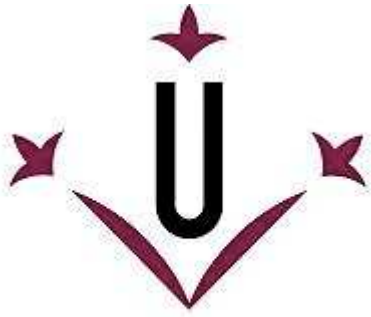
Universitat de Lleida
Escola Politècnica Superior
Enginyeria Tècnica en Informàtica de Sistemes

Treball de final de carrera



Registre d'Accions d'Usuari en Entorns Symbian

Autor: Carles Pastor Sumalla
Director: Juan Miguel López Gil
Juliol 2008

Índex de continguts

Índex de figures

Índex de taules

Capítol 1

Introducció

1.1 Motivació i objectius del projecte

La motivació principal d'aquest projecte, ha estat la de conèixer el món del desenvolupament per a sistemes operatius mòbils (computació mòbil) i aportar una millora al problema de comunicació entre l'usuari i el dispositiu. Per tant, podríem incloure el treball, en el que s'anomena *Interacció Persona-Ordenador* (IPO).

D'entre els diversos sistemes operatius per a dispositius mòbils (Windows Mobile, Linux...etc.), s'ha triat Symbian per que la seva quota de mercat és la més gran en l'actualitat, i s'ha triat la plataforma S60, per que és la plataforma que més dispositius mòbils Symbian utilitzen, tal i com podem apreciar en la gràfica següent:

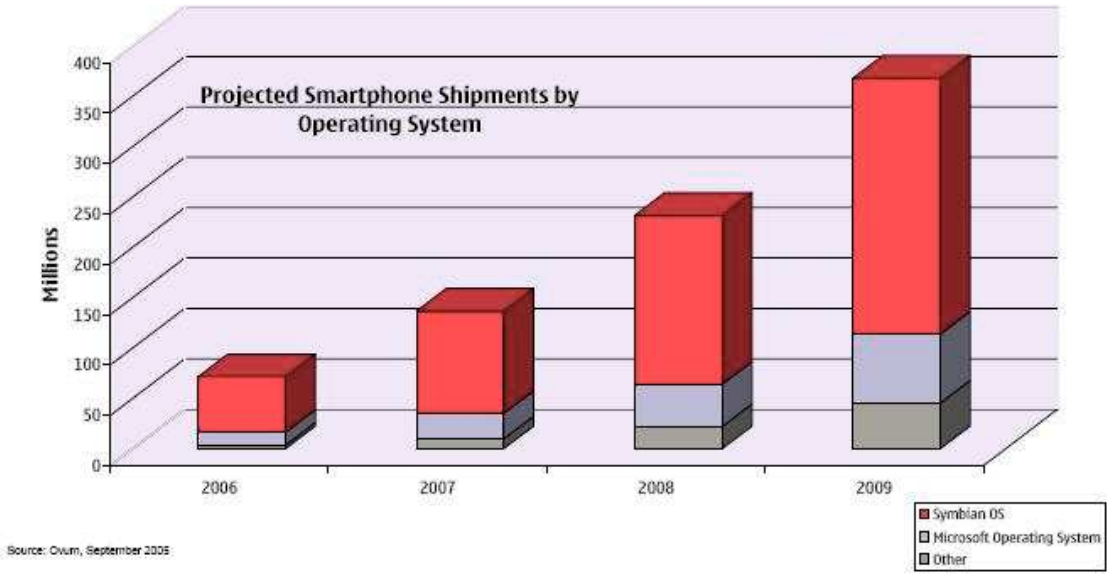


Figura 1: Gràfica de la quota de mercat en sistemes operatius mòbils (http://www.event.tietoenator.com/se/tcd04/presentations/PerA.pdf)

L'objectiu del projecte és desenvolupar una aplicació en l'entorn Symbian per a la plataforma S60 que sigui capaç de capturar i enregistrar les accions realitzades per un usuari a través del teclat físic del dispositiu.

La informació més important a capturar és el codi de la tecla premuda, però també s'hi pot associar altra informació útil, com pot ser l'hora i la data de la pulsació ó informació referent al procés actual i altres característiques. Aquesta informació es guardarà en forma de línies de text pla, una per a cada event de teclat, i es podrà llegir des de qualsevol editor de text convencional.

Les dades emmagatzemades poden ser utilitzades per a diverses finalitats (trobar errors, elaborar estadístiques d'ús, realitzar proves d'usuari, etc.), totes elles amb una finalitat en comú, millorar l'aplicació i fer més fàcil la comunicació amb l'usuari. D'altra banda, es pretén que aquest projecte serveixi de base per a futurs treballs.

Capítol 2 Anàlisi de requeriments

En aquest apartat s'intentarà analitzar els requeriments necessàris per a desenvolupar la nostra aplicació. S'analitzaràn els entorns mòbils i els seus Sistemes Operatius per tal de tenir prouta informació per a donar suport a la nostra elecció de Symbian com a Sistema Operatiu. També s'analitzaràn les particularitats pròpies de la tecnología mòbil així com la seva arquitectura, que haurem de tenir en compte a l'hora de treballar amb ella.

2.1 Què és Symbian?

En l'actualitat, els dispositius mòbils (telèfons, PDAS...), formen part de la vida quotidiana de la major part de les persones i segurament, aquesta tendència s'anirà incrementant. Els primers dispositius mòbils eren grans, amb poca autonomia i amb unes funcions limitades si les comparem amb els actuals, que han millorat molt tant en hardware com en software, convertint-se en autèntics ordenadors mòbils. Aquests ordenadors mòbils necessiten sistemes operatius cada cop més potents que siguin capaços d'aprofitar al màxim les capacitats tecnològiques de que disposen. Un d'aquests sistemes operatius (SO) és Symbian: The open mobile operating system (el sistema operatiu mòbil obert).

Segons la definició que podem trobar a Internet:

"*S60, the world.s most popular software for smatphones, lets you add new applications to your mobile device and keeps you connected to your favorite internet services in much the same way you do with your PC.*"

(S60, el sistema operatiu més popular per a smatphones, et permet afegir noves aplicacions al teu dispositu mòbil i et manté connectat als teus serveis d'Internet preferits de la mateixa manera que fas amb el teu PC.)

Figura 2: Logotip de Symbian

2.2 Qui ha creat Symbian?

Symbian va ser producte de l'aliança de diverses empreses de telefonia cel.lular, entre les que es troben Nokia, Sony Ericsson, PSION, Samsung, Siemens, Arima, Beng, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric (creador dels telèfons FOMA junt a Fujitsu, Sharp, etc.), Panasonic, Sharp. Els seus orígens provenen del seu avantpassat EPOC32, utilitzat en PDA's i Handhelds de PSION. En 2003 Motorola va vendre el 13 per cent de la seva participació a Nokia, el que va fer que es quedés amb el 32,2 per cent de la companyia, però més endavant Motorola, després de no tenir l'èxit esperat amb els seus terminals de Linux, va tornar al món de Symbian comprant-li a Sony Ericsson el 50% de les accions.

L'objectiu de Symbian, va ser el de crear un sistema operatiu per a terminals mòbils que pugues competir amb el Palm o el Windows Mobile de Microsoft. En la Figura 2 podem apreciar el percentatge d'ús de Symbian per part dels fabricants.

La majoría de mòbils amb Symbian són de la companyia Nokia:

- NOKIA 3650
- NOKIA 3660
- NOKIA 6120
- NOKIA 6260
- NOKIA 6600
- NOKIA 6620
- NOKIA 6630
- NOKIA 6680
- NOKIA 6681
- NOKIA 7650
- NOKIA 7610
- NOKIA 6670
- NOKIA 9300
- NOKIA 9500
- NOKIA 5700
- NOKIA N-Gage
- NOKIA N-Gage QD
- NOKIA E50
- NOKIA E60
- NOKIA E61
- NOKIA E61
- NOKIA E70
- NOKIA N72
- NOKIA N73

- NOKIA N80
- NOKIA N81
- NOKIA N82
- NOKIA N95
- NOKIA N96

Sony:

- Sony Ericsson P800
- Sony Ericsson P802
- Sony Ericsson P900
- Sony Ericsson P990
- Sony Ericsson W950
- Sony Ericsson M600i

Siemens:

- Siemens SX1

Motorola:

- Motorola A1000
- Motorola A1010
- Motorola A920
- Motorola A925
- Motorola A728
- Motorola Z8
- Motorola Z10

Foma: Tots

Samsung:

- SGH-D720
- SGH-D730

Panasonic:

- Panasonic X700
- Panasonic X800

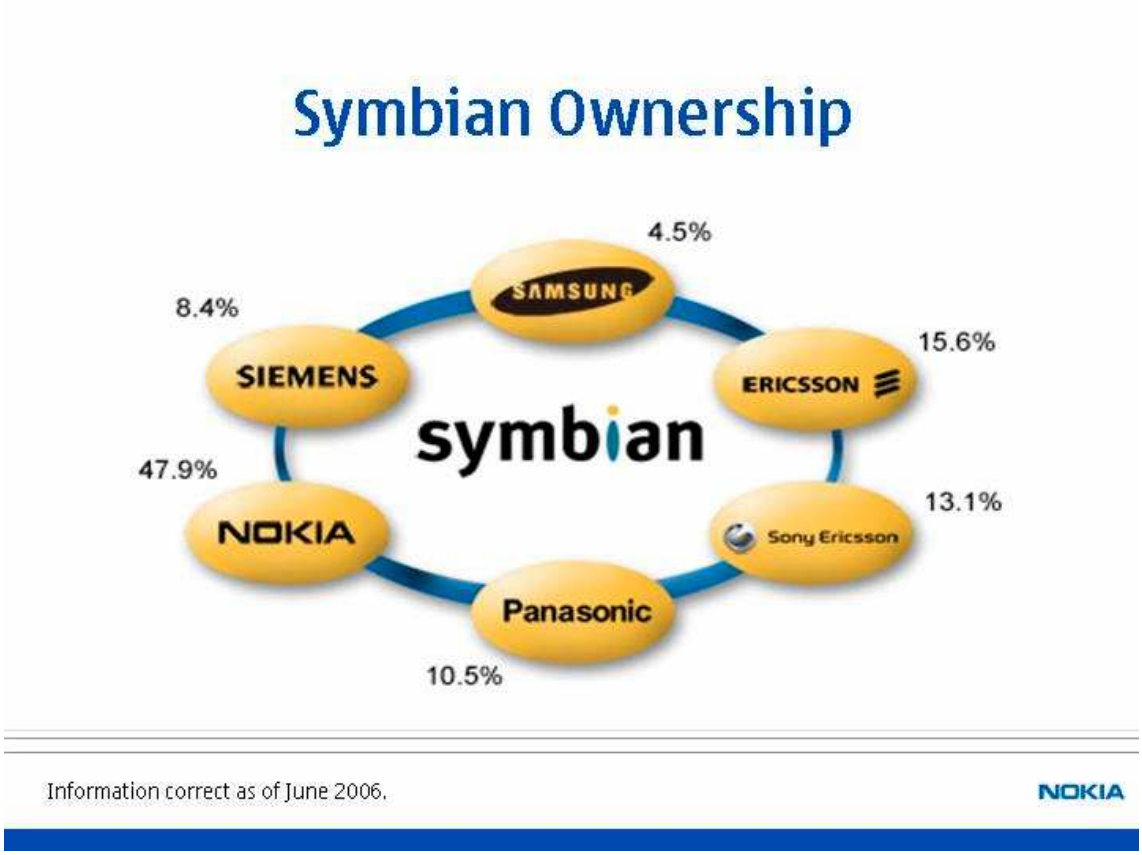


Figura 3: Companyíes que utilitzen Symbian

Tot i que Symbian és el SO més utilitzat, té diversos competidors com per exemple Windows Mobile, CE y Palm. El que distingeix a Symbian dels altres, és que va ser el sistema operatiu que va convertir els mòbils en computadores de butxaca sense passar pels PDA, una cosa semblant al que va fer Suunto al ser la marca que va convertir els rellotges d'esport en computadores de canyella. I també que Symbian és la garantia d'interoperatibitat, de compatibilitat y d'escalabilitat per estar en el centre dels principals fabricants de mòbils. La garantia d'una execució fiable en el complicat ecosistema mòbil gràcies a les aplicacions, fitxers, xarxes, comunicacions..etc. És com una espècie de Bluetooth per als dispositius mòbils. De fet, el propi nom Symbian, es va triar després d'un procés de *naming* en el que es va triar com a referència a la simbiosis, el concepte grec de varies parts que interactuen i conviuen en el mateix espai.

