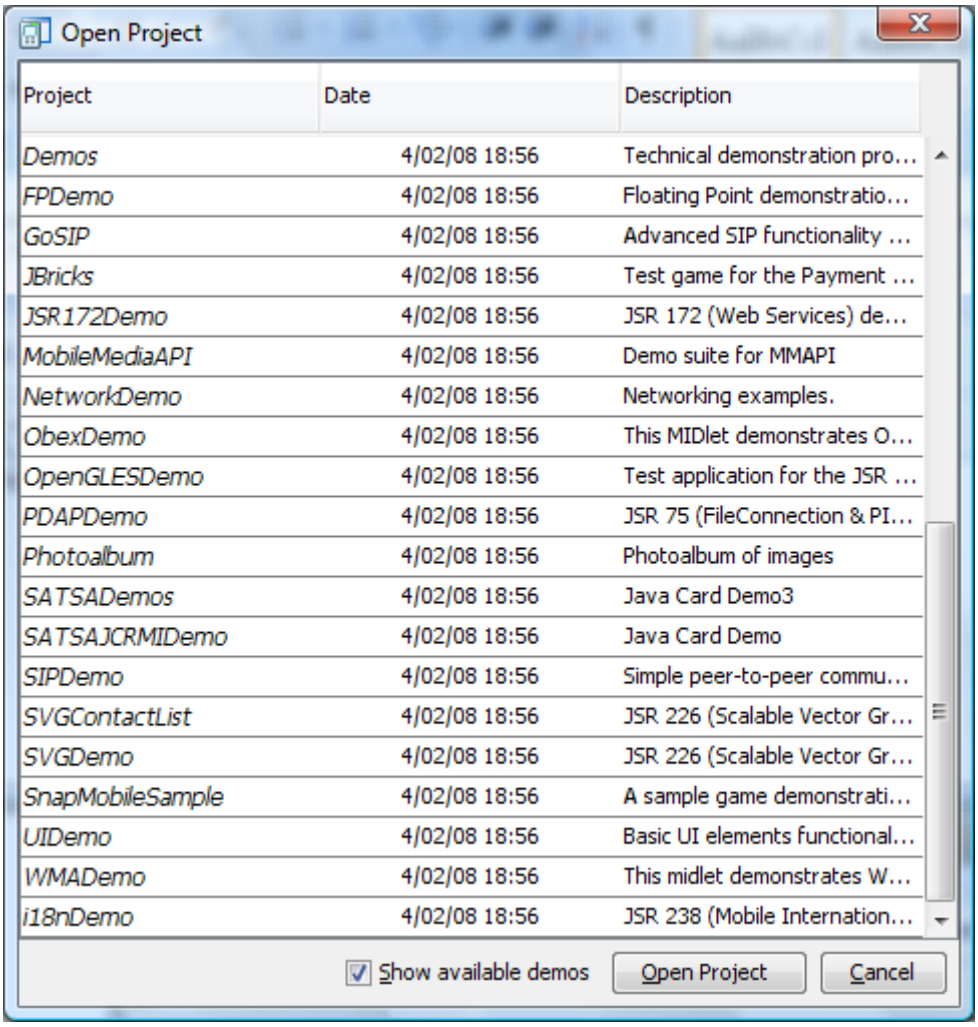


Figura IV.21: Projectes

En el nostre cas, el projecte sí està ubicat en el mateix directori, però l'aplicació no troba l'arxiu primeraApi.jad i per aquesta raó apareix el missatge que es pot visualitzar a la **figura IV.22**, **Missatge**:

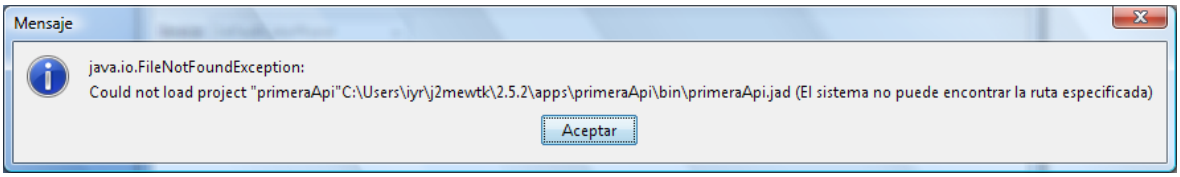


Figura IV.22: Missatge

Si mirem el contingut del directori primeraApi, es pot comprovar que els arxius *.jar* i *.jad* estan en un directori anomenat *dist*. Així queda exposat a la **figura IV.23**, **Directori dist**:

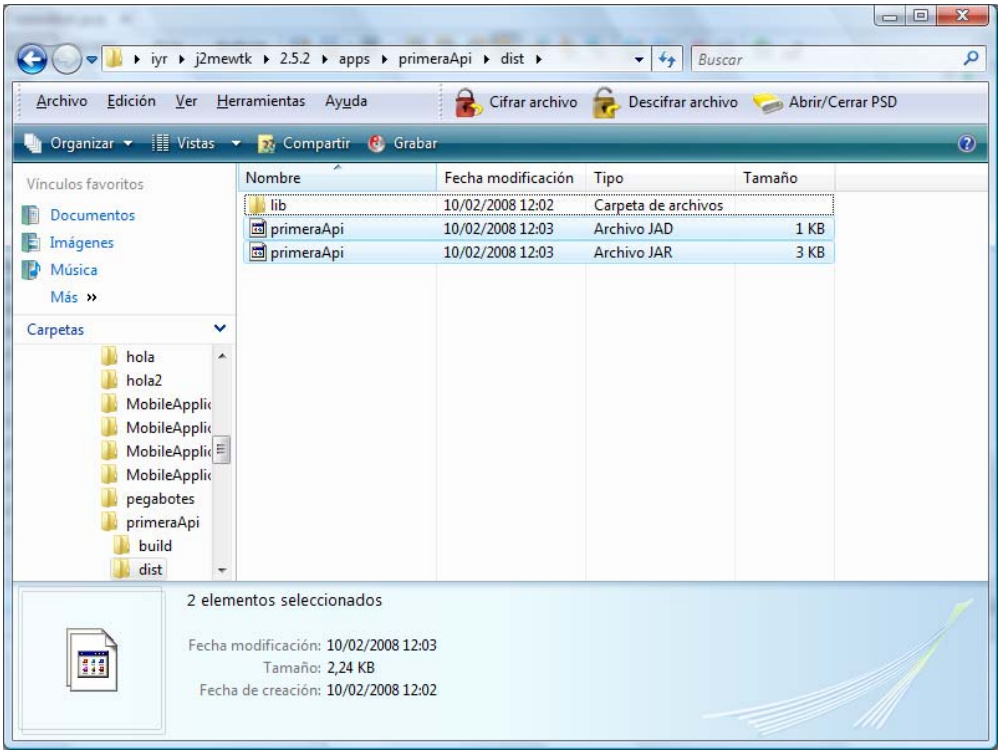


Figura IV.23: Directori dist

Per tant, el que farem és crear un directori anomenat *bin*, de forma que en el seu interior hi inclourem els arxius *.jad* i *.jar*, així no hi haurà problemes a la recerca que realitza *Ktoolbar*. La **figura IV.24**, **Creació del directori bin**, ens ho mostra:

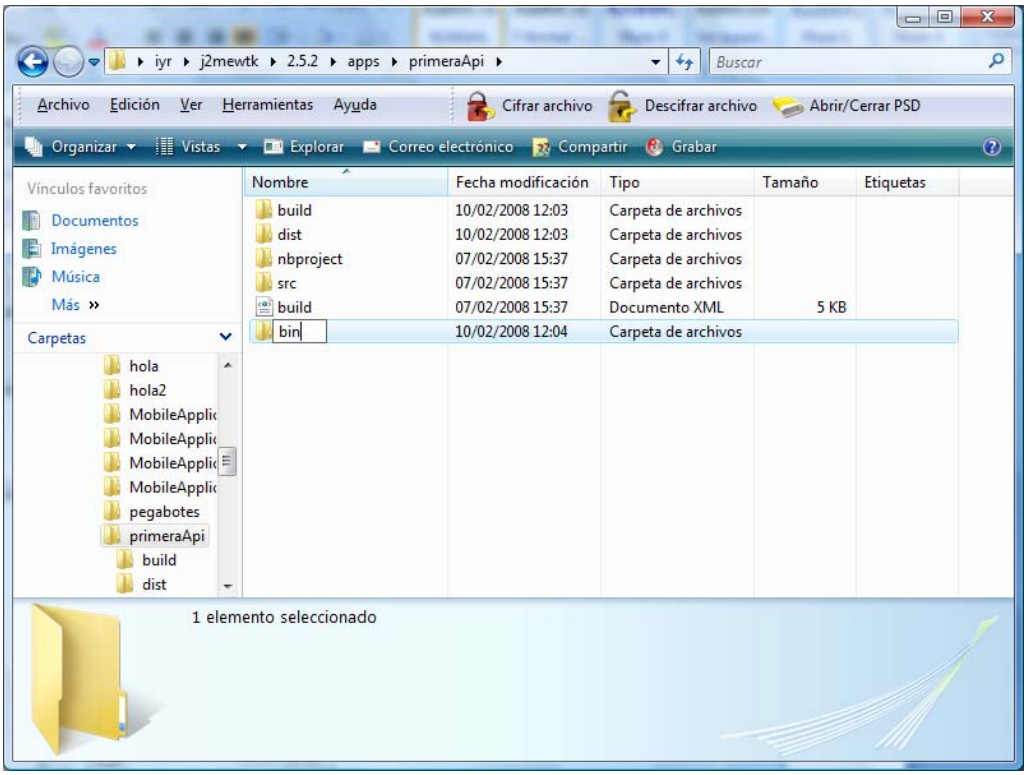


Figura IV.24: Creació del directori bin

A la següent **figura**, la **IV.25**, **Contingut del directori bin**, tal i com el seu nom indica, hi veiem el contingut del directori *bin*, que no és altra cosa que els fitxers en els que estem interessats:

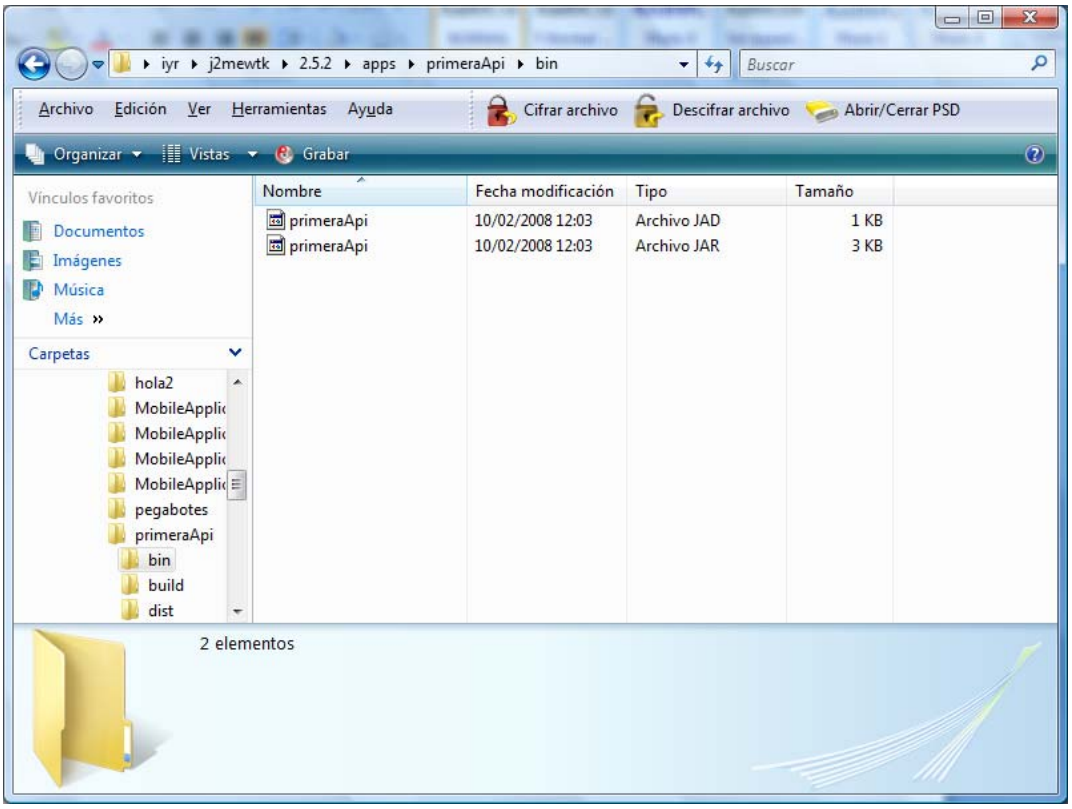


Figura IV.25: Contingut del directori bin

Ara que ja tenim els dos arxius dins del directori, tornarem a provar d'obrir el projecte amb el *Ktoolbar*, i comprovem que el resultat és exactament el mateix que amb el *NetBeans*, tal i com mostra la **figura** següent, la **IV.26**, **Resultat**:

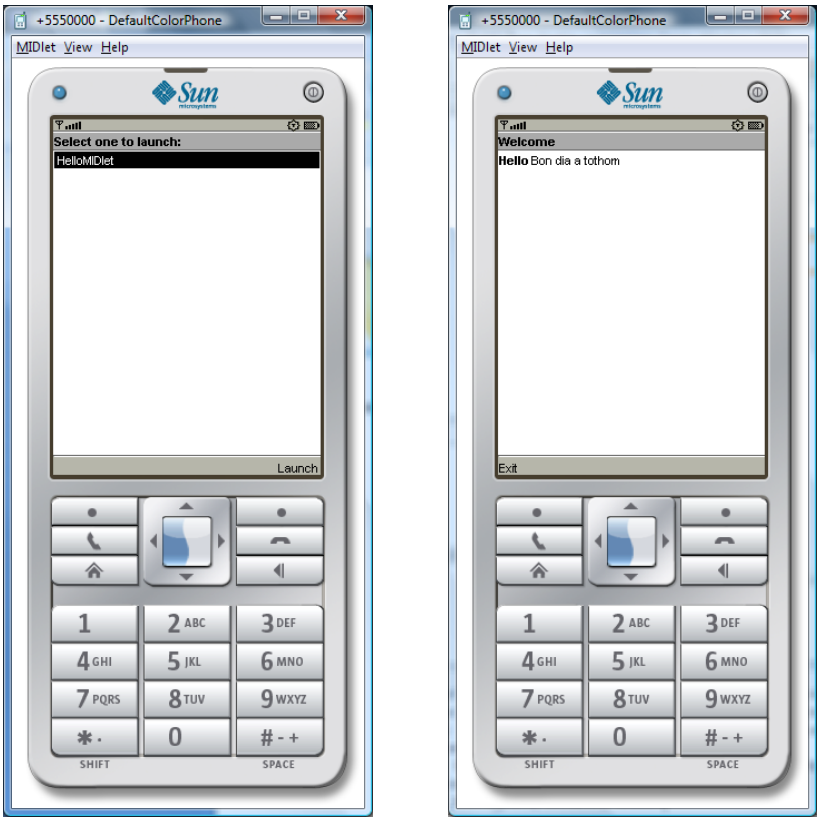


Figura IV.26: Resultat

A continuació anem a provar el funcionament del programa en un mòbil real. Cada dispositiu de telefonia mòbil té el seu propi mètode d'instal·lació d'aplicacions, en el nostre cas es farà servir el *Nokia PC Suite* per a un *Nokia N95*, ja que aquest és el mòbil de què disposem per fer la prova que ens ocupa.

S'ha de seleccionar l'arxiu PrimeraApi.jar i amb el botó dret del ratolí escollim l'opció <obrir amb>, tal i com es mostra a la **figura IV.27**, **PrimeraApi**.

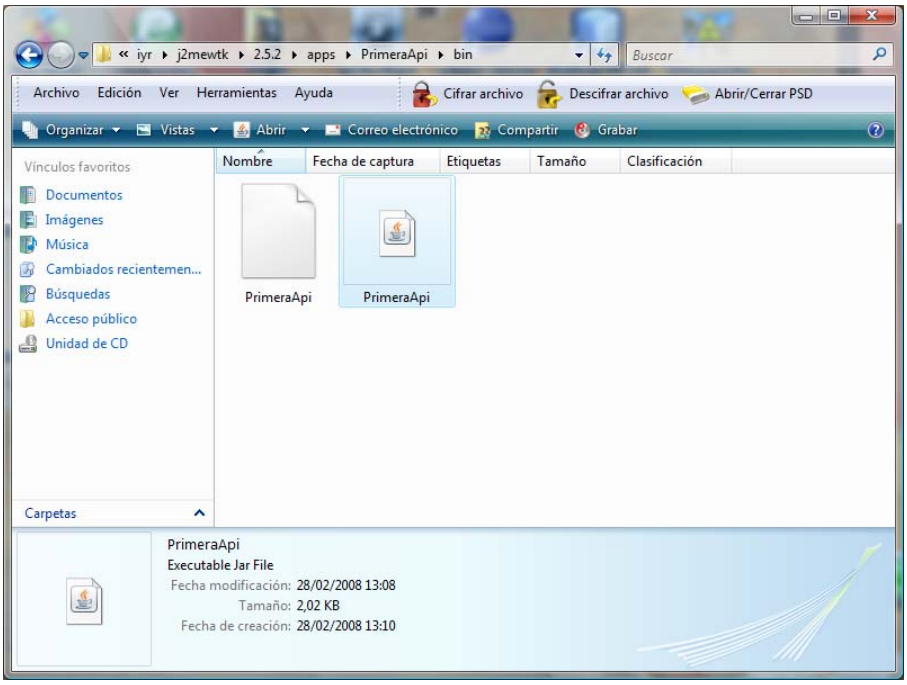


Figura IV.27: PrimeraApi

Tal i com hem indicat anteriorment, el programa que farem servir a tal fi serà el *Nokia Aplication Installer*, tal i com es veu a la **figura IV.28**, **Nokia Application Installer**:

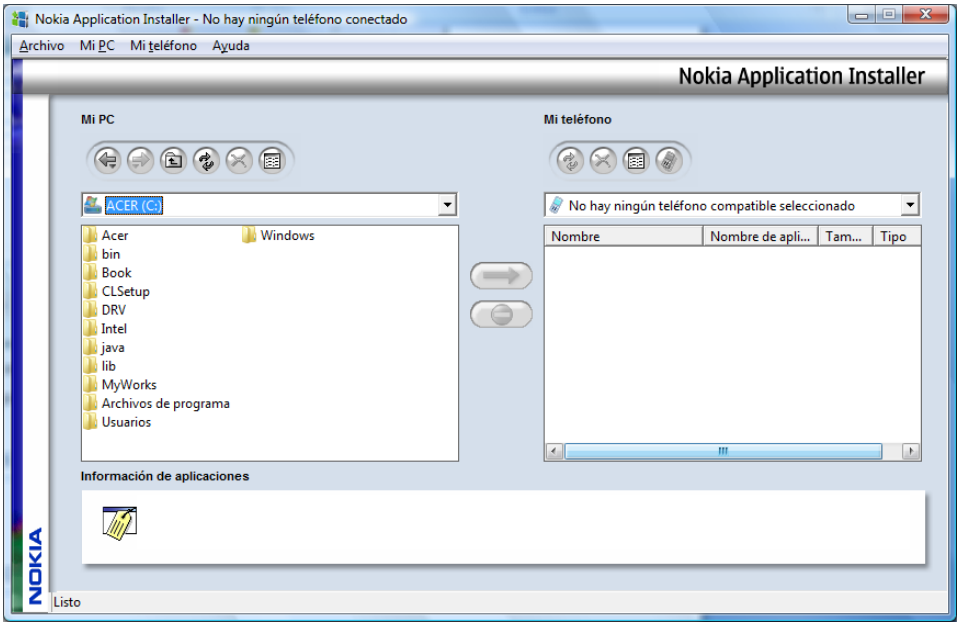


Figura IV.28: Nokia Application Installer

Si ens fixem en la **figura IV.28**, veurem que a la pantalla principal tenim una finestra a l'esquerra on hem de seleccionar l'aplicació *.jar* que acabem de fer. A continuació polsem

el botó que ve a representar una fletxa amb sentit cap a la dreta, que en fer la selecció es posa de color verd, indicant amb això que ja és possible fer *click* sobre ell. En el nostre cas, l'aplicació no trobava l'arxiu en el directori en qüestió, així que s'ha incorporat una copia al directori d'usuaris. Des d'aquí ja es pot obrir, tal i com queda exposat a la **figura IV.29**, **Memòria del telèfon**:

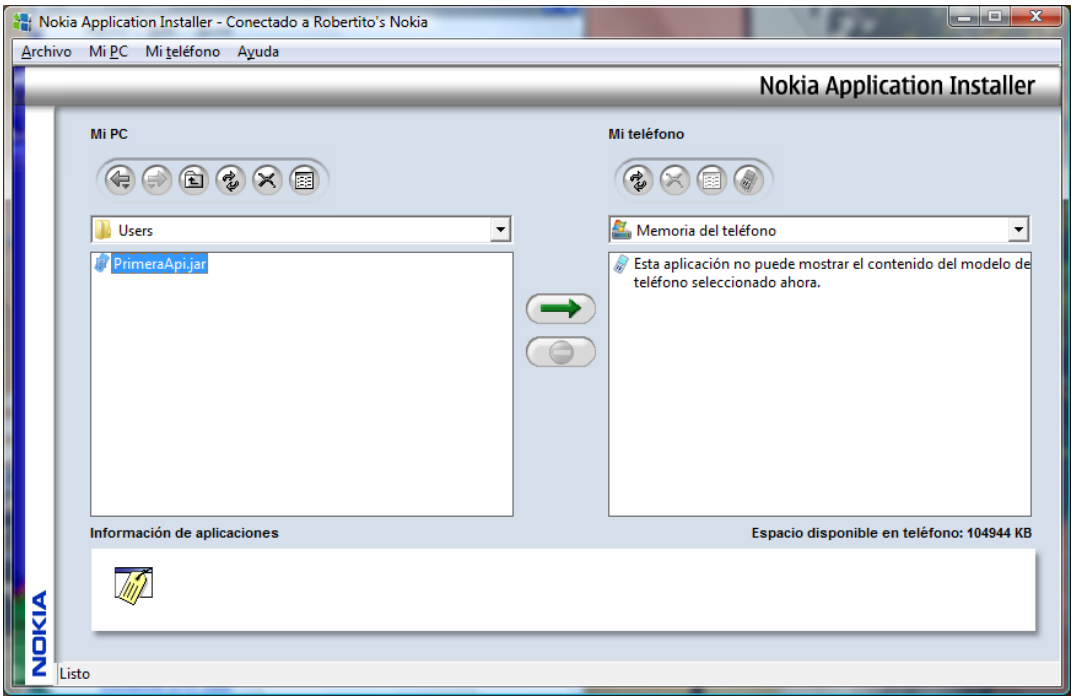


Figura IV.29: Memòria del telèfon

En cas de què la fletxa no es posés de color verd, podria ser un problema derivat de què el mòbil no estigués correctament connectat. En aquesta situació, s'hauria d'obrir el programa principal *Nokia Pc Suite*, fer servir el botó *conexion*, i seguir el menú per tal de connectar de forma correcta el mòbil amb el PC.

Una vegada connectat polsem la tecla que incorpora la fletxa a la dreta i apareix en el PC la finestra representada a la **figura IV.30**, **Instal·lació de la interfície de l'usuari del telèfon**:

Figura IV.30: Instal·lació de la interfície de l'usuari del telèfon

A partir d'aquest moment s'haurà de seguir un diàleg en el mòbil. El conjunt d'imatges que queden incloses a la **figura IV.31**, **Diàleg a seguir al mòbil**, incorporen justament els passos a seguir:

1. Quan se'ns pregunti si volem instal·lar PrimeraApi contestarem afirmativament, <Sí>.

2. Apareixerà una advertència típica en la que se'ns avisa de què l'aplicació que es pretén instal·lar no es original del fabricant, i per tant és classificada com a perillosa. Però ja sabem que aquest perill no és real, així que s'ha d'optar per l'opció <Continuar>.

3. La següent qüestió fa referència a quina és la memòria on volem que s'instal·li l'aplicació. En aquest cas, és una pregunta intranscendent, ja que tenim suficient espai en qualsevol de les dues memòries disponibles (la del telèfon i la de la targeta), per tant escollim amb plena llibertat qualsevol de les opcions.