2.3 Què és la Serie S60?

Symbian disposa de quatre plataformes per al seu Sistema Operatiu, denominades Serie 60, Serie 80, Serie 90, UIQ i Forma. La majoria dels

dispositius mòbils utilitzen la Serie 60, tot i que tots els de Sony Ericsson treballen amb UIQ, com també ho fa Motorola.

2.4 Particularitats generals dels dispositius mòbils

Els dispositius mòbils tenen característiques especials comparades amb els equips de sobretaula. Les restriccions físiques i la naturalesa de la seva funció inclouen característiques que s'han de tenir en compte quan ens disposem a desenvolupar per a aquests aparells.

Aquestes característiques són:

• Diferenciació entre dispositius, incloent el tamany, que afecta a limitacions de grandària de pantalla i d'introducció de dades per part de l'usuari

• Utilització de l'energia eléctrica. A diferència dels equips de sobretaula que estan permanentment connectats a una font d'energia elèctrica, els dispositius mòbils només disposen d'una quantitat limitada d'energia que es proporcionada per la bateria i per tant, tant el hardware com el software han de consumir la menor quantitat d'energia possible.

• Restriccions de memòria. Fins a l'actualitat i tot i que cada cop més els dispositius mòbils disposen de més memòria, la seva quantitat encara és molt inferior a la dels seus parents els equips de sobretaula.

• Mobilitat. Aquesta és la característica principal que s'ha de tenir en compte a l'hora de dissenyar una aplicació per a aquesta mena de dipositius.

• Connexió a xarxes. Ja siguin Wifi, UMTS... Cada cop hi ha més tipus de connexions disponibles per aquest tipus de dispositius i cada cop s'efectuen més connexions per part dels usuaris.

• Utilització continuada. És una altra de les diferències importants amb els equips de sobretaula, que es re inicien en repetides ocasions a diferència dels dispositius mòbils, que han de funcionar de forma continuada la major part del temps. Tant el sistema operatiu com les aplicacions, han de garantir un funcionament estable i amb un bon rendiment del dispositiu.

• Entrades per teclat limitades. Només alguns dispositius disposen de teclat qwerty però molts d'altres disposen d'un teclat amb molt poques tecles.

Aquestes característiques especials, requereixen una gran robustesa tant en la plataforma com en les aplicacions. Els dispositius s'utilitzen durant llargs períodes de temps sense re iniciar-se. La utilització de la memòria i el consum d'energia, són temes importants a tractar, com també ho són les situacions inesperades en les que la connexió de la xarxa es perd o una trucada entrant interromp l'aplicació en curs.

2.5 Requeriments

Per a que l'objectiu de l'aplicació que es pretén construir s'assoleixi amb èxit -un programa que registri en un arxiu de text les tecles d'usuari i algunes característiques relacionades de forma totalment transparent per a l'usuari-, són necessaris una sèrie de requisits que s'esmentaràn a continuació. El que es tracta és de trossejar el problema en altres de més petits per tenir una guia de treball.

Per una banda, és convenient estudiar el sistema de que disposa Symbian per a manipular arxius, ja que aquest serà un requisit indispensable quan hage'm d'escriure els events d'usuari. En cas de que ens trovessim davant d'un sistema poc intuïtiu, s'hauria de valorar la opció d'implementar una classe que ens permetés manipular els arxius de manera senzilla.

Un altre requisit a tenir en compte, és la possibilitat de configurar les dades que ens interessi capturar de l'usuari, és a dir, que pugem especificar si volem o no que un atribut quedi registrat a l'arxiu cada cop que es realitza un event.

També caldrà considerar un dels requeriments més importants, sense el qual l'aplicació no te sentit. És tracta de la capacitat d'execució en segon pla, de manera que el nostre programa pugui realitzar la seva feina de manera totalment invisible per a l'usuari. Sense aquesta característica, la nostra aplicació seria la única que l'usuari podria accedir.

Un cop resolts els problemes anteriors, haurem de plantejar la llista de dades que ens interessa capturar i trobar la forma d'aconseguir-ho.

Ja que cada model de dispositiu mòbil té unes característiques de hardware diferents, seria una bona idea capturar les dades més significatives de cada màquina (cpu, memòria, etc.) per guardar-les al nostre arxiu de text.

Una informació interessant sobre l'usuari, és conèixer la seva posició en la interfície d'usuari, és a dir, saber en quina aplicació es trova en un moment determinat ó en quin ítem del menú principal.

I per últim, encara que no es tracta d'una característica imprescindible, pot ser útil la possibilitat de dotar al nostre programa de la capacitat d'executar-se cada cop que s'inicia el dispositiu, sense haver d'executar l'aplicació de forma manual.

2.6 Característiques tècniques

2.6.1 Característiques generals

Hi ha diferents llenguatges per a desenvolupar aplicacions per a Symbian que es poden observar a la Figura 4 i que s'esmenten a continuació:

- Symbian C++ ofereix accés total a totes les característiques de la plataforma S60, i aporta integració amb altres aplicacions i serveis en S60. Open C i Open C++ afegeixen interfícies de plataforma independent a l'entorn natiu, obrint noves oportunitats a les comunitats open source. Permet desenvolupar aplicacions que es compilen en el llenguatge natiu del dispositiu i per tant aquests programes son els més ràpids i optimitzats.
- Java Mobile Edition (Java ME) ofereix la possibilitat de crear aplicacions portàtils que no solament s'executen en plataformes S60 sinó que s'estenen als 700 milions de dispositius amb Java disponibles al mercat.
- Python per S60 es un potent llenguatge de "scripts" que ofereix la possibilitat d'innovar i crear nous conceptes.
- Flash Lite Permet el ràpid desenvolupament d'aplicacions que utilitzen gràfics de forma intensiva.
- Web Run-Time Permet crear aplicacions mòbils riques i de forma fàcil utilitzant les tecnologies web estàndards com HTML, CSS i Javascript.

Figura 4: Components del desenvolupament en Symbian

2.6.2 Arquitectura

L'arquitectura Symbian, consta principalment de les *Aplicacions S60*, de les *Aplicacions Java*, dels *Serveis de la Plataforma s60*, de les *Extensions del Sistema Operatiu S60* i del propi *Sistema Operatiu Symbian*, com s'aprecia a la següent figura:

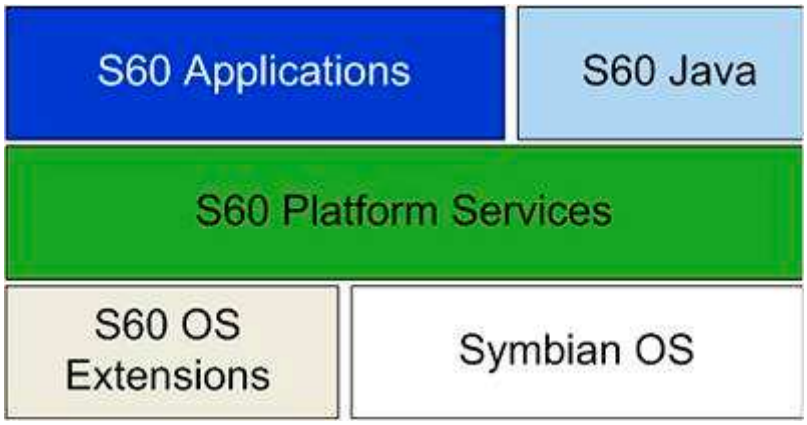


Figura 5: Arquitectura de Symbian

Extensions de Symbian OS

Les extensions de Symbian OS són una sèrie de capacitats que permeten a la plataforma S60 interactuar amb les funcions del hardware del dispositiu com l'alerta de vibració, les llums i l'estat de la carga de la bateria.

Serveis de la plataforma S60

Els serveis d'aplicació de S60 son una sèrie de capacitats que proveeixen certes funcionalitats bàsiques per a les aplicacions S60. Aquests serveis son utilitzats per les aplicacions S60 embedded i també estan disponibles per ser utilitzades en aplicacions de third-party. Aquests serveis inclouen:

- *Application Framework Services.* Proveeix de les capacitats bàsiques per executar aplicacions i servidors, state-persistence management i components d'interfície d'usuari.

- *UI Framework Services.* Proveeix d'una aparença concreta als components IU (interfície d'usuari) i manejar events IU.

- *Graphics Services.* Proveeix de capacitats per a la creació de gràfics i el seu dibuixat en la pantalla.

- *Location Services.* Permet a la plataforma S60 saber la posició en la que es troba.

- *Web-Based Services.* Proveeix serveis per establir connexions i interactua amb la funcionalitat basada en web, incloent navegació, descàrrega d'arxius i missatgeria.

- *Multimedia Services.* Proveeix de les capacitats per reproduir àudio i vídeo i també suporta "streaming" i reconeixement de veu.

- *Communication Services.* Proveeix el suport per a comunicacions tant locals com no locals, des de tecnologia Bluetooth a trucades de veu.

Application Framework Services

A continuació es nombren els serveis de tipus Application Franmework més importants:

- *PIM Application Services.* Proveeix les característiques principals per aplicacions PIM, inloent contactes, calendari i tasques de manteniment i també funcions associades com el bloc de notes i capacitats de rellotge.

- *Messaging Application Services.* Proveeix el suport per a varis tipus de missatges, com el servei de missatges curts (SMS), el servei de missatges multimèdia (MMS), e-mail, BIO missatges (smart messaging) i missatgeria instantània (IM).

- *Browser Application Services.* Proveeix de les capacitats per veure contingut Web, incloent suport per Flash Lite, renderitzat de video, *Scalable Vector Graphics–Tiny (SVG-T)* rendering i renderitzat d'àudio

S60 Java™ Technology Services

Els serveis de Tecnologia Java per S60 proveeixen suport per a la plataforma Java™ , Micro Edition (Java ™ ME) Java™ Technology for the Wireless Industry (JTWI) specification (JSR-185). El suport a la plataforma S60 inclou l'especificació Connected Limited Device Configuration 1.1 (JSR-139), i la extensió Mobile Information Device Profile 2.0 (JSR-118) per a aquesta especificació.

A més, es suporta un rang d'APIS addicional per permetre accedir al sistema d'arxius de S60, a les dades PIM, utilitzar la tecnologia Bluetooth, la missatgeria,

àudio, vídeo serveis Web, serveis de seguretat i confiança, informació de la localització, Protocol d'Inicialització de Sessió (SIP), i gràfics 3D.

Web Run-Time

Web Run-Time (WRT) es un entorn que permet als dispositius S60 executar "Web widgets". Presentat en la tercera edició (3rd Edition), Feature Pack 2, WRT està potenciat per WebKit - la mateixa tecnologia "open-source" que també és utilitzada en el navegador web de S60.

S60 Applications

Les aplicacions S60 estan integrades amb la plataforma que està disponible a l'usuari del dispositiu, incloent PIM, missatgeria, aplicacions multimèdia i perfils.

Tots els serveis de la plataforma S60 esmentats es poden veure reflectits en la figura següent:

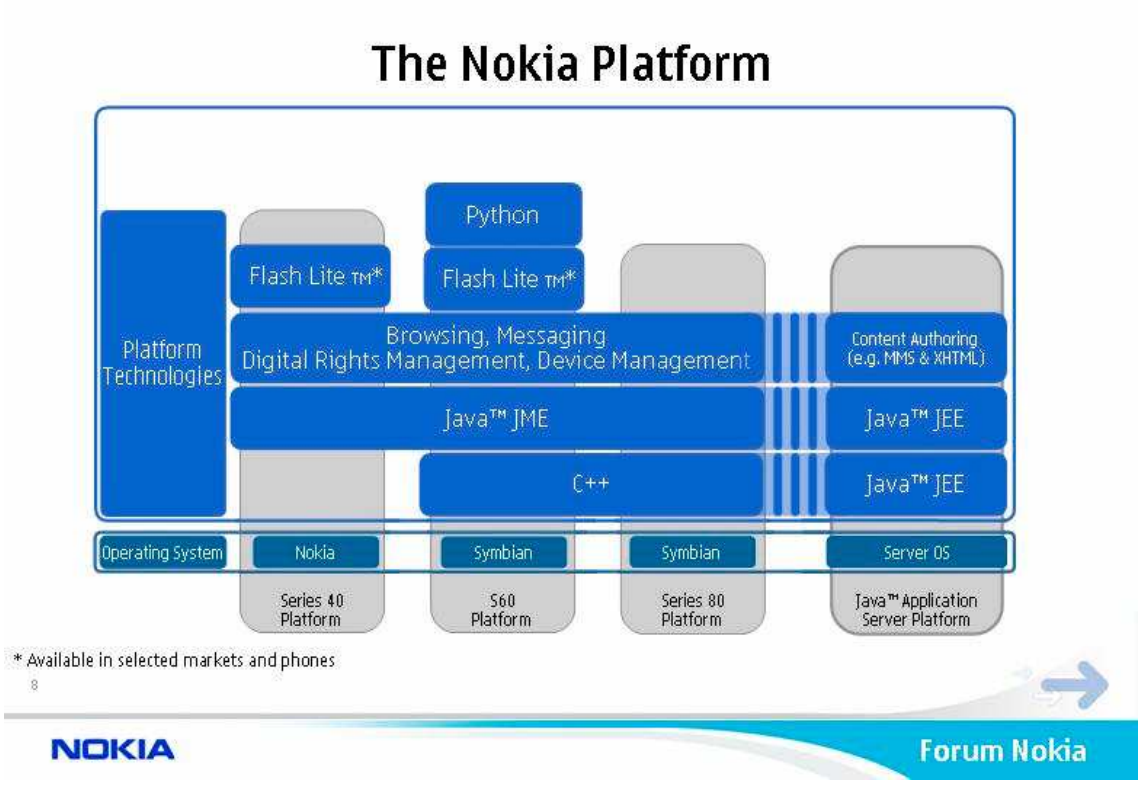


Figura 6: Plataforma NOKIA

2.6.3 Tecnologies de la plataforma S60

A continuació es nombraran les tecnologies clau de la plataforma S60 que estan disponibles en la majoria de dispositius S60:

2.6.3.1 Comunicacions

- Telefonia:

  És la tecnologia bàsica de qualsevol telèfon mòbil que permet la comunicació per mitjà de la veu.

- Infrarojos

  Sistema de transmissió de dades entre els dispositius a través d'infrarrojos. Cada cop en més desús a causa de la tecnologia bluetooth.

- Bluetooth

  Sistema de transmissió de dades entre dispositius a través de ones de ràdio, que permet més velocitat que els infrarojos i afegeix més llibertat de moviment.

- WLAN

  Sistema de transmissió de dades sense fils utilitzada habitualment per a establir connexions a internet.

2.6.3.2 Missatgeria

- SMS

  Sistema d'enviament de missatges curts de text entre dispositius mòbils a través de l'operadora corresponent.

- MMS

  Sistema d'enviament de missatges amb text i contingut multimèdia entre dispositius mòbils a través de l'operadora corresponent.

- E-mail

  Servei de correu electrònic.

- IM (Instant Messaging)

  Missatgeria Instantània. Permet la comunicació en temps real entre dos o més usuaris.

2.6.3.3 Navegació

Tots els dispositius S60 suporten navegació, alguns també WAP i HTTP. El navegador web de S60 està basat en el motor web de components open source de Apple Inc.'s - el software utilitzat en el navegador Safari de Apple, conegut com a *KHTML*.

2.6.4 Edicions S60

A mesura que les versions de Symbian han anat avançant, també ho han fet les versions de la plataforma S60, per adaptar-se als nous canvis. Lògicament, cada versió o edició te vàries té característiques pròpies i diferenciades de les altres edicions.

Algunes de les característiques comuns a totes les edicions són:

- Aplicacions Multimèdia. Càmera, Visor d'imatges, RealPlayer, aplicacions Media Gallery i gravadora de veu.

- Fons d'escriptori. Una aplicació per emmagatzemar informació personal protegida. Targetes virtuals, com targetes de crèdit, poden ser emmagatzemades i després ser utilitzades per transaccions de pagament a través d'Internet.

- Temes. Que permeten personalitzar a través de la interfície d'usuari i inclou imatge de fons, icones i bitmaps. Això permet donar una aparença personalitzada a cada dispositiu.

Els SDK de Symbian proporcionen una eina molt potent consistent en un emulador del sistema operatiu, és a dir una màquina virtual on executar els nostres programes i obtenir el mateix resultat que obtindríem si l'execució es realitzés en el propi dispositiu. A continuació trobem la captura d'aquest emulador en plena execució:

Figura 7: Emulador de 3rd Edition FP2 SDK

2.6.4.1 S60 1st Edition

S60 1st Edition està basat en Symbian OS v6.1 i va ser una evolució d'un sistema operatiu anomenat Epoc. Data de l'any 2000 i implementa les següents tecnologies:

- PIM com calendari, agenda, àlbum de fotos, agenda d'events i explorador d'arxius.

- Pinboard

- RealPlayer

- XHTML Mobile Profile (XHTML MP)

2.6.4.2 S60 2nd Edition

S60 2nd Edition està basat en Symbian OS v7.0s. i implementa les següents tecnologies:

- Multihoming

- JAVA MIDP 2.0

- Dual IP *stacks* suportant formats IPv4 i IPv6

- ECom

- EDGE

- Lightweight multithreaded multimedia framework.

- CDMA (WCDMA)

- WAP 2.0

Han hagut tres versions de"*Feature Pack*" per la 2nd Edition, que incrementen les característiques del software, com es mostra en la Taula següent:

Feature pack	**Característiques**	**Dispositiu exemple**
Feature Pack 1 (Symbian OS v7.0s)	Càmera Megapixel amb 4x zoom, gravació i reproducció de vídeo clips	Nokia 7610 Nokia 6670
Feature Pack 2 (Symbian OS v8.0a)	Càmera d' 1.3-megapixel amb 6x zoom, WCDMA/EDGE, IPv6	Nokia 6630

Feature Pack 3 (Symbian OS v8.1a)	Suport per IU escalablet (Tamanys de pantalla variables)	Nokia N90

Taula 1: Característiques del Feature Pack

2.6.4.3 S60 3rd Edition

És l'ultima edició del SDK de Symbian (actualment en la versió 9) i també la més complerta. Afegeix noves característiques però cancel.la la retrocompatibilitat amb l'edició anterior, és a dir que els programes que es desenvolupen en la 2nd Edition no funcionaran en aquesta.

Algunes de les novetats que inclou:

- Web Run-Time Widget
- CSS
- JavaScript
- AJAX.
- Web 2.0 services and Internet content.
- Transition Effects
- NetBeans IDE 6.0- Enhanced emulator startup speed
- Platform UID

Aquesta és la edició triada per desenvolupar el nostre projecte, que apart de ser la més complerta de totes, és la última i per tant la que millor funciona amb els últims aparells del mercat. La llista de dispositius que suporten aquesta edició és la següent:

- LG-KT610
- LG KS10
- Nokia E66
- Nokia E71
- Nokia 5320 XpressMusic
- Nokia N78
- Nokia 6220

- Nokia 6210 Navigator
- Nokia N96
- Nokia N82
- Nokia E51
- Nokia N95 8GB
- Nokia N81
- Nokia N81 8GB
- Nokia 6121
- Nokia 6120
- Nokia 5700
- Nokia N77
- Nokia E90
- Nokia E61i
- Nokia E65
- Nokia 6110 Navigator
- Nokia N93i
- Nokia N76
- Nokia 6290
- Nokia N75
- Nokia N95
- Nokia E62
- Nokia E50
- Nokia 5500 Sport
- Nokia N73
- Nokia N93
- Nokia N92
- Nokia N71
- Nokia N80
- Nokia E70
- Nokia E61
- Nokia E60
- Nokia 3250
- Nokia N91
- Samsung SGH-L870
- Samsung SGH-G810
- Samsung SGH-i560
- Samsung SGH-i550
- Samsung SGH-i450
- Samsung SGH-i400
- Samsung SGH-i520

2.7 L'IDE Carbide

Per a poder desenvolupar per a les plataformes de Symbian, NOKIA va crear un Framework que pretenia unificar totes les eines disponibles per al programador, aquest Framework s'anomena Carbide i permet treballar amb múltiples plataformes i múltiples llenguatges. Està clarament basat en el Framework obert Eclipse i pot ser estés amb altres productes i plug-ins d'Eclipse.

Carbide es centra en tres àrees de desenvolupament primari:

- Eines de desenvolupament Carbide per a Java
- Eines de desenvolupament Carbide per Symbian OS C++
    - Carbide.c++ és una família d'eines de desenvolupament basades en Eclipse que suporten desenvolupament Symbian OS en la plataforma S60 , la plataforma Series 80, UIQ i MOAP. Hi ha tres versions:
        - Carbide.c++ Professional Edition. Per a usuaris avançats.
        - Carbide.c++ Developer Edition. Eines productives per desenvolupar aplicacions
        - Carbide.c++ Express. Gratuït
    - Carbide.vs és un *plug-in* que permet als usuaris de Visual Studio desenvolupar en llenguatge C++.

Eines Carbide per personalitzar la interfície d'usuari

- Carbide.ui és una família d'eines gràfiques i WYSIWYG que permeten la personalització de IU en dispositius S60 i Series 40.El primer producte d'aquesta família és Carbide.ui Theme Edition.

Per a desenvolupar aquest projecte, s'ha triat Carbide com a *framework* pel fet d'estar basat en Eclipse, ja que es suposa que heretarà la major part de les seves bondats i a més, es l'entorn oficial creat per NOKIA que és el fabricant amb major nombre de dispositius Symbian. Com a llenguatge de programació, s'ha triat C++ ja que proporciona un accés més directe al *hardware* dels dispositius i ens permet realitzar certes accions que no ens permetria un llenguatge com Java, i a més també obtindrem una major velocitat d'execució.

Capítol 3

Tecnologia utilitzada

En aquest capítol es nombrarà la tecnologia utilitzada, tant per al desenvolupament com per al requeriment del software creat.

3.1 Hardware de desenvolupament

- Terminal NOKIA N6120
- PC AMD 3.2 Ghz i 512 de memòria RAM

3.2 Hardware requerit

- Qualsevol terminal que suporti l'entorn Symbian i la la tercera edició de la plataforma S60

3.3 Software de desenvolupament

- IDE: Carbide 1.3
- SDK: S60 3rd edition for Symbian OS Feature Pack 2 (3rd FP2)
- SDK: S60 2rd edition for Symbian OS Feature Pack 3 (2rd FP3)
- SO: Windows XP SP2
- Barak's singMe Beta 1.0.8

3.4 Conceptes de Seguretat

A l'hora de desenvolupar software per a dispositius mòbils, s'han de tenir molt en compte les restriccions aplicades pel propi sistema operatiu (capa de seguretat), que són necessàries, o del contrari qualsevol podria crear software perjudicial per al propi sistema o per al dispositiu. Per exemple, algunes d'aquestes restriccions es centren en l'accés a determinats arxius del sistema.

En Symbian, aquest control per a la seguretat es basa en el que s'anomena *capabilities* (capacitats), és a dir, les funcionalitats previstes per les *APIs* de la plataforma. Dit d'una altra manera, només podran accedir a determinades funcionalitats, aquelles aplicacions que hagin estat autoritzades.

L'accés a les capacitats ve determinat per la configuració del dispositiu i la manera com l'apliació ha set signada. Signar una aplicació, és un procés pel qual

aquesta rep un certificat (dos arxius amb extensió .cer i .key que s'han d'aplicar correctament al nostre programa), que li permet accedir a determinades capacitats.