4. Una vegada s'ha dut a terme aquests passos, ja es procedeix a la instal·lació.

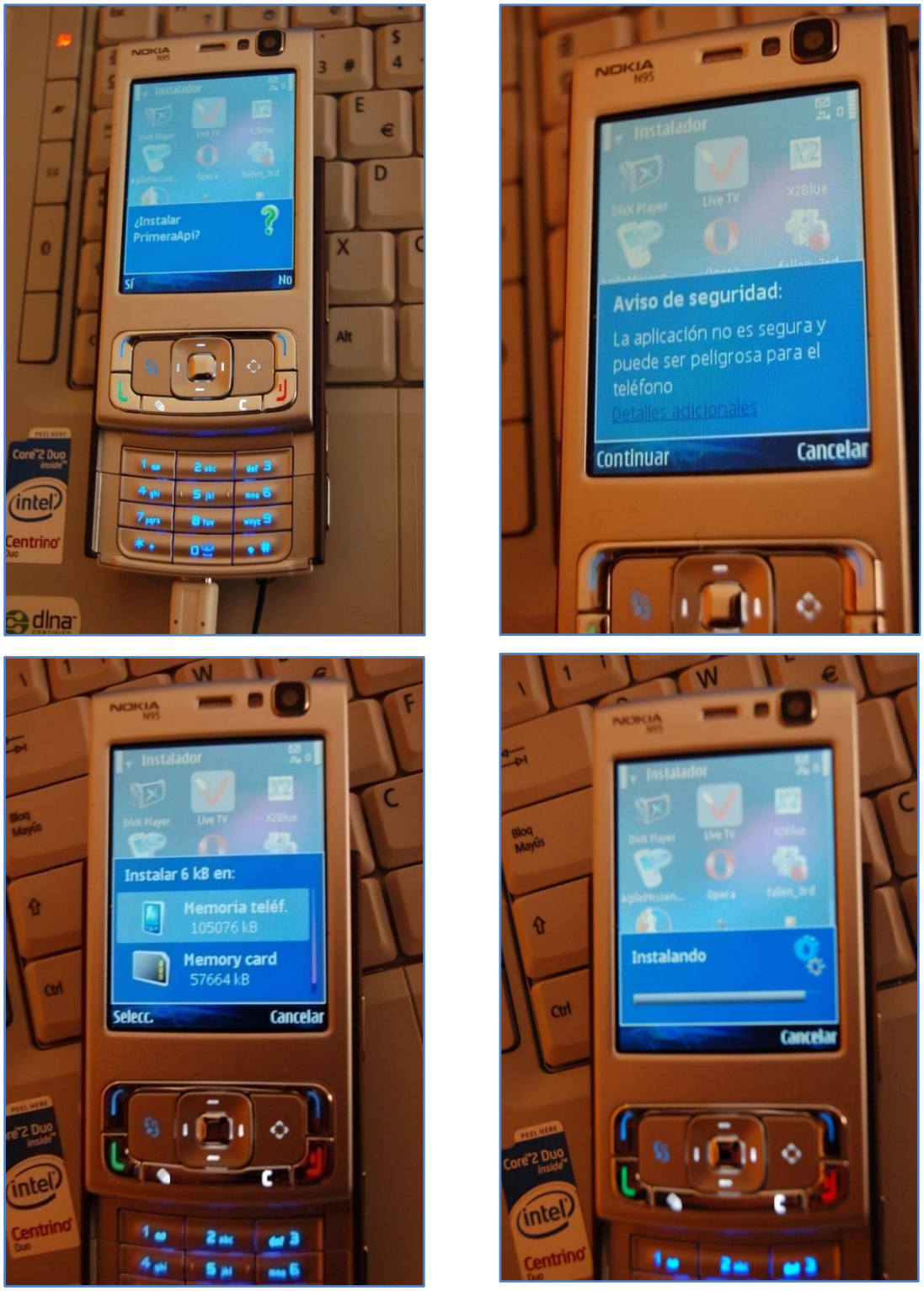


Figura IV.31: Diàleg a seguir al mòbil

A la següent imatge, la **figura IV.32**, **HelloMIDlet**, podem veure l'aplicació instal·lada al mòbil, i que porta per nom justament el mateix amb el que hem batejat la figura que ens ocupa, és a dir, HelloMIDlet:

Figura IV.32: HelloMIDlet

Finalment, ja només resta per comprobar si l'aplicació funciona de la mateixa forma en què ho fèia a la simulació del *NetBeans*. La **figura IV.33**, **Resultat real**, així ens ho confirma, ja que permet comprovar que, efectivament, funciona de manera correcta:

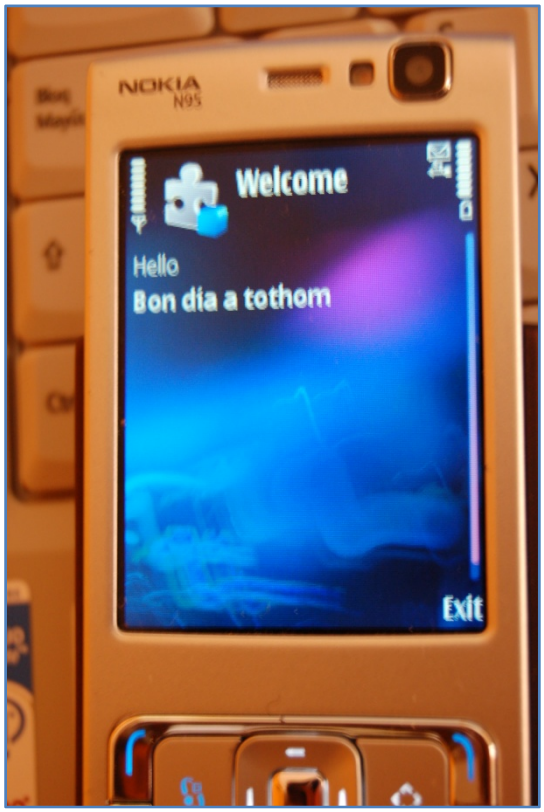


Figura IV.33: Resultat real

14. APÈNDIX V: L'APLICACIÓ "PFC2130406"

En aquest apartat s'inclou el codi font de les dues aplicacions, és a dir, dels dos MIDlets en el moment de la impressió de la memòria. Malgrat tot, pot patir certes modificacions fins la defensa del Projecte de Fi de Carrera.

14.1 APÈNDIX V.1: EL MIDlet "ENVIAMENT 6.0"

Codi font corresponent al MIDlet d'enviament:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
//import org.netbeans.microedition.lcdui.WaitScreen;
//import org.netbeans.microedition.util.SimpleCancellableTask;
/**
 * @author ROBERTO RODRIGUEZ FERNANDEZ
 */
public class enviament60 extends MIDlet implements CommandListener {

    private boolean midletPaused = false;
    //<editor-fold defaultstate="collapsed" desc=" Generated Fields ">
    private java.util.Hashtable __previousDisplayables = new java.util.Hashtable();
    private Command exitCommand;
    private Command screenCommand;
    private Command screenCommand1;
    private Command exitCommand1;
    private Form form;
    private StringItem stringItem;
    private WaitScreen waitScreen;
    private Form cuadretexte;
    private TextField textedeldgram;
    private TextBox textBox;
    private SimpleCancellableTask task;
    private Image image1;
    //</editor-fold>
    /**
     * The HelloMIDlet constructor.
     */
    //  public HelloMIDlet() {
    // }
    //<editor-fold defaultstate="collapsed" desc=" Generated Methods ">
```

```
    /**
     * Switches a display to previous displayable of the current displayable.
     * The <code>display</code> instance is obtain from the <code>getDisplay</code>
method.
     */
    private void switchToPreviousDisplayable() {
        Displayable __currentDisplayable = getDisplay().getCurrent();
        if (__currentDisplayable != null) {
            Displayable __nextDisplayable = (Displayable)
__previousDisplayables.get(__currentDisplayable);
            if (__nextDisplayable != null) {
                switchDisplayable(null, __nextDisplayable);
            }
        }
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method: initialize ">
    /**
     * Initilizes the application.
     * It is called only once when the MIDlet is started. The method is called before the
<code>startMIDlet</code> method.
     */
    private void initialize() {
    // write pre-initialize user code here

    // write post-initialize user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method: startMIDlet ">
    /**
     * Performs an action assigned to the Mobile Device - MIDlet Started point.
     */
```

```
    public void startMIDlet() {
        // write pre-action user code here
        switchDisplayable(null, getCuadretexte());
    // write post-action user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method: resumeMIDlet
">
    /**
     * Performs an action assigned to the Mobile Device - MIDlet Resumed point.
     */
    public void resumeMIDlet() {
        // write pre-action user code here
        switchDisplayable(null, getCuadretexte());
    // write post-action user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method:
switchDisplayable ">
    /**
     * Switches a current displayable in a display. The <code>display</code> instance
is taken from <code>getDisplay</code> method. This method is used by all actions in
the design for switching displayable.
     * @param alert the Alert which is temporarily set to the display; if
<code>null</code>, then <code>nextDisplayable</code> is set immediately
     * @param nextDisplayable the Displayable to be set
     */
    public void switchDisplayable(Alert alert, Displayable nextDisplayable) {
        // write pre-switch user code here
        Display display = getDisplay();
        Displayable __currentDisplayable = display.getCurrent();
        if (__currentDisplayable != null && nextDisplayable != null) {
```

```
            __previousDisplayables.put(nextDisplayable, __currentDisplayable);
        }
        if (alert == null) {
            display.setCurrent(nextDisplayable);
        } else {
            display.setCurrent(alert, nextDisplayable);
        }
    // write post-switch user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method: commandAction for Displayables ">
    /**
     * Called by a system to indicated that a command has been invoked on a particular displayable.
     * @param command the Command that was invoked
     * @param displayable the Displayable where the command was invoked
     */
    public void commandAction(Command command, Displayable displayable) {
        // write pre-action user code here
        if (displayable == cuadretexte) {
            if (command == exitCommand1) {
                // write pre-action user code here
                exitMIDlet();
            // write post-action user code here
            } else if (command == screenCommand1) {
                // write pre-action user code here
                switchToPreviousDisplayable();
            // write post-action user code here
            }
        } else if (displayable == form) {
            if (command == exitCommand) {
                // write pre-action user code here
```

```
            exitMIDlet();
        // write post-action user code here
        } else if (command == screenCommand) {
        // write pre-action user code here

        // write post-action user code here
        }
    } else {/* if (Displayable == waitScreen) {
    if (command == WaitScreen.FAILURE_COMMAND) {
    // write pre-action user code here
    // write post-action user code here
    } else if (command == WaitScreen.SUCCESS_COMMAND) {
    // write pre-action user code here
    // write post-action user code here
    }
    }*/
    // write post-action user code here
    }
//</editor-fold>
}
//<editor-fold defaultstate="collapsed" desc=" Generated Getter: exitCommand ">
/**
 * Returns an initiliazed instance of exitCommand component.
 * @return the initialized component instance
 */
public Command getExitCommand() {
    if (exitCommand == null) {
        // write pre-init user code here
        exitCommand = new Command("sortir", Command.EXIT, 0);
    // write post-init user code here
    }
    return exitCommand;
}
//</editor-fold>
```

```
    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: form ">
    /**
     * Returns an initiliazed instance of form component.
     * @return the initialized component instance
     */
    public Form getForm() {
        if (form == null) {
            // write pre-init user code here
            form = new Form("PROJECTE FINAL DE CARRERA ROBERTO RODRIGUEZ", new Item[]{getStringItem()});
            form.addCommand(getExitCommand());
            form.addCommand(getScreenCommand());
            form.setCommandListener(this);
        // write post-init user code here
        }
        return form;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: stringItem ">
    /**
     * Returns an initiliazed instance of stringItem component.
     * @return the initialized component instance
     */
    public StringItem getStringItem() {
        if (stringItem == null) {
            // write pre-init user code here
            stringItem = new StringItem("Hola:", "INICI DE PROGRAMA\n");
        // write post-init user code here
        }
        return stringItem;
    }
    //</editor-fold>
```

```
    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: screenCommand
">
    /**
     * Returns an initiliazed instance of screenCommand component.
     * @return the initialized component instance
     */
    public Command getScreenCommand() {
        if (screenCommand == null) {
            // write pre-init user code here
            screenCommand = new Command("Enviar", Command.OK, 0);
        // write post-init user code here
        }
        return screenCommand;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: waitScreen ">
    /**
     * Returns an initiliazed instance of waitScreen component.
     * @return the initialized component instance
     */
    /*
    public WaitScreen getWaitScreen() {
    if (waitScreen == null) {
    // write pre-init user code here
    waitScreen = new WaitScreen(getDisplay());
    waitScreen.setTitle("waitScreen");
    waitScreen.setCommandListener(this);
    waitScreen.setImage(getImage1());
    waitScreen.setTask(getTask());
    // write post-init user code here
    }
    return waitScreen;
    }
```

```
 */
//</editor-fold>

//<editor-fold defaultstate="collapsed" desc=" Generated Getter: task ">
/**
 * Returns an initiliazed instance of task component.
 * @return the initialized component instance
public SimpleCancellableTask getTask() {
if (task == null) {
// write pre-init user code here
task = new SimpleCancellableTask();
task.setExecutable(new org.netbeans.microedition.util.Executable() {
public void execute() throws Exception {
// write task-execution user code here
}
});
// write post-init user code here
}
return task;
} */
//</editor-fold>

//<editor-fold defaultstate="collapsed" desc=" Generated Getter: cuadretexte ">
/**
 * Returns an initiliazed instance of cuadretexte component.
 * @return the initialized component instance
 */
public Form getCuadretexte() {
    if (cuadretexte == null) {
        // write pre-init user code here
        String paraula = null;
        cuadretexte = new Form("cuadretexte", new Item[]{getTextedeldgram()});
        /*
        cuadretexte = new Form ("cuadretexte", new Item[] { getTextedeldgram () });
```

```
            cuadretexte.addCommand (getExitCommand1 ());
            cuadretexte.addCommand (getScreenCommand1 ());
            cuadretexte.setCommandListener (this);
             */

            getTextedeldgram();
            cuadretexte.addCommand(getExitCommand1());
            cuadretexte.addCommand(getScreenCommand1());
            cuadretexte.setCommandListener(this);
        // write post-init user code here
        }
        return cuadretexte;
    }
    //</editor-fold>
    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: textBox ">
    /**
     * Returns an initiliazed instance of textBox component.
     * @return the initialized component instance
     */
    public TextBox getTextBox() {
        if (textBox == null) {
            // write pre-init user code here
            textBox = new TextBox("textBox", null, 100, TextField.ANY);
        // write post-init user code here
        }
        return textBox;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: exitCommand1
">
    /**
     * Returns an initiliazed instance of exitCommand1 component.
     * @return the initialized component instance
```

```
     */
    public Command getExitCommand1() {
        if (exitCommand1 == null) {
            // write pre-init user code here
            exitCommand1 = new Command(" sortir aplicaci\u00F3", Command.EXIT,
1);
        // write post-init user code here
        }
        return exitCommand1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter:
screenCommand1 ">
    /**
     * Returns an initiliazed instance of screenCommand1 component.
     * @return the initialized component instance
     */
    public Command getScreenCommand1() {
        if (screenCommand1 == null) {
            // write pre-init user code here
            screenCommand1 = new Command("enviar ara", Command.OK, 0);
        // write post-init user code here
        }
        return screenCommand1;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: textedeldgram
">
    /**
     * Returns an initiliazed instance of textedeldgram component.
     * @return the initialized component instance
     */
```

```
    public TextField getTextedeldgram() {
        if (textedeldgram == null) {
            // write pre-init user code here
            textedeldgram = new TextField("ESCRIU UN MISSATGE:", null, 32,
TextField.ANY);
        // write post-init user code here
        }
        return textedeldgram;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: image1 ">
    /**
     * Returns an initiliazed instance of image1 component.
     * @return the initialized component instance
     */
    public Image getImage1() {
        if (image1 == null) {
            // write pre-init user code here
            image1 = Image.createImage(1, 1);
        // write post-init user code here
        }
        return image1;
    }
    //</editor-fold>
    /**
     * Returns a display instance.
     * @return the display instance.
     */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
```

```
     * Exits MIDlet.
     */
    public void exitMIDlet() {
        switchDisplayable(null, null);
        destroyApp(true);
        notifyDestroyed();
    }

    /**
     * Called when MIDlet is started.
     * Checks whether the MIDlet have been already started and initialize/starts or
resumes the MIDlet.
     */
    public void startApp() {

        if (midletPaused) {
            resumeMIDlet();
        } else {
            initialize();

            startMIDlet();


            getScreenCommand();
            if (screenCommand == null) {
                exitMIDlet();
            } else {
                try {



                    DatagramConnection sender =
                            (DatagramConnection)
Connector.open("datagram://192.168.1.255:32767");
```

```
                getDisplay().setCurrent(cuadretexte);
                TextField textedeldgram1 = null;

                do {
                   /*fins que textedeldgram1 tingui una mida de 10
                    * no continua
                    */
                   textedeldgram1 = getTextedeldgram();
                } while (textedeldgram1.size() < 16);

// creem un string anomenat paraula amb el contingut detextedeldgram1
                String paraula = textedeldgram1.getString();
// dataBytes conté els bytes de paraula
                byte[] dataBytes = paraula.getBytes();
// determinem la mida de dataBytes
                int llarg = dataBytes.length;
                /*es crea el datagrama per ser enviat per la
                que s'ha obert abans anomenada sender
                 */
                Datagram dgram = sender.newDatagram(llarg);

                byte[] buffer = dgram.getData();
                System.arraycopy(dataBytes, 0, buffer, 0, llarg);
                dgram.setLength(llarg);
                sender.send(dgram);

                //modtrem per pantalla el missatge enviat

                Form form3 = new Form("missatge:  " + paraula);
                form3.setCommandListener(this);
                getDisplay().setCurrent(form3);
                getDisplay().vibrate(2000);
```

```
                System.out.println("datagrama enviat");

            //getDisplay().setCurrent(new Form(paraula));
            } catch (Exception b) {
                b.printStackTrace();
                return;


            }

        }
    }
    midletPaused = false;
}

/**
 * deixem el MIDlet en pausa
 */
public void pauseApp() {
    midletPaused = true;
}