En altres versions del sistema operatiu Symbian, no era necessari certificar els programes per a poder-los instal.lar, però la 3a edició del SDK S60 introdueix la signatura obligatòria de les aplicacions. Això significa que l'aplicació no s'instal.la si no ha estat signada. En el gestor d'aplicacions del dispositiu, hi ha una configuració que controla si les aplicacions "no confiables" es poden instal.lar. Si aquesta configuració està definida com "*Solo firmadas*" no es podran instal.lar els programes auto firmats, però si posem l'opció a "*Todas*" es podran instal.lar.

Hi ha dos tipus de signatura:

- La signatura amb qualsevol clau privada. Per a aconseguir la unitat i confirmar la integritat de l'arxiu sis. L'aplicació *makekeys* pot ser utilitzada per a crear la clau necessària i *signsis* per a firmar l'aplicació. Les dues aplicacions son entregades amb el SDK.
- La signatura amb una clau privada específica. Per a aconseguir la certificació. Signar l'aplicació per a que un certificat arrel confiable en el dispositiu, certifiqui la signatura del paquet d'instal.lació de l'aplicació.

El primer cas s'ha de fer pel programador durant el de procés de desenvolupament per a que l'aplicació s'instal.li al dispositiu. L'altre cas es pot fer per mitjà del programa Symbian Signature i les capacitats poden ser atorgades a través d'aquest.

Si volem que la nostra aplicació sigui comercial, la millor opció es el Certificat Firmat per Symbian, que ens permet l'accés més alt ja que l'aplicació es tractada com a "confiable". Per a aconseguir una certificació d'aquest nivell, primer el programa ha de passar per les mans de l'equip Symbian que comprovaran que no es faci cap mal ús dels terminals. Aquest sistema te un preu que està entre els 185 $ i 350 $.

Encara hi ha un altre sistema de certificació que s'anomena Auto Certificació i que està pensada per a grans desarrolladores ja que dona "confiança" a les aplicacions de forma automàtica però amb un preu d'uns 10.000 $ a l'any.

3.5 Certificat a través de web

El principal mètode per firmar les aplicacions a través de la web, passa per *https://www.symbiansigned.com/* ,on podrem aconseguir diversos tipus de firma

de la pròpia Nokia. Hi ha quatre opcions que podrem triar: "*Open Signed Online*", "*Open Signed Offline*", "*Express Signed*" i "*Certified Signed*" essent la primera opció la que menys requeriments demana i la última la que més. Per aconseguir un certificat per a la nostra aplicació, s'ha utilitzat "*Open Signed Online*" i s'han realitzat els següents passos:

1- Seleccionar "*Open Signed Online*" de la web de *SymbianSigned*

2- Omplir les dades requerides:

   - IMEI o número de sèrie: Identificador del terminal. Per obtenir aquesta dada introduïm **#06#* a través del teclat del dispositiu i se'ns mostrarà el número.

   - Email: No pot ser d'una compta gratuïta (hotmail, gmail...etc.)

   - Aplicació: El nostre arxiu *.sis* que serà enviat a Nokia

3- Clicar sobre el *link* de confirmació que s'ha enviat al nostre correu

4- Clicar sobre el *link* de descàrrega que s'ha enviat al nostre correu

Amb aquests quatre passos, el que es fa basicament és enviar a Nokia el nostre IMEI i el nostre arxiu *sis*. Un cop rebut, l'empresa aplica de forma automàtica una sèrie de filtres per comprovar si es tracta d'una aplicació segura. Si el resultat es positiu s'envia un *link* al correu del propietari per a que pugui descarregar el *sis* firmat, però si el resultat és negatiu, el que s'enviarà serà una negativa explicant que l'aplicació és perillosa.

Un altre mètode que també s'ha seguit, és la firma a través d'una web china (http://www.opda.net.cn/) dedicada a dispositius mòbils que atorga certificats a canvi únicament de que l'usuari es registri al fòrum i envii un mínim de cinc missatges. Aquesta signatura, com l'anterior, només funcionarà per a un dispositiu mòbil en concret (per a un IMEI concret).

Les dades que s'han de deixar són, el model del mòbil, la data actual i el IMEI del dispositiu. Passats uns dies, es publiquen a la web, les URLS que permeten descarregar els arxius *.cer* i *.key* necessaris per a signar el nostre programa. Amb aquests dos arxius i qualsevol programa per a signar com *Barak's singMe Beta 1.0.8* es poden signar les aplicacions amb un resultat similar a l'aconseguit amb SymbianSygned.

Capitol 4

Desenvolupament de l'aplicació

En aquest capítol, es veurà tot el desenvolupament de la nostra aplicació, començant per la seva arquitectura i donant resposta a tots els requeriments plantejats en l'apartat 2.5 (Requeriments).

4.1 Arquitectura de l'aplicació

Com s'ha esmentat en l'anàlisi de requeriments, es vol construir una aplicació que sigui capaç de registrar en un arxiu de text els events d'usuari i algunes característiques relacionades de forma totalment transparent per a aquest. El cicle de funcionament comença quan l'usuari prem una tecla en qualsevol situació possible del dispositiu mòbil (ja sigui des d'una aplicació en curs o des de el menú principal). La tecla es recollida pel sistema operatiu i també per la nostra aplicació (que anomenarem *LOG2*) per després escriure's en l'arxiu de text. A continuació es mostra un esquema d'aquest procés:

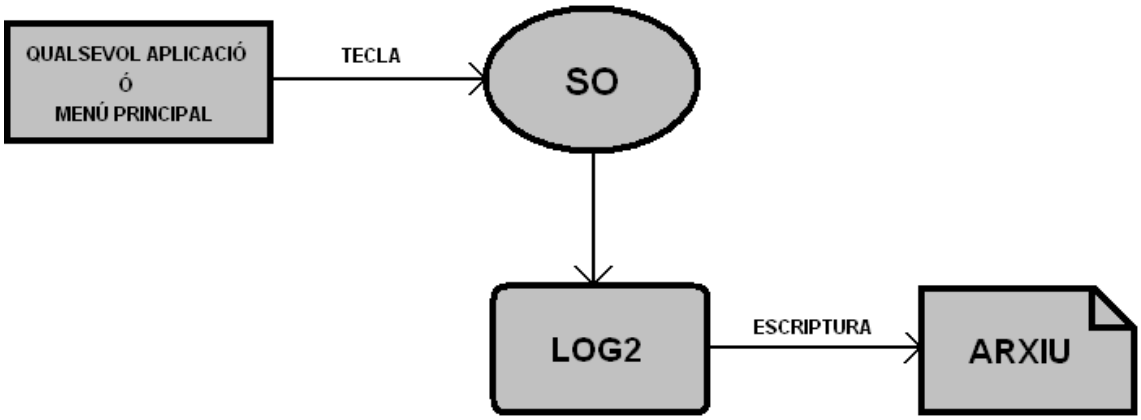


Figura 8: Esquema del cicle de funcionament general

Internament, l'aplicació consta d'un cicle amb 4 estats que s'executaran cada cop que es premi una tecla. Aquests estats o processos són: "escoltar" - on l'aplicació espera a que l'usuari premi alguna tecla-, "capturar event de teclat" - on s'obté la tecla concreta amb el seu codi-, "registrar dades addicionals" - on s'obtenen altres dades- i finalment "escriure arxiu de text" on les dades capturades s'escriuen en un fitxer. A continuació es mostra una figura esquemàtica d'aquest procés:

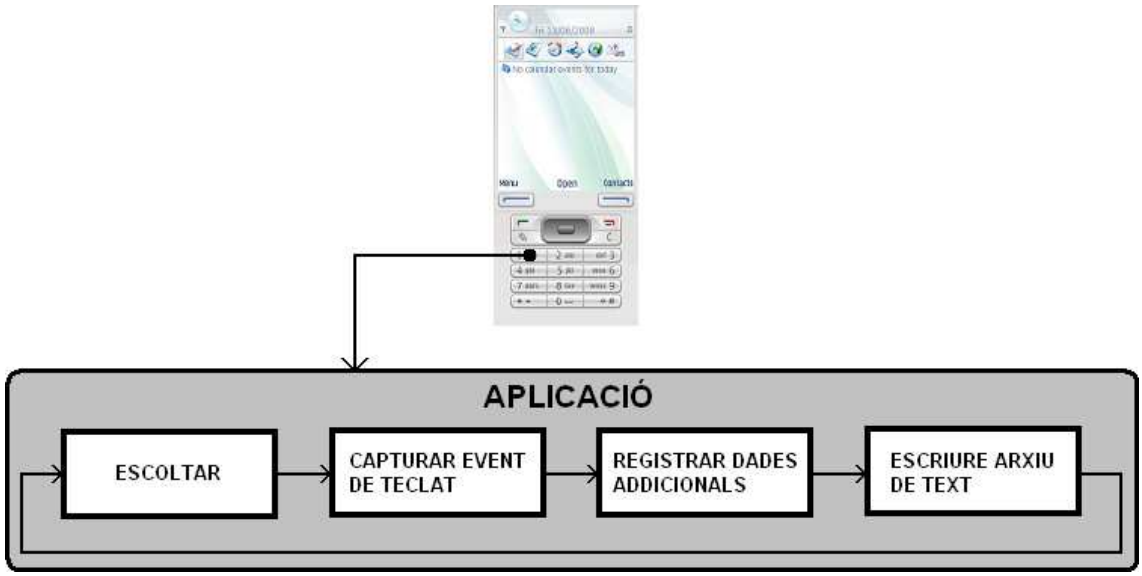


Figura 9: Esquema del cicle de funcionament de l'aplicació

Un cop que hage'm implementat aquesta aplicació, hauríem de ser capaços de crear un arxiu de text pla interpretable per qualsevol editor, que contingui totes les tecles pitjades per l'usuari en un moment determinat. Per a diferenciar el registre d'una tecla a una altra, es farà a través del salt de línia, és a dir que a cada tecla li correspondrà una línia diferent. Com les línies poden ser molt llargues depenent de la informació que recollim de cada tecla, la opció d'ajustament de línia del editor de text no hauria d'estar habilitada.

El resultat és pot observar en la figura següent (en aquest exemple s'ha guardat el codi de la tecla, l'hora completa i la data):

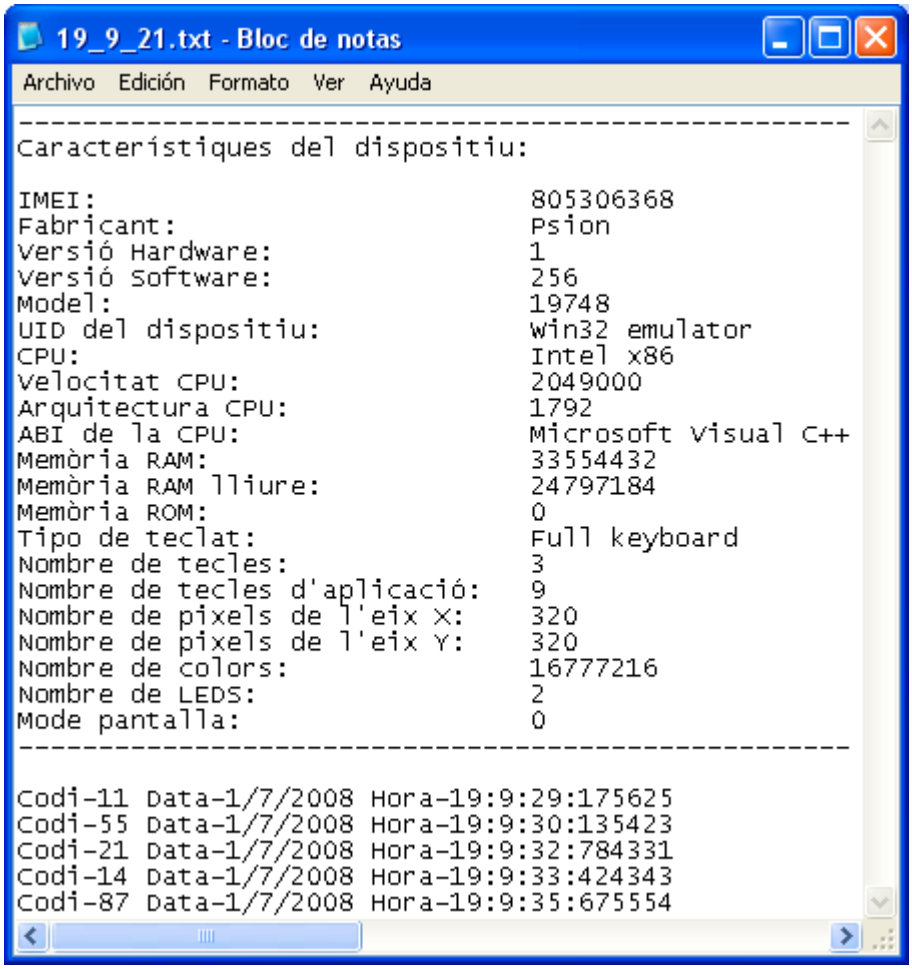


Figura 10: Exemple d'arxiu de text pla generat per l'aplicació LOG2

4.2 Manipulació d'arxius

4.2.1 Creació de la classe *file.h*

Tal i com s'ha esmentat a l'anàlisi de requeriments, l'escriptura i lectura d'arxius ens és fonamental, per això cal explorar l'ús dels arxius en Symbian C++. La classe RFile és la que s'encarrega de la manipulació de fitxers, però degut a que el seu ús no resulta massa còmode, crearem la classe *file.h* que ens facilitarà aquesta feina i farà el seu ús similar al que s'aconsegueix amb les funcions en C. En aquesta classe implementarem els mètodes més bàsics: *Open, Close, Read, Write i Seek.*

Cada cop que la aplicació s'executi, es crearà un nou arxiu de text amb el següent format:

- 8 dígits en total i extensió *txt*
- Els 2 primers dígits corresponen a l'hora actual
- Els 2 següents dígits corresponen al minut actual
- Els 2 següents dígits corresponen al segon actual

Per exemple, si l'aplicació s'executa a les 12:21:53, l'arxiu creat seria:
12_21_53.txt

Per a poder crear l'arxiu de text en cada execució, ho fem al mètode InitializeControlsL de la classe principal que es crida en el constructor de la mateixa classe (ConstructL), donant format a la variable global NomArxiu. Aquest format el creem a partir d'un rellotge del tipus TTIME que anomenem rellotge. El codi sería de la següent manera:

```
TTime rellotge;
rellotge.HomeTime();
NomArxiu.Format(_L("%d%d%d%d.txt"),rellotge.DateTime().Day(),rello
tge.DateTime().Month(),rellotge.DateTime().Hour(),rellotge.DateTime().
Minute());
file.Open(NomArxiu,File::OMWrite | File::OMCreate);
```

Un cop creat l'arxiu, escrivim, per a cada event de teclat, una línia amb les corresponents dades associades, com poden ser: codi de la tecla, hora actual, data actual, estat de la bateria, aplicació actual...etc.

Les dades escrites només poden ser de tipus caràcter, d'aquesta manera, podran ser interpretades per qualsevol editor de text. Com algunes de les dades es reben en format enter (int), s'han de transformar a caràcter (char). Aquí ens trovem amb un dels principals problemes de la programació Symbian C++, la conversió de formats. No s'ha trobat cap funció predefinida per a la conversió que necessitem. Sí que hi ha una funció en C (atoi) que serveix per a la operació inversa, és a dir, per a la conversió de caràcters a enters, però aquesta no ens interessa. L'única alternativa, ha estat la de crear una funció pròpia anomenada IntToChar que s'encarregui de guardar en format caràcter un enter rebut per paràmetre.

4.3 Arxiu configuració config.txt

4.3.1 Funcionament

Volem un sistema que ens permeti especificar en un arxiu de text pla i amb extensió *txt* quines dades volem capturar en l'arxiu de log. S'ha triat aquest format per que és el més senzill.

Aquest arxiu de configuració contindrà cada dada que es pot guardar per cada tecla premuda per l'usuari (codi de la tecla, data, hora, aplicació actual...) i igualada a "SI", en el cas de que la vulguem mostrar o "NO" en cas contrari, com es mostra en la figura següent:

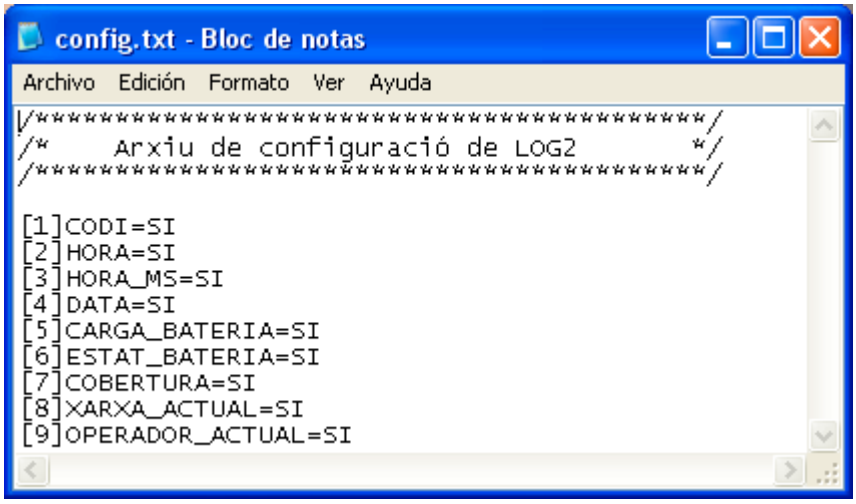


Figura 11: Arxiu de de configuració de LOG2

Per canviar aquest arxiu de configuració, es pot fer directament sobre l'arxiu de text amb qualsevol editor (des d'un equip de sobretaula, per exemple), canviat les igualtats a *SI* o *NO*. És important no modificar cap altre aspecte com pot ser l'ordre de les opcions, ja que podria donar lloc a errors.

Si l'arxiu *config.txt* no existeix a l'inici de l'aplicació, s'inicialitzen les variables globals de configuració per defecte, que són el codi de la tecla, l'hora i la data.

Aquest és un bon sistema de configuració per la seva senzillesa i la gran utilitat que aporta a l'aplicació, ja que permet triar les dades que interessen guardar en cada moment i no és necessari emmagatzemar per a cada event tota la informació recollida. En aquesta versió primera, es capturen poques dades i no s'aprecia tot el potencial d'aquesta opció, però la situació canviaria si tinguéssim molta informació per guardar. Per exemple, suposem que tenim al voltant de 100 característiques capturades per cada pulsació (codi, hora...etc), però que només

volem saber quantes tecles podem prémer amb una sola càrrega de bateria en una aplicació determinada; només caldria posar en l'arxiu de configuració BATERIA = SI i la resta a NO, per veure d'una manera ràpida el descens de càrrega de la bateria, i a més aconseguim un arxiu log molt més petit.

4.3.2 Implementacions realitzades

4.3.2.1 Primera implementació

La primera idea implementada consisteix en guardar com a variables globals booleanes, totes les opcions configurades en l'arxiu de text *config.txt*. Inicialment a *false*, aquestes variables es modifiquen quan s'inicia l'aplicació *LOG2* (amb el mètode *InitializeControlsL()* que crida a la funció d'assignació del valor de les variables (*assigVarConf*). Un cop establert el valor d'aquestes variables, podem decidir les dades que s'han d'escriure.

Un dels majors inconvenients d'aquesta opció, és l'enorme grandària que pot assolir la funció *assigVarConf* si disposem d'un gran nombre d'opcions i també el fet d'haver d'afegir massa codi cada cop que s'afegeix una opció nova:

4.3.2.2 Segona implementació

Per sol.lucionar el problema de la primera implementació, s'ha optat per guardar un array de tipus *char* amb longitud igual al nombre de configuracions (a cada configuració del *config.txt* trobem un número que facilita el recompte cada cop que s'afegeixen opcions).

D'aquesta manera, la funció *assigVarConf* només ha de plenar aquest array amb els caràcters *S* ó *N*, depenent de la configuració que s'hagi establert en el *config.txt*.

Exemple de l'array plè amb 14 opcions:

```
config = SNSNSSNSNSSSSN;
```

Amb aquest sistema, ja no necessitem guardar una variable global booleana per a cada opció nova que afegim, sinó que només haurem de guardar dues variables a les que augmentarem el seu valor coincidint amb el nombre d'opcions. I no farà falta implementar codi nou en la funció assigVarConf:

4.4 Captura d’events de teclat en segon plà

Seguint amb els requeriments plantejats a l'inici d'aquest document, ara ens plantejarem com capturar aquests events.

Amb "captura d'events de teclat" ens referim a la pulsació de les tecles d'un dispositiu efectuades per un usuari en un moment determinat. En cada pulsació capturarem el codi de la tecla i altres dades relacionades per a poder-les guardar en un arxiu de text.

4.4.1 Les classes *CCoeControl* i *CActive*

Per capturar els events de teclat, hem de tenir molt en compte la classe de la que hereta la nostra classe. Nosaltres ens centrarem en dues d'elles que son "filles" de la classe CBase, la CCoeControl i la CActive, tal i com es mostra en el següent esquema:

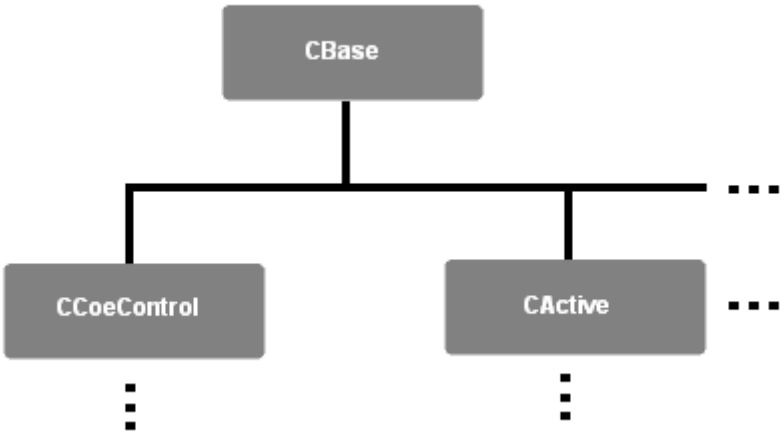


Figura 12: Herència de les classes *CCoeControl i CActive*

En la nostra aplicació, la classe principal haurà de ser "filla" d'alguna d'aquestes dues classes.

4.4.1.1 La classe *CCoeControl*

A primer cop d'ull, aquesta classe sembla totalment adient per a fer que la nostra classe hereti d'ella, ja que té el mètode OfferKeyEventL que s'encarrega de capturar i tractar els events de teclat, per tant, necessitem sobrecarregar-lo per poder capturar les dades que ens interessen. Consta de dos paràmetres, un del

tipus TKeyEvent que ens dona la informació que buscàvem sobre quina tecla s'ha pitjat (amb TKeyEvent.iScanCode obtindrem el codi de la tecla), i un altre del tipus TEventCode que ens indica si s'ha pitjat o s'ha soltat la tecla (EEventKeyDown o EEventKeyUp respectivament). Un cop hagem acabat amb el tractament de la tecla haurem de retornar EKeyWasConsumed ó EKeyWasNotConsumed en el cas de que no efectuéssim cap operació.

El funcionament és correcte quan l'aplicació s'executa en primer pla (Foreground) i permet guardar en un arxiu de text totes les tecles que es premen, però un cop enviem l'aplicació a segon pla (Background), es perd la captura de les tecles i aquesta passa a estar focalitzada en l'aplicació situada en primer pla.

Si busquem altres mètodes de la classe que ens donin un resultat similar a OfferKeyEventL, ens trobem amb HandleForegroundEventL que es llança cada vegada que el Window Group de l'aplicació reb o perd el focus. Consta d'un paràmetre booleà per indicar si l'aplicació reb (ETrue) o perd (EFalse) el focus.

També trobem HandleKeyEventL, que consta del paràmetre booleà aBackground, per indicar si l'aplicació es troba en segon pla. I també HandleWsEventL, que consta dels paràmetres de tipus TWsEvent i CCoeControl.

Malhauradament, amb aquests mètodes obtenim el mateix resultat que amb el primer, quan enviem l'aplicació a segon pla es perd completament la recepció d'event de teclat.