/**
 *tanquem el MIDlet
 */
public void destroyApp(boolean unconditional) {
}
}
```

14. APÈNDIX V.2: EL MIDlet "RECEPCIO 6.0"

Codi Font corresponent al MIDlet de recepció:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;

/**
 * @author ROBERTO RODRIGUEZ FERNANDEZ
 */
public class recepcio60 extends MIDlet implements CommandListener, ItemCommandListener {

    private boolean midletPaused = false;

    //<editor-fold defaultstate="collapsed" desc=" Generated Fields ">
    private TextBox textBox;
    private Form form;
    private ImageItem imageItem;
    private Command sortida;
    private Command exitCommand;
    private Command exitCommand1;
    private Ticker ticker;
    private Image image1;



    //<editor-fold defaultstate="collapsed" desc=" Generated Method: initialize ">
    /**
     * Initilizes the application.
     * It is called only once when the MIDlet is started. The method is called before the <code>startMIDlet</code> method.
```

```
     */
    private void initialize() {
    // write pre-initialize user code here

    // write post-initialize user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method: startMIDlet ">
    /**
     * Performs an action assigned to the Mobile Device - MIDlet Started point.
     */
    public void startMIDlet() {
        // write pre-action user code here
        switchDisplayable(null, getTextBox());
    // write post-action user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method: resumeMIDlet ">
    /**
     * Performs an action assigned to the Mobile Device - MIDlet Resumed point.
     */
    public void resumeMIDlet() {
        // write pre-action user code here
        switchDisplayable(null, getForm());
    // write post-action user code here
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method: switchDisplayable ">
    /**
```

```
    * Switches a current displayable in a display. The <code>display</code> instance is taken from <code>getDisplay</code> method. This method is used by all actions in the design for switching displayable.
    * @param alert the Alert which is temporarily set to the display; if <code>null</code>, then <code>nextDisplayable</code> is set immediately
    * @param nextDisplayable the Displayable to be set
    */
   public void switchDisplayable(Alert alert, Displayable nextDisplayable) {
      // write pre-switch user code here
      Display display = getDisplay();
      if (alert == null) {
         display.setCurrent(nextDisplayable);
      } else {
         display.setCurrent(alert, nextDisplayable);
      }
   // write post-switch user code here
   }
   //</editor-fold>

   //<editor-fold defaultstate="collapsed" desc=" Generated Getter: textBox ">
   /**
    * Returns an initiliazed instance of textBox component.
    * @return the initialized component instance
    */
   public TextBox getTextBox() {
      if (textBox == null) {
         // write pre-init user code here
         textBox = new TextBox("textBox", null, 100, TextField.ANY);
         textBox.setTicker(getTicker());
      // write post-init user code here
      }
      return textBox;
   }
   //</editor-fold>
```

```
//<editor-fold defaultstate="collapsed" desc=" Generated Getter: ticker ">
/**
 * Returns an initiliazed instance of ticker component.
 * @return the initialized component instance
 */
public Ticker getTicker() {
    if (ticker == null) {
        // write pre-init user code here
        ticker = new Ticker("");
    // write post-init user code here
    }
    return ticker;
}
//</editor-fold>

//<editor-fold defaultstate="collapsed" desc=" Generated Method: commandAction for Displayables ">
/**
 * Called by a system to indicated that a command has been invoked on a particular displayable.
 * @param command the Command that was invoked
 * @param displayable the Displayable where the command was invoked
 */
public void commandAction(Command command, Displayable displayable) {
    // write pre-action user code here
    if (displayable == form) {
        if (command == exitCommand1) {
        // write pre-action user code here

        // write post-action user code here
        } else if (command == sortida) {
            // write pre-action user code here
            exitMIDlet();
```

```
        // write post-action user code here
        }
    }
// write post-action user code here
}
//</editor-fold>

//<editor-fold defaultstate="collapsed" desc=" Generated Getter: form ">
/**
 * Returns an initiliazed instance of form component.
 * @return the initialized component instance
 */
public Form getForm() {
    if (form == null) {
        // write pre-init user code here
        form = new Form("form", new Item[]{getImageItem()});
        form.addCommand(getSortida());
        form.addCommand(getExitCommand1());
        form.setCommandListener(this);
    // write post-init user code here
    }
    return form;
}
//</editor-fold>
//<editor-fold defaultstate="collapsed" desc=" Generated Getter: sortida ">
/**
 * Returns an initiliazed instance of sortida component.
 * @return the initialized component instance
 */
public Command getSortida() {
    if (sortida == null) {
        // write pre-init user code here
        sortida = new Command("Exit", "exitMIDlet();", Command.EXIT, 0);
    // write post-init user code here
```

```
        }
        return sortida;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: imageItem ">
    /**
     * Returns an initiliazed instance of imageItem component.
     * @return the initialized component instance
     */
    public ImageItem getImageItem() {
        if (imageItem == null) {
            // write pre-init user code here
            imageItem = new ImageItem("imageItem", getImage1(), ImageItem.LAYOUT_DEFAULT, "<Missing Image>", Item.BUTTON);
            imageItem.addCommand(getExitCommand());
            imageItem.setItemCommandListener(this);
            imageItem.setDefaultCommand(getExitCommand());
        // write post-init user code here
        }
        return imageItem;
    }
    //</editor-fold>

    //<editor-fold defaultstate="collapsed" desc=" Generated Method: commandAction for Items ">
    /**
     * Called by a system to indicated that a command has been invoked on a particular item.
     * @param command the Command that was invoked
     * @param displayable the Item where the command was invoked
     */
    public void commandAction(Command command, Item item) {
        // write pre-action user code here
```

```
        if (item == imageItem) {
            if (command == exitCommand) {
            // write pre-action user code here

            // write post-action user code here
            }
        }
    // write post-action user code here
    }
    //</editor-fold>
    public Command getExitCommand() {
        if (exitCommand == null) {
            // write pre-init user code here
            exitCommand = new Command("Exit", Command.EXIT, 0);
        // write post-init user code here
        }
        return exitCommand;
    }

    //<editor-fold defaultstate="collapsed" desc=" Generated Getter: exitCommand1
">
    /**
     * Returns an initiliazed instance of exitCommand1 component.
     * @return the initialized component instance
     */
    public Command getExitCommand1() {
        if (exitCommand1 == null) {
            // write pre-init user code here
            exitCommand1 = new Command("Exit", Command.EXIT, 0);
        // write post-init user code here
        }
        return exitCommand1;
    }
    //</editor-fold>
```

```
//<editor-fold defaultstate="collapsed" desc=" Generated Getter: image1 ">
/**
 * Returns an initiliazed instance of image1 component.
 * @return the initialized component instance
 */
public Image getImage1() {
    if (image1 == null) {
        // write pre-init user code here
        try {
            image1 = Image.createImage("/icono.png");
        } catch (java.io.IOException e) {
            e.printStackTrace();
        }
    // write post-init user code here
    }
    return image1;
}
//</editor-fold>
/**
 * Returns a display instance.
 * @return the display instance.
 */
public Display getDisplay() {
    return Display.getDisplay(this);
}

/**
 * Exits MIDlet.
 */
public void exitMIDlet() {
    switchDisplayable(null, null);
    destroyApp(true);
    notifyDestroyed();
```

```
    }

    /**
     * Called when MIDlet is started.
     * Checks whether the MIDlet have been already started and initialize/starts or
resumes the MIDlet.
     */
    public void startApp() {
        if (midletPaused) {
            resumeMIDlet();
        } else {
            try {


                System.out.println("Establint connexio ");

                Form form1 = new Form("Establint connexió");
                getDisplay().setCurrent(form1);

                //crea una connexió anomenada receiver
                DatagramConnection receiver =
                        (DatagramConnection) Connector.open("datagram://:32767");


                byte[] buffer = new byte[256];
                System.out.println("preparant per rebre datagrama ");
                getDisplay().vibrate(20);

                Form form2 = new Form("Preparant per rebre datagrama");
                getDisplay().setCurrent(form2);

                Datagram dgram = receiver.newDatagram(buffer, buffer.length);
                for (;;) {
                    dgram.setLength(buffer.length);
```

```
            System.out.println("Esperant rebre datagrama ");
            Form form3 = new Form("Esperant rebre datagrama");
            form3.addCommand(new Command("Sortir", Command.EXIT, 0));
            form3.setCommandListener(this);
            getDisplay().setCurrent(form3);
            getDisplay().vibrate(1000);
            getDisplay().flashBacklight(2000);



// Es possa en espera per rebre un datagrama
            receiver.receive(dgram);



//Object display = Display.getDisplay(this);


            /*Form screen = new Form ("sdhjg");
            screen.append("fgjñlsgk");
            display = Display.getDisplay(this);
            display.toString();
             */
            System.out.println("datagrama rebut");
            Form form4 = new Form("connexió establerta");
            getDisplay().setCurrent(form4);
            getDisplay().vibrate(30);

            int llarg = dgram.getLength();
            System.out.println("Datagram received. Length is " + llarg);
            System.out.print(llarg);
```

```
                getDisplay().flashBacklight(100);
                buffer = dgram.getData();
                Form form5 = new Form("el datagrama ha arribat" + buffer);
                getDisplay().setCurrent(form5);

// Show the content of the datagram.
                buffer = dgram.getData();


                char char0 = (char) buffer[0];
                char char1 = (char) buffer[1];
                char char2 = (char) buffer[2];
                char char3 = (char) buffer[3];
                char char4 = (char) buffer[4];
                char char5 = (char) buffer[5];
                char char6 = (char) buffer[6];
                char char7 = (char) buffer[7];
                char char8 = (char) buffer[8];
                char char9 = (char) buffer[9];
                char char10 = (char) buffer[10];
                char char11 = (char) buffer[11];
                char char12 = (char) buffer[12];
                char char13 = (char) buffer[13];
                char char14 = (char) buffer[14];
                char char15 = (char) buffer[15];
                char char16 = (char) buffer[16];
                char char17 = (char) buffer[17];


                Form missatge = new Form("" + char0 + char1 + char2 + char3 + char4 +
char5 + char6 + char7 + char8 + char9 + char10 + char11 + char12 + char13 + char14
+ char15 + char16 + char17);
                missatge.addCommand(new Command("Sortir", Command.EXIT, 0));
                missatge.setCommandListener(this);
```

```
                getDisplay().setCurrent(missatge);
                getDisplay().wait();

                String paraula = "";
                for (int i = 0; i < llarg; i++) {
                   paraula = paraula + (char) buffer[i];
                }

                getDisplay().setCurrent(new Form(paraula));


                StringBuffer uu = new StringBuffer();
                for (int i = 0; i < llarg; i++) {
                   uu.append(buffer[i]);
                }
                getDisplay().setCurrent(new Form(uu.toString()));

             }


          } catch (Exception e) {
             e.printStackTrace();
             return;
          }

       }
       midletPaused = false;
    }

    /**
     * Called when MIDlet is paused.
     */
    public void pauseApp() {
       midletPaused = true;
```

```
    }

    /**
     * Called to signal the MIDlet to terminate.
     * @param unconditional if true, then the MIDlet has to be unconditionally terminated and all resources has to be released.
     */
    public void destroyApp(boolean unconditional) {
    }
}
```

15. APÈNDIX V: GLOSSARI DE TERMES

ABR (*Associative Based Routing*): Protocol reactiu d'encaminament en xarxes ad hoc que funciona per demanda iniciada a l'origen.

AODV (*Ad – hoc On – Demand Distance Vector routing*): Protocol d'encaminament ad hoc dirigit per taules d'encaminament.

API (*Application Programming Interface*): És una Interfície de Programació d'Aplicacions, és a dir, un conjunt de funcions, procediments i mètodes (en el cas de programació orientada a objectes) que ofereix certa llibreria amb la missió de ser utilitzat per un altra *software* com una capa d'abstracció.

BLUETOOTH: És un sistema sense fils que suporta la transmissió de dades. Té una abast d'uns 10 m i no és necessari que hi hagi visió directa entre els dispositius.

BROADCAST: Podríem traduir aquest terme per "difusió". Consisteix en una modalitat de transmissió de la informació en la que un node emissor envia informació de manera simultània a un conjunt de nodes receptors, sense que hi hagi necessitat de reproduir la mateixa transmissió node per node.

CDC (*Connected Device Configuration*): Fa referència a la configuració de dispositius connectats dins del conjunt de tecnologies per a computació mòbil J2ME, i estableix les capacitats bàsiques que ha de tenir un mòbil amb capacitat de connexió.

cHTML (*Compact HTML*): Subconjunt del llenguatge HTML per a dispositius petits com telèfons o PDA's. La reducció es deu a què en aquests aparells les capacitats de memòria o processament són molt més limitades.

DIRECCIÓ IP: Una direcció IP és un número que identifica de manera lògica i jeràrquica a una interfase d'un dispositiu ubicat a una xarxa que utilitza el protocol IP (*Internet Protocol*). Usualment, quan un usuari es connecta des del seu domicili a Internet ho fa utilitzant una direcció IP, direcció que pot canviar al reconnectar-se en un altre moment. Aquest tipus d'assignació dóna lloc a una adreça IP que rep el nom de "direcció IP dinàmica". Però hi ha d'altres llocs d'Internet que per pròpia natura necessiten estar permanentment connectats, per aquesta raó tenen una "direcció IP fixa", és a dir, que no canvia amb el temps. Aquest seria el cas dels servidors de correu, DNS, FTP públics i servidors de pàgines web, ja que així es possibilita la seva localització a la xarxa.

DSR (*Dynamic Source Routing*): Protocol reactiu amb encaminament a l'origen. La petició de ruta es fa per inundació

EclipseME: *Plugin* per a Eclipse que facilita la integració dels diferents *kits* de desenvolupament J2ME amb aquest IDE.

ENCAMINAMENT: És la funció per la qual es realitza la recerca d'un camí d'entre tots els possibles a una xarxa d'enviament de paquets caracteritzada per una topologia que gaudeix d'una gran connectivitat.

HOTSPOTS: És un punt d'accés a serveis de xarxa mitjançant un proveïdor de servei a Internet sense fils dins d'una zona de cobertura Wi-Fi. S'ubiquen a aeroports, biblioteques, centres de convencions, cafeteries, hotels, etc. Per tant, permet la connexió a Internet a llocs públics.

HTML (*Hyper Text Markup Language*): És un llenguatge per a la construcció de pàgines web. Permet la inclusió tant de text com d'imatges.

IDE (*Integrated Development Environment*): L'Entorn de Desenvolupament Integrat no és altra cosa que un programa que està composat per un conjunt d'eines que són d'utilitat al programador a la seva tasca. Consisteix en un editor de codi, un compilador, un depurador i un constructor d'interfície gràfica, que en conjunt conformen un marc de treball amigable.

I – MODE: Conjunt de tecnologies i protocols dissenyats amb l'objectiu de poder navegar per pàgines dissenyades específicament per a dispositius mòbils. És una tecnologia que competeix amb la WAP.

INFRAROJOS: Sistema que permet la transmissió de dades entre dispositius que disposin del port adient, i sense necessitat d'haver-los de connectar mitjançant un cable. L'abast aproximat és de 1 m, i és necessària la visió directe entre els dispositius.

J2EE (*Java 2 Enterprise Edition*): És una plataforma de programació per a desenvolupar i executar software d'aplicacions en llenguatge de programació JAVA amb arquitectura de N nivells distribuïda.

J2ME (*Java 2 Micro Edition*): La plataforma *Java*, *Micro Edition* o *Java ME* és un conjunt d'**API**s en llenguatge de programació Java, orientades a productes de consum com ara PDAs, telèfons mòbils o electrodomèstics. Es tracta d'una plataforma JAVA que proporciona un entorn robust i flexible d'aplicacions que es poden crear per executar-se a molts tipus de dispositius.

J2SE (*Java 2 Standard Edition*): Conjunt d'API's del llenguatge de programació JAVA útils per a molts programes de la plataforma JAVA.

MIDlet SUITE: Són petits programes implementats en J2ME que venen pre-instal·lats als telèfons mòbils o bé es poden descarregar-se fàcilment. S'executen en el mateix telèfon, i poden ser aplicacions d'oci, consulta, informació, etc. Els MIDlet's es poden descarregar, utilitzar, emmagatzemar, o esborrar en qualsevol moment.

PDA (*Personal Digital Assistant*): Computador de ma que inicialment es va dissenyar com a agenda electrònica amb sistema de reconeixement d'escriptura. Les seves funcionalitats han augmentat considerablement i actualment es poden fer servir com a una computadora domèstica.

***PEER TO PEER* (XARXA *PEER TO PEER*)**: Xarxa informàtica "entre iguals", és a dir, on no hi ha clients ni servidors, sinó que tots els nodes es comporten simultàniament com a clients i servidors respecte de la resta de nodes.

PLUGIN: Aplicació informàtica amb la missió d'interactuar amb una altra aplicació amb l'objectiu de donar-li una utilitat o funcionalitat específica.

PROTOCOL D'INTERNET: És un protocol no orientat a connexió que és emprat tant per l'origen com pel destí per a la comunicació de dades mitjançant una xarxa de paquets commutats.

PROTOCOL VoIP: Recursos que possibiliten el viatge de la senyal de veu per Internet sota un protocol IP, és a dir, la informació s'envia en format digital en paquets.