Si ens ho parem a pensar, això és un funcionament totalment lògic: normalment, només pot haver-hi un programa que capturi els events d'usuari al mateix temps, de la mateixa manera que això passa en qualsevol sistema operatiu. Per exemple, si suposem que un usuari d'un ordenador personal està escrivint en un processador de texts i canvia a un altre programa sense tancar-lo, el processador de texts es queda en segon pla i deixa de capturar les tecles, si no fos així, no es podrien compaginar diverses aplicacions d'usuari obertes al mateix temps.

4.4.1.2 La classe *CActive*

Per solucionar el problema anteriorment esmentat i permetre a la nostra aplicació capturar els events en segon pla, hem d'optar per la classe *CActive*.

Així doncs, canviem l'herència de la nostra classe principal (CLOG2Container) que inicialment heretava de classe CCoeControl i que ara l'hem fet heretar de la classe CActive, que segons la definició és la classe principal de l'abstracció d'objecte actiu. Això encapsula tan la emissió d'una petició a un proveïdor de servei asíncron como la manipulació de requests. Una aplicació pot tenir un o varis objectes actius, el tractament del qual es controlat per un planificador actiu.

Un cop heretem de la classe, ja podem sobrecarregar el mètode RunL que és el nou mètode encarregat dels events, en comptes de sobrecarregar el mètode OfferKeyEventL de la classe CCoeControl, tal i com haviem fet fins ara.

Aquesta elecció, ens obliga a capturar per separat, totes les tecles en el mètode contructor (ConstructL). Les tecles que no s'especifiquin aquí, seran ignorades i no podran ser guardades en un arxiu de text. S'ha de tenir en compte que cada dispositiu Symbian pot tenir una configuració de tecles diferent, per exemple, alguns models disposen de teclat qwerty i d'altres no. Per tant, la millor opció és incloure totes les tecles conegudes (TUint aKeyCode) a falta d'un codi genèric per a qualsevol tecla:

```
iHandle = iWg.CaptureKey(TUint aKeyCode,TUint aModifierMask,
TUint aModifier);
```

També s'han de crear dues funcions noves que anomenarem Listen() i DoCancel:

```
void CLOG2Container::Listen()
{
        iWsSession.EventReady(&iStatus);
        SetActive();
}

void CLOG2Container::DoCancel()
{
        iWsSession.EventReadyCancel();
}
```

El mètode RunL s'executarà cada cop que l'usuari premi una de les tecles especificades en el mètode ConstructL. El mètode RunL tindrà el següent format:

```
void CLOG2Container::RunL()
{
	TWsEvent e;
	iWsSession.GetEvent(e);

	//Processament de les tecles
	//...

	//Retornem el funcionament normal de la tecla
	TInt wgId = iWsSession.GetFocusWindowGroup();
	iWsSession.SendEventToWindowGroup(wgId, e);

	if (iStatus != KErrCancel)
	{
		Listen();
	}
}
```

Com es pot apreciar, amb iWsSession.GetEvent(e) obtenim totes les dades necessàries de la tecla capturada (iScanCode...etc) i en les últimes línies de codi, permetem retornar el funcionament normal dels events de teclat i preparem la següent captura amb la crida al mètode Listen() que prèviament hem implementat.

Finalment, hem aconseguit capturar els events de teclat sense interferir en l'activitat del usuari, fent la que nostra aplicació s'executi de manera invisible.

4.5 Enviament de l'aplicació a segon pla (Background)

Normalment, una aplicació s'executa en primer pla per a que l'usuari pugi interactuar amb aquesta, però hi ha aplicacions que es poden enviar a segon pla o *Background* per a que s'executin de forma invisible per a l'usuari i això li permeti utilitzar altres aplicacions alhora. Aquest és un requeriment fonamental per al nostre programa, ja que ens interessa capturar les tecles de l'usuari de forma totalment invisible per a l'usuari. Per a la realització d'aquesta funcionalitat disposem de diverses opcions d'implementació, que es detallen a continuació:

4.5.1 Opcions d'implementació

4.5.1.1 Opció 1, *WindowGroup*

Una opció d'implementació consisteix en crear un WindowGroup en el mètode constructor ConstructL:

```
CApaWindowGroupName*
wn=CApaWindowGroupName::NewLC(iWsSession);
wn->SetHidden(ETrue);
wn->SetWindowGroupName(iWg);
CleanupStack::PopAndDestroy();
```

Gràcies al WindowGroup creat, que anomenem wn, podem fer invisible l'aplicació gràcies al mètode següent:

```
wn->SetHidden(ETrue);
```

4.5.1.2 Opció 2, TApaTaskList i TApaTask

En la nostra aplicació, finalment s'ha optat per utilitzar una altra tècnica , enviar l'aplicació a segon pla a través de les classes TApaTaskList i TApaTask que permeten, entre altres coses, buscar i manipular aplicacions o tasks de manera fàcil i potent. Per enviar la nostra aplicació a segon pla, primer la busquem pel nom a través el mètode FindApp de la classe TApaTaskList i després utilitzem el mètode SentToBackground de la classe TApaTask:

```
TApaTaskList taskList(iWsSession);
TApaTask task = taskList.FindApp(_L("LOG2"));
```

```
task.SendToBackground();
```

Un cop executades aquestes línies de codi al principi de l'execució de l'aplicació (en el mètode ConstructL), ja tindrem el nostre programa en segon pla i podrem tornar-lo a portar a primer pla en qualsevol moment amb:

```
task.SendToBackground();
```

Una altres funció que també ens pot resultar d'utilitat en la nostra aplicació, és la proveïda pel mètode KillTask que ens permet tancar qualsevol aplicació immediatament. Això ens anirà bé quan vulguem aturar el nostre programa:

```
task.KillTask();
```

4.6 Captura de dades

Per cada event de teclat, podem capturar diversa informació del dispositiu que serà emmagatzemada en l'arxiu de text corresponent. La informació més bàsica i necessària, és un identificador de la tecla, la data i l'hora en que s'ha premut. Però podem recollir un gran volum de dades, tant del software com del hardware, depenent dels nostres interessos. En aquest projecte s'han recollit vuit informacions:

- Codi de la tecla
- La data actual
- L'hora actual
- Càrrega de la bateria
- Estat de la bateria
- Cobertura
- Xarxa
- Operador

Les captures s'han implementat dins de la funció RunL que s'executa cada cop que s'efectua un event.

4.6.1 Codi de la tecla

Per obtenir informació relativa a l'event, necessitem primer crear una variable de tipus TWsEvent i després fer anar el mètode GetEvent de iWsSession per plenar aquesta variable:

```
TWsEvent e;
iWsSession.GetEvent(e);
```

Un cop tenim la variable e amb l'event capturat, podem obtenir l'identificador de la tecla (ScanCode) de la següent manera:

```
e.Key()->iScanCode
```

Aquesta operació ens retornarà un enter, que podríem, si volguèssim, identificar amb un nom de tecla, com per exemple "Número 1".

4.6.2 La data actual

La data actual (la que està configurada al dispositiu) pot ser capturada fàcilment gràcies a la classe TTime.

Primer creem la variable time:

```
TTime time;
```

Per obtenir la data i l'hora actuals del sistema es pot fer servir el mètode HomeTime de la pròpia TTime.

```
time.HomeTime();
```

I ja podem accedir i guardar el dia, mes i any actuals:

```
time.DateTime().Day();
time.DateTime().Month();
time.DateTime().Year();
```

4.6.3 L'hora actual

De la mateixa manera que amb la data, podem obtenir l'hora, el minut, el segon i el microsegon en que s'ha efectuat el event:

```
time.DateTime().Hour()
time.DateTime().Minute()
time.DateTime().Second()
time.DateTime().MicroSecond()
```

4.6.4 Càrrega de la bateria

Per a obtenir aquesta informació, necessitem la classe CTelephony de la llibreria Etel3rdParty.h.

Creem una variable de tipo CTelephony::TBatteryInfoV1:

```
CTelephony::TBatteryInfoV1        iBatteryInfoV1;
```

I utilitzem el membre iChargeLevel per obtenir un enter que indica la càrrega actual de la bateria:

```
iBatteryInfoV1.iChargeLevel;
```

4.6.5 Estat de la bateria

La bateria es pot trobar en cinc estats diferents:

- *EPoweredByBattery*: La bateria està connectada i funcionant correctament.
- *EBatteryConnectedButExternallyPowered*: La bateria està connectada, però el dispositiu s'alimenta de corrent elèctrica a través del carregador.
- *ENoBatteryConnected*: La bateria no està connectada.
- *EPowerFault*: Hi ha un error en la bateria. *EPowerStatusUnknown*: Estat de la bateria desconegut.

Per mostrar en quin d'aquests possibles estats es troba la bateria, es pot fer de la mateixa manera que hem fet per obtenir la càrrega de la bateria (CTelephony::TBatteryInfoV1) i amb el membre iStatus que ens retornarà un d'aquest cinc estats esmentats.

4.6.6 Cobertura

Amb cobertura, ens referim a la força de la senyal disponible en un moment determinat.

Utilitzarem de nou, la classe CTelephony:

```
CTelephony::TSignalStrengthV1 iSignalStrengthV1;
```

I el membre iSignalStrength que ens retornarà un enter que especifica la força de la senyal:

```
iSignalStrengthV1.iSignalStrength;
```

4.6.7 Xarxa

Amb xarxa, ens referim al nom de la xarxa que s'està utilitzant.

Un altre cop, amb la mateixa classe:

```
CTelephony::TNetworkNameV1 iNetworkNameV1;
```

I amb el membre iNetworkName que ens retorna el nom:

```
iNetworkNameV1.iNetworkName;
```

4.6.8 Operador

Amb operador ens referim al nom de l'operador que s'utilitza (Movistar, Vodafone, Oragne...etc.).

Ho fem de manera similar a les anteriors, CTelephony i el membre NetworkName que ens retorna el nom:

```
CTelephony::TNetworkNameV1 iNetworkNameV1;

iNetworkNameV1.iNetworkName;
```

4.7 Captura de característiques del *hardware* del dispositiu

Aquest requeriment es basa en la idea d'obtenir les característiques del dispositiu cada cop que s'executi la nostra aplicació i guardar-les a l'inici de l'arxiu de text. Aquest procediment no s'executarà per cada tecla premuda com passava amb les altres dades obtingudes, només un sol cop.

Les dades del *hardware* que volem obtenir són les següents:

- Fabricant
- Versió del hardware
- Versió del software
- Model
- UID del dispositiu
- CPU
- Velocitat de la CPU
- Arquitectura de la CPU
- ABI de la CPU
- Memòria RAM
- Memòria RAM lliure
- Memòria ROM
- Tipo de teclat

Per a realitzar aquesta tasca, crearem una funció que anomenarem *getInfoDevice()* i que es cridarà des de el mètode *InitializeControlsL()* per assegurar que només s'executa un sol cop a l'inici del programa.

4.7.1 Les classes *CTelephony* i *HAL*

Per obtenir la majoria de característiques del hardware farem anar la classe *HAL* (*class HAL : public HALData*) a través del seu membre *Get()* que ens permet guardar en una variable un atribut del *hardware (static TInt Get(TAttribute anAttribute, TInt& aValue)*). Els paràmetres d'aquest membre són el atribut que volem obtenir (*TAttribute*) i una variable de tipus entera on guardar el resultat (*TInt&*).

Exemple:

```
HAL::Get(HAL::ECPUSpeed,cpuSpeed);
```

Per obtenir l'atribut IMEI o número de sèrie, que és un enter que identifica de manera única a cada dispositiu mòbil, necessitarem fer ús de la classe *CTelephony* (`class CTelephony : public CBase`) i el seu membre *iSerialNumber* de la manera següent:

```
CTelephony::TPhoneIdV1      iPhoneIdV1;
TBuf<50> IMEI  =            iPhoneIdV1.iSerialNumber;
```

4.8 Auto execució

Tal i com s'indica a l'anàlisi de requeriments, la possibilitat d'auto execució afegeix una característica útil al programa.Per defecte, una aplicació no s'inicia automàticament quan el dispositiu es reseteja. Algunes aplicacions poden requerir aquesta característica, per exemple en el nostre cas, si volem fer una estadística d'ús de les tecles durant una setmana, el fet de no haver d'iniciar l'aplicació cada cop que s'inicia el dispositiu ens estalvia feina i ens evita possibles oblits. Quan una aplicació està instal.lada. És important que l'usuari tingui en tot moment control sobre aquesta característica i la pugi activar o desactivar desde un menú d'usuari.

4.8.1 Procediment d'auto execució

Per a poder fer la nostra aplicació auto executable, hem d'aplicar una sèrie de modificacions la nostra aplicació:

1. Creem un arxiu amb extensió .rss amb el següent codi:

```
#include <startupitem.rh>

RESOURCE STARTUP_ITEM_INFO startexe
   {

   executable_name = "c:\\sys\\bin\\StartEXE.exe";
   recovery = EStartupItemExPolicyNone;

   }
```

2. Copiem aquest arxiu en la nostra carpeta *group* del projecte.

3. Obrim l'arxiu .MMP del nostre projecte i apliquem aquests canvis:

- Canviar el UID del nostre .EXE
- Per exemple: 0x06000001 i escrivim l'entrada següent a l'arxiu .MMP:

```
START RESOURCE 06000001.rss
END
```

5. Tornem a compilar per a que els canvis tinguin efecte.

6. Obrim l'arxiu .PKG i canviem el UID pel del arxiu .MMP (0x06000001).

7. Afegim la següent entrada a l'arxiu .PKG:

"C:\Symbian\9.1\S60_3rd_MR\Epoc32\data\06000001.rsc"-
"c:\private\101f875a\import\[06000001].rsc"

Després de realitzar aquests canvis, la nostra aplicació s'executarà sempre quan iniciem el nostre dispositiu. Aquest sistema funciona per a la 3a edició del S60, però probablement en dispositius anteriors s'hagi d'utilitzar un altre mètode. Els problemes amb que ens hem trobat, són els següents:

4.8.2 Possibles problemes:

- Si hi ha varies entrades per a diferents projectes en l'arxiu .PKG, llavors el UID utilitzat en la llista d'inici de l'arxiu RSS ha de coincidir amb la del paquet, per exemple, un paquet amb la capçalera {"LOG2"},(0x06000001), 1, 0, 1 ha de tenir el nom d'arxiu RSS [06000001].rss).

- Si l'EXE ja existeix quan s'està produint l'inici (*boot*), passats 5 segons desdel primer intent, es mostrarà per pantalla el missatge: "*Unable to start LOG2. Application may need to be removed.* " (Impossible iniciar LOG2. L'aplicació ha de ser borrada. ").

- Si la nostra aplicació no disposa d'una opció per la qual, l'usuari pugi activar/desactivar l'auto execució, ens trobarem amb un programa que no podem tancar mai sense cap ena externa, com un ordenador. Si a més, aquesta aplicació dona error i bloqueja el dispositiu, podem trobar una situació en la que sigui difícil recuperar el control del terminal.

- Aquest procés no funcionarà amb aplicacions auto signades, és a dir que necessitarem que la nostra aplicació sigui firmada amb un certificat de major grau (*Developer Certificate).* Aquest certificat ens l'haurà de donar la pròpia Symbian, després d'haver inspeccionat el nostre programa i d'haver-se convençut de que no causa cap dany al sistema. Els criteris que NOKIA aplica per a programes amb auto-start, són els següents:

    - Per defecte, només es permeten auto-starts per a certs tipus d'aplicacions que realment ho necessitin, com per exemple, un antivirus o com en el nostre cas, una aplicació de testeig.

- L'usuari podrà configurar aquesta característica quan vulgui des de un menú de l'aplicació, del contrari aquesta no serà acceptada.

- La característica d'auto inici ha hagut de ser implementada correctament, és a dir, utilitzant l'API de S60 3a edició anomenada *Startup List Management.*

- El fet de extreure la targeta de memòria quan l'aplicació s'està auto executant, no ha de causar cap excepció inesperada ni perillosa (bloqueig del dispositiu, reset, etc...).

4.9 Seguiment de la interfície d'usuari

Un cop tenim el funcionament bàsic i més important de l'aplicació (captura i enregistrament d'events en segon pla) ens podem centrar en altres funcionalitats addicionals com la captura d'altres dades que poden ser d'utilitat. Una de les més importants, és la funcionalitat de capturar l'aplicació en la que es troba l'usuari, per fer un seguiment més precís de l'activitat que està duent a terme. Degut a la falta de suport a nivell de l'API de Symbian, aquest requeriment no s'ha pogut implementar en aquest projecte, però s'han plantejat quatre estratègies o solucions que poden servir de base per a una futura implementació.

4.9.1 Estratègies

4.9.1.1 Mapejar totes les combinacions de tecles

Aquest mètode, només podria ser vàlid en el cas de que a l'inici del menú Symbian, el seleccionador comencés sempre a la mateixa posició, fet que es dóna, però també s'hauria de complir la premissa de que totes les icones de les aplicacions a executar, es trovessin sempre a la mateixa posició i així poder mapejar totes les combinacions possibles del cursor (camins). Per exemple, si entrem al menú i introduïm la següent combinació al cursor: esquerra - abaix - dreta, ens trobem seleccionant l'aplicació "Guia" que correspon a la guia de telèfons. Per desgràcia aquest mètode queda invalidat per la possibilitat que té l'usuari de canviar d'ordre les icones, que podria fer la predicció errònia.

4.9.1.2 Capturar imatges (*screenshots*)

Mitjançant aquesta tècnica podríem saber amb seguretat a quina aplicació es troba l'usuari en el menú i quina aplicació s'executa, independentment de l'ordre de les icones del menú.

Es tractaria de capturar un cop al principi, la pantalla actual i guardar-la en un arxiu amb format d'imatge (JPG seria probablement el format més indicat per la bona relació entre qualitat - tamany). D'aquesta manera podríem identificar les aplicacions que representen les icones del menú i mapejar-les en una matriu. Només s'hauria de tornar a capturar la pantalla quan es fes scroll en el menú per navegar entre les icones i puder capturar així, les noves icones que apareixen a la pantalla.

Com a característiques negatives, caldria destacar per una banda, la impossibilitat de reconèixer aplicacions noves que no haguéssim registrat prèviament en una llista, ja que les icones del menú haurien de ser comparades per obtenir la correspondència amb el seu nom. En el cas de que una icona no fos a la nostra

llista, s'hauria de guardar com a "no reconeguda" ó intentar obtenir llegir el nom d'aquesta directament de la icona, ja que totes les icones tenen un nom a sota, però això plantejaria un procediment difícil ja que s'hauria de reconèixer el text de les icones que està en format gràfic.

Un altre aspecte negatiu, encara que poc important, seria el referit al consum de recursos que ocasiona el tractament d'imatges (espai, capacitat de procés, memòria...) essent bastant més elevant que el tractament de qualsevol altre format, com per exemple, el text. Però aquest obstacle queda bastant superat per la gran potència i capacitats dels dispositius actuals, i també s'ha de tenir en compte, que les captures de pantalla completa només s'efectuarien al principi i en els moments de scroll.

Figura 13: Captura del menú Symbian

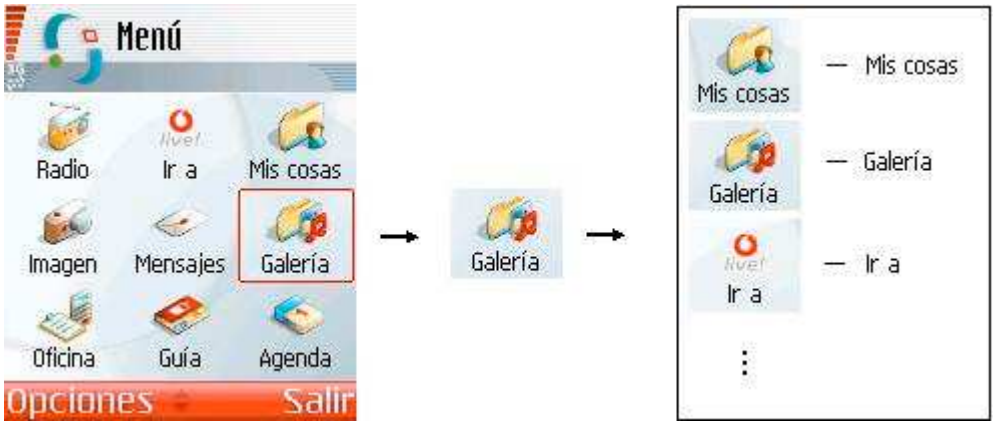


Figura 14: Identificació d'aplicació a partir de llista

Figura 15: Identificació d'aplicació sense llista

S'ha de tenir en compte que la interfície gràfica d'usuari pot canviar per modificacions efectuades pel propi usuari ó també per la instal.lació d'algunes aplicacions, que pot suposar un problema afegit.

4.9.1.3 Obtenir l'últim procés actiu

D'aquesta manera per cada registre guardat sobre la pulsació d'una tecla, podem guardar l'últim procés executat, que ens dona una pista sobre l'aplicació en la que es troba l'usuari, però s'ha de tenir en compte que una aplicació pot llançar diversos processos. Podem guardar el nom del procés i obtenir un nom del programa d'una llista de correspondències entre processos i noms d'aplicacions.

4.9.1.4 Utilitzar una classe pròpia del SDK

Sería coherent pensar que l'SDK permeti a través d'una classe, obtenir directament l'aplicació a la que l'usuari fa referència en el menú principal, ja que el sistema ha de portar en tot moment un control en aquest sentit i a més moltes aplicacions necessiten aquesta informació, per tant sembla raonable, la implementació per part dels responsables del SDK d'un membre d'alguna classe que serveixi aquesta informació.

4.9.1.4 Estratègia utilitzada

En un primer intent, s'ha optat per la estratègia "*4.9.1.4 Utilitzar una classe pròpia del SDK*" però després d'haver recorregut totes i cadascuna de les classes disponibles, no se n'ha trobat cap que respongui a les nostres necessitats, per tant, no podem implementar aquest requeriment. No obstant, s'ha programat una aproximació per l'estratègia de l'apartat "*4.9.1.3 Obtenir l'últim procés actiu*",

consistent en la utilització de les classes *TFindProcess* i *RProcess* de les llibreries del SDK de Symbian dins de la funció creada *SaveAppName* (que es crida cada cop que es prem una tecla), per a trobar l'últim procés actiu i obrir-lo, respectivament. Un bucle *while* recorre tots els processos actuals fins a arribar a l'últim i d'aquest se n'obté el nom i es guarda al fitxer de text de log.

5 Proves

5.1 Aspectes a tenir en compte en segon pla

Quan una aplicació perd el primer pla (foreground), s'ha d'esforçar per establir un estat estable, recuperable, és a dir, la suspensió de tot el processament i l'emmagatzematge de les dades per dos motius: Primer, la capacitat de processament s'ha de reservar per a l'aplicació de primer pla, ja que és l'aplicació amb la que l'usuari interactua. Si hi ha aplicacions en segon pla que continuen processant dades, l'usuari pot experimentar una reducció del rendiment (lentitud). Segon, les aplicacions en segon pla, es tancaran automàticament per sistema en cas de condicions baixes de memòria. Si es dona un apagat forçat , i les aplicacions en segon pla continuen processant dades, es pot donar el cas que sigui impossible restablir un estat estable i recuperable.