Wi-Fi (*WIreless – FIdelity*): Les sigles provenen de les paraules en anglès *Wireless – Fidelity*. Es tracta d'un conjunt d'estàndards dissenyats per a xarxes sense fils, i que es basen en les especificacions IEEE 802.

Wi-Max: Estàndard de comunicació per a ones de radio d'última generació. Permet la recepció de dades per microones i retransmet la informació mitjançant ones de ràdio. Presenta més cobertura i ample de banda que el Wi-Fi.

WIRELESS: Sistema de comunicació en el que no es fa servir cap mena de mitjà físic de propagació, sinó que es fa servir la modulació d'ones electromagnètiques.

15. APÈNDIX VI: GLOSSARI DE SIGLES

ABR*: Associative-Based Routing*

AODV*: Ad-hoc On-Demand Distance Vector routing*

API*: Application Programming Interface*

CDC*: Connected Device Configuration*

cHTML*: Compact HTML*

CLDC*: Connected Limited Device Configuration*

CVM*: Compact Virtual Machine*

DSR*: Dynamic Source Routing*

HTML*: Hyper Text Markup Language*

IDE*: Integrated Development Environment*

IMP*: Information Module Profile*

J2EE*: Java 2 Enterprise Edition*

J2ME*: Java 2 Micro Edition*

J2SE*: Java 2 Standard Edition*

KVM*: Kilo Virtual Machine*

MIDP*: Mobile Information Device Profile*

MLAN*: Metropolitan LAN, Metropolitan Local Area Networks*

PDA*: Personal Digital Assistant*

RLAN*: Radio LAN, Radio Local Area Networks*

SDK*: Software Development Kit*

SMS*: Sort Messaging Service*

TORA*: Temporary Ordered Routing Algorithm*

WAP*: Wireless Application Protocol*

WAS*: Wireless Access System*

Wi-Fi*: WIreless FIdelity*

Wi-Max*: WMAN*

WLAN*: Wireless Local Area Networks*

WMAN*: Wireless Metropolitan Area Networks*

VoIP: Veu sobre Protocol d'Internet

17. BIBLIOGRAFIA

17.1 LLIBRES I DOCUMENTACIÓ *ONLINE*

Adams, L. (2002). *Tutorial de programação J2ME. Parte 2. Plataforma Java 2 Micro Edição (J2ME).* Consultat el 10 / Octubre / 2007, a http://www.wirelessbrasil.org/wirelessbr/colaboradores/corbera_martins/j2me_02.html

Carniel, J., & Teixerira, C. (2005). *Apostila de J2ME*. Consultat el 25 / Octubre / 2007, a http://www.daneprairie.com

Chaparro, D., Pelegrín, J., & Rodríguez, R. (27 / Junio / 2003). *Algoritmo de encaminamiento para los RCX de los Legos Mindstorm. Universidad Rey Juan Carlos.* Consultat el 10 / Octubre / 2007, a http://dchaparro.net/estudios/encaminamiento_ad-hoc.pdf

ComNets, IKOM, University of Bremen. (24 / Juliol / 2003). *JAdhoc System Design.* Consultat el 12 / Febrer / 2007, a http://www.comnets.uni-bremen.de/~koo/JAdhoc.pdf

Diez-Andino, G., & García, R. M. (30 / Noviembre / 1999). *Java2 micro edition. Un primer vistazo*. Consultat el 28 / Octubre / 2007, a http://www.mygnet.net/manuales/j2me/java2_micro_edition_un_primer_vistazo.165

eclipseME. (2005). *eclipse ME. J2ME Development using Eclipse."From zero to mobile in minutes"*. Consultat el 28 / Octubre / 2007, a http://eclipseme.org/docs/installEclipseME.html#step2d

García, A. (10 / Agost / 2004). *Curso: Programación de juegos para móviles con J2ME. Capítulo 3: El lenguaje Java*. Consultat el 20 / Octubre / 2007, a http://www.mailxmail.com/curso/informatica/j2me/capitulo3.htm

García, R. (2002). *Tutorial de programação J2ME. Parte 1. Plataforma Java 2 Micro Edição (J2ME).* Consultat el 20 / Octubre / 2007, a http://www.wirelessbrasil.org/wirelessbr/colaboradores/corbera_martins/j2me_01.html

J2ME Polish. (2007). *J2ME Polish 2.0.2*. Consultat el 19 / Octubre / 2007, a http://www.j2mepolish.org/devices/platform.html

Lin, C. (2004). *AODV Routing Implementation for Scalable Wireless Ad-Hoc Network Simulation (SWANS).* Consultat el 12 / Febrer / 2008, a http://jist.ece.cornell.edu/docs/040421-swans-aodv.pdf

Mahmoud, Q. H. (2002). *Learning Wireless Java. Help for J2ME Developers.* Sebastopol: O'Reilly & Associates.

Subiela, R., & León, A. (2005). *Simulación de protocolos de encaminamiento en redes móviles ad hoc con NS-2.* Consultat el 19 / Octubre / 2007, a http://w3.iec.csic.es/ursi/articulos_gandia_2005/articulos/TE2/396.pdf

Sun microsystems . (2007). *Mobility Modules for NetBeans 2004Q1. Tutorial*. Consultat el 22 / Octubre / 2007, a http://www.sun.com/hwdocs/feedback

17.2 WEBS D'INTERÈS

Curso: Programación de juegos para móviles con J2ME. Capítulo 7: Nuestro primer MIDlet

http://www.mailxmail.com/curso/informatica/j2me/capitulo7.htm

En aquest text s'explica pas a pas cóm construir i executar un MIDlet amb *J2ME Wireless Toolkit 2.0*, més concretament, el que hem abordat a la present memòria *Hello World*.

ipoki mobile. Programación de dispositivos móviles y noticias relacionadas. Consejos para programadores J2ME (y BlackBerry) - 3

http://hipoqihmovil.wordpress.com/category/j2me/

Web on es parla del llenguatge J2ME, de l'aplicació NetBeans i del desenvolupament per a telèfons mòbils, entre d'altres temes.

J2ME, Java Wireless Message API (WMA)

http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=wma

Tutorial on es fa una introducció sobre J2ME i les seves característiques que té Java per l'enviament i recepció de missatges des d'aplicacions per a mòbils en format sms.

J2ME Polish

http://www.j2mepolish.org

Aquí hi trobarem dades sobre els dispositius que suporten CDC o CLDC.

JiST / SWANS Java in Simulation Time / Scalable Wireless Ad hoc Network Simulator

http://jist.ece.cornell.edu/sw.html

Descàrregues i *links* d'interès.

ktoolbar

http://nemesis-bcn.tripod.com/TEST/PDA/Tool.htm

On trobarem informació sobre el cicle que s'ha de seguir en el desenvolupament de projectes, i l'estructura que presenten.

Perlab: Pervasive Computing Laboratory

http://www.it.uc3m.es/pervasive

Es tracta d'una web desenvolupada pel departament d'enginyeria telemàtica de la universitat Carlos III de Madrid.

todo symbian. El sitio de los smartphones

http://www.todosymbian.com/secart25.html

En aquesta pàgina hi ha dades sobre el llenguatge J2ME, i l'IDE J2ME *Wireless Toolkit*.

Unión Internacional de telecomunicaciones. Tendencias del mercado mundial.

http://www.itu.int/osg/spu/spunews/2003/oct-dec/wi-fi-es.html

Plana que aborda els sistemes Wi-Fi.

Universität Bremen

http://www.aodv.org/

Plana de la universitat de Bremen des d'on es poden fer descàrregues relacionades amb AODV.

WEBSPHERE DEVICE DEVELOPER 5.6: Desarrollando Aplicaciones J2ME

http://www.it.uc3m.es/florina/docencia/swc/TutorialWSDD.htm

En aquest apartat de la web de la universitat Carles III de Madrid hi ha força dades sobre el desenvolupament d'aplicacions en J2ME.

17.3 SOFTWARE UTILITZAT

ECLIPSE (2007). Versió gratuïta i sense període de carència. Descarregat el 23 / Octubre / 2007 de www.eclipse.org

ECLIPSE ME (2007). Versió gratuïta i sense període de carència. Descarregat el 23 / Octubre / 2007 de http://eclipseme.sourceforge.net

GEL (2007). Versió gratuïta i sense període de carència. Descarregat el 23 / Octubre / 2007 de www.gexperts.com

SDK J2SE (2007). Versió gratuïta i sense període de carència. Descarregat el 23 / Octubre / 2007 de http://java.sun.com/j2se

Wireless Toolkit J2ME (2007). Versió gratuïta i sense període de carència. Descarregat el 23 / Octubre / 2007 de http://wireless.java.sun.com/allsoftware/