5.2 Proves efectuades

S'han efectuat una sèrie de proves en l'emulador amb totes les funcionalitats de l'aplicació per detectar possibles errors i corregir-los. Els resultats d'aquestes proves s'especifiquen en la taula següent:

Funcionalitat	Resultat de la prova
Execució	Correcte
Tancament	Correcte
Estabilitat	Correcte
Captura IMEI	Correcte
Captura Fabricant	Correcte
Captura Versió Hardware	Correcte
Captura Versió Software	Correcte
Captura Model	Correcte
Captura UID	Correcte
Captura CPU	Correcte
Captura Velocitat CPU	Correcte

Captura Arquitectura CPU	Correcte
Captura ABI de la CPU	Correcte
Captura Memòria RAM	Correcte
Captura Memòria RAM lliure	Correcte
Captura Memòria ROM	Correcte
Captura Tipo de teclat	Correcte
Captura Nombre de tecles	Correcte
Captura Nombre de tecles d'aplicació	Correcte
Captura Nombre de pixels eix X	Correcte
Captura Nombre de pixels eix Y	Correcte
Captura Nombre de colors	Correcte
Captura Nombre de LEDS	Correcte
Captura Mode pantalla	Correcte
Captura Codi tecla	Correcte
Captura Data	Correcte
Captura Hora	Correcte
Captura Carga bateria	Correcte
Captura Estat Bateria	Correcte
Captura Cobertura	Correcte
Captura Xarxa	Correcte
Captura Operador	Correcte

Taula 2: Proves

Aquest resultat final però, s'ha aconseguit després de diverses correccions efectuades al llarg de tot el procès d'implementació. Els apartats més conflictius han set el funcionament en segon plà i la captura de l'últim procès actiu.

També s'ha verificat la permanència dels arxius de text creats per la nostra aplicació en la memòria del emulador, inclús després d'apagar-lo i tornar-lo a encendre.

6 Conclusions generals y línies futures

Després de treballar una temporada amb el SDK de Symbian C++, puc afirmar que es tracta d'una eina molt completa i potent, que ens permetrà fer gairebé qualsevol cosa que el certificat de seguretat ens permeti. Algunes funcions a les que tenim accés a través del llenguatge natiu de Symbian, el C++, no les podríem utilitzar amb altres llenguatges com per exemple, Java (J2ME) i a més, la velocitat d'execució també és superior en C++. Altres avantatges:

- API per la càmera
- Bluetooth
- Grafics (3D)
- So (el suportat per Java es de qualitat inferior)

Com a part negativa, el fet de que C++ actuï tan directament amb el dispositiu, ens obliga a conèixer molt bé el SDK inclús per a realitzar tasques senzilles com una simple conversió de formats, per exemple, d'enter a caràcter, o per mostrar per pantalla determinada informació. D'altra banda, si volem que la nostra aplicació es pugui executar en un altre dispositiu desproveït del sistema operatiu Symbian, haurem d'optar per un altre llenguatge com podria ser Java.

També hi ha un altre aspecte negatiu, relatiu al emulador de Symbian. Tot i que l'emulador funciona perfectament, el fet de programar una aplicació requereix, al menys en el meu cas, de continues proves de funcionament, és a dir, que cada cop que es modifica el codi, aquest s'ha de provar a l'emulador, i aquesta feina porta molt temps d'espera. No és un problema del sistema en sí, sinó més aviat una característica de la programació d'aplicacions que utilitzen un sistema operatiu diferent al propi on es desenvolupa el programa i que requereix un emulador.

L'entorn Carbide, ha demostrat una estabilitat i qualitat generals molt bones (heretades de la plataforma Eclipse, en la que es basa), plantant cara Visual Studio de Microsoft. La versió Express (gratuïta) permet accés a la majoria del SDK, un fet molt important per a treballs acadèmics o independents .

El Debugger també s'emporta una bona nota, ja que m'ha facilitat en moltes ocasions la localització d'errors en el codi.

Com a part negativa, senyalaria la part gràfica o GUI (Graphic User Interface) que és millorable pel meu gust i la configuració dels SDK, que resulta una mica confusa i difícil de trobar al principi. També el fet de que el nom del nostre workspace no ha de tenir cap espai, si no volem tenir problemes en el futur.

Com a conclusió, un IDE totalment satisfactori i que aporta molt a la comunitat Open Source.

Si parlem del tema de la signatura d'aplicacions de Sybmian, hem de dir que els alts requeriments en matèria de seguretat dels dispositius mòbils, obliguen als desenvolupadors de sistemes operatius, a aplicar fortes capes de seguretat per a evitar possibles mals usos amb la creació de software malintencionat com per exemple virus i la signatura d'aplicacions de Symbian, assegura una alta seguretat en aquest aspecte, però també dificulta extraordinàriament el lliure desenvolupament , ja que qualsevol programa que vulguem provar en un dispositiu amb Symbian, avans haurà de ser evaluat pel sistema de filtres de Nokia i totes aquelles aplicacions que per les seves característiques, no passin els filtres, no podràn ser instal.lats en cap dispositiu físic. En aquest aspecte, he trobat una dificultat afegida al projecte, ja que la meva aplicació ha estat classificada pels filtres de Nokia com a virus potencial, dificultant l'obtenció de la signatura.

En definitiva, penso que s'han completat satisfactòriament la major part dels requeriments que s'havien definit a l'inici del projecte i que s'ha aconseguit una aplicació estable i el que és més important, és una base per a possibles projectes futurs.

Personalment he adquirit uns bons coneixements sobre Symbian i el desenvolupament d'aplicacions per a dispositius mòbils en general.

L'aplicació en sí, apart de l'aprenentatge, ja té varies utilitats pròpies. Per exemple, es pot utilitzar com a eina a l'hora de realitzar una aplicació en Symbian, per a obtenir el moment i la tecla concreta que fa que l'aplicació provoqui un error o excepció (debugeig). Només s'ha de consultar l'arxiu de text i llegir l'últim registre.

També pot servir per a l'elaboració d'estadístiques d'ús de les tecles d'una aplicació per part dels usuaris; així podem saber quines tecles s'utilitzen més i quines gairebé mai i podem dissenyar la propera versió de l'aplicació de manera més eficient.

A partir d'aquesta base, es poden crear moltes altres funcionalitats noves per a futurs projectes IPO i com a eina d'aprenentatge del món Symbian i dels dispositius mòbils en general.

En el moment d'escriure aquest projecte, la tecnologia tàctil s'està començant a imposar amb força en aquesta mena de dispositius , gràcies a l'aparició fa aproximadament un any, del terminal d'Apple, el iPhone. La majoria de marques ja disposen de dispositius totalment tàctils i tot fa pensar que el futur seguirà aquest camí. El més probable sigui que aquesta tecnologia funcioni d'una manera bastant diferent a la utilitzada pels dispositius amb teclat físic com els que s'han tractat en aquest projecte i això implica que l'aplicació de captura s'hauria d'adaptar a la nova generació de teclats virtuals si volem que funcioni en properes generacions.

Tenint en compte que l'objectiu d'aquest projecte és la seva utilització en el camp de la IPO (Interacció Persona Ordenador), hi ha un aspecte del seu funcionament

que s'ha de comentar: Totes les proves s'han efectuat en el propi emulador del SDK i no s'han realitzat en dispositius reals per temes de seguretat i certificats. La sol.lució passa per, o bé aconseguir un certificat efectuant un pagament, o bé realitzar una modificació no autoritzada del software del dispositiu (crack). El funcionament en l'emulador proporcionat pel fabricant és totalment correcte, el que indica que el software desenvolupat funcionaria perfectament en un entorn real si es disposés d'una certificació adequada.

7 Referències

7.1 Pàgines web:

- http://www.s60.com/life
- http://www.wikipedia.es
- http://www.forum.nokia.com/main/resources/tools_and_sdks/carbide/index.html
- http://msdn2.microsoft.com/en-us/library/ms997537.aspx
- http://www.paniccode.com/?p=93
- http://3lib.ukonline.co.uk/s60history.htm
- http://discussion.forum.nokia.com/forum
- http://www.newlc.com/
- http://www.symbian.com/developer
- http://goponygo.com/blog/
- http://developer.sonyericsson.com/site/global/techsupport/tipstrickscode/symbian/p_jogdial_symbian.jsp
- http://www.antonypranata.com/
- http://www.blocketpc.com/?s=zinc
- http://www.opda.net.cn/

7.2 Llibres i manuals:

- Morris, B. /*A Guide To Symbian Signed*/.
  London: Symbian Software Limited. 3rd edition (2008)

- Reiniö, T. /*Direct Screen Access in Series 60 Devices*/.
  London: NOKIA Forum 1st edition (2008)

- NOKIA, Corp. /*S60 Platform: Introductory Guide*/
  London: NOKIA Corporation 1st edition (2007)

- NOKIA, Corp. /*Symbian OS Basics (Course Pack)*/
  London: NOKIA Corporation 1st edition (2006)

- Digital Information Architects /*Programming for the Series 60*/
  England: John Wiley & Sons Ltd 1st edition (2003)

- Coulton, P. - Edwards,R. - Clemson, E. /*S60 Programming*/
  England: John Wiley & Sons Ltd 1st edition (2007)

7.3 Programes i exemples S60

- *YebSnaw* (capturador d'imatges)
  Codi font disponible: SÍ

- *TaskSpy* (administrador de tareas)
  Codi font disponible: SÍ

- *Screenshot* (capturador d'imatges)
  Codi font disponible: SÍ

- *Mobile Speak* (lector de pantalla per a mòbils, utilitzat per persones amb incapacitats)
  Codi font disponible: NO

ANNEX A Manual de l'aplicació *LOG2*

Instal.lació

Per a realitzar la instal.lació de l'aplicació des de un PC a un terminal mòbil, aquest a de ser compatible amb el sistema operatiu Symbian corresponent a la versió del SDK S60 3rd Edition. Tots els mòbils NOKIA recents suporten aquesta versió de Symbian. A continuació s'exposa una simulació aproximada de la instal.lació i funcionament del programa.

El primer pas és situar-nos en la carpeta on tinguem l'arxiu LOG2.sis en el nostre PC i executar-lo. Ens apareixerà un diàleg de confirmació semblant a la figura següent:

Figura 16: Confirmació de la instal.lació des de PC

Premem "Sí" i ens apareixerà un altre diàleg d'informació, avisant-nos de que continuem la instal.lació des de el dispositiu mòbil. Cliquem sombre el botó "Aceptar" i seguim a través del nostre dispositiu:

Figura 17: Diàleg informatiu

Si tot ha anat bé, s'haurà obert l'instal.lador d'aplicacions anomenat "Instalador" i s'haurà mostrat el missatge següent:

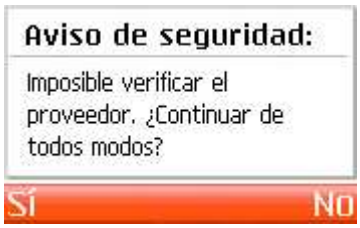


Figura 18: Confirmació de la instal.lació des de mòbil

Triem novament la opció "Sí" i un altre missatge de confirmació es mostrarà per pantalla:

Figura 19: Segona confirmació des de mòbil

Optem un altre cop per "Sí" i llavors se'ns mostra per pantalla tres opcions que podem triar: "Instalar", "Ver certificado" y "Ver detalles", com es mostra a la figura següent:

Figura 20: Opcions prèvies a la instal.lació

Seleccionem la primera opció: "Instalar" i tot seguit se'ns preguntarà on volem guardar l'aplicació, si ho volem fer a la memòria del telèfon o a la "Memory card" o memòria externa. Seleccionarem l'opció que més ens convingui:

Figura 21: Opcions d'ubicació de l'aplicació

Per últim, es mostra la barra de progrés de la instal.lació que finalitzarà quan arribi al 100 per cent:

Figura 22: Barra de progrés de la instal.lació

Execució de l'aplicació

Per executar l'aplicació, un cop instal.lada, s'ha de seguir el següent procediment. Des de el menú principal de Symbian, cliquem sobre la icona "Mis cosas":

Figura 22: Menú principal de Symbian

Tot seguit, busquem la icona del programa LOG2 i cliquem sobre ella per executar l'aplicació.

Figura 23: Icona de l'aplicació *LOG2* en *Mis cosas*

Sabrem que tot ha sortir bé si es mostra el missatge de text següent: "LOG2 iniciada...". A partir d'aquí, les accions de l'usuari s'escriuran en un arxiu de text. Per tancar l'aplicació haurem de premer el botó d'opcions i tot seguit el botó de borrar "C".

Obtenció de l'arxiu de text de registre

Per obrir des de un PC, l'arxiu de text que s'ha generat amb l'aplicació LOG2, podem utilitzar el buscador del sistema operatiu que fem anar i fer una cerca en el terminal per " *_*_*.txt ", la qual ens trobarà tots els arxius *log* guardats a la memòria del terminal mòbil i ja podrem obrir l'arxiu amb qualsevol editor de text.